Numéro d'ordre : 42

Université de Lille
Inria

École doctorale **École Graduée MADIS-631**
Unité de recherche **Centre de Recherche en Informatique, Signal et Automatique de Lille**

Thèse présentée par **Hector Kohler**

Soutenue le **1er décembre 2025**

En vue de l'obtention du grade de docteur de l'Université de Lille et de l'Inria

Discipline **Informatique**
Spécialité **Informatique et Applications**

Interpretabilité via l'Apprentissage Supervisé ou par Renforcement d'Arbres de Décisions

Thèse dirigée par	Philippe Preux	directeur
	Riad Akrour	co-directeur

Composition du jury

Rapporteurs	René Descartes	professeur à l'IHP	
	Denis Diderot	directeur de recherche au CNRS	
Examinateurs	Victor Hugo	professeur à l'ENS Lyon	président du jury
	Sophie Germain	MCF à l'Université de Paris 13	
	Joseph Fourier	chargé de recherche à l'INRIA	
	Paul Verlaine	chargé de recherche HDR au CNRS	
Invité	George Sand		
Directeurs de thèse	Philippe Preux	professeur à l'Université de Lille	
	Riad Akrour	Inria	

Colophon

Mémoire de thèse intitulé « Interpretabilité via l'Apprentissage Supervisé ou par Renforcement d'Arbres de Décisions », écrit par Hector Kohler, achevé le 13 juin 2025, composé au moyen du système de préparation de document LaTeX et de la classe yathesis dédiée aux thèses préparées en France.

Université de Lille
Inria

École doctorale **École Graduée MADIS-631**
Unité de recherche **Centre de Recherche en Informatique, Signal et Automatique de Lille**

Thèse présentée par **Hector Kohler**

Soutenue le **1er décembre 2025**

En vue de l'obtention du grade de docteur de l'Université de Lille et de l'Inria

Discipline **Informatique**

Spécialité **Informatique et Applications**

Interpretabilité via l'Apprentissage Supervisé ou par Renforcement d'Arbres de Décisions

Thèse dirigée par	Philippe Preux	directeur
	Riad Akrour	co-directeur

Composition du jury

Rapporteurs	René Descartes	professeur à l'IHP	
	Denis Diderot	directeur de recherche au CNRS	
Examinateurs	Victor Hugo	professeur à l'ENS Lyon	président du jury
	Sophie Germain	MCF à l'Université de Paris 13	
	Joseph Fourier	chargé de recherche à l'INRIA	
	Paul Verlaine	chargé de recherche HDR au CNRS	
Invité	George Sand		
Directeurs de thèse	Philippe Preux	professeur à l'Université de Lille	
	Riad Akrour	Inria	

Université de Lille
Inria

Doctoral School **École Graduée MADIS-631**

University Department **Centre de Recherche en Informatique, Signal et Automatique de Lille**

Thesis defended by **Hector Kohler**

Defended on **December 1, 2025**

In order to become Doctor from Université de Lille and from Inria

Academic Field **Computer Science**

Speciality **Computer Science and Applications**

Interpretability through Supervised or Reinforcement Learning of Decision Trees

Thesis supervised by	Philippe Preux	Supervisor
	Riad Akrour	Co-Supervisor

Committee members

Referees	René Descartes	Professor at IHP	
	Denis Diderot	Senior Researcher at CNRS	
Examiners	Victor Hugo	Professor at ENS Lyon	Committee President
	Sophie Germain	Associate Professor at Université de Paris 13	
	Joseph Fourier	Junior Researcher at INRIA	
	Paul Verlaine	HDR Junior Researcher at CNRS	
Guest	George Sand		
Supervisors	Philippe Preux	Professor at Université de Lille	
	Riad Akrour	Inria	

Interpretabilité via l'Apprentissage Supervisé ou par Renforcement d'Arbres de Décisions

Résumé

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Mots clés : apprentissage par renforcement, arbres de décision, interprétabilité, méthodologie

Centre de Recherche en Informatique, Signal et Automatique de Lille
Université de Lille-Campus scientifique – Bâtiment ESPRIT – Avenue Henri Poincaré – 59655 Villeneuve d'Ascq

Interpretability through Supervised or Reinforcement Learning of Decision Trees

Abstract

In this Ph.D. thesis, we study algorithms to learn decision trees for classification and sequential decision making. Decision trees are interpretable because humans can read through the decision tree computations from the root to the leaves. This makes decision trees the go-to model when human verification is required like in medicine applications. However, decision trees are non-differentiable making them hard to optimize unlike neural networks that can be trained efficiently with gradient descent. Existing interpretable reinforcement learning approaches usually learn soft trees (non-interpretable as is) or are ad-hoc (train a neural network then fit a tree to it) potentially missing better solutions.

In the first part of this manuscript, we aim to directly learn decision trees for a Markov decision process with reinforcement learning. In practice we show that this amounts to solving a partially observable Markov decision process. Most existing RL algorithms are not suited for POMDPs. This parallel between decision tree learning with RL and POMDPs solving help us understand why in practice it is often easier to obtain a non-interpretable expert policy–a neural network–and then distillate it into a tree rather than learning the decision tree from scratch.

The second contribution from this work arose from the observation that looking for a deicison tree classifier (or regressor) can be seen as sequentially adding nodes to a tree to maximize the accuracy of predictions. We thus formulate decision tree induction as sloving a Markov decision problem and propose a new state-of-the-art algorithm that can be trained with supervised example data and generalizes well to unseen data.

Work from the previous parts rely on the hypothesis that decision trees are indeed an interpretable model that humans can use in sensitive applications. But is it really the case? In the last part of this thesis, we attempt to answer some more general questions about interpretability: can we measure interpretability without humans? And are decision trees really more interpretable than neural networks?

Keywords: reinforcement learning, deicision trees, interpretability, methodology

Sommaire

Preliminary Concepts

Interpretable Sequential Decision Making

What is Sequential Decision Making ?

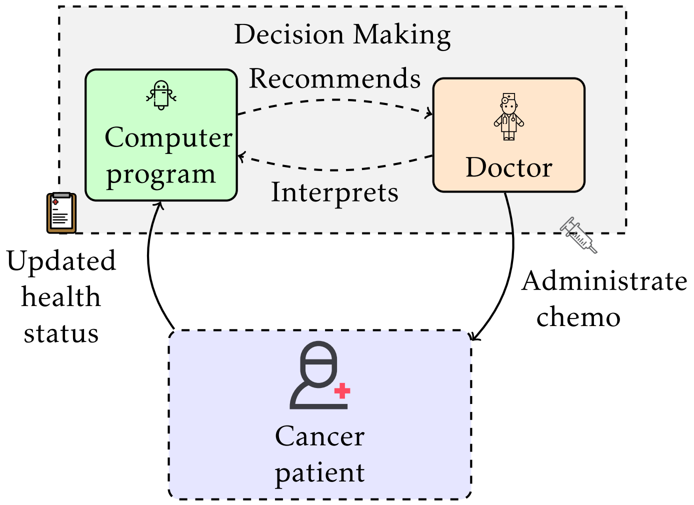


Figure 1 – Sequential decision making in cancer treatment. The AI system reacts to the patient's current state (tumor size, blood counts, etc.) and makes a recommendation to the doctor, who administers the chemotherapy to the patient. The patient's state is then updated, and this cycle repeats over time.

In this manuscript we study algorithms sequential decision making. Humans engage in sequential decision making in all aspects of life. In medicine, doctors have to decide how much chemotherapy to administrate based on the patient's current health in order to cure [1]. In agriculture, agronomists have to decide when to fertilize next based on the current soil and weather conditions in order to maximize plant growth [2]. In automotive, the auto-pilot system has to decide how to steer the wheel next based on lidar sensors in order to maintain a safe trajectory [4]. Those sequential decision making processes exhibits key similarities : an agent takes actions based on some current information to achieve some goal. As computer scientists, we ought to design computer programs [3] that can help humans during those sequential decision making processes. For example, as depicted in Figure 1, a doctor could benefit from a program that would recommend the "best" treatment given the patient's state. Machine learning algorithms [7] output such helpful programs. For non-sequential decision making, when the doctor only takes one decision and does not need to react to the updated patient's health, e.g. making a diagnosis about cancer type, a program can be fitted to example data : given lots of patient records and the associated diagnoses, the program learns to make the same diagnosis a doctor would given the same patient record, this is *supervised* learning [5]. In the cancer treatment example, the doctor follows its patient through time and adapts its treatment to the changing health of the patient. In that case, the program should learn to take decisions that lead to the patient recovery in the future based on how the patient's health changes from one chemo dose to another (this is *reinforcement* learning [6]). Every day, hundreds of new machine learning algorithms are published [1]. While most of those scientific articles focus on finding the "best" program possible, not many work make sure that their recommendations can be understood by humans. Next, we describe the notion of interpretability that is key to ensure safe deployment of computer programs trained with machine learning in critical sectors like medicine. However, as illustrated in Figure 2, complex machine learning systems often create black-box models that doctors cannot interpret, highlighting the need for interpretable approaches.

What is Interpretability ?

Interpretability is a crucial topic in modern science. While computer programs trained with machine learning algorithms have become more and more performing and made their way into our society, they are often black-box : their

[1] https://arxiv.org/list/cs.LG/pastweek?skip=0&show=2000

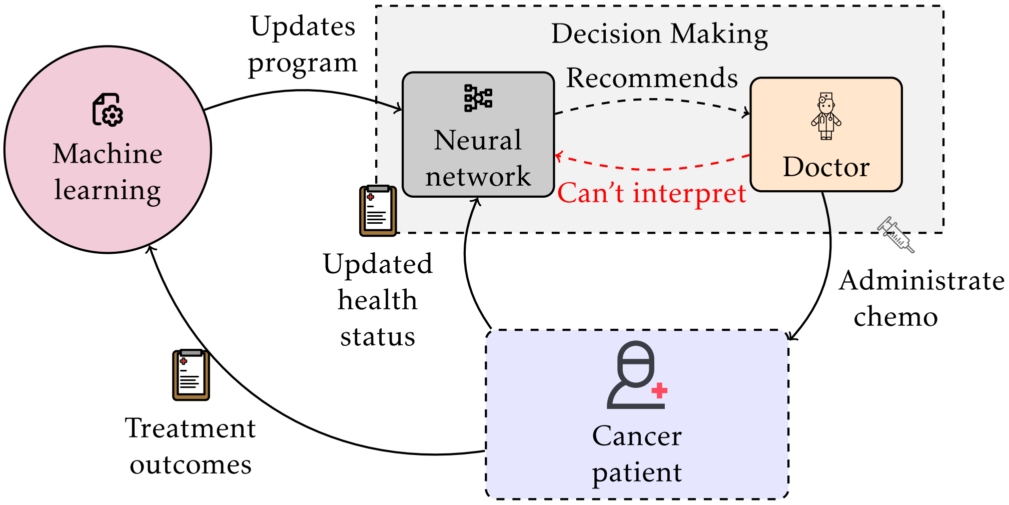
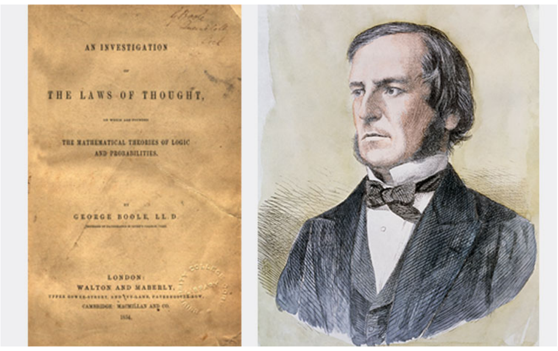


Figure 2 – Sequential decision making with machine learning in cancer treatment. The ML system learns from treatment outcomes to continuously improve the computer program's recommendations, creating a feedback loop for better patient care over time.


AN INVESTIGATION

OF

THE LAWS OF THOUGHT,

ON WHICH ARE FOUNDED

THE MATHEMATICAL THEORIES OF LOGIC
AND PROBABILITIES.

BY

GEORGE BOOLE, LL.D.

PROFESSOR OF MATHEMATICS IN QUEEN'S COLLEGE, CORK.

LONDON:
WALTON AND MABERLY,
UPPER GOWER-STREET, AND IVY-LANE, PATERNOSTER-ROW.
CAMBRIDGE: MACMILLAN AND CO.
1854.

Figure 3 – British logician and philosopher George Boole (1815-1864) next to its book *The Laws of Thoughts* (1854) that is the oldest known record of the word "interpretability".

outputs, e.g., the animals on the image (cite), tokamak control (cite), or even the abstract of your next article (cite), are computed with operations that are too complex for humans to fully understand. Indeed most of recent machine learning breakthroughs are obtained by training very large programs like neural networks (cite) which are black-box by definition since they perform sometimes millions of operations on their inputs. Originally, the etymology of "interpretability" is the Latin "interpretabilis" meaning "that can be understood and explained".

According to the Oxford English dictionary, the first recorded use of the english word "interpretability" dates back to 1854 when the british logician George Boole (Figure 3) described the addition of concepts :

> I would remark in the first place that the generality of a method in Logic must very much depend upon the generality of its elementary processes and laws. We have, for instance, in the previous sections of this work investigated, among other things, the laws of that logical process of addition which is symbolized by the sign +. Now those laws have been determined from the study of instances, in all of which it has been a necessary condition, that the classes or things added together in thought should be mutually exclusive. The expression x + y seems indeed uninterpretable, unless it be assumed that the things represented by x and the things represented by y are entirely separate ; that they embrace no individuals in common. And conditions analogous to this have been involved in those acts of conception from the study of which the laws of the other symbolical operations have been ascertained. The question then arises, whether it is necessary to restrict the application of these symbolical laws and processes by the same conditions of interpretability under which the knowledge of them was obtained. If such restriction is necessary, it is manifest that no such thing as a general method in Logic is possible. On the other hand, if such restriction is unnecessary, in what light are we to contemplate processes which appear to be uninterpretable in that sphere of thought which they are designed to aid ? [(**cite**)]

What is remarkable is that the supposedly first recorded occurrence of "interpretability" was in the context of (pre-)computer science. Boole asked : *when can we meaningfully apply formal mathematical operations beyond the specific conditions under which we understand them ?* In Boole's era, the concern was whether logical operations like addition could be applied outside their original interpretable contexts—where symbols and their sum represent concepts that humans can understand, e.g. red + apples = red apples. Today, we face an analogous dilemma with machine learning algorithms : neural networks learn complex unintelligible combinations of inputs (representations), but we often deploy them in contexts where operations should be understood by humans, e.g., in medicine.

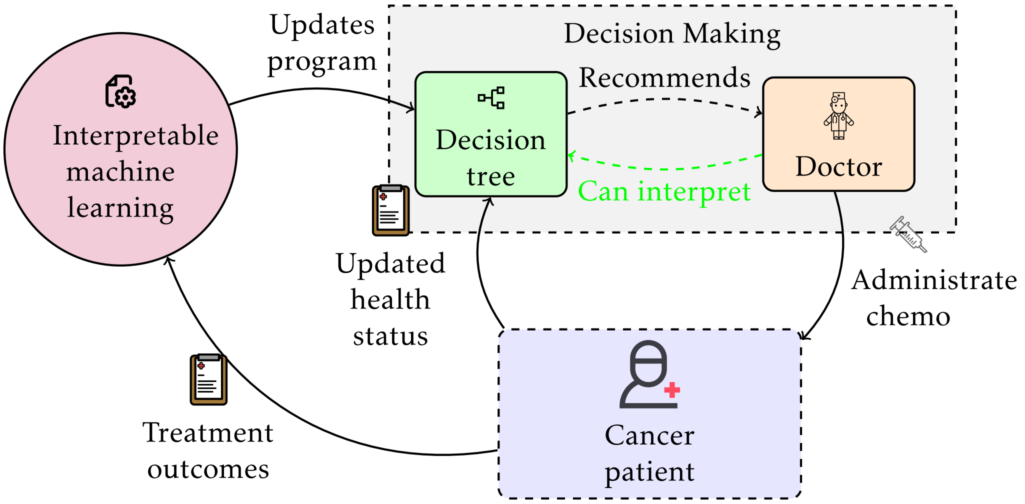


Figure 4 – Interpretable sequential decision making in cancer treatment. This shows a decision tree-based system where the doctor can understand and interpret the AI's recommendations, shown by the green "Can interpret" arrow, contrasting with the black-box neural network approach from Figure 2.

Circling back to our cancer treatment example, as illustrated in Figure 4, we would ideally want doctors to have access to computer programs that can recommend "good" treatments and which operations can be understood. Those two aspects of machine learning models–performance and interpretability–often compromise ; highly performing models like neural networks are often less interpretable and vice-versa (cite). However, we will observe later on that aiming for interpretability is not necessarily always constraining but can be a quite positive bias for performances in some domains like video games. Interestingly, one of the key challenges of doing research in interpretability is the lack of formalism ; there is no definition of what is an interpretable computer program. Throughout this manuscript we make the hypothesis that interpretability is the (space and time) complexity of a program (cite) and hence mostly focus on decision trees (low complexity) (cite) and neural networks (high complexity). Despite this lack of formalism the necessity of deploying interpretable models has sparked many works that we present next.

What are existing approaches for learning interpretable programs ?

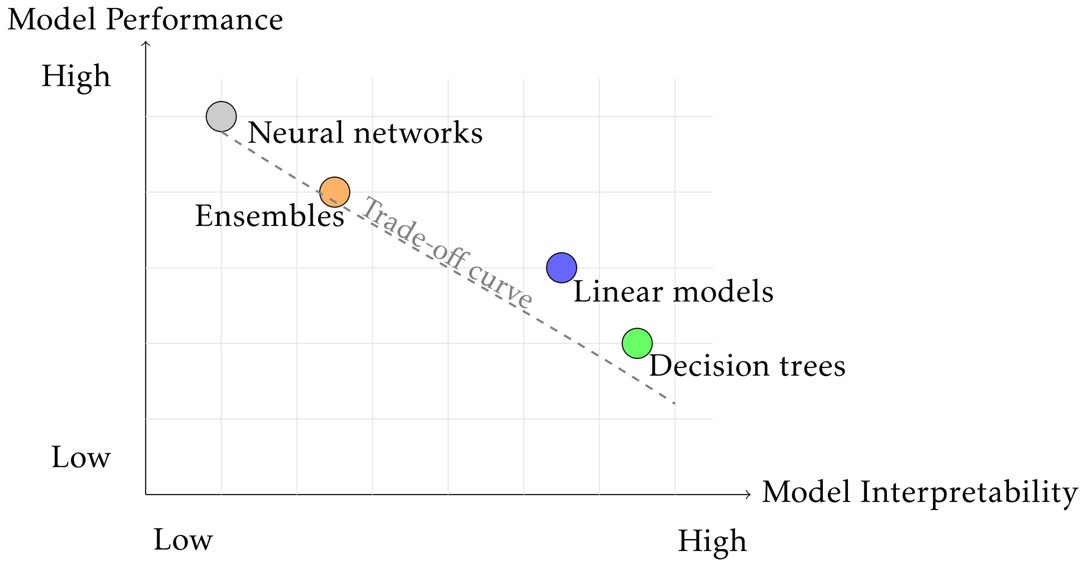


Figure 5 – The interpretability-performance trade-off in machine learning. Different model classes are positioned according to their typical interpretability and performance characteristics. The dashed line illustrates the general trade-off between these two properties.

Interpretable machine learning provides either local or global explanations (cite). Global methods output a whole model that is interpretable while local approaches output explanations of parts of the model such as how the model considers each part of a single specific input. In Figure 5 we present the popular trade-off between interpretability and performance of different model classes.

The most famous local explanation algorithm is LIME (Local Interpretable Model-agnostic Explanations) (cite). LIME works by perturbing the input around a specific instance and learning a simple interpretable model locally to explain that particular prediction. For each individual prediction, LIME provides explanations by identifying which features were most important for that specific decision. Global interpretable machine learning approaches are either direct or indirect. Direct algorithms like decision tree induction (cite) are algorithms that directly search a space of interpretable models. One of the key challenges that motivates this thesis is that decision tree induction is only defined for supervised learning but not for reinforcement learning. It means that to directly learn computer programs for interpretable sequential decision making, one has to design completely new algorithms. What most existing research have focused on so far, is to work around this limitation of decision tree induction to supervised learning and develop indirect methods. Indirect methods for interpretable sequential decision making–sometimes called post-hoc–start with the reinforcement learning of a non-interpretable computer program, e.g., deep reinforcement learning of a neural network, and then use supervised learning of an interpretable model with the objective to emulate the non-interpretable program. This approach is called behavior cloning or imitation learning and many, if not all, work on interpretable sequential decision making use this indirect approach (cite).

Researchers just recently started to study the advantage of direct over indirect learning of interpretable programs (cite). In short, the motivation behind developing direct methods is to have the interpretable program optimized to solve your actual goal, e.g. patient treatment, while indirect methods learn an interpretable program that is optimized to match the behaviour of a non-interpretable model that was itself optimized to solve your goal. There is no guarantee that optimizing this indirect objective yields the “best” interpretability-performance trade-offs. Hence, the ideal solution to interpretable sequential decision making would be to have global direct algorithms. Figure 6 illustrates the key difference between these two approaches.

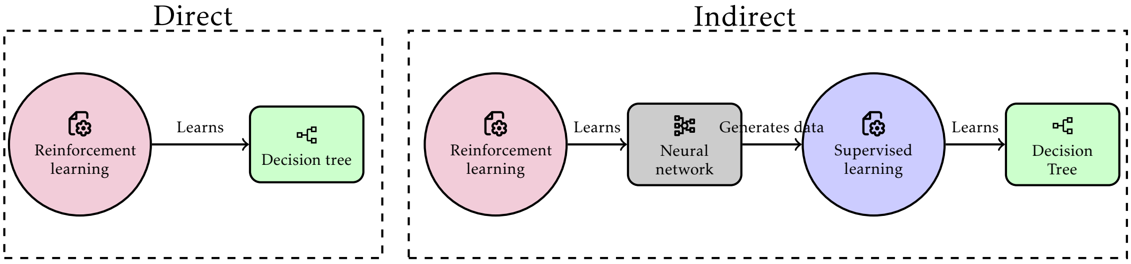


Figure 6 – Comparison of direct and indirect approaches for learning interpretable policies in sequential decision making

Outline of the Thesis

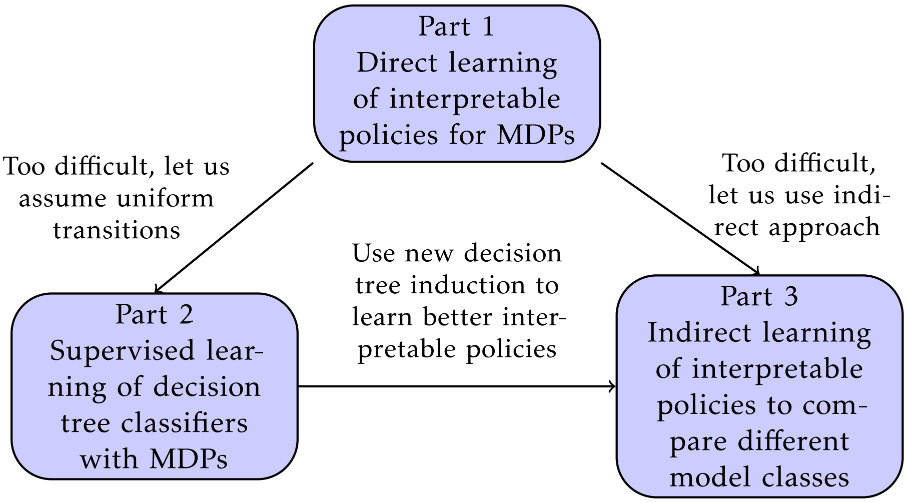

Part 1
Direct learning of interpretable policies for MDPs
Too difficult, let us assume uniform transitions
Too difficult, let us use indirect approach
Use new decision tree induction to learn better interpretable policies
Part 2
Supervised learning of decision tree classifiers with MDPs
Part 3
Indirect learning of interpretable policies to compare different model classes

Figure 7 – Thesis structure showing the progression from direct reinforcement learning of decision tree policies (Chapter 1) to simplified approaches : supervised learning with uniform transitions (Chapter 2) and indirect learning methods (Chapter 3).

In this thesis we take a stab at the difficult task of designing global direct algorithms for interpretable sequential decision making. In the first part of the manuscript we will present a mathematical formalism for the reinforcement learning of decision trees for sequential decision making. In particular we will show that indeed the direct approach can yield better trees than the indirect one. However the unfortunate key result from this opening part is that a good direct method cannot find decision trees for sequential decision making that are not *very* easy. Fortunately for us, in the second part of this manuscript, we show that some of these easy instances of interpretable sequential decision making tasks can be made *non*-sequential giving rise to a whole new decision tree induction algorithm for supervised learning. In particular, we thoroughly benchmark our new decision tree induction algorithm and claim the state-of-the-art for decision tree induction. Finally, after heavily studying decision trees and direct methods, we will leverage the diversity and simplicity of *indirect* methods to compare other model classes of programs and show that in some cases neural networks can be considered more interpretable than trees and there exist problems for which there is no need to trade-off performance for interpretability. We summarize the outline of the manuscript in Figure 7

Technical Preliminaries

What are decision trees ?

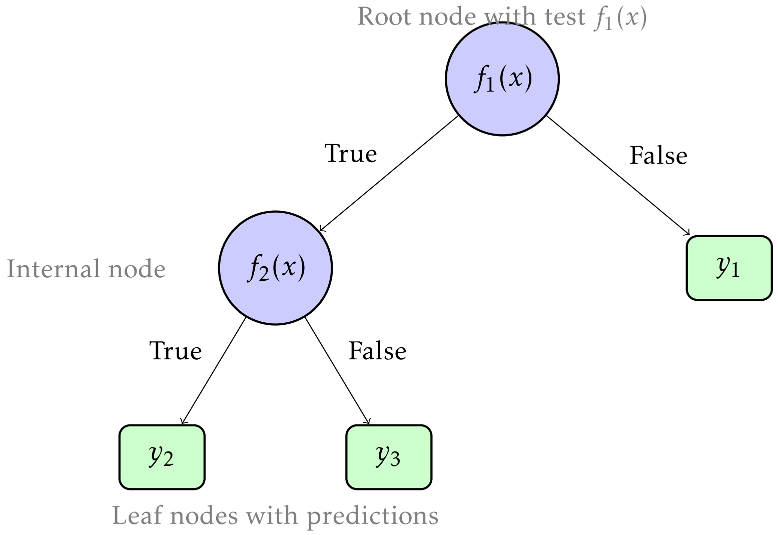


Figure 8 – A generic decision tree structure. Internal nodes contain test functions $f_v(x) : \mathcal{X} \to \{0,1\}$ that map input features to boolean values. Edges represent the outcomes of these tests (True/False), and leaf nodes contain predictions $y_\ell \in \mathcal{Y}$. For any input x, the tree defines a unique path from root to leaf.

Figure 9 – The american statistician Leo Breiman (1928-2005) author of *Classification and Regression Trees* (1984)

As mentioned earlier, as opposed to neural networks, decision trees are supposedly very interpretable because they only apply boolean operations on the program input without relying on internal complex representations.

Definition 1 (Decision tree)**.** *A decision tree is a rooted tree $T = (V, E)$ where :*

- *Each internal node $v \in V$ is associated with a test function $f_v : \mathcal{X} \to \{0, 1\}$ that maps input features $x \in \mathcal{X}$ to a boolean.*
- *Each edge $e \in E$ from an internal node corresponds to an outcome of the associated test function.*
- *Each leaf node $\ell \in V$ is associated with a prediction $y_\ell \in \mathcal{Y}$, where $\mathcal{Y}$ is the output space.*
- *For any input $x \in \mathcal{X}$, the tree defines a unique path from root to leaf, determining the prediction $T(x) = y_\ell$ where ℓ is the reached leaf.*

How to learn decision trees ?

The Classification and Regression Trees (CART) algorithm, developed by Leo Breiman and colleagues (Figure 9), is one of the most widely used methods for learning decision trees from supervised data. CART builds binary decision trees through a greedy, top-down approach that recursively partitions the feature space. At each internal node, the algorithm selects the feature and threshold that best splits the data according to a purity criterion such as the Gini impurity for classification or mean squared error for regression.

CART uses threshold-based test functions of the form $f_v(x) = \mathbb{I}[x[feature] \leq threshold]$ where $\mathbb{I}[\cdot]$ is the indicator function, consistent with the general decision tree definition above. The key idea is to find splits that maximize the homogeneity of the resulting subsets. For classification, this means finding test functions that separate different classes as cleanly as possible. The algorithm continues splitting until a stopping criterion is met, such as reaching a minimum number of samples per leaf or achieving sufficient purity. The complete CART procedure is detailed in Algorithm 1.

Markov decision processes/problems

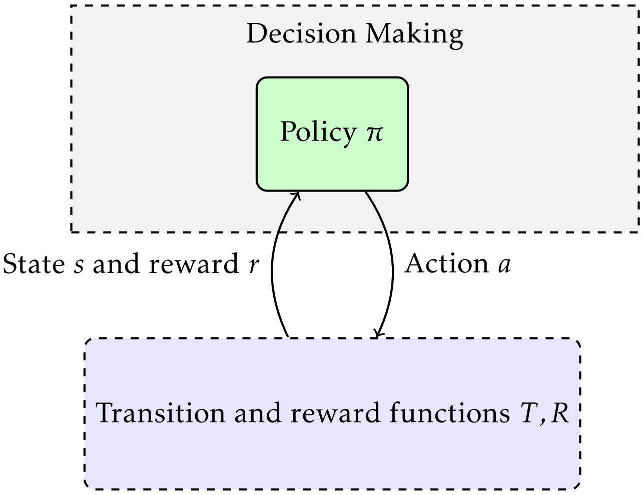


Figure 10 – Markov decision process

Markov decision processes (MDPs) were first introduced in the 1950s by Richard Bellman (cite). Informally, an MDP models how an agent acts over time to achieve its goal. At every timestep, the agent observes its current state, e.g. a patient weight and tumor size, and takes an action, e.g. injects a certain amount of chemotherapy. When doing a certain action in a certain state, the agent gets a reward that helps it evaluate the quality of its action with respect to its goal, e.g., the tumor size decrease when the agent has to cure cancer. Finally, the agent is

Algorithme 1 : CART Algorithm for Decision Tree Learning

```
Data : Training data $(X, y)$ where $X \in \mathbb{R}^{n \times d}$ and $y \in \{1, 2, \ldots, K\}^n$
Result : Decision tree $T$
Function BuildTree($X, y$) :
    if stopping criterion met then
        return leaf node with prediction MajorityClass($y$)
    end
    $(feature, threshold) \leftarrow$ BestSplit($X, y$)
    if no valid split found then
        return leaf node with prediction MajorityClass($y$)
    end
    Split data : $X_{left}, y_{left} = \{(x_i, y_i) : x_i[feature] \leq threshold\}$
                 $X_{right}, y_{right} = \{(x_i, y_i) : x_i[feature] > threshold\}$
    $left\_child \leftarrow$ BuildTree($X_{left}, y_{left}$)
    $right\_child \leftarrow$ BuildTree($X_{right}, y_{right}$)
    return internal node with test function
     $f_v(x) = \mathbb{I}[x[feature] \leq threshold]$ and children
     $(left\_child, right\_child)$
Function BestSplit($X, y$) :
    $best\_gain \leftarrow 0$
    $best\_feature \leftarrow None$
    $best\_threshold \leftarrow None$
    for each feature $f \in \{1, 2, \ldots, d\}$ do
        for each unique value $v$ in $X[:, f]$ do
            $y_{left} \leftarrow \{y_i : X[i, f] \leq v\}$
            $y_{right} \leftarrow \{y_i : X[i, f] > v\}$
            $gain \leftarrow \text{Gini}(y) - \frac{|y_{left}|}{|y|}\text{Gini}(y_{left}) - \frac{|y_{right}|}{|y|}\text{Gini}(y_{right})$
            if $gain > best\_gain$ then
                $best\_gain \leftarrow gain$
                $best\_feature \leftarrow f$
                $best\_threshold \leftarrow v$
            end
        end
    end
    return $(best\_feature, best\_threshold)$
Function Gini($y$) :
    return $1 - \sum_{k=1}^{K} \left(\frac{|\{i : y_i = k\}|}{|y|}\right)^2$ // Gini impurity
return BuildTree($X, y$)
```

provided with a new state, e.g. the updated patient state, and repeats this process over time. Following Martin L. Puterman's book on MDPs (cite), we formally define as follows.

Definition 2 (Markov decision process)**.** *An MDP is a tuple $\mathcal{M} = \langle S, A, R, T, T_0 \rangle$ where :*

— *S is a finite set of states $s \in \mathbb{R}^n$ representing all possible configurations of the environment.*

— *A is a finite set of actions $a \in \mathbb{Z}^m$ available to the agent.*

— *$R : S \times A \to \mathbb{R}$ is the reward function that assigns a real-valued reward to each state-action pair.*

— *$T : S \times A \to \Delta(S)$ is the transition function that maps state-action pairs to probability distributions over next states, where $\Delta(S)$ denotes the probability simplex over S.*

— *$T_0 \in \Delta(S)$ is the initial distribution over states.*

Now we can also model the "goal" of the agent. Informally, the goal of an agent is to behave such that it gets as much reward as it can over time. For example, in the cancer treatment case, the best reward the agent can get is to completely get rid of the patient's tumor after some time. Furthermore, we want our agent to prefer behaviour that gets rid of the patient's tumor as fast as possible. We can formally model the agent's goal as an optimization problem as follows.

Definition 3 (Markov decision problem)**.** *Given an MDP $\mathcal{M} = \langle S, A, R, T, T_0 \rangle$, the goal of an agent following policy $\pi : S \to A$ is to maximize the expected discounted sum of rewards :*

$$J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 \sim T_0, a_t = \pi(s_t), s_{t+1} \sim T(s_t, a_t)\right]$$

where $\gamma \in (0,1)$ is the discount factor that controls the trade-off between immediate and future rewards.

Hence, algorithms presented in this manuscript aim to find solutions to Markov decision problems, i.e. the optimal policy : $\pi^\star = \underset{\pi}{\text{argmax}} J(\pi)$ For the rest of this text, we will use an abuse of notation and denote both a Markov decision process and the associated Markov decision problem by MDP.

Exact solutions for Markov decision problems

It is possible to compute the exact optimal policy $\pi^\star$ using dynamic programming (cite). Indeed, one can leverage the Markov property to find for all states the best action to take based on the reward of upcoming states.

Definition 4 (Value of a state). *The value of a state $s \in S$ under policy π is the expected discounted sum of rewards starting from state s and following policy π :*

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s, a_t = \pi(s_t), s_{t+1} \sim T(s_t, a_t)\right]$$

Applying the Markov property gives a recursive definition of the value of s under policy π :

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T^{s'}(s, \pi(s)) V^\pi(s')$$

where $T^{s'}(s, \pi(s))$ is the probability of transitioning to state s' when taking action $\pi(s)$ in state s.

Definition 5 (Optimal value of a state). *The optimal value of a state $s \in S$, $V^\star(s)$, is the value of state s when following the optimal policy : $V^{\pi^\star}(s)$.*

$$V^\star(s) = V^{\pi^\star}(s) = \max_\pi [J(\pi)]$$

Definition 6 (Optimal value of a state-action pair). *The optimal value of a state-action pair $(s, a) \in S \times A$, $Q^\star(s, a)$, is the value of state s when taking action a and then following the optimal policy : $V^{\pi^\star}(s)$.*

$$Q^\star(s, a) = Q^{\pi^\star}(s) = R(s, a) + \gamma \sum_{s' \in S} V^\star(s')$$

Hence, the algorithms we study in the thesis can also be seen as solving the problem : $\pi^\star = \underset{\pi}{\operatorname{argmax}} \mathbb{E}[V^\pi(s_0) | s_0 \sim T_0]$. The well-known Value Iteration algorithm 2 solves this problem exactly (cite).

Algorithme 2 : Value Iteration

```
Data : MDP $\mathcal{M} = \langle S, A, R, T, T_0 \rangle$, convergence threshold $\theta$
Result : Optimal policy $\pi^*$
Initialize $V(s) = 0$ for all $s \in S$
repeat
    $\Delta \leftarrow 0$
    for each state $s \in S$ do
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_a [R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V(s')]$ // Bellman optimality update
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
    end
until $\Delta < \theta$;
for each state $s \in S$ do
    $\pi^*(s) \leftarrow \arg\max_a [R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V(s')]$ // Extract optimal policy
end
```

More realistically, neither the transition kernel T nor the reward function R of the MDP are known, e.g., the doctor can't **know** how the tumor and the patient health will change after a dose of chemotherapy, it can only **observe** the change. This distinction between the information available to the agent is paralleled with the distinction between dynamic programming and reinforcement learning (RL) that we describe next.

Reinforcement learning of approximate solutions to MDPs

Reinforcement learning algorithms popularized by Richard Sutton (Figure 11) (cite) don't **compute** an optimal policy but rather **learn** an approximate one based on sequences of observations $(s_t, a_t, r_t, s_{t+1})_t$. RL algorithms usually fall into two categories : value-based (cite) and policy gradient (cite). The first group of RL algorithms computes an approximation of $V^\star$ using temporal difference learning, while the second class leverages the policy gradient theorem to approximate $\pi^\star$. Examples of these approaches are shown in Algorithms 3 and 4.

(a) D. Bertsekas (b) M.L. Puterman (c) A. Barto (d) R. Sutton

Figure 11 – The godfathers of sequential decision making. Andrew Barto and Richard Sutton are the ACM Turing Prize 2024 laureate and share an advisor advisee relationship.

Algorithme 3 : Value-based RL (Q-Learning)

```
Data : MDP $\mathcal{M} = \langle S, A, R, T, T_0 \rangle$, learning rate $\alpha$, exploration rate $\epsilon$
Result : Policy $\pi$
Initialize $Q(s,a) = 0$ for all $s \in S, a \in A$
for each episode do
    Initialize state $s_0 \sim T_0$
    for each step $t$ do
        Choose action $a_t$ using $\epsilon$-greedy : $a_t = \arg\max_a Q(s_t, a)$ with prob.
          $1 - \epsilon$
        Take action $a_t$, observe $r_t = R(s_t, a_t)$ and $s_{t+1} \sim T(s_t, a_t)$
        $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)]$
        $s_t \leftarrow s_{t+1}$
    end
end
$\pi(s) = \arg\max_a Q(s,a)$ // Extract greedy policy
```

Algorithme 4 : Policy Gradient RL (REINFORCE)

```
Data : MDP $\mathcal{M} = \langle S, A, R, T, T_0 \rangle$, learning rate $\alpha$, policy parameters $\theta$
Result : Policy $\pi_\theta$
Initialize policy parameters $\theta$
for each episode do
    Generate trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$ following $\pi_\theta$
    for each timestep $t$ in trajectory do
        $G_t \leftarrow \sum_{k=t}^{T} \gamma^{k-t} r_k$ // Compute return
        $\theta \leftarrow \theta + \alpha G_t \nabla_\theta \log \pi_\theta(a_t|s_t)$ // Policy gradient update
    end
end
```

Both classes of algorithms are known to converge to the optimal value or policy under some conditions (cite) and have known great successes in real-world applications (cite). The books from Puterman, Bertsekas, Sutton and Barto, offer a great overview of MDPs and algorithm to solve them. There are many other ways to learn policies such as simple random search (cite) or model-based reinforcement learning. However, not many algorithms consider the learning of policies that can be easily understood by humans which we discuss next and that is the core of this manuscript.

Première partie

A difficult problem : Learning Decision Trees for MDP

I have not failed. I've just found 10.000 ways that won't work.

Thomas A. Edison

Chapitre 1

A Decision Tree Policy for an MDP is a Policy for some Partially Observable MDP

1.1 How to Learn a Decision Tree Policy for an MDP ?

1.1.1 Imitation

1.1.2 Soft Trees

1.1.3 Iterative Bounding MDPs

1.2 How to solve Iterative Bounding MDPs ?

1.2.1 Asymmetric Reinforcement Learning

1.2.2 Learning a decision tree policy is solving a POMDP

1.3 Is it hard to properly learn a Decision Tree Policy for an MDP ?

1.3.1 POMDPs are way harder to solve than MDPs

1.3.2 Memoryless approaches to solve POMDPs seem uneffective

Chapitre 2

An attempt at Learning Decision Tree Policies with Reinforcement Learning

2.1 Grid Worlds

$V(g) = \zeta \sum_{i=0}^{\infty} \gamma^{2i} + \sum_{i=0}^{\infty} \gamma^{2i+1}$
$V(0) = \zeta + \gamma^2 V(g)$
$V(1) = \zeta + \gamma^2 V(0)$
$\frac{1}{4}\gamma\frac{1}{1-\gamma} + \frac{1}{4}\frac{1}{1-\gamma} \leq \frac{1}{4}V(g) + \frac{2}{4}V(0) + \frac{1}{4}V(1)$
$\zeta \cdot \sum_{i=0}^{\infty} \gamma^i \leq \frac{1}{4}V(g) + \frac{2}{4}V(0) + \frac{1}{4}V(1)$
$\frac{1}{4}V(g) + \frac{1}{4}(\zeta + \gamma V(0)) + \frac{1}{4}(\zeta + \gamma V(1)) + \frac{1}{4}V(0) \leq \frac{1}{4}V(g) + \frac{2}{4}V(0) + \frac{1}{4}V(1)$
$\frac{1}{4}V(g) + \frac{1}{4}V(0) + \frac{1}{4}(\zeta + \gamma^2\zeta \sum_{i=0}^{\infty} \gamma^{2i}) + \frac{1}{4}(\zeta \sum_{i=0}^{\infty} \gamma^{2i}) \leq \frac{1}{4}V(g) + \frac{2}{4}V(0) + \frac{1}{4}V(1)$

2.1.1 Step-by-step derivation of the lower bound on ζ

Step 1 : Simplify the left side of the inequality

$$\frac{1}{4}\gamma\frac{1}{1-\gamma} + \frac{1}{4}\frac{1}{1-\gamma} = \frac{1}{4}\frac{1}{1-\gamma}(\gamma + 1) \tag{2.1}$$

$$= \frac{\gamma + 1}{4(1-\gamma)} \tag{2.2}$$

Step 2 : Express $V(g)$, $V(0)$, and $V(1)$ in simplified forms

$$V(g) = \zeta \sum_{i=0}^{\infty} \gamma^{2i} + \sum_{i=0}^{\infty} \gamma^{2i+1} \tag{2.3}$$

$$= \zeta \frac{1}{1-\gamma^2} + \gamma \frac{1}{1-\gamma^2} \tag{2.4}$$

$$= \frac{\zeta + \gamma}{1-\gamma^2} \tag{2.5}$$

$$V(0) = \zeta + \gamma^2 V(g) \tag{2.6}$$

$$= \zeta + \gamma^2 \frac{\zeta + \gamma}{1-\gamma^2} \tag{2.7}$$

$$= \frac{\zeta(1-\gamma^2) + \gamma^2(\zeta + \gamma)}{1-\gamma^2} \tag{2.8}$$

$$= \frac{\zeta + \gamma^3}{1-\gamma^2} \tag{2.9}$$

$$V(1) = \zeta + \gamma^2 V(0) \tag{2.10}$$

$$= \zeta + \gamma^2 \frac{\zeta + \gamma^3}{1-\gamma^2} \tag{2.11}$$

$$= \frac{\zeta(1-\gamma^2) + \gamma^2(\zeta + \gamma^3)}{1-\gamma^2} \tag{2.12}$$

$$= \frac{\zeta + \gamma^5}{1-\gamma^2} \tag{2.13}$$

Step 3 : Substitute into the right side of the inequality

$$\frac{1}{4}V(g) + \frac{2}{4}V(0) + \frac{1}{4}V(1) = \frac{1}{4}\frac{\zeta + \gamma}{1-\gamma^2} + \frac{1}{2}\frac{\zeta + \gamma^3}{1-\gamma^2} + \frac{1}{4}\frac{\zeta + \gamma^5}{1-\gamma^2} \tag{2.14}$$

$$= \frac{1}{4(1-\gamma^2)}[(\zeta + \gamma) + 2(\zeta + \gamma^3) + (\zeta + \gamma^5)] \tag{2.15}$$

$$= \frac{4\zeta + \gamma + 2\gamma^3 + \gamma^5}{4(1-\gamma^2)} \tag{2.16}$$

Step 4 : Set up the inequality

$$\frac{\gamma + 1}{4(1-\gamma)} \leq \frac{4\zeta + \gamma + 2\gamma^3 + \gamma^5}{4(1-\gamma^2)} \tag{2.17}$$

Step 5 : Use the identity $1 - \gamma^2 = (1-\gamma)(1+\gamma)$

$$\frac{\gamma + 1}{4(1-\gamma)} \leq \frac{4\zeta + \gamma + 2\gamma^3 + \gamma^5}{4(1-\gamma)(1+\gamma)} \tag{2.18}$$

Step 6 : Multiply both sides by $4(1-\gamma)$

$$\gamma + 1 \leq \frac{4\zeta + \gamma + 2\gamma^3 + \gamma^5}{1+\gamma} \tag{2.19}$$

Step 7 : Multiply both sides by $(1+\gamma)$

$$(\gamma + 1)(1+\gamma) \leq 4\zeta + \gamma + 2\gamma^3 + \gamma^5 \tag{2.20}$$

$$(\gamma + 1)^2 \leq 4\zeta + \gamma + 2\gamma^3 + \gamma^5 \tag{2.21}$$

Step 8 : Expand and rearrange

$$\gamma^2 + 2\gamma + 1 \leq 4\zeta + \gamma + 2\gamma^3 + \gamma^5 \tag{2.22}$$

$$4\zeta \geq \gamma^2 + 2\gamma + 1 - \gamma - 2\gamma^3 - \gamma^5 \tag{2.23}$$

$$4\zeta \geq \gamma^2 + \gamma + 1 - 2\gamma^3 - \gamma^5 \tag{2.24}$$

$$\zeta \geq \frac{\gamma^2 + \gamma + 1 - 2\gamma^3 - \gamma^5}{4} \tag{2.25}$$

Therefore, we obtain a **lower bound** on ζ :

$$\zeta \geq \frac{\gamma^2 + \gamma + 1 - 2\gamma^3 - \gamma^5}{4} \tag{2.26}$$

where $0 < \gamma < 1$.

2.1.2 Step-by-step derivation of the upper bound on ζ

Starting from the inequality :

$$\frac{1}{4}V(g) + \frac{1}{4}(\zeta + \gamma V(0)) + \frac{1}{4}(\zeta + \gamma V(1)) + \frac{1}{4}V(0) \leq \frac{1}{4}V(g) + \frac{2}{4}V(0) + \frac{1}{4}V(1)$$

Step 1 : Cancel the $\frac{1}{4}V(g)$ terms from both sides

$$\frac{1}{4}(\zeta + \gamma V(0)) + \frac{1}{4}(\zeta + \gamma V(1)) + \frac{1}{4}V(0) \leq \frac{2}{4}V(0) + \frac{1}{4}V(1) \tag{2.27}$$

Step 2 : Expand the left side

$$\frac{1}{4}\zeta + \frac{1}{4}\gamma V(0) + \frac{1}{4}\zeta + \frac{1}{4}\gamma V(1) + \frac{1}{4}V(0) \leq \frac{2}{4}V(0) + \frac{1}{4}V(1) \tag{2.28}$$

Step 3 : Combine like terms

$$\frac{1}{2}\zeta + \frac{1}{4}\gamma V(0) + \frac{1}{4}\gamma V(1) \leq \frac{2}{4}V(0) - \frac{1}{4}V(0) + \frac{1}{4}V(1) \tag{2.29}$$

$$\frac{1}{2}\zeta + \frac{1}{4}\gamma V(0) + \frac{1}{4}\gamma V(1) \leq \frac{1}{4}V(0) + \frac{1}{4}V(1) \tag{2.30}$$

Step 4 : Factor out common terms

$$\frac{1}{2}\zeta \leq \frac{1}{4}V(0) + \frac{1}{4}V(1) - \frac{1}{4}\gamma V(0) - \frac{1}{4}\gamma V(1) \tag{2.31}$$

$$\frac{1}{2}\zeta \leq \frac{1}{4}V(0)(1-\gamma) + \frac{1}{4}V(1)(1-\gamma) \tag{2.32}$$

$$\frac{1}{2}\zeta \leq \frac{1-\gamma}{4}(V(0) + V(1)) \tag{2.33}$$

$$\zeta \leq \frac{1-\gamma}{2}(V(0) + V(1)) \tag{2.34}$$

Step 5 : Substitute the expressions for $V(0)$ and $V(1)$

$$V(0)+V(1)=\frac{\zeta+\gamma^3}{1-\gamma^2}+\frac{\zeta+\gamma^5}{1-\gamma^2} \tag{2.35}$$

$$=\frac{2\zeta+\gamma^3+\gamma^5}{1-\gamma^2} \tag{2.36}$$

Step 6 : Substitute back into the inequality

$$\zeta\leq\frac{1-\gamma}{2}\cdot\frac{2\zeta+\gamma^3+\gamma^5}{1-\gamma^2} \tag{2.37}$$

$$=\frac{(1-\gamma)(2\zeta+\gamma^3+\gamma^5)}{2(1-\gamma^2)} \tag{2.38}$$

Step 7 : Use the identity $1-\gamma^2=(1-\gamma)(1+\gamma)$

$$\zeta\leq\frac{(1-\gamma)(2\zeta+\gamma^3+\gamma^5)}{2(1-\gamma)(1+\gamma)} \tag{2.39}$$

$$=\frac{2\zeta+\gamma^3+\gamma^5}{2(1+\gamma)} \tag{2.40}$$

Step 8 : Multiply both sides by $2(1+\gamma)$

$$2(1+\gamma)\zeta\leq 2\zeta+\gamma^3+\gamma^5 \tag{2.41}$$

$$2\zeta+2\gamma\zeta\leq 2\zeta+\gamma^3+\gamma^5 \tag{2.42}$$

$$2\gamma\zeta\leq\gamma^3+\gamma^5 \tag{2.43}$$

$$\zeta\leq\frac{\gamma^3+\gamma^5}{2\gamma} \tag{2.44}$$

$$\zeta\leq\frac{\gamma^2+\gamma^4}{2} \tag{2.45}$$

Therefore, we obtain an **upper bound** on ζ :

$$\zeta\leq\frac{\gamma^2+\gamma^4}{2} \tag{2.46}$$

Combined bounds :

$$\frac{\gamma^2+\gamma+1-2\gamma^3-\gamma^5}{4} \leq \zeta \leq \frac{\gamma^2+\gamma^4}{2} \tag{2.47}$$

where $0 < \gamma < 1$.

2.1.3 Step-by-step derivation for the third inequality

Starting from the inequality :

$$\zeta \cdot \sum_{i=0}^{\infty} \gamma^i \leq \frac{1}{4}V(g) + \frac{2}{4}V(0) + \frac{1}{4}V(1)$$

Step 1 : Simplify the left side using the geometric series

$$\zeta \cdot \sum_{i=0}^{\infty} \gamma^i = \zeta \cdot \frac{1}{1-\gamma} \tag{2.48}$$

$$= \frac{\zeta}{1-\gamma} \tag{2.49}$$

Step 2 : Use the previously derived expression for the right side From our earlier calculation :

$$\frac{1}{4}V(g) + \frac{2}{4}V(0) + \frac{1}{4}V(1) = \frac{4\zeta+\gamma+2\gamma^3+\gamma^5}{4(1-\gamma^2)} \tag{2.50}$$

Step 3 : Set up the inequality

$$\frac{\zeta}{1-\gamma} \leq \frac{4\zeta+\gamma+2\gamma^3+\gamma^5}{4(1-\gamma^2)} \tag{2.51}$$

Step 4 : Use the identity $1-\gamma^2 = (1-\gamma)(1+\gamma)$

$$\frac{\zeta}{1-\gamma} \leq \frac{4\zeta+\gamma+2\gamma^3+\gamma^5}{4(1-\gamma)(1+\gamma)} \tag{2.52}$$

Step 5 : Multiply both sides by $(1-\gamma)$

$$\zeta \le \frac{4\zeta + \gamma + 2\gamma^3 + \gamma^5}{4(1+\gamma)} \tag{2.53}$$

Step 6 : Multiply both sides by $4(1+\gamma)$

$$4(1+\gamma)\zeta \le 4\zeta + \gamma + 2\gamma^3 + \gamma^5 \tag{2.54}$$

$$4\zeta + 4\gamma\zeta \le 4\zeta + \gamma + 2\gamma^3 + \gamma^5 \tag{2.55}$$

Step 7 : Subtract 4ζ **from both sides**

$$4\gamma\zeta \le \gamma + 2\gamma^3 + \gamma^5 \tag{2.56}$$

$$\zeta \le \frac{\gamma + 2\gamma^3 + \gamma^5}{4\gamma} \tag{2.57}$$

$$\zeta \le \frac{1 + 2\gamma^2 + \gamma^4}{4} \tag{2.58}$$

Therefore, we obtain another **upper bound** on ζ :

$$\zeta \le \frac{1 + 2\gamma^2 + \gamma^4}{4} \tag{2.59}$$

Final combined bounds from all three inequalities :

$$\frac{\gamma^2 + \gamma + 1 - 2\gamma^3 - \gamma^5}{4} \le \zeta \le \min\left\{\frac{\gamma^2 + \gamma^4}{2}, \frac{1 + 2\gamma^2 + \gamma^4}{4}\right\} \tag{2.60}$$

where $0 < \gamma < 1$.

2.1.4 Step-by-step derivation for the fourth inequality

Starting from the inequality :

$$\frac{1}{4}V(g) + \frac{1}{4}V(0) + \frac{1}{4}(\zeta + \gamma^2\zeta\sum_{i=0}^{\infty}\gamma^{2i}) + \frac{1}{4}(\zeta\sum_{i=0}^{\infty}\gamma^{2i}) \le \frac{1}{4}V(g) + \frac{2}{4}V(0) + \frac{1}{4}V(1)$$

Step 1 : Cancel the $\frac{1}{4}V(g)$ terms from both sides

$$\frac{1}{4}V(0)+\frac{1}{4}(\zeta+\gamma^2\zeta\sum_{i=0}^{\infty}\gamma^{2i})+\frac{1}{4}(\zeta\sum_{i=0}^{\infty}\gamma^{2i})\leq\frac{2}{4}V(0)+\frac{1}{4}V(1) \tag{2.61}$$

Step 2 : Simplify using the geometric series $\sum_{i=0}^{\infty}\gamma^{2i}=\frac{1}{1-\gamma^2}$

$$\frac{1}{4}V(0)+\frac{1}{4}\left(\zeta+\gamma^2\zeta\frac{1}{1-\gamma^2}\right)+\frac{1}{4}\left(\zeta\frac{1}{1-\gamma^2}\right)\leq\frac{2}{4}V(0)+\frac{1}{4}V(1) \tag{2.62}$$

Step 3 : Factor out common terms

$$\frac{1}{4}V(0)+\frac{1}{4}\zeta\left(1+\frac{\gamma^2}{1-\gamma^2}\right)+\frac{1}{4}\zeta\frac{1}{1-\gamma^2}\leq\frac{2}{4}V(0)+\frac{1}{4}V(1) \tag{2.63}$$

Step 4 : Simplify the coefficient of ζ

$$1+\frac{\gamma^2}{1-\gamma^2}+\frac{1}{1-\gamma^2}=\frac{1-\gamma^2+\gamma^2+1}{1-\gamma^2}=\frac{2}{1-\gamma^2} \tag{2.64}$$

So the inequality becomes :

$$\frac{1}{4}V(0)+\frac{1}{4}\zeta\frac{2}{1-\gamma^2}\leq\frac{2}{4}V(0)+\frac{1}{4}V(1) \tag{2.65}$$

$$\frac{1}{4}V(0)+\frac{\zeta}{2(1-\gamma^2)}\leq\frac{1}{2}V(0)+\frac{1}{4}V(1) \tag{2.66}$$

Step 5 : Rearrange to isolate the ζ term

$$\frac{\zeta}{2(1-\gamma^2)}\leq\frac{1}{2}V(0)-\frac{1}{4}V(0)+\frac{1}{4}V(1) \tag{2.67}$$

$$\frac{\zeta}{2(1-\gamma^2)}\leq\frac{1}{4}V(0)+\frac{1}{4}V(1) \tag{2.68}$$

$$\zeta\leq\frac{(1-\gamma^2)}{2}(V(0)+V(1)) \tag{2.69}$$

Step 6 : Substitute the expressions for $V(0)$ and $V(1)$

$$V(0)+V(1) = \frac{\zeta+\gamma^3}{1-\gamma^2} + \frac{\zeta+\gamma^5}{1-\gamma^2} \tag{2.70}$$

$$= \frac{2\zeta+\gamma^3+\gamma^5}{1-\gamma^2} \tag{2.71}$$

Step 7 : Substitute back into the inequality

$$\zeta \leq \frac{(1-\gamma^2)}{2} \cdot \frac{2\zeta+\gamma^3+\gamma^5}{1-\gamma^2} \tag{2.72}$$

$$= \frac{2\zeta+\gamma^3+\gamma^5}{2} \tag{2.73}$$

Step 8 : Multiply both sides by 2

$$2\zeta \leq 2\zeta+\gamma^3+\gamma^5 \tag{2.74}$$

$$0 \leq \gamma^3+\gamma^5 \tag{2.75}$$

$$0 \leq \gamma^3(1+\gamma^2) \tag{2.76}$$

Since $0 < \gamma < 1$, we have $\gamma^3 > 0$ and $(1+\gamma^2) > 0$, so this inequality is always satisfied. This means the fourth inequality does not provide an additional constraint on ζ.

Updated final bounds from all four inequalities :

$$\frac{\gamma^2+\gamma+1-2\gamma^3-\gamma^5}{4} \leq \zeta \leq \min\left\{\frac{\gamma^2+\gamma^4}{2}, \frac{1+2\gamma^2+\gamma^4}{4}\right\} \tag{2.77}$$

where $0 < \gamma < 1$. The fourth inequality is automatically satisfied and does not further constrain the bounds.

2.2 Q-Learning

2.3 Preferences over Decision Tree Policies

2.4 Results

Chapitre 3

Conclusion

3.1 What happens when the MDP's transitions are independent of the current state ?

Deuxième partie

An easier problem : Learning Decision Trees for MDPs that are Classification tasks

Chapitre 4

DPDT-intro

Chapitre 5

DPDT-paper

Chapitre 6

Conclusion

Troisième partie

Beyond Decision Trees : what can be done with other Interpretable Policies ?

Conclusion générale

Bibliographie

[1] Jan-Niklas Eckardt et al. « Reinforcement learning for precision oncology ». In : *Cancers* 13.18 (2021), p. 4624.

[2] Romain Gautron. « FApprentissage par renforcement pour l'aide à la conduite des cultures des petits agriculteurs des pays du Sud : vers la maîtrise des risques. » Thèse de doct. Montpellier SupAgro, 2022.

[3] Donald Ervin Knuth. « Finite semifields and projective planes ». Thèse de doct. California Institute of Technology, 1963.

[4] Edouard Leurent. « Safe and Efficient Reinforcement Learning for Behavioural Planning in Autonomous Driving ». Thèse de doct. Université de Lille, 2020.

[5] Ameet Talwalkar Mehryar Mohri Afshin Rostamizadeh. *Foundations of Machine Learning*. MIT Press, 2012.

[6] Richard S. Sutton et Andrew G. Barto. *Reinforcement Learning : An Introduction*. Cambridge, MA : The MIT Press, 1998.

[7] Alan Turing. « Computing Machinery and Intelligence ». In : *Mind* (1950).

Annexe A

Programmes informatiques

Les listings suivants sont au cœur de notre travail.

Listing A.1 – Il est l'heure

```
#include <stdio.h>
int heures, minutes, secondes;

/****************************************************/
/*                                                  */
/*            print_heure                           */
/*                                                  */
/*   But:                                           */
/*      Imprime l'heure␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣*/
/*␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣*/
/*␣␣␣Interface:␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣*/
/*␣␣␣␣␣␣Utilise␣les␣variables␣globales␣␣␣␣␣␣␣␣␣␣␣␣␣␣*/
/*␣␣␣␣␣␣␣␣␣␣␣heures,␣minutes,␣secondes␣␣␣␣␣␣␣␣␣␣␣␣␣␣*/
/*␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣*/
/****************************************************/

void␣print_heure(void)
{
␣␣printf("Il␣est␣%d␣heure",heures);
␣␣if␣(heures␣>␣1)␣printf("s");
␣␣printf("␣%d␣minute",minutes);
␣␣if␣(minutes␣>␣1)␣printf("s");
␣␣printf("␣%d␣seconde",secondes);
␣␣if␣(secondes␣>␣1)␣printf("s");
␣␣printf("\n");
}
```

Listing A.2 – Factorielle

```
int factorielle(int n)
{
  if (n > 2) return n * factorielle(n - 1);
  return n;
}
```

Table des matières