CHALMERS | UNIVERSITY OF GOTHENBURG

Master's Thesis

Decision Making Under Uncertainty:

A Robust Optimization

EMMANOUIL ANDROULAKIS

Department of Mathematical Sciences
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2014
Master's Thesis

Abstract

Conventional approaches for decision making often assume that access to full information is possible. Nevertheless, such explicit knowledge about the model's dynamics is seldom available in practical applications. In this thesis the problem of the construction of a plan for a sequence of decisions under an uncertain adversarial environment is addressed. The uncertainty of the information is modeled via a set of sequential Markov decision processes and a number of methods are utilized in order to produce a robust plan, depending on the setting. Additionally, the intractability of the computation of an exact solution, with the Cutting Plane method, is shown, in the case where policy value hyperplanes are viewed as potential cuts.

Keywords. Markov decision process, uncertainty, worst-case prior, robustness, cutting-plane, NP-hard, weighted majority algorithm, wma-pusr, contextual bandits.

Acknowledgement

It seems a rather impossible task to list, in so little space, all the persons that had an influence on me, during the writing of my thesis. I feel lucky to have had so many inspirational people present in my student life.

First of all I would like to express my gratitude to one of the most intelligent persons I have ever met, my advisor Christos Dimitrakakis, for without his priceless guidance this thesis wouldn't exist. Thank you for your trust and for giving me the opportunity to work with such an interesting project.

Furthermore, I would like to thank Laerti Vasso and Martynas Šeškaitis for the stimulating and interesting (mathematical or not) discussions we had and for keeping me company, especially the past year. Loyal friends are not easy to come by.

Of course a big thanks goes to all my friends for the wonderful time we had together in Sweden. In random order: Johannes, Angie, George, Stavros, Giannis, Michael, Marine, Vasiliki, Angelos, Johanna, Christina, Maria, Yosi, Andy and every other person that has spent time with me.$\ddot{\smile}$

I would also like to mention and bow down to some of the professors that had significantly impacted my way of thinking, during my masters' studies: Alexander Herbertsson, Patrik Albin, Mattias Sundén and Sergey Zuev.

Moreover, I feel the need to thank professors Dimitris Cheliotis, Costis Melolidakis and Giannis Kougioumoutzakis here, for their encouragement and assistance. If it weren't for them I wouldn't have ended up studying for a master's degree.

Naturally, I owe everything to my family. My brother for his devotion and aid, my mother who is the reason that I love books (you can't avoid genes I guess) and my father that gave me the best advice ever: *'When things are about to get tough, just take a deep breath and dive right into it'.*

Finally, I would like to thank the most wonderful, amazing and magnificent person I have ever met, Suvi, for being in my life.

Thank you all.

Androulakis G. Emmanouil, Gothenburg, 2014.

Contents

1 Introduction **1**
- 1.1 Uncertainty . . . 1
- 1.2 Sequential MDPs . . . 2
- 1.3 The Problem . . . 6
  - 1.3.1 A Model of Possible Scenarios . . . 6
  - 1.3.2 The Worst-Case Prior . . . 7
- 1.4 Contribution of the Thesis . . . 7

2 The C-P Method **10**
- 2.1 Values & Visuals . . . 10
  - 2.1.1 Supremum Values . . . 12
- 2.2 The Cutting-Plane Method . . . 14
  - 2.2.1 Finding the cuts . . . 16
  - 2.2.2 Using the Cutting-Plane Method . . . 18
- 2.3 NP-hardness . . . 19

3 A Naive Algorithm **21**
- 3.1 Obtaining Policy Values . . . 21
  - 3.1.1 Notation & Definitions . . . 21
  - 3.1.2 Approximating the Policy Values with Uniform Sampling . . . 23
  - 3.1.3 The Uniform Sampling Algorithm . . . 24
  - 3.1.4 Analysis . . . 24

4 Weighted Majority **29**
  - 4.0.5 A zero-sum game: Decision Maker VS Nature . . . 30
  - 4.0.6 Notation & Definitions . . . 30
  - 4.0.7 WMA . . . 31
  - 4.0.8 Analysis . . . 34
  - 4.0.9 Generalizing WMA: WMA-PUSR . . . 34
  - 4.0.10 Walk-through . . . 36

4.0.11 Analysis . . . 37

5 Contextual Bandits **45**
5.1 Bandits . . . 45
5.1.1 The Multi-Armed Bandit Problem . . . 45
5.1.2 Contextual Bandits . . . 45
5.2 LinRel & SupLinRel . . . 46
5.2.1 Associative reinforcement learning with linear value functions . . . 47
5.2.2 The Algorithms . . . 48
5.2.3 Analysis . . . 49

6 Conclusion **52**
6.0.4 Cutting-plane . . . 52
6.0.5 Uniform Sampling . . . 52
6.0.6 Weighted Majority . . . 52
6.0.7 Contextual Bandits . . . 53
6.1 Future Directions . . . 53

Appendices **57**

Appendix A Preliminaries **58**
A.1 Linear Algebra . . . 58
A.1.1 Eigenvalues & Eigenvectors . . . 58
A.2 Analysis . . . 58
A.2.1 Convex Analysis . . . 58
A.2.2 Taylor . . . 59
A.3 Measure Theory . . . 59
A.4 Computational Theory . . . 61
A.4.1 Computational Complexity . . . 61

List of Figures

1.1 Reading Guide 9

2.1 2D slice plot of policy values for three distinct policies. 11
2.2 3D slice plot of the policy value for a policy. 12
2.3 Convex supremum 13
2.4 Dominated policies 14
2.5 A cutting-plane 15
2.6 Neutral vs deep cuts 15
2.7 A decreasing sequence of polyhedron cuts 16
2.8 A robust policy. 18

3.1 Useless policies 22
3.2 MC approximated policy values 23

List of Algorithms

1 Cutting-plane 19
2 Uniform Sampling 24
3 WMA 32
4 WMA-PUSR 35
5 LinRel 48
6 SupLinRel 49

1

Introduction

'Uncertainty is the only certainty there is.'

—John Allen Paulos

1.1 Uncertainty

There are times where a decision needs to be made with incomplete or censored information. This lack of knowledge leads unavoidably to occasions where the expected outcome of an action is inaccurate. Such inaccuracies in planning are usually extremely undesirable, and thus managing to operate in such uncertain conditions is an important issue.

Uncertainty about the environment generates a lot of issues for optimal planning. Conventional approaches for decision making usually assume perfect information. That means that the parameters of the entertained models are accurately known and all the relevant probability distributions for the participating random variables are explicitly specified. Nonetheless, these assumptions are a bit unrealistic and this kind of definite knowledge about the involved dynamics is infrequently encountered in practical applications. A policy, i.e. a plan, constructed based on inaccurate calibrations is bound to

suffer from inadequate performance or -even worse- infeasibility of the actions prescribed by the policy may arise.

When one is forced to take action while the information at hand is deficient, mistakes happen with higher probability. Any structured and methodical approach to plan optimally in such cases carries a risk, rooted on that uncertainty. Nevertheless the execution of an act however certain or uncertain one is for it's efficiency, reveals potential information about the dynamics of the environment and this new knowledge can be used to further formulate a plan of actions. Therefore deciding to explore the effects that actions have to the environment may be useful if the information at hand at that point is believed not to be sufficient. On the other hand if one has adequate information, then there is no use to choose speculative actions that may eventually end-up being sub-optimal. Learning to manage the trade-off between exploration and exploitation is an essential component of efficient planning under uncertainty and thus artificial intelligence algorithms which balance these concepts effectively can demonstrate an extraordinary degree of competency in situations where the dynamics are unclear.

1.2 Sequential Markov Decision Problems

There are many ways to approach sequential decision making. In this section we describe and discuss the Markov decision process framework under which we will be operating. The Markov decision process model (often encountered as stochastic dynamic programs or stochastic control programs in the literature) is useful for modelling sequential decision making when the outcomes are not certain.

To describe this procedure, consider a Decision Maker, who at a specified point of time, faces the problem of taking a decision. She observes her environment and considers the alternatives that are available to her at this given point. After evaluating her options, she decides on an action and executes it. This action has two immediate effects:

1. the Decision Maker receives a reward (or pays a cost)
2. the environment is affected perchance by the action in some way.

At this consequent point in time, the Decision Maker faces an analogous problem, but now the environment may have changed and the available actions may not be the same any more. This sequence of decisions generates a string of rewards. The Decision Maker tries to plan her actions accordingly with the goal of maximizing her total reward. Of course, if the rewards are negative, they can be interpreted as costs, and then the intention of the Decision Maker would be to minimize the total cost.

In order to model and approach rigorously the above succession of events, we give the following definition:

Definition 1.1. A *Markov decision process* is a quadruple $\mu = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$, where

- $\mathcal{S}$ is a set of *states* (we will be referring to this as the *state space*)

- if $\mathcal{A}_s$ is a set of *actions* that are available to the Decision Maker while in state $s \in \mathcal{S}$, then denote by $\mathcal{A}$ the collection of all possible actions, i.e. $\mathcal{A} = \bigcup_{s \in \mathcal{S}} \mathcal{A}_s$ (we will be referring to $\mathcal{A}$ as the *action space*)
- $\mathcal{R}(\omega, a, s)$ is a *reward function* that describes the distribution over the rewards realized when selecting the action $a \in \mathcal{A}$, while in state $s \in \mathcal{S}$. The argument ω is used to generate stochastic rewards.
- $\mathcal{P}_a(s' \mid s)$ is the *transition probability* from state s to state s' if action a is chosen while in state s'.

Remark 1.2. One factor that is also important to consider is the time horizon $\mathcal{T}$, which might be finite or infinite. To include the time horizon in the description of the problem we may write $\mu = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \mathcal{T} \rangle$ and refer to this 5-tuple as a *Markov decision problem.*

Now we can describe the decision making procedure in the language of Markov decision problems as follows.

A Decision Maker has to take a sequence of decisions. At each decision epoch $t \leq \mathcal{T}$, she observes her environment, represented by a system *state* $s \in \mathcal{S}$ and evaluates her choices, by examining the action space $\mathcal{A}$. She selects and performs an *action* $a \in \mathcal{A}$. As a result to this action

1. she receives an immediate *reward* $r_{a,s}^{(t)}$ according to $\mathcal{R}(\omega, a, s)$ **and**
2. the system advances to a new state $s' \in \mathcal{S}$ at a later point in time $t' = t + 1$, according to a probability distribution $\mathcal{P}_a(s'|s)$ imposed by the chosen action.

Both the rewards and the transition probabilities depend on previous states and actions *only* through the current system state. That means that $\mathcal{P}_a(s' \mid s)$ depends only the previous state s (and the action a), and not on older states that the system might have occupied (or older actions taken by the Decision Maker). Thus

$$\mathbb{P}\left[\, S_{n+1} = s \mid S_1 = s_1, S_2 = s_2, \ldots S_n = s_n \,\right] = \mathbb{P}\left[\, S_{n+1} = s \mid S_n = s_n \,\right],$$

where $S_i, \quad i = 1,2,\ldots,n, \quad n \in \mathbb{N}$, are random variables representing the state of the system at the time time point t_i.

As this procedure moves forward in time, the Decision Maker makes choices in the different system states, resulting in a (finite or infinite) sequence of rewards (or costs).

States

At each decision time point, one system state is active. Recall that we denoted the state space by $\mathcal{S}$. The set $\mathcal{S}$ may be one of the following types:

- an arbitrary finite set
- an arbitrary infinite, but countable set
- a compact subset of a Euclidean space of finite dimension
- a non-empty Borel subset of complete, separable metric spaces

Actions

Actions represent the Decision Maker's alternatives on how to deal with each state. Since the system is ongoing and next states depend on previous actions (through a probability distribution), the Decision Maker needs to avoid being short-sighted and try not to take any decision myopically. An action that may seem very attractive now, may be, in reality, not the optimal choice, as there is a possibility that such an action will drop the system into some very unfavorable states in the future. Anticipating rewards on the future states can be a deciding factor on how well the Decision Maker will perform in total. The set of available actions A_s, while in state s, can be either one of the types described for the state space.

Rewards

Every time the Decision Maker chooses an action from the action space, she receives a reward $r_{a,s} := r_{a,s}(\omega)$. These rewards are *stochastic* and are generated according to a probability distribution. More specifically, they depend on the selected action $a \in \mathcal{A}$, on the current system state $s \in \mathcal{S}$ and on the outcome of an *experiment* ω in an outcomes space Ω. The set Ω must be non-empty and can contain anything. The action a must be decided before knowing the outcome of the experiment ω.

Assumption 1.3. (Outcomes). For every $a \in \mathcal{A}$ and $s \in \mathcal{S}$ there exists a probability measure P on the measurable space $\langle \Omega, \Sigma \rangle$ such that the probability of the random outcome ω being in $E \subset \Omega$ is

$$P(E) = \mathbb{P}\left[\ \omega \in E\ \right], \quad \forall E \in \Sigma.$$

Definition 1.4. (Reward function). A *reward function* $\mathcal{R} : \Omega \times \mathcal{A} \times \mathcal{S} \longrightarrow \mathbb{R}$ defines the reward obtained by action $a \in \mathcal{A}$, while in state $s \in \mathcal{S}$ and the experiment outcome is $\omega \in \Omega$:

$$r_{a,s} = \mathcal{R}(\omega, a, s).$$

There will be a reward for each time epoch t, so $\{r_{a,s}^{(t)}\}_{t \leq \mathcal{T}}$ will be a sequence of random variables. The rewards have the *markovian* property, that is they depend only on the current state and action and not on the history of decisions or states. Adding up all the rewards creates the *total reward*. Maximizing the total reward is the main intent of the Decision Maker.

Transition Probabilities

Every action the Decision Maker takes may have an effect to the environment. Hence every action causes the system to evolve to a new state. The way that the system jumps from one state to another, is dictated by a probability distribution $\mathcal{P}_a(\cdot \mid \cdot) : \mathcal{S}^2 \times \mathcal{A} \to [0,1]$, which has the markovian property as well. Of course the

system is allowed to jump right into the same state, i.e. $\mathcal{P}_a(s \mid s)$ can be positive. Also, for every $s \in \mathcal{S}$ we assume

$$\sum_{s' \in \mathcal{S}} \mathcal{P}_a(s' \mid s) \leq 1$$

The expected value of a state s, at decision time t, may be evaluated as follows:

$$\sum_{s' \in \mathcal{S}} \mathbb{E}\left[\ r_{a,s'}\ \right] \cdot \mathcal{P}_a(s' \mid s)$$

Decision Rules

Decision rules are a way to describe how the Decision Maker decides on actions. They act as a prescription on what action to choose while in a certain state.

Decision rules can be

- History dependent
$$d : (\mathcal{S} \times \mathcal{A})^T \times \mathcal{S} \to \mathcal{A}$$
where $T \leq \mathcal{T} - 1$ or
- Markovian (memoryless)
$$d : \mathcal{S} \to \mathcal{A}$$

according to their degree of dependence to past information and can also be classified as

- Deterministic or
- Randomized

All the above combinations create four types of decision rules.

Policies

Define a *policy*, *strategy*, or *plan* as

$$\pi = (d_1, d_2, \dots, d_t, \dots)$$

which is a vector with dimension $\mathcal{T}$, containing an action (specified by a decision rule) for every decision time point t, $t \leq \mathcal{T}$. A policy instructs the decision maker on what action choices should be made in any possible future state. We call a policy *stationary* if it has the form:

$$\pi = (d, d, \dots)$$

We can classify policies in the following categories:

- History dependent or
- Markovian

depending to their degree of dependence to past information and can also be separated to

- Deterministic or
- Randomized

The most general type is policies which are randomized and history dependent, whereas the most specific are stationary Markov deterministic policies.
If at time t the system occupies the state $s^{(t)}$ and actions a follow a specific policy π, then we will use the following explicit notation for the rewards:

$$r^{(t)}_{a\sim\pi,s^{(t)}}$$

or the more simper $r_{a,s}$ if the above is implied easily from the context.

Discounting

Consider a Markov decision problem, with an *infinite* time horizon $\mathcal{T}$. Let π_1, π_2 be two policies and their corresponding rewards for each time epoch:

$$R_1 = (r_1, r_2, \dots) \text{ for } \pi_1 \text{ and}$$

$$R_2 = (2r_1, 2r_2, \dots) \text{ for } \pi_2,$$

where $r_t \geq \frac{1}{2},\ t = 1, 2, ...$

Obviously, policy π_2 seems to be more attractive than policy π_1, since the reward on each time epoch is double. However, adding up all the rewards to obtain the total reward we get $\sum_{t=1,2,\dots} r_t = \sum_{t=1,2,\dots} 2r_t = \infty$, making the two policies incomparable with respect to their total value.

One way to solve this issue is to introduce a *discount factor* $\gamma \in [0,1)$. Then

$$\sum_{t=1,2,\dots} \gamma^t r_t \leq \sum_{t=1,2,\dots} 2\gamma^t r_t < \infty$$

and we can easily decide which policy is preferable.

An intuitive explanation for the discount factor γ is that it balances relative preferable weights of current and future payments, with small values of γ prioritizing short-term rewards and larger values giving more emphasis to long-term gains.

1.3 Description & Formulation of the Problem

1.3.1 A Model of Possible Scenarios

Consider a Decision Maker who faces the problem of making a sequence of decisions, but her knowledge about the environment and what would happen precisely if she interacted with it, is *limited* or *noisy*. The first assumption we do is that the Decision Maker is

a reasonable thinker with no gambling tendencies and so the strategy of blindly selecting actions in the hope of landing something good, is not in the list of considerations. Hence, since we suppose that she wants to try and design a reasonable plan, she needs to take advantage of the available information. Considering that she does not have precise knowledge of the dynamics at play, trying to specify the parameters/variables of an explicit model in her attempt to optimize her actions, would be much of a risky speculation. Thus, based on the limited information she possess, we assume that she has some kind of belief about the dynamics of the world and she is willing to consider different possible scenarios. Instead of considering a Markov decision problem with uncertain dynamics, that might be proven to be completely off, we choose to model the uncertainty in the following way:

$\rightarrow$ we consider a *set* $\mathcal{M}$ of Markov decision problems, that contains candidates

$$\mu_j = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \mathcal{T} \rangle \quad j = 1, \ldots, |\mathcal{M}|.$$

Each one of the μ_j's describes an alternative possibility for the properties of the environment. If the Decision Maker is very unsure about the dynamics of the model she is interacting with, then the set $\mathcal{M}$ will contain a variety of very different μ_j's, whereas if she has a strong belief of what the dynamics look like then the set $\mathcal{M}$ can be less diversified. So the cardinality and properties of the set $\mathcal{M}$ depend on the amount and nature of the available information.

In order to find the safest possible *policy* (which will produce different rewards under different μ_j's!) we adopt one more hypothesis: we assume that the MDP μ that the Decision Maker is going to interact with, is chosen by an Adversary, in the most unfavourable (for the Decision Maker) way.

1.3.2 The Worst-Case Prior

Denote by $\xi \in [0,1]^M$ the probability distribution that represents the Decision Maker's belief of the selection of μ by the Adversary. To be more specific

$$\xi \triangleq (\xi_1, \xi_2, \ \ldots, \ \xi_M)^\top, \tag{1.1}$$

where $M = |\mathcal{M}|$ and

$$\sum_{m \in \{1,2,\ldots,M\}} \xi_m = 1 \tag{1.2}$$

so every ξ_m assigns a probability to the possibility of interacting with μ_m. One of the issues that the Decision Maker will have to face, by using such a model, will be to determine the *worst case prior distribution* $\xi^\star$, in order to base her decisions on and pick a robust policy $\pi^\star$.

1.4 Overview of the Solution & Contribution of the Thesis

A formulation of uncertainty via a set of possible Markov decision problem environments and a minimax optimization over them has, to our best knowledge, not been addressed in the literature so far. The prime motivation for proceeding with such modelling is:

- Decision making schemes that consider only one model and try to estimate its uncertain dynamics, might suffer from approximation errors. These inaccuracies may be proven catastrophic for the performance of the policies produced within this kind of model.
- Considering distinct (mutually exclusive or not) possible scenarios to deal with decision making is an effective problem solving technique that allows for more flexibility and guarantees that no stone will be left unturned.
- Minimax decisions are the best possible play against the worst case scenario. This is a natural approach when we want to guard against an adversarial environment.

The first part of the thesis deals with the case of an infinite number of decisions. We illustrate how approaching any kind of optimization (not just a minimax) with the Cutting-plane method, that seems rather promising, given the visual representation of a possible solution, turns out to be intractable.

In the second part, where a finite decision horizon is assumed, we start by approaching the problem in a very straightforward way and use uniform sampling to proceed.

Afterwards, the Weighted Majority algorithm is applied to our problem, under specific assumptions. These assumptions are weakened in the next section and we modify the algorithm to fit the more general case. We prove performance guarantees and a regret bound theorem for the modified algorithm (Lemma 4.1, Theorem 4.5, Corollary 4.6, Theorem 4.8, Corollary 4.9).

Lastly, the reinforced learning algorithm `SupLinRel` is used, in the most general setting and the regret bound is given, for our case (Theorem 5.2).

Proofs of theorems that already exist in other sources are omitted and all the basic mathematical concepts involved can be found in the Appendix.

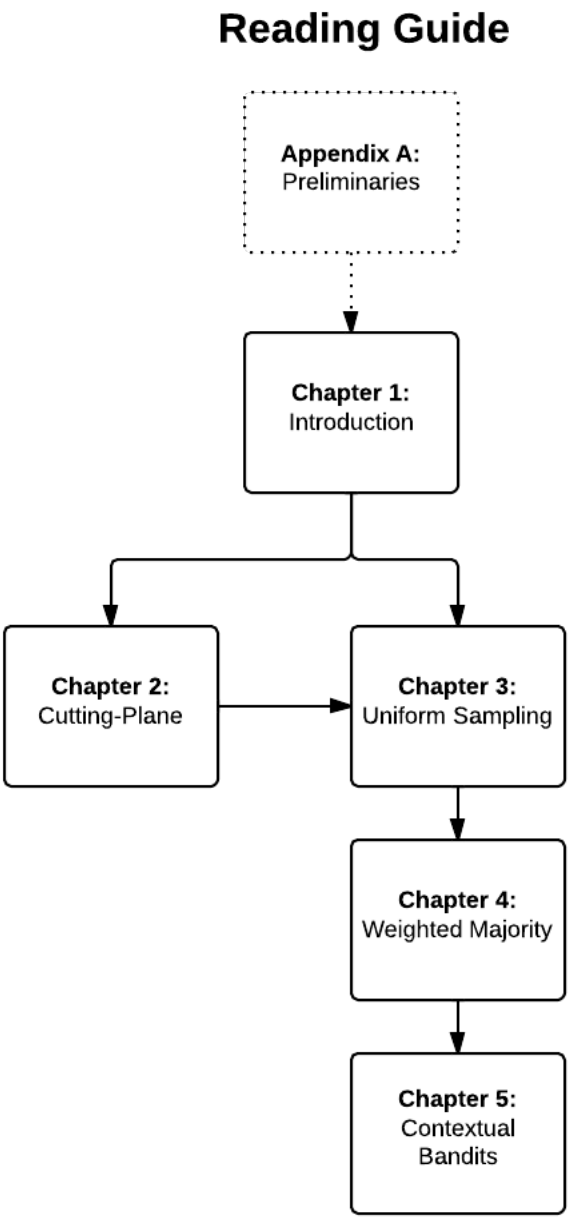

Figure 1.1: Reading Guide.

2

The Cutting-Plane Method

'Prediction is very difficult, especially about the future.'

—Niels Bohr

In this chapter we consider Markov decision problems with infinite horizon $\mathcal{T}$. Motivated by the visual representation of the solution(s), we investigate if the Cutting-plane method can be used in order to retrieve an optimal policy.

2.1 Policy Values and Visuals

Consider a Markov decision problem $\mu = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \mathcal{T} \rangle$, $\mathcal{T} = \infty$ and an arbitrary policy set Π. It is important to note that we do not restrict the policy space Π. These policies in Π can be of any kind: deterministic, randomized, may or may not have the Markov property etc. When the Decision Maker chooses actions a according to some policy π in each time step t, she receives a reward $r^{(t)}_{a\sim\pi,s^{(t)}}$ that depends on the current state of the system and the action a through a probabilistic distribution $\mathcal{R}_{a,s}$. That means that each reward $r^{(t)}_{a\sim\pi,s^{(t)}}$ is a random variable.

The *value* of action $a \in \mathcal{A}$ when in state $s \in \mathcal{S}$:

$$V_s^a \triangleq \mathbb{E}\left[\, r_{a,s} \,\right].$$

So, for a given state s, for every action a corresponds a $V_s^a \in \mathbb{R}$.

Since the reward of each action is a random variable, the discounted total reward from following actions prescribed by a policy π: $\sum_{t=1}^{\mathcal{T}} \gamma^t r^{(t)}_{a\sim\pi,s^{(t)}}$, (where $\gamma \in (0,1)$), is also a random variable. Define the *policy value* of policy π when in μ as the expected

value of the total reward:

$$V_{\mu}^{\pi} \triangleq \mathbb{E}\left[\sum_{t=1}^{\mathcal{T}} \gamma^t r_{a\sim\pi, s^{(t)}}^{(t)}\right].$$

So, for each $\pi \in \Pi$ corresponds a $V_{\mu}^{\pi} \in \mathbb{R}$.

Now, consider a set of Markov decision problems $\mathcal{M}$, with $|\mathcal{M}| = M$ and let ξ be a vector of probabilities in $[0,1]^M$ as in (1.1).

Let V_{π} be the $1 \times M$ vector of values for policy π for the given set of Markov decision problems $\mathcal{M}$:

$$V_{\pi} \triangleq (V_{\mu_1}^{\pi}, V_{\mu_2}^{\pi}, \ \ldots \ , V_{\mu_M}^{\pi})$$

and let V_{ξ}^{π} be the weighted mean value of the policy π with respect to the distribution vector $\xi \in [0,1]^{M\times 1}$:

$$V_{\xi}^{\pi} \triangleq V_{\pi} \cdot \xi = \sum_{m=1}^{M} V_{\mu_m}^{\pi} \xi_m$$

.

So, each policy π receives a different value $V_{\xi_m}^{\pi} \in \mathbb{R}$ depending on ξ_m, $m = 1,2,\ldots,M$. That essentially means that each policy value is an M-dimensional hyperplane.

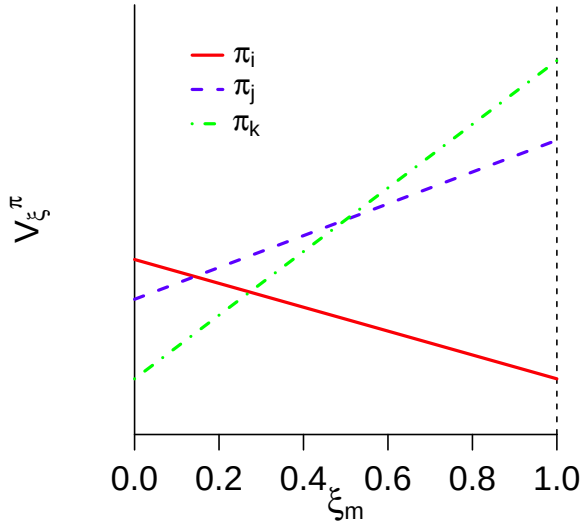

Figure 2.1: 2D slice plot of policy values for three distinct policies. Policy values are in fact M-dimensional hyperplanes.

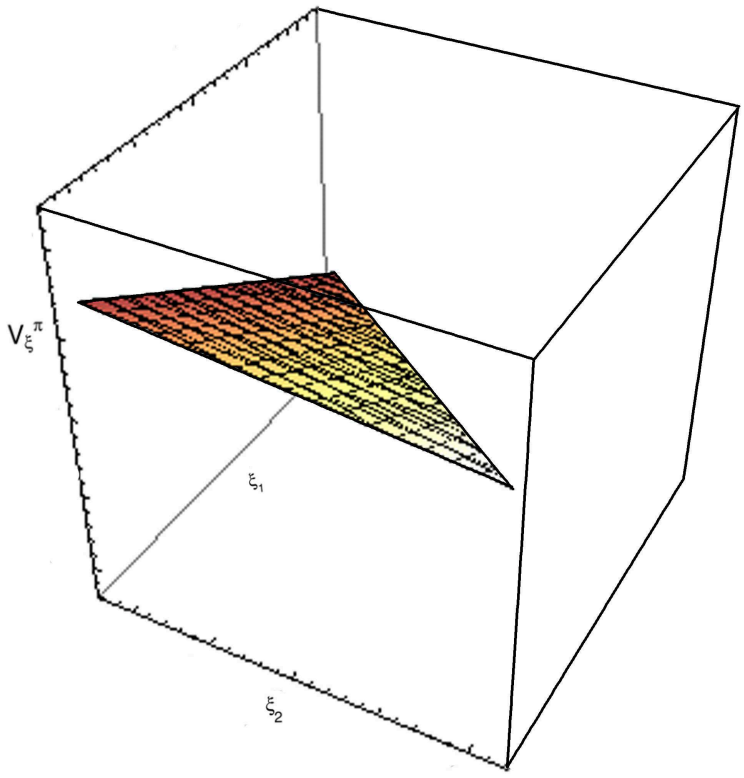

Figure 2.2: 3D slice plot of the policy value for a policy.

Furthermore, denote by $\mathcal{V}_\xi$ the $1 \times N$ vector containing the ξ-weighted mean values for each policy:

$$V_\xi = (V_\xi^{\pi_1}, V_\xi^{\pi_2}, \ldots, V_\xi^{\pi_N}),$$

where $\pi_i \in \Pi$ and $|\Pi| = N$.

2.1.1 Supremum Values

The Decision Maker tries to maximize her total reward, so policies with a higher value are obviously preferred. When $V_\xi^\pi > V_\xi^{\pi'}$ for all ξ, then π *strongly* dominates π'.

Denote by $\overline{\mathcal{V}}_{\mathcal{M},\Pi}$ (or simply $\overline{\mathcal{V}}$) the lowest upper bound of V_ξ^π for $\pi \in \Pi$, $\xi \in [0,1]^M$. Since $\overline{\mathcal{V}}$ is an upper bound for the policy values, there does not exist a policy π such that $\overline{\mathcal{V}} < V_\xi^\pi$. The Decision Maker, thus, tries to find a policy that is as close to the $\overline{\mathcal{V}}$ as possible. Of course some policies may have higher values than others for a specific ξ and a lower value for another ξ'. When $V_\xi^\pi > V_\xi^{\pi'}$ for some ξ then π *weakly* dominates π' in these ξ's. Our goal is to estimate the worst case ξ and pick the policy that is closer to $\overline{\mathcal{V}}$ at that point.

The following theorem will help us to see how an optimal solution, among the policies of Π, will look like.

Theorem 2.1. *Let $(f_i)_{i\in I}$ be convex functions on a convex compact set $X \subseteq R^N$. Then $f \triangleq \sup_i f_i$ is convex.*

Proof. Let x, $y \in X$ and $\theta \in [0,1]$. Every f_i is convex and $f \geq f_i$ for every i. Thus

$$f_i\left(\theta x + (1-\theta)y\right) \leq \theta f_i(x) + (1-\theta) f_i(y) \leq \theta f(x) + (1-\theta) f(y) \quad \forall i \in I$$

Taking the sup over all i's we obtain that f is convex:

$$f\left(\theta x + (1-\theta)y\right) \leq \theta f(x) + (1-\theta) f(y).$$

□

Corollary 2.2. Since the expected value of a random variable is linear (and thus convex), the above holds for V_ξ^π and their supremum $\overline{\mathcal{V}}$.

Additionally, if the policy space Π contains an infinite number of policies then $\overline{\mathcal{V}}$ can be strictly convex.

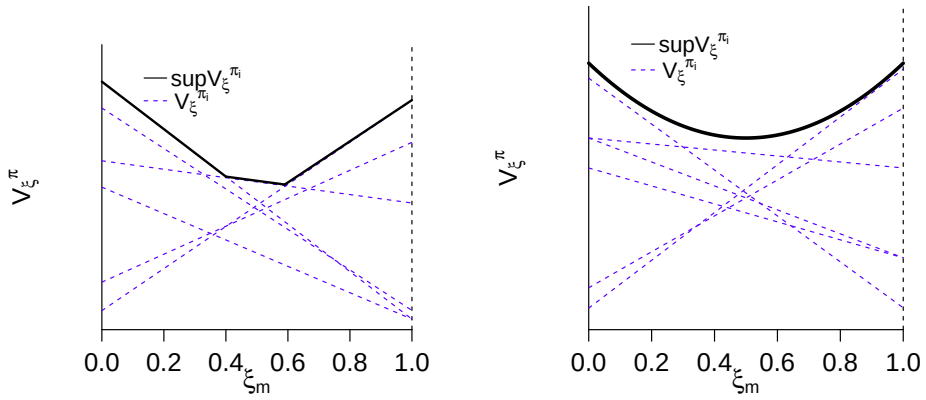

Figure 2.3: (Left) A convex $\overline{\mathcal{V}}$. (Right)A strictly convex $\overline{\mathcal{V}}$. This can only happen if $|\Pi| = \infty$.

If, given a certain policy $\pi \in \Pi$ that has a value V_ξ^π for some ξ, we can exclude all arbitrary policies that perform worse than π (the ones that have values less than π's for that ξ), then we decrease the possibility to choose a sub-optimal policy. Then we can focus on the policies that have a value higher than V_ξ^π and repeat the above. This way we can continually improve our selections until we are as close to $\overline{\mathcal{V}}$ as desired.

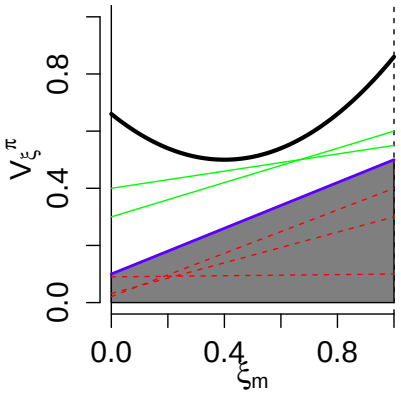

Figure 2.4: All policies that lie in the shaded area have a value lower than the selected policy (blue) and thus can be excluded.

So, our goal is to find a policy that maximizes the total reward (or minimizes the distance between $\overline{\mathcal{V}}$ and the hyperplane that corresponds to the reward of the policy) subject to a number of linear constraints. Each constraint is the hyperplane of the policy values. In the following segment, we describe a method that can be used to approach this problem.

2.2 The Cutting-Plane Method

In this section a method for solving convex optimization problems is described. The method is based on the utilization of *cutting-planes*, which are hyperplanes that divide the space into two subspaces: one that contains the optimal points and one that does not. The objective of cutting-plane methods is to detect a point in a convex set $X \subseteq R^n$, which is called the *target set*. In an optimization problem, X can be taken as the set of optimal (or ϵ-suboptimal) points for the problem and so by using this method we can find an optimal (or ϵ-suboptimal) point which will be the solution.

This is done in two steps. First, we pick a point $x \in \mathbb{R}^n$. Then we query an *oracle*, which examines the position of x and returns the following information:

- either $x \in X$ and thus we have a solution to the optimization problem
- or $x \notin X$ and the oracle produces a separating hyperplane between x and X, i.e., $a \neq 0$ and b such that

$$a^\top z \leq b \text{ for } z \in X, \quad a^\top x \geq b.$$

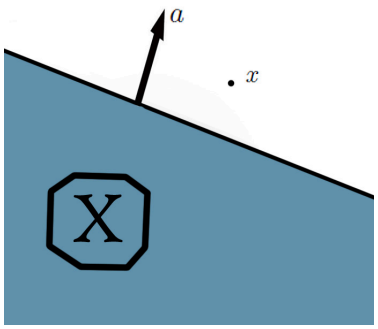

Figure 2.5: A cutting-plane, for the target set X, at the query point x, is defined by the inequality $a^\top z \leq b$. The search for an optimal point $x^\star \in X$ can be continued only within the shaded half-space. The unshaded half-space $\{z \mid a^\top z > b\}$ does not contain any points of X.

Cuts & Polyhedrons

The above hyperplane is called a *cutting-plane* since it cuts out the half-space $\{z \mid a^\top z > b\}$. No such point could be in the target set X and therefore we stop considering all these points in our investigation towards a solution (Figure (2.5)).

There are two types of cuts that can be considered:

- *neutral cuts*, where the query point x is contained in the cutting plane $z^t z = b$
- *deep cuts*, where the query point x lies in the interior of the half-space that is being excluded from the search

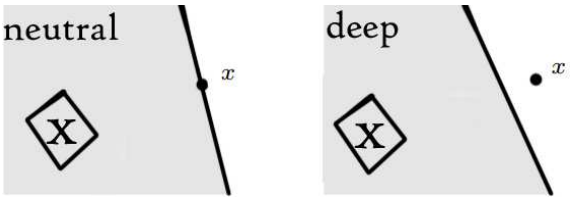

Figure 2.6: A neutral and a deep cut. In the neutral cut the query point x is on the boundary of the half-space that is about to be excluded.

These cuts form a decreasing sequence of polyhedrons $\mathcal{P}$ that contain the target set X.

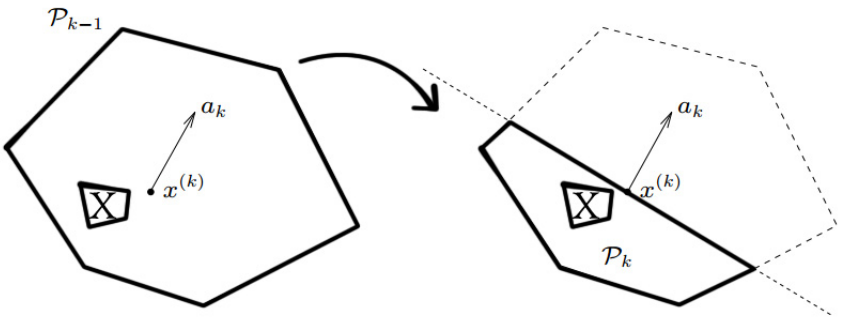

Figure 2.7: $X \subseteq \cdots \subseteq \mathcal{P}_{k+1} \subseteq \mathcal{P}_k \subseteq \ldots$

2.2.1 Finding the cuts

After picking the query point x there are two things that need to be decided by the oracle: 1)the *feasibility* and 2)*optimality* of the query point x must be assessed. We illustrate how the above issues can be approached separately and then we combine them together in order to see how an optimal point of a constrained optimization can be retrieved.

Unconstrained minimization

First, consider the optimization problem

$$\min f_0(x),$$

where f_0 is convex and no more constraints apply. In order to construct a cutting-plane, at x we may proceed as follows:

- Find a sub-gradient $g \in \partial f_0(x)$. If $f_0(x)$ is differentiable then $g = \nabla f_0(x)$
- If $g = 0$ then $x \in X$ and we are done
- If $g \neq 0$ then:
  - By the definition of the sub-gradient:

    $$f_0(x) + g^\top (z - x) \leq f_0(z)$$

    So if z satisfies $g^\top \cdot (z - x) > 0$, then $f_0(z) > f_0(x)$. This means that z is not optimal. So for a point z to be optimal (i.e. for $z \in X$) we need:

    $$g^\top (z - x) \leq 0 \tag{2.1}$$

    and $g^\top (z - x) = 0$ for $z = x$. So (2.1) is a neutral cutting-plane at x.

That means that we can remove the half-space $\{z \mid g^\top \cdot (z - x) > 0\}$ from consideration since all points in it have an objective value larger than the point x, and so cannot be optimal. Figure 2.5.

The Problem of Feasibility

Consider the following problem

$$\text{Find } x$$

$$\text{subject to } f_i(x) \leq 0,\ j = 1,2,\ldots,m,$$

where f_i are convex. Here we take the target set X as the feasible set.

To find a cut for this problem at the point x we continue as follows:

- if x is feasible then it satisfies $f_i(x) \leq 0$ for all $i = 1, 2, ..., m$. Then $x \in X$.
- if $x \notin X$ then $\exists j : f_j(x) > 0$. Let $g_j \in \partial f_j(x)$ be a sub-gradient. Since $f_i(z) \geq f_j(x) + g_j^\top (z - x)$, if $f_j(x) + g_j^\top (z - x) > 0$ then $f_j(z) > 0$ and z violates the j-th constraint. That means that any feasible z satisfies

  $$f_j(x) + g^\top (z - x) \leq 0.$$

  This is a deep cut, since $f_j(x) > 0$. Here we remove the half-space $\{z \mid f_j(x) + g^\top (z-x) \geq 0$ because all points that lie in it violate the j-th constraint, as x does, and thus they are not feasible.

Constrained Optimization Problem

By combining the above methods, we can find a cut for the problem:

$$\min f_0(x)$$

subject to

$$f_i(x) \leq 0,\ i = 1, 2, ..., m,$$

where f_j, $j = 0, 1, ..., m$ are convex. Here X is the set of optimal points.

Pick a query point x First we need to check if it is feasible or not.

- if x is infeasible we can produce the following cut:

  $$f_j(x) + g_j^\top (z - x) \leq 0,$$

  where j is the index of the violated constraint and $g_j \in \partial f_j(x)$. This cut is called a *feasibility cut*, since we filter out the half-plane of infeasible points (the ones that violate the j-th constraint).

- if x is feasible, then find $g_0 \in \partial f_0(x)$. If $g_0 = 0$ then x is optimal.

  If $g_0 \neq 0$ we can construct a cutting-plane

  $$g_0^\top (z - x) \leq 0$$

  which is an *objective cut*. The half-space $\{z \mid g_0^\top (z - x) > 0\}$ is put out of consideration, since all such points have an objective value larger than x, and thus are sub-optimal.

Selecting the query point x

The query point x can be chosen in many ways. We would like to exclude as much of the previous polyhedron as possible, with every iteration, therefore $x^{(k+1)}$ should lie near the center of the polyhedron $\mathcal{P}_k$. Some alternatives are listed below. Choose $x^{(k+1)}$ as:

- the center of gravity of $\mathcal{P}_k$
- the center of the largest ball contained in $\mathcal{P}_k$ (Chebyshev center)
- the center of maximum volume ellipsoid contained in $\mathcal{P}_k$ (MVE)
- the analytic center of the inequalities defining $\mathcal{P}_k$ (ACCPM).

2.2.2 Using the Cutting-Plane Method

Our problem seems to have features that make the cutting-plane method very promising. Every policy $\pi \in \Pi$ has a corresponding hyperplane and we can view every hyperplane as a potential cut. Choosing the optimal set X to be the set of points $z \in \mathbb{R}^M$ such that $\overline{\mathcal{V}} \leq z$, the cutting-planes produced will come closer to $\overline{\mathcal{V}}$ with every iteration. In the last iteration we will have a number of cutting-planes very close to $\overline{\mathcal{V}}$ and each one of them corresponds to a policy. We can choose one of them (or combine them) to create a (randomizing) policy which will exhibit close to optimal performance. Ideally a convex combination of the cutting-planes will be touching $\overline{\mathcal{V}}$ in some point. If this optimal policy touches $\overline{\mathcal{V}}$ in the most unfavourable point ($\min_\xi \max_\pi V_\xi^\pi$), then it is a *robust* policy.

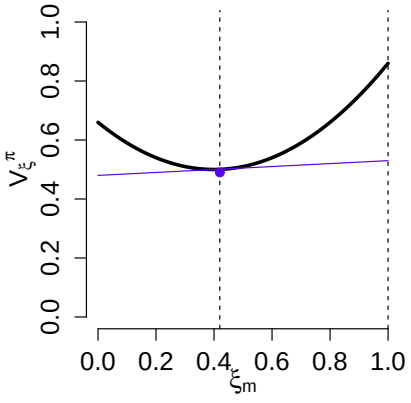

Figure 2.8: A robust policy.

The Algorithm

Algorithm 1 Cutting-plane

We are given an initial polyhedron $\mathcal{P}_o : X \subseteq \mathcal{P}_o$, where X is the (target) set of optimal points.

$k \leftarrow 0$
loop

Query the cutting plane oracle at x_{k+1}
if oracle decides that $x_{k+1} \in X$, **then Quit**
else add the new cutting plane inequality:
$\mathcal{P}_{k+1} \leftarrow \mathcal{P}_k \cap \{z \mid a^T z \leq b\}$
end if

if $\mathcal{P}_k = \emptyset$ **then Quit**
end if

$k \leftarrow k + 1$

end loop

2.3 NP-hardness

There are several issues that need to be addressed in the above algorithm, especially concerning how exactly the oracle works. However, let's skip forward. In the end[1] of the procedure we would have to match the cutting-planes that form the last polyhedron to specific policies. Therefore we arrive in the following decision problem:

Definition 2.3. *(The stochastic-blind-policy problem).* Given a discounted Markov decision problem and a target policy value $V \in \mathbb{R}^M$, is there a mixed policy π that earns $V_\pi \geq V$?

This problem is already addressed by Vlassis et al in [VLB12]. As it turns out, the stochastic-blind-policy problem is NP-hard[2] and hence intractable. This means that since we need to solve this decision problem to complete our optimization, independently on how we arrive at this stage, if we need to match a hyperplane-cut to a policy, then the problem cannot be solved in polynomial time.

[1] It seems reasonable to argue that this decision problem of definition 2.3 needs to be dealt by the oracle, in each iteration, as well. Nevertheless, in the end of the procedure we can't avoid that we will need to match the edges of the polyhedron (the cuts) to specific policies, even if the oracle manages to bypass this issue somehow.

[2] More specifically the stochastic-blind-controller problem is NP-hard, in PSPACE and SQRT-SUM-hard.

The complexity of approximate optimizations for the stochastic-blind-policy problem is still an open question. Only the case of the deterministic controllers is addressed in the related literature (see [LGM01]).

3

A Naive Algorithm

As the Decision Maker takes actions, a stream of rewards is generated. One of the issues here is that these rewards are stochastic: they are random variables that follow some distribution. Thus, in order to evaluate what is to be expected by following each policy, a Monte Carlo sampling can be performed to obtain approximations of the expected value of the policies' total rewards. By performing a minimax optimization, using these approximations, an estimate of the worst case prior distribution ξ can be retrieved. The algorithm laid out here has a major downside though: it is possible that some of the policies do not influence the outcome of the optimization in any way, and therefore the algorithm loses time with approximations that turn out to be useless (see Figure 3.1). We feel that the name *naive* describes this drawback of the algorithm appropriately. More sophisticated approaches to follow in later chapters.[1]

3.1 Uniform Sampling

3.1.1 Notation & Definitions

Before proceeding, we need some definitions.

Let $\mathcal{M}$ be a set of Markov decision problems and let Π be a set of policies, which provide decision rules for each state $s \in \mathcal{S}$.

Definition 3.1. Define the *discounted realized utility* of a policy as the discounted sum of the rewards received at each time step, while in $\mu \in \mathcal{M}$:

$$U_\mu^\pi \triangleq \sum_{1 \leq t \leq \mathcal{T}} \gamma^t \, r^{(t)}_{a \sim \pi, s^{(t)}},$$

[1]For instance, in Chapter 4 the estimation of ξ is approached obliquely by comparing policy performance directly. By following the procedure described, the Decision Maker obtains a robust policy without the need of an explicit calculation of ξ beforehand. In Chapter 5 the policies that perform sub-optimally with high probability, are filtered out and therefore there is no time spent dealing with inefficient policies.

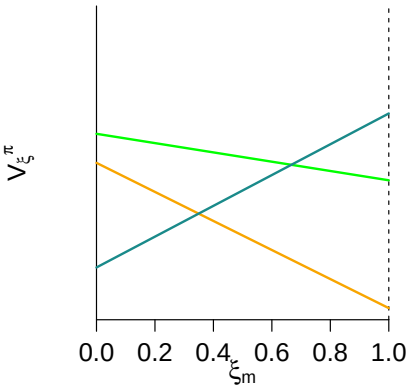

Figure 3.1: Not all policies contribute to the minimax optimization. If all policies are dealt uniformly, a lot of time is spent sampling useless policies.

where the rewards $r^{(t)}_{a\sim\pi,s^{(t)}}$ were generated by the reward function that corresponds to μ, actions a follow the policy plan π and γ represents a discount factor[2] such that $0 \leq \gamma \leq 1$.

For each policy $\pi \in \Pi$, denote the *value of the policy*, while in $\mu \in \mathcal{M}$, as the expected utility obtained from following this policy, as follows:

$$V_\mu^\pi \triangleq \mathbb{E}\left[\, U_\mu^\pi \,\right].$$

We can approximate the true value of each policy by utilizing a Monte Carlo method and so we need the following notation:

Denote the Monte Carlo approximation of policy $\pi \in \Pi$, while in $\mu \in \mathcal{M}$, at the S^{th} iteration as

$$\hat{V}_\mu^{\pi,(S)} \triangleq \frac{1}{S}\sum_{s=1}^{S} U_\mu^{\pi,(S)},$$

where $U_\mu^{\pi,(S)}$ is the discounted realized utility of the policy π (as defined above) at the S^{th} iteration.

Let $e_\mu^{\pi,(S)}$ be the *error* of each Monte Carlo approximation (after S iterations) for the policy π while in μ, ie.

$$e_\mu^{\pi,(S)} \triangleq |\hat{V}_\mu^{\pi,(S)} - V_\mu^\pi|.$$

Let V_π be the $1 \times M$ vector of values for policy π for the given set of Markov decision problems $\mathcal{M}$:

$$V_\pi = (V_{\mu_1}^\pi, V_{\mu_2}^\pi, \ldots, V_{\mu_M}^\pi).$$

[2]If the time horizon $\mathcal{T}$ is not finite then the discount factor γ needs to be strictly less than unity, otherwise the sum of the discounted rewards might explode to infinity. In this section the time horizon will be finite though.

and let V_ξ^π be the weighted mean value of the policy π with respect to the distribution vector $\xi \in [0,1]^{M\times 1}$:

$$V_\xi^\pi \triangleq V_\pi \cdot \xi.$$

Definition 3.2. Let $\mathcal{C}^{(s)}$ be a *confidence set* for the episode s, i.e.

$$\mathcal{C}^{(s)} \triangleq \left\{\pi : |\hat{V}_\mu^{\pi,(S)} - V_\mu^\pi| < \varepsilon \text{ with probability } 1-\delta\right\}, \quad \delta \in (0,1).$$

3.1.2 Approximating the Policy Values with Uniform Sampling

Here we focus on the case when $\mathcal{T} < \infty$ and Π is a set of arbitrary policies. We start by approximating the values of the policies in Π with a Monte Carlo simulation for S iterations. A visualization of an approximated policy value can be seen in Figure 3.2.

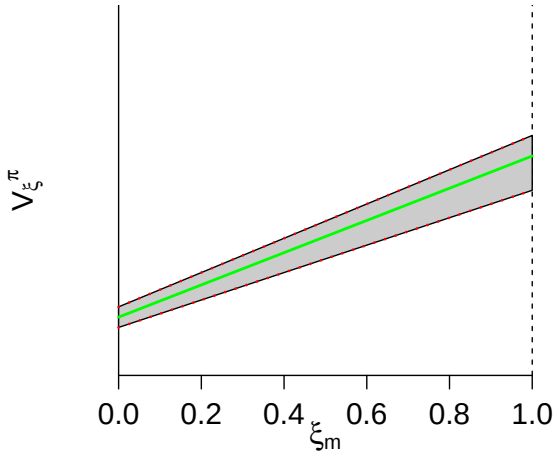

Figure 3.2: The Monte Carlo approximated value of a policy given different values for ξ. The true value of V_μ^π lies somewhere inside the shaded area. The width of the shaded area diminishes with the iterations, since the error becomes smaller.

After obtaining these values, choose a ξ such that

$$\min_\xi \max_\pi V_\xi^\pi$$

for policies $\pi \in \Pi$. This will be a close approximation to the true $\xi^\star$, since $\pi \in \mathcal{C}^{(S)}$ with high probability. We define the confidence sets $\mathcal{C}^{(S)}$ in the next section, after retrieving the relevant error bounds.

A robust policy can be chosen as a π such that maximizes $V_{\hat{\xi}^\star}^\pi$.

3.1.3 The Uniform Sampling Algorithm

Algorithm 2 Uniform Sampling

Parameters: $\delta \in (0,1)$, $\gamma \in (0,1)$,S>0
Inputs : $\mathcal{M}$, Π
For $s = 1,2,\ldots,S$ **do**:

$$U_{\mu}^{\pi,(s)} \leftarrow \sum_{1 \leq t \leq \mathcal{T}} \gamma^t \; r^{(t)}_{a \sim \pi, s^{(t)}} \;\; \forall \mu \in \mathcal{M} \; \forall \pi \in \Pi$$

End For
$\hat{V}_{\mu}^{\pi,(S)} \leftarrow \frac{1}{S} \sum_{s=1}^{S} U_{\mu}^{\pi,(s)}, \;\; \forall \mu \in \mathcal{M} \; \forall \pi \in \Pi$
Set $\hat{\xi}^{\star}$ so that $\min_{\xi} \max_{\pi} V_{\xi}^{\pi}$
Select $\pi \;\; \in \operatorname{argmax} V_{\hat{\xi}^{\star}}^{\pi}$

3.1.4 Analysis

Error bounds

In this section, we retrieve bounds for the errors in order to estimate how close to the true value of $\xi^{\star}$ our estimated $\hat{\xi}^{\star}$ is. We assume that $V_{(\cdot)}^{\pi} \in [0,1] \; \forall \pi$. This condition can be achieved by using appropriate scaling.

Lemma 3.1. *For each policy $\pi \in \Pi$ and each $\mu \in \mathcal{M}$, after $S > 0$ iterations, the estimation of the error is at most ε with probability $1 - \exp\left\{-2\varepsilon^2 S\right\}$, i.e.*

$$\mathbb{P}\left[\; e_{\mu}^{\pi,(S)} \geq \varepsilon \; \right] = e^{-2\varepsilon^2 S}.$$

Proof. We will use the Chernoff-Hoeffding inequalities (See Appendix A).

Let $\varepsilon > 0$.
The probability of the error exceeding ε:

$$\mathbb{P}\left[\; e_{\mu}^{\pi,(S)} \geq \varepsilon \; \right] =$$

$$\mathbb{P}\left[\; |\hat{V}_{\mu}^{\pi,(S)} - V_{\mu}^{\pi}| \geq \varepsilon \; \right] =$$

$$\mathbb{P}\left[\left| \frac{1}{S} \sum_{s=1}^{S} U_{\mu}^{\pi,(s)} - \frac{1}{S} \sum_{s=1}^{S} \mathbb{E}\left[\; U_{\mu}^{\pi,(s)} \; \right] \right| \geq \varepsilon \right] \leq \; \text{(by using Theorem A.18)}$$

$$\exp\left\{ -2 \frac{(\varepsilon S)^2}{S} \right\} \; = \; e^{-2\varepsilon^2 S}. \tag{3.1}$$

□

Now, using the above result, we can define the confidence set for episode s:

$$\mathcal{C}^{(s)} \triangleq \left\{ \pi : |\hat{V}_\mu^{\pi,(s)} - V_\mu^\pi| < \sqrt{\frac{-\log_e \delta}{2s}} \text{ with probability } 1-\delta \right\}, \quad \delta \in (0,1).$$

A number of policies $\pi_i,\ i = 1,2,...,N$ can be combined to create a *mixed policy* π

with weights $w_i \geq 0$, where not all w_i are zero, that is, the Decision Maker assigns a probability

$$p_j = \frac{w_j}{\sum_{i=1}^N w_i}$$

to each pure policy $\pi_j,\ j = 1,2,...,N$ and randomly selects one, using these probabilities.

Then we can use Lemma 3.1 to bound the total error.

Lemma 3.2. *For a mixed policy π the total error $e_\mu^{\pi,(S)}$ is at most ε with probability* $1 - \sum_{j=1}^{N} \exp\left\{-2\varepsilon^2 S\right\}$, *i.e.*

$$\mathbb{P}\left[\ e_\mu^{\pi,(S)} \geq \varepsilon\ \right] \leq \sum_{j=1}^{N} \exp\left\{-2\varepsilon^2 S\right\},$$

where S is the number of Monte-Carlo simulations used to approximate the value of the policy π_j, $\quad \forall j \in \{1,2,\ldots,N\}$.

Proof.
Let $\varepsilon > 0$.
If

$$p_j e_j < p_j \varepsilon \quad \forall j \in \{1,2,\ldots,N\}$$

then by summing up for all j we obtain

$$\sum_{j=1}^{N} p_j e_j < \sum_{j=1}^{N} p_j \varepsilon = \sum_{j=1}^{N} \frac{w_j \varepsilon}{\sum_{i=1}^N w_i} = \varepsilon \frac{\sum_{j=1}^N w_j}{\sum_{i=1}^N w_i} = \varepsilon.$$

Hence, the event $(p_j e_j < p_j \varepsilon\ \forall j \in \{1,2,...,N\})$ implies $\left(\sum_{j=1}^N p_j e_j < \varepsilon\right)$.
Thus, $\left(\sum_{j=1}^N p_j e_j \geq \varepsilon\right)$ implies ($p_j e_j \geq p_j \varepsilon$ for some j). Therefore, the probability of the first is less than the probability of the second event. It follows that

$$\mathbb{P}\left[\ e_\mu^{\pi,(S)} \geq \varepsilon\ \right] = \mathbb{P}\left[\ \sum_{j=1}^{N} p_j e_\mu^{\pi_j,(S)} \geq \varepsilon\ \right] <$$

$$\mathbb{P}\left[\ p_j e_\mu^{\pi_j,(S)} \geq p_j \varepsilon, \text{ for some } j\ \right] =$$

$$\mathbb{P}\left[\ e_\mu^{\pi_j,(S)} \geq \varepsilon, \text{ for some } j\ \right] =$$

$$\mathbb{P}\left[\ \exists j \in \{1,2,...,N\} : e_\mu^{\pi_j,(S)} > \varepsilon\ \right] =$$

$$\mathbb{P}\left[\ \bigcup_{j=1}^{N}\left\{e_\mu^{\pi_j,(S)} > \varepsilon\right\}\ \right] \overset{\text{sub-additivity}}{\leq}$$

$$\sum_{j=1}^{N}\mathbb{P}\left[\ e_\mu^{\pi_j,(S)} \geq \varepsilon\ \right] \overset{\text{Lemma 3.1}}{\leq}$$

$$\sum_{j=1}^{N}\exp\left\{-2\varepsilon^2 S\right\}.$$

□

Similarly, the following Lemma holds.

Lemma 3.3. *For a mixed policy π (with weights $p_j = w_j/\sum_i w_i$) and for a distribution $\xi = (\xi_1, \ldots, \xi_M)$ with $\sum_m \xi_m = 1$, after S Monte Carlo simulations it holds:*

$$\mathbb{P}\left[\ \sum_{m=1}^{M}\sum_{j=1}^{N} e_\mu^{\pi_j,(S)} \geq \varepsilon\ \right] \leq \sum_{m=1}^{M}\sum_{j=1}^{N}\exp\left\{-2\left(\xi_m\frac{w_j\varepsilon}{\sum_{i=1}^{N} w_i}\right)^2 S\right\}.$$

Proof.
Let $\varepsilon > 0$.
The event $\left(e_j < \xi_m\frac{w_j\varepsilon}{\sum_{i=1}^{N} w_i}\ \forall m \in \{1,2,\ldots,M\} \text{ and } \forall j \in \{1,2,...,N\}\right)$ implies $\left(\sum_{m=1}^{M}\sum_{j=1}^{N} e_j < \varepsilon\right)$. Thus $\left(\sum_{m=1}^{M}\sum_{j=1}^{N} e_j \geq \varepsilon\right)$ implies $\left(e_j \geq \xi_m\frac{w_j\varepsilon}{\sum_{i=1}^{N} w_i} \text{ for some } m \text{ and some } j\right)$. Hence the probability of the first is less than the probability of the second event. It follows that

$$\mathbb{P}\left[\ \sum_{j=1}^{N} e_\mu^{\pi_j,(S)} \geq \varepsilon\ \right] <$$

$$\mathbb{P}\left[\ e_\mu^{\pi_j,(S)} \geq \xi_m\frac{w_j\varepsilon}{\sum_{i=1}^{N} w_i}, \text{ for some } m \text{ and some } j\ \right] =$$

$$\mathbb{P}\left[\ \exists m \in \{1,2,...,M\} \text{ and } \exists j \in \{1,2,...,N\} : e_\mu^{\pi_j} > \xi_m\frac{w_j\varepsilon}{\sum_{i=1}^{N} w_i}\ \right] =$$

$$\mathbb{P}\left[\ \bigcup_{m=1}^{M}\bigcup_{j=1}^{N}\left\{e_\mu^{\pi_j} > \xi_m\frac{w_j\varepsilon}{\sum_{i=1}^{N} w_i}\right\}\ \right] \overset{\text{sub-additivity}}{\leq}$$

$$\sum_{m=1}^{M}\sum_{j=1}^{N}\mathbb{P}\left[\, e_{\mu}^{\pi_j} \geq \xi_m \frac{w_j\varepsilon}{\sum_{i=1}^{N} w_i} \,\right] \overset{(3.1)}{\leq}$$

$$\sum_{m=1}^{M}\sum_{j=1}^{N}\exp\left\{-2\left(\xi_m \frac{w_j\varepsilon}{\sum_{i=1}^{N} w_i}\right)^2 S\right\}.$$

□

True vs Sampled ξ

At this point we can retrieve probabilistic bounds on the error of the estimation of $\xi^\star$ if we bound the optimal value function appropriately.

To that end, choose two appropriate quadratics $\overline{V}_\xi$ and $\underline{V}_\xi$ that bound $V_\xi^\star$ from above and below respectively.

To be more exact, define the optimal value function as:

$$V_\xi^\star = \max_\pi V_\xi^\pi,$$

then we can define the upper and lower bounds respectively as:

$$\overline{V}_\xi = u + (\xi - \xi^\star)^\top U(\xi - \xi^\star) \quad \text{and} \quad \underline{V}_\xi = \ell + (\xi - \xi^\star)^\top L(\xi - \xi^\star)$$

for some ℓ,$u \in \mathbb{R}$, $L, U \in R^{M\times M}$, with the norms of the sub-gradients to obey:

$$\|\nabla\underline{V}_\xi\| \leq \|\nabla V_\xi^\star\| \leq \|\nabla\overline{V}_\xi\|. \tag{3.2}$$

Then we can prove the following:

Theorem 3.3. *Let $\varepsilon > 0$ and let $\underline{V}_\xi$, $\overline{V}_\xi$ be two quadratic functions such that*

$$\|\nabla\underline{V}_\xi\| \leq \|\nabla V_\xi^\star\| \leq \|\nabla\overline{V}_\xi\|.$$

Then the error in the estimation of $\xi^\star$ is at most ε with probability $1-\sum_{j=1}^{N}\exp\left\{-2\left(\varepsilon||\nabla\underline{V}_\xi||\right)^2 S\right\}$.

Proof.
Let $\varepsilon > 0$ and let $\underline{V}_\xi$, $\overline{V}_\xi$ be as described above. Then Taylor expansion series together with inequality (3.2) give

$$|V_\xi - V_\xi^\star| \leq \|\nabla\overline{V}_\xi^\star\|\|\xi - \xi^\star\| \text{ and } |V_\xi - V_\xi^\star| \geq \|\nabla\underline{V}_\xi^\star\|\|\xi - \xi^\star\|$$

or

$$\frac{|V_\xi - V_\xi^\star|}{\|\nabla\overline{V}_\xi\|} \leq \|\xi - \xi^\star\| \leq \frac{|V_\xi - V_\xi^\star|}{\|\nabla\underline{V}_\xi\|} \tag{3.3}$$

The right hand side of inequality (3.3) implies that:

$$\text{if} \quad \frac{|V_\xi - V_\xi^\star|}{\|\nabla \underline{V}_\xi\|} < \varepsilon \quad \text{then} \quad \|\xi - \xi^\star\| < \varepsilon.$$

Hence the first event implies the second, and consequently:

$$\mathbb{P}\left[\ \|\xi - \xi^\star\| < \varepsilon\ \right] \geq \mathbb{P}\left[\ |V_\xi - V_\xi^\star| < \varepsilon ||\nabla \underline{V}_\xi||\ \right]$$

which means

$$\mathbb{P}\left[\ \|\xi - \xi^\star\| \geq \varepsilon\ \right] < \mathbb{P}\left[\ |V_\xi - V_\xi^\star| \geq \varepsilon ||\nabla \underline{V}_\xi||\ \right] = \mathbb{P}\left[\ e_\mu^\pi \geq \varepsilon ||\nabla \underline{V}_\xi||\ \right].$$

By using Lemma 3.2, we obtain the result. □

Remark 3.4. Similarly, by using the upper bound (left hand side of ineq.(3.3)) we can bound the probability of the error in V by the error in ξ.

4

The Weighted Majority Algorithm

In some occasions the outcomes of *all* the available actions are revealed *fully* or *partially*, after choosing one of them (e.g. in the stock market, the historical prices of all stocks are available for examination), so the alternatives can be compared by using this information, to assess the degree of mistake of the last decision. In this chapter we consider this case, and leave the alternative case, where only the reward/cost of the decided action can be observed, after executing this particular action, for the next chapter.

Recall that the Decision Maker in her attempt to maximize her total reward, under the uncertainty about her environment, envisions a *set* $\mathcal{M}$ of Markov decision processes, that contains M candidates

$$\mu_i = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle, \quad i = 1, 2, \ \dots \ , M.$$

So, each one of the μ_i's describes a possible environment that she needs to deal with. Now she needs to allocate probabilities to each one of the components of $\mathcal{M}$, so she can estimate the value of each alternative policy that she might consider applying. We denoted this distribution of probabilities by $\xi \in \mathbb{R}^M$.

In order to decide on how to achieve this, a very reasonable way of proceeding would be to start with an initial distribution, choose the policies accordingly, observe the outcomes and modify the weights of each μ_i along the way. However, we can avoid the trouble of computing ξ and focus directly on the evaluation of the policies. The idea is that, if we find a policy that performs as desired, then we don't really care if we are dealing with μ_j or $\mu_i, \ i \neq j$!

The next important idea, considering that we assumed adversarial behaviour for the environment (because we are interested in finding a robust policy) is that we can view our problem as a zero-sum game, where the Decision Maker competes against Nature, by choosing policies.

In this chapter we illustrate how the standard *weighted majority algorithm*(WMA) can be used in fictitious play in order to identify a policy that outperforms the others with high probability. Then we modify the algorithm to obtain a more general version (WMA-PUSR) that fits better the uncertainty we are dealing with. The main idea of the algorithm is that the Decision Maker gives higher weights to policies that perform better and chooses what to do using a probability distribution based on these weights. By observing the outcomes, she decreases the weights of policies that err, over time, in order to arrive in a desired mixed policy that pays-off adequately.

4.0.5 A zero-sum game: Decision Maker VS Nature

Players:	**Decision Maker**	**Nature(Adversary)**
Actions:	π	μ (or a distribution ξ among them)
Reward at round k:	$x(\mu_{(k)},\pi_{(k)})$	$-x(\mu_{(k)},\pi_{(k)})$

In each round k the Decision Maker adopts a policy $\pi_{(k)}$ by using a choice distribution $\mathcal{Q}_{(k)}$. Then, Nature reveals a $\xi_{(k)}$, which is chosen in an adversarial way, *against* the Decision Maker's *choice distribution* $\mathcal{Q}_{(k)}$. The Decision Maker receives a reward that depends on $\pi_{(k)}$ and $\xi_{(k)}$ (or rather on the $\mu_{(k)}$ that was chosen by $\xi_{(k)}$) and Nature receives minus that reward. Both players try to maximize their total reward.

To be more specific on how ξ and π influence the rewards, we proceed with the definitions section.

4.0.6 Notation & Definitions

We use similar notation, as in the previous chapter.

Let $\mathcal{M}$ be a set of Markov decision problems and let Π be a set of policies, which provide decision rules for each state $s \in \mathcal{S}$.

For each policy $\pi \in \Pi$, define the true *value of the policy*, while in $\mu \in \mathcal{M}$, as the expected total reward obtained from following this policy:

$$V_\mu^\pi \triangleq \mathbb{E}\left[\sum_{1\le t\le \mathcal{T}} r^{(t)}_{a\sim\pi,s^{(t)}}\right],$$

where actions a follow the policy π and the rewards $r^{(t)}_{a\sim\pi,s^{(t)}}$ were generated with the reward function that corresponds to the Markov decision problem μ. We assume that the value of each policy lies in $[-1,1]$.

Let V_π be the $1\times M$ vector of values for policy π for the given set of Markov decision problems $\mathcal{M}$:

$$V_\pi = (V^\pi_{\mu_1}, V^\pi_{\mu_2}, \ \dots\ , V^\pi_{\mu_M})$$

and let V_{ξ}^{π} be the weighted mean value of the policy π with respect to the distribution vector $\xi \in [0,1]^{M\times 1}$:

$$V_{\xi}^{\pi} \triangleq V_{\pi} \cdot \xi.$$

Moreover denote by $\mathcal{V}_{\xi}$ the $1 \times N$ vector containing the ξ-weighted mean values for each policy:

$$V_{\xi} = (V_{\xi}^{\pi_1}, V_{\xi}^{\pi_2}, \ \dots \ , V_{\xi}^{\pi_N}).$$

Denote by $\hat{V}_{\xi}^{\pi,(S)}$ the approximated policy value, after S rounds of Monte-Carlo sampling.

Denote by $\mathbb{E}_{\mathcal{Q}}$ the mean:

$$\mathbb{E}_{\mathcal{Q}}\left[\mathcal{V}_{\xi}\right] = \mathcal{V}_{\xi} \cdot \mathcal{Q}$$

where $\mathcal{Q}$ is a $N \times 1$ vector of probabilities that sum up to unity.

Let $\Phi_{(k)} \triangleq \sum_{i=1}^{N} w_{k,i}$ be the *potential function* for step k.

Moreover, denote by $x_{\pi_i,k}$ the *total reward* obtained by following policy π_i, i = 1,2,...,N, in the k-th round. Observe that each $x_{\pi_i,k}$ is a random variable that has an expected value, equal to $V_{\xi}^{\pi_i}$.

Finally, denote by $\mathbf{x}_{(k)}$ the vector of rewards of all policies up to round k:

$$\mathbf{x}_{(k)} = (x_{\pi_1,k}, \ x_{\pi_2,k}, \ \ \dots \ , \ x_{\pi_N,k}).$$

4.0.7 The Weighted Majority Algorithm - The Standard Version

We start by assuming that in each round k the Decision Maker has full access to the information regarding the rewards of the past round. That means that she can observe the outcomes of *all* the actions that were available previously. Furthermore, we assume that all rewards $\boldsymbol{x}_{(k)}$ are not stochastically generated, for the time being. We gradually weaken this assumptions in the next sections.

Algorithm 3 WMA

Input:

- A set of policies Π, with $|\Pi| = N$
- A set of weights $w_{(k)} = \left(w_{i,k}\right)_{i=1}^{N}$, a learning rate $0 < \ell \leq 1/2$.

Initialize: $w_{i,1} = 1$.
For each round k:

1: DM(Decision Maker) normalizes the weights to get a distribution

$$\mathcal{Q}_{(k)} = \frac{w_{(k)}}{\Phi_{(k)}}$$

2: DM selects $\pi_{(k)}$ among $\pi_i,\ i = 1,2,...,N$ according to the distribution $\mathcal{Q}_{(k)}$
3: Adversary chooses $\xi_{(k)} \in \operatorname{argmin}_{\xi_{(k)}} \mathbb{E}_{\mathcal{Q}_{(k)}} \left[\mathcal{V}_{\xi_{(k)}}\right]$
4: DM receives reward $x_{k,\pi_{(k)}}$ and observes x_{k,π_i} for all policies $\pi_i \in \Pi$
5: DM calculates the next set of weights for $i = 1,\ \ldots\ , N$:

$$w_{i,k+1} = (1 + \ell\ x_{k,\pi_i}) w_{i,(k)}$$

Remark 4.1. The Adversary chooses a randomizing distribution among the Markov decision problems $\mathcal{M}$ and not necessarily a specific $\mu \in \mathcal{M}$, which allows for more flexibility in the model. For instance, if there exist two optimal policies with the same worst-case values for the Decision Maker (and thus she issues equal weights to them), then there are three ξ's for the adversary to choose as an optimal response (if the Adversary's action space consists of ξ's), but only two μ's (if the action space consists of Markov decision problems). So the Decision Maker can include, in the way the problem is approached here, more possibilities of what can happen in the future (more adversary actions to be encountered). However, without loss of generality, we can reduce the search space for ξ by restricting the adversarial moves to only deterministic choices. That means that the Adversary can always choose a specific $\mu \in \mathcal{M}$, i.e. a $\xi_{(k)}$ of the form

$$\xi_{(k)} = (0, 0, ..., 0, 1, 0, ..., 0) \tag{4.1}$$

to minimize the Decision Maker's gains.

Indeed:

Proof. (By contradiction).

Fix a policy π and let

$$V_{\xi}^{\pi} < V_{\xi_d}^{\pi}$$

$$\text{for every } \xi_d = (0,\ 0,\ \ldots,\ 0,\ \overset{\substack{d\text{-th}\\ \text{position}}}{\overset{\wedge}{1}}\ ,\ 0,\ \ldots,\ 0), \quad d = 1, \ldots, M,$$

where

- $\xi = (\xi_1, \ldots, \xi_M)$
- at least two of the ξ_m's $(1 \leq m \leq M)$ in ξ are not zero and
- no two Markov decision problems in $\mathcal{M}$ give equal[1] values for policy π.

Then

$$\begin{aligned} V_{\xi}^{\pi} &< V_{\xi_d}^{\pi}, \quad d = 1, \ldots, M \\ V_{\pi} \cdot \xi &< V_{\pi} \cdot \xi_d, \quad d = 1, \ldots, M \\ \sum_{1 \leq m \leq M} \xi_m V_{\mu_m}^{\pi} &< V_{\mu_d}^{\pi}, \quad d = 1, \ldots, M \end{aligned} \tag{4.2}$$

Now, observe that

$$\sum_{1 \leq m \leq M} \xi_m V_{\mu_m}^{\pi} > \sum_{1 \leq m \leq M} \xi_m \min_{1 \leq m \leq M} V_{\mu_m}^{\pi} = \min_{1 \leq m \leq M} V_{\mu_m}^{\pi} \sum_{1 \leq m \leq M} \xi_m = \min_{1 \leq m \leq M} V_{\mu_m}^{\pi} \tag{4.3}$$

therefore

$$\overset{(4.3),(4.2)}{\Longrightarrow} \min_{1 \leq m \leq M} V_{\mu_m}^{\pi} < V_{\mu_d}^{\pi}, \quad d = 1, \ldots, M \tag{4.4}$$

Since equation (4.4) holds for every $d = 1, \ldots, M$, it also holds for the d that minimizes $\mathrm{V}_{\mu_d}^{\pi}$. Thus, by taking minimum over all d's, (4.4) yields

$$\min_{1 \leq m \leq M} V_{\mu_m}^{\pi} < \min_{1 \leq d \leq M} V_{\mu_d}^{\pi}$$

Contradiction. □

Remark 4.2. Observe that in equation (4.3) the inequality is strict, since at least two of the ξ_m's $(1 \leq m \leq M)$ in ξ are not zero and we assumed that every Markov decision problem gives a different reward for this policy. If we allow equal rewards for two different Markov decision problems and it happens that the corresponding μ_m's for these ξ_m's give equal values to $V_{\mu_m}^{\pi}$ then the inequality is not strict, but this is an uninteresting case, since the same policy performs equally well in both situations and so we can view these different Markov decision problems as one (for this particular policy). In any case, the Adversary cannot worsen the Decision Maker's position by randomizing his choices of Markov decision problems.

Another way to arrive to the same conclusion is to use the well known game-theoretic result that if one player knows what action the other player has chosen, then there always exists a deterministic optimal response.

So, in practice, we can reduce the adversarial action space to the ξ's that demonstrate the above form (eq.(4.1)), rather than the much larger one defined by equations (1.1) and (1.2) (page 7).

[1] See also Remark 4.2

In the following section we lay out a performance guarantee for this standard version of the algorithm. Proofs of the two theorems below (in a cost, and not rewards form, though) can be found in [AHK12]. However, in Section 4.0.9 we modify the algorithm to allow for more uncertainty and prove a more general version of these theorems, for a setting where not all policy rewards can be observed, after each round. There we take a closer look on what happens in each iteration and discuss a possible scenario in order to obtain a better understanding of how things work.

4.0.8 Analysis

The expected reward for sampling a policy π from the distribution $\mathcal{Q}_{(k)}$ is

$$\mathbb{E}_{\pi\sim\mathcal{Q}_{(k)}}\left[x_{k,\pi}\right] = \mathbf{x}_{(k)} \cdot \mathcal{Q}_{(k)}.$$

The total expected reward over all rounds is therefore

$$\mathcal{V}_{\mathrm{WMA}}^{(K)} \triangleq \sum_{k=1}^{K} \mathbf{x}_{(k)} \cdot \mathcal{Q}_{(k)}.$$

Theorem 4.3. *Assume that all policy rewards lie in* $[-1,1]$. *Let* $0 < \ell \leq 1/2$. *Then the Multiplicative Weights algorithm guarantees that after* K *rounds, for any policy* π_i, $i = 1,2,...,N$, *it holds:*

$$\mathcal{V}_{WMA}^{(K)} \geq \sum_{k=1}^{K} x_{k,\pi_i} - \ell \sum_{k=1}^{K} |x_{k,\pi_i}| - \frac{\log_e N}{\ell}$$

Theorem 4.4. *The Multiplicative Weights algorithm also guarantees that after* K *rounds, for any distribution* $\mathcal{Q}$ *on the decisions, it holds:*

$$\mathcal{V}_{WMA}^{(K)} \geq \sum_{k=1}^{K} (\boldsymbol{x}_k - \ell|\boldsymbol{x}_k|) \cdot \mathcal{Q} - \frac{\log_e N}{\ell}$$

where $|\boldsymbol{x}_k|$ *is the vector obtained by taking the coordinate-wise absolute value of* $\boldsymbol{x}_k$.

Proofs of the above theorems can be found in [AHK12], but can also be obtained as specific cases of the results of the next section.

4.0.9 The Weighted Majority Algorithm / Unknown Stochastic Rewards Variation with Partial Information

Here we relax the assumption of the non-randomness of the rewards. Moreover, we assume that in each round k, *only* the reward of the chosen action can be observed, but there is some partial information available about the outcomes of the other (previous) alternatives, in the form of the distribution $\xi_{(k)}$. More specifically, the information $\xi_{(k)}$ that is revealed *after* the Decision Maker makes a move, concerns the action that the

adversary chose, but it can be used to estimate the expected values of the rest of the alternative policies. That means that in each round k the Adversary chooses and reveals an unfavorable (for the Decision Maker) distribution $\xi_{(k)}$ (against the choice distribution $\mathcal{Q}_{(k)}$ that the Decision Maker has) and based on that selects a $\mu \in \mathcal{M}$. The Decision Maker can't compare the rewards of each policy directly, since they are not revealed, but approximates their expected value using the $\xi_{(k)}$ and updates the rewards according to these approximations.

We proceed by generalizing the Weighted Majority algorithm and the relevant theorems accordingly.

Algorithm 4 WMA-PUSR

Input:

- A set of policies Π, with $|\Pi| = N$
- A set of weights $w_{(k)} = \left(w_{i,(k)}\right)_{i=1}^{N}$, a learning rate $0 < \ell \leq 1/2$.

Initialize: $w_{i,1} = 1$.
For each round k:

1: DM(Decision Maker) normalizes the weights to get a distribution

$$\mathcal{Q}_{(k)} = \frac{w_{(k)}}{\Phi_{(k)}}$$

2: DM selects policy $\pi_{(k)}$ according to the distribution $\mathcal{Q}_{(k)}$
3: Adversary chooses $\xi_{(k)} \in \operatorname{argmin}_{\xi_{(k)}} \mathbb{E}_{\mathcal{Q}_{(k)}} \left[\mathcal{V}_{\xi_{(k)}}\right]$
4: Adversary *reveals* $\xi_{(k)}$ to the DM
5: DM receives reward $x_{k,\pi_{(k)}}$ and approximates $V_{\xi_{(k)}}^{\pi_i}$ for all policies $\pi_i \in \Pi$
6: DM calculates the next set of weights for $i = 1, \ldots, N$:

$$w_{i,k+1} = (1 + \ell \hat{V}_{\xi_{(k)}}^{\pi_i}) w_{i,(k)},$$

where the approximations $\hat{V}_{\xi_{(k)}}^{\pi_i}$ for $i = 1, \ldots, N$, are obtained by sampling the Markov decision problems indicated by $\xi_{(k)}$.

Now, the exact reward value of each alternative policy in the past round is not known, but the DM calculates an approximation (since $\xi_{(k)}$ is revealed) in each round and compares the policies based on that.

4.0.10 Walk-through

By now, things may seem a bit complicated. To get a better insight of what happens in each iteration, we demonstrate a possible, simple scenario.

Imagine that the Decision Maker in the beginning of round k has the weights w_{k-1} from the previous round. She normalizes them to obtain the distribution $\mathcal{Q}_{(k)}$ which describes the way that the policies are chosen (**step 1** of the algorithm). For instance, assume that there are N available policies $\pi_1, \pi_2, \ldots, \pi_N$. The distribution $\mathcal{Q}_{(k)} = (q_{1,k},\ q_{2,k},\ \ldots\ q_{N,k})$ assigns to the the i-th policy $\pi_{i,k}$ a probability $q_{i,k}$. Naturally, for each k, the $q_{i,k}$'s sum up to unity. The Decision Maker randomizes her action by using $\mathcal{Q}_{(k)}$ and plays a policy $\pi_{(k)}$.

Then, in **step 2**, the Adversary, knowing the values of all the policies, chooses a distribution $\xi_{(k)}$ such that the expected reward for following the randomizing distribution is minimized. Hence $\xi_{(k)}$ is selected to minimize

$$V_{\xi_{(k)}} \cdot \mathcal{Q}_{(k)} = q_{1,k} V^{\pi_1}_{\xi_{(k)}} + \ \ldots\ + q_{N,k} V^{\pi_N}_{\xi_{(k)}}$$

If we restrict ξ's to vectors that look like $\xi_{(k)} = (0, 0, ..., 0, 1, 0, ..., 0)$ (see Remark 4.1, page 32), then the Adversary chooses deterministically one Markov decision problem $\mu_{\star,k}$, rather than randomizing between many μ's. This $\xi_{(k)}$ minimizes the convex combination of the policy values (and thus is a best response), but all comes down to the expected policy value under that specific Markov decision problem $\mu_{\star,k}$. Indeed, by following the definitions of page 30:

$$V_{\xi_{(k)}} \cdot \mathcal{Q}_{(k)} = q_{1,k} V^{\pi_1}_{\xi_{(k)}} + \cdots + q_{N,k} V^{\pi_N}_{\xi_{(k)}} = q_{1,k} V_{\pi_1} \cdot \xi_{(k)} + \cdots + q_{N,k} V_{\pi_N} \cdot \xi_{(k)} =$$

$$\sum_{i=1}^{N} q_{i,k} V_{\pi_i} \cdot \xi_{(k)}$$

$$\sum_{i=1}^{N} q_{i,k} (V^{\pi_i}_{\mu_1}, V^{\pi_i}_{\mu_2},\ \ldots\ , V^{\pi_i}_{\mu_\star}, \ldots,\ V^{\pi_i}_{\mu_M}) \cdot (0,\ ...,\ 0,\ 1,\ 0,\ ...,\ 0)^\top =$$

$$\sum_{i=1}^{N} q_{i,k} V^{\pi_i}_{\mu_{\star,k}} = V_{\mu_{\star,k}} \cdot \mathcal{Q}_{(k)},$$

which is the expected total reward, by following $\mathcal{Q}_{(k)}$ when in $\mu_{\star,k}$. In short, the Adversary chooses the Markov decision problem in which the Decision Maker's choice will perform the worst.

Thereupon, the Adversary shows that $\xi_{(k)}$ to the Decision Maker.

So, the Decision Maker's move in this round of the game is a randomizing policy $\mathcal{Q}_{(k)}$ and the Adversary's move is a distribution $\xi_{(k)}$. The Decision maker plays the mixed policy $\mathcal{Q}_{(k)}$ which results to the policy π_i to be implemented, with probability $q_{i,k}$. The Adversary uses $\xi_{(k)}$ to select the Markov decision problem $\mu_{\star,k}$. That means that the

Decision Maker takes the actions[2] prescribed by this policy π_i in each time step of the Markov decision problem $\mu_\star$ and in the way collects all the rewards $r_{\mu_{\star,k},\pi_{i,k}}$. These make up the total realized reward $x_{k,\pi_{(k)}}$ of the policy.

Therefore, in **step 4**, the Decision Maker receives a reward $x_{k,\pi_{(k)}}$ (which is a random variable with expected value $V_{\mu_{\star,k}}^{\pi_{(k)}}$). Thus, at this point the only new information available to the Decision Maker is:

$$\xi_{(k)} \quad \text{and} \quad x_{k,\pi_{(k)}}.$$

Now the Decision Maker knows which Markov decision problem she was interacting with (or the distribution that was used to choose which one was it, if we don't restrict ξ's to the deterministic choices, according to Remark 4.1), so she needs to compare the performance of all the available pure policies, to see which of them performed better and improve her randomizing rules if needed. To this end, she samples the policy values from the chosen Markov decision problem(s) and updates the weights based on these approximations (**step 5**).

4.0.11 Analysis

Since the information that can be observed in each round does not include the specific policy rewards $x_{k,\pi}$ for all policies π (except from the one that is received), but the Decision Maker is given only $\xi_{(k)}$, and since the algorithm uses $\hat{V}_{\xi_{(k)}}^{\pi,(S)}$'s and not $x_{k,\pi}$'s to update the weights, Theorem 4.3 does not hold anymore. However, we can retrieve similar results and obtain performance guarantees, by generalizing Theorem 4.3 and its proof.

The value earned by using WMA-PUSR, over all rounds is

$$\mathcal{V}_{\text{WMA-PUSR}}^{(K)} \triangleq \mathbb{E}\left[\sum_{k=1}^{K} \mathbf{x}_{(k)} \cdot \mathcal{Q}_{(k)}\right] = \sum_{k=1}^{K} V_{\xi_{(k)}} \cdot \mathcal{Q}_{(k)}$$

where $\mathcal{Q}_{(k)} = (q_{k,1}, \ldots, q_{k,N})$.

First, we prove a lemma for the approximated expected values.

Lemma 4.1. *Assume that all policy rewards lie in* $[-1,1]$. *Let* $0 < \ell \leq \frac{1}{2}$. *Then after* K *rounds, it holds:*

$$V_{WMA\text{-}PUSR}^{(K)} \geq \sum_{k=1}^{K} \hat{V}_{\xi_{(k)}}^{\pi_i} - \ell \sum_{k=1}^{K} |\hat{V}_{\xi_{(k)}}^{\pi_i,(S)}| - \frac{\log_e N}{\ell}$$

for all $i = 1,2,...,N$.

[2]Observe that the *actions* of the zero-sum game and the *actions* during the Markov decision problem are not the same! In this game, an action for the Decision Maker is a mixed policy, whereas during the Markov decision problem, the actions are elements of the set $\mathcal{A}$, as described in the Introduction, Definition 1.1 and Section 1.2, pages 2 and 4.

Proof.

$$\Phi_{(k+1)} =$$

$$\sum_{i=1}^{N} w_{i,(k+1)} =$$

$$\sum_{i=1}^{N} w_{i,(k)} \left(1 + \ell \hat{V}_{\xi_{(k)}}^{\pi_i,(S)}\right) = \quad (\text{since } q_{i,(k)} = \frac{w_{i,(k)}}{\Phi_{(k)}})$$

$$\Phi_{(k)} - \ell \Phi_{(k)} \sum_{i=1}^{N} \hat{V}_{\xi_{(k)}}^{\pi_i,(S)} q_{i,(k)} =$$

$$\Phi_{(k)} \left(1 + \ell \hat{V}_{\xi_{(k)}}^{(S)} \cdot \mathcal{Q}_{(k)}\right) \leq \Phi_{(k)} e^{\ell \hat{V}_{\xi_{(k)}}^{(S)} \cdot \mathcal{Q}_{(k)}} \tag{4.5}$$

where the inequality $1 + x \leq e^x \quad \forall x$ was used.

Therefore, after K rounds, by repeatedly applying inequality (4.5)

$$\Phi_{(k+1)} \leq \Phi_{(k)} \exp\left(\ell \hat{V}_{\xi_{(k)}}^{(S)} \cdot \mathcal{Q}_{(k)}\right) \leq$$

$$\left(\Phi_{K-1} \exp\left(\ell \hat{V}_{\xi_{(K-1)}}^{(S)} \cdot \mathcal{Q}_{(K-1)}\right)\right) \exp\left(\ell \hat{V}_{\xi_{(k)}}^{(S)} \cdot \mathcal{Q}_{(k)}\right) \leq \dots$$

$$\cdots \leq \Phi_{(1)} \exp\left\{\ell \sum_{k=1}^{K} \hat{V}_{\xi_{(k)}}^{(S)} \cdot \mathcal{Q}_{(k)}\right\} =$$

$$N \exp\left\{\ell \sum_{k=1}^{K} \hat{V}_{\xi_{(k)}}^{(S)} \cdot \mathcal{Q}_{(k)}\right\}, \tag{4.6}$$

since $\Phi_{(1)} = \sum_{i=1}^{N} w_{i,1} = \sum_{i=1}^{N} 1 = N$

Now, by using Bernoulli's inequality:

$$(1 + \ell)^x \leq (1 + \ell x) \text{ for } x \in [0,1]$$

and

$$(1 - \ell)^{-x} \leq (1 - \ell x) \text{ for } x \in [-1,0],$$

since $\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} \in [-1,1]$, we have:

$$\Phi_{(k+1)} \geq w_{i,(k+1)} =$$

$$w_{i,(k)} \left(1 + \ell \hat{V}_{\xi_{(k)}}^{\pi_i,(S)}\right) =$$

$$\left(w_{i,K-1} \left(1 + \ell \hat{V}_{\xi_{(K-1)},S}^{\pi_i}\right)\right) \left(1 + \ell \hat{V}_{\xi_{(k)}}^{\pi_i,(S)}\right) = \dots$$

$$\cdots = \prod_{k=1}^{K} \left(1 + \ell \hat{V}_{\xi_{(k)}}^{\pi_i,(S)}\right) \geq (1+\ell)^A \cdot (1-\ell)^{-B} \tag{4.7}$$

where $A = \sum_{\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} \geq 0} \hat{V}_{\xi_{(k)}}^{\pi_i,(S)}$ and $B = \sum_{\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} < 0} \hat{V}_{\xi_{(k)}}^{\pi_i,(S)}$.

Combining (4.6) and (4.7)

$$N \exp\left\{\ell \sum_{k=1}^{K} \hat{V}_{\xi_{(k)}} \cdot \mathcal{Q}_{(k)}\right\} \geq (1+\ell)^A \cdot (1-\ell)^{-B}$$

Taking logarithms (and substituting A for $\sum_{\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} \geq 0} \hat{V}_{\xi_{(k)}}^{\pi_i,(S)}$ and B for $\sum_{\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} < 0} \hat{V}_{\xi_{(k)}}^{\pi_i,(S)}$) we obtain

$$\log N + \ell \sum_{k=1}^{K} \hat{V}_{\xi_{(k)}}^{(S)} \cdot \mathcal{Q}_{(k)} \geq \sum_{\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} \geq 0} \hat{V}_{\xi_{(k)}}^{\pi_i,(S)} \log(1+\ell) - \sum_{\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} < 0} \hat{V}_{\xi_{(k)}}^{\pi_i,(S)} \log(1-\ell),$$

or, by re-arranging and dividing by ℓ:

$$\sum_{k=1}^{K} \hat{V}_{\xi_{(k)}}^{(S)} \cdot \mathcal{Q}_{(k)} \geq$$

$$\frac{1}{\ell} \sum_{\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} \geq 0} \hat{V}_{\xi_{(k)}}^{\pi_i,(S)} \log(1+\ell) - \frac{1}{\ell} \sum_{\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} < 0} \hat{V}_{\xi_{(k)}}^{\pi_i,(S)} \log(1-\ell) - \frac{\log N}{\ell} \xrightarrow[-\log(1-\ell) = \log\left(\frac{1}{1-\ell}\right)]{\text{Since}}$$

$$= \frac{1}{\ell} \sum_{\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} \geq 0} \hat{V}_{\xi_{(k)}}^{\pi_i,(S)} \log(1+\ell) + \frac{1}{\ell} \sum_{\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} < 0} \hat{V}_{\xi_{(k)}}^{\pi_i,(S)} \log\left(\frac{1}{1-\ell}\right) - \frac{\log N}{\ell} \overset{(4.8)}{\geq}$$

$$\frac{1}{\ell} \sum_{\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} \geq 0} \hat{V}_{\xi_{(k)}}^{\pi_i,(S)} \left(\ell - \ell^2\right) + \frac{1}{\ell} \sum_{\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} < 0} \hat{V}_{\xi_{(k)}}^{\pi_i,(S)} (\ell + \ell^2) - \frac{\log N}{\ell} =$$

$$\sum_{k=1}^{K} \hat{V}_{\xi_{(k)}}^{\pi_i,(S)} - \ell \sum_{\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} \geq 0} \hat{V}_{\xi_{(k)}}^{\pi_i,(S)} + \ell \sum_{\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} < 0} \hat{V}_{\xi_{(k)}}^{\pi_i,(S)} - \frac{\log N}{\ell} =$$

$$\sum_{k=1}^{K} \hat{V}_{\xi_{(k)}}^{\pi_i,(S)} - \ell \sum_{k=1}^{K} |\hat{V}_{\xi_{(k)}}^{\pi_i,(S)}| - \frac{\log N}{\ell},$$

where we used that for $\ell \leq \frac{1}{2}$:

$$\log(1+\ell) \geq \ell - \ell^2 \text{ and } \log\left(\frac{1}{1-\ell}\right) \leq \ell + \ell^2. \tag{4.8}$$

□

Observe that if the rewards are not stochastic, then Lemma 4.1 is reduced to Theorem 4.3.

Transitioning from the approximations $\hat{V}_{\xi_{(k)}}^{\pi_i,(S)}$'s to the true values $V_{\xi_{(k)}}^{\pi_i}$ an error term $\mathcal{E}_{(k)}^{(S)}$ (which depends on the number S of Monte Carlo iterations) needs to be introduced in each round k. We bound each error term with the following theorem.

Theorem 4.5. (MAIN) *Assume that all policy rewards lie in* $[-1,1]$. *Let* $0 < \ell \leq \frac{1}{2}$. *Let* $\varepsilon > 0$. *Then after* K *rounds, for the total expected rewards, it holds:*

$$V_{WMA\text{-}PUSR}^{(K)} \geq \sum_{k=1}^{K} V_{\xi_{(k)}}^{\pi_i} - \ell \sum_{k=1}^{K} |V_{\xi_{(k)}}^{\pi_i}| - \frac{\log_e N}{\ell} - \sum_{k=1}^{K} \mathcal{E}_{(k)}^{(S)} \tag{4.9}$$

for all $i = 1,2,...,N$, *where* $\xi_{(k)} = (\xi_{1,k}, \ldots, \xi_{M,k})$, $\sum_m \xi_{m,k} = 1$, $\mathcal{Q}_{(k)} = (q_{1,k}, \ldots, q_{N,k})$, $\sum_i q_{i,k} = 1$, $\mathcal{E}_{(k)}^{(S)}$ *is the error term of the* k*-th round (and* S *denotes the number of Monte Carlo simulations during the sampling process):*

$$\mathcal{E}_{(k)}^{(S)} = \left| V_{\xi_{(k)}} \cdot \mathcal{Q}_{(k)} - V_{\xi_{(k)}}^{\pi_i} + \ell |V_{\xi_{(k)}}^{\pi_i}| + \frac{\log_e N}{\ell} - \left(\hat{V}_{\xi_{(k)}}^{(S)} \cdot \mathcal{Q}_{(k)} - \hat{V}_{\xi_{(k)}}^{\pi_i,(S)} + \ell |\hat{V}_{\xi_{(k)}}^{\pi_i,(S)}| + \frac{\log_e N}{\ell} \right) \right|$$

and

$$\mathbb{P}\left[\mathcal{E}_{(k)}^{(S)} < \varepsilon \right] \geq 1 - \sum_{j=1}^{N} \sum_{m=1}^{M} \exp\left\{ -\frac{1}{2} \left(\xi_{m,z} \frac{w_{j,z}\varepsilon}{(1+\ell)\Phi_z} \right)^2 S \right\}.$$

where $z = 1,2,...,k$.

Proof. For each round k and every policy π_i, $i \in \{1,2,\ldots,N\}$, the error:

$$\mathcal{E}_{(k)}^{(S)} =$$

$$\left| V_{\xi_{(k)}} \cdot \mathcal{Q}_{(k)} - V_{\xi_{(k)}}^{\pi_i} + \ell |V_{\xi_{(k)}}^{\pi_i}| + \frac{\log_e N}{\ell} - \left(\hat{V}_{\xi_{(k)}}^{(S)} \cdot \mathcal{Q}_{(k)} - \hat{V}_{\xi_{(k)}}^{\pi_i,(S)} + \ell |\hat{V}_{\xi_{(k)}}^{\pi_i,(S)}| + \frac{\log_e N}{\ell} \right) \right| =$$

$$\left| V_{\xi_{(k)}} \cdot \mathcal{Q}_{(k)} - \hat{V}_{\xi_{(k)}}^{(S)} \cdot \mathcal{Q}_{(k)} - V_{\xi_{(k)}}^{\pi_i} + \hat{V}_{\xi_{(k)}}^{\pi_i,(S)} + \ell |V_{\xi_{(k)}}^{\pi_i}| - \ell |\hat{V}_{\xi_{(k)}}^{\pi_i,(S)}| + \cancel{\frac{\log_e N}{\ell}} - \cancel{\frac{\log_e N}{\ell}} \right| =$$

$$\left| \left(V_{\xi_{(k)}} - \hat{V}_{\xi_{(k)}}^{(S)} \right) \cdot \mathcal{Q}_{(k)} + \left(\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} - V_{\xi_{(k)}}^{\pi_i} \right) + \ell \left(|V_{\xi_{(k)}}^{\pi_i}| - |\hat{V}_{\xi_{(k)}}^{\pi_i,(S)}| \right) \right| =$$

$$\left| \sum_{j=1}^{N} \left(V_{\xi_{(k)}}^{\pi_j} - \hat{V}_{\xi_{(k)}}^{\pi_j,(S)} \right) q_{k,j} + \left(\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} - V_{\xi_{(k)}}^{\pi_i} \right) + \ell \left(|V_{\xi_{(k)}}^{\pi_i}| - |\hat{V}_{\xi_{(k)}}^{\pi_i,(S)}| \right) \right|.$$

Below, where inequalities are involved, we use the monotonicity of the measure:

if the event A implies the event B

then

$$\mathbb{P}\left[\ A\ \right] \leq \mathbb{P}\left[\ B\ \right]$$

and so if α, $\beta \in \mathbb{R}$ such that $\alpha \leq \beta$ then

$$\mathbb{P}\left[\ \alpha \geq \varepsilon\ \right] \leq \mathbb{P}\left[\ \beta \geq \varepsilon\ \right],$$

since $\alpha \geq \varepsilon$ implies $\beta \geq \varepsilon$.

$$\mathbb{P}\left[\ \mathcal{E}_{(k)}^{(S)} \geq \varepsilon\ \right] =$$

$$\mathbb{P}\left[\ \left|\sum_{j=1}^{N}\left(V_{\xi_{(k)}}^{\pi_j} - \hat{V}_{\xi_{(k)}}^{\pi_j,(S)}\right) q_{k,j} + \left(\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} - V_{\xi_{(k)}}^{\pi_i}\right) + \ell\left(|V_{\xi_{(k)}}^{\pi_i}| - |\hat{V}_{\xi_{(k)}}^{\pi_i,(S)}|\right)\right| \geq \varepsilon\ \right] \leq \ \text{(triangle inequality)}$$

$$\mathbb{P}\left[\ \sum_{j=1}^{N}\left|V_{\xi_{(k)}}^{\pi_j} - \hat{V}_{\xi_{(k)}}^{\pi_j,(S)}\right| q_{k,j} + \left|\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} - V_{\xi_{(k)}}^{\pi_i}\right| + \ell\left||V_{\xi_{(k)}}^{\pi_i}| - |\hat{V}_{\xi_{(k)}}^{\pi_i,(S)}|\right| \geq \varepsilon\ \right] \leq \ \text{(since } q_{k,j} \leq 1\text{)}$$

$$\mathbb{P}\left[\ \sum_{j=1}^{N}\left|V_{\xi_{(k)}}^{\pi_j} - \hat{V}_{\xi_{(k)}}^{\pi_j,(S)}\right| + \left|\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} - V_{\xi_{(k)}}^{\pi_i}\right| + \ell\left||V_{\xi_{(k)}}^{\pi_i}| - |\hat{V}_{\xi_{(k)}}^{\pi_i,(S)}|\right| \geq \varepsilon\ \right] \leq \ \text{(reverse triangle inequality)}$$

$$\mathbb{P}\left[\ \sum_{j=1}^{N}\left|V_{\xi_{(k)}}^{\pi_j} - \hat{V}_{\xi_{(k)}}^{\pi_j,(S)}\right| + \left|\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} - V_{\xi_{(k)}}^{\pi_i}\right| + \ell\left|V_{\xi_{(k)}}^{\pi_i} - \hat{V}_{\xi_{(k)}}^{\pi_i,(S)}\right| \geq \varepsilon\ \right] =$$

$$\mathbb{P}\left[\ \sum_{j=1}^{N}\left|V_{\xi_{(k)}}^{\pi_j} - \hat{V}_{\xi_{(k)}}^{\pi_j,(S)}\right| + \left|\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} - V_{\xi_{(k)}}^{\pi_i}\right| (1+\ell) \geq \varepsilon\ \right]$$

and since

$$\sum_{j=1}^{N}\left|V_{\xi_{(k)}}^{\pi_j} - \hat{V}_{\xi_{(k)}}^{\pi_j,(S)}\right| + \left|\hat{V}_{\xi_{(k)}}^{\pi_i,(S)} - V_{\xi_{(k)}}^{\pi_i}\right| \leq 2\sum_{j=1}^{N}\left|V_{\xi_{(k)}}^{\pi_j} - \hat{V}_{\xi_{(k)}}^{\pi_j,(S)}\right|$$

the above probability is less or equal to

$$\mathbb{P}\left[\ 2\,(1+\ell)\sum_{j=1}^{N}\left|V_{\xi_{(k)}}^{\pi_j} - \hat{V}_{\xi_{(k)}}^{\pi_j,(S)}\right| \geq \varepsilon\ \right] =$$

$$\mathbb{P}\left[\ \sum_{j=1}^{N}\left|V_{\xi_{(k)}}^{\pi_j} - \hat{V}_{\xi_{(k)}}^{\pi_j,(S)}\right| \geq \frac{\varepsilon}{2\,(1+\ell)}\ \right] =$$

$$\mathbb{P}\left[\ \sum_{j=1}^{N}\sum_{m=1}^{M}\left|V_{\mu_m}^{\pi_j} - \hat{V}_{\mu_m}^{\pi_j,(S)}\right| \xi_{m,k} \geq \frac{\varepsilon}{2\,(1+\ell)}\ \right] \leq \ \text{(since } \xi_{m,k} \leq 1\text{)}$$

$$\mathbb{P}\left[\sum_{j=1}^{N}\sum_{m=1}^{M}\left|V_{\mu_m}^{\pi_j}-\hat{V}_{\mu_m}^{\pi_j,(S)}\right|\geq\frac{\varepsilon}{2\,(1+\ell)}\right]=$$

$$\mathbb{P}\left[\sum_{j=1}^{N}\sum_{m=1}^{M}e_{\mu_m}^{\pi_j,(S)}\geq\frac{\varepsilon}{2\,(1+\ell)}\right]\leq\big(\text{Lemma 3.3 (page 26)}\big)$$

$$\sum_{j=1}^{N}\sum_{m=1}^{M}\exp\left\{-2\left(\xi_{m,z}\frac{w_{j,z}\varepsilon}{2(1+\ell)\Phi_z}\right)^2 S\right\},$$

with z$\in\{1,2\ldots,k\}$ $\square$

The error bound can be further improved by applying Azuma's Lemma, since the difference of the true minus the approximated values possess the martingale property:

$$\mathbb{P}\left[\sum_{1\leq k\leq K}\mathcal{E}_{(k)}^{(S)}<\varepsilon\right]\geq 1-2\exp\{-k\epsilon^2/2\}.$$

We can also obtain a result for a distribution $\mathcal{P}$ over π_i's, $i = 1,2,...,N$.

Corollary 4.6. After K rounds, for any distribution $\mathcal{P}\in R^{N\times 1}$ on the decisions, it holds:

$$\sum_{k=1}^{K}V_{\xi_{(k)}}\cdot\mathcal{Q}_{(k)}\geq\sum_{k=1}^{K}\Big(V_{\xi_{(k)}}-\ell|V_{\xi_{(k)}}|\Big)\cdot\mathcal{P}-\frac{\log_e N}{\ell}-\sum_{k=1}^{K}\mathcal{E}_{(k)}^{(S)}$$

where $|V_{\xi_{(k)}}|$ is the vector obtained by taking the coordinate-wise absolute value of $V_{\xi_{(k)}}$.

Proof. This result follows from Theorem 4.5, by taking convex combinations of the inequalities over all decisions π, with any distribution $\mathcal{P}$:

Let $\mathcal{P}$ be an arbitrary distribution on the decisions $\pi\in\Pi$, that is

$$\mathcal{P}=(p_1,\ p_2,\ \ldots\ ,p_N)\ \text{ with }\ \sum_{i=1}^{N}p_i=1.$$

For every $i = 1,2,\ \ldots,\ N$ multiply the inequality (4.9) with p_i and sum them up. We obtain:

$$\sum_{i=1}^{N}p_i\sum_{k=1}^{K}V_{\xi_{(k)}}\cdot\mathcal{Q}_{(k)}\geq\sum_{i=1}^{N}p_i\left(\sum_{k=1}^{K}V_{\xi_{(k)}}^{\pi_i}-\ell\sum_{k=1}^{K}|V_{\xi_{(k)}}^{\pi_i}|-\frac{\log_e N}{\ell}-\sum_{k=1}^{K}\mathcal{E}_{(k)}^{(S)}\right)$$

$$=\sum_{k=1}^{K}\Big(V_{\xi_{(k)}}-\ell|V_{\xi_{(k)}}|\Big)\cdot\mathcal{P}-\frac{\log_e N}{\ell}-\sum_{k=1}^{K}\mathcal{E}_{(k)}^{(S)}.$$

$\square$

Definition 4.7. The regret of the learning algorithm against the optimal distribution $\mathcal{P}^\star \in \text{argmax}_P\{V_{\xi_{(k)}} \cdot \mathcal{P}\}$ is

$$B(K) = \sum_{k=1}^{K} V_{\xi_{(k)}} \cdot \mathcal{P}^\star - \sum_{k=1}^{K} V_{\xi_{(k)}} \cdot \mathcal{Q}_{(k)}$$

Corollary 4.6 can be used in order to bound the regret. To that end, we first prove the following.

Theorem 4.8. *After K rounds of applying the modified weighted majority algorithm WMA-PUSR, for any distribution $\mathcal{P}$ it holds:*

$$\sum_{k=1}^{K} V_{\xi_{(k)}} \cdot \mathcal{P} - \sum_{k=1}^{K} V_{\xi_{(k)}} \cdot \mathcal{Q}_{(k)} \leq 2\sqrt{\log_e NK} + \sum_{k=1}^{K} \mathcal{E}_{(k)}^{(S)}$$

Proof. In what follows $|V_{\xi_{(k)}}|$ is the vector obtained by taking the coordinate-wise absolute value of $V_{\xi_{(k)}}$.

$$\sum_{k=1}^{K} V_{\xi_{(k)}} \cdot \mathcal{P} - \sum_{k=1}^{K} V_{\xi_{(k)}} \cdot \mathcal{Q}_{(k)} \leq \text{ (by re-arranging Corollary 4.6)}$$

$$\ell \sum_{k=1}^{K} |V_{\xi_{(k)}}| \cdot \mathcal{P} + \frac{log_e N}{\ell} + \sum_{k=1}^{K} \mathcal{E}_{(k)}^{(S)} \leq \text{ (Since } \sum_{k=1}^{K} |V_{\xi_{(k)}}| \cdot \mathcal{P} \leq K)$$

$$\ell K + \frac{log_e N}{\ell} + \sum_{k=1}^{K} \mathcal{E}_{(k)}^{(S)}$$

Substituting $\ell = \sqrt{\frac{\log_e N}{K}}$ we obtain

$$\sum_{k=1}^{K} V_{\xi_{(k)}} \cdot \mathcal{P} - \sum_{k=1}^{K} V_{\xi_{(k)}} \cdot \mathcal{Q}_{(k)} \leq \sqrt{\frac{\log_e N}{K}} K + \frac{log_e N}{\sqrt{\frac{\log_e N}{K}}} + \sum_{k=1}^{K} \mathcal{E}_{(k)}^{(S)} =$$

$$\sqrt{\log_e NK} + \frac{\sqrt{log_e NK}}{K} + \sum_{k=1}^{K} \mathcal{E}_{(k)}^{(S)} = \sqrt{\log_e NK}\left(1 + \frac{1}{K}\right) \leq 2\sqrt{\log_e NK} + \sum_{k=1}^{K} \mathcal{E}_{(k)}^{(S)}.$$

□

Corollary 4.9. When algorithm WMA-PUSR is run with parameter $\ell = \sqrt{\frac{\log_e N}{K}}$ then the regret of the algorithm is bound by

$$B(K) \leq 2\sqrt{\log_e NK} + \sum_{k=1}^{K} \mathcal{E}_{(k)}^{(S)}.$$

Proof. Since Theorem 4.8 holds for every distribution $\mathcal{P}$, it holds for $\mathcal{P}^\star$ as well, and thus

$$B(K) = \sum_{k=1}^{K} V_{\xi_{(k)}} \cdot \mathcal{P}^\star - \sum_{k=1}^{K} V_{\xi_{(k)}} \cdot \mathcal{Q}_{(k)} \leq 2\sqrt{\log_e NK} + \sum_{k=1}^{K} \mathcal{E}_{(k)}^{(S)}.$$

□

5

Contextual Bandits

In the previous chapter we presumed that (some) information about all the previously feasible alternatives becomes available after each round. Here, we relax this assumption and we view our problem as a *contextual bandits* problem. In this setting only the outcome of the selected action is revealed, making the assessment of the efficiency of each decision more obscure.

5.1 Bandits

5.1.1 The Multi-Armed Bandit Problem

First let us describe the standard multi-armed bandit problem. We encounter the opportunity to play a row of slot-machines - also known as *one-armed bandits*, because of their design: they were originally build with a lever attached on their side (*arm*) that triggered the play and for their capacity to empty the players' wallets (*bandit*). A reward is produced from each machine every time its lever is pulled. This reward follows a probability distribution, that corresponds to each machine. After every play, we only observe the reward of the chosen machine only, so we don't have full information about all the machines. Assuming that some machines pay more than others and based on the information we acquire by testing the machines for their rewards, we try to decide how to play, in order to maximize our total profit, generated after a sequence of plays.

5.1.2 The Contextual Multi-Armed Bandit Problem

The main difference in this variation of the problem is that some contextual side-information about the machines is available prior to the decision.

The multi-armed bandits model finds application in research project management (e.g. pharmaceutical companies), where an allocation of recources among competing projects must be done, under a fixed budget. The features of each project are not

explicitly known, but (hopefully since they are research projects) they may become better understood in the near future. Furthermore, the contextual bandits model has been extensively used for targeted advertising, given the side information generated by search engines.

It's useful to think this context information as a multidimensional vector that contains some clues for each machine. To fit this model to our problem we use a Monte Carlo estimation of the worst case prior distribution ξ as side-information. The similarities between the two problems seem auspicious. Observe Table 5.1.

Contextual Bandits	**Our problem**
machines	policies
context information	an estimate of ξ
stochastic rewards for each machine	stochastic policy values for each policy
different distribution of rewards for each machine	different distribution of policy values for each policy
we observe one reward in every round	we observe one policy value in every round

Table 5.1: Comparison between the contextual bandits problem and the problem of finding a robust policy for a set of Markov decision problems

5.2 Algorithms LinRel & SupLinRel

If we do not want to make the assumption that $V_\xi^\star$ is bounded by quadratics, as in Chapter 3, or to use the uniform sampling algorithm and loose time involving suboptimal policies, then we may proceed differently. We utilize the algorithms `LinRel` and `SupLinRel`, often used in contextual bandit problems, from [Aue02]. The algorithm makes use of a feature vector of side information in order to create upper confidence bounds for the expected reward of each policy. The side information we use here is an estimate of $\xi^\star$, chosen by carrying out a minimax over confidence bounds assigned to each policy. That means that the extracted policy will be a minimax policy over the upper confidence bounds. The algorithms deal with the trade-off between exploration and exploitation (ie. investigating for a balance between testing the environment to find high profit actions while taking the empirically best action as often as possible) by calculating these confidence bounds for the expected reward of each policy. Then the policy with the greatest upper confidence bound is chosen. An important thing to note in the algorithm is that, in our case, the feature vector $\xi_{i,(k)}$ is the same for every policy, but the weight vector V_i is different for every i. For that reason we need to slightly modify the involved performance theorem, to fit our case, and we will obtain a lightly worse bound on the loss/regret, by a constant (the number of policies).

The reason that we might need to do some exploration (that is to test alternatives for which we are still uncertain about their outcome), is that in each round, only the chosen policy's outcome is revealed. We need to make sure we have enough information about the policies' rewards in order to exploit them efficiently.

To give a sneak peak, the way that the reinforce learning algorithm `SupLinRel` operates is the following. The feature vectors $\xi_{(k)}$ are filtered until further exploration is needed or until the confidence is small enough, in each stage k. A policy π is allowed to pass to the next stage, only if it is adequately close to the optimal choice, with high probability. Policies that are not optimal are identified by comparing the width that is assigned to them. By eliminating choices which are obviously bad, the possible loss is reduced in the next stage.

5.2.1 Associative reinforcement learning with linear value functions

The setting

Denote by x_i the *total reward* of policy π_i. Recall that $V_{\mu_j}^{\pi_i}$ is the *value* of policy $\pi_i \in \Pi$, $i = 1,2,...,N$ under $\mu_j \in \mathcal{M}$, $j = 1,2,...,M$, where $N = |\Pi|$ and $M = |\mathcal{M}|$.

For every policy π_i consider the vector of values

$$V_i = (V_{\mu_1}^{\pi_i}, V_{\mu_2}^{\pi_i}, \ldots, V_{\mu_M}^{\pi_i}).$$

So every element of V_i corresponds to the value of a certain policy π_i under a different MDP μ_j. Also recall, once more, that the Decision Maker's belief of which decision problem she is interacting with, at stage k, is described by the vector of probabilities $\xi_{(k)} \in \mathbb{R}^M$, which assigns a probability to each $\mu_j \in \mathcal{M}$.

At each time point k, the learning algorithm utilizes the *feature* vector $\xi_{(k)}$ [1]. As in [Aue02] we assume that all rewards x_i are independent random variables with expectation:

$$\mathbb{E}[\, x_i \,] = V_i^\top \xi_{(k)}, \qquad i = 1,2,\ldots,N,$$

where V_i is a vector in $\mathbb{R}^M$ (which the reinforcement learning algorithm needs to learn and approximate) and $\xi_{(k)}$ is the estimated worst case prior at time t. Furthermore we assume that $||V_i|| \leq 1$ (and of course $||\xi_{(k)}|| \leq 1$). Appropriate scaling may be required to achieve this condition.

We apply the learning algorithms from [Aue02] in order to calculate upper confidence bounds for the expected reward $\mathbb{E}[\, x_i \,] = V_i^\top \xi_{(k)}$ of each policy. The algorithms handles the trade off between *exploitation* (controlled by the estimation of the expectation) and *exploration* (controlled by the width of the confidence interval) by calculating these upper confidence bounds for $\mathbb{E}[\, x_i \,] = V_i^\top \xi_{(k)}$. Then the policy with the greatest upper confidence bound is chosen.

[1] In [Aue02] ξ is referred as a *feature vector* z_i and V_i is a *weight vector* f, the same for every policy. In our case ξ is the same for every policy π_i and V_i are different.

Choosing the Feature Vector

The only thing left, is to provide a feature vector of side-information to the algorithm at each stage k. We select $\xi_{(k)}$ by performing a minimax according to the current confidence bounds.

5.2.2 The Algorithms

Algorithm 5 LinRel

Parameters: $\delta \in [0,1]$, the number of trials K

Inputs: The indexes of chosen feature vectors, $\Psi_{(k)} \subseteq \{1,2,\ldots,t-1\}$, the new feature vectors $\xi_{1,(k)},\ldots,\xi_{N,(k)}$.

1: Let $\Xi(m) = (\boldsymbol{\xi}_{i_{(\tau)},(\tau)})_{\tau \in \Psi_{(k)}}$ be the matrix of the selected feature vectors and $\mathbf{x}(m) = (x_{i_{(\tau)},(\tau)})_{\tau \in \Psi_{(k)}}$ the vector of corresponding rewards.

2: Calculate the eigenvalue decomposition, $i = 1,\ldots,N$,

$$\Xi_{(k)} \cdot \Xi_{(k)}^{\top} = U_{(k)}^{\top} \cdot \Delta(\lambda_{1,(k)},\ldots,\lambda_{d,(k)}) \cdot U_{(k)}$$

where $\lambda_{1,(k)},\ldots,\lambda_{n,(k)} \geq 1, \lambda_{n+1,(k)},\ldots,\lambda_{d,(k)} < 1$, and $U_{(k)}^{\top} \cdot U_{(k)} = \Delta(1,\ldots,1)$.

3: For each feature vector $\xi_{i,(k)}$ set $\tilde{\boldsymbol{\xi}}_{i,(k)} = (\tilde{\xi}_{i,1,(k)},\ldots,\tilde{\xi}_{i,d,(k)}) = U_{(k)} \cdot \boldsymbol{\xi}_{i,(k)}$ and $\tilde{\mathbf{u}}_{i,(k)} = (\tilde{\xi}_{i,1,(k)},\ldots,\tilde{\xi}_{i,k,(k)},0,\ldots)^{\top}, \tilde{\mathbf{v}}_{i,(k)} = (0,\ldots,0,\tilde{\xi}_{i,k+1,(k)},\ldots,\tilde{\xi}_{i,d,(k)})^{\top}$.

4: Calculate the upper confidence bounds and their widths

$$\mathrm{ucb}_i(k) = x(k) \cdot a_i(k)^{\top} + \mathrm{width}_i(k).$$

where

$$\mathrm{width}_i(k) = ||a_{i,(k)}|| \left(\sqrt{log_e(2NK/\delta)}\right) + ||\tilde{\mathbf{v}}_i(k)||,$$

5: Select that alternative $i(k)$ which maximizes the upper confidence bound $\mathrm{ucb}_i(k)$.

Algorithm 6 SupLinRel

Parameters: $\delta \in [0,1]$, the number of trials K

Initialization: Let $S = \log K$ and set $\Psi^{(1)}(1) = \cdots = \Psi^{(S)}(1) = \emptyset$

1: **for** $k = 1, \ldots, N$ **do**

2: Initialize the set of the indexes of the feasible alternatives $A_1 := \{1, \ldots, N\}$, set $s := 1$.

3: Repeat until an alternative $\pi_{i(k)}$ is chosen:

I) Use `LinRel` with $\Psi^{(s)}(k)$ to calculate the upper confidence bound $\mathrm{ucb}_i^{(s)}(k)$ and its widths $\mathrm{width}^{(s)}(k)$ for all $i \in A_{(s)}$.

II) If $\mathrm{width}_i^{(s)}(k) > \frac{1}{2^s}$ for some $i \in A_s$ then choose this alternative and store the corresponding trial in $\Psi^{(s)}$,

$$\Psi^{(s)}(1+k) = \Psi^{(s)}(k) \cup \{k\}, \Psi^{(\sigma)}(1+k) = \Psi^{(\sigma)}(k) \text{ for } s \neq \sigma.$$

III) Else if $\mathrm{width}_i^{(s)}(k) \leq \frac{1}{\sqrt{K}} \forall i \in A_s$ then choose that alternative $i \in A_s$ which maximizes the maximum upper confidence bound $\mathrm{ucb}_i^{(s)}(k)$. Do not store this trial,

$$\Psi^{(\sigma)}(1+k) = \Psi^{(\sigma)}(k) \quad \forall \sigma = 1, \ldots, S.$$

IV) Else if $\mathrm{width}_i^{(s)}(k) \leq \frac{1}{2^s} \quad \forall i \in A_s$ then set

$$A_{s+1} = \left\{ i \in A_s \,\middle|\, \mathrm{ucb}_i^{(s)}(k) \geq \max_{j \in A_s} \mathrm{ucb}_j^{(s)(k)-\frac{2}{2^s}} \right\}$$

and increase s by 1.

5.2.3 Analysis

In this section the aim, ultimately, is to bound the regret (ie. a measure of the degree of mistakes) of the algorithm `SupLinRel`.We lay out the relevant results from [Aue02] to arrive to a bound of order $\tilde{O}(K^{\frac{1}{2}})$, over K trials.

Estimating $\mathbb{E}[\, x_i \,]$

The calculation of upper confidence bounds for $\mathbb{E}[\, x_i \,] = V_i^\top \xi_{(k)}$ is based on a weighted sum of past rewards, as follows:

We write $\xi_{(k)}$ as a *linear* combination of previously chosen vectors $\xi_{(\tau)}$, where $\tau \in \Psi_{(k)} \subseteq \{1,2,...,k-1\}$ are the previously selected indexes:

$$\xi_{(k)} = \sum_{\tau \in \Psi_{(k)}} a(\tau) \xi_{(\tau)} = \Xi(m) \cdot a(m)^\top,$$

for some coefficients $a(k) \in \mathbb{R}^{1 \times |\Psi_{(k)}|}$, where $\Xi(k)$ is a matrix of *previously* selected

feature vectors. Then $x(k) \cdot a(k)^\top$ is a good *estimator* for $V_i \cdot \xi_{(k)}$ since

$$V_i \cdot \xi_{(k)} = V_i \sum_{\tau \in \Psi_{(k)}} a(\tau)\xi_{(\tau)} = \sum_{\tau \in \Psi_{(k)}} a(\tau)(V_i \cdot \xi_{(\tau)}) = \sum_{\tau \in \Psi_{(k)}} a(\tau)\mathbb{E}\left[\ x_{i(\tau)}(\tau)\ \right] = \mathbb{E}\left[\ \mathbf{x}(k)\ \right] \cdot a(k)^\top$$

Probability of the error

Consider the eigenvalue decomposition

$$\Xi(k) \cdot \Xi(k)^\top = U(k)^\top \cdot \Delta(\lambda_1(k), \ldots, \lambda_d(k)) \cdot U(k),$$

where $\lambda_1(k), \ldots, \lambda_k(k) \geq 1$, $\lambda_{k+1}, \ldots, \lambda_d(k) < 1$ and $U(k)^\top U(k) = \Delta(1,...,1)$. Set

$$\tilde{\xi}_{(k)} = \left(\tilde{\xi}_1(k), \ldots, \tilde{\xi}_d(k)\right)^\top = U(k) \cdot \xi_{(k)}$$

and

$$\tilde{u}(k) = \left(\tilde{\xi}_1(k), \ldots, \tilde{\xi}_k(k), 0, \ldots, 0\right)^\top, \tilde{v}(k) = \left(0, \ldots, 0, \tilde{\xi}_{k+1}(k), \ldots, \tilde{\xi}_d(k)\right)^\top.$$

We use a lemma from [Aue02] to bound the expected loss of the algorithm's selection against the optimal choice:

Lemma 5.1. *Let $\delta \in [0,1]$ and let $\Psi_{(k)}$ be constructed in such a way that for fixed $z_{i(\tau)}(\tau), \tau \in \Psi_{(k)}$, the rewards $x_{i(\tau)}(\tau), \tau \in \Psi_{(k)}$, are independent random variables with means $\mathbb{E}\left[\ x_{i(\tau)}(\tau)\ \right] = V_i \cdot \xi_{(\tau)}$. Then, for all $i \in \{1, \ldots, N\}$,*

$$\mathbb{P}\left[\ |x(k) \cdot a(k)^\top - V_i \cdot \xi_{(k)}| \leq ||a(k)|| \left(\sqrt{2\log(2KN/\delta)}\right) + ||\tilde{v}(k)||\ \right] = 1 - \frac{\delta}{K}.$$

where K is the number of trials.

Remark 5.1. For the above result to hold, we need the rewards $x_{i(k)}(k)$ to be *independent*. This is directly related to the choices of the index sets $\Psi^{(s)}(k)$. We see how this is accomplished later. Summing up the above error inequalities for all $i = 1,2,\ldots,N$, the bound becomes worse by a factor of N.

Confidence bounds

Before continuing to the confidence bounds, we need first to bound $||a(k)||$ and $||v(k)||$. Auer et al prove the following lemma in [Aue02]:

Lemma 5.2. *Let $\Psi_{(k+1)} = \Psi_{(k)} \cup \{k\}$. The eigenvalues $\lambda_1(k), \ldots, \lambda_d(k)$ of $\Xi(k) \cdot \Xi(k)^\top$ and the eigenvalues $\lambda_1(k+1), \ldots, \lambda_d(k+1)$ of $\Xi(k+1) \cdot \Xi(k+1)^\top$ can be arranged in such a way that $\lambda_j(k) \leq \lambda_j(k+1)$ and*

$$||a(k)||^2 \leq 10 \sum_{j:\lambda_j(k) \geq 1} \frac{\lambda_j(t+1) - \lambda_j(k)}{\lambda_j(k)}$$

$$||v(k)||^2 \leq 4 \sum_{j:\lambda_j(t+1) < 5} |\lambda_j(t+1) - \lambda_j(k)|.$$

The *upper confidence bounds* for every policy π_i that `LinRel` constructs are:

$$\text{ucb}_i(k) = x(k) \cdot a(k)^\top + \text{width}_i(k),$$

where

$$\text{width}_i(k) = ||a(k)|| \left(\sqrt{\log(2KN/\delta)}\right) + ||\tilde{v}(k)||.$$

In order to ensure that for each $\Psi^{(s)}(k)$ the chosen rewards $x_{i(k)}(k)$ are independent for all $t \in \Psi^{(s)}(K+1)$ the algorithm `SupLinRel`(which uses `LinRel` as a subroutine) can be used.

The regret

If the Decision Maker knew the true values V_i for all policies $\pi_i \in \Pi$ then she would choose the policy that maximizes the expected reward, ie. $\pi^\star \in \text{argmax}_i \left\{V_i \xi_{(k)}\right\}$. Thus the regret of a learning algorithm against this optimal decision is given by

$$B(k) = \sum_{k=1}^{K} x^\star_{i(k)}(k) - \sum_{k=1}^{K} x_{i(k)}(k).$$

For our case the performance of `SupLinRel` is a bit worse (by a factor of N) than in [Aue02], since the expected reward of each policy π_i is governed by a different vector V_i (and not with the same weight vector f as in [Aue02]) -see Remark 5.1. However the the overall bound is still $\tilde{O}\left(\sqrt{K}\right)$. Again, from [Aue02], the following theorem holds:

Theorem 5.2. *When algorithm* `SupLinRel` *is run with parameter* $\delta/(1+\ln(K))$ *then with probability* $1-\delta$ *the regret of the algorithm is bounded by*

$$B(K) \leq 44N \left(1 + \log\left(2NK\log(K)\right)\right)^{\frac{3}{2}} \sqrt{KN} + 2N\sqrt{K}.$$

6

Conclusion

Decision makers often encounter the issue of uncertainty while trying to form plans of actions. In this thesis instead of using uncertain model dynamics that might be proven to be fatally erroneous, the uncertainty is modelled via a set of Markov decision problems that represent different possible scenarios that might be true. The focus lies in the question how to produce a minimax policy, ie. how to act optimally in the worst possible case.

Both cases of an infinite number of decisions and a finite decision horizon are considered and depending on the case a number of different approaches are utilized.

6.0.4 Cutting-plane

This method is motivated by the visual representation of the solutions. We show that, if each policy value hyperplane is viewed as a potential cut, then the problem of finding an efficient policy against the set of Markov decision problems cannot be solved in polynomial time, unless $P = NP$.

6.0.5 Uniform Sampling

Here we deal with the stochasticity of the policy rewards in a uniform way. Additional assumptions of bounding the value function with quadratics allow us to retrieve a probabilistic bound for the error of the approximation of the worst-case prior distribution of the Markov decision problems. Knowledge of this worst-case prior enables us to choose a policy that maximizes the rewards in this worst case scenario, with high probability.

6.0.6 Weighted Majority

We formulate our problem as a zero-sum game and directly apply a rewards-version weighted majority algorithm to our problem, by making the assumption that all information about the previous rewards becomes available in each round. We also show how

the worst-case prior distribution can be restricted without loss of generality, reducing the search space, in practice, to a smaller one (Remark 4.1). We give the standard performance bound for a deterministic-rewards version of our case.

Additionally, we weaken the assumption of the availability of full past information and the non-randomness of the outcomes and construct a more general variation of the weighted majority algorithm that operates with stochastic rewards. Based on the idea of the original performance theorem, we prove performance guarantees for this modified algorithm and bound the regret, with high probability, by an order of $O(\sqrt{\log K})$, over K trials.

6.0.7 Contextual Bandits

In this setting the presumption of knowledge of all previous alternatives is fully relaxed. A slight modification of the algorithms `SupLinRel` and `LinRel` is used to explore and exploit the policy space with a regret bound of $\tilde{O}(\sqrt{K})$.

6.1 Future Directions

The Cutting Plane algorithm's complexity classification as at least as hard as NP, as demonstrated in this thesis, leads naturally to the question of the existence of a tractable approximation of the problem's solution, as the next step. The complexity of approximate optimizations for the stochastic-blind-policy problem is still an open question. Addressing this question will allow us more insight on the usefulness of the cutting plane method in solving sequential decision problems.

Bibliography

[ACBF02] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[AHK12] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[Aue02] Peter Auer. Using confidence bounds for exploitation-exploration tradeoffs. *Journal of Machine Learning Research*, 3:397–422, 2002.

[BV08] Stephen Boyd and Lieven Vandenberghe. Localization and cutting-plane methods. *Lecture notes, Stanford University, www. stanford. edu/class/ee392o/localization-methods. pdf*, 2008.

[BVS08] Stephen Boyd, Lieven Vandenberghe, and Joelle Skaf. Analytic center cutting-plane method, 2008.

[Car00] Neal L Carothers. *Real analysis*. Cambridge University Press, 2000.

[DeG04] Morris DeGroot. *Optimal statistical decisions*. Wiley-Interscience, Hoboken, N.J, 2004.

[EM75] Jack Elzinga and Thomas G Moore. A central cutting plane algorithm for the convex programming problem. *Mathematical Programming*, 8(1):134–145, 1975.

[GD04] Peter D Grünwald and A Philip Dawid. Game theory, maximum entropy, minimum discrepancy and robust bayesian decision theory. *Annals of Statistics*, pages 1367–1433, 2004.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[KLC98] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.

[KNMS10] Robert Kleinberg, Alexandru Niculescu-Mizil, and Yogeshwer Sharma. Regret bounds for sleeping experts and bandits. *Machine learning*, 80(2-3):245–272, 2010.

[LGM01] Christopher Lusena, Judy Goldsmith, and Martin Mundhenk. Nonapproximability results for partially observable markov decision processes. *J. Artif. Intell. Res.(JAIR)*, 14:83–103, 2001.

[Lit94] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *ICML*, volume 94, pages 157–163, 1994.

[MMWW02] Hugues Marchand, Alexander Martin, Robert Weismantel, and Laurence Wolsey. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 123(1):397–446, 2002.

[NK86] John L Nazareth and Ram B Kulkarni. Linear programming formulations of markov decision processes. *Operations research letters*, 5(1):13–16, 1986.

[ORVR13] Ian Osband, Dan Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, pages 3003–3011, 2013.

[PG04] Laurent Péret and Frédérick Garcia. On-line search for solving markov decision processes via heuristic sampling. *learning*, 16:2, 2004.

[Put09] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*, volume 414. John Wiley & Sons, 2009.

[RW82] Uriel G Rothblum and Peter Whittle. Growth optimality for branching markov decision chains. *Mathematics of operations research*, 7(4):582–601, 1982.

[SCZ+05] Daniel Szer, Francois Charpillet, Shlomo Zilberstein, et al. Maa*: A heuristic search algorithm for solving decentralized pomdps. In *21st Conference on Uncertainty in Artificial Intelligence-UAI'2005*, 2005.

[Sip12] Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2012.

[ST01] Jamieson Schulte and Sebastian Thrun. A heuristic search algorithm for acting optimally in markov decision processes with deterministic hidden state. *Unpublished Manuscript*, 2001.

[Str03] Gilbert Strang. Introduction to linear algebra. *Cambridge Publication*, 2003.

[Sze10] Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 4(1):1–103, 2010.

[VLB12] Nikos Vlassis, Michael L Littman, and David Barber. On the computational complexity of stochastic controller optimization in pomdps. *ACM Transactions on Computation Theory (TOCT)*, 4(4):12, 2012.

[VLL90] Jan Van Leeuwen and Jan Leeuwen. *Handbook of theoretical computer science: Algorithms and complexity*, volume 1. Elsevier, 1990.

[YDM14] Saba Q Yahyaa, Madalina M Drugan, and Bernard Manderick. Knowledge gradient for multi-objective multi-armed bandit algorithms. In *ICAART 2014: International Conference on Agents and Artificial Intelligence.*, 2014.

Appendices

A

Preliminaries

This chapter contains the basic notions involved that are used in this thesis.

A.1 Linear Algebra

A.1.1 Eigenvalues & Eigenvectors

Definition A.1. Let A be a matrix. If there is a vector $v \in \mathbb{R}^n$ such that

$$Av = \lambda v \quad \text{and} \quad v \neq \mathbf{0}$$

for some scalar λ, then v is called an *eigenvector* and λ is called an *eigenvalue* of the matrix A.

Theorem A.2. *Let M be a matrix, with N linearly independent eigenvectors v_i, and corresponding eigenvalues λ_i, $i = 1,2,\ldots,N$. Then there exist a $N \times N$ matrix U such that matrix M can be written as:*

$$M = U\Delta U^\top$$

where the i^{th} column of U is the eigenvector v_i of M and Δ is the diagonal matrix with the eigenvalues λ_i as elements.

A.2 Analysis

A.2.1 Convex Analysis

Definition A.3. Let C be a set of points. If

$$x,\ y \in C, \quad 0 \leq \theta \leq 1$$

implies

$$\theta x + (1 - \theta)y \in C,$$

then C is called a *convex set.*

Definition A.4. Let X be a convex set and let $f : X \to \mathbb{R}$ be a function. If

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y)$$

for all x,$y \in X$ and for all $t \in [0,1]$, then f is called a *convex* function. If the inequality is strict, then f is called *strictly convex.*

Theorem A.5. *(Supporting Hyperplane Theorem). For every convex set C and point $x \in \vartheta C$, there exists a hyperplane P through x, such that C is contained in one of the half-spaces of P.*

Theorem A.6. *(Seperating Hyperplane Theorem). Let H_1 and H_2 be convex sets in $\mathbb{R}^n$ with disjoint interior. Then ther eis a hyperplane $\{x \mid a^\top x = b\}$ that seperates H_1 and H_2.*

A.2.2 Taylor

Theorem A.7. *Let $f : \mathbb{R} \to \mathbb{R}$ be a k-times differentiable function at $x_0 \in \mathbb{R}$, then there exists a real-valued function $R_n(x,x_0)$ so that*

$$f(x) - f(x_0) = \sum_{n=1}^{k} \frac{f^{(n)}(x_0)}{n!} (x - x_0)^n + R_n(x,x_0),$$

where R_n is called remainder and $f^{(n)}$ is the derivative of $f^{(n-1)}$. The remainder can be written in the form

$$R_n(x,x_0) = \frac{1}{(n+1)!} f^{(n+1)}(\vartheta_n)(x - x_0)^{n+1}$$

for some point $\vartheta \in (x_0,\, x) \cup (x,\, x_0)$.

A.3 Measure Theory

Definition A.8. Let Ω be a set. A set $\Sigma \subseteq \mathcal{P}(\Omega)$ is called a σ-algebra on Ω if the following conditions are true:

- $\Omega \in \Sigma$
- if $A \in \Sigma$ then $A^c \in \Sigma$
- if $A_i \in \mathcal{P}(\Omega), i = 1,2,\ldots,$ then $\bigcup_i A_i \in \Omega$

Definition A.9. A structure $\langle \Omega, \Sigma \rangle$ is called a *measurable space* if Σ is a $\sigma-$algebra on Ω. The elements of Σ are called events.

Definition A.10. Let $\langle \Omega, \Sigma \rangle$ be a measurable space. A function $\mu : \Sigma \to [-\infty, +\infty]$ is called a *measure* on $\langle \Omega, \Sigma \rangle$ if it satisfies the following properties:

- $\mu(\emptyset) = 0$
- If $(A_i)_{i=1,2,...}$ is a sequence of pairwise disjoint elements of the $\sigma-$algebra Σ, then

$$\mu(\bigcup_{i=1,2...} A_i) = \sum_{i=1,2,...} \mu(A_i)$$

If Ω has measure equal to 1, ie. $\mu(\Omega) = 1$ then μ is called a *probability* measure.

Definition A.11. A *measure space* is a measurable space equipped with a non-negative measure. That is, the triple $\langle \Omega, \Sigma, \mu \rangle$ is called a *measure space* if $\mu : \Sigma \to [0, +\infty]$ is a measure on $\langle \Omega, \Sigma \rangle$. Additionally, if μ is a probability measure, then $\langle \Omega, \Sigma, \mu \rangle$ is called a *probability space*.

Definition A.12. Let $E \subset \mathbb{R}$. Define the *Lebesgue outer measure* $\mu^\star(E)$ as:

$$\mu^\star(E) \triangleq \inf \left\{ \sum_{k=1}^{\infty} \text{length}(I_k) \ : \ I_k \text{ are open intervals with } \bigcup_k I_k \supset E \right\}.$$

Definition A.13. A set $E \subset \mathbb{R}$ is called *Lebesgue measurable* if for every subset A of R, it holds:

$$\mu^\star(A) = \mu^\star(A \cup E) + \mu^\star(A \cup E^c).$$

Then, the *Lebesgue measure* $\ell(E)$ of the Lebesgue measurable set E is defined to be its outer measure $\mu^\star(E)$.

Definition A.14. Let $\langle \Omega_1, \Sigma_1 \rangle$ and $\langle \Omega_2, \Sigma_2 \rangle$ be measurable spaces. Then, a function $f : \Omega_1 \to \Omega_2$ is called Σ_1, Σ_2-measurable if $\forall \Lambda \in \Omega_2 \ : \ f^{-1}(\Lambda) \in \Sigma_1$. If $\langle \Omega_2, \Sigma_2 \rangle = \langle \mathbb{R}, \mathcal{B}, \ell \rangle$, where $\mathcal{B}$ is the Borel σ-algebra (ie. the smallest σ-algebra that contains the open intervals, on $\mathbb{R}$), and ℓ is the Lebesgue measure, then we call f Σ_1-*measurable*.

Definition A.15. Let $\langle \Omega, \Sigma, \mu \rangle$ be a measure space. A function $f : \Omega \to \mathbb{R}$ is called a probability *density* function on Ω if f is:

- Σ-measurable
- non-negative μ-almost everywhere, and
- $\int_\Omega f(x) d\mu = 1$.

A probability density function f on $\langle \Omega, \Sigma, \mu \rangle$ generates a probability measure $\mathbb{P}$ on $\langle \Omega, \Sigma \rangle$ defined as

$$\mathbb{P}(A) \triangleq \int_{x \in A} f(x) d\mu$$

Definition A.16. Let $\langle \Omega, \Sigma, \mathbb{P} \rangle$ be a probability space. If a function $Y : \Omega \to \mathbb{R}$ is measurable, then it is called a *random variable*.

Definition A.17. Let Y be a random variable defined on a probability space $\langle \Omega, \Sigma, \mathbb{P} \rangle$. The *expectation* or *expected value* of the random variable Y is defined as

$$\mathbb{E}[\ Y\ ] \triangleq \int_{\Omega} Y d\mathbb{P}$$

—

Theorem A.18. *(Chernoff-Hoeffding) Let $X_1, \ldots, X_n$ be independent random variables in $[0, 1]$, with expected values $\mathbb{E}[\ X_i\ ]$ (not necessarily equal). Then, for $\lambda > 0$*

$$\mathbb{P}\left[\sum_{i=1}^{n} X_i \leq \sum_{i=1}^{n} \mathbb{E}[\ X_i\ ] - \lambda \right] \leq \exp\{-2\lambda^2/n\}$$

and

$$\mathbb{P}\left[\sum_{i=1}^{n} X_i \geq \sum_{i=1}^{n} \mathbb{E}[\ X_i\ ] + \lambda \right] \leq \exp\{-2\lambda^2/n\}.$$

A.4 Computational Theory

A.4.1 Computational Complexity

Big- and Soft-Oh Notation

Definition A.19. Let $f, g : \mathbb{R} \to \mathbb{R}$ be functions. Then g is an *asymptotic upper bound* for f, and we write

$$f(x) \in O\left(g(x)\right),$$

if a positive integer c exists, such that

$$f(x) \leq cg(x), \quad \text{as } x \to \infty.$$

.

Definition A.20. Let $f, g : \mathbb{R} \to \mathbb{R}$ be functions. We write $f(x) \in \tilde{O}\left(g(x)\right)$ if

$$f(x) \in O\left(g(x) \log^{\theta} g(x)\right)$$

for some θ.

Definition A.21. A set Σ is called an *alphabet* if

$$\Sigma \neq \emptyset \text{ and } |\Sigma| < \infty.$$

A *reduction* is a way of transforming a problem X to another problem Y, so the solution of Y can be used to solve X. More formally:

Definition A.22. Let $A \subseteq \Sigma^*$ and $B \subseteq T^*$ be non-empty sets and let $\phi : \Sigma^* \to T^*$ be a function. f is called a *reduction* of A to B if for every w:

$$w \in A \Leftrightarrow \phi(w) \in B.$$

Definition A.23. A *Turing machine* is a 7-tuple $\langle Q, \Sigma, \Gamma, \delta, q_o, q_{\text{accept}}, q_{\text{reject}} \rangle$, where Q, Σ, Γ are all finite sets and

- Q is the set of states
- Σ is the input alphabet, excluding the blank symbol $\sqcup$.
- Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function
- $q_0 \in Q$ is the starting state, q_{accept} is the accepting state, q_{reject} is the rejectint state
- $q_{\text{accept}} \neq q_{\text{reject}}$.

Definition A.24. A *non-deterministic Turing machine* is a turing machine with a transition function

$$\delta : Q \times \Gamma \to \mathcal{P}\left(Q \times \Gamma \times \{L, R\}\right),$$

where $\mathcal{P}(\cdot)$ is the power-set operator.

Definition A.25.

- A problem belongs to the computational class NP if it is solvable in polynomial time by a *non-deterministic* Turing machine.
- A problem belongs to the computational class NP-complete if it is in NP and if every other NP problem can be reduced do it.
- A problem is said to belong to the computational class NP-hard if there exists an NP-complete problem that is reducible in polynomial time to that problem.

Note that NP-hard problems cannot be solved in polynomial time unless P=NP ([VLL90]).