POLITECNICO DI MILANO
School of Industrial and Information Engineering

Master of Science in Computer Science and Engineering

Reward Estimation of Risk Sensitive Agents via Gradient Based Inverse Reinforcement Learning

Supervisor: Prof. Marcello Restelli

Candidate:
Federico Sandrelli
Matr. 901425

Academic Year 2019 - 2020

That quite definitely is the answer.
I think the problem, to be quite honest with you,
is that you've never actually known what the question is.
Douglas Adams

Abstract

In a standard Reinforcement Learning problem, an agent learns how to successfully behave in an environment through exploration by maximizing the expected return over trajectories. While this might be the optimal approach in some cases, there are scenarios in which it is convenient to include some notion of risk in the agent's optimization problem. For example, a car learning how to drive itself should not head full speed towards a wall just because it has not tried that yet. This concern led to the formulation of functions of the return (objective functions) that, when optimized, induce a more conservative approach in the exploration phase.

In this work, we focus on the Inverse RL problem for risk-averse agents. The goal is to infer the parameters of the reward function when the objective function is a known risk measure, by observing the behavior of an agent (expert). We follow a gradient-based approach that takes advantage of the fact that an expert's policy gradients are zeroed out at convergence. The intuition is that, since the real parameters of the reward induce the zero equality, we can search for them by minimizing some norm of the policy gradients w.r.t. such parameters. We present an empirical evaluation of this approach with several risk measures (Mean-Variance, Mean-Vola, ERM) and different environments (Portfolio Management, Trading, Grid World).

Keywords: Inverse Reinforcement Learning, Risk-Averse, Gradient-Based, Mean-Variance, Mean-Volatility, Entropic Risk Measure, Trading, Portfolio Management

Estratto

In un classico problema di Reinforcement Learning, un agente impara a comportarsi in modo proficuo all'interno di un ambiente tramite l'esplorazione con l'obbiettivo di massimizzare il valore medio di ritorno sulle proprie traiettorie. Questo approccio può risultare ottimo per certi ambianti, ma spesso conviene includere il concetto di rischio nell'ottimizzazione dell'agente. Per esempio, un veicolo autonomo che sta imparando a muoversi sulla strada non dovrebbe dirigersi a piena velocità verso un muro, solo perché non ne ha ancora esplorato le conseguenze. Questo aspetto ha portato alla formulazioni di funzioni del ritorno (*objective functions*) che, quando massimizzate dall'agente, inducano in esso un comportamento avverso al rischio e quindi anche una maggior cautela nella fase di esplorazione dello spazio.

In questo lavoro ci concentriamo sul problema inverso di trarre informazioni da agenti esperti (agenti che hanno imparato un comportamento proficuo) tramite l'osservazione delle loro traiettorie. Cioò che vogliamo dedurre da queste osservazioni sono i pesi che caratterizzano la funzione di reward quando l'obbiettivo dell'agente e' una *risk-averse objective function*. Utilizzeremo un approccio gradient-based che si basa sul fatto che i gradienti *log-policy* di un agente esperto siano identicamente uguali a zero. Cerchiamo quindi nello spazio delle reward, i pesi che minimizzano la norma dei gradienti calcolati su un batch di traiettorie dell'esperto. Presentiamo una valutazione empirica dell'approccio utilizzando diverse misure di rischio (Mean-Variance, ERM, Mean-Vola) su diversi ambienti (Portfolio Management, Trading, Grid World).

Parole Chiave: Inverse Reinforcement Learning, Risk-Averse, Gradient-Based, Mean-Variance, Mean-Volatility, Entropic Risk Measure, Trading, Portfolio Management

Contents

List of Figures

Chapter 1

Introduction

There are two steps to solving a problem. The first is to find a good representation of the problem and the second one is to find a solution through this representation. The first step can be sometimes overlooked because for humans it often happens unconsciously and in a very personal manner, but it is paramount for Machine Learning algorithms. The elements of Reinforcement Learning allow us to represent the classic Artificial Intelligence problems in a very succinct and effective way: an agent lives in an environment, performs actions, receives observations, and tries to optimize some objective. The agent learns by choosing among the actions it is capable of performing and by observing the outcome of its behavior. Every time the agent performs an action, its state changes and it receives a reward. By performing an action at every time-step and seeing which state-action pairs determine which reward, the agent learns how to behave to maximize, along a trajectory of several time steps, a function of the reward: the Objective Function. The first phase of this learning process is highly explorative as the agent has yet to learn how the environment works. When dealing with a simulation there is no concern regarding this aspect of the learning process, but there are situations where catastrophic outcomes are best to be avoided. Along with this idea, Risk Measures were introduced. Risk Measures allow the agent to gain insight into the most negative possible outcomes of its actions and help to avoid them. Risk-averse agents seek to maximize an Objective Function which is a trade-off between the expected cumulative return and some measure of risk. It is fascinating that an artificial agent can learn by itself in a way that is very relatable to how we as humans learn. However, we are still missing one of the fundamental aspects of this training process: pedagogy. Just as we learn by observing our peers, an agent can learn by observing an expert. This process is called Inverse Reinforcement Learning (IRL). The work of this thesis focuses on the IRL problem for risk-sensitive agents and in particular aims to estimate the reward function that induced the expert's policy to converge towards a certain behavior. Starting from an existing approach that operates

with risk-neutral agents ([13] and [14]), we aim to extend the framework to risk-averse agents. The challenge arises from the fact that risk-averse objectives are non-linear with respect to the reward, making the inverse problem harder to tackle.

1.1 Contributions

We can summarize the contributions of our work in two main sections. The first one envelops the theoretical advancements, while the second one concerns the empirical evaluations of this theory. From the theoretical point of view, we have devised a framework for solving the Risk-Averse Inverse Reinforcement Learning problem using an approach that is completely *model-free* and without posing any constraint on the objective function optimized by the expert. We provide the formulations for three different risk measures that allow us to simultaneously estimate both the weights of the reward function optimized by the expert and its level of risk aversion. Empirically, we evaluate all three risk measures on various domains of increasing complexity: a Grid World environment, a Portfolio Management environment, and a Trading simulation based on real data from the S&P500 index.

1.2 Outline of the Thesis

The Thesis is structured as follows. The next two chapters are dedicated to providing the necessary background knowledge and context on which the work builds upon. In particular, we will firstly cover some definitions of sequential decision problems, which are at the basis of the Reinforcement Learning problem formalization. We will then move on to a brief overview of the most relevant concepts of Reinforcement Learning, drawing attention to the concepts of risk-aversion and Inverse Reinforcement Learning. In particular, we will explore in detail the various risk measures that we have used in our framework.

Following these two chapters, we dedicate some time to exploring the current literature concerned with Risk-Averse Inverse Reinforcement Learning (RA-IRL), which is the branch of RL in which this thesis falls. Since there are not many publications for this particular field, we have broadened the scope of our literature overview to the areas of Risk-Averse Reinforcement Learning (RA-RL), and Inverse Reinforcement Learning (IRL). Some of the papers mentioned in this chapter provide the fundamental blocks for the work presented in this thesis.

Finally, we move onto the chapters that contain the novel contributions of our work. Here, we present in great detail the theoretical principles and mathematical formulations that we derived for our solution to the RA-IRL problem. In particular, we do so for

the three risk measures we decided to use to test our approach. We then move onto the actual implementation of the approach, describing the environments and main elements at play within the simulations. This portion of the Thesis contains the empirical results obtained and their respective evaluations. The last chapter contains the conclusions, dedicated to summarize and comment on the work we have done and the results we obtained.

Chapter 2

Sequential Decision Problems

As the name states, in a Sequential Decision Problem (SDP) the decision-maker (*agent*) must take a sequence of decisions that will allow it to progress within an environment. The agent's *utility* (what it wants to maximise), depends on the whole history of decisions (*trajectory*). In a scenario where there is complete knowledge of the environment and the consequence of actions (the agent's decisions) is deterministic, the problem reduces to simple action planning. Things become more complex and interesting when uncertainty is added to the environment. Every time the agent performs an action, it changes its state, for example, a pawn moving forward on a chessboard. When the environment transitions are stochastic, the agent's actions affect the state in unpredictable ways. Therefore a fixed sequence of actions (action plan) is insufficient to successfully navigate the environment. The agent needs something called a *policy* which is nothing more than a mapping from states to actions. The Sequential Decision Problem can be effectively represented by the Markov Decision Process.

2.1 The Markov Decision Process

The MDP can be formally defined as follows:

Definition 2.1.1. (Markov Decision Process). A Markov Decision Process (MDP) is a discrete time stochastic control process, defined by the tuple $< S, A, P, R, \gamma >$, where

- S is defined as the State Space.
- A is defined as the Action Space.
- $P : S \times A \times S \to R$ is a Markov transition kernel, i.e.
  - for every $s \in S$ and $a \in A$, P defines a distribution over the state space S.

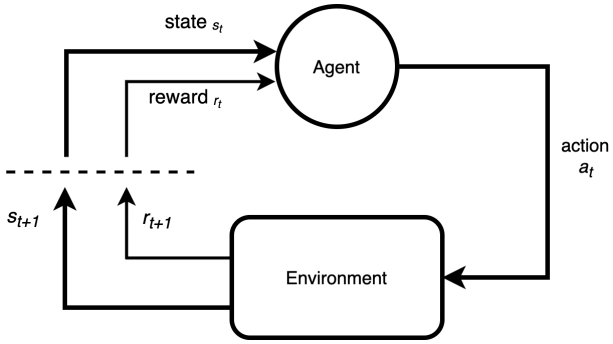


Figure 2.1: The Markov Decision Process

  - for every $B \in S$, $(s, a) \to P(s, a, B)$ is a measurable function on $S \to A$.

- $R : S \times A \to R$ is the reward function.
- $\gamma \in (0, 1)$ is the discount factor.

It is worth to focus for a moment on a few of the elements that have been introduced. The Markov transition kernel P defines the dynamics of the environment as it describes how the state of the agent will evolve at every step, given the action it performs. One should note the fact that the transition is stochastic and that the agent's action may, but won't necessarily, influence the transition. Another important characteristic of the Markov transition kernel is that the transition to a new state only depends on the previous state the agent was in and the action it performed. The path that has led the agent to the previous state and all of the history of the environment is not taken into account. It is this history independence that makes the model "Markovian" and it is a fundamental property to limit the dimension of the problem which would explode if one had to account for all possible histories and keep track of them.

The discount factor γ, instead, is crucial to the foresight of the agent. The higher the value, the more the agent will tend to compromise its desire to act greedily with the value it gives to future opportunities. This will be discussed in more detail after the policy and the Objective Function have been introduced.

Definition 2.1.2. (Policy). A policy is a function $\pi : S \times A \to R$ such that

- for every $s \in S$, $\pi(s)$ defines a probability distribution over the action space A.
- for every $C \in A$, $s \to \pi(s, C)$ is measurable.

In practice, the policy maps the state of the system to a probability distribution over the actions that the agent can perform in that particular state. One should note that just as the transition kernel, the policy does not take history into account and

only depends on the state. This is because we are considering Markovian, stationary policies. Some policies may depend on the history (history-dependent Policies) or on the time-step (non-stationary policies) but these lie outside of the scope of our work. Intuitively, the policy determines the behavior of the agent in the environment and the ultimate goal of the RL problem is to find the best policy. To understand how a policy is evaluated, let's explore further the *reward function* and the metrics that derive from it.

Definition 2.1.3. (Trajectory) Given an initial state s_0 in S and a policy π, a trajectory is a random sequence of state action pairs $\mathcal{T}_t = \{(S_t, A_t)\}_{t\geq 0}$ defined on a probability space that is tied to π and the transition kernel P, such that for $t = 0, 1, ...$

$$\begin{cases} S_0 = s_0 \\ A_t \sim \pi(S_t, \cdot) \\ S_{t+1} \sim P(S_t, A_t, \cdot) \end{cases} \tag{2.1}$$

‘

Definition 2.1.4. (Return) The Return is the cumulative reward, multiplied by the discount factor γ, obtained along a trajectory starting at time-step t.

$$G_t = \sum_{i=t}^{\inf} \gamma^i R_i \tag{2.2}$$

In the common scenario, the agent will try to find the trajectory that yields the greatest return and it will do so by maximizing the expected return over trajectories. Since the policy indirectly determines a probability distribution over the trajectories, we can evaluate a policy based on its expected return, weighted on the trajectory probabilities.
It is now clear how the discount factor can be used to distribute the attention over the future rewards along a trajectory to be followed. If $\gamma = 0$ then the agent will only build its policy considering the immediate reward (it will act greedily). On the other hand, if $gamma = 1$ the agent will assign equal importance to immediate and future rewards, which is usually suboptimal in a stochastic environment. The best value lies in between these two extremes and is problem-dependent.

From the Return, we can derive two fundamental metrics.

Definition 2.1.5. (State-Value Function) The State - Value function $V : S \to R$ is the expected cumulative reward from a given state.

$$V_\pi(s) = E[G_t | s_t = s] \tag{2.3}$$

The State-Value function allows the agent to evaluate the states, but since the agent only has control over the actions, we need also the Q function.

Definition 2.1.6. (Q-Value function) The Q-value function $Q : S \times A \to R$ ts the expected cumulative reward starting from a state $s \in S$ and performing an action $a \in A$.

$$Q_\pi(s, a) = E[G_t | s_t = s, a_t = a] \tag{2.4}$$

In both these metrics, the subscript π indicates that the expectation is evaluated over a given policy π. The two values are tied together by the following relationship:

$$V_\pi(s) = \int_A \pi(a|s) Q_\pi(s, a) da \tag{2.5}$$

Now that we have defined these metrics, we shall see how they can be computed for a given environment. It is necessary to note that the following equations assume that the transition kernel P is known.

Definition 2.1.7. (Bellman Expectation Equations) At the core of solving an MDP are the Bellman Equations. These equations provide a recursive definition of the value functions that make it possible to evaluate the state space through recursive algorithms that we will see later on. The idea is that the value of a state (or state-action pair) should be the sum of the values of the other states, weighted by the probability of reaching them.

$$V_\pi(s) = R_\pi(s) + \gamma \cdot \int_A \pi(a|s) \int_S P(s'|s, a) V_\pi(s') ds' da \tag{2.6}$$

$$\begin{aligned} Q_\pi(s, a) &= R(s, a) + \gamma \cdot \int_S P(s'|s, a) \int_A \pi(a|s') Q_\pi(s', a) da ds' \\ &= R(s, a) + \gamma \cdot \int_S P(s'|s, a) V_\pi(s) ds' \end{aligned} \tag{2.7}$$

where

$$R_\pi(s) = \int_A \pi(a|s) R(s, a) da \tag{2.8}$$

From the Bellman Expectation Equations, we can derive the following update rule which will allow us to iteratively compute the State-Value function (and analogously the Q-Value function).

$$v_{k+1}(s) = \sum_{a \in A} \pi(a|s) \left(R(s, a) \gamma \sum_{s' \in S} P(s'|s, a) v_k(s) \right) \tag{2.9}$$

It can be proven that, for $k \to \inf$, v_k will converge to the true value V_π.

2.2 Dynamic Programming

Dynamic Programming is a mathematical optimization method, conceived by Richard Bellman, which makes use of the homonymous Bellman Equations to solve a large problem by splitting it into smaller problems which are solved recursively.

Applied to SDPs, DP algorithms are guaranteed to converge to the optimal solution with an infinite number of iterations. This property makes them very desirable, but it comes with a cost. As a matter of fact, the iterative approach, which requires the state values to be re-computed at every iteration, is very computationally expensive. When the state space is large enough (e.g. the game of Chess) the approach becomes infeasible. Furthermore, DP algorithms rely on the strong assumption that the agent has complete and perfect knowledge of the environment. This means knowing a priori the transition kernel P. This limitation is often unmet in real-world applications, which resulted in the development of different approaches to solve SDPs. Nevertheless, The principles of DP are fundamental concepts that subsequent approaches build upon and their understanding is therefore of paramount importance. In the next sections, we introduce these concepts.

2.2.1 Evaluating the Policy

To find the "best" policy we must be able to evaluate one. One of the ways we can evaluate a policy is computing the value function that derives from it by iteratively applying the Bellman equations.

$$\begin{aligned} v_{k+1}(s) &= \sum_{a \in A} \pi(a|s)(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_k(s')) \\ \boldsymbol{v}_{k+1} &= \boldsymbol{R}^\pi + \gamma \boldsymbol{P}^\pi \boldsymbol{v}_k \end{aligned} \tag{2.10}$$

The value function can be seen as a vector of the dimension of the environment's state space. It is initialized with all values set to zero and at every iteration, each element of the vector representing a state s is updated to contain the reward obtained from s, plus the sum of the values of all elements representing the states s' reachable from s, weighted by the probabilities given by the policy. The process will converge to the real Value function. The quality of the policy can be then estimated from the value of the starting state $v(s_0)$ which corresponds to the discounted return the agent can expect to incur starting from that state and following policy π.

2.2.2 Improving the Policy

The value function we just computed can then be used to improve the policy. Following the new value function, the policy is updated so that for each state, it will greedily prefer

the action that leads to the reachable state with the highest value $v(s)$. The quality of actions is formalized in the Q-function.

$$q_\pi(s,a) = R(s,a) + \gamma \cdot \sum_{s \in S} P(s'|s,a) V_\pi(s) \tag{2.11}$$

This back and forth procedure will eventually converge to an optimal policy for the environment into consideration. While the convergence is guaranteed, it is quite slow since every time the policy is updated, the new value function must be computed ground up.

2.2.3 Value Iteration

To reduce the computational burden of the approach we just saw, the policy can be taken out of the equation. Instead of evaluating a state by weighting on the actions based on the current policy ($\pi(a|s)$), the Value Iteration algorithm simply uses the action that leads to the best next state (of course, still taking into account the stochasticity of the environment). This algorithm drastically speeds up the convergence to the real value function, from which the optimal policy can easily be derived through greedy selection. The new update rule can be formalized as such:

$$\begin{aligned} v_{k+1}(s) &= \max_{a \in A}(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_k(s')) \\ \boldsymbol{v_{k+1}} &= \max_{\boldsymbol{a \in A}} \boldsymbol{R^a} + \boldsymbol{\gamma P^a v^k} \end{aligned} \tag{2.12}$$

2.2.4 Generalised Policy Iteration

The general approach of DP algorithms consists of a two-step phase where the policy is firstly evaluated and then improved with respect to this evaluation. This concept is denoted as *Generalised Policy Iteration* (GPI) and it effectively describes not only DP algorithms but most RL approaches.

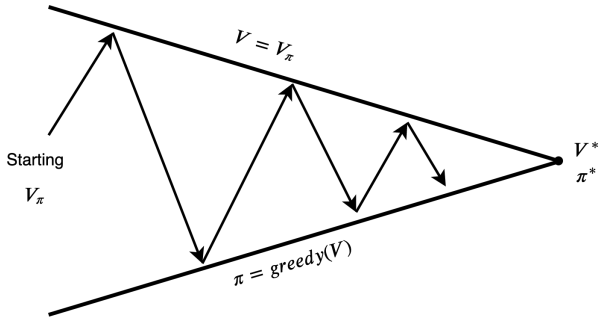


Figure 2.2: Convergence to optimality in a Generalised Policy Iteration process.

Chapter 3

Reinforcement Learning

At the basis of the human learning scheme, is the ability to interact with the environment and observing how it changes as a consequence of the interaction. As we gather experiences, we become more aware of the relational cause-effect link that exists between our actions and how the environment evolves around us. As the famous idiom states "we learn by making mistakes". Reinforcement Learning (RL) is the algorithmic equivalent of this fundamental concept. The subject of this process is the Agent, and its goal is to learn how to successfully interact with the environment it lives in. Initially oblivious to the dynamics of the world that surrounds it, the agent must gain experiences through which it can learn profitable strategies. In RL this process is effectively represented by the Markov Decision Process. The MDP models the interactions between an Agent and the Environment. In this framework, the experiences of the agent are modeled as a Sequential Decision Process. Time is discretized into steps and at every step, the agent can perform an action and observe how the environment evolves. This observation is not active on the part of the agent, but rather passive, as it is the environment that receives the action and returns a limited description of the new state of the environment (the Observation). Together with the observation, the agent receives from the environment a very important element, the *reward*. The *reward* is a numerical value through which the Agent can evaluate the interaction and deem it as successful or not. The goal of the agent is to develop an interaction strategy (so-called *policy*) to maximize some function of the *reward* over a succession of time steps (which may be finite or infinite).

3.1 Model Free Reinforcement Learning

Reinforcement Learning lies somewhat in between the two main paradigms of Machine Learning which are Supervised and Unsupervised Learning. It does not rely on labeled data and in this way it differs from Supervised Learning, but at the same time, as

the agent becomes more knowledgeable of the environment, the training becomes more conscious of what data represents (good or bad states and actions in trajectories). This leads to what is known as the *exploitation-exploration trade-off*. The agent must choose whether to explore unseen states and actions, or whether to exploit the knowledge it has gained so far from the environment to optimize its strategy within the limits of its knowledge. The idea is that the agent should vary this trade-off over time. An initial highly explorative phase will allow it to gain a broader view of what the possible strategies lead to, while as the training progresses, a more conservative approach can yield better results more quickly by avoiding returning to states that have already shown to be less proficuous.

Algorithms used to solve the problem of finding the optimal policy, lie in two major categories: Model-free and Model-based approaches. Model-based approaches require previous knowledge of the model such as the state space S and the transition kernel P. These approaches are usually more efficient and yield better results but aren't always applicable. It is often the case that the model cannot be known and a policy must be derived simply by exploring the environment. So all the information in posses of model-free approaches is the trajectories of the agent that is being trained. To this end, the problem can be tackled in three ways:

- *Model approximation*: this approach consists in estimating the transition matrix P and the entire state space S through a large number of randomized trajectories. Once the model has been estimated, any standard model-based approach can be used to derive a policy. The shortcoming of this approach is that a good estimate of the model requires a large number of trajectories. In real-world applications, data efficiency is highly desirable (this becomes even more true in risk-averse settings), therefore this limitation is very strong.

- *Direct RL approaches*: this class of model-free approaches completely overcomes the necessity of a model and only rely on trajectories provided by the expert. Within this class, we can find *value approximation* algorithms, which try to estimate the value function (V or Q) from which to derive the optimal strategy and *policy approximation* algorithms, which directly optimise the policy.

Direct RL techniques are split in two main categories: *online* and *offline* approaches. Both of these approaches rely on observed trajectories to update the policy or value-function. The difference lies in the fact that the former performs a parameter update with every new step observation while the latter works in a batch-fashion as it collects a large number of trajectories before performing an update based on the average performance of these trajectories. Online approaches are useful in scenarios where sample efficiency is essential, but they require more attention concerning the exploration-exploitation trade-off we introduced earlier because updates are performed even when the collected

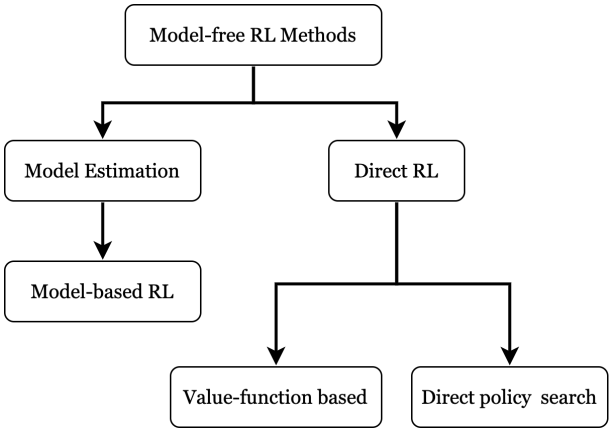


Figure 3.1: Model Free RL Scheme

data is small and the agent still has very limited experience of the model. Batch based approaches instead allow for a more robust descent but require a larger amount of trajectories to be collected. For the scope of this thesis, we decided to implement Offline batch-based methods since they are well suited for simulated environments such as the ones we will be dealing with in chapter 6.

3.1.1 Policy Gradient Methods

As we have stated, policy gradient methods directly act on the policy by updating its parameters to iteratively improve its performance measured by the objective function. The value of the policy is given by the objective function itself:

$$J(\theta) = \sum_{s \in S} d^{\pi}(s) \sum_{a \in A} \pi(a|s) Q^{\pi}(s,a) \tag{3.1}$$

where $d^{\pi}(s)$ is the state distribution given the policy π. This parameter is tricky to estimate given that the model is unknown. Luckily, the gradient of the objective function with respect to the policy parameters does not depend on the transition model at all as stated in the Policy Gradient Theorem [21]:

$$\frac{\partial J(\theta)}{\partial \theta} = \int_S d^{\pi}(s) \int_A \frac{\partial \pi(a|s)}{\partial \theta} Q^{\pi}(s,a) da ds \tag{3.2}$$

At every iteration, the parameters θ of the policy are updated using the well-known gradient ascent (descent) so as to maximize (minimize) the objective function value.

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmax}}\, J(\theta) \tag{3.3}$$

$$\theta_{k+1} = \theta_k + \alpha_k \nabla_{\theta} J(\theta) \tag{3.4}$$

REINFORCE

REINFORCE, also known as Monte Carlo Policy Gradient, relies on sampled trajectories to estimate the gradients. Since the expectation computed over N samples corresponds to the actual gradient, this approach has been proven to converge, for an infinite number of samples, to the real gradient of the objective function. In the case of a finite horizon, for a given trajectory τ, we have:

$$\begin{aligned} \theta^* &= \underset{\theta}{\text{argmax}} \underset{\tau \sim \pi_\theta}{\mathbb{E}}[G(\tau)] \\ &= \underset{\theta}{\text{argmax}}\, J(\theta) \end{aligned} \tag{3.5}$$

where $r(\tau)$ is the cumulative reward over the trajectory τ. Using Monte Carlo roll-outs, we can estimate $J(\theta)$ from the trajectories.

$$J(\theta) = \underset{\pi_\theta}{\mathbb{E}}[G(\tau)] \sim \frac{1}{N} \sum_{i=1}^{N} G(\tau_i) \tag{3.6}$$

$$\begin{aligned} \nabla_\theta J(\theta) &= \underset{\pi_\theta}{\mathbb{E}}[\nabla_\theta log(\pi_\theta(\tau)) G(\tau)] \\ &\sim \frac{1}{N} \sum_{i=1}^{N} [\nabla_\theta log(\pi_\theta(\tau_i)) G(\tau_i)] \end{aligned} \tag{3.7}$$

Intuitively, what is happening here is that we are fitting the model to estimate the return. The left-hand side term is simply a Maximum Likelihood Estimator which tries to fit θ so to maximize the probability of a given example trajectory τ_i. The return $G(\tau_i)$ acts as a weight to increase the probability of profitable trajectories.

An improvement that will yield a much faster convergence can be achieved by adding a baseline to the reward. The baseline acts as a means of comparison for the reward. Since the reward acts as a weight to the MLE, instead of assigning large positive weights to good trajectories and small positive weights to bad ones, the baseline will act so that bad trajectories are weighted negatively, effectively reducing their probability of occurrence rather than increasing it slightly.

$$\nabla_\theta J(\theta) \sim \frac{1}{N} \sum_{i=1}^{N} \left[\nabla_\theta log(\pi_\theta(\tau_i)) \left(G(\tau_i) - b\right)\right] \tag{3.8}$$

A simple but effective baseline is the average reward of the return over the batch of trajectories.

$$b = \frac{1}{N} \sum_{i=0}^{N} G(\tau_i) \tag{3.9}$$

GPOMDP

One of the shortcomings of Monte Carlo policy methods is the high variance of the policy gradient. An effective countermeasure consists in taking into consideration the causality of actions. The unrolling of a trajectory from a given state should not be influenced by the rewards encountered up to that moment but only from the future rewards. The notion of causality can be introduced as follows:

$$\nabla_\theta J(\theta) \sim \frac{1}{N} \sum_{i=1}^{N} \left[\sum_{t=1}^{T} \nabla_\theta log(\pi_\theta(\tau_i)) \left(\sum_{t'=t}^{T} G(s_{it'}, a_{it'}) - b \right) \right] \tag{3.10}$$

Equation 3.10 formalizes the well know Gradient of the average reward in Partially Observable Markov Decision Processes. Here, half of the cross products are effectively removed from the equation, yielding a much more stable gradient descent (ascent) compared to the simpler REINFORCE approach.

Natural Policy Gradient

As we have seen, reducing the variance of policy gradient methods significantly improves their performance allowing for faster and smoother convergence. Nevertheless, these improvements do not help to tackle large plateaux in the return landscape. In these regions of the objective function, gradients become very small and stall the learning curve. The Natural Gradient technique allows us to redefine the meaning of the step size, resulting in a much more efficient gradient descent that tends to avoid pitfalls such as these plateaux. This technique was firstly introduced in the RL setting in [5] and later applied to actor-critic algorithms in [12].

In regular policy gradient methods, the policy parameters are changed by an amount that depends on the magnitude of the computed gradient of the objective function. How much these changes will affect the behavior of the policy is not taken into account. Natural Policy Gradient follows a completely different approach by defining a step size that depends on the behavior shift magnitude, in other words, it penalizes changes in the parameters that result in drastic changes of the policy's behavior. Since policies define a probability distribution over actions (and implicitly over states), it is reasonable to make use of the Kullback-Leiber divergence to measure the difference between π_θ and $\pi_{\theta+\nabla\theta}$. The *Fisher Information Matrix* allows us to measure this divergence with respect to the policy parameters θ.

The idea is that fixed the value of the KL-divergence, among all the possible parameter combinations that induce the desired value, we choose the ones that maximise the objective function.

Mathematically, The Fisher Information Matrix is the second derivative of the KL divergence of the updated policy parameters with respect to the original parameters:

Algorithm 1 Natural Policy Gradient

Input: initial policy parameters θ_0, learning rate α_k

Output: approximated optimal parameters θ^*

repeat

  Approximate Fisher Matrix F_θ

  Approximate vanilla policy gradient: $\hat{g} \approx \nabla_\theta J(\theta_k)$

  Approximate natural policy gradient: $\hat{g}_N = F_\theta^{-1}\hat{g}$

  Approximate parameters using gradient ascent: $\theta_{k+1} = \theta_k + \alpha_k \hat{g}_N$

  $k \leftarrow k + 1$

until *convergence*;

$$d_{KL}(\pi_\theta, \pi_{\theta+\delta\theta}) \sim \frac{1}{2}\delta\theta^T F_\theta \delta\theta \tag{3.11}$$

Therefore, the problem to be solved which is constrained by a fixed value c of the KL-divergence is:

$$\delta\theta = \underset{\delta\theta : d_{KL}=c}{\operatorname{argmax}} J(\theta + \delta\theta) = \alpha F_\theta^{-1} \nabla_\theta J(\theta) \tag{3.12}$$

Given that the Fisher matrix can be approximated as:

$$F_\theta(s) = \mathop{\mathbb{E}}_{\pi_\theta(a|s)} \left[\frac{\partial log \pi_\theta(a|s)}{\partial \theta_i} \frac{\partial log \pi_\theta(a|s)}{\partial \theta_i} \right] \tag{3.13}$$

it can be approximated from a batch of trajectories as follows:

$$F_\theta(s) = \frac{1}{N} \sum_{i=0}^{N-1} \left[\sum_{t=0}^{T-1} \left(\sum_{t'=0}^{t} \nabla_\theta log \pi_\theta(a_j|s_j) \right) \nabla_\theta log \pi_\theta(a_i|s_i)^T \right] \tag{3.14}$$

TRPO and PPO

While ensuring some exceptional theoretical advantages, Natural Gradient does not scale well due to the computational burden given by the second-order derivative matrix. When the problem state space is large, the approach becomes unusable. To tackle this shortcoming, two efficient algorithms were developed: TRPO [16] and PPO [17]. The former relies on the *trust-region* method: the policy update is performed within a region that ensures the KL-divergence resulting from the policy update won't exceed a given threshold. Furthermore, following the *conjugate gradient method*, the algorithm ensures that every gradient step is orthogonal to previous steps to maximise step efficiency. The drawback of this approach is that to allow for the trust-region to have an acceptable size, some of the constraints are loosened up resulting in the possibility of performing degrading

steps. PPO, instead, follows another approach by using a tradeoff rather than imposing a hard boundary to limit the policy step divergence. The divergence is included in the objective function as a penalty so that the agent is drawn to minimize it. This approach makes it possible to use simpler gradient descent methods that are easier to implement and more computationally efficient.

Stochastic Policies and Boltzmann

The gradient descent approaches we have just seen only work when the policy into consideration is stochastic, meaning that it will output a distribution over the admissible actions that is not deterministic (i.e. $\pi(a|s) > 0 \forall s \in S \forall$ admissible a in s). This restriction does not allow us to use deterministic policies which can be quite limiting. Ways to overcome this limitations are using *hyper-policies* or *Deterministic Policy Gradient* [18]. Nevertheless, it is not of our concern for the scope of the project since we will be exploring environments where stochastic policies are well suited. A common choice for discrete action spaces that we have chosen to use for this work is the Boltzmann policy (also known as Softmax) [8].

$$\pi_\theta(s,a) = \frac{e^{\theta^T \phi(s,a)}}{\sum_{b \in A} e^{\theta^T \phi(s,b)}} \tag{3.15}$$

where $\phi(s,a)$ is a feature vector that identifies a specific state-action pair. The likelihood gradient of the policy can be formalized as

$$\nabla_\theta \log \pi_\theta(s,a) = \phi(s,a) - \sum_{b \in A} \pi_\theta(s,b)\phi(s,b) \tag{3.16}$$

3.2 Inverse Reinforcement Learning

Consider a Markov Decision Process as defined in 2.1.1 by the tuple $< S, A, P, R >$. This time though, the reward function is not provided, we have an MDP in the form of $< S, A, P >$ and, along with the model, we can observe an expert acting in the environment. In particular, we have access to a set of its trajectories $\tau_i (i \in [0, N])$. Differently from the goal of classic Reinforcement Learning approaches, we have seen, here the objective is to estimate the *reward function*, given a set of demonstrations (trajectories) form an optimal agent. Instead of learning through exploration, IRL is concerned with learning through observation. This setting is useful for learning optimal behaviors in applications for which it might be difficult to define an objective function that expresses the trade-off that the expert wants to optimize.

A fundamental paper for IRL is Algorithms for Inverse Reinforcement Learning by Andrew Ng. et al. [9]. In this work, the authors devise a set of algorithms to tackle the IRL problem both when the policy is known and in the case where the only information provided is the expert's observations. The key issue that arises is *degeneracy*: the observed behavior can be optimal for a large space of reward functions, making it impossible to provide a single solution to the problem. The authors partially overcome this obstacle through heuristic approaches. A different approach has been presented by Ziebart et al. in [30] where the IRL problem is solved through a probabilistic method based on the principle of maximum entropy. The authors test the algorithm in an interesting environment derived from real data collected from taxi drivers in a metropolis. Other approaches have been proposed in [19] and in [13]. In particular, we will concentrate on the GIRL algorithm presented in [13] and extend its applicability as we shall see later on. It is worth noting that some approaches suffer from the drawback of requiring the solution to the forward problem (solving the MDP). This step is very computationally expensive and approaches that manage to avoid it (such as GIRL [13]) present a clear advantage.

3.2.1 IRL vs Behavioural Cloning

It is worth to stress the fact that Inverse Reinforcement Learning is not concerned with estimating the policy of the agent, which is a process defined as Behavioural Cloning. There are cases in which behavioral cloning can be the right choice, for example when we have a large number of samples and very little domain shift but it is not necessarily the most generalized representation of the task. The observation might miss out on the salient part of the behavior, this means that if our agent learns to mimic the expert, it will mimic every aspect of the expert's behavior, whether it was relevant to the task or not. Finally, the expert might have capabilities that our agent does not. In these cases, knowledge of the objective rather than the behavior will be more effective and will allow for a better transition to new domains (improved generalization). So rather than dealing with an imitation problem, we are dealing with an intent inference problem.

3.3 Risk-Aversion in Reinforcement Learning

As previously stated, a strategy which maximises the expected return on trajectories may not be optimal for some applications. For example, it can be useful to avoid risk in environments where catastrophic scenarios are not recoverable. In Markov Decision Processes the concept of risk comes from two distinct uncertainties. The first one, called "inherent uncertainty", derives from the stochasticity of the environment itself. Tackling this type of uncertainty requires a transformation of the objective function.

It is important to note that the probability distribution of the reward could remain the same or change over time, in the former case we talking of static risk, while in the latter of dynamic risk. The second type of uncertainty is Model uncertainty or parametric uncertainty. It is defined for robust MDPs and derives from uncertainty over the parameters of the model.

There are several ways to account for risk:

- **Utility Approach**: the standard value function is replaced with the expected value of a utility function of the return that contains the notion of risk.
- **Risk Measures**: this strategy consists in the addition of a nonlinear term that describes the risk. Examples of these terms are the Variance or the Conditional Value at Risk.
- **Robust MDPs**: in this context, there is no risk term added. The risk is accounted for by considering the worst possible case scenario for the parameters of the model.

3.3.1 Utility functions

This approach builds on the Utility Theory, introduced in [3]. At every time-step, the agent is presented with a set of alternatives in the form of actions. A utility function provides a way of assigning a preference order to this set of alternatives so that the agent can choose the action that maximises the utility. These functions can be formulated so that some notion of risk is included in the valuation of the alternatives. An example is the variability of the outcome that follows each choice. The idea behind this approach to Risk-Averse RL is to apply these utility functions to the cumulative return so that the optimization process will yield risk-averse policies.
The risk aversion of a utility function is directly tied to its convexity as decision-makers optimising concave functions are prone to diversification which is a risk-averse attitude. The following formula expressed the concept of convexity as diversification.

$$\rho(\beta X + (1-\beta)Y)) > \beta\rho(X) + (1-\beta)\rho(Y) \tag{3.17}$$

Where X and Y are two random variables representing two choices and β is the probability to take the first one. It is possible to choose a utility function that best suits our needs, for example, the exponential function $u(x) = e^{\lambda x}$ applied to the return is a popular choice as its second-order approximation is equal to the common mean-variance optimization problem.

$$U(T, \pi) = \mathbb{E}_\pi[R] + \lambda Var_\pi[R] + \mathcal{O}(\lambda^2) \tag{3.18}$$

3.3.2 Risk Measures

The risk measures approach explicitly incorporates the concept of risk.
We can see our risk measure as a mapping $\rho : Z \to \mathbb{R}$ from a random variable Z to the domain or real numbers.

Variance

A popular risk measure is the Variance of the return. This measure effectively incorporates the notion of risk defined as uncertainty in the outcome. When the return value over trajectories is unstable this means that the process is less controllable and less predictable, and therefore, in some way, riskier. Mathematically, the return variance can be formalised as follows:

$$Var(G) = \mathbb{E}[(G - J_\pi)^2] = \mathbb{E}[G^2] - J_\pi^2 \tag{3.19}$$

From the above formulation, we can derive a new objective function called Mean-Variance [25], which tries to maximize a trade-off parametrized by lambda between the expected return value and its variance.

$$MeanVar = J_\pi - \lambda Var_\pi[G] \tag{3.20}$$

This approach does have some shortcomings, primarily given by the fact that return values at the extremes of the distribution are weighted equally without considering whether they are positive or negative (it is symmetric). Nevertheless, its simplicity and intuitiveness make it a risk measure worth exploring.

Entropic Risk Measure

As the Variance, this measure allows taking into account higher-order statistics of the reward that give a better insight into the riskiness of a policy. The ERM induces risk-averse behaviors by overestimating negative outcomes, thanks to the exponential utility function. The formulation is the following:

$$ERM(G) = \frac{1}{\lambda} log\, \mathbb{E}\left[exp(-\lambda G)\right] \tag{3.21}$$

where the parameter λ determines the level of risk aversion induced by this measure.

It is worth noting that the second Taylor expansion of the ERM with respect to the risk aversion parameter λ is equal to the Mean Variance objective function:

$$ERM(G) \approx \mathbb{E}[G] + \lambda Var[G] + O[\lambda^2] \tag{3.22}$$

Reward Volatility

Introduced in [1], the reward volatility has a very similar meaning to the measures we have just seen with the difference that instead of describing the uncertainty of the accumulated return over whole trajectories, it is concerned with the level of step-wise uncertainty. It is defined as the variance of the step reward under the state-occupancy measure:

$$\nu^2_{\mu,\pi} = \mathop{\mathbb{E}}_{\substack{s\sim d_{\mu,\pi} \\ s\sim\pi(\cdot|s)}} \left[(R(s,a) - J_\pi)^2\right] \tag{3.23}$$

Sharpe Ratio

Another popular measure is the Sharpe ratio in which the expected value is divided by the inverse of the square root of the Variance. This measure is widely used in finance because it provides an effective way of evaluating the performance of a portfolio with respect to its riskiness. In practice, it measures how well the return compensates any added risk compared to a portfolio composed of risk-free assets only. It is defined as:

$$Sh = \frac{\mathbb{E}_\pi[G]}{\sqrt{Var_\pi[G]}} \tag{3.24}$$

uguale alla var ma non lo usiamo perche' e' piu' scomodo perche' ha una fomrulazione con un gradiante complesso per dia della divisione

Coherent Risk Measures

In [23], the authors identify four useful properties that risk measures should abide to in order to be "rational". Risk measures that satisfy these properties are called Coherent Risk Measures and the properties are:

1. Convexity: $\forall\alpha \in [0,1]$, $\rho(\alpha X + (1-\alpha)Y \leq \alpha\rho(X) + (1-\lambda)\rho(Y)$
   This property defines how diversification leads to reduced risk.

2. Monotonicity: if $X \leq Y$, then $\rho(X) \leq \rho(Y)$ This property means that to higher costs are associated higher risks.

3. Translation invariance: $\forall b \in \mathbb{R}, \rho(X+b) = \rho(X) + b$ This property ensures that the constant part of the cost does not influence risk.

4. Positive homogeneity: if $\alpha \geq 0$, then $\rho(\alpha X) = \alpha\rho(X)$ This property ensures that an outcome times a multiplicative factor α has *alpha* times the risk.

Where X and $Y \in Z$, the space of random variables

3.3.3 Robust MDPs

As we have seen, Sequential Decision Problems can be effectively described by Markov Decision Processes, which are models parametrized by the transition probabilities and reward function. These parameters are often estimated from noisy data (which might render them unreliable) and may even change during a simulation. To tackle this problem, the framework of robust MDPs has been devised in [10] and [4]. The robust MDP framework assumes that the parameters of the MDP exist within an uncertainty set U and therefore the optimization process must be performed with respect to the worst realization of the parameters within the set. The main drawback of this approach is that, if the uncertainty set is too large, it can lead to policies that are too conservative. Some works have successfully mitigated this issue such as [20] and [29]. An interesting publication by Takayuki Osogami [11] explores the connection between robustness and risk sensitivity. The author shows how a risk-sensitive MDP that minimizes an *iterated risk measure* (composed of coherent risk measures which abide by certain restrictions) behaves equally to a robust MDP that minimizes the worst-case expectation when the possible deviation from nominal values is characterized by a concave function. Furthermore, he demonstrates how the particular case of an MDP that minimizes the *expected exponential utility function* is equivalent to a robust MDP that minimizes the worst-case expectation with a penalty from deviation (measured by the KL-divergence) of uncertain parameters from their nominal values.

Chapter 4

Literature Review

In this chapter, we are going to present the main related works about Policy Gradient algorithms for Mean-Variance optimization and other interesting works within Risk-Averse Inverse Reinforcement Learning. We will deeply analyze only those works that are strictly connected to our solution while giving a brief overview of other relevant researches.

4.1 Forward RL

In this section, we focus on some works in the area of forward Reinforcement Learning, i.e. the process of training an agent to perform successfully in a given environment. In particular, we draw the attention to three publications concerning the optimization of risk-averse agents through the risk measures we introduced in chapter 3: Variance, Volatility, and Entropic Risk. The approaches presented have been used in this thesis to generate the experts from which the Inverse RL process can begin.

4.1.1 Variance Adjusted Actor-Critic Algorithms

The first paper we are going to present from Tamar [25] concerns the derivation of the gradient for the *variance* risk measure. The authors present an actor-critic algorithm that optimizes the variance penalized expected return. The approach is novel being the first actor-critic framework that, through the newly introduced concept of *compatible features*, extends the PGT to optimize the true objective function through gradients that require a single trajectory for the estimation. In particular, the work extends previous policy gradient algorithms [24], that simultaneously estimate the reward to go and the second moment of the reward to go, to the variance penalized objective.

The algorithm develops over a few assumptions:

- The standard MDP is modified so to always have a *terminal state* which yields a 0-valued reward and from which the agent cannot exit once entered.
- Both the state space S and the action space A are finite.
- Both the reward function and the policy gradient are deterministic and bounded.
- Each state $s \in S$ has a positive probability of being reached in an infinite number of steps.

it is based on the following definitions (some of which, have already been introduced):

$$G = \sum_{t=0}^{T-1} \gamma^t R(s_t)$$

$$J_\theta(s) = \mathop{\mathbb{E}}_{\theta}\big[G|s_0 = s\big]$$

$$V_\theta(s) = Var\theta\big[G|s_0 = s\big]$$

$$M_\theta(s) = \mathop{\mathbb{E}}_{\theta}\big[G^2|s_0 = s\big]$$

and the objective is

$$\eta(\theta) = J_\theta(s_0) - \lambda V_\theta(s_0) \tag{4.1}$$

Since we know that the variance of the return can be expressed in terms of the expected return and the expected second moment, we have that:

$$V(s_0) = \mathbb{E}[s_0^2] - (\mathbb{E}[s_0])^2 \tag{4.2}$$

$$\frac{dV}{d\theta} = \frac{d\,\mathbb{E}[s_0^2]}{d\theta} - 2\,\mathbb{E}[x]\frac{d\,\mathbb{E}[s_0]}{d\theta} \tag{4.3}$$

$$\frac{d\eta(\theta)}{d\theta} = \frac{dM_\theta(s_0)}{d\theta} - 2J_\theta(s_0)\frac{dJ_\theta(s_0)}{d\theta} \tag{4.4}$$

The next step consists in providing an extension of the policy gradient theorem that uses $J_\theta(s)$ and $M_\theta(s)$. The authors provide formulations derived from expectations over trajectories:

$$J_\theta(s_0)\frac{\partial J_\theta(s_0)}{\partial \theta_j} = \mathop{\mathbb{E}}_{\theta}\left[J_\theta(s_0)\sum_{t=0}^{\inf}\frac{\partial log\pi_\theta(a_t|s_t)}{\partial \theta_j}J_\theta(s_t, a_t)\right] \tag{4.5}$$

$$\begin{aligned}\frac{\partial M_\theta(s_0)}{\partial \theta_j} =& \mathop{\mathbb{E}}_{\theta}\left[\sum_{t=0}^{\inf} \frac{\partial log \pi_\theta(a_t|s_t)}{\partial \theta_j} M_\theta(s_t, a_t)\right] \\ &+ 2 \mathop{\mathbb{E}}_{\theta}\left[\sum_{t=0}^{\inf} \frac{\partial log \pi_\theta(a_t|s_t)}{\partial \theta_j} J_\theta(s_t, a_t) \sum_{t'=0}^{t-1}\right]\end{aligned} \tag{4.6}$$

The equations we have just seen allow us to perform a gradient update to the policy π_θ by observing a trajectory and using the estimated value functions $J_\theta(s)$ and $M_\theta(s)$. Within the *actor-critic* framework, this step is defined as the *actor update*. The role of the critic, instead, is to keep track of the value functions J and M. When the state space is very large, some approximations must be used to make the problem tractable. The paper tackles the difficulties that arise from these approximations leveraging on the concept of *compatible features*. Nevertheless, the issue lies outside the scope of this thesis and will therefore not be explored. For further information, the reader can refer to [24].

4.1.2 Entropic Risk Measure in Policy Search

Among the risk measures, we decided to test with the Inverse framework is the Entropic Risk Measure (ERM) [2]. It is worth noting that the ERM is not a Coherent Risk Measure, but rather a Convex Risk Measure (CRM) which is a less restrictive class of risk measures where the notions of Sub-additivity and Positive Homogeneity are replaced by the Convexity.

In [7], the authors derive the gradient of this measure, allowing for the applicability of policy gradient methods. This step is essential for our approach since through these gradients we can create the experts and derive the formulations that allow us to solve the Inverse problem, given the experts. Given the general sense of this measure, we can now dive into some definitions.

$$U(G) = exp(-\gamma G) \tag{4.7}$$

4.7 is the Exponential Utility function, which received much attention from various fields [27]. The risk factor γ regulates the level of risk sensitivity and depending on the sign, determines whether $U(R)$ should be minimised ($\gamma > 0$) to induce a risk-averse behavior or maximised ($\gamma < 0$) to induce a risk-seeking behavior.

$$\begin{aligned}J_{ERM}(G) &= U^{-1}\,\mathbb{E}[U(G)] \\ &= -\frac{1}{\gamma} log\,\mathbb{E}[exp(-\gamma G)]\end{aligned} \tag{4.8}$$

When $\gamma > 0$, equation 4.8 induces a risk averse behaviour and is called the Entropic Risk Measure[2] while for $\gamma < 0$ it induces a risk seeking behaviour. From this formulation, given a parametrized policy π_θ, the authors derive the log-likelihood gradient:

$$\nabla_\theta J_{ERM} = \mathbb{E}\left[\nabla_\theta log\pi_\theta \left\{-\frac{1}{\gamma}e^{-\gamma(G-\Psi_\gamma(\pi_\theta))}\right\}\right] \tag{4.9}$$

Where the term inside the curly braces is the log-partition function [26]:

$$\Psi_\gamma(\pi_\theta) = -\gamma^{-1} log \mathop{\mathbb{E}}_{\theta}[exp(-\gamma G)] \tag{4.10}$$

A very useful interpretation of 4.9 is that it can be viewed as a risk neutral policy gradient where the reward has been transformed with the exponential function:

$$G_{exp} = -\frac{1}{\gamma}e^{-\gamma(G-\Psi_\gamma(\pi_\theta))} \tag{4.11}$$

We can note that $\Psi_\gamma(\pi_\theta)$ has the role of a multiplicative baseline which smooths out the convergence of the policy search. As mentioned in chapter 3, an interesting interpretation of the Entropic Risk Measure is given by Osogami in [11] where the author highlights the strong connection between the ERM and robust MDPs.

In the next chapter, we will see how from this formulation we derived the gradients used to solve the Inverse problem for reward parameters estimation.

4.1.3 Mean - Volatility

Finally, we decided to test out our IRL pipeline with the Mean-Vola risk measure introduced in [1]. This measure differs from the previously seen Mean-Variance and ERM because it allows the agent to take into account the short-term risk in addition to the long term one. Mean-Vola is defined as the variance of the step reward under the state-occupancy measure. This formulation helps to reduce the step-wise variance of the expert while bounding the variance of the total return over trajectories. The authors provide a linear formulation of the Bellman Equations under the Mean-Vola objective function that makes it possible to implement it with policy gradient algorithms that guarantee a continuous improvement such as TRPO and PPO.

The first element we should take into consideration is the discounted state-occupancy measure induced by a policy π. This measure is the cumulative sum over time of the probability of reaching a certain state s at time-step t.

$$d_{\mu,\pi}(s) := (1-\gamma)\int_S \mu(s_0)\sum_{t=0}^{\infty}\gamma^t p_\pi(s_0 \to^t s)ds_0 \tag{4.12}$$

Where $(s_0 \xrightarrow{t} s)$ is the probability of reaching state s at time-step t, starting from state s_0 and following policy π. Following this equation, we can formally define the previously mentioned expected reward under the state occupancy measure:

$$
\begin{aligned}
J_\pi &:= (1-\gamma) \mathop{\mathbb{E}}_{\substack{s_0\sim\mu \\ a_t\sim\pi(\cdot|s) \\ s_{t+1}\sim P(\cdot|s_t,a_t)}} \left[\sum_{t=0}^{\inf} \gamma^t R(s_t,a_t)\right] \\
&= \mathop{\mathbb{E}}_{\substack{s\sim d_{\mu,\pi} \\ s\sim\pi(\cdot|s)}} [R(s,a)]
\end{aligned}
\tag{4.13}
$$

Where $(1-\gamma)$ is the normalization factor used to take into account the time-step discount γ when computing the expected return over trajectories which is $J_\pi/(1-\gamma)$.

Given equation 4.13, we can proceed to define the Reward Volatility in terms of the distribution $d_{\mu,\pi}$:

$$
\nu^2_{\mu,\pi} = \mathop{\mathbb{E}}_{\substack{s\sim d_{\mu,\pi} \\ s\sim\pi(\cdot|s)}} \left[(R(s,a) - J_\pi)^2\right] \tag{4.14}
$$

from which we derive the Mean-Vola objective function for policy π:

$$
\eta_\pi := J_\pi - \lambda \nu^2_{\mu,\pi} \tag{4.15}
$$

where the parameter $\lambda > 0$ regulates the level of risk aversion that the objective induces. An important lemma provided in the paper shows how the minimization of this measure sets an upper bound to the variance of the return, effectively showing the dual benefit of the Mean-Vola objective function.

$$
\sigma^2_\pi \leq \frac{\nu^2_{\mu,\pi}}{(1-\gamma)^2} \tag{4.16}
$$

The takeaway here is that the Reward Volatility does not account for the temporal correlations between rewards, therefore allowing for the definition of the Bellman Equations used for policy gradient methods, nevertheless, it retains the desired effect of reducing the return variance. Given these preliminaries, the authors move on to defining two more measures that allow formulating the Mean-Vola policy gradient.

$$
X_\pi(s,a) = \mathop{\mathbb{E}}_{\substack{s_{t+1}\sim\mathcal{P}(\cdot|s_t,a_t) \\ a_{t+1}\sim\pi(\cdot|s_{t+1})}} \left[\sum_{t}^{\infty} \gamma^t \big(R(s_t,a_t) - J^{\pi_\theta}_\mu\big)^2 | s,a\right], \tag{4.17}
$$

$$
\begin{aligned}
W_\pi(s,a) &= \mathop{\mathbb{E}}_{\substack{a_t\sim\pi(\cdot|s_t) \\ s_{t+1}\sim\mathcal{P}(\cdot|s_t,a_t)}} \left[\sum_{t}^{\infty} \gamma^t \big(R(s_t,a_t) - J^{\pi}_\mu\big)^2 | s\right] \\
&= \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)} \Big[X_\pi(s,a)\Big]
\end{aligned}
\tag{4.18}
$$

The newly introduced $X_\pi(s,a)$ is the *action-volatility*, which corresponds to the volatility we expect to observe starting from state s and performing action a and following π thereafter. $W_\pi(s)$ is the *state-volatility*, of which the meaning is evident from X_π.

Following the derivation of [1], we can write the gradient of $\eta_\mu^{\pi_\theta}$ using a trade-off between the action-value Q_{π_θ} and the action-volatility X_{π_θ}.

$$Q_\pi^\lambda(s,a) := Q_\pi(s,a) - \lambda X_\pi(s,a) \tag{4.19}$$

The Mean-Vola objective function gradient can be now written as:

$$\nabla \eta_\mu^{\pi_\theta} = \mathop{\mathbb{E}}_{\substack{s_0 \sim \mu \\ a_t \sim \pi_\theta(\cdot|s_t) \\ s_{t+1} \sim \mathcal{P}(\cdot|s_t,a_t)}} \Big[\nabla \log \pi_\theta(a_t|s_t) Q_\pi^\lambda(s_t,a_t) \Big], \tag{4.20}$$

for which in [1], the authors provide an unbiased estimator:

$$\widehat{\nabla}_N \eta_\mu^{\pi_\theta} = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{t=0}^{T-1} \gamma^t \left(\sum_{t'=t}^{T-1} \gamma^{t'-t} \left[R_{t'}^i - \lambda \frac{1-\gamma}{1-\gamma^T} (R_{t'}^i - \hat{J})^2 \right] \right) \nabla_\theta \log \pi_\theta(a_t^i|s_t^i) \tag{4.21}$$

4.2 Inverse RL

In this section, we will present some of the approaches that have been proposed recently in the field of Inverse Reinforcement Learning. The works we will talk about are not tied to the solution presented in this thesis but serve the purpose of giving a general idea of how this space of research is being explored.

4.2.1 Maximum Entropy Inverse Reinforcement Learning

The work [30] presented by Ziebart et al. explores a novel approach for the Inverse RL problem, addressing a very important ambiguity issue that had not been confronted previously. The authors follow a probabilistic approach, based on the concept of *maximum entropy* to recover the parameters of the utility function optimized by an expert, given a set of demonstrations. More generally, the aim is to find the parameters that induce a behavior which is similar to the one observed.

Every state of the MDP, where the expert demonstrated its behaviour, is described by a set of features. Along a trajectory, many states are visited and the features observed can be summed up to form the trajectories' *feature count*. Using a state-only dependant reward function, its formulation with respect to the feature count follows immediately:

$$\boldsymbol{f}_\tau = \sum_{s_t \in \tau} \boldsymbol{f}_{s_t} \tag{4.22}$$

$$G(\boldsymbol{f}_\tau) = \omega^T \boldsymbol{f}_\tau = \sum_{s_t \in \tau} \omega^T \boldsymbol{f}_{s_t} \tag{4.23}$$

$$\tilde{\boldsymbol{f}} = \frac{1}{N} \sum_{i=0}^{N-1} \boldsymbol{f}_{\tau_i} \tag{4.24}$$

Given a large set of trajectories, as required to solve the IRL problem, one can define the *empirical feature count* as the average observed feature count. It is worth noting that the features do not univocally identify a state, and therefore the feature count does not identify a single trajectory but rather a set of possible trajectories. Consequently, recovering the reward features ω is an ill-posed problem because many weight configurations lead to a particular distribution over trajectories and many distributions over trajectories leading to a particular feature count. This problem becomes even more evident when the expert providing the demonstrations is not behaving optimally. As a solution to this ambiguity, the authors propose that, among the possible trajectory distributions that satisfy the feature count constraint, the distribution with the highest entropy should be chosen. The idea is that the estimated weights should not induce any preference over the trajectories that is not implied by the given demonstrations. Under

this assumption, further ambiguity over distributions with the same feature count and entropy, trajectories yielding a higher reward are exponentially preferred to others.

Equation 4.25 defines the visitation probability over trajectories, depending on the *partition function* $Z(\omega)$.

$$P(\tau|\omega, T) = \frac{1}{Z(\omega, T)} e^{\omega^T \boldsymbol{f}_\tau} \prod_{s_{t+1}, s_t, a_t \in \tau} P_T(s_{t+1}|s_t, a_t) \tag{4.25}$$

When the partition function converges, this distribution over trajectories defines a stochastic policy where the probability of choosing an action in a particular state is weighted by the expected exponential return of all trajectories starting from that state-action pair. The maximisation problem is then formulated as such:

$$\omega^* = \underset{\omega}{\operatorname{argmax}}\, L(\omega) = \underset{\omega}{\operatorname{argmax}} \sum_{\text{examples}} \log P(\hat{\tau}|\omega, T) \tag{4.26}$$

The derivative of this loss function and can be expressed in terms of the feature count and of the state visitation frequencies d_s, making it possible to utilize classic gradient-based optimization methods.

$$\nabla L(\omega) = \hat{\boldsymbol{f}} - \sum_\tau P(\tau|\omega, T) \boldsymbol{f}_\tau = \hat{\boldsymbol{f}} - \sum_{s_i} d_{s_i} \boldsymbol{f}_{s_i} \tag{4.27}$$

4.2.2 Risk-sensitive Inverse Reinforcement Learning via Semi- and Non-Parametric Methods

Another interesting research work is the one carried out by Singh et al. in [19]. The idea behind it is to provide a framework for Risk-Averse Inverse Reinforcement Learning with models based on coherent risk measures. The goal is to find the expert's risk measure and reward function through the maximum likelihood approach.

Coherent Risk Measure Representation Theorem

As explained in chapter 3, Coherent Risk Measures (CRMs) are mapping functions of random variables to Real values that satisfy some suitable properties. Those properties are Monotonicity, Translation Invariance, Positive Homogeneity, and Subadditivity. One of the most frequently used CRM is the Conditional Value at Risk (CVaR):

$$CVaR_{1-\tau}(Z) = \mathbb{E}[Z|Z \geq v_\tau(Z)] \tag{4.28}$$

where $v_\tau(Z) = \inf\{z \in \mathbb{R}|P(Z \leq z) \geq \tau\}$ is the so-called Value-at-Risk, which represents the τ-quantile of the cost distribution. Intuitively, the CVaR is the average value of the

tail of the cost distribution, where the tail is delimited by the Value at Risk. The position of VaR is determined by the value of τ, where the two extremes are reached with $\tau = 0$ (VaR positioned at the smallest value of the distribution) and with $\tau = 1$ (VaR positioned at the highest value of the distribution). Since we are optimising the average of what is to the right of VaR, it is evident that with $\tau = 0$ we are optimising the average cost (risk neutrality) and with $\tau = 0$ we are minimising the highest cost (worst case criterion). Everything in between is a tradeoff between these two approaches. An interesting aspect of Coherent Risk Measures is highlighted by the representation theorem stated in [19], which shows that optimising a CRM can be interpreted as optimising the Expected Value (EV) of a random variable over several probability distributions taking the worst-case distribution. The final formulation is the following

$$\rho(Z) = \max_{\zeta \in B} \mathbb{E}_{q_\zeta}[Z] = \max_{\zeta \in B} \sum_{i=1}^{|\Omega|} p(i)\zeta(i)Z(i) \tag{4.29}$$

where Z is the usual Random variable, q_ζ is a distorted distribution $q_\zeta = p \cdot \zeta$, in which ζ is in the set B, defined as the polytope. In other words, the coherent Risk Measure can be interpreted as computing the worst-case expectation considering the distorted distribution. The goal of [19] is to estimate this set of distribution through the derivation of the polytope B. This allows one to develop an algorithm to find the cost function and the risk measure adopted by the expert.

Risk Envelope

Now let's focus on the definition of Polytope. First of all, given the $|\Omega|-$dimensional probability simplex

$$\Delta^\Omega = q \in \mathbb{R}^\Omega | \sum_{i=1}^{\Omega} q(i) = 1, q \geq 0\} \tag{4.30}$$

If we absorb the density ζ in q we can derive the following representation of the risk measure (that we can rename as polytopic risk measure):

$$\rho(Z) = \max_{q \in P} \mathbb{E}_{||}[Z] \tag{4.31}$$

in which P is a polytopic subset of the probability simplex Δ^Ω

$$P = \{q \in \Delta^\Omega | A_{ineq} q \leq b_{ineq}\} \tag{4.32}$$

where A_{ineq} and b_{ineq} determine the halfspace constraints. The polytope P is also known as risk envelope and can show different degrees of risk-aversion from risk-neutral (P=p)

to worst-case scenario ($P = \Delta^{|\Omega|}$). Thus if we find an estimation of the risk envelope we can derive the expert's behavior with respect to the risk.

Risk-Averse IRL: One step

The optimization problem of the expert can be expressed as

$$\begin{aligned} \tau^* &= \min_{a_0 \in A} \rho(C(s_0, a_0)) = \min_{a_0 \in A} \max_{q \in P} \mathbb{E}_q[C(s_0, a_0)] \\ &= \min_{a_0 \in A} \max_{q \in P} g(s_0, a_0)^T q \end{aligned} \tag{4.33}$$

As we can see, the inner optimization problem is linear in q, so the optimal value τ^* is derived when we are in the vertex of the Polytope P.
This allows us to rewrite 4.33 as

$$\min_{a_0 \in A, \tau} \tau \quad s.t. \quad \tau \geq g(s_0, a_0)^T v_i, \quad i \in \{1, ..., N_V\} \tag{4.34}$$

in which $vert(P) = \{v_i\}$ are the set of vertices of the risk envelope. Now we can use the Karush-Kuhn-Tucker (KKT) conditions to constraint where the vertices of P should be, knowing that the state-action pairs that we are considering (x^*, u^*) are optimal for problem 4.33. The derived optimization problem can be seen as an iterated halfspace pruning, in which we are approximating the risk envelope more and more precisely.
[19] shows also the case in which we don't know the cost function and so we use the same solution but with weighted features to express it.

Risk-Averse IRL: Multi step

For the multi-step case, we should analyze the behavior of the human expert with respect to the disturbance within the environment. The authors generalise the approach for the single step scenario to one where a disturbance is sampled every N steps and held constant interim. [19] split the expert's policy in two: the "prepare" phase and the "react" phase. The former is concerned with the moments preceding the disturbance, where the agent ought to find a balanced state from which it will be able to handle the disturbance. The latter instead focuses on the moments following the disturbance where the main goal is to recover and re-gain a stable regime.
For this reason, the Dynamic Risk Measures are presented. Starting from the definition of Coherent One-Step Conditional Risk Measures, with the same properties of CRMs but as a mapping $\rho_t = Z_{t+1} \to Z_t$, we can derive the following formulation

$$\begin{aligned} \rho_{t:t'} &= Z_t + \rho_t(Z_{t+1} + \rho_{t+1}(Z_{t+2} + ... + \rho_{t'-1}(Z_{t'})...)) \\ &\quad \rho_t \circ ... \circ \rho_{t'-1}(Z_t + ... + Z'_t) \end{aligned} \tag{4.35}$$

This is a really important property for the two phases structure we explained before. Indeed, denoting with $k' \in [tN, (t+1)N-1]$ the steps of the "prepare-react" policy, with t representing the stage number (a stage is one "prepare-react" sequence) and N the time-steps of the stage itself, and knowing that the disturbance is sampled at $N - n_d$ step of the considered stage, we have that the cumulative cost over times step k' is

$$C_{tN:(t+1)N-1}(x_{tN|k}, \hat{\pi}_t(\cdot)) = \sum_{k'=tN}^{(t+1)(N-1)} C(x_{k'|k}, \hat{\pi}_t(x_{k'|k})) \tag{4.36}$$

Then we can define the optimization problem of the expert as

$$\min_{\hat{\pi}_t, t\in[0,T-1]} \rho_0\Big(C_{0:N-1}(\cdot, \hat{\pi}_0) + \rho_1(C_{N:2N-1}(\cdot, \hat{\pi}_1) + ... + \rho_{T-1}(C_{(T-1)N:TN-1}(\cdot, \hat{\pi}_{T-1})))...\Big) \tag{4.37}$$

At this point, we are not able to use the KKT conditions because the problem has become non-convex. Hence, we need to, first of all, define the semi-parametric representation of the Coherent Risk Measure

$$P_r = \{v \in \Delta^L | a_j^T v \leq b(j) - r(j), j = 1, ..., M\} \tag{4.38}$$

where r is a parameter vector with dimension M.
Then, applying the Constrained Maximum Likelihood as defined in [19] we will be able to find the optimal values of these parameters r that can suitably approximate the Risk-envelope as in the one-step case.

4.2.3 Risk-Sensitive Inverse Reinforcement Learning via Gradient Methods

The work presented by Mazumdar et. al. [15] introduces an effective framework for solving the Inverse RL problem of estimating the parameters of a decision-making model from a batch of given demonstrations. The paper focuses in particular on models that accurately describe the human decision-making process although the proposed approach is extendable to other objective functions. Briefly stated, the approach consists of minimising a loss metric that measures the distance between the estimated policy and the observed behavior.

The classic RL model, where the objective is to find the policy parameters that maximise the expected cumulative value of some *utility function* of the reward over a trajectory 4.39, is transformed so to incorporate relevant aspects of the human decision-making process such as *loss aversion*, *reference point dependence*, *framing effects*.

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \mathop{\mathbb{E}}_{\pi_\theta} \left[\sum_{t=0}^{T-1} \gamma^t u\left(R(s_t, a_t)\right) \right] \tag{4.39}$$

To this end, the authors make use of a particular class of value functions that effectively capture some of these aspects: the value functions of the *Prospect Theory*. The general definition for the value function in *Prospect Theory* is defined as follows:

$$u(y) = \begin{cases} c_+(y - y_0)^{\rho_+}, & y > y_0 \\ -c_-(y - y_0)^{\rho_-}, & otherwise \end{cases} \tag{4.40}$$

where y_0 is the reference point that the decision-maker compares the outcome to evaluate its success, while the parameters c and ρ determine the level of risk aversion that the value function induces the agent to. Furthermore, the authors introduce the *valuation functions*, which extend the classical value functions to better capture risk sensitivity and, in particular, employ the *shortfall valuation*. The authors generate several *risk-sensitive Q-learning agents* that model human decision-makers, presenting different levels of risk aversion.

These definitions allow for the introduction of a new set of generalized Bellman equations for *risk sensitive Q-learning*, through which the agents are trained to generate the experts. Given a set of experts demonstrating different levels of risk aversion, the inverse process can begin. Since the method proposed is based on gradient descent, there must be some metric of performance over which to compute the gradients. In particular, the authors define a loss function which depends on the parameters we want to optimize (in this case the parameters θ of the policy π) and the available data, i.e. the batch of trajectories generated from the experts $\mathscr{D}$.

$$\ell(\boldsymbol{\pi_\theta}; \mathscr{D}) \tag{4.41}$$

The authors then proceed to define two suitable loss functions: the *negative log-likelihood* of the demonstrated behaviour 4.42 and the *Kullback Leibler divergence* 4.43 between the empirical distribution of state-action pairs (given by the batch of trajectories) and the theoretical distribution induced by the estimated policy.

$$\ell(\boldsymbol{\theta}) = \sum_{(s,a)\in\mathscr{D}} \frac{n(s,a)}{N} log(\boldsymbol{\pi_\theta}(x,a)) \tag{4.42}$$

$$\ell(\boldsymbol{\theta}) = \sum_{s\in\mathscr{D}} D_{KL}(\hat{\boldsymbol{\pi}}(\cdot|s)||\boldsymbol{\pi_\theta}(\cdot|s)) \tag{4.43}$$

Both of these measures revolve around the idea of measuring a distance between the expected behavior of the estimated policy and the behavior demonstrated by the experts.

4.2.4 Gradient based Inverse Reinforcement Learning (GIRL)

The following paper is the foundation on which the work presented in this thesis builds upon. The authors of the GIRL algorithm [13] present a novel and effective approach to the IRL problem in the case of objective functions that are linear w.r.t. the reward features. As we shall see later on we extend the approach to the nonlinear space.

The main idea behind GIRL is that, given a policy that is optimal for a certain reward, then the log policy gradient computed with the said reward function w.r.t. the objective function should be identically equal to zero. In a common RL problem, the reward function is known and the policy can be found through gradient descent algorithms. The situation is the opposite in the case of IRL where the reward must be estimated, given the fixed policy of the expert. The authors solve the problem by applying the same principle, finding a reward function that yields zeroed out gradients of the known policy. The reward is computed by minimizing the L2 norm of the policy gradient calculated with the estimated reward. The key advantages of this approach are that the optimization problem is quadratic, given reasonable assumptions on the reward that will be discussed in further detail, and that it does not need to solve the forward MDP problem and it is completely model-free. This means that all that is needed for the reward estimation is a batch of trajectories provided by the expert. Since the problem is solved in a single step, collecting a large batch becomes feasible and this translates into accurate gradient estimation and reliable results. Furthermore, the whole process is extremely fast compared to approaches that require estimation of the model or solution of the forward RL problem.

As we said, the GIRL algorithm aims at estimating the parameters of the reward function of an MDP, by observing the demonstrations (trajectories) of an expert agent. To understand how the algorithm works, we should first observe how the expert's behaviour is characterized.
Given a MDP and and agent's policy π, we can evaluate the performance of the policy as such:

$$J_D(\pi) = \int_S \sum_{t=0}^{\infty} \gamma^t Pr(s_t = s|s_0, \pi) \int_A \pi(a|s) R(a, s; \boldsymbol{\omega}) da ds \tag{4.44}$$

The J_D value gives us the expected cumulative reward over an infinite trajectory, starting

from state s_0. The definition can be easily extended to the finite horizon case. Given a parametrized policy π_θ, differentiable in θ, we can write the gradient of J_D with respect to π_θ and $R(\cdot, \mathbf{w})$ as:

$$\nabla_\theta J_D(\pi_\theta, \mathbf{w}) = \int_S \sum_{t=0}^{T-1} \gamma^t Pr(s_t = s|s_0, \pi) \int_A \nabla_\theta \pi_\theta(a|s) Q^\pi(a, s; \boldsymbol{\omega}) dads \tag{4.45}$$

Given that we are using a model-free approach, the gradients of the policy must be estimated through a set of the expert's trajectories. This can be achieved using any standard policy gradient approach such as REINFORCE [28] or A3C [6]. For this work, we will be using the GPOMDP algorithm.

$$\widehat{g}_{GPOMDP} = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{t=0}^{T-1} \nabla_\theta log \pi_\theta(s_t^{(m)}, a_t^{(m)}) \left(\sum_{t'=1}^{T-1} \gamma^{t'} r_{t'+1}^{(m)} - b \right) \tag{4.46}$$

If we assume that the expert's policy π_θ^E is optimal with respect to the reward function $R(s, a, \boldsymbol{\omega}^E)$ then we can state that the associated policy gradient $\nabla_\theta J$ will be identically equal to zero.

$$\nabla_\theta J_D(\pi_\theta^E, \boldsymbol{\omega}) = 0 \tag{4.47}$$

Given that fact that the expert's policy π_θ^E is a stationary point for $\nabla_\theta J$, the goal of the GIRL algorithm is to find the unknown reward weights $\boldsymbol{\omega}$ that yield that same stationary point. This is achieved by solving the following minimization problem:

$$\boldsymbol{\omega}^A = \underset{\omega}{\operatorname{argmin}}\, C_x^y(\pi_\theta^E, \boldsymbol{\omega}) = \underset{\omega}{\operatorname{argmin}}\, \frac{1}{y} \left\| \nabla_\theta J(\pi_\theta^E, \boldsymbol{\omega}) \right\|_x^y \tag{4.48}$$

As previously hinted, one of the fundamental aspects of the GIRL algorithm is that the only requirement for this minimization problem to be convex is that the reward function $R(a, s, \boldsymbol{\omega})$ is convex with respect to the reward parameters $\boldsymbol{\omega}$. This restriction reduces the admissible class of rewards but it is loose enough to still encapsulate many practical applications.

4.2.5 In Case of Sub-optimality

It is worth to note that the expert providing the demonstrations may not be completely optimal with respect to the reward function $R(\cdot, w)$. This may be the case for several reasons. The first possibility is that the actual reward function optimized by the agent lies outside the restricted class of convex functions that we confined our search to. Another scenario is the one in which the expert simply has not reached convergence. The GIRL approach successfully overcomes these obstacles. The intuitive interpretation of

the algorithm is that the solution to the minimization problem will yield with the reward weights $\mathbf{w}$ that best fit the behavior demonstrated by the Expert. Nevertheless, the approximation could lead to unreliable results. For the solution to be considered reliable, it is required that the norm of the gradient does not exceed a given threshold which is problem-dependent.

4.2.6 The Linear-Reward Risk-Neutral Setting

When the reward can be expressed as a linear combination of basis functions, the estimation of the weights that multiply these basis functions, so as to minimize the $L2$ norm of the objective function gradient, reduces to a simple quadratic problem. To put this into formulas we have:

$$R(s, a, \boldsymbol{\omega}) = \boldsymbol{\omega} \cdot \boldsymbol{\phi}(s, a) \tag{4.49}$$

In this case we can rewrite Q in this way:

$$\begin{aligned} Q_{\pi_\theta}(s, a) &= \mathop{\mathbb{E}}_{\substack{a_t \sim \pi_\theta(\cdot|s_t) \\ s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)}} \left[\sum_t^\infty \gamma^t \sum_{i=1}^q \omega_i \phi_i(s_t, a_t) | s_0 = s, a_0 = a \right] \\ &= \sum_{i=1}^q \omega_i \mathop{\mathbb{E}}_{\substack{a_t \sim \pi_\theta(\cdot|s_t) \\ s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)}} \left[\sum_t^\infty \gamma^t \phi_i(s_t, a_t) | s_0 = s, a_0 = a \right], \end{aligned} \tag{4.50}$$

and the optimization can be written as:

$$\begin{aligned} \boldsymbol{\omega}^E &= \operatorname*{argmin}_{\boldsymbol{\omega}} \frac{1}{y} \left\| \nabla_\theta J_\theta \right\|_x^y \\ &= \operatorname*{argmin}_{\boldsymbol{\omega}} \frac{1}{y} \left\| \mathop{\mathbb{E}}_{\substack{s_0 \sim \mu \\ a_t \sim \pi_\theta(\cdot|s_t) \\ s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)}} \left[\sum_t^\infty = \nabla_\theta \log \pi_\theta(a_t|s_t) Q_{\pi_\theta}(s_t, a_t) \right] \right\|_x^y \\ &= \operatorname*{argmin}_{\boldsymbol{\omega}} \frac{1}{y} \left\| \mathop{\mathbb{E}}_{\substack{s_0 \sim \mu \\ a_t \sim \pi_\theta(\cdot|s_t) \\ s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)}} \left[\sum_t^\infty \nabla_\theta \log \pi_\theta(a_t|s_t) \mathop{\mathbb{E}}_{\substack{a_k \sim \pi_\theta(\cdot|s_k) \\ s_{k+1} \sim \mathcal{P}(\cdot|s_k, a_k) \\ s_0 = s_t, a_0 = a_t}} \left[\sum_{k=t}^\infty \gamma^{k-t} \boldsymbol{\phi}(s_k, a_k) \right] \cdot \boldsymbol{\omega} \right] \right\|_x^y \end{aligned} \tag{4.51}$$

The objective is linear in $\boldsymbol{\omega}$, then the problem is convex. In particular, if the expert is optimal and the real reward is linear, it is then possible to solve the problem in closed form.

Chapter 5

Risk Averse Gradient Based Inverse Reinforcement Learning

The following chapter is dedicated to explaining the theoretical principles behind the RA-GIRL algorithm proposed in this thesis. RA-GIRL comes as a direct extension of the GIRL algorithm [13] presented in chapter 4. The paper introduces a fast, model-free and sample efficient approach to the IRL problems that yields promising results when dealing with objective functions that are linear in the reward. Our contribution is to extend the approach to the nonlinear space of objective functions to include risk measures such as the Variance. This extension brings a new approach to the field of risk-averse RL which is still greatly unexplored.

It is important to note that the work we present is based on two strong assumptions: Both the policy and the objective function must be known a priori. Nevertheless, the policy can be easily estimated through MLE.

5.1 Mean Variance with GIRL

Having presented the founding methods we studied and adopted it is time to expose the actual theoretical solution we reached. As we explained before, the GIRL algorithm [13] works within a risk-neutral setting, and its goal is to derive the unknown reward function given the optimal policy of an expert. What we want to do is to move to a risk-averse setting, where the expert is supposed to optimize some risk measure instead of the usual expected return. In particular, the *mean-variance* objective function, which induces the agent to adopt a behavior that optimizes a tradeoff between the expected return and its variance. This trade-off depends on the parameter of risk aversion λ. The first task will be to estimate λ, while knowing everything else (the policy and the reward). The

second task will be to derive both the parameter and the reward function given only the expert policy.

5.1.1 Risk-Averse optimization

The first step required is to generate the risk-averse expert. To do this, we used the Mean-Variance gradients derived by Tamar [22]. The expert is obtained by training an agent through any gradient descent approach (such as GPOMDP) with the Mean-Variance objective function, until convergence. As the agent gets closer to convergence, the accuracy and reliability of RA-GIRL improves drastically since it relies on the assumption of zeroed-out gradients (total convergence). It is worth to note that the forward problem is decoupled from the inverse one. The expert could be generated through any optimization approach as long as it is optimising the same risk measure for which we compute the gradients for the Inverse problem. Furthermore, as we will see, the approach is easily extendable to other risk measures.

5.1.2 Estimating the Gradient of the Variance

The objective is to estimate the parameters of the environment's reward when the expert is optimising the Mean-Variance objective function. To solve this problem using the GIRL algorithm, it is first necessary to express the gradient of the mean-variance objective function. Tamar's publication [25] offers a clear formulation for this by extending the policy evaluation algorithms proposed in [22] to the variance-penalized objective function. To this purpose, let us first introduce some new notations:

Definition 5.1.1. (Variance of the reward to go) The Variance of the reward to go is the expected variance of the cumulative discounted return, following a given policy.

$$V^{\theta}(s) = Var^{\theta}[G_t|s_0 = s] \tag{5.1}$$

To derive the gradient of the variance, we also need to define the second moment of the reward to go.

Definition 5.1.2. (Second Moment of the reward to go) The Second Moment of the reward to go is the expected squared cumulative discounted return, following a given policy.

$$M^{\theta}(s) = E^{\theta}[G_t^2|s_0 = s] \tag{5.2}$$

The goal of our risk-sensitive agent is to find a policy that optimizes the discounted return-variance trade-off, parametrized by the risk aversion parameter λ which controls the weight associated with the variance penalty.

$$\eta(\theta) = J^{\theta}(s_0) - \lambda V^{\theta}(s) \tag{5.3}$$

The gradient of the expected reward is well known and can be easily estimated from a batch of trajectories using the GPOMDP algorithm mentioned earlier. The gradient of the variance can be found starting from the following relation:

$$\begin{aligned} V &= E[G^2] - (E[G])^2 \\ &= M - J^2 \end{aligned} \tag{5.4}$$

Therefore we have that

$$\frac{\partial V^{\theta}}{\partial \theta} = \frac{\partial M^{\theta}(x_0)}{\partial \theta} - 2J^{\theta}(x_0)\frac{\partial J^{\theta}(x_0)}{\partial \theta} \tag{5.5}$$

The gradient we just introduced can be estimated from trajectories.

$$J^{\theta}(x_0)\frac{\partial J^{\theta}(x_0)}{\partial \theta} = \frac{1}{N}\sum_{i=0}^{N-1}\left[J^{\theta}(x_0)\sum_{t=0}^{T-1}\frac{\partial log\pi_{\theta}(u_t|x_t)}{\partial \theta}J^{\theta}(x_t,u_t)\right] \tag{5.6}$$

$$\begin{aligned} \frac{\partial M^{\theta}(x_0)}{\partial \theta} = \quad & \frac{1}{N}\sum_{i=0}^{N-1}\left[\sum_{t=0}^{T-1}\gamma^{2t}\frac{\partial log\pi_{\theta}(u_t|x_t)}{\partial \theta}M^{\theta}(x_t,u_t)\right] \\ & + 2\gamma\frac{1}{N}\sum_{i=0}^{N-1}\left[\sum_{t=1}^{T-1}\left(\gamma^{t}\frac{\partial log\pi_{\theta}(u_t|x_t)}{\partial \theta}J^{\theta}(x_t,u_t)\sum_{t'=0}^{t-1}\gamma^{2t}r(x_{t'})\right)\right] \end{aligned} \tag{5.7}$$

5.1.3 Estimating risk parameter λ

As we said, the first task deals with the optimization of the risk-averse parameter that in the case of Mean-Variance optimization is represented by λ. Indeed, it is the parameter that balances the importance to give to the Variance in respect of the Mean Value in the gradient computation.

Knowing both the policy of the expert (found by the GPOMD in the forward problem) and the reward function we need to derive a formula to find the approximated value of λ. Starting from the mean-variance gradient, we have

$$\nabla_{\theta}L(\theta,\lambda) = \nabla_{\theta}J(\theta) - \lambda\nabla_{\theta}V(\theta) \tag{5.8}$$

for which we want to find the minimum w.r.t. λ

$$\begin{aligned}\min_{\lambda} ||\nabla_\theta L(\theta,\lambda)||_2^2 &= \min_{\lambda} \sum_d (\nabla_{\theta_d} L(\theta,\lambda))^2 \\ &= \min_{\lambda} \sum_d (\nabla_{\theta_d} J(\theta) - \lambda \nabla_{\theta_d} V(\theta))^2 \\ &= \min_{\lambda} F(\lambda)\end{aligned} \tag{5.9}$$

At this point, to minimize the function we have to compute the gradient of the function $F(\lambda)$ with respect to λ itself. The closed-form of this minimization problem follows immediately by imposing the zero-equality. We can observe how the problem only depends on the unchanged gradients of the average Return J and of the Variance V.

$$\begin{aligned}\frac{dF(\lambda)}{d\lambda} &= -2 \sum_d (\nabla_{\theta_d} J(\theta) - \lambda \nabla_{\theta_d} V(\theta)) \nabla_{\theta_d} V(\theta) \\ &\implies \sum_d \nabla_{\theta_d} J(\theta)(\nabla_{\theta_d} V(\theta)) - \lambda \sum_d \nabla^2_{\theta_d} V(\theta) = 0 \\ &\implies \lambda = \frac{\sum_d \nabla_{\theta_d} J(\theta) \cdot \nabla_{\theta_d} V(\theta)}{\sum_d \nabla^2_{\theta_d} V(\theta)} \\ &\implies \frac{\sum_d \nabla_{\theta_d} J(\theta) \cdot \nabla_{\theta_d} V(\theta)}{||\nabla^2_{\theta_d} V(\theta)||}\end{aligned} \tag{5.10}$$

Having the possibility to compute the parameter in a closed-form is a good advantage both for the simplicity of the computation and for the precision. The gradients computed for the inverse problem can be estimated from a single very large batch of trajectories since it is a 1-step procedure.

5.1.4 Estimating reward weights and risk parameter

Our second inquiry explores the possibility of finding the reward function used by a Risk-Averse expert, observing only some trajectories sampled from its policy. We pose ourselves in the case where the reward function is a linear combination of basis functions (reward features). We want to know how to approximate the weights of the features of this reward, together with the risk aversion parameter λ, knowing that the objective function is now in the form of the well known Mean-Variance. The difficulty that arises from this formulation is that the mean-variance objective function is no longer linear with respect to the basis functions of the reward. This implies that the minimization problem is not quadratic and cannot be solved with the straight forward methods we have used so far.

As in the risk neutral case, we are again minimizing the norm of the estimated gradient of the objective function. Since we are still considering the linear reward case, it is convenient to re-write M in the following way:

$$\begin{aligned} M_{\pi_\theta}(s,a) &= \mathop{\mathbb{E}}_{\substack{a_t\sim\pi_\theta(\cdot|s_t)\\ s_{t+1}\sim\mathcal{P}(\cdot|s_t,a_t)}} \left[\left(\sum_t^\infty \gamma^t \sum_{i=1}^q \omega_i \phi_i(s_t) \right)^2 | s_0 = s, a_0 = a \right] \\ &= \mathop{\mathbb{E}}_{\substack{a_t\sim\pi_\theta(\cdot|s_t)\\ s_{t+1}\sim\mathcal{P}(\cdot|s_t,a_t)}} \left[\left(\sum_{i=1}^q \omega_i \sum_t^\infty \gamma^t \phi_i(s_t) \right)^2 | s_0 = s, a_0 = a \right]. \end{aligned} \tag{5.11}$$

We can now write the optimization problem as:

$$\begin{aligned} \boldsymbol{\omega}^E &= \operatorname*{argmin}_{\boldsymbol{\omega}} \frac{1}{y} \left\| \nabla_\theta J_\theta - \lambda(\nabla_\theta M - 2J\nabla_\theta J) \right\|_x^y \\ &= \operatorname*{argmin}_{\boldsymbol{\omega}} \frac{1}{y} \Bigg\| \mathop{\mathbb{E}}_{\substack{s_0\sim\mu\\ a_t\sim\pi_\theta(\cdot|s_t)\\ s_{t+1}\sim\mathcal{P}(\cdot|s_t,a_t)}} \left[\sum_t^\infty \nabla_\theta \log \pi_\theta(a_t|s_t) Q_{\pi_\theta}(s_t,a_t) \right] \\ &\quad -\lambda \Bigg(\mathop{\mathbb{E}}_{\substack{s_0\sim\mu\\ a_t\sim\pi_\theta(\cdot|s_t)\\ s_{t+1}\sim\mathcal{P}(\cdot|s_t,a_t)}} \left[\sum_t^\infty \nabla_\theta \log \pi_\theta(a_t|s_t)\ M_\theta(s_t,a_t) \right] \\ &\qquad + 2\gamma \mathop{\mathbb{E}}_{\substack{s_0\sim\mu\\ a_t\sim\pi_\theta(\cdot|s_t)\\ s_{t+1}\sim\mathcal{P}(\cdot|s_t,a_t)}} \left[\sum_{t=1}^\infty \left(\nabla_\theta \log \pi_\theta(a_t|s_t) Q_\theta(s_t,a_t) \sum_{k=0}^{t-1} \gamma^{2t} R(s_k) \right) \right] \\ &\qquad - 2J_\theta \mathop{\mathbb{E}}_{\substack{s_0\sim\mu\\ a_t\sim\pi_\theta(\cdot|s_t)\\ s_{t+1}\sim\mathcal{P}(\cdot|s_t,a_t)}} \left[\sum_t^\infty \nabla_\theta \log \pi_\theta(a_t|s_t) Q_\theta(s_t,a_t) \right] \Bigg) \Bigg\|_x^y \end{aligned}$$

The PGT estimator can be trivially derived by approximating the expectations through sampling as seen before.

$$\begin{aligned}\boldsymbol{\omega}^E = \underset{\boldsymbol{\omega}}{\operatorname{argmin}} \frac{1}{y}\Bigg\| & \frac{1}{N}\sum_{i=0}^{N-1}\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\boldsymbol{\omega}\cdot\sum_{k=t}^{T-1}\gamma^{k-t}\boldsymbol{\phi}(s_k)\right] \\ -\lambda\Bigg(& \frac{1}{N}\sum_{i=0}^{N-1}\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\ \left(\boldsymbol{\omega}\cdot\sum_{k=t}^{T-1}\gamma^{k-t}\boldsymbol{\phi}(s_k)\right)^2\right] \\ & + 2\gamma\frac{1}{N}\sum_{i=0}^{N-1}\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_k|s_k)\quad \left(\boldsymbol{\omega}\cdot\sum_{k=t}^{T-1}\gamma^{k-t}\boldsymbol{\phi}(s_k)\right)\left(\boldsymbol{\omega}\cdot\sum_{h=0}^{t-1}\gamma^{2h}\boldsymbol{\phi}(s_h)\right)\right] \\ & - 2J_\theta\frac{1}{N}\sum_{i=0}^{N-1}\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\quad \boldsymbol{\omega}\cdot\sum_{k=t}^{T-1}\gamma^{k-t}\boldsymbol{\phi}(s_k)\right]\Bigg)\Bigg\|_x^y\end{aligned} \tag{5.12}$$

Given that we are considering the squared Euclidean Norm, we are minimizing a function of the form:

$$f(g(\theta,\omega)) = \|g(\theta,\omega)\|_2^2 = \left(\sqrt{g(\theta_i,\omega)^2 + ... + g(\theta_n,\omega)^2}\right)^2 = \sum_i g(\theta_i,\omega)^2 \tag{5.13}$$

We have that

$$\begin{aligned} g(\theta_j,\boldsymbol{\omega}) = & \frac{1}{N}\sum_{i=0}^{N-1}\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\boldsymbol{\omega}\cdot\sum_{k=t}^{T-1}\gamma^{k-t}\boldsymbol{\phi}(s_k)\right] \\ & -\lambda\Bigg(\frac{1}{N}\sum_{i=0}^{N-1}\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\ \left(\boldsymbol{\omega}\cdot\sum_{k=t}^{T-1}\gamma^{k-t}\boldsymbol{\phi}(s_k)\right)^2\right] \\ & + 2\gamma\frac{1}{N}\sum_{i=0}^{N-1}\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_k|s_k)\quad \left(\boldsymbol{\omega}\cdot\sum_{k=t}^{T-1}\gamma^{k-t}\boldsymbol{\phi}(s_k)\right)\left(\boldsymbol{\omega}\cdot\sum_{h=0}^{t-1}\gamma^{2h}\boldsymbol{\phi}(s_h)\right)\right] \\ & - 2J_\theta\frac{1}{N}\sum_{i=0}^{N-1}\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\quad \boldsymbol{\omega}\cdot\sum_{k=t}^{T-1}\gamma^{k-t}\boldsymbol{\phi}(s_k)\right]\Bigg)\end{aligned} \tag{5.14}$$

Grouping common factors, we can reduce this formula to a more compact form:

$$
\begin{aligned}
&g(\theta_j, \boldsymbol{\omega}) = \\
&\frac{1}{N}\sum_{i=0}^{N-1}\left[\sum_{t=0}^{T-1}\left[\frac{\partial \log \pi_\theta(a_t|s_t)}{\partial \theta_j}\left(\left(\boldsymbol{\omega}\cdot\sum_{k=t}^{T-1}\gamma^{k-t}\boldsymbol{\phi}(s_k)\right)\right.\right.\right. \\
&\left.\left.\left.\left(1+2\lambda\left(\boldsymbol{\omega}\cdot\frac{1}{N}\sum_{i'=0}^{N-1}\sum_{t'=0}^{T-1}\gamma_{i'}^{t'}\boldsymbol{\phi}(s_{i't'})\right)-\lambda\gamma^t\left(\boldsymbol{\omega}\cdot\sum_{k=t}^{T-1}\gamma^{k-t}\boldsymbol{\phi}(s_k)\right)-2\lambda\gamma\left(\boldsymbol{\omega}\cdot\sum_{h=0}^{t-1}\gamma^{2h}\boldsymbol{\phi}(s_h)\right)\right)\right)\right]\right]
\end{aligned}
\tag{5.15}
$$

where the derivative of g w.r.t $\boldsymbol{\omega}$ can be written as:

$$
\begin{aligned}
&\frac{\partial}{\partial \omega_l} g(\theta_j, \boldsymbol{\omega}) = \\
&\frac{1}{N}\sum_{i=0}^{N-1}\left[\sum_{t=0}^{T-1}\left[\frac{\partial \log \pi_\theta(a_t|s_t)}{\partial \theta_j}\left(\right.\right.\right. \\
&\left(\sum_{k=t}^{T-1}\gamma^{k-t}\phi_l(s_k)\right) \\
&\left(1+2\lambda\left(\boldsymbol{\omega}\cdot\frac{1}{N}\sum_{i'=0}^{N-1}\sum_{t'=0}^{T-1}\gamma_{i'}^{t'}\boldsymbol{\phi}(s_{i't'})\right)-\lambda\gamma^t\left(\boldsymbol{\omega}\cdot\sum_{k=t}^{T-1}\gamma^{k-t}\boldsymbol{\phi}(s_k)\right)-2\lambda\gamma\left(\boldsymbol{\omega}\cdot\sum_{h=0}^{t-1}\gamma^{2h}\boldsymbol{\phi}(s_h)\right)\right) \\
&+\left(\boldsymbol{\omega}\cdot\sum_{k=t}^{T-1}\gamma^{k-t}\boldsymbol{\phi}(s_k)\right) \\
&\left.\left.\left.\left(2\lambda\left(\frac{1}{N}\sum_{i'=0}^{N-1}\sum_{t'=0}^{T-1}\gamma_{i'}^{t'}\phi_l(s_{i't'})\right)-\lambda\gamma^t\left(\sum_{k=t}^{T-1}\gamma^{k-t}\phi_l(s_k)\right)-2\lambda\gamma\left(\sum_{h=0}^{t-1}\gamma^{2h}\phi_l(s_h)\right)\right)\right)\right]\right].
\end{aligned}
\tag{5.16}
$$

And its derivative w.r.t λ can be written as:

$$
\begin{aligned}
&\frac{\partial}{\partial \lambda} g(\theta_j, \boldsymbol{\omega}) = \\
&\frac{1}{N}\sum_{i=0}^{N-1}\left[\sum_{t=0}^{T-1}\left[\frac{\partial \log \pi_\theta(a_t|s_t)}{\partial \theta_j}\left(\left(\boldsymbol{\omega}\cdot\sum_{k=t}^{T-1}\gamma^{k-t}\boldsymbol{\phi}(s_k)\right)\right.\right.\right. \\
&\left.\left.\left.\left(2\left(\boldsymbol{\omega}\cdot\frac{1}{N}\sum_{i'=0}^{N-1}\sum_{t'=0}^{T-1}\gamma_{i'}^{t'}\boldsymbol{\phi}(s_{i't'})\right)-\gamma^t\left(\boldsymbol{\omega}\cdot\sum_{k=t}^{T-1}\gamma^{k-t}\boldsymbol{\phi}(s_k)\right)-2\gamma\left(\boldsymbol{\omega}\cdot\sum_{h=0}^{t-1}\gamma^{2h}\boldsymbol{\phi}(s_h)\right)\right)\right)\right]\right]
\end{aligned}
\tag{5.17}
$$

We can now write a birds-eye view of the minimization problem:

$$\boldsymbol{\omega^E} = \underset{\boldsymbol{\omega}}{\operatorname{argmin}} \sum_j g(\theta_j, \boldsymbol{\omega})^2 \tag{5.18}$$

for which the gradient w.r.t. $\boldsymbol{\omega}$ is

$$\frac{\partial}{\partial \omega_l} f(\boldsymbol{\theta}, \omega_l) = \sum_j 2g(\theta_j, \boldsymbol{\omega}) \frac{\partial}{\partial \omega_l} g(\theta_j, \omega_l) \tag{5.19}$$

5.2 ERM with GIRL

Just as the Mean-Variance objective function, the ERM allows us to control the tradeoff between average return and its variance over multiple trajectories. It can be seen as a different representation of Mean-Variance with gradients that are easier to compute. Following the formulations presented in [7], we derive the gradients with respect to the weights of the reward features that allow us to estimate these weights through the RA-GIRL framework.

As a reminder, these are the formulations presented in [7], already seen in chapter 4.

$$\begin{aligned} J_{ERM}(R) &= U^{-1}\,\mathbb{E}[U(R)] \\ &= -\frac{1}{\lambda} log\,\mathbb{E}[exp(-\lambda R)] \end{aligned} \tag{4.8}$$

$$\nabla_\theta J_{ERM} = \mathbb{E}\left[\nabla_\theta log \pi_\theta \left\{-\frac{1}{\lambda} e^{-\lambda(R - \Psi_\lambda(\pi_\theta))}\right\}\right] \tag{4.9}$$

$$\Psi_\lambda(\pi_\theta) = -\frac{1}{\lambda} log\,\mathbb{E}[exp(-\lambda G)] \tag{5.20}$$

Note that the *log-partition function* $\Psi_\lambda(\pi_\theta)$ is actually equal to J_{ERM}. We can therefore re-write the gradient as:

$$\nabla_\theta J_{ERM} = \mathbb{E}\left[\nabla_\theta log \pi_\theta \left\{-\frac{1}{\lambda} e^{-\lambda(R - J_{ERM})}\right\}\right] \tag{4.9}$$

As usual, we can estimate the expectation by averaging over a batch of sampled trajectories using a reinforce-like estimator:

$$\hat{\nabla}_\theta J_{ERM} = \frac{1}{N} \sum_{i=0}^{N-1} \left[\left(\sum_{t=0}^{T-1} \nabla_\theta log \pi_t^i\right)\left(-\frac{1}{\lambda} \exp(-\lambda(G(\tau_i) - J_{ERM})\right)\right] \tag{5.21}$$

And the same goes for the log-partition function:

$$\hat{J}_{ERM} = -\frac{1}{\lambda} log \frac{1}{N} \sum_{i=0}^{N-1} [exp(-\lambda G(\tau_i))] \tag{5.22}$$

where, for both equations, $G(\tau_i)$ is the cumulative return over the trajectory τ_i.

Now, as seen for the Mean Variance objective, the goal is to find the reward feature weights that minimize the L2 norm of the gradient:

$$\begin{aligned} \omega^* &= \underset{\omega}{\operatorname{argmin}} \, \|g(\theta, \boldsymbol{\omega})\|_2^2 \\ &= \underset{\omega}{\operatorname{argmin}} \sum_i g(\theta_i, \boldsymbol{\omega})^2 \end{aligned} \tag{5.23}$$

where $g = \hat{\nabla} J_{ERM}$. To do so we must compute the derivatives with respect to the weights of the reward features ω:

$$\frac{\partial g}{\partial \omega_l} = \frac{1}{N} \sum_{i=0}^{N-1} \left[\left(\sum_{t=0}^{T-1} \nabla_\theta log \pi(s_t^i, a_t^i) \right) \left(\left[\sum_{t=0}^{T-1} \gamma^t \phi_{l,t} \right] + \frac{\partial J_{ERM}}{\partial \omega_l} \right) \exp(-\lambda (G(\tau_i) - J_{ERM}) \right] \tag{5.24}$$

$$\frac{\partial J_{ERM}}{\partial \omega_l} = \frac{\sum_{i'}^{N-1} \left(\left[\sum_{t=0}^{T-1} \gamma^t \phi_{l,t} \right] \exp\big(-\lambda G(\tau_{i'}) \big) \right)}{\sum_{i'}^{N-1} \exp\big(-\lambda G(\tau_{i'}) \big)} \tag{5.25}$$

and with respect to the risk aversion parameter λ:

$$\frac{\partial g}{\partial \lambda} = \frac{1}{N} \sum_{i=0}^{N-1} \left[\left(\sum_{t=0}^{T-1} \nabla_\theta log \pi(s_t^i, a_t^i) \right) \left(\frac{1}{\lambda^2} \exp(Z(\lambda)) - \frac{1}{\lambda} \frac{\partial Z}{\partial \lambda} \exp(Z(\lambda)) \right) \right] \tag{5.26}$$

where

$$Z(\lambda) = -\lambda (G(\tau_i) - J_{ERM}) \tag{5.27}$$

$$\frac{\partial Z}{\partial \lambda} = -(G(\tau_i) - J_{ERM}) + \lambda \frac{\partial J_{ERM}}{\partial \lambda} \tag{5.28}$$

$$\frac{\partial J_{ERM}}{\partial \lambda} = \frac{\frac{1}{N} \sum_{i=0}^{N-1} log(\exp(-\lambda G(\tau_i)))}{\lambda^2} + \frac{\frac{1}{N} \sum_{i=0}^{N-1} G(\tau_i) \exp(-\lambda G(\tau_i))}{\frac{1}{N} \sum_{i=0}^{N-1} \exp(-\lambda G(\tau_i))} \tag{5.29}$$

The formulations here derived can now be plugged in equations 5.18 and 5.19 to solve for the reward feature weights.

5.3 Mean-Volatility with GIRL

Finally, we repeat the steps with the Mean-Vola risk-averse objective function. As we have seen, the volatility risk measure is very interesting for the economic scenarios we aim to test our approach because it helps to reduce step-wise variance.

We recall some formulations from chapter 4:

$$\begin{aligned} J_\pi &:= (1-\gamma) \mathop{\mathbb{E}}_{\substack{s_0\sim\mu \\ a_t\sim\pi(\cdot|s) \\ s_{t+1}\sim P(\cdot|s_t,a_t)}} \left[\sum_{t=0}^{\inf} \gamma^t R(s_t,a_t)\right] \\ &= \mathop{\mathbb{E}}_{\substack{s\sim d_{\mu,\pi} \\ s\sim\pi(\cdot|s)}} [R(s,a)] \end{aligned} \tag{4.13}$$

$$\nu^2_{\mu,\pi} = \mathop{\mathbb{E}}_{\substack{s\sim d_{\mu,\pi} \\ s\sim\pi(\cdot|s)}} \left[(R(s,a)-J_\pi)^2\right] \tag{4.14}$$

The estimator from which we will derive the necessary derivations has been provided by the authors in [1]:

$$\widehat{\nabla}_N \eta^{\pi_\theta}_\mu = \frac{1}{N}\sum_{i=0}^{N-1}\sum_{t=0}^{T-1}\gamma^t \left(\sum_{t'=t}^{T-1}\gamma^{t'-t}\left[R^i_{t'} - \lambda\frac{1-\gamma}{1-\gamma^T}(R^i_{t'}-\hat{J})^2\right]\right)\nabla_\theta \log \pi_\theta(a^i_t|s^i_t) \tag{4.21}$$

To simplify the representation, we bring the discount factor γ^t inside the time-step sum and apply the following substitutions:

$$\beta = \frac{1-\gamma}{1-\gamma^t} \tag{5.30}$$

$$\begin{aligned} A(\boldsymbol{\omega}) &= R^i_{t'} - \hat{J} \\ &= \boldsymbol{\omega}\cdot\boldsymbol{\phi^i_{t'}} - \frac{1}{N}\sum_{i''}^{N}\sum_{t''}^{T}\gamma^{t''}\boldsymbol{\omega}\cdot\boldsymbol{\phi^{i''}_{t''}} \end{aligned} \tag{5.31}$$

So that equation 4.21 becomes:

$$\widehat{\nabla}_N \eta^{\pi_\theta}_\mu = \frac{1}{N}\sum_{i=0}^{N-1}\sum_{t=0}^{T-1}\left(\sum_{t'=t}^{T-1}\gamma^{t'}\left[\boldsymbol{\omega}\cdot\boldsymbol{\phi^i_{t'}} - \lambda\beta A(\boldsymbol{\omega})^2\right]\right)\nabla_\theta \log \pi_\theta(a^i_t|s^i_t) \tag{5.32}$$

The minimization problem is the same as the one seen for the ERM:

$$\begin{aligned}\omega^* &= \underset{\omega}{\operatorname{argmin}} \|g(\theta, \boldsymbol{\omega})\|_2^2 \\ &= \underset{\omega}{\operatorname{argmin}} \sum_i g(\theta_i, \boldsymbol{\omega})^2\end{aligned} \tag{5.23}$$

Where this time $g = \widehat{\nabla}_N \eta_\mu^{\pi_\theta}$. The derivatives with respect to $\boldsymbol{\omega}$ and the risk parameter λ follow immediately:

$$\frac{\partial g}{\partial \omega_l} = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{t=0}^{T-1} \left(\sum_{t'=t}^{T-1} \gamma^{t'} \left[\phi_l - \lambda \beta 2 A(\boldsymbol{\omega}) \frac{\partial A(\boldsymbol{\omega})}{\partial \omega_l} \right] \right) \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \tag{5.33}$$

where

$$\frac{\partial A}{\partial \omega_l} = \phi_l - \frac{1}{N} \sum_{i''}^{N} \sum_{t''}^{T} \gamma^{t''} \phi_l \tag{5.34}$$

While the derivative with respect to λ is very simple since the factor only appears once in linear form:

$$\frac{\partial g}{\partial \lambda} = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{t=0}^{T-1} \left(\sum_{t'=t}^{T-1} \gamma^{t'} \left[-\beta A(\boldsymbol{\omega})^2 \right] \right) \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \tag{5.35}$$

Again, 5.18 and 5.19 can be used with the derivatives of Mean-Vola to solve the Inverse problem. It is worth to mention that, just as for the case of the Mean-Variance objective function, the parameter of risk aversion λ can be found through a closed form equation. In the case of the Mean-Volatility this equation is quadratic instead of being linear, but still offers a finite number of solutions. Nevertheless, we have not reported it here because it did not result useful in our implementations.

Chapter 6

Implementation and Results

6.1 General Domain description

The domain is where we define the Markov Decision Process of our IRL problem. This includes the state space, the action space, the transition kernel, and the reward (cost) function. We used the **Gym** library from **Openai** to implement the environments. Every gym environment must offer an interface of four methods that make it conform to the structure of a classical MDP: initialization, step, reset, and render.

- **Initialization**
  This method is invoked at the beginning of the training process. All parameters needed to set up the environment are passed and saved within the env. instance. The starting position of the agent is assigned.

- **Step**
  Intuitively, this method models the dynamics of the environment. At every time-step, the agent chooses an action and forwards this choice to the **step** function. The method returns a tuple describing the evolution of the environment that followed the agent's action: <*observation, reward, done, info*>.

  observation: as we have already seen, the observation is a limited description of the state (the observable part of it) that allows the agent to learn how its actions influence the state.

  reward: the reward is a real number that quantifies the value of the transition so that the agent can understand whether it performed a good action or not.

  done: this is a flag that tells us whether the agent has reached the end of the simulation. Some environments have episodes of fixed length (and therefore the flag is redundant) and environments where the end of the simulation depends on the state of the agent (variable episode length that makes the flag necessary).

info: this is a dictionary containing diagnostics information that is useful to the developer for debugging but has no direct impact on the simulation.

- **Reset**
  Every time the **step** method returns a positive *done* flag, the environment must be set to its initial state. This is achieved through the **reset** method.

- **Render**
  This method provides a graphical representation of the training process so that it can be observed, analysed, and more easily understood by humans.

6.1.1 Feature Design

For all environments, we chose to generate a nonlinear policy implemented with a *neural network*. The weights of the network represent the policy parameters and, differently from policies with handcrafted features (e.g. linear policies), we cannot know what these weights represent. The network architecture we adopted is a *Multi-layer Perceptron* with 2 hidden layers and 64 neurons as shown in figure (6.1).

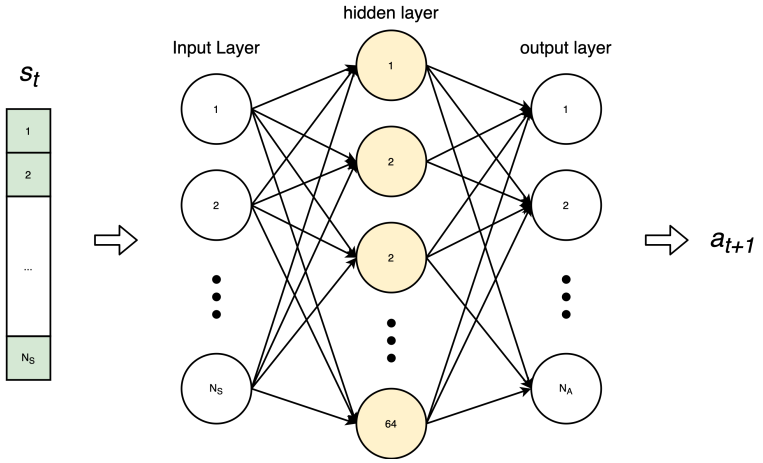


Figure 6.1: MLP agent

In the following sections of this chapter, we will explore, one by one, the environments we adopted to test our IRL approach. In every section, we will detail the logic and dynamics of the environment and then present the empirical results, followed by some considerations on the performance.

6.2 Portfolio Management

In this environment, an agent has access to a limited resource called *liquid asset*, which it must choose whether to invest or not in *non-liquid* assets to maximise the total liquidity at the end of an episode. The assets are affected by an interest rate which increases the value of the asset at every time step. The liquid asset has a fixed interest rate r_l wile the non liquid asset has a variable interest rate which can switch between two values r_{nl}^{low} and r_{nl}^{high} with probability p_{switch}. Every time the agent decides to invest in a non-liquid asset, it can buy one quantity for a fixed price α. After a fixed number of time-steps, the non-liquid asset reaches maturity and if it does not default (which can happen with probability p_{risk}), it is turned back into liquid. The agent can only buy a single non-liquid asset at each time-step and can only own a maximum of M non-liquid assets at the same time. The reward at every time-step t is given by the difference in the total amount of liquid assets between the states s_{t-1} and s_t. These dynamics can be represented with a vector as we can see in figure 6.2 with a simple example. In this scenario, the agent chooses to buy a single unit of non-liquid asset at time 0, the variable interest rate remains fixed at 1.1.

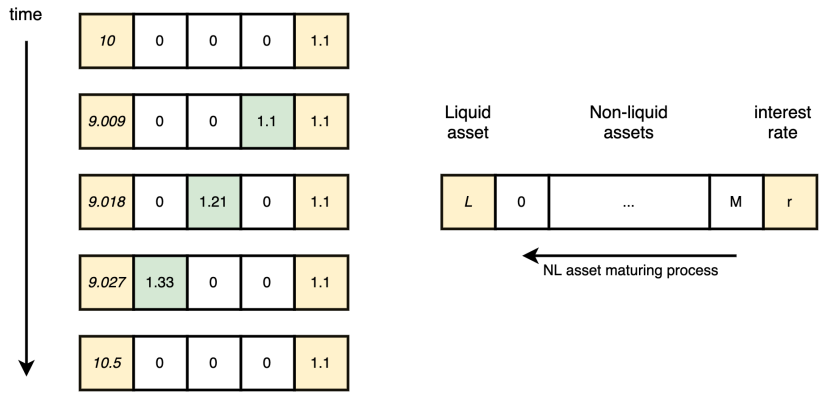


Figure 6.2: Portfolio Management state dynamics.

The values for the variables that were chosen for our implementation are:

$$\begin{array}{ccc} T = 50 & r_l = 1.001 & N = 4 \\ r_{nl}^{high} = 2 & r_{nl}^{low} = 1.1 & M = 10 \\ p_{risk} = 0.005 & p_{switch} = 0.1 & \alpha = \frac{0.2}{M} \end{array}$$

Table 6.1: Portfolio env. values

It is evident that never investing in non-liquid assets is the safest choice because the agent would be guaranteed a steady positive reward given by the liquid's fixed interest

rate r_l and could avoid completely the risk of ending up with less liquid than what it started with. Nevertheless, a "skilled" agent might be able to obtain a higher average return by investing the right amount at the right moments. As we will see, the parameter of risk aversion will play a fundamental role in balancing these two behaviors.

6.2.1 Numerical results

We trained our agents to optimize the *mean-variance* objective function. The training process was performed using the standard REINFORCE algorithm we introduced in chapter 3. Several agents were trained with different values for the risk-aversion parameter λ. The range was chosen empirically to create experts with distinguishable strategies. This can be observed by the difference in the average return and return variance of the five agents at convergence. The following graphs show the trend of the forward learning process and we can see how the behaviors progressively drift apart from one another.

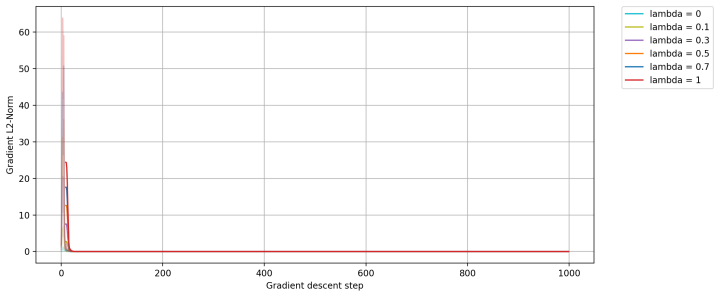


Figure 6.3: Portfolio forward problem: Gradient norm

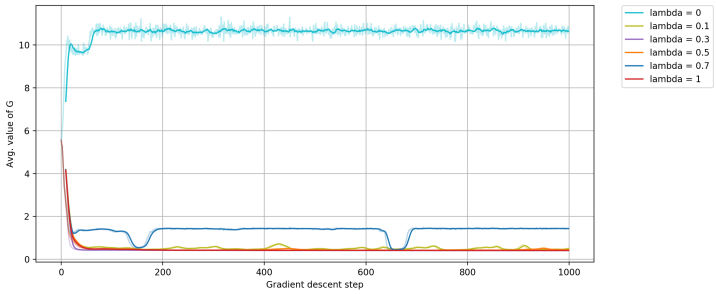


Figure 6.4: Portfolio forward problem: Average cumulative return

In figure 6.3, we can observe the gradient descent progresses gradually zeroing out the L2-norm of the gradient. This means that the agents are converging to an optimal strategy for the objective function they are optimizing, i.e. they are becoming *experts*. Since the inverse problem relies on the assumption that the expert has reached total

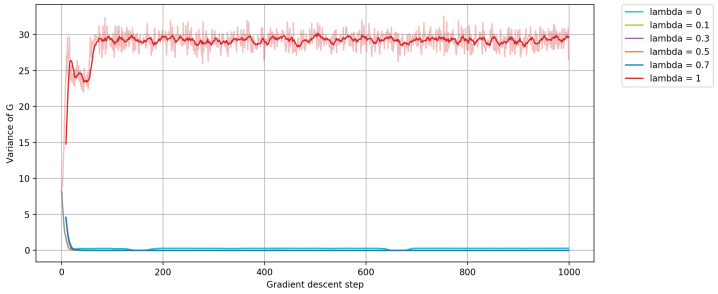


Figure 6.5: Portfolio forward problem: Variance of cumulative return

convergence, the closer the gradients are to zero, the better. In figures 6.4 and 6.5 we can see that the objective function is being optimized and the magnitude of the effect of risk aversion parameter over the performances. Agents optimizing a trade-off on the higher spectrum of λ values shows a significant difference in the return variance compared to the risk-neutral agent ($\lambda = 0$).

RA-GIRL

Once the experts have been generated, the second step of the problem consists of estimating the feature weights $\boldsymbol{\omega}$ of the environment's reward function, together with the parameter of risk aversion λ. This is achieved through the formulations seen in chapter 5. The reward function for the Portfolio environment is the following:

$$R_t = liquid_{s_t} - liquid_{s_{t-1}} \tag{6.1}$$

We can easily identify the two features ($liquid_{s_t}$ and $liquid_{s_{t-1}}$) and the relative weights (1 and -1). Since we want our weights to be o=confined to the unitary simplex so to avoid degenerate solutions, we can re-formulate the reward function as such:

$$R_{norm_t}(\boldsymbol{\omega}) = \omega_1(2 * liquid_{s_t}) - \omega_2(2 * liquid_{s_{t-1}}) \tag{6.2}$$

so that the correct weights become $\omega_1 = \omega_2 = 0.5$.

We used the optimization library from **SciPy**, which allows us to specify the Jacobian of the function we intend to minimize, to solve the minimisation problem. The results are summarised in table 6.1.

Portfolio Mean-Variance

Weights Configuration	Expected Values	Estimated Values	Gradient L2-norm
Weight 1	0.5	0.5345	1.54e-9
Weight 2	0.5	0.4655	
λ	0	0.0091	
	0.5	0.4899	1.52e-9
	0.5	0.5101	
	0.1	0.1020	
	0.5	0.4976	1.62e-5
	0.5	0.5023	
	0.3	0.8175	
	0.5	0.4978	3.37e-7
	0.5	0.5021	
	0.5	1.0483	
	0.5	0.5003	1.3e-5
	0.5	0.4996	
	0.7	0.6923	
	0.5	0.4998	7.96e-8
	0.5	0.5002	
	1	0.9865	

Table 6.2:

In the table summarizing the results of the estimations, we can see that the weights of the reward function have been accurately estimated for every agent. In the case of the level of risk aversion determined by the parameter λ, some estimations resulted very accurate while others were not. This can be due to the fact that the experts for which the estimation is not precise have converged to a local optimum of the optimization problem and are following a strategy that is not necessarily optimal for the level of risk

aversion they were trained with.

6.3 S&P 500 Trading

The second environment we decided to implement is a Trading system that runs on historical data, in particular the daily prices of the S&P index from the 1980s, until 2019. At every time-step the agent must choose among three possible actions: *long*, *short* or *flat*. Note that we assume that selling an action we do not own (short selling) is allowed. The amount invested when going *short* or *long* is fixed, meaning that the gains (and the losses) are not reinvested. Therefore, the total return over a trajectory is the accumulated sum of the step-wise rewards. At every step, the value of the asset at time t is p_t, and the reward is equal to

$$R_t = P_t \frac{price_t - price_{t-1}}{price_{t-1}} - f|P_t - P_{t-1}|, \tag{6.3}$$

where P_t is the portfolio value at time t and f is the proportionality constant, set to $7 \cdot 10^{-5}$. The state consists of the last 10 days of percentage price changes, with the addition of the previous portfolio position as well as the time remaining until the end of the episode, with episodes 50 days long. Intuitively, the most risk-averse behavior consists in always choosing to *flat*.

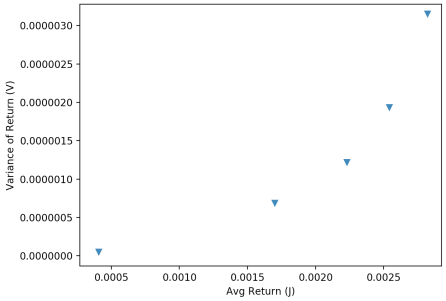
6.3.1 Numerical Results

Using the same network structure seen at the beginning of the chapter, we trained our agents to optimize two different risk-sensitive objective functions: *Mean Variance* and *Mean Volatility*. These measures share the fact that they employ a *risk parameter* that regulates the level of risk aversion induced. For each measure, we generated several experts with different levels of risk aversion and then applied the GIRL algorithm to estimate the reward weights. In figure 6.7, we can see the pareto frontier generated by the experts, which highlights the different trade-offs the experts converge to between average return and variance of the return.

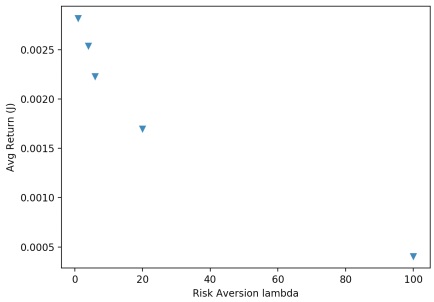
RA-GIRL

We then computed the policy gradients for each of the experts by generating a batch of 49000 steps (1k trajectories) for each expert and used the formulations from chapter 5 to estimate the reward weights $\boldsymbol{\omega}$ and level of risk aversion given by λ.

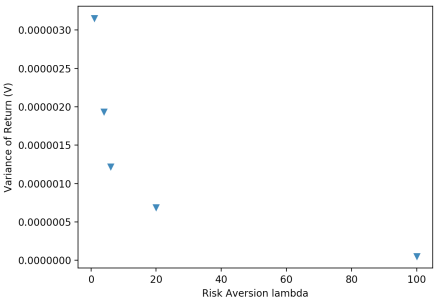
The reward function for the Trading environment is similar to the Portfolio management's one, except for a small fee given for every transaction. As seen before, we should re-formulate equation 6.3 so to be compatible with normalized feature weights:



(a) Mean-Variance frontier



(b) Average return frontier

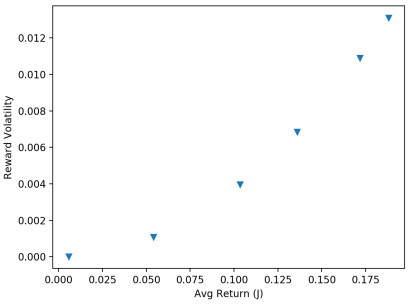


(c) Return variance frontier

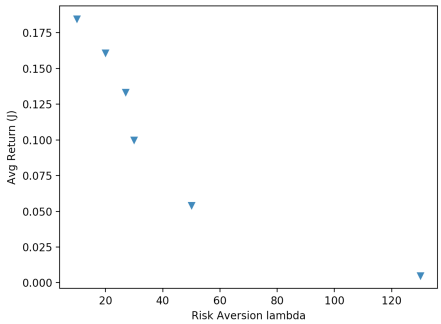
Figure 6.6: Trading: Mean-Variance Pareto frontiers

$$R_{norm_t}(\boldsymbol{\omega}) = \omega_1(n * P_t \frac{price}{price_{t-1}}) - \omega_2(n * P_t) - \omega_3|P_t - P_{t-1}|, \tag{6.4}$$

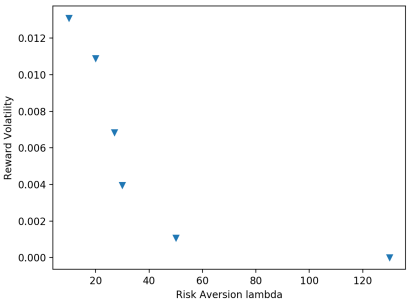
Where $n = 2 + 7e - 5$ is the normalization factor. The corresponding weights are $\omega_1 = 4.99982501e - 01, \omega_2 = 4.99982501e - 01, \omega_3 = 3.49987750e - 05$. In tables 6.3 and 6.4 we present the results of the predictions based on these features.



(a) Mean-Volatility frontier



(b) Average return frontier



(c) Reward Volatility frontier

Figure 6.7: Trading: Mean-Volatility Pareto frontiers

Trading Mean-Variance

Weights Configuration	Expected Values	Estimated Values	Gradients L2-norm
Weight 1	0.499982501	0.5004	0.02465
Weight 2	0.499982501	0.4996	
Weight 3	3.50e-05	4.23-18	
λ	1	0.0	
	0.499982501	0.5002	0.04384
	0.499982501	0.4997	
	3.50E-05	13.65e-5	
	4	7.5232	
	0.499982501	0.5002	0.03020
	0.499982501	0.4998	
	3.50E-05	8.47e-22	
	6	15.2795	
	0.499982501	0.5001	0.01984
	0.499982501	0.4999	
	3.50E-05	0.0	
	20	21.2222	
	0.499982501	0.5001	0.00135
	0.499982501	0.4997	
	3.50E-05	2.08e-4	
	100	2.5507	

Table 6.3:

Trading Mean-Volatility

Weights Configuration	Expected Values	Estimated Values	Gradients L2-norm
Weight 1	0.499982501	0.4994	2.4e-7
Weight 2	0.499982501	0.4973	
Weight 3	3.50e-05	3.26e-3	
λ	10	0.0837	
	0.499982501	0.5009	3.5e-10
	0.499982501	0.4991	
	3.50E-05	4.07e-23	
	20	5.8117	
	0.499982501	0.5002	1.25e-9
	0.499982501	0.4993	
	3.50E-05	4.72e-4	
	27	6.8618	
	0.499982501	0.5036	3.5e-17
	0.499982501	0.4963	
	3.50E-05	3.31e-5	
	30	17.9961	
	0.499982501	0.5017	3.0e-16
	0.499982501	0.4976	
	3.50E-05	7.18e-4	
	50	29.5076	
	0.499982501	0.5020	3.4e-7
	0.499982501	0.4873	
	3.50E-05	1.07e-2	
	130	189.9996	

Table 6.4:

For this environment we obtained similar results for both of the two risk measures

we tested. The reward weights are estimated quite well for all agents, although we can see that the solutions were less precise for the third feature weight. This is probably due to the fact that the third weight (representing the transaction fees of the environment) has a value that is on a smaller scale compared to the others. In the future, we will try to reformulate the reward features so that all the weights are on the same scale. Regarding the parameter of risk aversion, although the estimations are not precise, the result is quite satisfying when viewed comparatively. In fact, the estimations follow the same increasing trend as the real values, which means that the level of risk aversion of every expert compared to the other experts was estimated correctly.

6.4 Grid-World

The Grid World environment is a standard environment in RL literature as it offers an intuitive layout where behaviors are easy to interpret. The environment consists of a map with a finite number of locations in which the agent can find itself. Some of these locations can be more or less harmful to the agent, while others can be advantageous (for example a goal state). As we can see in figure (6.8), our environment has three types of cells. Safe cells (in light gray) give a reward of -0.1. Obstacle cells instead give a reward that is ten times as negative (-1.0). The goal state, instead, is a terminal state (the agent cannot exit the cell once entered) and it yields a reward of 0 for every time instant the agent is in it. Since our agent wants to maximise the total return, it will try to find the quickest path to the goal, trying to avoid the obstacles. Here, the stochasticity of the environment comes into play. Every action of the agent has a probability of failing, in which case the agent moves either to the right of to the left of the desired direction. This will push the agent towards paths that do not come close to the obstacles. Again in figure 6.8, we can see three different paths that reach the goal from the starting point. The paths are progressively more risk-averse, trading a higher step count to avoid stepping on the obstacles. As we will see later on, these three behaviors were observed for experts optimising different values of λ.

Figure 6.8: Grid-World map

6.4.1 Generating the Experts

For this particular environment we chose to generate a set of experts that optimize the *Entropic Risk Measure* (ERM) defined in equation 4.8. This choice followed from empirical evaluation of the performance of different risk measures (ERM, Mean-Variance, and Mean-Volatility) which showed the ERM to be the most suited in generating distinguishable behaviors for different levels of risk aversion. Figures 6.9, 6.10 and 6.11 show how the training process reached convergence on different levels of average return and variance of the return during a training process of one thousand iterations. Figure 6.12 gives us a better understanding of how the parameter of risk aversion affected the overall performance of the agent at convergence. As λ increases, the average return decreases as the agent progressively chooses longer paths that stay away from obstacles, this in turn also decreases the return variance. Overall there is a clear trade-off relation between the two performance metrics J and V.

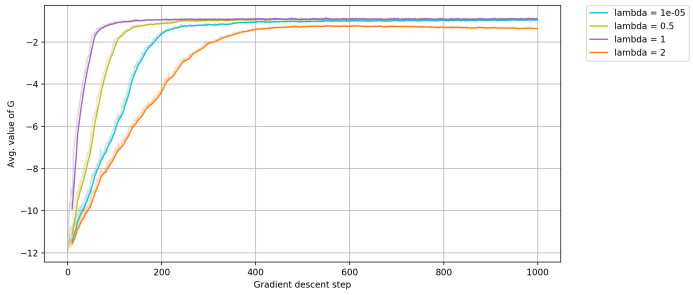


Figure 6.9: Grid World: Average return

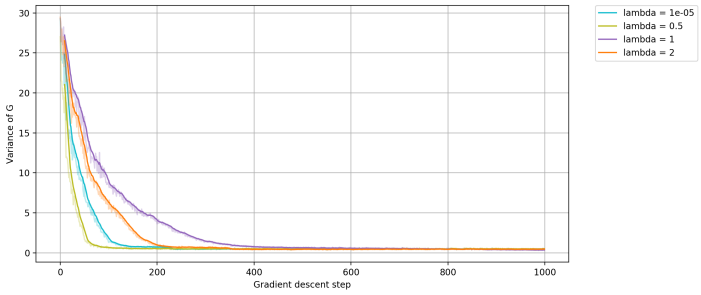


Figure 6.10: Grid World: Variance of return

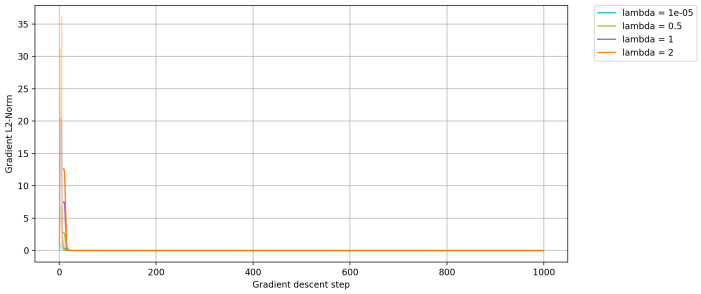
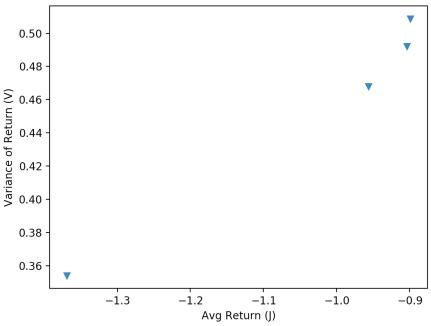
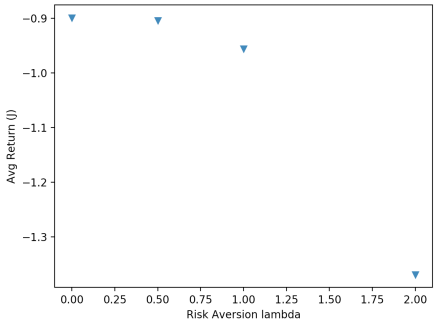


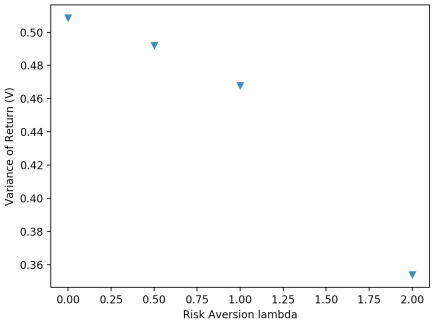
Figure 6.11: Grid World: Policy gradient convergence



(a) Mean-Variance frontier



(b) Average return frontier

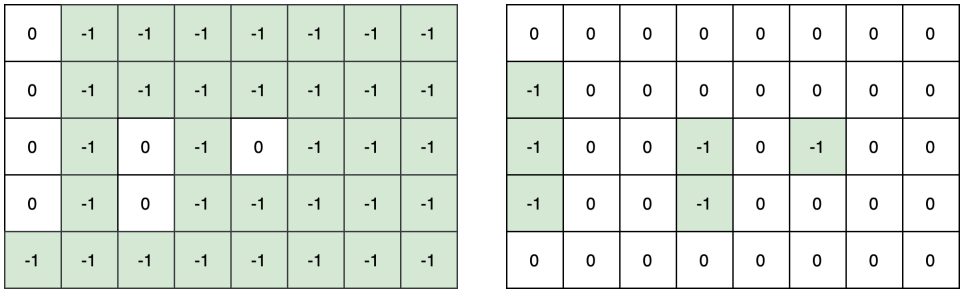


(c) Return variance frontier

Figure 6.12: Grid World: Pareto frontiers

6.4.2 RA-GIRL

As seen for the previous environments, the goal here is to retrieve the weights that characterize the reward function of the environment. In this case, the estimate for the level of risk aversion has not been computed due to experimental difficulties. For the map, we generated the reward is composed of two features corresponding to regular step tiles and obstacle tiles. The goal and starting state are not part of the equation. We can visualize how these two features are mapped out on the grid in figure 6.13. It is easy to see that the real weights for the step tile and obstacle tile features are respectively 0.1 and 1 (which, when normalised, become $0.\overline{09}$ and $0.\overline{90}$). The final step consisted of using the policy gradients obtained from a batch of 5000 steps performed by the experts and applying the equations seen in chapter 5 to estimate the feature weights. The results are summarized in table 6.5.



(a) Step tile feature (b) Obstacle tile feature

Figure 6.13: Grid World: Reward features

Grid World Entropic Risk Measure

Weights Configuration	Expected Values	Estimated Values	Gradient L2-norm
Weight 1	$0.\overline{09}$	0.3933	9.5457
Weight 2	$0.\overline{90}$	0.6066	
λ	0.3		
	$0.\overline{09}$	1.0256e-17	149.48
	$0.\overline{90}$	1.0	
	0.5		
	$0.\overline{09}$	0.1396	3.3876
	$0.\overline{90}$	0.8604	
	0.7		
	$0.\overline{09}$	0.1230	5.202
	$0.\overline{90}$	0.8760	
	1		
	$0.\overline{09}$	0.1633	165.9
	$0.\overline{90}$	0.8367	
	2		

Table 6.5:

For the Grid World Environment we had a hard time obtaining precise results due to the difficulties encountered in obtaining a good convergence for the forward problem. In Particular the gradients of the experts were not small enough for a decent estimation, even after a very long period of training. We, therefore, decided to reduce the dimensionality of the inverse problem by only trying to estimate the reward weights while providing the real value for the parameter of risk aversion λ. We can see that the estimation correctly identifies the higher cost of holes compared to the regular tiles of the grid. As we can see from the numbers obtained, the estimates improve for higher levels of risk aversion. This is probably due to the fact that the parameter λ appears as a denominator in equation 5.22 and this complicates the estimate when λ is close to zero.

Chapter 7

Conclusions and Future Works

In a reality where human-machine interaction is becoming ever more present, the role of Inverse Reinforcement Learning grows of importance. A novel approach to IRL proposed in [13] is the GIRL algorithm, which allows one to estimate the weights that characterize a linear reward of an agent in a *batch-based* and *model-free* manner. These characteristics of GIRL make it very interesting to explore and expand because it is often the case, for real world applications, that the model is not known and cannot be easily estimated. The GIRL algorithm has been shown to work for the case in which the agent is optimizing a reward function that can be expressed as a linear combination of basis functions with a standard risk-neutral objective function. The focus of this thesis has been to explore whether the approach can be extended to the space of nonlinear objective functions. The non-linear space embraces a much broader set of functions, including ones that contain a notion of risk. This extension is extremely relevant in the applicability of the approach to scenarios where the experts are humans. Humans have been shown to be very sensitive to risk and therefore being able to solve the IRL problem in a risk-sensitive framework is a fundamental step in this field of research. In particular, we focused our attention on three different risk measures: Mean-Variance, Entropic Risk Measure, and Mean-Volatility. Upon this premise we set ourselves two goals: derive the level of risk aversion of our experts, and estimate with sufficient precision the weights of the reward features, given that we are now dealing with a non-linear combination.

The analysis unfolded as follows: for each of the risk measures mentioned, we generated several experts presenting different levels of risk sensitivity (from risk neutrality to extreme risk aversion). We repeated this process for three different environments: Portfolio Management, Stock Trading, and Grid World, selecting the appropriate risk measure(s) for each of them. The different sensitivities yielded a set of distinguishable behaviors for every environment in which the agents were trained. Totaling over 20 experts for the various risk-measure - environment combinations, we proceeded with solving the

inverse problem of estimating the weights that characterize the environment's reward function, and the parameter of risk aversion in the objective function, by observing the expert acting within the environment.

From this intent, comes the first contribution of our work: starting from the trajectory-based gradient estimators seen in Chapter 5, which we obtained from an adaptation of the formulations found in [23] for the variance, and extended to the other risk measures, we derived the formalization for the inverse problem and devised a framework in which to solve it. For the case of the Mean-Variance objective function, we also provided a closed-form equation to compute the level of risk aversion of the expert when the reward function is known. The relevance of this contribution comes from the fact that we provide a unified approach to the problem of risk-sensitive Inverse Reinforcement Learning which, while still maintaining the properties of being *model-free* and *batch based*, can be applied on any class of risk measures, unlike [15] which is limited to Coherent Risk Measures. Our approach is also flexible, allowing one to easily reduce the estimation to the reward weights only, when the parameter of risk aversion is known, and vice-versa, to estimate accurately the parameter of risk aversion when the reward is known. In particular, the estimation of the risk aversion, given the reward weights, has been formalized as a quadratic problem which has a single optimal solution, yielding a very robust estimation.

This brings us to the second contribution of our work, which consists of the empirical evaluation of the proposed theoretical framework. We proceeded to implement and test the GIRL algorithm through the gradient-based minimization problem that allowed us to estimate, at the same time, both the expert's level of risk aversion and the parameter weights of the reward function. For all scenarios and risk measures we achieved a good degree of precision of the estimations, as can be seen in the results tables of Chapter 6.

The results obtained in this work pave the way for future research in several directions. Examples include testing out new risk measures, different environments, and further analysis of the correlation between the expert's gradient norm and the accuracy of the predictions. Regarding the reward function, in particular, our work could be extended to focus on studying how to choose the best reward features and how their shape influences the estimations. Finally, our approach could be generalized to non-linear reward functions so to encode the notion of risk within the reward function itself.

Bibliography

[1] Lorenzo Bisi, Edoardo Vittori, Luca Sabbioni, Matteo Papini, and Marcello Restelli. "Risk averse policy gradient for reward-volatility reduction". In: (2019).

[2] Hans Föllmer and Thomas Knispel. "Entropic risk measures: Coherence vs. convexity, model ambiguity and robust large deviations". In: *Stochastics and Dynamics* 11.02n03 (2011), pp. 333–351.

[3] Christian Gollier. *The Economics of Risk and Time*. Jan. 2001. DOI: 10.7551/mitpress/2622.001.0001.

[4] Garud N Iyengar. "Robust dynamic programming". In: *Mathematics of Operations Research* 30.2 (2005), pp. 257–280.

[5] Sham M Kakade. "A natural policy gradient". In: *Advances in neural information processing systems*. 2002, pp. 1531–1538.

[6] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. "Asynchronous methods for deep reinforcement learning". In: *International conference on machine learning*. 2016, pp. 1928–1937.

[7] David Nass, Boris Belousov, and Jan Peters. "Entropic Risk Measure in Policy Search". In: *arXiv preprint arXiv:1906.09090* (2019).

[8] PIERPAOLO GIORGIO NECCHI. "Policy gradient algorithms for the asset allocation problem". In: (2016).

[9] Andrew Y Ng, Stuart J Russell, et al. "Algorithms for inverse reinforcement learning." In: *Icml*. Vol. 1. 2000, pp. 663–670.

[10] Arnab Nilim and Laurent El Ghaoui. "Robust control of Markov decision processes with uncertain transition matrices". In: *Operations Research* 53.5 (2005), pp. 780–798.

[11] Takayuki Osogami. “Robustness and risk-sensitivity in Markov decision processes”. In: *Advances in Neural Information Processing Systems*. 2012, pp. 233–241.

[12] Jan Peters and Stefan Schaal. “Reinforcement learning of motor skills with policy gradients”. In: *Neural networks* 21.4 (2008), pp. 682–697.

[13] Matteo Pirotta and Marcello Restelli. “Inverse reinforcement learning through policy gradient minimization”. In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[14] Giorgia Ramponi, Amarildo Likmeta, Alberto Maria Metelli, Andrea Tirinzoni, and Marcello Restelli. “Truly Batch Model-Free Inverse Reinforcement Learning about Multiple Intentions”. In: *Proceedings of the Twenty-Third International Conference on Artificial Intelligence and Statistics*. PMLR, 2020.

[15] Lillian J. Ratliff and Eric V. Mazumdar. “Risk-Sensitive Inverse Reinforcement Learning via Gradient Methods”. In: *ArXiv* abs/1703.09842 (2017).

[16] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. “Trust region policy optimization”. In: *International Conference on Machine Learning*. 2015, pp. 1889–1897.

[17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[18] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. “Deterministic policy gradient algorithms”. In: 2014.

[19] Sumeet Singh, Jonathan Lacotte, Anirudha Majumdar, and Marco Pavone. “Risk-sensitive inverse reinforcement learning via semi-and non-parametric methods”. In: *The International Journal of Robotics Research* 37.13-14 (2018), pp. 1713–1740.

[20] Malcolm Strens. “A Bayesian framework for reinforcement learning”. In: *ICML*. Vol. 2000. 2000, pp. 943–950.

[21] Richard S Sutton, Andrew G Barto, et al. *Reinforcement Learning: an Introduction*. MIT press Cambridge, 2018.

[22] Aviv Tamar, Dotan Di Castro, and Shie Mannor. “Temporal Difference Methods for the Variance of the Reward To Go”. In: PMLR, 2013.

[23] Aviv Tamar, Yinlam Chow, Mohammad Ghavamzadeh, and Shie Mannor. "Policy gradient for coherent risk measures". In: *Advances in Neural Information Processing Systems*. 2015, pp. 1468–1476.

[24] Aviv Tamar, Dotan Di Castro, and Shie Mannor. "Temporal difference methods for the variance of the reward to go". In: *International Conference on Machine Learning*. 2013, pp. 495–503.

[25] Aviv Tamar and Shie Mannor. "Variance adjusted actor critic algorithms". In: *arXiv preprint arXiv:1310.3697* (2013).

[26] Martin J Wainwright, Michael I Jordan, et al. "Graphical models, exponential families, and variational inference". In: *Foundations and Trends® in Machine Learning* 1.1–2 (2008), pp. 1–305.

[27] Peter Whittle. "Risk sensitivity, a strangely pervasive concept". In: *Macroeconomic Dynamics* 6.1 (2002), pp. 5–18.

[28] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3-4 (1992), pp. 229–256.

[29] Huan Xu and Shie Mannor. "Robustness and generalization". In: *Machine learning* 86.3 (2012), pp. 391–423.

[30] Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. "Maximum entropy inverse reinforcement learning". In: (2008).