Learning Algorithms for Intelligent Agents and Mechanisms

Jad Rahme

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Program in
Applied and Computational Mathematics
Adviser: Ryan P. Adams

September 2022

Abstract

The ability to learn from past experiences and adapt one's behavior accordingly within an environment or context to achieve a certain goal is a characteristic of a truly intelligent entity. Developing efficient, robust, and reliable learning algorithms towards that end is an active area of research and a major step towards achieving artificial general intelligence. In this thesis, we research learning algorithms for optimal decision making in two different contexts, Reinforcement Learning in Part I and Auction Design in Part II.

Reinforcement learning (RL) is an area of machine learning that is concerned with how an agent should act in an environment in order to maximize its cumulative reward over time. In Chapter 2, inspired by statistical physics, we develop a novel approach to RL that not only learns optimal policies with enhanced desirable properties but also sheds new light on maximum entropy RL. In Chapter 3, we tackle the generalization problem in RL using a Bayesian perspective. We show that imperfect knowledge of the environment's dynamics effectively turn a fully-observed Markov Decision Process (MDP) into a Partially Observed MDP (POMDP) that we call the Epistemic POMDP. Informed by this observation, we develop a new policy learning algorithm LEEP which has improved generalization properties.

An auction is the process of organizing the buying and selling of products and services that is of great practical importance. Designing an incentive compatible, individually rational auction that maximizes revenue is a challenging and intractable problem. Recently, a deep learning based approach was proposed to learn optimal auctions from data. While successful, this approach suffers from a few limitations, including sample inefficiency, lack of generalization to new auctions, and training difficulties. In Chapter 4, we construct a symmetry preserving neural network architecture, EquivariantNet, suitable for anonymous auctions. EquivariantNet is not only more sample efficient but is also able to learn auction rules that generalize well to

other settings. In Chapter 5, we propose a novel formulation of the auction learning problem as a two player game. The resulting learning algorithm, ALGNet, is easier to train, more reliable and better suited for non stationary settings.

Acknowledgements

First of all, I would like to thank my adviser, Ryan Adams, for his guidance and support throughout my doctoral studies at Princeton. I'm grateful for his availability, flexibility, and generosity, especially when it comes to sharing research ideas and insights on a wide range of topics.

Throughout my PhD, I was fortunate to have the support of many Princeton faculty and administrative staff. I'm grateful to Matt Weinberg for his guidance and mentorship. His knowledge and expertise in mechanism design were critical when it came to bringing the second part of this thesis to fruition. I would also like to thank Peter Ramadge, Szymon Rusinkiewicz, Karthik Narasimhan for completing my thesis committee. I extend my gratitude to the supportive PACM department and Davis International Center.

I was also fortunate to take part in many opportunities outside of Princeton. I would like to express my gratitude to Sergey Levine for inviting me to visit his lab at UC Berkeley during the 2020-2021 academic year. His extensive knowledge and perspective on reinforcement learning helped me gain novel insights in the field. I'm also grateful for the summer internship opportunities I had at Quantlab, Susquehanna, and D.E. Shaw & Co. Despite being mostly virtual due to the pandemic, these experiences were very enriching and helped me acquire valuable knowledge, both theoretical and practical.

I would also like to thank my co-authors and collaborators including: Samy Jelassi, Aviral Kumar, Benjamin Eysenbach, Bianca Dumitrascu, Dibya Ghosh, Joan Bruna and Amy Zhang. I extend my gratitude to my colleagues in the Laboratory for Intelligent Probabilistic Systems (LIPS) at Princeton.

Last but not least, I would like to thank my parents, brother, sister, grandparents, family members, and Iryna for their never-ending support.

To my parents.

Contents

Chapter 1

Introduction

1.1 Overview

Reinforcement learning (RL) is an area of machine learning that is concerned with how an agent should act in an environment in order to maximize its cumulative reward over time. Whenever the agent takes an action, the state of the environment changes and the agent is rewarded accordingly. Based on observations of the dynamics, the agent gains a better understanding of its environment, which enables it to refine its strategy, also called *policy*, by choosing actions that are increasingly closer to optimality. The difficulty in reinforcement learning is that the dynamics of the environment and their associated rewards (the rules of the game) are unknown to the agent in advance; they can only be inferred through trial and error. This translates in practice to a tension between two types of behavior: exploration and exploitation. Exploration consists of trying new strategies with the hope of finding better ones or gaining a better knowledge of the rules of the game. Exploitation on the other hand is a more conservative attitude that consists of accumulating rewards via strategies that are already known to be good. Optimally balancing these two behaviors is at the core of RL and remains a major open question to this day.

Recent advancements on this challenging problem resulted in successes on tasks that were thought to be out of reach for our current technology. One of the most notable examples is AlphaGo Zero (Silver et al., 2017), a computer program that achieved super-human performance in the traditional board game of Go through self-play. The scope of reinforcement learning is, however, not limited to games; RL offers a very general framework to reason about a broad range of problems such as robotic manipulation and dexterity (Andrychowicz et al., 2020), data center cooling (Lazic et al., 2018), and optimizing chemical reactions (Zhou et al., 2017). Recently RL has been successfully applied on various physics problems including optimal jet grooming (Carrazza and Dreyer, 2019), quantum state preparation (Bukov et al., 2018; Bukov, 2018; Albarrán-Arriagada et al., 2018), quantum gate design (Niu et al., 2019) and quantum error correction (Fösel et al., 2018), often outperforming previous optimization methods.

Many approaches could be taken to tackle a reinforcement learning problem. Most of the successful ones however end up learning a quantity called a value function in one way or another. A value function is a function that takes a state of the environment and returns the expected cumulative rewards an agent can achieve starting from that state. A more technical definition of the value function can be found in Section 1.3. In a game of chess or Go, a value function could for example look at the state of the board and compute the probability that Black wins.

Rewards and value functions have a very similar flavor to energies - they are extensive quantities and the agent is trying to find a path that maximizes them. Many natural phenomena can be understood via an extremization principle. For example, in classical mechanics or electrodynamics, the *principle of least action* dictates that a mass or light will follow the path that minimizes a physical quantity called the *action*. Similarly, in thermodynamics, a system with many degrees of freedom—such as a gas—will explore its configuration space in search of a configuration that minimizes

its free energy. In RL, value functions are often treated as the central object of study. This stands in contrast to statistical physics formulations of such problems in which a quantity called the *partition function* is the primary abstraction, from which all the relevant thermodynamic quantities—average energy, entropy, heat capacity—can be derived. A natural question to ask is whether there exists a theoretical framework for reinforcement learning that is centered on a partition function, in which value functions can be interpreted via average energies?

This question is explored in Chapter 2. Inspired by the construction of partition functions in Statistical Physics, we construct a partition function for every state of the environment from the ensemble of possible trajectories spanning from that state. Although value functions can be derived from these partition functions and interpreted via average energies, we show that our purely partition function based approach can form the basis of alternative dynamic programming approaches.

Compared to classical reinforcement learning methods, our approach has three main benefits. First, in deterministic environments, partition functions obey linear Bellman equations allowing direct solutions that were unavailable for the nonlinear equations associated with the use of traditional value functions. Second, our approach is able to treat all rewards equally over time, which contrasts with traditional approaches that need to discount future rewards in order to get well defined Bellman equations. Third, our approach learns policies that are qualitatively different from the ones found by classical RL algorithms. These policies not only optimize for energy (the sum of future rewards) but also take *entropy* into account, favoring states from which many good outcomes are possible. To illustrate that point, let's consider a simple setting in which an agent is trying to go from point A to point B. At point A, two actions are possible: going up or going down. If the agent chooses up then there is only one path that leads to B, but if he chooses down then there are 99 valid paths that lead to B. Let's further assume that all these paths are equally good. Traditional RL approaches

will not have a preference between going up or going down as both of them lead to B. Our approach however will prefer going down and even prescribes choosing the down action 99 times more frequently than the up action. Such policies are desirable for their exploratory and robustness properties.

From this example, we can see that our statistical physics based approach to reinforcement learning naturally leads our agent to select action in a stochastic way - this is referred to as a stochastic policy. In contrast, a policy is deterministic if the agent always selects the same action given a certain state. If an environment is known and Markovian, commonly referred to as a Markov Decision Process (MDP), one could prove that there always exists a deterministic policy that acts optimally in that environment. As a result, one could think that learning a deterministic policy is optimal (Sutton and Barto, 2018). However, when the environment is not fully known, this is not the case and deterministic policies can then fail in a miserable way.

To illustrate that, consider a simple task in which an agent is first presented with an image of an animal and then has identify it with as few guesses as possible. In this example, a policy is just a mapping from images to guesses or labels. During the training phase, the agent is presented with images of dogs, cats and other animals from the training set and learns how to identify them. During the testing phase the agent is presented with new images of the same animals. In the hypothetical case where the training set is exhaustive, containing every picture of every animal from every angle, background and lighting condition, a deterministic policy learned on the training set will perform equally as good on the testing set. In real life however, data is limited and it is very unlikely that the agent will be able to learn a perfect classifier that correctly generalizes to new images with perfect accuracy. As a result, during the testing phase, the agent will at some point encounter an image that he is unable to classify correctly, not only in his first guess but in all the subsequent infinite number of available guesses as well because the policy is deterministic. This is problematic,

especially given that even a completely random guessing policy will eventually guess the correct label.

This simple toy experiment seems to indicate that there is a benefit in learning a stochastic policy to improve generalization. In fact, many successful RL algorithms encourage the agent to learn a stochastic policy by explicitly regularizing the entropy of the policy in the optimization objective. Adding uniform randomness everywhere is, however, sub optimal. The randomness in the agent's policy should reflect his uncertainty and confidence about his decision making. In Chapter 3 we study the generalization problem in Reinforcement Learning from a Bayesian perspective by modeling the uncertainty the agent has about the environment. We show how incomplete and imperfect knowledge of the environment implicitly turns a fully observed and Markovian environment (MDP), into a Partially Observed MDP (POMDP) (Sondik, 1971) which we call the Epistemic POMDP. This novel point of view allows us to derive a new RL algorithm, LEEP, with improved generalization properties.

Having robust, data efficient, and reliable learning algorithms is a necessary ingredient to solve real world problems with RL. An equally important and crucial ingredient is having a good reward function. After all, this is the quantity that RL algorithms are trying to optimize. Many real world problems do not come with natural reward functions - these are usually crafted by humans. Even in situations where there is a natural reward function to optimize, there is still some benefit in engineering a new reward function that makes the optimization problem easier to solve. For instance in the game of chess, a natural reward function is to reward an agent with a +1 for win, 0 for a draw and -1 for a loss. This is an example of a sparse reward function because the agent only gets rewarded at the end of the game without intermediate feedback. Sparse reward functions are hard to optimize and it is sometimes helpful to introduce intermediate rewards that are positive when the agent captures an opponents' piece

and negative when they lose one. While it's helpful to introduce intermediate rewards and more generally handcraft a reward function, reward shaping introduces human bias into the problem and in many cases, the policy that the agent learns will exploit the reward function in ways that the designer did not foresee. In some cases, the policy that maximizes the human engineered reward function does not solve the original task. One example of that is a game called Coast Runners where boats compete to finish a race as quickly as possible. To help them, intermediate targets were designed along the racetrack that not only help them get speed boosts but also reward them with extra bonus points when they are hit. It turned out that these intermediate targets disincentivized the agent from learning to win the race. The highest possible score is achieved by ignoring the race and focusing on hitting these intermediate targets (Clark and Amodei, 2016).

Unintended negative consequences of an incentive are not restricted to games or RL, they can be found in many real life societal policies as well. The Cobra effect (Siebert, 2001) is a historical anecdote used to illustrate perverse incentives that presumably occurred in India during the British rule. The British government, concerned about the increasing number of venomous cobras in Delhi, offered a bounty incentive for every dead cobra in hopes of reducing their numbers. This policy was very successful at reducing the number of cobras until the people realized that they could significantly increase their income by breeding cobras. The policy was subsequently canceled and the final situation was worse than the starting point.

Designing reward functions that are robust to these perverse incentives is not a very well studied subject within reinforcement learning. It is however a central theme in Mechanism Design, a sub field of Game Theory. Game Theory studies the emergent macroscopic behavior resulting from known microscopic interactions between agents. Mechanism Design goes the other way - it starts from a desirable macroscopic behavior and then tries to design mechanisms and incentives, or the rules of the game, that

would result in this global behavior, assuming individuals act rationally. That's why it's sometimes referred to as reverse game theory. Mechanism design has been applied to many fields from economics and politics to many problems such as market design, auction theory, social choice theory, voting systems, networked-systems, and many others. In this thesis, we will focus on Auction Theory, and while some of the ideas and methods we introduce are specific to that domain, others are more generally applicable to other domains in mechanism design.

An auction is a process of organizing the buying and selling of products and services that is of great practical importance in many private and public sectors. Examples include the sales of treasury bills by the US government, radio wave frequencies by the FCC, art by Christie's, or ads by Google. A simple auction model goes as follows: at the start of the auction, each one of the bidders place a bid on each one of the items. All of these bids are then collected by the Auctioneer who decides the item allocation as well as the amount each bidder has to pay for their participation in the auction. While any mapping from bids to allocations and from bids to payments could constitute a valid auction mechanism, in practice, we prefer auctions that verify some desirable properties.

The first desirable property is called *incentive compatibility*. An incentive compatible auction mechanism is one where the utility of each bidder is maximized by bidding truthfully on each one of the items. This means that the optimal bid on an item is exactly the amount of money that the bidder is willing to pay for the item. If an auction is not incentive compatible, bidders could strategically choose their bids and potentially bid untruthfully to maximize the value they get from participating in the auction. Enforcing incentive compatibility disincentivises any strategic behavior and as a result levels the playing field for all the bidders, irrespective of their experience, motivation, and means. Incentive compatible auctions are sometimes referred to as strategy-proof auctions.

The second desirable property is called *individual rationality*. An individually rational auction is one where a bidder is never worse off after participating in the auction as long as their bid is truthful. For instance, this means that a bidder will not be charged for participating in the auction if she didn't end up getting any of the items. More generally, the value of the items that a truthful bidder gets is always greater than or equal to the amount she has to pay to the auctioneer. Individual rationality encourages participation in the auction.

How to design an incentive compatible, individually rational auction mechanism that maximizes revenue for the auctioneer? Despite its apparent simplicity, this problem turns out to be surprisingly hard. In the case where there is a single item for sale, the solution is known from Myerson's seminal piece of work (Myerson, 1981). Beyond the single item setting, the problem is not completely resolved even for auctions as simple as two bidders and two items, despite forty years of mathematical research. Another line of work to confront this theoretical hurdle consists in building automated methods to find the optimal auction, typically by framing the problem as a linear program. However, this approach suffers from severe scalablility issues as the number of constraints and variables grows exponentially with the number of bidders and items (Guo and Conitzer, 2010).

A recent line of work initiated by Duetting et al. (2019) leverages the expressivity and scalibility of neural networks to go beyond the limitations of linear programs. Their idea is to parametrize the allocation and payment functions with deep neural networks and use gradient descent to learn an incentive compatible, individually rational auction mechanism that maximizes revenue. Their algorithm, RegretNet is capable of finding near-optimal results in several known settings and obtaining new mechanisms in unknown cases. While very successful, RegretNet suffers from two weakness. In practice, the algorithm is sample inefficient and hard to train.

RegretNet can require a large number of samples to learn an optimal auction. Furthermore, it is incapable of generalizing to new auctions with a different number of bidders and items. An optimal mechanism learned by RegretNet for an auction consisting of n bidders and m items can only be used on auctions with n bidders and m items because the neural network expects inputs of a specific dimension. If an additional bidder were to join or leave the auction or a new item was added or removed, then we would have to build and train a new auction mechanism from scratch. In Chapter 4, we tackle these two issues in the case of symmetric auctions. These are auctions which are invariant to the relabeling of the items or bidders. More specifically, such auctions are anonymous (in that they can be executed without any information about the bidders, or labeling them) and item-symmetric (in that it only matters what bids are made for an item, and not its a priori label). We prove that these auctions always admit optimal allocation and payment functions that are equivariant and then proceed to build equivariant neural network architectures that respect this symmetry. Our algorithm, EquivariantNet, is more sample efficient than RegretNet and can also generalize to different auctions.

The loss function in RegretNet is non stationary. It depends on several hyperparameters whose values change over time according to a predefined schedule. This makes RegretNet hard to train in practice. We also observe experimentally that the algorithm is very sensitive to the choice of these hyperparameters, converging to a suboptimal mechanism when these are not picked appropriately. Furthermore, these hyperparameters are setting dependent, their values depend on the number of bidders and items in the auction, and are currently found through an expensive hyperpameter search. In Chapter 5, we construct a novel, stationary, and hyperparameter-free loss function inspired by recent theoretical results from Auction Theory and propose a novel formulation of the auction learning problem as a two player game. The first player, the Auctioneer, proposes new auction rules. The second player, the

Misreporter, is trying to exploit these rules and find optimal ways to bid untruthfully. These two players interact and over time, the Misreporter becomes better at finding optimal bids and the Auctioneer becomes better at proposing auction mechanisms that increasingly get closer to being incentive compatible. We call this algorithm ALGNet and we show that it's as good or better than RegretNet, while being nearly hyper-parameter free.

1.2 Summary of Contributions

The contributions of this dissertation are summarized below:

- In Chapter 2, we propose a novel approach to the Reinforcement Learning problem. Inspired by Statistical Physics, we construct a partition function for each state of the environment and derive the corresponding Bellman equation. Our approach has three main benefits. First, it results in simpler equations, especially if the environment is deterministic. Second, it is able to treat all rewards equally over time (no need for a discount factor). Third, it learns policies that not only optimize for rewards but also take entropy into account, favoring states from which many good outcomes are possible.

  Chapter 2 is based on the following work:

  Jad Rahme and Ryan P. Adams. A theoretical connection between statistical physics and reinforcement learning. arXiv preprint arXiv:1906.10228*, 2019.*

- In Chapter 3, we study the generalization problem in Reinforcement Learning from a Bayesian perspective by modeling the uncertainty the agent has about the environment. We show how incomplete and imperfect knowledge of the environment implicitly turns a fully observed and Markovian environment (MDP) into a Partially Observed MDP (POMDP), which we call the Epistemic

POMDP. This novel point of view allows us to derive a new RL algorithm, LEEP, with improved generalization properties.

Chapter 3 is based on the following work (* indicates equal contribution):

Dibya Ghosh, Jad Rahme*, Aviral Kumar, Amy Zhang, Ryan P. Adams, and Sergey Levine. Why generalization in rl is difficult: Epistemic pomdps and implicit partial observability.* Advances in Neural Information Processing Systems*, 34, 2021.*

- In Chapter 4, we prove that symmetric auctions always admit optimal allocation and payment functions that are equivariant and then proceed to build an equivariant neural network architecture that respects this symmetry. We show that our algorithm, EquivariantNet, is not only more sample efficient than previous methods but can also generalize well to different auctions.

  Chapter 4 is based on the following work:

  Jad Rahme, Samy Jelassi, Joan Bruna, and S. Matthew Weinberg. A permutation-equivariant neural network architecture for auction design. Proceedings of the AAAI Conference on Artificial Intelligence*, 35(6):5664–5672, May 2021a. URL* `https://ojs.aaai.org/index.php/AAAI/article/view/16711`.

- In Chapter 5, we construct a novel, stationary, and hyperparameter-free loss function for the auction learning problem inspired by recent theoretical results from auction theory, and propose a novel formulation of the auction learning problem as a two player game (similar to GANs, (Goodfellow et al., 2014)). We show that the resulting algorithm ALGNet is as good or better than RegretNet, while being nearly hyper-parameter free.

  Chapter 5 is based on the following work:

Jad Rahme, Samy Jelassi, and S. Matthew Weinberg. Auction learning as a two-player game. In 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. *OpenReview.net, 2021c. URL* *https://openreview.net/forum?id=YHdeAO61l6T*.

The following two sections cover some background material on Reinforcement Learning and Auction Theory.

1.3 Background in Reinforcement Learning

In this section, we review the setup of the Reinforcement Learning problem as well as some of its basic concepts and approaches. A good reference on Reinforcement Learning can be found in Sutton and Barto (2018).

1.3.1 The RL Problem

RL as a Markov Decision Problem

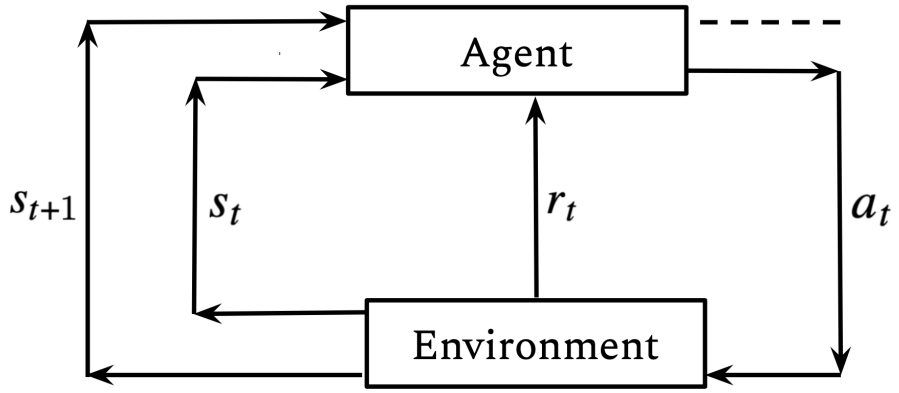


Figure 1.1: Representation of the agent–environment interaction in a Markov decision process.

Reinforcement Learning (RL) is an area of Machine Learning (ML) that studies how agents should behave in an environment in order to maximize their cumulative reward. The agent's sequential decision-making process is usually modeled as a Markov Decision Process (MDP).

At every time step t, the agent observes the state of the environment, s_t, and then decides what action to take, a_t. This action has two effects - first, it changes the state of the environment from s_t to s_{t+1} and second, it rewards the agent with a reward r_t. The state s_{t+1} and the reward r_t are random variables - the same causes don't necessarily results in the same effects. In an MDP, the environment's dynamics are allowed to be stochastic but they have to be Markovian. This means that the

distribution of states s_{t+1} that the agents lands in after taking action a_t from state s_t only depends on s_t and a_t, and not on past states $\{s_{t'}\}_{t'<t}$ or past actions $\{a_{t'}\}_{t'<t}$. This also holds for the reward r_t: its distribution is only a function of the initial state s_t, the action taken a_t, and the landing state s_{t+1}.

More formally, an MDP is defined by the objects $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P})$ where:

- $\mathcal{S}$ is the set of states the environment can be in,
- $\mathcal{A}$ is the set of actions an agent can take,
- $\mathcal{P}(s, a, s') = \mathbb{P}(s' \mid s, a)$ is the probability of landing in state s' after taking action a while still in state s,
- $\mathcal{R}(s, a, s')$ is the reward resulting from the transition $s \xrightarrow{a} s'$. $\mathcal{R}(s, a, s')$ is a random variable. We usually assume that all rewards are bounded from above by $\mathcal{R}_{\max}$.

Deterministic MDPs: An MDP is deterministic if $\mathcal{P}(s, a, s')$ is 0 for all states s' except one, which will be conveniently denoted by $s' = s + a$. In this case we have $\mathcal{P}(s, a, s') = \delta_{s+a}(s')$ and we will concisely denote $\mathcal{R}(s, a, s + a)$ by $\mathcal{R}(s, a)$. $\mathcal{R}(s, a)$ is also assumed to be deterministic.

Policies

In RL, the policy π describes how the agent acts in the environment. $\pi(a \mid s)$ denotes the probability that the agent picks action a while in state s. If we denote the reward resulting from the t-th transition by r_t, so that $r_t \coloneqq \mathcal{R}(s_t, a_t, s_{t+1})$, we can express the cumulative reward $\mathcal{R}_{\text{total}}(\pi)$ of such a policy as:

$$\mathcal{R}_{\text{total}}(\pi) = \mathbb{E}_{\substack{a_t \sim \pi(.|s_t) \\ s_{t+1} \sim \mathbb{P}(.|s_t, a_t)}} \left[\sum_{t=0}^{+\infty} \gamma^t r_t \right], \tag{1.1}$$

where the expectations are taken with respect to the realized sequences of states and actions, according to the policy and environment dynamics. Here $\gamma \in [0, 1)$ is called the *discount factor* that can be interpreted as a preference for immediate rewards over future ones. The discount factor γ is also necessary for mathematical reasons: without this discount factor, many quantities in RL are not well defined. For example, the infinite series determining $\mathcal{R}_{\text{total}}(\pi)$ could diverge.

The RL Objective

The goal of RL is to find an optimal policy π^* that maximizes $\mathcal{R}_{\text{total}}(\pi)$:

$$\pi^* = \arg\max_{\pi} \ \mathcal{R}_{\text{total}}(\pi) \, .$$

There are two main approaches to solving a RL problem: value function type approaches and policy gradient type approaches. In the following sections we will give a quick exposition of both of these approaches. These sections are not meant to be exhaustive or extensive by any measure, their goal is to give the reader some background that could help them contrast traditional approaches to RL with the novel approaches and methods that we propose in Chapters 2 and 3.

1.3.2 Value Function Approaches

Value functions associated with a policy

The value function V associated with a policy π is a function of the state s that measures the expected cumulative reward an agent will get by following the policy π starting from state s:

$$V^{\pi}(s) = \mathbb{E}_{\substack{a_t \sim \pi(.|s_t) \\ s_{t+1} \sim \mathbb{P}(.|s_t, a_t)}} \left[\sum_{t=0}^{+\infty} \gamma^t r_t \mid s_0 = s \right] .$$

The value functions at different states are connected by a recursion called the *Bellman equation*:

$$V^{\pi}(s) = \mathbb{E}_{\substack{a\sim\pi(.|s) \\ s'\sim\mathbb{P}(.|s,a)}} \left[\mathcal{R}(s,a,s') + \gamma\, V^{\pi}(s')\right] .$$

In the Bellman equation above, the expectation is taken with respect to a single action and a single state transition. Similar to V^{π}, we can define another type of value function Q^{π} (the *"Q-function"*) which is a function of a state-action pair (s,a), now measuring expected cumulative reward from following the policy π after taking action a from state s:

$$Q^{\pi}(s,a) = \mathbb{E}_{\substack{s_{t+1}\sim\mathbb{P}(.|s_t,a_t) \\ a_{t+1}\sim\pi(.|s_{t+1})}} \left[\sum_{t=0}^{+\infty} \gamma^t r_t \mid s_0 = s, a_0 = a\right] .$$

Q^{π} also follows a Bellman equation given by:

$$Q^{\pi}(s,a) = \mathbb{E}_{\substack{s'\sim\mathbb{P}(.|s,a) \\ a'\sim\pi(.|s')}} \left[\mathcal{R}(s,a,s') + \gamma\, Q^{\pi}(s',a')\right] .$$

Optimal value functions

When the policy π is optimal, $\pi = \pi^*$, the Bellman equations for V and Q are referred to as the *optimal Bellman equations* and are given by:

$$\begin{aligned} V^*(s) &= \max_{a\in\mathcal{A}} \mathbb{E}_{s'\sim\mathbb{P}(.|s,a)} \left[\mathcal{R}(s,a,s') + \gamma V^*(s')\right] , \\ Q^*(s,a) &= \mathbb{E}_{s'\sim\mathbb{P}(.|s,a)} \left[\mathcal{R}(s,a,s') + \gamma \max_{a'\in\mathcal{A}} Q^*(s',a')\right] . \end{aligned} \tag{1.2}$$

These optimal Bellman equations are fixed point equations. The mapping underlying these fixed point equations is called the *Bellman operator*. One can show that when $0 \leqslant \gamma < 1$, these Bellman operators are contractions of norm γ. As a result, when the Bellman operator is known, one can converge to V^* and Q^* by successive

iterations of their Bellman operators starting from any initialization (via the Banach fixed-point theorem). The optimal policy π^* can then be recovered from the optimal value function, as:

$$\pi^*(a \mid s) = \begin{cases} 1 & \text{if } a = \arg\max_{a' \in \mathcal{A}} \ Q^*(s, a') \,, \\ 0 & \text{otherwise} \,. \end{cases}$$

Finding the optimal value function through iteration of the Bellman operator is not always possible. In most settings, the dynamics of the MDP, $\mathcal{P}$, and the reward function, $\mathcal{R}$, are not fully known and as a result, it's not possible to solve the RL problem through an exact fixed point iteration scheme. Furthermore, even when the dynamics of the environment are known, it is not always possible to proceed through fixed point iteration and this is especially true for MDPs with large state and actions spaces.

Exploration

When the dynamics of the environment ($\mathcal{R}$ and $\mathcal{P}$) are unknown, the expectations in the Bellman equations cannot be computed exactly; they can only be estimated using samples collected through interactions with the environment.

The policy $\pi_{\text{exploration}}$ used by the agent to collect these samples and learn about the environment is called the *exploration policy*. Depending on the problem, some exploration policies can be better than others. Two of the most popular exploration policies are:

- ε-*greedy*: Pick $\arg\max_{a' \in \mathcal{A}} \ Q(s, a')$ with probability $1 - \varepsilon$, and any action uniformly at random with probability ε.
- *Boltzmann exploration of parameter* β: At state s, pick action a proportionally to $\exp\left[\beta \ Q(s, a)\right]$.

Both of these exploration policies use current estimates of the Q-function. In the beginning, when the agent only had a few interactions with the environment, the estimate for the optimal Q-function is very uncertain, and typical values for ε and β are chosen to be 1 and 0 respectively. This corresponds to taking actions uniformly at random. As the agent interacts more with the environment and collects more data, ε is decreased to 0 and β is increased to a large number. This reflects our confidence that the Q-function is becoming more accurate over time.

Some RL algorithms (e.g. Q-learning) only use the latest transition seen by the exploration policy to update the Q values and as a result don't need to store past past transitions. Other algorithms (e.g. DQN) don't limit themselves to the latest transition but also use previously seen transitions to update their current Q values. When that is the case, all the interactions with the environment are recorded and stored in a dataset called a *Replay Buffer*. A typical entry in the replay buffer takes the form a tuple $(s_t, a_t, s_{t+1}, r_t, f_t)$, where f_t is a Boolean entry that indicates whether the episode ended or is still ongoing.

Q-learning

The Q-learning algorithm is a popular value function based, RL learning algorithm first introduced in Watkins (1989). Q-learning does not assume that the dynamics of the environment are known - it's a *model-free* algorithm.

The algorithm goes as follows: Initially, all Q values are initialized to arbitrary values (or randomly). Then, at each time step t, the agent looks at the state s_t of the environment, takes action a_t, and observes the next state, s_{t+1}, and the reward associated with the transition, r_t. The algorithm then updates the Q value of the state-action pair (s_t, a_t) according the following learning rule:

$$\underbrace{Q(s_t, a_t)}_{\text{updated value}} \longleftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha_t}_{\text{learning rate}} \left(\underbrace{r_t + \underbrace{\gamma}_{\text{discount factor}} \times \underbrace{\max_{a \in \mathcal{A}} Q(s_{t+1}, a)}_{\text{optimal Q value at } s_{t+1}}}_{\text{target}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right). \tag{1.3}$$

The parameter $\alpha_t \in (0, 1)$ in this equation is called the *learning rate* and can be time-dependent. Intuitively, this update rule is trying to close the gap between the left hand side and the right hand side of the optimal Bellman equations for the Q-function (Equation 1.2). The Q-learning algorithm is summarized in Algorithm 1.1:

Algorithm 1.1 Q learning

1: Initialize $Q(s, a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$ arbitrarily.
2: If s is a terminal state, set $Q(s, a) = 0$ for all actions a.
3: **for** each episode **do**
4: **while** episode has not ended **do**
5: Observe s_t and choose action a_t according to an exploration policy.
6: Take action a_t and observe s_{t+1} and r_t.
7: Update $Q(s_t, a_t)$ according to the update rule 1.3.
8: Return Q.

The Q-learning algorithm provably learns the optimal Q function under some reasonable assumptions:

- The learning rate α_t has to decrease to 0 but not too fast ($\sum \alpha_t$ should diverge and $\sum \alpha_t^2$ should converge).
- Each state action pair (s, a) must be visited an infinite number of times by the exploration policy.

A more technical statement of these assumptions and a proof of the convergence result can be found in Watkins and Dayan (1992).

Q-learning works well for MDPs with small state and action spaces but becomes intractable in larger ones. In the following section, we will see how to scale Q-learning to larger MDPs with the help of function approximations and deep learning.

Deep Q Networks (DQN)

In the previous section, the learning algorithm had to learn a table of $|\mathcal{S}| \times |\mathcal{A}|$ Q values, one for each state-action pair (s, a). This is not possible for MDPs with a large state and action space, such as MDPs with continuous state spaces. This is where function approximations become useful.

The idea is to parametrize the Q function with a family of functions $\{Q_\theta\}_{\theta \in \mathbb{R}^d}$, typically a neural network, and then learn the optimal value of the parameter θ such that we have $Q_\theta(s, a) \approx Q^*(s, a)$. Note that this problem is tractable because the dimensionality of the learning problem, d, is independent of the size of the MDP. The optimal value of the parameter θ is learned by minimizing the Bellman error with (a variant of) gradient descent:

$$\mathcal{L}(\theta) = \mathbb{E}_{s_t, a_t, s_{t+1}}\Big[\ \big(Q_\theta(s_t, a_t) - \underbrace{\big[r_t + \gamma \times \max_{a' \in \mathcal{A}} Q_\theta(s_{t+1}, a')\big]}_{\text{target}}\ \big)^2\ \Big]\,. \tag{1.4}$$

The expectation in $\mathcal{L}(\theta)$ is empirically estimated by sampling a batch of random transitions from the replay buffer. Optimizing $\mathcal{L}(\theta)$ is not as straightforward as it seems and many tricks are required to stabilize the learning algorithm. For instance, when computing the gradient of $\mathcal{L}(\theta)$, the target is treated as a constant and does not contribute to the overall gradient. In fact, the target is computed using a "delayed" version of the parameter θ.

The details of the training procedure and the empirical tricks needed to stabilize the learning algorithm can be found in the original DQN paper (Mnih et al., 2013). Many additional improvements to the DQN algorithm were discovered since its initial publication and some of the major ones are reported in Hessel et al. (2018).

1.3.3 Policy Gradient Approaches

Instead of learning an optimal value function from which an optimal policy can be inferred, policy gradient approaches tackle the RL problem by learning the optimal policy directly. In the following we will paramatrize the policy space by a family of functions $\{\pi_\theta\}_{\theta \in \mathbb{R}^d}$.

REINFORCE

The RL objective (Equation 1.1) can be re-written more explicitly as an expectation over trajectories $\tau = \{(s_t, a_t s_{t+1})\}_{0 \leqslant t \leqslant T}$, as:

$$J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta} \left[R(\tau) \right] ,$$

where $R(\tau) = \sum_t \gamma^t r_t$ is the total reward encountered by the trajectory τ, and:

$$\rho_\theta(\tau) := \underbrace{p_0 \left(s_0 \right)}_{\text{distribution of the initial state}} \prod_{t=0}^{T} \pi_\theta \left(a_t \mid s_t \right) \mathbb{P} \left[s_{t+1} \mid s_t, a_t \right]$$

is the probability of observing trajectory τ by following the policy π_θ. Using the *log trick*, $\nabla_\theta \rho_\theta(\tau) = \rho_\theta(\tau) \nabla_\theta \log \rho_\theta(\tau)$, we can write $\nabla_\theta J(\theta)$ as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta} \left[\nabla_\theta \log \rho_\theta(\tau) R(\tau) \right] .$$

Since $\mathbb{P}\left[s_{t+1} \mid s_t, a_t\right]$ does not depend on θ, we have:

$$\nabla_\theta \log \rho_\theta(\tau) = \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta\left(s_t, a_t\right) ,$$

and finally we find a very simple estimate of $\nabla_\theta J(\theta)$:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta\left(s_t, a_t\right) R(\tau)\right] . \tag{1.5}$$

This result was first derived by Williams (1992). Optimizing $J(\theta)$ by following an empirical estimate of the gradient above (Equation 1.5) is known at the REINFORCE algorithm.

Algorithm 1.2 Vanilla REINFORCE

1: Set: α: learning rate, N: number of iterations, B: sample size.
2: Initialize π_θ.
3: **for** i from 1 to N **do**
4: Set $\nabla_\theta J(\theta) = 0$.
5: **for** b from 1 to B **do**
6: Sample one trajectory τ and compute its total rewards $R(\tau)$.
7: Gradient accumulation: $\nabla_\theta J(\theta) \longleftarrow \nabla_\theta J(\theta) + \frac{1}{B} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta\left(s_t, a_t\right) R(\tau)$.
8: Update θ: $\theta \longleftarrow \theta + \alpha \nabla_\theta J(\theta)$.
9: Return π_θ.

Beyond REINFORCE

The gradient estimates in REINFORCE are usually very noisy as they suffer from high variance which can destabilise the learning process. The algorithm can be made more reliable through the usage of variance reduction schemes. These usually involve introducing value function estimates inside the REINFORCE formula. For instance,

further analysis of Equation 1.5 shows that it can be re-written as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\rho_\theta, a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) Q^{\pi_\theta}(s,a)\right],$$

where Q^{π_θ} is the Q function of policy π_θ and ρ_θ is the state marginal distribution resulting from following the policy π_θ. This expression could be further re-written as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\rho_\theta, a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) A^{\pi_\theta}(s,a)\right], \tag{1.6}$$

where $A^{\pi_\theta}(s,a) := Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$ is called the advantage function. The advantage function measures how good an action is at a given state. A positive (negative) advantage indicates that the action is better (worse) than average. Using Equation 1.6 to estimate gradients leads to less noisy estimates and results in a more stable and improved policy learning algorithm. Further details on how to estimate A^{π_θ} as well as a technical analysis of variance reduction schemes in policy gradient methods can be found in Schulman et al. (2015b) and Greensmith et al. (2004).

REINFORCE minimizes the RL objective by taking a step in the direction of the gradient $\nabla_\theta J(\theta)$. While this can provably improve the policy in the limit of small step sizes, it is not necessarily the best direction to follow. Indeed, the gradient points towards the direction of steepest ascent when distances are measured using the Euclidean distance in the parameter space, $d(\theta_1, \theta_2) = ||\theta_1 - \theta_2||_2$. Without any additional assumptions on the mapping $\theta \rightarrow \pi_\theta$, measuring distances in the parameter space might not be appropriate. Small differences in the parameter θ could lead to large differences in the policy space and as a result, in the agent's performance. Conversely, large changes in θ could correspond to infinitesimal policy changes and imperceptible changes of the agent's behavior.

Consequently, it seems more appropriate to measure distances at the policy level instead of the underlying parameter's level. This results in a different gradient called

the *natural gradient* and a different policy gradient algorithm called the *Natural Policy Gradient* (NPG) (Kakade, 2001). Many policy gradient methods were then developed following up and improving on that line of work, including *Trust Region Policy Optimization* (TRPO) by Schulman et al. (2015a), *Proximal Policy Optimization* (PPO) by Schulman et al. (2017), and *Actor-Critic using Kronecker-factored Trust Region* (ACKTR) by Wu et al. (2017).

1.4 Background in Auction Theory

In this section, we give a brief high level introduction to auction theory that gives more context for Chapters 4 and 5. A good reference on auction theory and mechanism design more generally can be found in Nisan et al. (2007) or Roughgarden (2016).

1.4.1 A Simple Model For Auctions

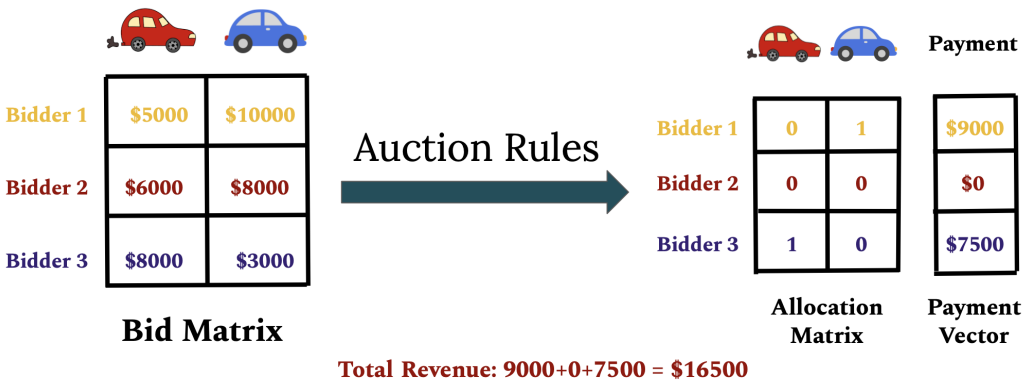


Figure 1.2: Illustration of the auction mechanics.

Setting

An auction consists of n bidders and m items. Let $N := \{1, \cdots, n\}$ and $M := \{1, \cdots, m\}$ denote the set of bidders and items respectively. At the start of the auction, each one of the bidders bids a sum of money on each one of the items. We will denote the bid of bidder i on item j by b_{ij}. These bids can be grouped into a matrix $B = \{b_{ij}\}_{i \in N, j \in M}$ called the bid matrix.

The allocation and payment functions

An auction mechanism is characterized by two functions, the allocation function $g : \mathbb{R}^{n \times m} \to [0, 1]^{n \times m}$ and the payment function $p : \mathbb{R}^{n \times m} \to \mathbb{R}^n$, both of which have the bid matrix B as an input.

The allocation function computes the allocation matrix $g(B)$ where $g(B)_{ij}$ is the probability that bidder i gets item j. Since it's possible for an item to not be allocated to any of the bidders, we have $\forall j \in M,\ \sum_i g(B)_{ij} \leqslant 1$. It is sometimes convenient to denote the allocation function restricted to bidder i, by by g_i, i.e. $g_i(B) = [g(B)_{ij}]_{j \in M}$.

The payment function computes the payment vector $p(B)$ where $p(B)_i$ is the amount of money bidder i has to pay to the Auctioneer. The revenue of the Auctioneer P is the sum of the payments made by all the bidders, $P = \sum_{i=1}^{n} p(B)_i$.

Additive auctions

A subset of items $S \subseteq M$ does not have the same value for each one of the bidders. Each bidder i has his own valuation function, $v_i : 2^M \to \mathbb{R}$, where $v_i(S)$ denotes how much bidder i values the subset of items S. In principle v_i can be arbitrary, assigning arbitrary values to each one of the possible subsets.

Depending on the context, it makes sense to consider simpler valuation functions that have more structure to them. For example, we could consider valuation functions in which the value of a basket of items is equal to the highest individual item value in that basket: $v_i(S) = \max_{j \in S} v_i(\{j\})$. This is called a *unit-demand valuation function*. Other examples are value functions in which the value of a basket of items is equal to the sum of the values of its components: $v_i(S) = \sum_{j \in S} v_i(\{j\})$. Such valuations are called *additive*. Additive valuations are among the most studied valuation functions in auction theory and will be the main focus of this thesis.

An additive valuation function is fully specified by the individual values for each one of the items. In the following, we denote the value that bidder i gives for item j by v_{ij}. These values can be grouped into a matrix $V = \{v_{ij}\}_{i \in N, j \in M}$ called the *valuation matrix*. Note that in general, the bid matrix can be different than the value matrix. In the following we will denote the i-th row of the value matrix V by $\vec{v}_i$.

The utility of a bidder

The utility of a bidder is the net amount of value obtained as a result of his participation in the auction. We can compute it as the difference between the total value of the items that the bidder got and the amount he had to pay to the auctioneer. The utility of bidder i, u_i, is given by :

$$u_i(v_i, B) = \underbrace{v_i(g(B))}_{\text{Value received by bidder i}} - \underbrace{p(B)_i}_{\text{Payment of bidder i}} .$$

Notice that the total value received by bidder i in this expression is computed using the valuation function of bidder i. For additive auction, this expression can be re-written as:

$$u_i(\vec{v}_i, B) = \sum_{j=1}^{m} g(B)_{ij}[\vec{v}_i]_j - p(B)_i = \sum_{j=1}^{m} g(B)_{ij} V_{ij} - p(B)_i \,. \tag{1.7}$$

1.4.2 Problem Statement

While there exist infinite choices of allocation functions g and payment functions p that constitute a valid auction mechanism, in practice, we care about auctions that satisfy certain desirable properties. In the following we will focus on two desirable properties, *incentive compatibility* and *individual rationality*.

Notation: Given a matrix $B \in \mathbb{R}^{n \times m}$ and $i \in \{1, \cdots, n\}$ we will denote the i-th row of the matrix B by $\vec{b}_i$ and the $(n-1) \times m$ matrix that one gets from B by removing i-th row by B_{-i}. Given a vector $\vec{b}_i{}' \in \mathbb{R}^m$, we will denote the matrix that we get by replacing row $\vec{(}b)_i$ with $\vec{b}_i{}'$ in B by $(\vec{b}'_i, B_{-i})$. The rows of $(\vec{b}'_i, B_{-i})$ are $[\vec{b}_1, \cdots, \vec{b}_{i-1}, \vec{b}_i{}', \vec{b}_{i+1}, \cdots, \vec{b}_n]$.

If B is a bid matrix, then $\vec{b}_i$ is the vector of bids of bidder i, B_{-i} is the bid matrix of all the bidders except bidder i, and $(\vec{b}_i{}', B_{-i})$ is a bid matrix that we get if bidder i modifies his bid from $\vec{b}_i$ to $\vec{b}_i{}'$.

Incentive Compatibility

Strategic bidders seek to maximize their utility and may report bids that are different from their true valuations ($\vec{b}_i \neq \vec{v}_i$). In hindsight, once all the bids are known, the optimal bid of bidder i, $\vec{b}_i^*$, is given by:

$$\vec{b}_i^* = \underset{\vec{b}_i{}' \in \mathbb{R}^m}{\arg\max} \; u_i\left(\vec{v}_i, \left(\vec{b}_i', B_{-i}\right)\right) .$$

In general, we should expect that $\vec{b}_i^*$ is different from $\vec{v}_i$. In some auctions however, the utility of a bidder is always maximized when his bid is *truthful* regardless of the other bids and we have $\vec{b}_i^* = \vec{v}_i$. These auctions are called dominant strategy incentive compatible auctions (DSIC). The following provides a formal definition.

Definition 1. *An auction (g, p) is dominant strategy incentive compatible (DSIC) if each bidder's utility is maximized by reporting truthfully no matter what the other bidders report. For every bidder i, valuation $\vec{v}_i$, bid $\vec{b}_i{}'$ and bids B_{-i}, we have: $u_i(\vec{v}_i, (\vec{v}_i, B_{-i})) \geqslant u_i(\vec{v}_i, (\vec{b}_i{}', B_{-i}))$.*

DSIC is a desirable property because it levels the playing field to all the bidders. It not only makes it easier for bidders to bid optimally (by bidding truthfully), but also makes it easier for the auctioneer to predict the outcome of an auction since the optimal bids are easily characterized. DSIC auctions are sometimes called *strategy-proof auctions*.

Individual Rationality

The payment function p in an auction can, in principle, charge a positive amount of money to a bidder who has not been allocated any of the items. It can also charge a bidder more than the total value of the items he got from the auction. In both of these cases, the utility of the bidder is negative, which means that the bidder is worse off after participating in the auction. An individually rational auction guarantees that such cases cannot happen to a bidder as long as he's bidding truthfully. A truthful bidder always has a non negative utility function regardless of what the other bidders decide to do. The following provides a formal definition.

Definition 2. *An auction is individually rational (IR) if for all i, $\vec{v}_i$ and B_{-i} we have:*

$$u_i(\vec{v}_i, (\vec{v}_i, B_{-i})) \geqslant 0. \tag{1.8}$$

Individually rational auctions are desirable because they encourage bidder participation in the auction.

The auction learning problem

Each bidder knows how much each item is worth to them. This information is not known to the other bidders nor to the auctioneer. This setting is referred to as a *private value auction.* However, a common assumption is that each bidder draws their value vector, $\vec{v}_i$, from some prior probability distribution, D_i, which is common knowledge.

The goal of the auctioneer is to design an incentive compatible, individually rational auction that maximizes his expected revenue given a prior on the bidder's value vectors $(\vec{v}_1, \cdots, \vec{v}_n) \sim (D_1, \cdots, D_n)$.

Formally, we can rewrite the problem as:

$$\min_{(g,p)\in\mathcal{M}} -\mathbb{E}_{V\sim D}\left[\sum_{i=1}^{n} p_i(V)\right] \quad \text{s.t.} \quad (g,p) \text{ is a DSIC auction}\,,$$
$$(g,p) \text{ is a IR auction}\,.$$

1.4.3 Optimal Single Item Auction

This section is intended to give the curious reader a peek into one of the most celebrated results in auction theory. While reading this section is not strictly necessary to understand the work presented in this thesis, it could be interesting to contrast the analytical approach to finding the optimal auction presented in this section with the machine learning based approaches of Chapters 4 and 5.

Finding the optimal single item auction was fully resolved by Myerson in his seminal 1981 paper (Myerson, 1981). In this section, we include a high level derivation of Myerson's result that makes some simplifying assumptions. A complete and technical derivation can be found in Myerson's orginal paper (Myerson, 1981).

The allocation function is monotonic.

Our goal in this section is to prove that the optimal allocation function is monotonic non-decreasing with respect to the bids of each one of the bidders:

$$\text{The function: } b_i \to [g((b_i, B_{-i}))]_i \text{ is monotonic non-decreasing}\,.$$

We remind the reader that since we're in a single item setting, $\vec{b}_i$ is a real number that we conveniently represent by b_i . Intuitively this result makes sense. A bidder can expect to increase the probability of getting the item by increasing his bid, assuming that all the other bids remain constant. We will see that this result follows naturally

from the DSIC property.

Notation: In the following, we will fix a bidder i and the bids of all the other bidders B_{-i}. This allows us to adopt the following convenient notations: $g(b) := [g(b, B_{-i})]_i$, $p(b) := [p(b, B_{-i})]_i$ and $u(v, b) := u_i(v, (b, B-i)) = g(b) \times v - p(b)$. With these simplifying notations, our goal is to prove that the function $b \to g(b)$ is monotonic non decreasing.

For a DSIC auction, we have $u(v, v) \geqslant u(b, v)$ for all v and b. Through simple manipulations we can re-write this inequality as:

$$\begin{aligned} u(v,v) \geqslant u(b,v) &\iff g(v) \times v - p(v) \geqslant g(b) \times v - p(b)\,, \\ &\iff (g(v) - g(b)) \times v \geqslant p(v) - p(b)\,. \end{aligned}$$

By permuting the roles of b and v, we also get that $(g(v) - g(b)) \times b \leqslant p(v) - p(b)$. We conclude that:

$$(g(v) - g(b)) \times v \geqslant p(v) - p(b) \geqslant (g(v) - g(b)) \times b\,. \tag{1.9}$$

This inequality implies that $(g(v) - g(b)) \times (v - b) \geqslant 0$ which proves that the function g is monotonic non decreasing as claimed.

A relation between the allocation and payment functions

In this section, we derive a relation between the allocation function g and payment function p for a truthful mechanism. For the sake of simplicity, we will make the assumption that the function g is differentiable. While this assumption simplifies the proof, it is not a necessary one. Only the monotonicity of g is required in the more general proof (Myerson, 1981).

Taking $b < v$, we can rewrite equation 1.9 as:

$$\frac{(g(v) - g(b))}{v - b} \times v \geqslant \frac{(p(v) - p(b))}{v - b} \geqslant \frac{(g(v) - g(b))}{v - b} \times b \,. \tag{1.10}$$

In the limit of $b \longrightarrow v$, we find that:

$$\frac{d}{dz} p(z) = z \times \frac{d}{dz} g(z) \,. \tag{1.11}$$

For an individually rational auction, we have $p(0) = 0$ and we get:

$$p(b) = \int_0^b dz \;\; z \times \frac{d}{dz} g(z) \,. \tag{1.12}$$

Going back to our original (non simplified) notation, we can rewrite this equation as:

$$[p(b_i, B_{-i})]_i = \int_0^{b_i} db \;\; b \times \frac{d}{db} g(b, B_{-i}) \,. \tag{1.13}$$

Equation 1.13 shows that given an allocation function g there is at most one candidate payment function p such that the mechanism defined by (g, p) is incentive compatible and individually rational. Conversely, one can prove that given a monotonic non decreasing allocation function g, the mechanism defined by (g, p) where p is given by equation 1.13 is DSIC and IR.

Deriving the optimal allocation function

Now that we have the characterization of the space of incentive compatible individually rational auctions, we move on to finding the revenue maximizing auction.

Let's denote by $(D_1, \cdots, D_n)$ the probability distributions from which the bidders' value is sampled: $(v_1, \cdots, v_n) \sim (D_1, \cdots, D_n)$. Since we're in a single-item setting, the distribution D_i is a probability distribution over the real numbers. To simplify the derivation, we will assume that all values are bounded by $v_{\max}$ and that D_i has a

continuous probability density function which we'll denote as f_i. We will use F_i to denote the corresponding cumulative distribution function.

Conditioned on B_{-i}, the expected payment of bidder i, p_i, is given by:

$$\begin{aligned} p_i &= \mathbb{E}_{v_i \sim D_i}\left[p_i(v_i, B_{-i})\right] , \\ &= \int_0^{v_{\max}} p_i(v_i, B_{-i}) f_i\left(v_i\right) dv_i , \\ &= \int_0^{v_{\max}} \left[\int_0^{v_i} z \times g_i'\left(z, B_{-i}\right) dz\right] f_i\left(v_i\right) dv_i , \end{aligned} \tag{1.14}$$

where in the last equality we used the characterization of the payment function in terms of the allocation function as found in equation 1.13. This expression can be further simplified by first permuting the order of integration and then by integrating by parts:

$$\begin{aligned} \int_0^{v_{\max}} \left[\int_0^{v_i} z \times g_i'\left(z, B_{-i}\right) dz\right] f_i\left(v_i\right) dv_i &= \int_0^{v_{\max}} \left[\int_z^{v_{\max}} f_i\left(v_i\right) dv_i\right] z \times g_i'\left(z, B_{-i}\right) dz , \\ &= \int_0^{v_{\max}} \left(1 - F_i(z)\right) \times z \times g_i'\left(z, B_{-i}\right) dz , \\ &= -\int_0^{v_{\max}} g_i\left(z, B_{-i}\right) \times \left(1 - F_i(z) - z f_i(z)\right) dz , \\ &= \int_0^{v_{\max}} \underbrace{\left(z - \frac{1 - F_i(z)}{f_i(z)}\right)}_{:=\varphi_i(z)} \times g_i\left(z, B_{-i}\right) \times f_i(z) dz . \end{aligned}$$

The quantity $\varphi_i(z)$ that appears under the integral is called the *virtual valuation* of bidder i. Note that this quantity only depends on bidder i, it does not depend on any of the other bidders, and it can be negative.

The total expected revenue for the auctioneer is then given by:

$$P = \mathbb{E}_{V \sim D}\left[\sum_{i=1}^{n} p_i(V)\right] = \mathbb{E}_{V \sim D}\left[\sum_{i=1}^{n} \varphi_i(v_i) \times g(V)_i\right] . \tag{1.15}$$

From equation 1.15 we can see that the expected revenue is a linear combination of the virtual values.

If all these virtual values are negative, then the optimal allocation is $g(V)_i = 0$ for all bidders. Otherwise, optimality is reached by allocating the item to the bidder with the highest virtual value. The optimal payment function can then be inferred using equation 1.13.

An example

To illustrate how that works in practice, let's consider the case where $D_1 = \cdots = D_n = \text{Uniform}([0,1])$. The virtual valuation function is then given by $\varphi(b) = 2b - 1$.

If all the bids are smaller than $r = \frac{1}{2}$, then all the virtual bids are negative and the item is not allocated, so none of the bidders have to pay any amount of money to the auctioneer. r is called the *reserve price*, which is the value under which the auctioneer is not willing to sell his item.

If this is not the case, then there is at least one bidder that bid more than r. Without loss of generality we can assume that bidder 1 has the highest bid and bidder 2 has the second highest bid. In this case, bidder 1 gets the item. The allocation function of bidder 1 is given by:

$$g_1(b_1) = \begin{cases} 0 & \text{if } b_1 < \max(r, v_2)\,, \\ 1 & \text{if } b_1 \geqslant \max(r, v_2)\,. \end{cases}$$

Its derivative is given by $g_1'(b_1) = \delta_{\min(r,v_2)}(b_1)$ where δ is the Dirac function. By plugging this expression into equation 1.13, we find that bidder 1 has to pay $p_1 = \min(r, v_2)$ to the auctioneer. The optimal auction can be summed up by the

following two cases:

$$\begin{cases} \text{If all the bids are below } r : \text{no one gets the item and no one pays.} \\ \text{Else: the highest bidder gets the item and pays } \max(r, \text{ second highest bid}). \end{cases}$$

This is called a *second price auction with a reserve price*.

Generalizing these results to larger auctions is not straightforward. In fact, despite decades of research, there is no known analytical derivation that would enable us to systematically derive optimal mechanism for a general auction. In Chapter 4 and 5 we will see a machine learning based approach to approximate optimal auctions.

Part I

Reinforcement Learning

Chapter 2

A Theoretical Connection Between Statistical Physics and Reinforcement Learning

2.1 Abstract

Sequential decision making in the presence of uncertainty and stochastic dynamics gives rise to distributions over state/action trajectories in reinforcement learning (RL) and optimal control problems. This observation has led to a variety of connections between RL and inference in probabilistic graphical models (PGMs). Here we explore a different dimension to this relationship, examining reinforcement learning using the tools and abstractions of statistical physics. The central object in the statistical physics abstraction is the idea of a partition function $\mathcal{Z}$, and here we construct a partition function from the ensemble of possible trajectories that an agent might take in a Markov decision process. Although value functions and Q-functions can be derived from this partition function and interpreted via average energies, the $\mathcal{Z}$-function provides an object with its own Bellman equation that can form the basis of alternative dynamic

programming approaches. Moreover, when the MDP dynamics are deterministic, the Bellman equation for $\mathcal{Z}$ is linear, allowing direct solutions that are unavailable for the nonlinear equations associated with traditional value functions. The policies learned via these $\mathcal{Z}$-based Bellman updates are tightly linked to Boltzmann-like policy parameterizations. In addition to sampling actions proportionally to the exponential of the expected cumulative reward as Boltzmann policies would, these policies take *entropy* into account favoring states from which many outcomes are possible.

2.2 Introduction

One of the central challenges in the pursuit of machine intelligence is robust sequential decision making. In a stochastic and uncertain environment, an agent must capture information about the distribution over ways they may act and move through the state space. Indeed, the algorithmic process of planning and learning itself can lead to a well-defined distribution over state/action trajectories. This observation has led to a variety of connections between reinforcement learning (RL) and inference in probabilistic graphical models (PGMs) (Levine, 2018). In some ways this connection is unsurprising: belief propagation (and its relatives such as the sum-product algorithm) is understood to be an example of dynamic programming (Koller and Friedman, 2009) and dynamic programming was developed to solve control problems (Bellman, 1966; Bertsekas, 1995). Nevertheless, the exploration of the connection between control and inference has yielded fruitful insights into sequential decision making algorithms (Kalman, 1960; Attias, 2003; Ziebart, 2010; Kappen, 2011; Levine, 2018).

In this chapter, we present another point of view on reinforcement learning as a distribution over trajectories, one in which we draw upon useful abstractions from statistical physics. This view is in some ways a natural continuation of the agenda of connecting control to inference, as many insights in probabilistic graphical models

have deep connections to, e.g., spin glass systems (Hopfield, 1982; Yedidia et al., 2001; Zdeborová and Krzakala, 2016). More generally, physics has often been a source of inspiration for ideas in machine learning (MacKay, 2003; Mezard and Montanari, 2009). Boltzmann machines (Ackley et al., 1985), Hamiltonian Monte Carlo (Duane et al., 1987; Neal et al., 2011; Betancourt, 2017) and, more recently, tensor networks (Stoudenmire and Schwab, 2016) are a few examples. In addition to direct inspiration, physics provides a compelling framework to reason about certain problems. The terms *momentum*, *energy*, *entropy*, and *phase transition* are ubiquitous in machine learning. However, abstractions from physics have generally not been so far helpful for understanding reinforcement learning models and algorithms. That is not to say there is a lack of interaction; RL is being used in some experimental physics domains, but physics has not yet as directly informed RL as it has, e.g., graphical models (Carleo et al., 2019).

Nevertheless, we should expect deep connections between reinforcement learning and physics: an RL agent is trying to find a policy that maximizes expected reward and many natural phenomena can be viewed through a minimization principle. For example, in classical mechanics or electrodynamics, a mass or light will follow a path that minimizes a physical quantity called the *action*, a property known as the *principle of least action*. Similarly, in thermodynamics, a system with many degrees a freedom—such as a gas—will explore its configuration space in the search for a configuration that minimizes its free energy. In reinforcement learning, rewards and value functions have a very similar flavor to energies, as they are extensive quantities and the agent is trying to find a path that maximizes them. In RL, however, value functions are often treated as the central object of study. This stands in contrast to statistical physics formulations of such problems in which the *partition function* is the primary abstraction, from which all the relevant thermodynamic quantities—average energy, entropy, heat capacity—can be derived. It is natural to ask, then, *is there*

a theoretical framework for reinforcement learning that is centered on a partition function, in which value functions can be interpreted via average energies?

In this chapter, we show how to construct a partition function for a reinforcement learning problem. In a deterministic environment (Section 2.3), the construction is elementary and very natural. We explicitly identify the link between the underlying average energies associated with these partition functions and value functions of Boltzmann-like stochastic policies. As in the inference-based view on RL, moving from deterministic to stochastic environments introduces complications. In Section 2.4.2, we propose a construction for stochastic environments that results in realistic policies. Finally, in Section 2.5, we show how the partition function approach leads to an alternative model-free reinforcement learning algorithm that does not explicitly represent value functions.

We model the agent's sequential decision-making task as a Markov decision process (MDP), as is typical. The agent selects actions in order to maximize its cumulative expected reward until a final state is reached. The MDP is defined by the objects $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P})$. $\mathcal{S}$ and $\mathcal{A}$ are the sets of states and actions, respectively. $\mathcal{P}(s, a, s') = \mathbb{P}(s' \mid s, a)$ is the probability of landing in state s' after taking action a from state s. $\mathcal{R}(s, a, s')$ is the reward resulting from this transition. We also make the following additional assumptions:

1. $\mathcal{S}$ is finite,

2. all rewards $\mathcal{R}(s, a, s')$ are bounded from above by $\mathcal{R}_{\max}$ and are deterministic,

3. the number of available actions is uniformly bounded over all states by d.

We also allow for terminal states to have rewards even though there are no further actions and transitions. We denote these final-state rewards by $\mathcal{R}(s_f)$. By shifting all rewards by $\mathcal{R}_{\max}$ we can assume without loss of generality that $\mathcal{R}_{\max} = 0$ making

all transition rewards $\mathcal{R}(s, a, s')$ non positive. The final state rewards $\mathcal{R}(s_f)$ are still allowed to be positive however.

2.3 Partition Functions for Deterministic MDPs

Our starting point is to consider deterministic Markov decision processes. Deterministic MDPs are those in which the transition probability distributions assign all their mass to one state. Deterministic MDPs are a widely studied special case (Madani, 2002; Wen and Van Roy, 2013; Dekel and Hazan, 2013) and they are realistic for many practical control problems, such as robotic manipulation and locomotion, drone maneuver or machine-controlled scientific experimentation. For the deterministic setting, we will use $s + a$ to denote the state that follows the taking of action a in state s. Similarly, we will denote the reward more concisely as $\mathcal{R}(s, a)$.

2.3.1 Construction of State-Dependent Partition Functions

To construct a partition function, two ingredients are needed: a statistical ensemble, and an energy function E on that ensemble. We will construct our ensembles from trajectories through the MDP; a trajectory ω is a sequence of tuples $\omega = (s_0, a_0, r_0), (s_1, a_1, r_1), \ldots, (s_T, a_T, r_T)$ such that state s_{T+1} is a terminal state. We use the notation $s_t(\omega)$, $a_t(\omega)$, and $r_t(\omega)$ to indicate the state, action, and reward, respectively, of trajectory ω at step t. Each state-dependent ensemble $\mathbf{\Omega}(s)$ is then the set of all trajectories that start at s, i.e., for which $s_0(\omega) = s$. We will use these ensembles to construct a partition function for each state $s \in \mathcal{S}$. Taking $|\omega|$ to be the length of the trajectory, we write the energy function as

$$E(\omega) = -\sum_{t=0}^{|\omega|-1} r_t(\omega) - R(s_{|\omega|}) = -\sum_{t=0}^{|\omega|} r_t(\omega)\,. \tag{2.1}$$

The form on the right takes a notational shortcut of defining $r_{|\omega|}(\omega) := R(s_{T+1})$ for the reward of the terminal state. Since the agent is trying to maximize their cumulative reward, $E(\omega)$ is a reasonable measure of the agent's preference for a trajectory in the sense that lower energy solutions accumulate higher rewards. Note in particular that the ground state configurations are the most rewarding trajectories for the agent. With the ingredients $\Omega(s)$ and $E(\omega)$ defined, we get the following partition function

$$\mathcal{Z}(s,\beta) = \sum_{\omega\in\Omega(s)} e^{-\beta E(\omega)} \tag{2.2}$$

$$= \sum_{\omega\in\Omega(s)} e^{\beta \sum_{t=0}^{|\omega|} r_t(\omega)} . \tag{2.3}$$

In this expression, $\beta \geqslant 0$ is a hyper-parameter that can be interpreted as the inverse of a temperature. (This interpretation comes from statistical physics where $\beta = \frac{1}{K_B T}$, where K_B is the Boltzmann constant.) This partition function does not distinguish between two trajectories having identical cumulative rewards but different lengths. However, among equivalently rewarding trajectories, it seems natural to prefer shorter trajectories. One way to encode this preference is to add an explicit penalty $\mu \leqslant 0$ on the length $|\omega|$ of a trajectory, leading to a partition function

$$\mathcal{Z}(s,\beta) = \sum_{\omega\in\Omega(s)} e^{-\beta E(\omega)+\mu|\omega|} . \tag{2.4}$$

In statistical physics, μ is called a *chemical potential* and it measures the tendency of a system (such as a gas) to accept new particles. It is sometimes inconvenient to reason about systems with a fixed number of particles, adding a chemical potential offers a way to relax that constraint, allowing a system to have a varying number of particles while keeping the average fixed.

Note that since MDPs can allow for both infinitely long trajectories and infinite sets of finite trajectories, $\Omega(s)$ can be infinite even in relatively simple settings. In

Appendix 2.A.1, we find that a sufficient condition for $\mathcal{Z}(s,\beta)$ to be well defined is taking $\mu < -\log d$. As written, the partition function in Eq. 2.4 is ambiguous for final states. For clarity we define $\mathcal{Z}(s_f,\beta) := e^{\beta R(s_f)}$ for a terminal state s_f. We will refer to these as the boundary conditions.

Mathematically, the parameter μ has a similar role as the one played by γ, the discount rate commonly used in reinforcement learning problems. They both make infinite series convergent in an infinite horizon setting, and ensure that the Bellman operators are contractions in their respective frameworks (Appendices 2.A.3 and 2.B.3). However, when using γ, the order in which the rewards are observed can have an impact on the learned policy which does not happen when μ is used. This could be a desirable property for some problems as it uncouples rewards from preferences for shorter paths.

2.3.2 A Bellman Equation for $\mathcal{Z}$

As we have defined an ensemble $\Omega(s)$ for each state $s \in \mathcal{S}$, there is a partition function $\mathcal{Z}(s,\beta)$ defined for each state. These partition functions are all related through a Bellman-like recursion:

$$\mathcal{Z}(s,\beta) = \sum_a e^{\beta \mathcal{R}(s,a)+\mu}\, \mathcal{Z}(s+a,\beta)\,, \tag{2.5}$$

where, as before, $s+a$ indicates the state deterministically following from taking action a in state s. This Bellman equation can be easily derived by decomposing each trajectory $\omega \in \Omega(s)$ into two parts: the first transition resulting from taking initial action a and the remainder of the trajectory ω' which is a member of $\Omega(s+a)$. The total energy and length can also be decomposed in the same way, so that:

$$
\begin{aligned}
\mathcal{Z}(s,\beta) &= \sum_{\omega\in\Omega(s)} e^{-\beta E(\omega)+\mu|\omega|} \\
&= \sum_{\omega\in\Omega(s)} e^{\beta\sum_{t=0}^{|\omega|} r_t(\omega)+\mu|\omega|} \\
&= \sum_{a\in\mathcal{A}} e^{\beta\mathcal{R}(s,a)+\mu} \sum_{\omega'\in\Omega(s+a)} e^{\beta\sum_{t=1}^{|\omega|} r_t(\omega)+\mu(|\omega|-1)} \\
&= \sum_{a\in\mathcal{A}} e^{\beta\mathcal{R}(s,a)+\mu} \sum_{\omega'\in\Omega(s+a)} e^{-\beta E(\omega')+\mu|\omega'|} \\
&= \sum_{a} e^{\beta\mathcal{R}(s,a)+\mu}\ \mathcal{Z}(s+a,\beta)\,.
\end{aligned}
$$

Note in particular that this Bellman recursion is **linear** in $\mathcal{Z}$.

2.3.3 The Underlying Value Function and Policy

The partition function can be used to compute an average energy to shed light on the behavior of the system. This average is computed under the Boltzmann (Gibbs) distribution induced by the energy on the ensemble of trajectories :

$$
\mathbb{P}(\omega\,|\,\beta,\mu,s_0(\omega)=s) = \frac{\mathbf{1}_{\Omega(s)}(\omega)}{\mathcal{Z}(s,\beta)} e^{-\beta E(\omega)+\mu|\omega|}\,. \tag{2.6}
$$

In probabilistic machine learning, this is usually how one sees the partition function: as the normalizer for an energy-based learning model or an undirected graphical model (see, e.g., Murray and Ghahramani (2004)). Under this probability distribution, high-reward trajectories are the most likely but sub-optimal ones could still be sampled. This approach is closely related to the *soft-optimality* approach to RL (Levine, 2018). This distribution over trajectories allows us to compute an average energy for state s either as an explicit expectation or as the partial derivative of the log partition function

with respect to the inverse temperature:

$$\begin{aligned}\langle E\rangle &= \sum_{\omega\in\Omega(s)} \frac{1}{\mathcal{Z}(s,\beta)} e^{-\beta E(\omega)+\mu|\omega|} E(\omega) \\ &= -\frac{\partial}{\partial\beta}\log\mathcal{Z}(s,\beta)\,. \end{aligned} \tag{2.7}$$

The negative of the average energy is the value function:

$$V(s,\beta) := -\langle E\rangle = \frac{\partial}{\partial\beta}\log\mathcal{Z}(s,\beta).$$

This is an intuitive result: recall that the energy $E(\omega)$ is low when the trajectory ω accumulates greater rewards, so lower average energy indicates that the expected cumulative reward—the value—is greater. Since the partition functions $\{\mathcal{Z}(s,\beta)\}_{s\in S}$ are connected by a Bellman equation, we expect that the underlying value functions $\{V(s,\beta)\}_{s\in S}$ would be connected in a similar way, and there is indeed a non-linear Bellman recursion:

$$\begin{aligned} V(s,\beta) &= \frac{\partial}{\partial\beta}\log\mathcal{Z}(s,\beta) \\ &= \frac{1}{\mathcal{Z}(s,\beta)}\frac{\partial}{\partial\beta}\mathcal{Z}(s,\beta) \\ &= \frac{1}{\mathcal{Z}(s,\beta)}\frac{\partial}{\partial\beta}\sum_{a\in\mathcal{A}} e^{\beta\mathcal{R}(s,a)+\mu}\ \mathcal{Z}(s+a,\beta) \\ &= \frac{1}{\mathcal{Z}(s,\beta)}\sum_{a\in\mathcal{A}} e^{\beta\mathcal{R}(s,a)+\mu}\frac{\partial}{\partial\beta}\mathcal{Z}(s+a,\beta) + \mathcal{R}(s,a)e^{\beta\mathcal{R}(s,a)+\mu}\ \mathcal{Z}(s+a,\beta)\,. \end{aligned}$$

The derivative rule for natural log gives us:

$$\begin{aligned}\frac{\partial}{\partial\beta}\mathcal{Z}(s,\beta) &= \mathcal{Z}(s,\beta)\ \frac{\partial}{\partial\beta}\log\mathcal{Z}(s,\beta) \\ &= \mathcal{Z}(s,\beta)\ V(s,\beta)\end{aligned}$$

and as a result we have:

$$V(s,\beta) = \frac{1}{\mathcal{Z}(s,\beta)} \sum_{a\in\mathcal{A}} e^{\beta\,\mathcal{R}(s,a)+\mu}\mathcal{Z}(s+a,\beta)V(s+a,\beta) + \mathcal{R}(s,a)e^{\beta\,\mathcal{R}(s,a)+\mu}\ \mathcal{Z}(s+a,\beta)$$
$$= \frac{1}{\mathcal{Z}(s,\beta)} \sum_{a\in\mathcal{A}} e^{\beta\,\mathcal{R}(s,a)+\mu}\mathcal{Z}(s+a,\beta)\ [V(s+a,\beta)+\mathcal{R}(s,a)]\ . \tag{2.8}$$

Note that the quantities $e^{\beta\,\mathcal{R}(s,a)+\mu}\mathcal{Z}(s+a,\beta)$ inside the summation of Equation 2.8 are positive and sum to $\mathcal{Z}(s,\beta)$ due to the Bellman recursion for $\mathcal{Z}(s,\beta)$ from Equation 2.5. Thus we can view this Bellman equation for $V(s,\beta)$ as an expectation under a distribution on actions, i.e., a *policy*:

$$V(s,\beta) = \sum_{a\in\mathcal{A}} \pi(a\mid s)\ [V(s+a,\beta)+\mathcal{R}(s,a)] \tag{2.9}$$
$$\pi(a\mid s) = \frac{\mathcal{Z}(s+a,\beta)}{\mathcal{Z}(s,\beta)} e^{\beta\,\mathcal{R}(s,a)+\mu}\ . \tag{2.10}$$

The policy π resembles a Boltzmann policy but strictly speaking it is not. A Boltzmann policy π_B selects actions proportionally to the exponential of their expected cumulative reward:

$$\pi_{\mathrm{B}}(a\mid s) \propto \exp\left(\beta\,[\mathcal{R}(s,a)+V(s+a)]\right).$$

In particular, π_B does not take *entropy* into account: if two actions have the same expected optimal value, they will be picked with equal probability regardless of the possibility that one of them could achieve this optimality in a larger number of ways. In the partition function view, π does take entropy into account and to clarify this difference we will look at the two extreme cases $\beta \to \{0,\infty\}$.

When $\beta \to 0$, where the temperature of the system is infinite, rewards become irrelevant and we find that: $\pi(a\mid s) \propto \sum_{\omega\in\Omega(s+a)} e^{\mu|\omega|}$. This means that π is picking action a proportionally to the number of trajectories that begin with $s+a$. Here the

counting of trajectories happens in a weighted way: longer trajectories contribute less than shorter ones. This is different from a Boltzmann policy that would pick actions uniformly at random.

When $\beta \to \infty$, the low-temperature limit, we find in Section 2.A.2 that:

$$\pi(a \mid s) \propto N_{\max}(s+a) \exp\left(\beta \left[\mathcal{R}(s,a) + V(s+a)\right]\right)$$

where $N_{\max}(s+a)$ is a weighted count of the number of **optimal** trajectories that begin at the state $s+a$. Boltzmann policies completely ignore the $N_{\max}$ entropic factor.

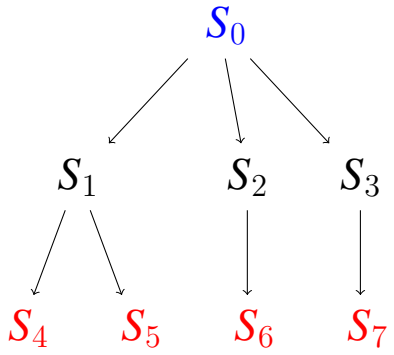


Figure 2.1: Decision Tree MDP

To illustrate this difference more clearly, we consider the deterministic decision tree MDP shown in Figure 2.1 where S_0 is the initial state and the leafs S_4, S_5, S_6, and S_7 are the final states. The arrows represent the actions available at each state. There are no rewards and the boundary conditions are: $\mathcal{R}(S_4) = \mathcal{R}(S_5) = \mathcal{R}(S_6) = 1$ and $\mathcal{R}(S_7) = 0$. This gives us the boundary condition:

$$\mathcal{Z}(S_4, \beta) = \mathcal{Z}(S_5, \beta) = \mathcal{Z}(S_6, \beta) = e^{\beta} \text{ and } \mathcal{Z}(S_7, \beta) = 1.$$

Computing the $\mathcal{Z}$-functions at the intermediate states S_1, S_2 and S_3 we find:

$$\mathcal{Z}(S_1, \beta) = 2e^{\beta+\mu}, \quad \mathcal{Z}(S_2, \beta) = e^{\beta+\mu}, \quad \mathcal{Z}(S_3, \beta) = e^{\mu}.$$

Finally we have $\mathcal{Z}(S_0, \beta) = 3e^{\beta+2\mu} + e^{2\mu}$. The underlying policy for picking the first action is given by:

$$\pi_\beta(1 \mid 0) = \frac{2e^{\beta+2\mu}}{3e^{\beta+2\mu} + e^{2\mu}} \qquad \pi_\beta(2 \mid 0) = \frac{e^{\beta+2\mu}}{3e^{\beta+2\mu} + e^{2\mu}} \qquad \pi_\beta(3 \mid 0) = \frac{e^{2\mu}}{3e^{\beta+2\mu} + e^{2\mu}}. \tag{2.11}$$

When $\beta \to 0$, we get:

$$\pi_0(1 \mid 0) = \frac{1}{2}, \quad \pi_0(2 \mid 0) = \frac{1}{4}, \quad \pi_0(3 \mid 0) = \frac{1}{4}$$

. A Boltzmann policy would pick these three actions with equal probability. The policy π is biased towards the heavier subtree.

When $\beta \to \infty$ we get:

$$\pi_\infty(1 \mid 0) = \frac{2}{3}, \quad \pi_\infty(2 \mid 0) = \frac{1}{3}, \quad \pi_\infty(3 \mid 0) = 0.$$

A Boltzmann policy would pick action 1 and 2 with a probability of $\frac{1}{2}$. π prefers states from which many possible optimal trajectories are possible.

2.3.4 A Planning Algorithm

When the dynamics of the environment are known, it is possible to to learn $\mathcal{Z}(s, \beta)$ by exploiting the Bellman equation (2.5). We denote by $s \to s'$ the property that there exists an action a that takes an agent from state s to state s'. The reward associated with this transition will be denoted $\mathcal{R}(s \to s')$. Let $\mathcal{Z}(\beta) = [\mathcal{Z}(s, \beta)]_{s \in \mathcal{S}}$ be the vector of all partition functions and $C(\beta) \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ be the matrix:

$$C(\beta)_{s,s'} = \mathbf{1}_{s \to s'} e^{\beta \mathcal{R}(s \to s') + \mu} + \mathbf{1}_{s=s' = \text{final state}} \tag{2.12}$$

$C(\beta)$ is a matrix representation of the Bellman operator in Equation 2.5. With these notations, the Bellman equations in (2.5) can be compactly written as:

$$\mathcal{Z}(\beta) = C(\beta)\,\mathcal{Z}(\beta) \tag{2.13}$$

highlighting the fact that $\mathcal{Z}(\beta)$ is a fixed point of the map:

$$\varphi : X \rightarrow \; C(\beta) \;\; X. \tag{2.14}$$

In Appendix 2.A.3, we show that φ is a contraction which makes it possible to learn $\mathcal{Z}(\beta)$ by starting with an initial vector $\mathcal{Z}_0$ having compatible boundary conditions and successively iterating the map φ: $\mathcal{Z}_{n+1} = C(\beta)\;\mathcal{Z}_n$. We could also interpret $\mathcal{Z}(\beta)$ as an eigenvector of $C(\beta)$. In this context, this algorithm is simply doing a power method.

Interestingly, we can learn $\mathcal{Z}(\beta)$ by solving the underdetermined linear system $[I_{|\mathcal{S}|} - C(\beta)]\;\mathcal{Z}(\beta) = 0_{|\mathcal{S}|}$ with the right boundary conditions. We show in Appendix 2.A.2 that the policies learned are related to Boltzmann policies which produce non linear Bellman equations at the value function level:

$$V(s,\beta) = \sum_a \frac{e^{\beta(\mathcal{R}(s,a)+\gamma V(s+a,\beta))}}{\mathcal{W}(s,\beta)}[r_{(s,a)} + \gamma V(s+a,\beta)] \tag{2.15}$$

where γ is the discount factor and $\mathcal{W}(s,\beta) = \sum_a e^{\beta(\mathcal{R}(s,a)+\gamma V(s+a,\beta))}$ is a normalization constant different from $\mathcal{Z}(s,\beta)$. By working with partition functions we transformed a non linear problem into a linear one. This remarkable result is reminiscent of linearly solvable MDPs (Todorov, 2007).

Once $\mathcal{Z}$ is learned the agent's policy is given by: $\mathbb{P}(a \mid s) \propto \; e^{\beta\mathcal{R}(s,a)}\mathcal{Z}(s+a,\beta)$.

2.4 Partition functions for Stochastic MDPs

We now move to the more general MDP setting. The dynamics of the environment can now be stochastic. However, as mentioned at the end of the introduction, we still assume that given an initial state s, an action a, and a landing state s', the reward $\mathcal{R}(s, a, s')$ is deterministic.

2.4.1 A First Attempt: Averaging the Bellman Equation

A first approach to incorporating the stochasticity of the environment is to average the right-hand side of the Bellman Equation 2.5 and define $\mathcal{Z}(s, \beta)$ as the solution of:

$$\begin{aligned} \mathcal{Z}(s, \beta) &= \sum_{a} \mathbb{E}_{s'|s,a} \left[e^{\beta \mathcal{R}(s,a,s') + \mu} \; \mathcal{Z}(s', \beta) \right] \\ &= \sum_{a,s'} \mathbb{P}(s' \mid s, a) \, e^{\beta \mathcal{R}(s,a,s') + \mu} \; \mathcal{Z}(s', \beta) \, . \end{aligned} \tag{2.16}$$

Interestingly, the solution of this equation can be constructed in the same spirit of Section 2.3.1 by summing a functional over the set of trajectories. If we define $L(\omega)$ to be the log likelihood of a trajectory: $L(\omega) = \sum_{t=0}^{|\omega|-1} \log \mathbb{P}(s_{t+1} \mid s_t, a_t)$ then $\mathcal{Z}(s, \beta)$ is defined by

$$\mathcal{Z}(s, \beta) = \sum_{\omega \in \Omega(s)} e^{-\beta E(\omega) + \mu |\omega| + L(\omega)} \, , \tag{2.17}$$

satisfies the Bellman Equation 2.16. The proof can be found in Appendix 2.B.1. In Appendix 2.B.2 we derive the Bellman equation satisfied by the underlying value function $V(s, \beta)$ and we find:

$$V(s, \beta) = \sum_{a,s'} \frac{e^{\beta \mathcal{R}(s,a,s') + \mu} \;\; \mathcal{Z}(s', \beta)}{\mathcal{Z}(s, \beta)} \times \mathbb{P}(s' \mid s, a) \times (\mathcal{R}(s, a, s') + V(s', \beta)) \; . \tag{2.18}$$

This Bellman equation does not correspond to a realistic policy; the policy depends on the landing state s' which is a random variable. The agent's policy and the environment's transitions cannot be decoupled. This is not surprising, from Equation 2.17 we see that $\mathcal{Z}$ puts rewards and transition probabilities on an equal footing. As a result an agent believes they can choose any available transition as long as they are willing to pay the price in log probability. This encourages risky behavior: the agent is encouraged to bet on highly unlikely but beneficial transitions. These observations were also noted in Levine (2018).

2.4.2 A Variational Approach

Constructing a partition function for a stochastic MDP is not straightforward because there are two types of randomness: the first comes from the agent's policy and the second from stochasticity of the environment. Mixing these two sources of randomness can lead to unrealistic policies as we saw in Section 2.4.1. A more principled approach is needed.

We construct a new deterministic MDP $(\tilde{\mathcal{S}}, \tilde{\mathcal{A}}, \tilde{\mathcal{R}}, \tilde{\mathcal{P}})$ from $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P})$. We take $\tilde{\mathcal{S}}$ to be the space of probability distributions over $\mathcal{S}$, similar to belief state representations for partially-observable MDPs (Astrom, 1965; Sondik, 1978; Kaelbling et al., 1998). We make the assumption that the actions $\mathcal{A}$ are the same for all states and take $\tilde{\mathcal{A}} = \mathcal{A}$. For $\rho \in \tilde{\mathcal{S}}$ and $a \in \tilde{\mathcal{A}}$ we define $\tilde{\mathcal{P}}(\rho, a) := {P_a}^T \rho$ where P_a is the transition matrix corresponding to choosing action a in the original MDP. We define $\tilde{\mathcal{R}}(\rho, a) := \mathbb{E}_{s\sim\rho}\left[\mathbb{E}_{s'|s,a}[\mathcal{R}(s, a, s')]\right]$.

$\mathcal{S}$ being finite, it has a finite number M of final states which we denote $\{f_i\}_{i\in\{1,\cdots,M\}}$. The final states of $\tilde{\mathcal{S}}$ are of the form $\rho_f = \sum_{i=1}^{M} \alpha_i \delta_{f_i}$ where $0 \leqslant \alpha_i \leqslant 1$ verify $\sum_{i=1}^{M} \alpha_i = 1$ and δ_{f_i} is a Dirac delta function at state f_i. The intrinsic value ρ_f of such a final state

is then given by $\mathcal{R}(\rho_f) = \sum_{i=1}^{M} \alpha_i \mathcal{R}(f_i)$. This leads to the boundary conditions:

$$\begin{aligned} \mathcal{Z}(\rho_f) &= \exp\left(\beta \sum_{i=1}^{M} \alpha_i \mathcal{R}(s_{f_i})\right) \\ &= \prod_{i=1}^{M} \mathcal{Z}(f_i, \beta)^{\alpha_i} . \end{aligned} \tag{2.19}$$

This new MDP $(\tilde{\mathcal{S}}, \tilde{\mathcal{A}}, \tilde{\mathcal{R}}, \tilde{\mathcal{P}})$ is deterministic, and we can follow the same approach of Section 2.3 and construct a partition function $\mathcal{Z}(\rho, \beta)$ on $\tilde{\mathcal{S}}$. $\mathcal{Z}(s, \beta)$ can be recovered by evaluating $\mathcal{Z}(\delta_s, \beta)$. From this construction we also get that $\mathcal{Z}(\rho, \beta)$ satisfies the following Bellman equation:

$$\mathcal{Z}(\rho, \beta) = \sum_a e^{\beta \mathcal{R}(\rho, a) + \mu} \; \mathcal{Z}({P_a}^T \rho, \beta) . \tag{2.20}$$

Just as it is the case for deterministic MDPs, the Bellman operator associated with this equation is a contraction. This is proved in Appendix 2.B.3. However $\tilde{\mathcal{S}}$ is now infinite which makes solving Equation 2.20 intractable. We adopt a variational approach which consists in finding the best approximation of $\mathcal{Z}(\rho, \beta)$ within a parametric family $\{\mathcal{Z}_\theta\}_{\theta \in \Theta}$. We measure the fitness of a candidate through the following loss function:

$$\Delta(\theta) = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \left(\mathcal{Z}_\theta(\delta_s, \beta) - \sum_a e^{\beta \mathcal{R}(\delta_s, a) + \mu} \; \mathcal{Z}_\theta({P_a}^T \delta_s, \beta) \right)^2$$

.

For illustration purposes, and inspired by the form of the boundary conditions (Equation 2.19), we consider a simple parametric family given by the partition functions of the form $\mathcal{Z}_\theta(\rho) = \prod_{i=1}^{|\mathcal{S}|} {\theta_i}^{\rho_i}$, where $\theta \in \mathbb{R}^{|\mathcal{S}|}$. The optimal θ can be found using usual optimization techniques such as gradient descent. By evaluation of $\mathcal{Z}_\theta$ at $\rho = \delta_{S_i}$ we see that we must have $\theta_i = \mathcal{Z}(\delta_{S_i}) = \mathcal{Z}(S_i)$ and consequently we have $\mathcal{Z}_\theta(\rho) = \prod_{i=1}^{|\mathcal{S}|} \mathcal{Z}(S_i)^{\rho_i}$.

The optimal solution satisfies the following Bellman equation:

$$\mathcal{Z}(s,\beta) \approx \sum_{a} \prod_{s' \in \mathcal{S}} \left[e^{\beta \mathcal{R}(s,a,s') + \mu} \; \mathcal{Z}(s',\beta) \right]^{\mathbb{P}(s'|s,a)} \tag{2.21}$$

The underlying value function verifies:

$$V(s,\beta) \approx \sum_{a,s'} \pi(a \mid s)\, \mathbb{P}(s' \mid s,a)\, (\mathcal{R}(s,a,s') + V(s',\beta))$$

where the policy π is given by $\pi(a \mid s) \propto \prod_{s' \in \mathcal{S}} \left[e^{\beta \mathcal{R}(s,a,s') + \mu} \; \mathcal{Z}(s',\beta) \right]^{\mathbb{P}(s'|s,a)}$. This approach leads to a realistic policy as its only dependency is on the current state, not a future one, unlike the policies arising from Equation 2.18.

2.5 The Model-Free Case

2.5.1 Construction of State-Action-Dependent Partition Function

In a model free setting, where the transition dynamics are unknown, state-only value functions such as $V(s)$ are less useful than state-action value functions such as $Q(s,a)$. Consequently, we will extend our construction to state-action partition functions $\mathcal{Z}(s,a,\beta)$. For a deterministic environment, we extend the construction in Section 2.3 and define $\mathcal{Z}(s,a,\beta)$ by

$$\mathcal{Z}(s,a,\beta) = \sum_{\omega \in \Omega(s,a)} e^{-\beta E(\omega) + \mu |\omega|} \tag{2.22}$$

$$= \sum_{\omega \in \Omega(s,a)} e^{\beta \sum_{i=0}^{|\omega|} r_i + \mu |\omega|} \tag{2.23}$$

where $\Omega(s,a)$ denotes the set of trajectories having $(s_0,a_0) = (s,a)$. Since $\Omega(s) = \bigcup_{a \in \mathcal{A}} \Omega(s,a)$, we have $\mathcal{Z}(s,\beta) = \sum_a \mathcal{Z}(s,a,\beta)$. As a consequence of this

construction, $\mathcal{Z}(s, a, \beta)$ satisfies the following linear Bellman equation:

$$\mathcal{Z}(s, a, \beta) = e^{\beta \mathcal{R}(s,a)+\mu} \sum_{a'} \mathcal{Z}(s + a, a', \beta) \,. \tag{2.24}$$

This Bellman equation can be easily derived by decomposing each trajectory $\omega \in \Omega(s, a)$ into two parts: the first transition resulting from taking initial action a and the remainder of the trajectory ω' which is a member of $\Omega(s+a, a')$ for some action $a' \in \mathcal{A}$. The total energy and length can also be decomposed in the same way, so that:

$$\begin{aligned}
\mathcal{Z}(s, a, \beta) &= \sum_{\omega \in \Omega(s,a)} e^{-\beta E(\omega)+\mu|\omega|} \\
&= \sum_{\omega \in \Omega(s,a)} e^{\beta \sum_{i=0}^{|\omega|} r_i + \mu|\omega|} \\
&= e^{\beta \mathcal{R}(s,a)+\mu} \sum_{\omega \in \Omega(s,a)} e^{\beta \sum_{t=1}^{|\omega|} r_t(\omega) + \mu(|\omega|-1)} \\
&= e^{\beta \mathcal{R}(s,a)+\mu} \sum_{\omega' \in \Omega(s+a)} e^{-\beta E(\omega')+\mu|\omega'|} \\
&= e^{\beta \mathcal{R}(s,a)+\mu} \sum_{a' \in \mathcal{A}} \sum_{\omega' \in \Omega(s+a,a')} e^{-\beta E(\omega')+\mu|\omega'|} \\
&= e^{\beta \mathcal{R}(s,a)+\mu} \sum_{a' \in \mathcal{A}} \mathcal{Z}(s + a, a', \beta).
\end{aligned}$$

In the same spirit of Section 2.3.3, one can show that the average underlying value function $Q(s, a, \beta) = \frac{\partial}{\partial \beta} \log \mathcal{Z}(s, a, \beta)$ satisfies a Bellman equation:

$$Q(s, a, \beta) = \mathcal{R}(s, a) + \sum_{a'} \pi(a' \mid s + a)\ Q(s + a, a', \beta) \tag{2.25}$$

$$\pi(a \mid s) = \frac{\mathcal{Z}(s, a, \beta)}{\sum_{a'} \mathcal{Z}(s, a', \beta)} \tag{2.26}$$

$Q(s, a, \beta)$ can be then reinterpreted as the Q-function of the policy π. Similarly to the results of Section 2.3.3 and Appendix 2.A.2, the policy π can be thought of a Boltzmann policy of parameter β that takes entropy into account. This construction

can be extend to a stochastic environments by following the same approach used in Section 2.4.2.

In the following we show how learning the state-action partition function $\mathcal{Z}(s, a, \beta)$ leads to an alternative approach to model-free reinforcement learning that does not explicitly represent value functions.

2.5.2 A Learning Algorithm

In Q-Learning, the update rule typically consists of a linear interpolation between the current value estimate and the one arising *a posteriori*:

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha\left(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})\right) \tag{2.27}$$

where $\alpha \in [0, 1]$ is the learning rate and γ is the discount factor. For $\mathcal{Z}$-functions we will replace the linear interpolation with a geometric one. We take the update rule for $\mathcal{Z}$-functions to be the following:

$$\mathcal{Z}(s_t, a_t, \beta) \leftarrow \mathcal{Z}(s_t, a_t, \beta)^{1-\alpha} \times \left(e^{\beta r_t + \mu} \sum_{a_{t+1}} \mathcal{Z}(s_{t+1}, a_{t+1}, \beta)\right)^{\alpha}. \tag{2.28}$$

To understand what this update rule is doing, it is insightful to look at how how the underlying Q-function, $Q(s, a) = \frac{\partial}{\partial \beta} \log \mathcal{Z}(s_t, a_t, \beta)$ is updated. We find:

$$Q(s_t, a_t, \beta) \leftarrow (1-\alpha)Q(s_t, a_t, \beta) + \alpha\left(r_t + \sum_{a_{t+1}} \frac{\mathcal{Z}(s_{t+1}, a_{t+1}, \beta)}{\sum_{a'} \mathcal{Z}(s_{t+1}, a', \beta)} Q(s_{t+1}, a_{t+1}, \beta)\right). \tag{2.29}$$

We see that we recover a weighted version of the SARSA update rule. This update rule is referred to as *expected* SARSA. Expected SARSA is known to reduce the variance

in the updates by exploiting knowledge about stochasticity in the behavior policy and hence is considered an improvement over vanilla SARSA (Van Seijen et al., 2009).

Since the underlying update rule is equivalent to the expected SARSA update rule, we can use any exploration strategy that works for expected SARSA. One exploration strategy could be ε-greedy which consists in taking action $a = \text{argmax}_{a \in \mathcal{A}} \mathcal{Z}(s, a, \beta)$ with probability $1 - \varepsilon$ and picking an action uniformly at random with probability ε. Another possibility would be a Boltzmann-like exploration which consists in taking action a with probability $\mathbb{P}(a \mid s) \propto \mathcal{Z}(s, a, \beta)$.

We would like to emphasize that even though the expected SARSA update is not novel, the learned policies through this updates rule are proper to the partition-function approach. In particular, the learned policies $\pi(a \mid s) \propto \mathcal{Z}(s, a, \beta)$ are Boltzmann-like policies with some entropic preference properties as described in Section 2.3.3 and Appendix 2.A.2.

2.6 Conclusion

In this chapter we discussed how planning and reinforcement learning problems can be approached through the tools and abstractions of statistical physics. We started by constructing partition functions for each state of a deterministic MDP and then showed how to extend that definition to the more general stochastic MDP setting through a variational approach. Interestingly, these partition functions have their own Bellman equation making it possible to solve planning and model-free RL problems without explicit reference to value functions. Nevertheless, conventional value functions can be derived from our partition function and interpreted via average energies. Computing the implied value functions can also shed some light on the policies arising from these algorithms. We found that the learned policies are closely related to Boltzmann policies with the additional interesting feature that they take *entropy* into consideration by

favoring states from which many trajectories are possible. Finally, we observed that working with partition functions is more natural in some settings. In a deterministic environment for example, near-optimal Bellman equations become linear which is not the case in a value-function-centric approach.

Appendix

2.A Deterministic MDPs

2.A.1 $\mathcal{Z}(s,\beta)$ is well defined

Proposition 1. *$\mathcal{Z}(s,\beta) = \sum_{\omega\in\Omega(s)} e^{\beta\sum_{i=0}^{|\omega|} r_i + \mu|\omega|}$ is well defined for $\mu < -\log d$.*

Proof. The MDP being finite, $\mathcal{S}$ has a finite number of final state we can then find a constant K such that, for all final states s_f we have $\mathcal{R}(s_f) \leqslant K$.

$$
\begin{aligned}
\mathcal{Z}(s,\beta) &= \sum_{\omega\in\Omega(s)} e^{\beta\sum_{i=0}^{|\omega|} r_i + \mu|\omega|} \\
&= \sum_{\omega\in\Omega(s)} e^{\beta\sum_{i=0}^{|\omega|-1} r_i + \beta\mathcal{R}(s_{|\omega|}) + \mu|\omega|} \\
&\leqslant e^{\beta K} \sum_{\omega\in\Omega(s)} e^{\beta\sum_{i=0}^{|\omega|-1} r_i + \mu|\omega|} \\
&\leqslant e^{\beta K} \sum_{\omega\in\Omega(s)} e^{\mu|\omega|} \\
&\leqslant e^{\beta K} \sum_{n\in\mathbb{N}} e^{\mu n} \sum_{\omega\in\Omega(s),\,|\omega|=n} 1 \\
&\leqslant e^{\beta K} \sum_{n\in\mathbb{N}} e^{\mu n} d^n \\
&= e^{\beta K} \sum_{n\in\mathbb{N}} (e^{\mu+\log d})^n
\end{aligned}
$$

Where used the fact that all rewards $\{r_i\}_{i\in\{0,\dots,|\omega|-1\}}$ are non positive and that the number of available actions at each state is bounded by d. When $\mu < -\log d$, the sum $\sum_{n\in\mathbb{N}}(e^{\mu+\log d})^n$ becomes convergent and $\mathcal{Z}(s,\beta)$ is well defined. □

Remark 2.A.1. *$\mu < -\log d$ is a sufficient condition, but not a necessary one. $\mathcal{Z}(s,\beta)$ could be well defined for all values of μ. This happens for instance when $\Omega(s)$ is finite for all s.*

2.A.2 The underlying policy is Boltzmann-like

For high values of β, the sum $\sum_{\omega\in\Omega(s)} e^{\beta\sum_{i=0}^{|\omega|} r_i+\mu|\omega|}$ will become dominated by the contribution of few of its terms. As $\beta \to +\infty$, the sum will be dominated by the contribution of the paths with the biggest reward. We have

$$\log \mathcal{Z}(s,\beta) \underset{\beta\to\infty}{\sim} \beta \max\left\{\sum_{i=0}^{|\omega|} r_i(\omega),\ \omega\in\Omega(s)\right\}$$

We see that $V(s,\beta) = \frac{\partial}{\partial\beta}\log\mathcal{Z}(s,\beta) \underset{\beta\to\infty}{\to} \max\left\{\sum_{i=0}^{|\omega|} r_i(\omega),\ \omega\in\Omega(s)\right\}$.

Since the MDP is finite and deterministic, it has a finite number of transitions and rewards. Consequently, the set $\left\{\sum_{i=0}^{|\omega|} r_i(\omega),\ \omega\in\Omega(s)\right\}$ takes discrete values, in particular, there is a finite gap Δ between the maximum value and the second biggest value of this set. Let's denote by $\Omega_{\max}(s)$ the set of trajectories that achieve this maximum and by $N_{\max}(s) = \sum_{\omega\in\Omega_{\max}(s)} e^{\mu|\omega|}$.

$N_{\max}(s)$ counts the number of trajectories $\Omega_{\max}(s)$ in a weighted way: longer trajectories contribute less than shorter ones. It is a measure of the size of $\Omega_{\max}(s)$ that takes into account our preference for shorter trajectories. Putting everything together we get:

$$\left(\frac{\mathcal{Z}(s,\beta)}{e^{\beta V(s,\beta)}} - N_{\max}(s)\right) \underset{\beta\to\infty}{\leqslant} e^{-\beta\Delta}\sum_{\omega\in\Omega(s)} e^{\mu|\omega|} \underset{\beta\to\infty}{\to} 0$$

This shows that $\mathcal{Z}(s,\beta) \underset{\beta\to\infty}{\sim} N_{\max}(s)\ e^{\beta V(s,\beta)}$, which results in the following policy for $\beta >> 1$:

$$\pi(a \mid s) \underset{\beta\to\infty}{\propto} N_{\max}(s+a)e^{\beta(\mathcal{R}(s,a)+V(s+a,\beta))}$$

π differs from a traditional Boltzmann policy in the following way: if we have two actions a_1 and a_2 such that $\mathcal{R}(s,a_1)+V(s+a_1,\beta) = \mathcal{R}(s,a_2)+V(s+a_2,\beta)$ but there are twice more optimal trajectories spanning from $s+a_1$ than there are from $s+a_2$ then action a_1 will be chosen twice as often as a_2. This is to contrast with the usual Boltzmann policy that will pick a_1 and a_2 with equal probability. When $N_{\max}(s)$ is the same for all s, we recover a Boltzmann policy. When $\beta \to +\infty$ the policy converges to a an optimal policy and V converges to the optimal value function.

2.A.3 $X \to C(\beta)X$ is a contraction

Proposition 2. *Let* $\mathcal{X}(\beta) = \left\{Z \in \mathbb{R}_+^{|\mathcal{S}|} \text{ such for all final states } s_f \text{ we have } Z_{s_f} = e^{\beta\mathcal{R}(s_f)}\right\}$ *and let* $C(\beta)_{s,s'} = \mathbf{1}_{s\to s'}e^{\beta\mathcal{R}(s\to s')+\mu} + \mathbf{1}_{s=s'=\text{final state}}$. *The map defined by*

$$\psi : \begin{cases} \mathcal{X}(\beta) & \to \mathcal{X}(\beta) \\ X & \to C(\beta)\,X \end{cases}$$

is a contraction for the sup-norm: $||x||_\infty = \max_{i\in\{1,\cdots,|\mathcal{S}|\}} |x_i|$.

Proof. $\mathcal{X}(\beta)$ is the set of all possible partition functions with compatible boundary conditions. The matrix $C(\beta)$ is more explicitly defined by:

$$C(\beta)_{s,s'} = \begin{cases} 1 & \text{if s = s' and state s is a final state.} \\ 0 & \text{if there is no one step transition from state s to state s'.} \\ e^{\beta\mathcal{R}(s\to s')+\mu} & \text{if the transition from state s to state s' has reward } \mathcal{R}(s\to s'). \end{cases}$$

Because $C(\beta)_{s,s} = 1$ when s is a final state, the map ψ is well defined (i.e. $\mathcal{X}(\beta) \to \mathcal{X}(\beta)$). Since the MDP is finite, it has a finite number of final state so there exists a constant K such that, for all final states s_f we have $\mathcal{R}(s_f) \leqslant K$.

Let $X_1,\ X_2\ \in \mathcal{X}(\beta)$ we have:

$$\|\psi(X_1) - \psi(X_2)\|_\infty = \max_{i \in \{1,\cdots,|\mathcal{S}|\}} \left|(C(\beta)X_1 - C(\beta)X_2)_i\right|$$

Without loss of generality we can assume that the MDP has m final states that are labeled $|\mathcal{S}| - m + 1, \cdots, |\mathcal{S}|$. Under this assumption we have:

$$\max_{i \in \{1,\cdots,|\mathcal{S}|\}} |(X_1 - X_2)_i| = \max_{i \in \{1,\cdots,|\mathcal{S}|-m\}} |(X_1 - X_2)_i|$$

This is because X_1 and X_2 have the same boundary conditions:

$$\forall\, s_f \in \{|\mathcal{S}| - m + 1, \cdots, |\mathcal{S}|\},\ (X_1)_{s_f} = (X_2)_{s_f}$$

. Since $C(\beta)_{s_f,s_f} = 1$ if s_f is the index a final state, $C(\beta)X_1$ and $C(\beta)X_2$ still have the same boundary conditions, we have:

$$\forall s_f \in \{|\mathcal{S}| - m + 1, \cdots, |\mathcal{S}|\},\ [C(\beta)X_1]_{s_f} = [C(\beta)X_2]_{s_f}$$

. This gives us:

$$\max_{s \in \{1,\cdots,|\mathcal{S}|\}} \left|(C(\beta)X_1 - C(\beta)X_2)_s\right| = \max_{s \in \{1,\cdots,|\mathcal{S}|-m\}} \left|(C(\beta)X_1 - C(\beta)X_2)_s\right|$$

For $s \in \{1, \cdots, |\mathcal{S}|\}$, we have: $\left|(C(\beta)X_1 - C(\beta)X_2)_s\right| = \left|\sum_{s'=1}^{|\mathcal{S}|} [C(\beta)]_{s,s'}\ (X_1 - X_2)_{s'}\right|$.

Since there are at most d available actions at each state and the environment

is deterministic, at most d coefficients $C(\beta)_{s,s'}$ in this sum are non zero. Because the rewards are non positive, the non zero ones can be bounded by e^{μ}.

Putting all these pieces together we can write:

$$\left|\sum_{s'=1}^{|\mathcal{S}|}[C(\beta)]_{s,s'}\ (X_1 - X_2)_{s'}\right| \leqslant d \times e^{\mu}\|X_1 - X_2\|_{\infty}$$

.

Finally we get:

$$\|C(\beta)X_1 - C(\beta)X_2\|_{\infty} \leqslant \underbrace{d \times e^{\mu}}_{<1 \text{ because } \mu<-\log d} \|X_1 - X_2\|_{\infty}$$

This proves that ψ is a contraction. □

Remark 2.A.2. *We see here another mathematical similarity between the discount factor $\gamma < 1$ usually used in RL and the chemical potential $\mu < -\log d$. They both ensure that the Bellman operators are contractions.*

2.B Stochastic MDPs

2.B.1 Averaging the Bellman Equation and adding a likelihood cost are equivalent

Proposition 3. *The partition function* $\mathcal{Z}(s,\beta)$ *defined by* $\mathcal{Z}(s,\beta) := \sum_{\omega\in\Omega(s)} e^{-\beta E(\omega)+\mu|\omega|+L(\omega)}$ *satisfies the following Bellman equation:*

$$\mathcal{Z}(s,\beta) = \sum_a \mathbb{E}_{s'|s,a}\left[e^{\beta\mathcal{R}(s,a,s')+\mu}\ \mathcal{Z}(s',\beta)\right]$$

Proof. The proof follows the same path as the one in Section 2.3.2. We decompose each trajectory $\omega \in \Omega$ into two parts: the first transition resulting from taking a first action a and the rest of the trajectory ω'. The energy, the length and the likelihood of the trajectory can be decomposed in a similar way as the sum of the contribution of the first transition and the contribution of the rest of the trajectory. We get:

$$\begin{aligned}
\mathcal{Z}(s,\beta) &= \sum_{\omega\in\Omega(s)} e^{-\beta E(\omega)+\mu|\omega|+L(\omega)} \\
&= \sum_{a,s'} e^{\beta\mathcal{R}(s,a,s')+\mu+\log(\mathbb{P}(s'|s,a)} \sum_{\omega'\in\Omega(s')} e^{-\beta E(\omega')+\mu|\omega'|+L(\omega')} \\
&= \sum_{a,s'} e^{\beta\mathcal{R}(s,a,s')+\mu+\log(\mathbb{P}(s'|s,a)}\ \mathcal{Z}(s',\beta) \\
&= \sum_{a,s'} \mathbb{P}(s' \mid s,a)\ e^{\beta\mathcal{R}(s,a,s')+\mu}\ \mathcal{Z}(s',\beta) \\
&= \sum_a \mathbb{E}_{s'|s,a}\left[e^{\beta\mathcal{R}(s,a,s')+\mu}\ \mathcal{Z}(s',\beta)\right]
\end{aligned}$$

This proves the equivalence. □

2.B.2 Deriving the Unrealistic Bellman Equation

Proposition 4. *The value function* $V(s,\beta) = \frac{\partial}{\partial\beta}\log \mathcal{Z}(s,\beta)$ *where*

$$\mathcal{Z}(s,\beta) = \sum_{\omega\in\Omega(s)} e^{-\beta E(\omega)+\mu|\omega|+L(\omega)}$$

satisfies the following Bellman equation:

$$V(s,\beta) = \sum_{a,s'} \frac{e^{\beta\mathcal{R}(s,a,s')+\mu}\ \mathcal{Z}(s',\beta)}{\mathcal{Z}(s,\beta)}\ \mathbb{P}(s' \mid s,a)\left[\mathcal{R}(s,a,s') + V(s',\beta)\right]$$

Proof. From Appendix 2.B.1 we known that $\mathcal{Z}(s,\beta)$ satisfies the Bellman equation: $\mathcal{Z}(s,\beta) = \sum_a \mathbb{E}_{s'|s,a}\left[e^{\beta\mathcal{R}(s,a,s')+\mu}\ \mathcal{Z}(s',\beta)\right]$.

$$\begin{aligned}
V(s,\beta) &= \frac{\partial}{\partial\beta}\log\mathcal{Z}(s,\beta)\\
&= \frac{\partial}{\partial\beta}\log\left(\sum_a \mathbb{E}_{s'|s,a}\left[e^{\beta\mathcal{R}(s,a,s')+\mu}\ \mathcal{Z}(s',\beta)\right]\right)\\
&= \frac{1}{\mathcal{Z}(s,\beta)}\ \frac{\partial}{\partial\beta}\left(\sum_a \mathbb{E}_{s'|s,a}\left[e^{\beta\mathcal{R}(s,a,s')+\mu}\ \mathcal{Z}(s',\beta)\right]\right)\\
&= \frac{1}{\mathcal{Z}(s,\beta)}\ \sum_a \mathbb{E}_{s'|s,a}\left[\frac{\partial}{\partial\beta}\left(e^{\beta\mathcal{R}(s,a,s')+\mu}\ \mathcal{Z}(s',\beta)\right)\right]\\
&= \frac{1}{\mathcal{Z}(s,\beta)}\ \sum_a \mathbb{E}_{s'|s,a}\left[\mathcal{R}(s,a,s')\ e^{\beta\mathcal{R}(s,a,s')+\mu}\ \mathcal{Z}(s',\beta) + e^{\beta\mathcal{R}(s,a,s')+\mu}\ \frac{\partial}{\partial\beta}\mathcal{Z}(s',\beta)\right]\\
&= \frac{1}{\mathcal{Z}(s,\beta)}\ \sum_a \mathbb{E}_{s'|s,a}\left[\left(\mathcal{R}(s,a,s') + \frac{\frac{\partial}{\partial\beta}\mathcal{Z}(s',\beta)}{\mathcal{Z}(s',\beta)}\right)\ e^{\beta\mathcal{R}(s,a,s')+\mu}\ \mathcal{Z}(s',\beta)\right]\\
&= \frac{1}{\mathcal{Z}(s,\beta)}\ \sum_a \mathbb{E}_{s'|s,a}\left[\left(\mathcal{R}(s,a,s') + \frac{\partial}{\partial\beta}\log\mathcal{Z}(s',\beta)\right)\ e^{\beta\mathcal{R}(s,a,s')+\mu}\ \mathcal{Z}(s',\beta)\right]\\
&= \frac{1}{\mathcal{Z}(s,\beta)}\ \sum_a \mathbb{E}_{s'|s,a}\left[\left(\mathcal{R}(s,a,s') + V(s',\beta)\right)\ e^{\beta\mathcal{R}(s,a,s')+\mu}\ \mathcal{Z}(s',\beta)\right]\\
&= \sum_{a,s'} \frac{e^{\beta\mathcal{R}(s,a,s')+\mu}\ \mathcal{Z}(s',\beta)}{\mathcal{Z}(s,\beta)}\ \mathbb{P}(s' \mid s,a)\left[\mathcal{R}(s,a,s') + V(s',\beta)\right]
\end{aligned}$$

□

2.B.3 The Bellman operator of $\mathcal{Z}(\rho, \beta)$ is a contraction

Proposition 5. *Let* $\mathcal{D} = \{\alpha \in \mathbb{R}^{|\mathcal{S}|}$ *such that* $\forall i \in 1, \cdots, |\mathcal{S}|,\ 0 \leqslant \alpha_i \leqslant 1$ *and* $\sum_{i=1}^{|\mathcal{S}|} \alpha_i = 1\}$ *and*

$$\mathcal{X}(\beta) = \left\{ X \in C^0(\mathcal{D}, \mathbb{R}) \ \ s.t.\ X(\rho_f) = \exp\left[\beta \sum_{i=1}^{M} \alpha_i \mathcal{R}(f_i)\right] \right.$$
$$\left. \text{for mixtures of final states } \rho_f = \sum_{i=1}^{M} \alpha_i \delta_{f_i} \right\} \quad .$$

The map defined by

$$\psi : \begin{cases} \mathcal{X}(\beta) & \to \mathcal{X}(\beta) \\ X & \to \begin{cases} \mathcal{D} & \to \mathbb{R} \\ \rho & \to \sum_a e^{\beta \mathcal{R}(\rho, a) + \mu}\ X({P_a}^T \rho, \beta) \end{cases} \end{cases}$$

is a contraction for the sup-norm: $\|X\|_\infty = \max_{\rho \in \mathcal{D}} |X(\rho)|$.

Proof. $\mathcal{D}$ is the standard $(|\mathcal{S}| - 1)$-simplex in $\mathbb{R}^{|\mathcal{S}|}$ and $\mathcal{X}(\beta)$ be the set of continuous functions on $\mathcal{D}$ satisfying the right boundary conditions. The original MDP is finite, consequently it has a finite number M of final state and it is possible to find a constant K such that, for all final states s_f we have $\mathcal{R}(s_f) \leqslant K$.

Let $X_1, X_2 \in \mathcal{X}(\beta)$. X_1 and X_2 have the same boundary conditions by construction. Not only that, $\psi(X_1)$ and $\psi(X_2)$ have also the same boundary conditions since the map ψ doesn't alter boundary conditions. Consequently we have:

$$\|\psi(X_1) - \psi(X_2)\|_\infty = \max_{\rho \in \mathcal{D}} |\psi(X_1)(\rho) - \psi(X_2)(\rho)| = \max_{\rho \in \mathcal{D},\ \rho \text{ non final}} |\psi(X_1)(\rho) - \psi(X_2)(\rho)|$$

Finally we can write:

$$
\begin{aligned}
\|\psi(X_1) - \psi(X_2)\|_\infty &= \max_{\rho \in \mathcal{D},\ \rho \text{ non final}} |\psi(X_1)(\rho) - \psi(X_2)(\rho)| \\
&= \max_{\rho \in \mathcal{D},\ \rho \text{ non final}} \left| \sum_a e^{\beta \mathcal{R}(\rho,a)+\mu} \left[X_1({P_a}^T \rho, \beta) - X_2({P_a}^T \rho, \beta) \right] \right| \\
&\leqslant \max_{\rho \in \mathcal{D},\ \rho \text{ non final}} \left| \sum_a e^{\beta \mathcal{R}(\rho,a)+\mu} \right| \times \|X_1 - X_2\|_\infty \\
&\leqslant \underbrace{d \times e^{\mu}}_{<1 \text{ because } \mu < -\log d} \|X_1 - X_2\|_\infty
\end{aligned}
$$

Where we use the fact that all rewards are non positive and that the number of available actions is bounded by d. This concludes the proof that the Bellman operator of $\mathcal{Z}(\rho, \beta)$ is a contraction.

This proof is generalization of the proof presented in Appendix 2.A.3 for MDPs with finite state spaces. □

Chapter 3

Why Generalization in RL is Difficult: Epistemic POMDPs and Implicit Partial Observability

Abstract

Generalization is a central challenge for the deployment of reinforcement learning (RL) systems in the real world. In this chapter, we show that the sequential structure of the RL problem necessitates new approaches to generalization beyond the well-studied techniques used in supervised learning. While supervised learning methods can generalize effectively without explicitly accounting for epistemic uncertainty, we show that, perhaps surprisingly, this is not the case in RL. We show that generalization to unseen test conditions from a limited number of training conditions induces implicit partial observability, effectively turning even fully-observed MDPs into POMDPs. Informed by this observation, we recast the problem of generalization in RL as solving the induced partially observed Markov decision process, which we call the epistemic POMDP. We demonstrate the failure modes of algorithms that do not appropriately

handle this partial observability, and suggest a simple ensemble-based technique for approximately solving the partially observed problem. Empirically, we demonstrate that our simple algorithm derived from the epistemic POMDP achieves significant gains in generalization over current methods on the Procgen benchmark suite.

3.1 Introduction

Generalization is a central challenge in machine learning. However, much of the research on reinforcement learning (RL) has been concerned with the problem of optimization: how to master a specific task through online or logged interaction. Generalization to new test-time contexts has received comparatively less attention, although several works have observed empirically (Farebrother et al., 2018; Zhang et al., 2018c; Justesen et al., 2018; Song et al., 2020) that generalization to new situations poses a significant challenge to RL policies learned from a fixed training set of situations. In standard supervised learning, it is known that in the absence of distribution shift and with appropriate inductive biases, optimizing for performance on the training set (i.e., empirical risk minimization) translates into good generalization performance. It is tempting to suppose that the generalization challenges in RL can be solved in the same manner as empirical risk minimization in supervised learning: when provided a training set of contexts, learn the optimal policy within these contexts and then use that policy in new contexts at test-time.

Perhaps surprisingly, we show that such "empirical risk minimization" approaches can be sub-optimal for generalizing to new contexts in RL, even when these new contexts are drawn from the same distribution as the training contexts. As an anecdotal example of why this sub-optimality arises, imagine a robotic zookeeper for feeding otters that must be trained on some set of zoos. When placed in a new zoo, the robot must find and enter the otter enclosure. It can use one of two strategies:

either peek through all the habitat windows looking for otters, which succeeds with 95% probability in all zoos, or to follow an image of a hand-drawn map of the zoo that unambiguously identifies the otter enclosure, which will succeed as long as the agent is able to successfully parse the image. In every training zoo, the otters can be found more reliably using the image of the map, and so an agent trained to seek the optimal policy in the training zoos would learn a classifier to predict the identity of the otter enclosure from the map, and enter the predicted enclosure. This classification strategy is optimal on the training environments because the agent can learn to perfectly classify the training zoo maps, but it is *sub-optimal* for generalization, because the learned classifier will never be able to perfectly classify every new zoo map at test-time. Note that this task is *not* partially observed, because the map provides full state information even for a memoryless policy. However, if the learned map classifier succeeds on anything less than 95% of new zoos at test-time, the strategy of peeking through the windows, although always sub-optimal in the training environments, turns out to be a more reliable strategy for finding the otter habitat in a *new* zoo, and results in higher expected returns at test-time.

Although with enough training zoos, the zookeeper can learn a policy by solving the map classification problem, to generalize optimally when given a limited number of zoos requires a more intricate policy that is not learned by standard RL methods. How can we more generally describe the set of behaviors needed for a policy to generalize from a finite number of training contexts in the RL setting? We make the observation that, even in fully-observable domains, the agent's epistemic uncertainty renders the environment *implicitly* partially observed at test-time. In the zookeeper example, although the hand-drawn map provides the exact location of the otter enclosure (and so the enclosure's location is technically fully observed), the agent cannot identify the true parameters of the map classifier from the small set of maps seen at training time, and so the location of the otters is implicitly obfuscated from the agent. We formalize this

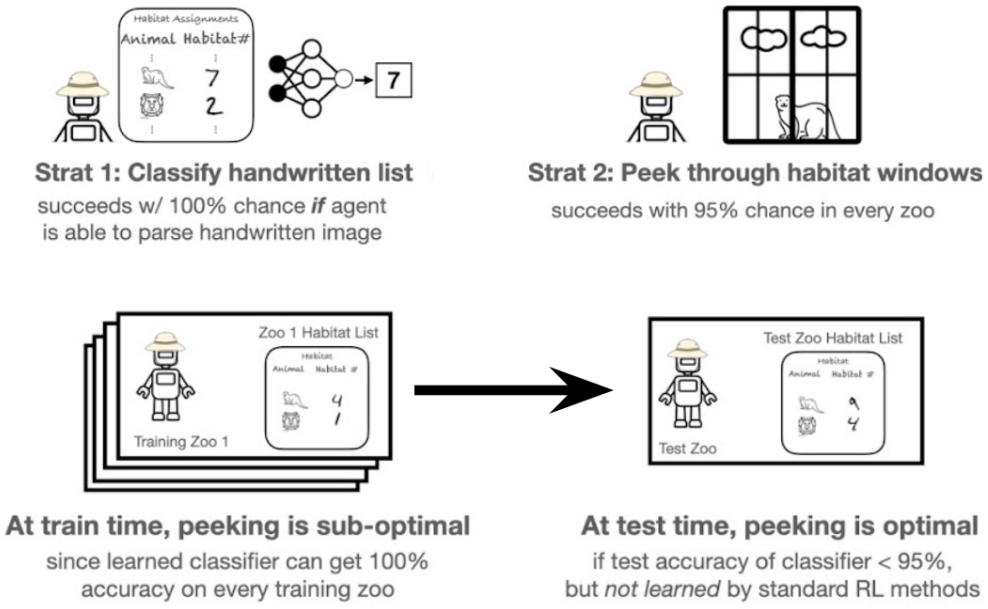


Figure 3.1.1: **Visualization of the robotic zookeeper example.** Standard RL algorithms learn the classifier strategy, since it is optimal in every training zoo, but this strategy is sub-optimal for generalization because peeking generalizes better than the classifier at test-time. This failure occurs due to the following disconnect: while the task is *fully-observed* since the image uniquely specifies the location of the otter habitat, to an agent that has limited training data, the location is *implicitly partially observed at test-time* because of the agent's epistemic uncertainty about the parameters of the image classifier.

observation, and show that generalizing optimally at test-time corresponds to solving a partially-observed Markov decision process that we call an **epistemic POMDP**, induced by the agent's epistemic uncertainty about the test environment.

That uncertainty about MDP parameters can be modelled as a POMDP is well-studied in Bayesian RL when training and testing on a single task in an online setting, primarily in the context of exploration (Dearden et al., 1998; Duff and Barto, 2002; Strens, 2000; Ghavamzadeh et al., 2015). However, as we will discuss, this POMDP interpretation has significant consequences for the generalization problem in RL, where an agent cannot collect more data online, and must instead learn a policy from a fixed set of training contexts that generalizes to new contexts at test-time. We show that standard RL methods that do not explicitly account for this implicit partial observability can be arbitrarily sub-optimal for test-time generalization in theory and

in practice. The epistemic POMDP underscores the difficulty of the generalization problem in RL, as compared to supervised learning, and provides an avenue for understanding how we should approach generalization under the sequential nature and non-uniform reward structure of the RL setting. Maximizing expected return in an approximation of the epistemic POMDP emerges as a principled approach to learning policies that generalize well, and we propose LEEP, an algorithm that uses an ensemble of policies to approximately learn the Bayes-optimal policy for maximizing test-time performance.

The primary contribution of this chapter is to use Bayesian RL techniques to reframe generalization in RL as the problem of solving a partially observed Markov decision process, which we call the *epistemic POMDP*. The epistemic POMDP highlights the difficulty of generalizing well in RL, as compared to supervised learning. We demonstrate the practical failure modes of standard RL methods, which do not reason about this partial observability, and show that maximizing test-time performance may require algorithms to explicitly consider the agent's epistemic uncertainty during training. Our work highlights the importance of not only finding ways to help neural networks in RL generalize better, but also on learning policies that degrade gracefully when the underlying neural network eventually does fail to generalize. Empirically, we demonstrate that LEEP, which maximizes return in an approximation to the epistemic POMDP, achieves significant gains in test-time performance over standard RL methods on several ProcGen benchmark tasks.

3.2 Related Work

Many empirical studies have demonstrated the tendency of RL algorithms to overfit significantly to their training environments (Farebrother et al., 2018; Zhang et al., 2018c; Justesen et al., 2018; Song et al., 2020), and the more general increased

difficulty of learning policies that generalize in RL as compared to seemingly similar supervised learning problems (Zhang et al., 2018b,a; Whiteson et al., 2011; Liu et al., 2020). These empirical observations have led to a newfound interest in algorithms for generalization in RL, and the development of benchmark RL environments that focus on generalization to new contexts from a limited set of training contexts sharing a similar structure (state and action spaces) but possibly different dynamics and rewards (Nichol et al., 2018; Cobbe et al., 2019; Kuttler et al., 2020; Cobbe et al., 2020; Stone et al., 2021).

Generalization in RL. Approaches for improving generalization in RL have fallen into two main categories: improving the ability of function approximators to generalize better with inductive biases, and incentivizing behaviors that are easier to generalize to unseen contexts. To improve the representations learned in RL, prior work has considered imitating environment dynamics (Jaderberg et al., 2017; Stooke et al., 2020), seeking bisimulation relations (Zhang et al., 2020; Agarwal et al., 2021), and more generally, addressing representational challenges in the RL optimization process (Igl et al., 2019; Jiang et al., 2020). In image-based domains, inductive biases imposed via neural network design have also been proposed to improve robustness to certain factors of variation in the state (Lee et al., 2020a; Kostrikov et al., 2020; Raileanu et al., 2020). The challenges with generalization in RL that we will describe in this chapter stem from the deficiencies of MDP objectives, and cannot be fully solved by choice of representations or functional inductive biases. In the latter category, one approach is domain randomization, varying environment parameters such as coefficients of friction or textures, to obtain behaviors that are effective across many candidate parameter settings (Sadeghi and Levine, 2017; Tobin et al., 2017; Rajeswaran et al., 2017; Peng et al., 2018; Kang et al., 2019). Domain randomization sits within a class of methods that seek robust policies by injecting noise into the agent-environment loop, whether in the state (Stulp et al., 2011), the action (e.g., via max-entropy RL) (Cobbe et al.,

2019), or intermediary layers of a neural network policy (e.g., through information bottlenecks) (Igl et al., 2019; Lu et al., 2020). In doing so, these methods effectively introduce partial observability into the problem; while not necessarily equivalent to that of the epistemic POMDP, it may indicate why these methods generalize well empirically.

Bayesian RL: Our work recasts generalization in RL within the Bayesian RL framework, the problem of acting optimally under a belief distribution over MDPs (see Ghavamzadeh et al. (Ghavamzadeh et al., 2015) for a survey). Bayesian uncertainty has been studied in many sub-fields of RL (Ramachandran and Amir, 2007; Lazaric and Ghavamzadeh, 2010; Jeon et al., 2018; Zintgraf et al., 2020), the most prominent being for exploration and learning efficiently in the online RL setting. Bayes-optimal behavior in RL is often reduced to acting optimally in a POMDP, or equivalently, a belief-state MDP (Duff and Barto, 2002), of which our epistemic POMDP is a specific instantiation. Learning the Bayes-optimal policy exactly is intractable in all but the simplest problems (Weber, 1992; Poupart et al., 2006), and many works in Bayesian RL have studied relaxations that remain asymptotically optimal for learning, for example with value of perfect information (Dearden et al., 1998, 1999) or Thompson sampling (Strens, 2000; Osband et al., 2013; Russo and Roy, 2014). Our main contribution is to revisit these classic ideas in the context of generalization for RL. We find that the POMDP interpretation of Bayesian RL (Dearden et al., 1998; Duff and Barto, 2002; Ross et al., 2007) provides new insights on inadequacies of current algorithms used in practice, and explains why generalization in RL can be more challenging than in supervised learning. Being Bayesian in the generalization setting also requires new tools and algorithms beyond those classically studied in Bayesian RL, since test-time generalization is measured using regret over a *single* evaluation episode, instead of throughout an online training process. As a result, algorithms and policies that minimize short-term regret (i.e., are more exploitative) are preferred over traditional

algorithms like Thompson sampling that explore thoroughly to ensure asymptotic optimality at the cost of short-term regret.

3.3 Problem Setup

We consider the problem of learning RL policies given a set of training contexts that generalize well to new unseen contexts. This problem can be formalized in a Markov decision process (MDP) where the agent does not have full access to the MDP at training time, but only particular initial states or conditions. Before we describe what this means, we must describe the MDP $\mathcal{M}$, which is given by a tuple $(\mathcal{S}, \mathcal{A}, r, T, \rho, \gamma)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, Markovian transition function $T(s_{t+1}|s_t, a_t)$, bounded reward function $r(s_t, a_t)$, and initial state distribution $\rho(s_0)$. A policy π induces a discounted state distribution $d^\pi(s) = (1-\gamma)\mathbb{E}_\pi[\sum_{t\geqslant 0} \gamma^t 1(s_t = s)]$, and achieves return $J_\mathcal{M}(\pi) = \mathbb{E}_\pi[\sum_{t\geqslant 0} \gamma^t r(s_t, a_t)]$ in the MDP. Classical results establish that a deterministic Markovian (memoryless) policy π^* maximizes this objective amongst all history-dependent policies.

We focus on generalization in contextual MDPs where the agent is only trained on a training set of contexts, and seeks to generalize well to new contexts. A contextual MDP is an MDP in which the state can be decomposed as $s_t = (c, s'_t)$, a context vector $c \in \mathcal{C}$ that remains constant throughout an episode, and a sub-state $s' \in \mathcal{S}'$ that may vary: $\mathcal{S} := \mathcal{C} \times \mathcal{S}'$. Each context vector corresponds to a different situation that the agent might be in, each with slightly different dynamics and rewards, but some shared structure across which an agent can generalize. During training, the agent is allowed to interact only within a sampled subset of contexts $\mathcal{C}_{\text{train}} \subset \mathcal{C}$. The generalization performance of the agent is measured by the return of the agent's policy in the full contextual MDP $J(\pi)$, corresponding to expected performance when placed in potentially new contexts. While our examples and experiments will be in contextual

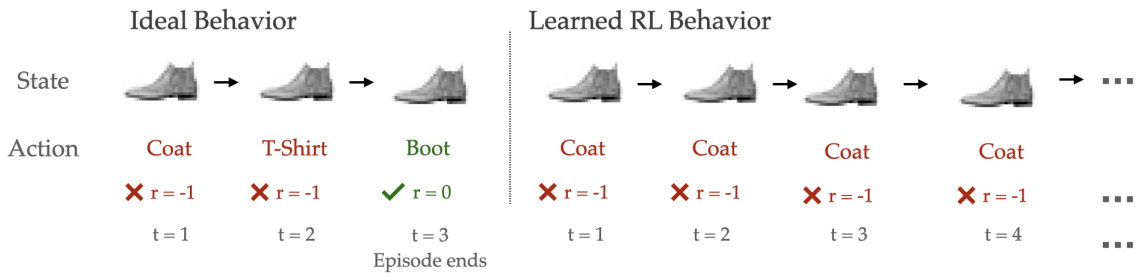


Figure 3.4.1: **Sequential Classification RL Problem.** In this task, an agent must keep guessing the label for an image until it gets it correct. To avoid low test return, policies should change actions if the label guessed was incorrect, but standard RL methods fail to do so, instead guessing the same incorrect label repeatedly.

MDPs, our theoretical results also apply to other RL generalization settings where the full MDP cannot be inferred unambiguously from the data available during training, for example in offline reinforcement learning (Levine et al., 2020).

3.4 Warmup: A Sequential Classification RL Problem

We begin our study of generalization in RL with an example problem that is set up to be close to a supervised learning task where generalization is relatively well understood: image classification on the FashionMNIST dataset (Xiao et al., 2017). In this environment (visualized in Figure 3.4.1), an image from the dataset is sampled (the context) at the beginning of an episode and held fixed; the agent must identify the label of the image to complete the episode. If the agent guesses correctly, it receives a reward of 0 and the episode ends; if incorrect, it receives a reward of -1 and the episode continues, so it must attempt another guess for the *same* image at the next time step. This RL problem is near identical to supervised classification, the core distinction being that an agent may interact with the same image over several timesteps in an episode instead of only one attempt as in supervised learning. Note

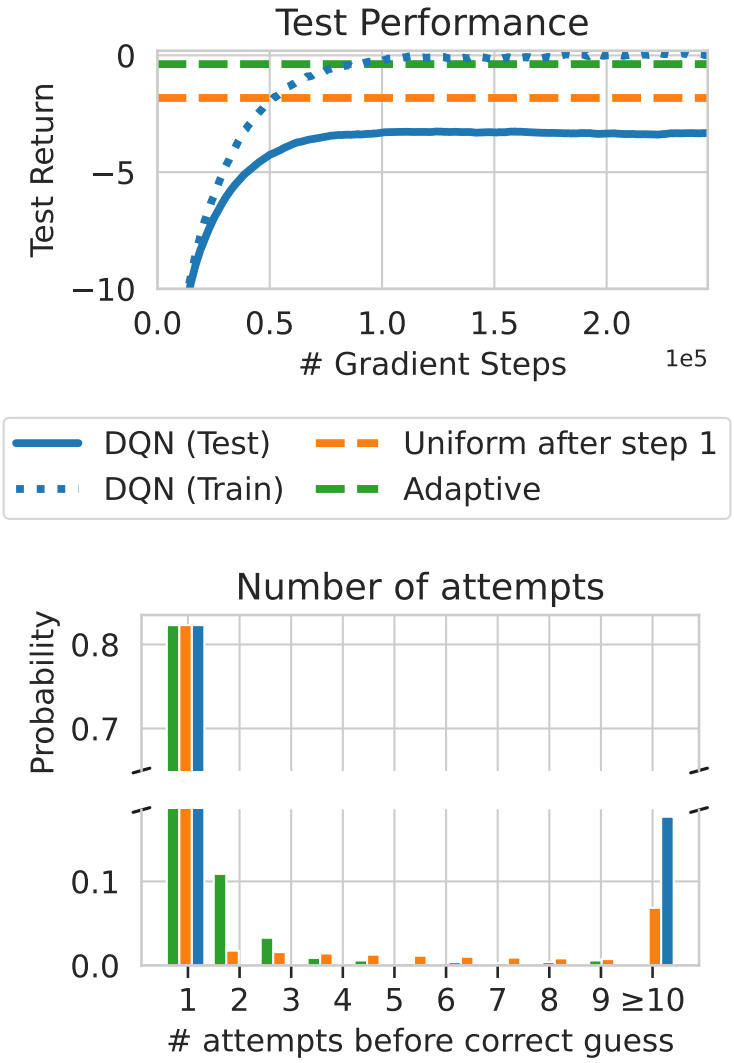


Figure 3.4.2: **DQN on RL FashionMNIST.** DQN achieves lower test performance than simple variants that leverage the structure of the RL problem.

that since episodes may last longer than a single timestep, this problem is not a contextual bandit.

The optimal policy in both the one-step and sequential RL version of the problem deterministically outputs the correct label for the image, because the image fully determines the label (in other words, it is a fully observed MDP). However, this optimal strategy generally cannot be learned from a finite training set, since some generalization error is unavoidable. With a fixed training set, the strategy for generalizing in classification remains the same: deterministically choose the label the agent is most confident about. However, the RL setting introduces two new factors: the agent gets multiple tries at classifying the same image, and it knows if an attempted label is incorrect. To generalize best to new test images, an RL policy must leverage this additional structure, for example by trying many possible labels, or by changing actions if the previous guess was incorrect.

A standard RL algorithm, which estimates the optimal policy on the empirical MDP defined by the dataset of training images does not learn to leverage these factors, and instead learns behavior highly sub-optimal for generalization. We obtained a policy by running DQN (Mnih et al., 2015) (experimental details in Appendix 3.A.1), whose policy deterministically chooses the same label for the image at every timestep. Determinism is not specific to DQN, and is inevitable in any RL method that models the problem as an MDP because the optimal policy in the MDP is always deterministic and Markovian. The learned deterministic policy either guesses the correct label immediately, or guesses incorrectly and proceeds to make the same incorrect guess on every subsequent time-step. We compare performance in Figure 3.4.2 with a version of the agent that starts to guess randomly if incorrect on the first timestep, and a different agent that acts by process of elimination: first choosing the action it is most confident about, if incorrect, then the second, and so forth. Although all three versions have the same training performance, the learned RL policy generalizes

more poorly than these alternative variants that exploit the sequential nature of the problem. In Section 3.5.2, we will see that this process-of-elimination is, in some sense, the optimal way to generalize for this task. This experiment reveals a tension: learning policies for generalization that rely on an MDP model fail, even though the underlying environment *is* an MDP. This failure holds in any MDP model with limited data, whether the empirical MDP or more sophisticated MDPs that use uncertainty estimates in their construction.

3.5 Modeling Generalization in RL as an Epistemic POMDP

To better understand test-time generalization in RL, we study the problem under a Bayesian perspective. We show that training on limited training contexts leads to an implicit partial observability at test-time that we describe using a formalism called the epistemic POMDP.

3.5.1 The Epistemic POMDP

In the Bayesian framework, when learning given a limited amount of evidence $\mathcal{D}$ from an MDP $\mathcal{M}$, we can use a prior $\mathcal{P}(\mathcal{M})$ to construct a posterior belief distribution $\mathcal{P}(\mathcal{M}|\mathcal{D})$ over the identity of the MDP. For learning in a contextual MDP, $\mathcal{D}$ corresponds to the environment dynamics and reward in training contexts $\mathcal{C}_{\text{train}}$ that the agent can interact with, and the posterior belief distribution $\mathcal{P}(\mathcal{M}|\mathcal{D})$ models the agent's uncertainty about the behavior of the environment in contexts that it has not seen before (e.g. uncertainty about the label for a test-set image in the example from Section 3.4).

Since the agent only has partial access to the MDP $\mathcal{M}$ during training, the agent does not know which MDP from the posterior distribution is the true environment,

and must act at test-time under this uncertainty. Following a reduction common in Bayesian RL (Duff and Barto, 2002; Ghavamzadeh et al., 2015), we model this test-time uncertainty using a partially observed MDP that we will call the **epistemic POMDP**. The epistemic POMDP is structured as follows: each new episode in the POMDP begins by sampling a single MDP $\mathcal{M} \sim \mathcal{P}(\mathcal{M}|\mathcal{D})$ from the posterior, and then the agent interacts with $\mathcal{M}$ until the episode ends in this MDP. The agent does not observe *which* MDP was sampled, and since the MDP remains fixed for the duration of the episode, this induces implicit partial observability.

Effectively, each episode in the epistemic POMDP corresponds to acting in one of the possible environments that is consistent with the evidence that the agent is allowed access to at training time.

The epistemic POMDP is formally defined as the tuple $\mathcal{M}^{\text{po}} = (\mathcal{S}^{\text{po}}, \mathcal{O}^{\text{po}}, \mathcal{A}, T^{\text{po}}, r^{\text{po}}, \rho^{\text{po}}, \gamma)$. A state in this POMDP $s_t^{\text{po}} = (\mathcal{M}, s_t)$ contains the identity of the current MDP being acted in $\mathcal{M}$, and the current state in this MDP s_t; we write the state space as $\mathcal{S}^{\text{po}} = \mathbf{M} \times \mathcal{S}$, where $\mathbf{M}$ is the space of MDPs with support under the prior.

The agent only observes $o_t^{\text{po}} = s_t$, the state in the MDP ($\mathcal{O}^{\text{po}} = \mathcal{S}$), but **not** the identity of the MDP, $\mathcal{M}$. The initial state distribution is defined by the posterior distribution: $\rho^{\text{po}}((\mathcal{M}, s_0)) = \mathcal{P}(\mathcal{M}|\mathcal{D})\rho_{\mathcal{M}}(s_0)$, and the transition and reward functions in the POMDP reflect the dynamics in the current MDP:

$$T^{\text{po}}((\mathcal{M}', s') \mid (\mathcal{M}, s), a) = \delta(\mathcal{M}' = \mathcal{M})T_{\mathcal{M}}(s'|s, a) \qquad r^{\text{po}}((\mathcal{M}, s), a) = r_{\mathcal{M}}(s, a)\,. \quad (3.1)$$

Example (Sequential Image Classification)**.** We begin by explicitly describing the induced epistemic POMDP for the task from Section 3.4. The agent's uncertainty concerns how images are mapped to labels, and each MDP $\mathcal{M}$ in the posterior distribution corresponds to a different potential labelling function $Y_{\mathcal{M}} : x \mapsto y$ that is

consistent with the training dataset. Each episode in the epistemic POMDP, a different MDP $\mathcal{M}$ and corresponding labeller $Y_{\mathcal{M}}$ is sampled from the posterior distribution, alongside an image $x \sim p(x)$. The agent must guess the label assigned by this labelling function $y \coloneqq Y_{\mathcal{M}}(x)$, but is only provided the image x and **not** the identity of the labeller $Y_{\mathcal{M}}$. We emphasize that the context remains *fully observed* in the epistemic POMDP (the image x is provided to the agent); what is partially observed is how the environment dynamics will behave for the context (what label the image corresponds to).

What makes the epistemic POMDP a useful tool for understanding generalization in RL is that performance in the epistemic POMDP $\mathcal{M}^{\text{po}}$ corresponds exactly to the expected return of the agent at test-time when the prior is well-specified.

Proposition 3.5.1. *If the true MDP $\mathcal{M}$ is sampled from $\mathcal{P}(\mathcal{M})$, and evidence $\mathcal{D}$ from $\mathcal{M}$ is provided to an algorithm during training, then the expected test-time return of π is equal to its performance in the epistemic POMDP $\mathcal{M}^{po}$.*

$$J_{\mathcal{M}^{po}}(\pi) = \mathbb{E}_{\mathcal{M}\sim\mathcal{P}(\mathcal{M})}[J_{\mathcal{M}}(\pi) \mid \mathcal{D}]. \tag{3.2}$$

In particular, the optimal policy in $\mathcal{M}^{po}$ is Bayes-optimal for generalization to the unknown MDP $\mathcal{M}$: it receives the highest expected test-time return amongst all possible policies.

The epistemic POMDP is based on well-understood concepts in Bayesian reinforcement learning, and Bayesian modeling more generally. However, in contrast to prior works on Bayesian RL, we are specifically concerned with settings where there is a training-test split, and performance is measured by a single test episode. While using Bayesian RL to accelerate exploration or minimize regret has been well-explored (Ghavamzadeh et al., 2015), we rather use the Bayesian lens specifically to understand generalization – a perspective that is distinct from prior work on

Bayesian RL. Towards this goal, the equivalence between test-time return and expected return in the epistemic POMDP allows us to use performance in the POMDP as a proxy for understanding how well current RL methods can generalize.

3.5.2 Understanding Optimality in the Epistemic POMDP

We now study the structure of the epistemic POMDP, and use it to characterize properties of Bayes-optimal test-time behavior and the sub-optimality of alternative policy learning approaches. The majority of our results follow from well-known results about POMDPs, so we present them here informally, with formal statements and proofs in Appendix 3.B.

Example ctd. Acting optimally in the epistemic POMDP for the sequential image classification task requires maximizing return over the distribution of labels that is induced by the posterior distribution $p(y|x, \mathcal{D}) = \mathbb{E}_{\mathcal{M}\sim\mathcal{P}(\mathcal{M}|\mathcal{D})}[1(Y_{\mathcal{M}}(x) = y)]$. A deterministic policy (as is learned by standard RL algorithms) is a high-risk strategy in the POMDP; it receives exceedingly low return if the labeller outputs a different label than the one predicted. The Bayes-optimal generalization strategy corresponds to a process of elimination: first choose the most likely label $a = \arg\max p(y|x, \mathcal{D})$; if this is incorrect, eliminate it and choose the next-most likely, repeating until the correct label is finally chosen. Amongst memoryless policies, the optimal behavior is stochastic, sampling actions according to the distribution $\pi^*(a|x) \propto \sqrt{p(y|x, \mathcal{D})}$ (derivation in Appendix 3.A.2).

The characteristics of the optimal policy in the epistemic POMDP for the image classification RL problem match well-established results that optimal POMDP policies are generally memory-based (Monahan, 1982), and amongst memoryless policies, the optimal policy may be stochastic (Singh et al., 1994; Montúfar et al., 2015). Because of the equivalence between the epistemic POMDP and test-time behavior, these maxims are also true for Bayes-optimal behavior when maximizing test-time performance.

Remark 3.5.1. *The Bayes-optimal policy for maximizing test-time performance is in general non-Markovian. When restricted to Markovian policies, the Bayes-optimal policy is in general stochastic.*

The reason that Bayes-optimal generalization often requires memory is that the experience collected thus far in the episode contains information about the identity of the MDP being acted in (which is hidden from the agent observation), and to maximize expected return, the agent must adapt its subsequent behavior to incorporate this new information. The fact that acting optimally at test-time formally requires adaptivity (or stochasticity for memoryless policies) highlights the difficulty of generalizing well in RL, and provides a new perspective for understanding the success various empirical studies have found in improving generalization performance using recurrent networks (Akkaya et al., 2019; Peng et al., 2017) and stochastic regularization penalties (Stulp et al., 2011; Cobbe et al., 2019; Igl et al., 2019; Lu et al., 2020).

It is useful to understand to what degree the partial observability plays a role in determining Bayes-optimal behavior. When the partial observability is insignificant, the epistemic POMDP objective can coincide with a surrogate MDP approximation, and Bayes-optimal solutions can be attained with standard fully-observed RL algorithms. For example, if there is a policy that is simultaneously optimal in *every* MDP from the posterior, then an agent need not worry about the (hidden) identity of the MDP, and just follow this policy. Perhaps surprisingly, this kind of condition is difficult to relax: we show in Proposition 3.B.1 that even if a policy is optimal in many (but not all) of the MDPs from the posterior, this seemingly "optimal" policy can generalize poorly at test-time.

Moreover, under partial observability, optimal policies for the MDPs in the posterior may differ substantially from Bayes-optimal behavior: in Proposition 3.B.2, we show that the Bayes-optimal policy may take actions that are sub-optimal in *every* environment in the posterior. These results indicate the brittleness of learning policies

based on optimizing return in an MDP model when the agent has not yet fully resolved the uncertainty about the true MDP parameters.

Remark 3.5.2 (Failure of MDP-Optimal Policies, Propositions 3.B.1, 3.B.2)**.** *The expected test-time return of policies that are learned by maximizing reward in any MDP from the posterior, as standard RL methods do, may be arbitrarily low compared to that of Bayes-optimal behavior.*

As Bayes-optimal memoryless policies are stochastic, one may wonder if simple strategies for inducing stochasticity, such as adding ε-greedy noise or entropy regularization, can alleviate the sub-optimality that arose with deterministic policies in the previous paragraph. In some cases, this may be true; one particularly interesting result is that in certain goal-reaching problems, entropy-regularized RL can be interpreted as optimizing an epistemic POMDP objective with a specific form of posterior distribution over reward functions (Proposition 3.B.3) (Eysenbach and Levine, 2019). For the more general setting, we show in Proposition 3.B.4 that entropy regularization and other general-purpose techniques can similarly catastrophically fail in epistemic POMDPs.

Remark 3.5.3 (Failure of Generic Stochasticity, Proposition 3.B.4)**.** *The expected test-time return of policies learned with stochastic regularization techniques like maximum-entropy RL that are agnostic of the posterior* $\mathcal{P}(\mathcal{M}|\mathcal{D})$ *may be arbitrarily low compared to that of Bayes-optimal behavior.*

This failure happens because the degree of stochasticity used by the Bayes-optimal policy reflects the agent's epistemic uncertainty about the environment; since standard regularizations are agnostic to this uncertainty, the learned behaviors often do not reflect the appropriate level of stochasticity needed. A maze-solving agent acting Bayes-optimally, for example, may choose to act deterministically in mazes like those it has seen at training, and on others where it is less confident, rely on random

exploration to exit the maze, inimitable behavior by regularization techniques agnostic to this uncertainty.

Our analysis of the epistemic POMDP highlights the difficulty of generalizing well in RL, in the complexity of Bayes-optimal policies (Remark 3.5.1) and the deficiencies of our standard MDP-based RL algorithms (Remark 3.5.2 and Remark 3.5.3). While MDP-based algorithms can serve as a useful starting point for acquiring generalizable skills, learning policies that perform well in new test-time scenarios may require more complex algorithms that attend to the epistemic POMDP structure that is implicitly induced by the agent's epistemic uncertainty.

3.6 Learning Policies that Generalize Well Using the Epistemic POMDP

When the epistemic POMDP $\mathcal{M}^{\text{po}}$ can be exactly obtained, we can learn RL policies that generalize well to the true (unknown) MDP $\mathcal{M}$ by learning an optimal policy in the POMDP. In this oracle setting, any POMDP-solving method will suffice, and design choices like policy function classes (e.g. recurrent vs Markovian policies) or agent representations (e.g. belief state vs PSRs) made based on the requirements of the specific domain. However, in practice, the epistemic POMDP can be challenging to approximate due to the difficulties of learning coherent MDP models and maintaining a posterior over such MDP models in high-dimensional domains.

In light of these challenges, we now focus on practical methods for learning generalizable policies when the exact posterior distribution (and therefore true epistemic POMDP) cannot be recovered exactly. We derive an algorithm for learning the optimal policy in the epistemic POMDP induced by an approximate posterior distribution $\hat{\mathcal{P}}(\mathcal{M}|\mathcal{D})$ with finite support. We use this to motivate LEEP, a simple ensemble-based algorithm for learning policies in the contextual MDP setting.

3.6.1 Policy Optimization in an Empirical Epistemic POMDP

Towards a tractable algorithm, we assume that instead of the true posterior $\mathcal{P}(\mathcal{M}|\mathcal{D})$, we only have access to an empirical posterior distribution $\hat{\mathcal{P}}(\mathcal{M}|\mathcal{D})$ defined by n MDP samples from the posterior distribution $\{\mathcal{M}_i\}_{i\in[n]}$. This empirical posterior distribution induces an empirical epistemic POMDP $\hat{\mathcal{M}}^{\text{po}}$; our ambition is to learn the optimal policy in this POMDP. Rather than directly learning this optimal policy as a generic POMDP solver might, we recognize that $\hat{\mathcal{M}}^{\text{po}}$ corresponds to a collection of n MDPs [1] and decompose the optimization problem to mimic this structure. We will learn n policies $\pi_1, \cdots, \pi_n$, each policy π_i in one of the MDPs $\mathcal{M}_i$ from the empirical posterior, and combine these policies together to recover a single policy π for the POMDP. Reducing the POMDP policy learning problem into a set of MDP policy learning problems can allow us to leverage the many recent advances in deep RL for scalably solving MDPs. The following theorem links the expected return of a policy π in the empirical epistemic POMDP $\hat{\mathcal{M}}^{\text{po}}$, in terms of the performance of the policies π_i on their respective MDPs $\mathcal{M}_i$.

Proposition 3.6.1. *Let $\pi, \pi_1, \cdots \pi_n$ be memoryless , and define $r_{\max} = \max_{i,s,a} |r_{\mathcal{M}_i}(s,a)|$. The expected return of π in $\hat{\mathcal{M}}^{po}$ is bounded below as:*

$$J_{\hat{\mathcal{M}}^{po}}(\pi) \geqslant \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i) - \frac{\sqrt{2}r_{\max}}{(1-\gamma)^2 n}\sum_{i=1}^{n}\mathbb{E}_{s\sim d^{\pi_i}_{\mathcal{M}_i}}\left[\sqrt{D_{KL}\left(\pi_i(\cdot|s) \;||\; \pi(\cdot|s)\right)}\right], \qquad (3.3)$$

This proposition indicates that if the policies in the collection $\{\pi_i\}_{i\in[n]}$ all achieve high return in their respective MDPs (first term) and are imitable by a single policy π (second term), then π is guaranteed to achieve high return in the epistemic POMDP. In contrast, if the policies cannot be closely imitated by a single policy, this collection

[1] Note that when the true environment is a contextual MDP, the sampled MDP $\mathcal{M}_i$ does not correspond to a single context within a contextual MDP — each MDP $\mathcal{M}_i$ is an *entire* contextual MDP with many contexts.

of policies may not be useful for learning in the epistemic POMDP using the lower bound. This means that it may not sufficient to naively optimize each policy π_i on its MDP $\mathcal{M}_i$ without any consideration to the other policies or MDPs, since the learned policies are likely to be different and difficult to jointly imitate. To be useful for the lower bound, each policy π_i should balance between maximizing performance on its MDP and minimizing its deviation from the other policies in the set. The following proposition shows that if the policies are trained jointly to ensure this balance, it in fact recovers the optimal policy in the empirical epistemic POMDP.

Proposition 3.6.2. *Let $f : \{\pi_i\}_{i\in[n]} \mapsto \pi$ be a function that maps n policies to a single policy satisfying $f(\pi, \cdots, \pi) = \pi$ for every policy π, and let α be a hyperparameter satisfying $\alpha \geqslant \frac{\sqrt{2}r_{max}}{(1-\gamma)^2 n}$. Then letting $\pi_1^*, \ldots \pi_n^*$ be the optimal solution to the following optimization problem:*

$$\{\pi_i^*\}_{i\in[n]} = \arg\max_{\pi_1,\cdots,\pi_n} \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i) - \alpha \sum_{i=1}^{n} \mathbb{E}_{s\sim d^{\pi_i}_{\mathcal{M}_i}} \left[\sqrt{D_{KL}\left(\pi_i(\cdot|s) \;||\; f(\{\pi_i\})(\cdot|s)\right)}\right], \tag{3.4}$$

the policy $\pi^* \coloneqq f(\{\pi_i^*\}_{i\in[n]})$ is optimal for the empirical epistemic POMDP $\hat{\mathcal{M}}^{po}$.

3.6.2 A Practical Algorithm for Contextual MDPs: LEEP

Proposition 3.6.2 provides a foundation for a practical algorithm for learning policies when provided training contexts $\mathcal{C}_{\text{train}}$ from an unknown contextual MDP. In order to use the proposition in a practical algorithm, we must discuss two problems: how posterior samples $\mathcal{M}_i \sim \mathcal{P}(\mathcal{M}|\mathcal{D})$ can be approximated, and how the function f that combines policies should be chosen.

Approximating the posterior distribution: Rather than directly maintaining a posterior over transition dynamics and reward models, which is especially difficult with image-based observations, we can approximate samples from the posterior via a bootstrap sampling technique (Osband et al., 2016). To sample a candidate MDP

$\mathcal{M}_i$, we sample with replacement from the training contexts $\mathcal{C}_{\text{train}}$ to get a new set of contexts $\mathcal{C}^i_{\text{train}}$, and define $\mathcal{M}_i$ to be the empirical MDP on this subset of training contexts. Rolling out trials from the posterior sample $\mathcal{M}_i$ then corresponds to selecting a context at random from $\mathcal{C}^i_{\text{train}}$, and then rolling out that context. Crucially, note that $\mathcal{M}_i$ still corresponds to a *distribution* over contexts, not a single context, since our goal is to sample from the posterior entire contextual MDPs.

Choosing a link function: The link function f in Proposition 3.6.2 that combines the set of policies together effectively serves as an inductive bias: since we are optimizing in an approximation to the true epistemic POMDP and policy optimization is not exact in practice, different choices can yield combined policies with different characteristics. Since optimal behavior in the epistemic POMDP must consider all actions, even those that are potentially sub-optimal in all MDPs in the posterior (as discussed in Section 3.5.2), we use an "optimistic" link function that does not dismiss any action that is considered by at least one of the policies, specifically $f(\{\pi_i\}_{i\in[n]}) = (\max_i \pi_i)(a|s) \coloneqq \frac{\max \pi_i(a|s)}{\sum_{a'} \max \pi_i(a'|s)}$.

Algorithm: We learn a set of n policies $\{\pi_i\}_{i\in[n]}$, using a policy gradient algorithm to implement the update step. To update the parameters for π_i, we take gradient steps via the surrogate loss used for the policy gradient, augmented by a disagreement

Algorithm 3.1 Linked Ensembles for the Epistemic POMDP (LEEP)

1: Receive training contexts $\mathcal{C}_{\text{train}}$, number of ensemble members n
2: Bootstrap sample training contexts to create $\mathcal{C}^1_{\text{train}}, \ldots \mathcal{C}^n_{\text{train}}$, where $\mathcal{C}^i_{\text{train}} \subset \mathcal{C}_{\text{train}}$.
3: Initialize n policies: $\pi_1, \ldots \pi_n$
4: **for** iteration $k = 1, 2, 3, \ldots$ **do**
5: **for** policy $i = 1, \ldots, n$ **do**
6: Collect environment samples in training contexts $\mathcal{C}^i_{\text{train}}$ using policy π_i
7: Take gradient steps wrt π_i on these samples with augmented RL loss:

$$\pi_i \leftarrow \pi_i - \eta \nabla_i (\mathcal{L}^{RL}(\pi_i) + \alpha \mathbb{E}_{s\sim\pi_i, \mathcal{C}^i_{\text{train}}} [D_{KL}(\pi_i(a|s) \| \max_j \pi_j(a|s))])$$

8: Return $\pi = \max_i \pi$: $\pi(a|s) = \frac{\max_i \pi_i(a|s)}{\sum_{a'} \max_i \pi_i(a'|s)}$.

penalty between the policy and the combined policy $f(\{\pi_i\}_{i\in[n]})$ with a penalty parameter $\alpha > 0$, as in Equation 3.5:

$$\mathcal{L}(\pi_i) = \mathcal{L}^{RL}(\pi_i) + \alpha \mathbb{E}_{s\sim\pi_i,\mathcal{M}_i}[D_{KL}(\pi_i(a|s)\| \max_j \pi_j(a|s))]. \tag{3.5}$$

Combining these elements together leads to our method, LEEP, which we summarize in Algorithm 3.1. In our implementation, we use PPO for $\mathcal{L}^{RL}(\pi_i)$ (Schulman et al., 2017). In summary, LEEP bootstrap samples the training contexts to create overlapping sets of training contexts $\mathcal{C}^1_{\text{train}}, \ldots \mathcal{C}^n_{\text{train}}$. Every iteration, each policy π_i generates rollouts in training contexts chosen uniformly from its corresponding $\mathcal{C}^i_{\text{train}}$, and is then updated according to Equation 3.5, which both maximizes the expected reward and minimizes the disagreement penalty between each π_i and the combined policy $\pi = \max_j \pi_j$.

While this algorithm is structurally similar to algorithms for multi-task learning that train a separate policy for each context or group of contexts with a disagreement penalty (Teh et al., 2017; Ghosh et al., 2018), the motivation and the interpretation of these approaches are completely distinct. In multi-task learning, the goal is to solve a given set of tasks, and these methods promote transfer via a loss that encourages the solutions to the tasks to be in agreement. In our setting, while we also receive a set of tasks (contexts), the goal is not to maximize performance on the training tasks, but rather to learn a policy that maximizes performance on unseen test tasks. The method also has a subtle but important distinction: each of our policies π_i acts on a sample from the contextual MDP posterior (which captures epistemic uncertainty), *not* a single training context (Teh et al., 2017) or element from a disjoint partitioning (Ghosh et al., 2018) (which does not). This distinction is crucial, since our generalization performance requires our aim is not to make it easier to solve the training contexts, but the opposite: prevent the algorithm from overfitting to the individual training

contexts. Correspondingly, our experiments confirm that such multi-task learning approaches do not provide the same generalization benefits as our approach.

3.7 Experiments

The primary ambition of our empirical study is to test the hypothesis that policies that are learned through (approximations of) the epistemic POMDP do in fact attain better test-time performance than those learned by standard RL algorithms. We do so on the Procgen benchmark (Cobbe et al., 2020), a challenging suite of diverse tasks with image-based observations testing generalization to unseen contexts.

1. Does LEEP derived from the epistemic POMDP lead to improved test-time performance over standard RL methods?
2. Can LEEP prevent overfitting when provided a limited number of training contexts?
3. How do different algorithmic components of LEEP affect test-time performance?

The Procgen benchmark is a set of procedurally generated games, each with different generalization challenges. In each game, during training, the algorithm can interact with 200 training levels, before it is asked to generalize to the full distribution of levels. The agent receives a $64 \times 64 \times 3$ image observation, and must output one of 15 possible actions. We instantiate our method using an ensemble of $n = 4$ policies, a penalty parameter of $\alpha = 1$, and PPO (Schulman et al., 2017) to train the individual policies (full implementation details in Appendix 3.C).

We evaluate our method on four games in which prior work has found a large gap between training and test performance, and which we therefore conclude pose a significant generalization challenge (Cobbe et al., 2020; Jiang et al., 2020; Raileanu et al., 2020): Maze, Heist, BigFish, and Dodgeball. In Figure 3.7.1, we compare the test-time performance of the policies learned using our method to those learned

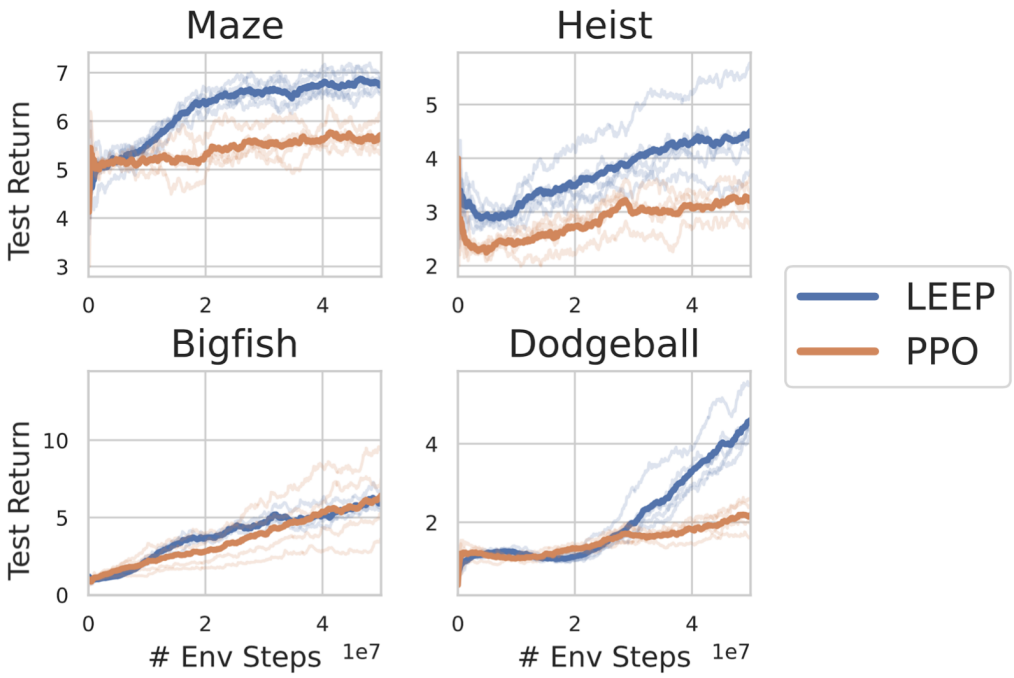
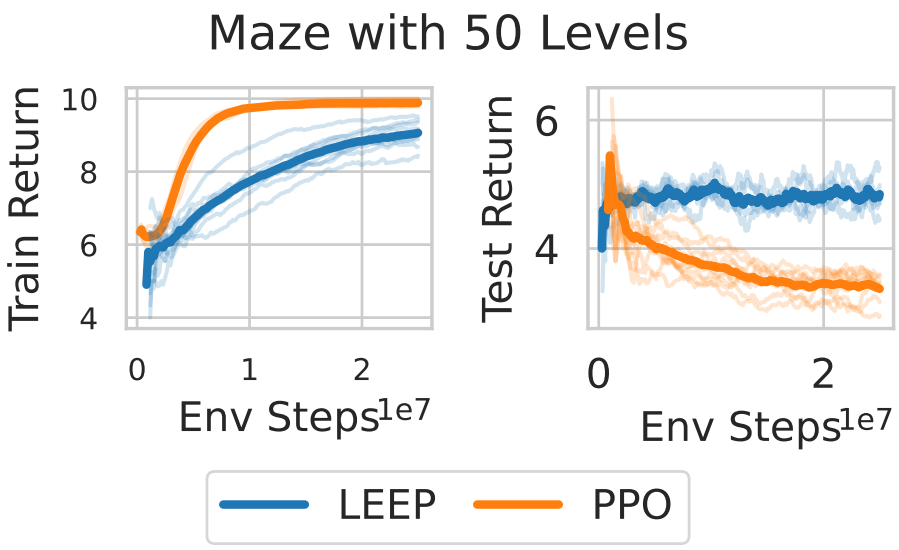


Figure 3.7.1: Test set return for LEEP and PPO throughout training in four Procgen environments (averaged across 5 random seeds). LEEP achieves higher test returns than PPO on three tasks (Maze, Heist and Dodgeball) and matches test return on Bigfish while having less variance across seeds.

by a PPO agent with entropy regularization. In three of these environments (Maze, Heist, and Dodgeball), our method outperforms PPO by a significant margin, and in all cases, we find that the generalization gap between training and test performance is lower for our method than PPO (Appendix 3.D.1). To understand how LEEP behaves with fewer training contexts, we ran on the Maze task with only 50 levels (Figure 3.7.2 (top)); the test return of the PPO policy decreases through training, leading to final performance worse than the starting random policy, but our method avoids this degradation.

We perform an ablation study on the Maze and Heist environments (Maze in Figure 4, Heist in Appendix 3.D.1) to rule out potential confounding causes for the improved generalization that our method displays on the Procgen benchmark tasks. First, to see if the performance benefit derives solely from the use of ensembles, we compare LEEP to a Bayesian model averaging strategy that trains an ensemble of policies without regularization ("Ensemble (no reg)"), and uses a mixture of these policies. This



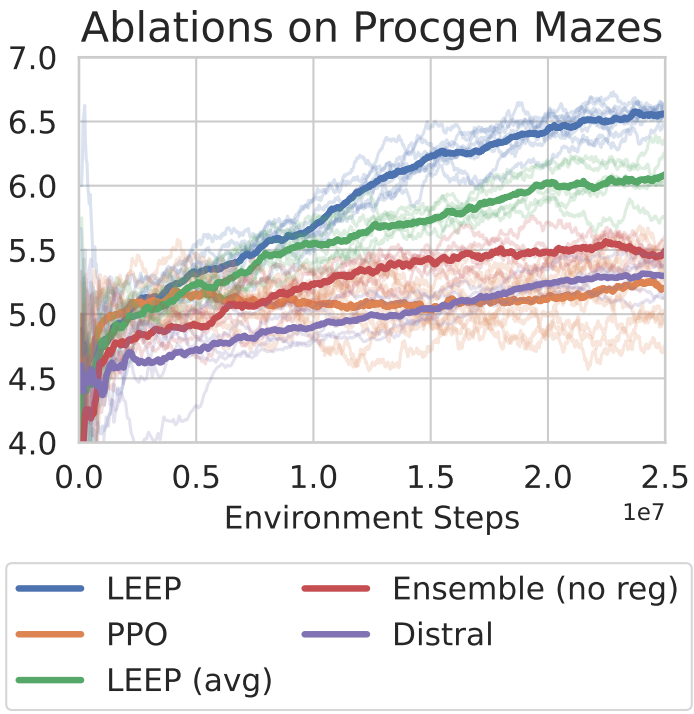


Figure 3.7.2: **(top)** Performance of LEEP and PPO with only 50 training levels on Maze. **(bottom)** Ablations of LEEP in Maze.

strategy does improve performance over the PPO policy, but does not match LEEP, indicating the usefulness of the regularization. Second, we compared to a version of LEEP that combines the ensemble policies together using the average $\frac{1}{n}\sum_{i=1}^{n}\pi_i(a|s)$ ("LEEP (avg)"). This link function achieves worse test-time performance than the optimistic version, which indicates that the inductive bias conferred by the $\max_i \pi_i$ link function is a useful component of the algorithm. We also compare to Distral, a multi-task learning method with different motivations but similar structure to LEEP: this method helps accelerate learning on the provided training contexts (figures in Appendix 3.D.1), but does not improve generalization performance as LEEP does. We additionally ablated the two key hyperparameters in LEEP, the number of ensemble members n and the penalty coefficient α (Table in Appendix 3.D.2).

3.8 Discussion

It has often been observed experimentally that generalization in RL poses a significant challenge, but it has so far remained an open question as to whether the RL setting itself presents additional generalization challenges beyond those seen in supervised learning. In this chapter, we answer this question in the affirmative, and show that, in contrast to supervised learning, generalization in RL results in a new type of problem that cannot be solved with standard MDP solution methods, due to partial observability induced by epistemic uncertainty. We call the resulting partially observed setting the epistemic POMDP, where uncertainty about the true underlying MDP results in a challenging partially observed problem. We present a practical approximate method that optimizes a bound for performance in an approximation of the epistemic POMDP, and show empirically that this approach, which we call LEEP, attains significant improvements in generalization over other RL methods that do not properly incorporate the agent's epistemic uncertainty into policy optimization. A limitation

of this approach is that it optimizes a crude approximation to the epistemic POMDP with a small number of posterior samples, and may be challenging to scale to better approximations to the true objective. Developing algorithms that better model the epistemic POMDP and optimize policies within is an exciting avenue for future work, and we hope that this direction will lead to further improvements in generalization in RL.

Appendix

3.A FashionMNIST Classification

3.A.1 Implementation details

Environment: The RL image classification environment consists of a dataset of labelled images. At the beginning of each episode, a new image and its corresponding label are chosen from the dataset, and held fixed for the entire episode. Each time-step, the agent must pick an action corresponding to one of the labels. If the picked label is correct, the agent gets a reward of $r = 0$, and the episode ends, and if the picked label is incorrect, then the agent gets a reward of $r = -1$, and the episode continues to the next time-step (where it must guess another label for the *same* image). The total return for a trajectory corresponds to the number of incorrect guesses the agent makes for the image. We enforce a time-limit of 20 timesteps in the environment to prevent infinite-length trajectories of incorrect guessing.

We train the agent on a dataset of 10000 FashionMNIST images subsampled from the training set, and test on the FashionMNIST test dataset. Note that this task is very similar, but not exactly equivalent to maximizing predictive accuracy for supervised classification: if the episode ended regardless of whether or not the agent was correct, then it would correspond exactly to classification.

Algorithm: We train a DQN agent on the training environment using the min-Q update rule from TD3 (Fujimoto et al., 2018). The Q-function architecture is a

convolutional neural network (CNN) with the architecture from Kostrikov et al (Kostrikov, 2018). To ensure that the agent does not suffer from poor exploration during training, the replay buffer is pre-populated with one copy of every possible transition in the training environment (that is, where every action is taken for every image in the training dataset). The variant labelled "Uniform after step 1" in Figure 3.4.2 follows the DQN policy for the first time-step, and if this was incorrect, then at all subsequent time-steps, takes a random action uniformly amongst the 10 labels. For the variant labelled "Adaptive", we train a classifier $p_\theta(y|x)$ on the training dataset of images with the same architecture as the DQN agent. The adaptive agent follows a process-of-elimination strategy; formally, the action taken by the adaptive agent at time-step t is given by $\arg\max_{a \notin \{a_1,\ldots,a_{t-1}\}} p_\theta(y = a|x)$.

3.A.2 Derivation of Bayes-optimal policies

In the epistemic POMDP for the RL image classification problem, each episode, an image $x \in \mathcal{X}$ is sampled randomly from the dataset, and a label $y \in \mathcal{Y}$ sampled randomly for this image from the distribution $p(y|x, \mathcal{D})$. This label is *held fixed* for the entire episode. For notation, let $Y = \{1, \ldots, d\}$, so that a label distribution $p(y|x)$ can be written as a vector in the probability simplex on $\mathbb{R}^d$. We emphasize two settings: $\gamma = 0$ (the supervised learning setting), and $\gamma = 1$ (an RL setting), where the expected return of an agent is the average number of incorrect guesses made.

Memory-based policy

Since the optimal memory-based policy in a POMDP is deterministic (Monahan, 1982), we restrict ourselves to analyzing the performance of deterministic memory-based policies. In the following we will narrow the search space even further.

Since the episode ends after the agent correctly classifies an image and the reward structure incentives the agent to solve the task as quickly as possible, an agent acting

optimally will never repeat the same action twice. Indeed, the agent will not have the opportunity to repeat the right action twice because the episode would have ended after the first time it tried it. Furthermore, trying a wrong action twice is also not optimal as in incurs addition negative reward. Therefore, we can limit our search space to policies that try each action once. These policies differ by the ordering in which they try each one of these d labels.

At the beginning of every episode, a image x is sampled uniformly at random among all training images and its true label y (during that episode) is sampled from $p(y|x, \mathcal{D})$. Let π be policy that tries each of the d actions exactly once in its first d trials. Let T_y^π denotes the time when policy π tries action y. Note that $(T_y^\pi)_{y\in\mathcal{Y}}$ is a permutation. When the label chosen is y, the cumulative reward of π for that episode is given by $r = \frac{\gamma^{T_y^\pi}-1}{1-\gamma}$ and the expected cumulative reward (across episodes) is given by:

$$\begin{aligned} J(\pi) :=& \sum_{y\in\mathcal{Y}} p(y|x,\mathcal{D}) \frac{\gamma^{T_y^\pi}-1}{1-\gamma} \\ =& \frac{1}{1-\gamma}\left(\sum_{y\in\mathcal{Y}} p(y|x,\mathcal{D})\gamma^{T_y^\pi} - 1\right) \end{aligned} \tag{3.6}$$

From that expression, we see that in order to maximize its expected cumulative reward, a policy π has to maximize $\sum_{y\in\mathcal{Y}} p(y|x,\mathcal{D})\gamma^{T_y^\pi}$ which can be interpreted as the dot product of the vector $[p(y|x,\mathcal{D})]_{y\in\mathcal{Y}}$ and $[\gamma^{T_y^\pi}]_{y\in\mathcal{Y}}$. By the rearrangement inequality, we know that this dot product is maximized when the components of the vectors are arranged in the same ordering.

If we denote by $y_{(1)}, \ldots y_{(d)}$ be the labels sorted in order of probability under the belief distribution: $p(y_{(1)}|x,\mathcal{D}) \geqslant p(y_{(2)}|x,\mathcal{D}) \geqslant \ldots \geqslant p(y_{(d)}|x,\mathcal{D})$. Since $0 < \gamma < 1$ the rearrangement inequality implies that the expected return is maximized when $T_{y_{(t)}}^\pi = t$. This corresponds to a policy that tries the labels sequentially from the most likely to the least likely.

Memoryless policy policy

Consider a memoryless policy that takes actions according to the distribution $\pi(\cdot|x)$ for the image x. When the true label is y for the image x, the number of incorrect guesses is distributed as $\text{Geom}(p = \pi(y|x))$.

When the agent guesses correctly the label y at the $t-$th guess then the cumulative reward is given by $r = -\frac{1-\gamma^t}{1-\gamma}$. This happens with probability $(1 - \pi(y \mid x))^t \times \pi(y \mid x)$. The expected return for policy π evaluated on image x is then given by:

$$\begin{aligned} J(\pi|x) &= -\sum_{y\in\mathcal{Y}} \sum_{t=0}^{\infty} (1-\pi(y|x))^t \pi(y|x) \frac{1-\gamma^t}{1-\gamma} p(y|x,\mathcal{D}) \\ &= \sum_{y\in\mathcal{Y}} p(y|x,\mathcal{D}) \frac{\pi(y \mid x) - 1}{1-\gamma(1-\pi(y|x))} \end{aligned} \tag{3.7}$$

When $\gamma = 0$ (supervised learning problem), $J(\pi) = \sum_{y\in\mathcal{Y}} p(y|x,\mathcal{D})\,(\pi(y \mid x) - 1)$ is a linear function of π and as expected, the optimal policy is to deterministically choose the label with the highest probability: $\pi^*(y \mid x) = 1\left[y = \arg\max_{y\in\mathcal{Y}} p(y|x,\mathcal{D})\right]$.

When $\gamma > 0$, the optimal policy is the solution to a constrained optimization problem that can be solved with Lagrange multipliers. When $\gamma = 1$, the optimal policy can be written explicitly as:

$$\pi^*(y \mid x) = \frac{1}{\lambda}\sqrt{p(y|x,\mathcal{D})} \tag{3.8}$$

where λ is a normalization constant.

3.B Theoretical Results

Proposition 3.5.1. *If the true MDP $\mathcal{M}$ is sampled from $\mathcal{P}(\mathcal{M})$, and evidence $\mathcal{D}$ from $\mathcal{M}$ is provided to an algorithm during training, then the expected test-time return of π is equal to its performance in the epistemic POMDP $\mathcal{M}^{po}$.*

$$J_{\mathcal{M}^{po}}(\pi) = \mathbb{E}_{\mathcal{M}\sim\mathcal{P}(\mathcal{M})}[J_{\mathcal{M}}(\pi) \mid \mathcal{D}]. \tag{3.2}$$

In particular, the optimal policy in $\mathcal{M}^{po}$ is Bayes-optimal for generalization to the unknown MDP $\mathcal{M}$: it receives the highest expected test-time return amongst all possible policies.

Proof. This proposition follows directly from the definition of the epistemic POMDP. If the MDP $\mathcal{M}$ is sampled from $\mathcal{P}(\mathcal{M})$ and $\mathcal{D}$ is witnessed, then the posterior distribution over MDPs is given by $\mathcal{P}(\mathcal{M}|\mathcal{D})$, and the expected test-time return of π given the evidence is

$$\mathbb{E}_{\mathcal{M}\sim\mathcal{P}(\mathcal{M})}[J_{\mathcal{M}}(\pi)|\mathcal{D}] \coloneqq \mathbb{E}_{\mathcal{M}\sim\mathcal{P}(\mathcal{M}|\mathcal{D})}[J_{\mathcal{M}}(\pi)].$$

In the epistemic POMDP, where an episode corresponds to randomly sampling an MDP from $\mathcal{P}(\mathcal{M}|\mathcal{D})$, and a single episode being evaluated in this MDP, the expected return can be expressed identically:

$$J_{\mathcal{M}^{\mathrm{Po}}}(\pi) \coloneqq \mathbb{E}_{\mathcal{M}\sim\mathcal{P}(\mathcal{M}|\mathcal{D})}[\mathbb{E}_{\pi,\mathcal{M}}[\sum_{i=0}^{\infty}\gamma^t r(s_t,a_t)]] \tag{3.9}$$

$$= \mathbb{E}_{\mathcal{M}\sim\mathcal{P}(\mathcal{M}|\mathcal{D})}[J_{\mathcal{M}}(\pi)]. \tag{3.10}$$

□

3.B.1 Optimal MDP Policies can be Arbitrarily Suboptimal

Proposition 3.B.1. *Let $\varepsilon > 0$. There exists posterior distributions $\mathcal{P}(\mathcal{M}|\mathcal{D})$ where a deterministic Markov policy π is optimal with probability at least $1 - \varepsilon$,*

$$P_{\mathcal{M}\sim\mathcal{P}(\mathcal{M}|\mathcal{D})}\left(\pi \in \arg\max_{\pi'} J_{\mathcal{M}}(\pi')\right) \geqslant 1 - \varepsilon, \tag{3.11}$$

but is outperformed by a uniformly random policy in the epistemic POMDP: $J_{\mathcal{M}^{po}}(\pi) < J_{\mathcal{M}^{po}}(\pi_{unif})$.

Proof. Consider two deterministic MDPs, $\mathcal{M}_A$, and $\mathcal{M}_B$ that both have two states and two actions: "stay" and "switch". In both MDPs, the reward for the "stay" action is always zero. In $\mathcal{M}_A$ the reward for "switch" is always 1, while in $\mathcal{M}_B$ the reward for "switch" is $-c$ for $c > 0$. The probability of being in $\mathcal{M}_B$ is ε while the probability of being in $\mathcal{M}_A$ is $1 - \varepsilon$. Clearly, the policy "always switch" is optimal in $\mathcal{M}_A$ and so is ε-optimal under the distribution on MDPs. The expected discounted reward of the "always switch" policy is:

$$\begin{aligned} J(\pi_{\text{always switch}}) &= (1-\varepsilon)\frac{1}{1-\gamma} - \varepsilon\frac{c}{1-\gamma} && (3.12)\\ &= \frac{1}{1-\gamma}(1 - \varepsilon - c\varepsilon) && (3.13)\\ &= \frac{1}{1-\gamma}(1 - (c+1)\varepsilon)\,. && (3.14) \end{aligned}$$

On the other hand, we can consider a policy which selects actions uniformly at random. In this case, the expected cumulative reward is

$$J(\pi_{\text{random}}) = (1-\varepsilon)\frac{1}{2}\frac{1}{1-\gamma} - \varepsilon\frac{c}{2}\frac{1}{1-\gamma} \tag{3.15}$$

$$= \frac{1}{2}\frac{1}{1-\gamma}(1-\varepsilon-c\varepsilon) \tag{3.16}$$

$$= \frac{1}{2}\frac{1}{1-\gamma}(1-(c+1)\varepsilon) \tag{3.17}$$

$$= \frac{1}{2}J(\pi_{\text{always switch}})\,. \tag{3.18}$$

Thus for any ε we can find a $c > \frac{1}{\varepsilon} - 1$ such that both policies have negative expected rewards and we prefer the random policy for being half as negative. □

3.B.2 Bayes-optimal Policies May Take Suboptimal Actions Everywhere

We formalize the remark that optimal policies for the MDPs in the posterior distribution may be poor guides for determining what the Bayes-optimal behavior is in the epistemic POMDP. The following proposition shows that there are epistemic POMDPs where the support of actions taken by the MDP-optimal policies is disjoint from the actions taken by the Bayes-optimal policy, so no method can "combine" the optimal policies from each MDP in the posterior to create Bayes-optimal behavior.

Proposition 3.B.2. *There exist posterior distributions* $\mathcal{P}(\mathcal{M}|\mathcal{D})$ *where the support of the Bayes-optimal memoryless policy* $\pi^{*po}(a|s)$ *is disjoint with that of the optimal policies in each MDP in the posterior. Formally, writing* $\text{supp}(\pi(a|s)) = \{a \in \mathcal{A} : \pi(a|s) > 0\}$*, then* $\forall \mathcal{M}$ *with* $\mathcal{P}(\mathcal{M}|\mathcal{D}) > 0$ *and* $\forall s$*:*

$$\text{supp}(\pi^{*po}(a|s)) \cap \text{supp}(\pi^*_{\mathcal{M}}(a|s)) = \emptyset$$

Proof. The proof is a simple modification of the construction in Proposition 3.B.1. Consider two deterministic MDPs, $\mathcal{M}_A$, and $\mathcal{M}_B$ with equal support under the posterior, where both have two states and three actions: "stay", "switch 1", and "switch 2". In both MDPs, the reward for the "stay" action is always zero. In $\mathcal{M}_A$ the reward for "switch" is always 1, while in $\mathcal{M}_B$ the reward for "switch" is -2. The reward structure for "switch 2" is flipped: in $\mathcal{M}_A$, the reward for "switch 2" is -2, and in $\mathcal{M}_B$, the reward is 1. Then, the policy "always switch" is optimal in $\mathcal{M}_A$, and the policy "always switch 2" is optimal in $\mathcal{M}_B$. However, any memoryless policy that takes either of these actions receives negative reward in the epistemic POMDP, and is dominated by the Bayes-optimal memoryless policy "always stay", which achieves 0 reward. □

3.B.3 MaxEnt RL is Optimal for a Choice of Prior

We describe a special case of the construction of Eysenbach and Levine (Eysenbach and Levine, 2019), which shows that maximum-entropy RL in a bandit problem recovers the Bayes-optimal POMDP policy in an epistemic POMDP similar to that described in the RL image classification task.

Consider the family of MDPs $\{\mathcal{M}_k\}_{k\in[n]}$ each with one state and n actions, where taking action k in MDP $\mathcal{M}_k$ yields zero reward and the episode ends, and taking any other action yields reward -1 and the episode continues. Effectively, $\mathcal{M}_k$ corresponds to a first-exit problem with "goal action" k. Note that this MDP structure is exactly what we have for the RL image classification task for a single image. Also consider the surrogate bandit MDP $\hat{\mathcal{M}}$, also with one state and n actions, but in which taking action k yields reward r_k with immediate episode termination. The following proposition shows that running max-ent RL in $\hat{\mathcal{M}}$ recovers the optimal memoryless policy in a particular epistemic POMDP supported on $\{\mathcal{M}_k\}_{k\in[n]}$.

Proposition 3.B.3. *Let* $\pi^* = \arg\max_{\pi \in \Pi} J_{\hat{\mathcal{M}}}(\pi) + \mathcal{H}(\pi)$ *be the max-ent solution in the surrogate bandit MDP* $\hat{\mathcal{M}}$*. Define the distribution* $\mathcal{P}(\mathcal{M}|\mathcal{D})$ *on* $\{\mathcal{M}_k\}_{k \in [n]}$ *as* $\mathcal{P}(\mathcal{M}_k|\mathcal{D}) = \frac{\exp(2r_k)}{\sum_j \exp(2r_j)}$*. Then,* π *is the optimal memoryless policy in the epistemic POMDP* $\mathcal{M}^{po}$ *defined by* $\mathcal{P}(\mathcal{M}|\mathcal{D})$*.*

Proof. See Eysenbach and Levine (Eysenbach and Levine, 2019, Lemma 4.1). The optimal policy π^* is given by $\pi^*(a = k) = \frac{\exp(r_k)}{\sum_j \exp(r_j)}$. We know from Appendix 3.A.2 that this policy is optimal for epistemic POMDP $\mathcal{M}^{\text{po}}$ when $\gamma = 1$. □

If allowing time-varying reward functions, this construction can be extended beyond "goal-action taking" epistemic POMDPs to the more general "goal-state reaching" setting in an MDP, where the agent seeks to reach a specific goal state, but the identity of the goal state hidden from the agent (Eysenbach and Levine, 2019, Lemma 4.2).

3.B.4 Failure of MaxEnt RL and Uncertainty-Agnostic Regularizations

We formalize the remark made in the main text that while the Bayes-optimal memoryless policy is stochastic, methods that promote stochasticity in an uncertainty-agnostic manner can fail catastrophically. We begin by explaining the significance of this result: it is well-known that stochastic policies can be arbitrarily sub-optimal in a single MDP, and can be outperformed by deterministic policies. The result we describe is more subtle than this: there are epistemic POMDPs where any attempt at being stochastic in an uncertainty-agnostic manner is sub-optimal, and *also* any attempt at acting completely deterministically is also sub-optimal. Rather, the characteristic of Bayes-optimal behavior is to be stochastic in *some* states (where it has high uncertainty), and not stochastic in others, and a

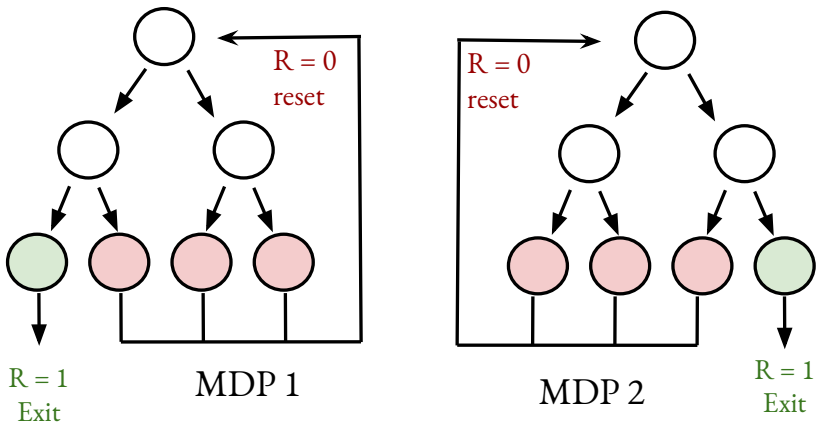


Figure 3.B.1: Visual description of Binary Tree MDPs described in proof of Proposition 3.B.4 with depth $n = 3$.

useful stochastic regularization method must modulate the level of stochasticity to calibrate with regions where it has high epistemic uncertainty.

Proposition 3.B.4. *Let $\alpha > 0, c > 0$. There exist posterior distributions $\mathcal{P}(\mathcal{M}|\mathcal{D})$, where the Bayes-optimal memoryless policy π^{*po} is stochastic. However, every memoryless policy π_s that is "everywhere-stochastic", in that $\forall s \in \mathcal{S} : \mathcal{H}(\pi_s(a|s)) > \alpha$, can have performance arbitrarily close to the uniformly random policy:*

$$\frac{J(\pi_s) - J(\pi_{unif})}{J(\pi^{*po}) - J(\pi_{unif})} < c$$

Proof. Consider two binary tree MDP with n levels, $\mathcal{M}_1$ and $\mathcal{M}_2$. A binary tree MDP, visualized in Figure 3.B.1, has n levels, where level k has 2^k states. On any level $k < n$, the agent can take a "left" action or a "right" action, which transitions to the corresponding state in the next level. On the final level, if the state corresponds to the terminal state (in green), then the agent receives a reward of 1, and the episode exits, and otherwise a reward of 0, and the agent returns to the top of the binary tree. The two binary tree MDPs $\mathcal{M}_1$ and $\mathcal{M}_2$ are identical except for the final terminal state: in $\mathcal{M}_1$, the terminal state is the left-most state in the final level, and in $\mathcal{M}_2$, the terminal state is the right-most state. Reaching the goal in $\mathcal{M}_1$ corresponds to taking

the "left" action repeatedly, and reaching the goal in $\mathcal{M}_2$ corresponds to taking the "right" action repeatedly. We consider the posterior distribution that places equal mass on $\mathcal{M}_1$ and $\mathcal{M}_2$, $\mathcal{P}(\mathcal{M}_1|\mathcal{D}) = \mathcal{P}(\mathcal{M}_2|\mathcal{D}) = \frac{1}{2}$. A policy that reaches the correct terminal state with probability p (otherwise reset) will visit the initial state a Geom(p) number of times, and writing $\overline{\gamma} \coloneqq \gamma^n$, will achieve return $\frac{\overline{\gamma}p}{1-\overline{\gamma}+p\overline{\gamma}} = \frac{1}{1+\frac{1}{p}\frac{1-\overline{\gamma}}{\overline{\gamma}}}$.

Uniform policy: A uniform policy randomly chooses between "left" and "right" at all states, and will reach all states in the final level equally often, so the probability it reaches the correct goal state is $\frac{1}{2^n}$. Therefore, the expected return is $J(\pi_{\text{unif}}) = \frac{1}{1+2^n\frac{1-\overline{\gamma}}{\overline{\gamma}}}$.

Bayes-optimal memoryless policy: The Bayes-optimal memoryless policy $\pi^{*\text{po}}$ chooses randomly between "left" and "right" at the top level; on every subsequent level, if the agent is in the left half of the tree, the agent deterministically picks "left" and on the right half of the tree, the agent deterministically picks "right". Effectively, this policy either visits the left-most state or the right-most state in the final level. The Bayes-optimal memoryless policy returns to the top of the tree a Geom($p = \frac{1}{2}$) number of times, and the expected return is given by $J(\pi^{*\text{po}}) = \frac{1}{1+2\frac{1-\overline{\gamma}}{\overline{\gamma}}}$.

Everywhere-stochastic policy: Unlike the Bayes-optimal policy, which is deterministic in all levels underneath the first, an everywhere-stochastic policy will sometimes take random actions at these lower levels, and therefore can reach states at the final level that are neither the left-most or right-most states (and therefore always bad). We note that if $\mathcal{H}(\pi(a|s)) > \alpha$, then there is some $\beta > 0$ such that $\max_a \pi(a|s) < 1 - \beta$. For an α-everywhere stochastic policy, the probability of taking at least one incorrect action increases as the depth of the binary tree grows, getting to the correct goal at most probability $\frac{1}{2}(1-\beta)^{n-1}$. The maximal expected return is therefore $J(\pi_s) \leqslant \frac{1}{1+2(\frac{1}{1-\beta})^{n-1}\frac{1-\overline{\gamma}}{\overline{\gamma}}}$

$$J(\pi^{*\text{po}}) = \frac{1}{1+2\frac{1-\overline{\gamma}}{\overline{\gamma}}} \qquad J(\pi_s) = \frac{1}{1+2(\frac{1}{1-\beta})^{n-1}\frac{1-\overline{\gamma}}{\overline{\gamma}}} \qquad J(\pi_{\text{unif}}) = \frac{1}{1+2^n\frac{1-\overline{\gamma}}{\overline{\gamma}}}$$

As $n \to \infty$, $J(\pi^{*\text{po}}), J(\pi_s)$ and $J(\pi_{\text{unif}})$ will converge to zero. Using asymptotic analysis we can determine their speed of convergence and find that:

$$J(\pi^{*\text{po}}) \sim \frac{\overline{\gamma}}{2} \qquad J(\pi_s) \sim \frac{\overline{\gamma}}{2(\frac{1}{1-\beta})^{n-1}} \qquad J(\pi_{\text{unif}}) \sim \frac{\overline{\gamma}}{2^n}$$

Using these asymptotics, we find that

$$\frac{J(\pi_s) - J(\pi_{\text{unif}})}{J(\pi^{*\text{po}}) - J(\pi_{\text{unif}})} \sim \frac{1}{(\frac{1}{1-\beta})^{n-1}} = (1-\beta)^{n-1},$$

which shows that this ratio can be made arbitrarily small as we increase n. □

An aside: deterministic policies While this proposition only discusses the failure mode of stochastic policies, *all* deterministic memoryless policies in this environment also fail. A deterministic policy π_d in this environment continually loops through one path in the binary tree repeatedly, and therefore will only ever reach one goal state, unlike the Bayes-optimal policy which visits both possible goal states. The best deterministic policy then either constantly takes the "left" action (which is optimal for $\mathcal{M}_1$), or constantly takes the "right" action (which is optimal for $\mathcal{M}_2$). Any other deterministic policy reaches a final state that is neither the left-most nor the right-most state, and will always get 0 reward. The expected return of the optimal deterministic policy is $J(\pi_d) = \frac{\overline{\gamma}}{2}$, receiving $\overline{\gamma}$ reward in one of the MDPs, and 0 reward in the other. When the discount factor γ is close to 1, the maximal expected return of a deterministic policy is approximately $\frac{1}{2}$, while the expected return of the Bayes-optimal policy is approximately 1, indicating a sub-optimality gap.

□

3.B.5 Proof of Proposition 3.6.1

Proposition 3.6.1. *Let $\pi, \pi_1, \cdots \pi_n$ be memoryless , and define $r_{\max} = \max_{i,s,a} |r_{\mathcal{M}_i}(s,a)|$. The expected return of π in $\hat{\mathcal{M}}^{po}$ is bounded below as:*

$$J_{\hat{\mathcal{M}}^{po}}(\pi) \geqslant \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i) - \frac{\sqrt{2}r_{\max}}{(1-\gamma)^2 n}\sum_{i=1}^{n}\mathbb{E}_{s\sim d_{\mathcal{M}_i}^{\pi_i}}\left[\sqrt{D_{KL}\left(\pi_i(\cdot|s)\;||\;\pi(\cdot|s)\right)}\right], \tag{3.3}$$

Proof. Before we begin, we recall some basic tools from analysis of MDPs. For a memoryless policy π, the state-action value function $Q^{\pi}(s,a)$ is given by:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[\sum_{t\geqslant 0}\gamma^t r(s_t,a_t)|s_0 = s, a_0 = a]. \tag{3.19}$$

The advantage function $A^{\pi}(s,a)$ is defined as:

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^{\pi}(s,a)]. \tag{3.20}$$

The performance difference lemma (Kakade and Langford, 2002) relates the expected return of two policies π and π' in an MDP $\mathcal{M}$ via their advantage functions as

$$J_{\mathcal{M}}(\pi') = J_{\mathcal{M}}(\pi) + \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\mathcal{M}}^{\pi'}}[\mathbb{E}_{a\sim\pi'}[A_{\mathcal{M}}^{\pi}(s,a)]]. \tag{3.21}$$

We now begin the derivation of our lower bound:

$$\begin{aligned}
J_{\hat{\mathcal{M}}^{\mathrm{po}}}(\pi) &= \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi) \\
&= \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i) + \frac{1}{n}\sum_{i=1}^{n}\left[J_{\mathcal{M}_i}(\pi) - J_{\mathcal{M}_i}(\pi_i)\right] \\
&= \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i) - \frac{1}{n(1-\gamma)}\sum_{i=1}^{n}\mathbb{E}_{s\sim d_{\mathcal{M}_i}^{\pi_i}}\left[\mathbb{E}_{a\sim\pi_i}\left[A_{\mathcal{M}_i}^{\pi}(s,a)\right]\right] \\
&= \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i) - \frac{1}{n(1-\gamma)}\sum_{i=1}^{n}\mathbb{E}_{s\sim d_{\mathcal{M}_i}^{\pi_i}}\left[\mathbb{E}_{a\sim\pi_i}\left[A_{\mathcal{M}_i}^{\pi}(s,a)\right] - \mathbb{E}_{a\sim\pi}\left[A_{\mathcal{M}_i}^{\pi}(s,a)\right]\right]
\end{aligned} \tag{3.22}$$

In the last equality we used the fact that $\mathbb{E}_{a\sim\pi}\left[A^{\pi}(s,a)\right]=0$. From there we proceed to derive a lower bound:

$$\begin{aligned}
J_{\hat{\mathcal{M}}^{\mathrm{po}}}(\pi) &= \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i) - \frac{1}{n(1-\gamma)}\sum_{i=1}^{n}\mathbb{E}_{s\sim d^{\pi_i}_{\mathcal{M}_i}}\left[\mathbb{E}_{a\sim\pi_i}\left[A^{\pi}_{\mathcal{M}_i}(s,a)\right]-\mathbb{E}_{a\sim\pi}\left[A^{\pi}_{\mathcal{M}_i}(s,a)\right]\right] \\
&\geqslant \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i) - \frac{2r_{max}}{n(1-\gamma)^2}\sum_{i=1}^{n}\mathbb{E}_{s\sim d^{\pi_i}_{\mathcal{M}_i}}\left[D_{TV}\left(\pi_i(\cdot\mid s);\pi(\cdot\mid s)\right)\right] \\
&\geqslant \frac{1}{n}\sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i) - \frac{\sqrt{2}r_{max}}{(1-\gamma)^2 n}\sum_{i=1}^{n}\mathbb{E}_{s\sim d^{\pi_i}_{\mathcal{M}_i}}\left[\sqrt{D_{KL}\left(\pi_i(\cdot\mid s)\;||\;\pi(\cdot\mid s)\right)}\right]
\end{aligned} \tag{3.23}$$

where the first inequality is since $|A^{\pi}_{\mathcal{M}_i}(s,a)|\leqslant\frac{r_{\max}}{1-\gamma}$ and the second from Pinsker's inequality. Our intention in this derivation is not to obtain the tighest lower bound possible, but rather to illustrate how bounding the advantage can lead to a simple lower bound on the expected return in the POMDP. The inequality can be made tighter using other bounds on $|A^{\pi}_{\mathcal{M}_i}(s,a)|$, for example using $A_{\max}=\max_{i,s,a}|A^{\pi}_{\mathcal{M}_i}(s,a)|$, or potentially a bound on the advantage that varies across state. □

3.B.6 Proof of Proposition 3.6.2

Proposition 3.6.2. *Let $f:\{\pi_i\}_{i\in[n]}\mapsto\pi$ be a function that maps n policies to a single policy satisfying $f(\pi,\cdots,\pi)=\pi$ for every policy π, and let α be a hyperparameter satisfying $\alpha\geqslant\frac{\sqrt{2}r_{max}}{(1-\gamma)^2 n}$. Then letting $\pi^*_1,\dots\pi^*_n$ be the optimal solution to the following optimization problem:*

$$\{\pi^*_i\}_{i\in[n]}=\underset{\pi_1,\cdots,\pi_n}{\arg\max}\,\frac{1}{n}\sum_{i=1}^{n}J_{\mathcal{M}_i}(\pi_i)-\alpha\sum_{i=1}^{n}\mathbb{E}_{s\sim d^{\pi_i}_{\mathcal{M}_i}}\left[\sqrt{D_{KL}\left(\pi_i(\cdot|s)\;||\;f(\{\pi_i\})(\cdot|s)\right)}\right], \tag{3.4}$$

the policy $\pi^*:=f(\{\pi^*_i\}_{i\in[n]})$ is optimal for the empirical epistemic POMDP $\hat{\mathcal{M}}^{po}$.

Proof. By Proposition 3.6.1 we have that $\forall \alpha \geqslant \frac{\sqrt{2} r_{\max}}{(1-\gamma)^2 n}$:

$$J_{\hat{\mathcal{M}}^{\text{po}}}(f(\{\pi_i^*\})) \geqslant \frac{1}{n} \sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi_i^*) - \alpha \sum_{i=1}^{n} \mathbb{E}_{s \sim d_{\mathcal{M}_i}^{\pi_i^*}} \left[\sqrt{D_{KL}\left(\pi_i^*(\cdot|s) \;||\; f(\{\pi_i^*\})(\cdot|s)\right)} \right]. \tag{3.24}$$

Now, write $\pi'^* \in \arg\max_\pi J_{\hat{\mathcal{M}}^{\text{po}}}(\pi)$ to be an optimal policy in the empirical epistemic POMDP, and consider the collection of policies $\{\pi'^*, \pi'^*, \ldots, \pi'^*\}$. Since $\{\pi_i^*\}$ is the optimal solution to Equation 3.4, we have

$$\begin{aligned} J_{\hat{\mathcal{M}}^{\text{po}}}(f(\{\pi_i^*\})) &\geqslant \frac{1}{n} \sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi'^*) - \alpha \sum_{i=1}^{n} \mathbb{E}_{s \sim d_{\mathcal{M}_i}^{\pi'^*}} \left[\sqrt{D_{KL}\left(\pi'^*(\cdot|s) \;||\; f(\{\pi'^*\})(\cdot|s)\right)} \right] \\ &= \frac{1}{n} \sum_{i=1}^{n} J_{\mathcal{M}_i}(\pi'^*) \\ &= J_{\hat{\mathcal{M}}^{\text{po}}}(\pi'^*), \end{aligned} \tag{3.25}$$

where the second line here uses the fact that $f(\pi'^*, \ldots, \pi'^*) = \pi'^*$. Therefore $\pi^* \coloneqq f(\{\pi_i^*\})$ is optimal for the empirical epistemic POMDP. □

3.C Procgen Implementation and Experimental Setup

We follow the training and testing scheme defined by Cobbe et al. (Cobbe et al., 2020) for the Procgen benchmarks: the agent trains on a fixed set of levels, and is tested on the full distribution of levels. Due to our limited computational budget, we train on the so-called "easy" difficulty mode using the recommended 200 training levels. Nonetheless, many prior work has found a significant generalization gap between test and train performance even in this easy setting, indicating it a useful benchmark for generalization (Cobbe et al., 2020; Raileanu et al., 2020; Jiang et al., 2020). We implemented LEEP on top of an existing open-source codebase released by Jiang et al. (Jiang et al., 2020). Full code is provided in the supplementary for reference.

LEEP maintains $n = 4$ policies $\{\pi_i\}_{i\in[n]}$, each parameterized by the ResNet architecture prescribed by Cobbe et al. (2020). In LEEP, each policy is optimized to maximize the entropy-regularized PPO surrogate objective alongside a one-step KL divergence penalty between itself and the linked policy $\max_i \pi_i$; gradients are not taken through the linked policy.

$$\mathbb{E}_{\pi_i}[\min(r_t(\pi)A^{\pi}(s,a), \text{clip}(r_t(\pi), 1-\varepsilon, 1+\varepsilon)\ A^{\pi}(s,a) + \beta\mathcal{H}(\pi_i(a|s)) - \alpha D_{KL}(\pi_i(a|s)\|\max_j \pi_j(a|s))]$$

Note that this update in Equation 6 is not exactly solving the optimization problem dictated by Equation 5, since it leverages a one-step estimator for the gradient of the KL penalty in the PG objective, a heuristic known to lead to better optimization in PPO and other deep policy gradient methods. If the proper estimator for the KL penalty is substituted in, then the Bayes-optimal policy in the empirical epistemic POMDP is an optimal solution for Equation 6.

The penalty hyperparameter α was obtained by performing a hyperparameter search on the Maze task for all the comparison methods (including LEEP) amongst $\alpha \in [0.01, 0.1, 1.0, 10.0]$. Since LEEP trains 4 policies using the same environment budget as a single PPO policy, we change the number of environment steps per PPO iteration from 16384 to 4096, so that the PPO baseline and each policy in our method takes the same number of PPO updates. All other PPO hyperparameters are taken directly from Jiang et al. (2020).

In our implementation, we parallelize training of the policies across GPUs, using one GPU for each policy. We found it infeasible to run more ensemble members due to GPU memory constraints without significant slowdown in wall-clock time. Running LEEP on one Procgen environment for 50 million steps requires approximately 5 hrs in our setup on a machine with four Tesla T4 GPUs.

3.D Procgen Results

3.D.1 Main Experimental Results

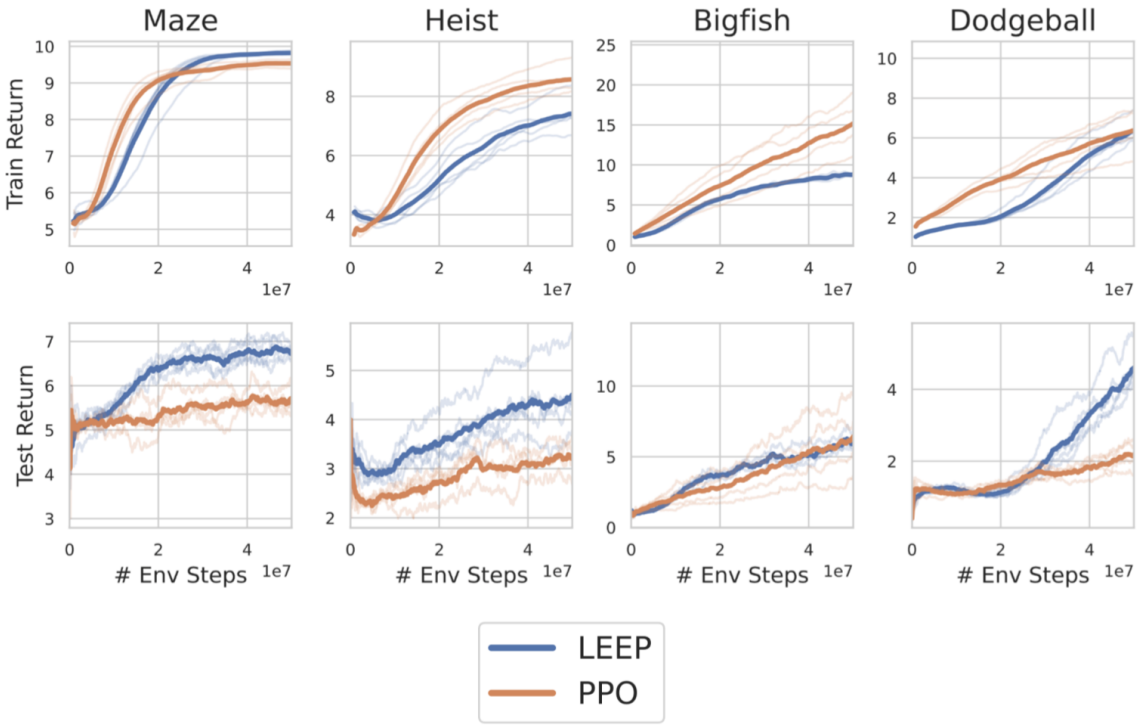


Figure 3.D.1: Training (top) and test (bottom) returns for LEEP and PPO on four Procgen environments. Results averaged across 5 random seeds. LEEP achieves equal or higher training return compared to PPO, while having a lower generalization gap between test and training returns.

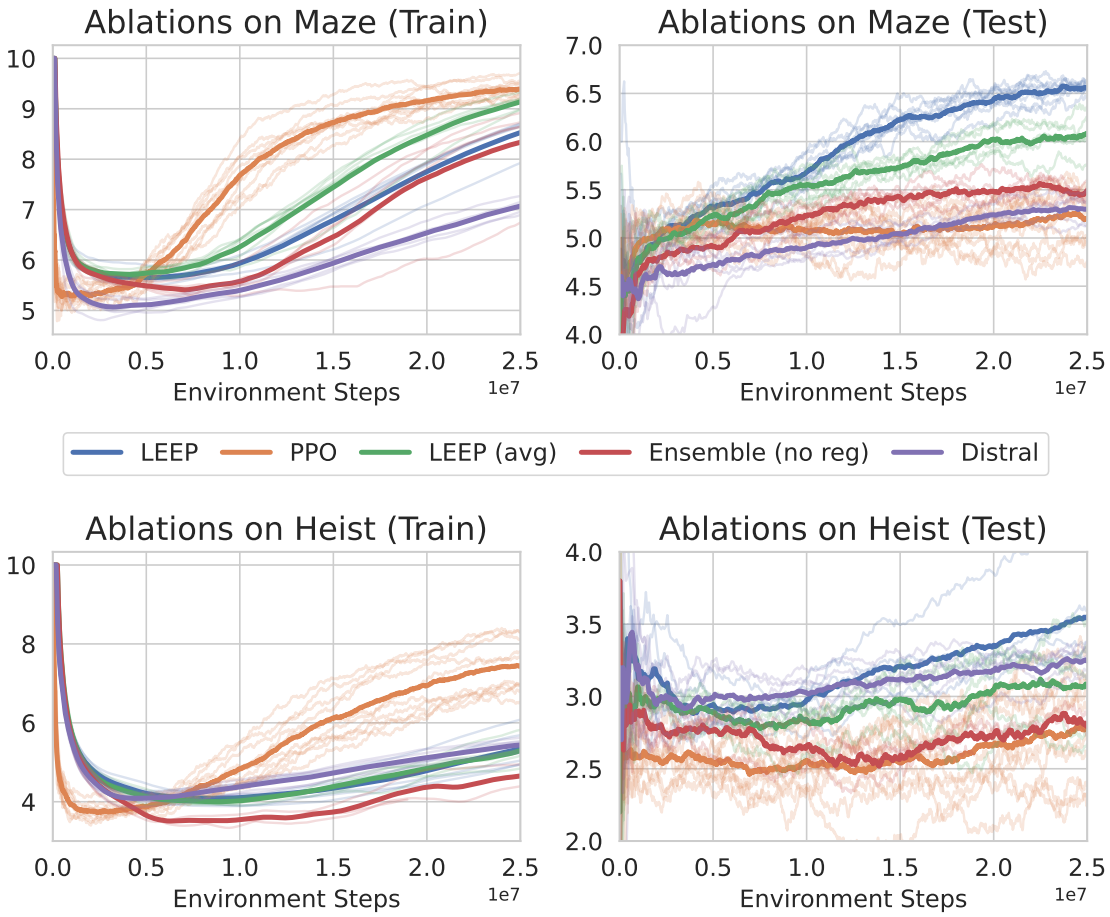


Figure 3.D.2: Training and test returns for various ablations and comparisons of LEEP.

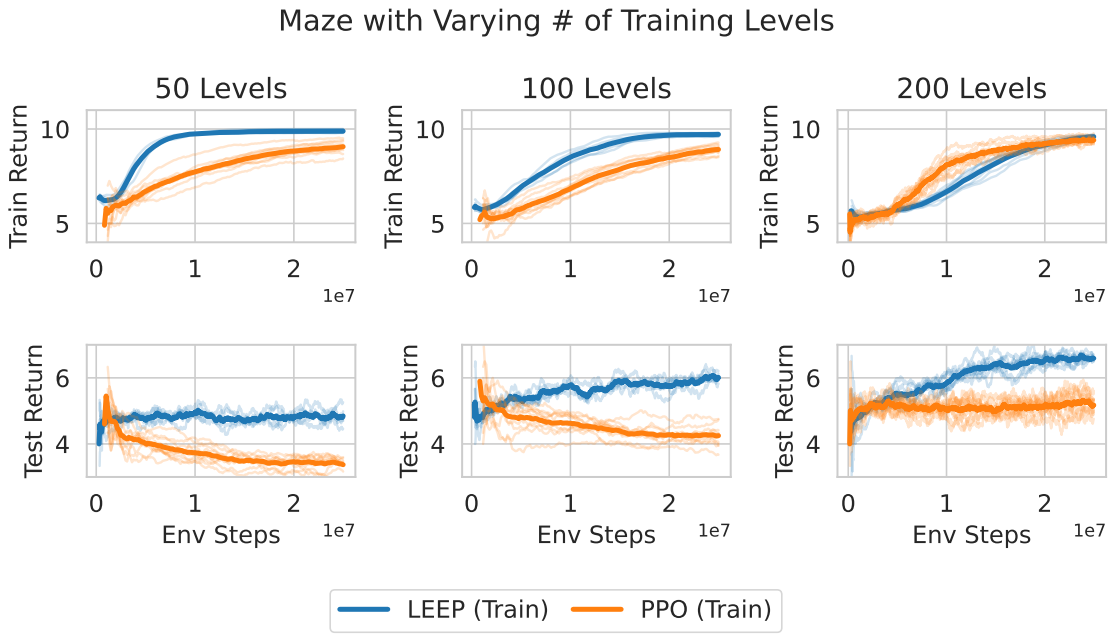


Figure 3.D.3: Performance of LEEP and PPO as the number of training levels provided varies. While the learned performance of the PPO policy is worse than a *random policy* with less training levels, LEEP avoids this overfitting and in general, demonstrates a smaller train-test performance gap than PPO.

3.D.2 Ablations of LEEP Hyperparameters

Number of ensemble members (n): We ran an ablation study on the Procgen Maze task to understand how the number of ensemble members affects the performance of LEEP. We found that for an equal number of gradient steps per ensemble member, LEEP does equally well with $n = 4$ and 8 ensemble members, but poorly with only 1 or 2 ensemble members (see Figure attached). These results indicate that at least on the Maze task, using n=4 ensemble members is an appropriate balance between approximating the true epistemic POMDP with higher fidelity and minimizing the sample complexity incurred by needing to train more ensemble members with on-policy RL methods.

# Ensemble members (n)	**1**	**2**	**4**	**8**
Maze	5.11 ± 0.24	5.85 ± 0.4	6.53 ± 0.12	6.91 ± 0.1

Penalty coefficient (α): We performed a coarse hyperparameter sweem on the four Procgen domains, testing values $\alpha \in 10^{\{-2,-1,0,1,2\}}$. The results in the table below indicated that performance is roughly coniststent for $\alpha = \{0.1, 1, 10\}$, so while performance does depend on this hyperparameter, it is not overly sensitive, and values around 1 are likely to be a good default initialization.

Penalty parameter(α)	**0**	**0.01**	**0.1**	**1**	**10**	**100**
Maze	5.78	5.725	5.94 ± 0.22	6.53 ± 0.12	6.54 ± 0.15	5.7
Heist	3.3	3.4	3.2 ± 0.6	3.73 ± 0.45	3.65 ± 0.5	3.15
Bigfish	1.57	2.35	2.85 ± 0.64	4.16 ± 0.42	3.30 ± 0.38	1.21
Dodgeball	0.65	0.94	0.78 ± 0.2	1.69 ± 0.18	1.42 ± 0.4	1.64

3.D.3 LEEP and *implicit* partial observability

One common confusion that may arise is that LEEP seeks to overcome partial observability of the contexts, as is done for dynamics generalization in POMDPs (e.g. (Lee et al., 2020b)). This is not the case. Works on dynamics generalization in POMDPs assume that contexts in the true underlying environment are partially observable (e.g. friction coefficients unobserved by a robot without the proper sensors), and the aim to infer this context using memory. In the epistemic POMDP, *the context is not partially observable*; rather, what is partially observable is how the system dynamics will behave for any provided context, capturing the agent's epistemic uncertainty that stems from the limited training contexts.

We conducted a didactic experiment on Procgen to empirically support the claim that the partial observability modelled by dynamics generalization methods Lee et al. (2020b) does not replace explicit handling of epistemic uncertainty provided by our method (since this is a different problem). We train a recurrent context encoder that takes in the trajectory seen so far and predicts the identity of the training level. The

last hidden layer of this encoder is taken as a "context vector" and fed in as input into a policy alongside the original state, creating an adaptive recurrent policy since this context vector can change through a trajectory. We tested this model on our four Procgen tasks, and made two observations. First, the learned policy, despite being recurrent, does not achieve higher test-time performance than PPO. This is not surprising, because the task is fully observed at training-time. Second, the learned context encoder is able to predict the identity of the training level with $> 99\%$ accuracy; that is, the contexts are fully observed and so mechanisms that try to predict the context are unlikely to provide benefit.

The issue is that recurrency and adaptation by themselves are not sufficient to ensure high generalization performance; rather they must be combined with the appropriate model of partial observability that captures the agent's epistemic uncertainty (for LEEP, by statistical bootstrapping on the set of training contexts) to achieve good generalization.

Test Return after 25M steps	**Maze**	**Heist**	**Bigfish**	**Dodgeball**
PPO	5.11 ± 0.24	2.84 ± 0.46	3.89 ± 1.64	1.68 ± 0.33
PPO + Recurrent Context Encoder	5.25 ± 0.5	2.83 ± 1.04	2.74 ± 1.1	1.57 ± 0.3
LEEP	6.53 ± 0.12	3.73 ± 0.45	4.16 ± 0.42	1.69 ± 0.18

Part II

Auction Design

Chapter 4

A Permutation-Equivariant Neural Network Architecture For Auction Design

Designing an incentive compatible auction that maximizes expected revenue is a central problem in Auction Design. Theoretical approaches to the problem have hit some limits in the past decades and analytical solutions are known for only a few simple settings. Computational approaches to the problem through the use of LPs have their own set of limitations. Building on the success of deep learning, a new approach was recently proposed by Duetting et al. (2019) in which the auction is modeled by a feed-forward neural network and the design problem is framed as a learning problem. The neural architectures used in that work are general purpose and do not take advantage of any of the symmetries the problem could present, such as permutation equivariance. In this chapter, we consider auction design problems that have permutation-equivariant symmetry and construct a neural architecture that is capable of perfectly recovering the permutation-equivariant optimal mechanism, which we show is not possible with the previous architecture. We demonstrate that

permutation-equivariant architectures are not only capable of recovering previous results, they also have better generalization properties.

4.1 Introduction

Designing truthful auctions is one of the core problems that arise in economics. Concrete examples of auctions include sales of treasury bills, art sales by Christie's or Google Ads. Following seminal work of Vickrey (Vickrey, 1961) and Myerson (Myerson, 1981), auctions are typically studied in the *independent private valuations* model: each bidder has a valuation function over items, and their payoff depends only on the items they receive. Moreover, the auctioneer knows aggregate information about the population that each bidder comes from, modeled as a distribution over valuation functions, but does not know precisely each bidder's valuation. Auction design is challenging since the valuations are private and bidders need to be encouraged to report their valuations truthfully. The auctioneer aims at designing an incentive compatible auction that maximizes revenue.

While auction design has existed as a subfield of economic theory for several decades, complete characterizations of the optimal auction only exist for a few settings. Myerson resolved the optimal auction design problem when there is a single item for sale (Myerson, 1981). However, the problem is not completely understood even in the extremely simple setting with just a single bidder and two items. While there have been some partial characterizations (Manelli and Vincent, 2006, 2010; Pavlov, 2011; Wang and Tang, 2014; Daskalakis et al., 2017), and algorithmic solutions with provable guarantees (Alaei, 2011; Alaei et al., 2012, 2013; Cai et al., 2012a,b), neither the analytic nor algorithmic approach currently appears tractable for seemingly small instances.

Another line of work to confront this theoretical hurdle consists in building automated methods to find the optimal auction. Early works (Conitzer and Sandholm, 2002, 2004) framed the problem as a linear program. However, this approach suffers from severe scalablility issues as the number of constraints and variables is exponential in the number of bidders and items (Guo and Conitzer, 2010). Later, Sandholm and Likhodedov (2015) designed algorithms to find the optimal auction. While scalable, they are however limited to specific classes of auctions known to be incentive compatible.

A more recent research direction consists in building deep learning architectures that design auctions from samples of bidder valuations. Duetting et al. (2019) proposed RegretNet, a feed-forward architecture to find near-optimal results in several known multi-item settings and obtain new mechanisms in unknown cases. This architecture however is not data efficient and can require a large number of valuation samples to learn an optimal auction in some cases. This inefficiency is not specific to RegretNet but is characteristic of neural network architectures that do not incorporate any inductive bias.

In this chapter, we build a deep learning architecture for *multi-bidder symmetric auctions*. These are auctions which are invariant to relabeling the items or bidders. More specifically, such auctions are *anonymous* (in that they can be executed without any information about the bidders, or labeling them) and *item-symmetric* (in that it only matters what bids are made for an item, and not its a priori label).

It is now well-known that when bidders come from the same population that the optimal auction itself is anonymous. Similarly, if items are *a priori indistinguishable* (e.g. different colors of the same car — individuals certainly value a red vs. blue car differently, but there is nothing objectively more/less valuable about a red vs. blue car), the optimal auction is itself item-symmetric. In such settings, our approach will approach the true optimum in a way which retains this structure (see Contributions

below). Even without these conditions, the optimal auction is often symmetric anyway: for example, "bundling together" (the auction which allows bidders to pay a fixed price for all items, or receive nothing) is item-symmetric, and is often optimal even when the items are a priori distinguishable.

Beyond their frequent optimality, such auctions are desirable objects of study *even when they are suboptimal*. For example, seminal work of Hartline and Roughgarden which pioneered the study of "simple vs. optimal auctions" analyzes the approximation guarantees achievable by anonymous auctions Hartline and Roughgarden (2009), and exciting recent work continues to improve these guarantees Alaei et al. (2015); Jin et al. (2019b,a). Similarly, Daskalakis and Weinberg (2012) develop algorithms for item-symmetric instances, and exciting recent work show how to leverage item-symmetric to achieve near-optimal auctions in completely general settings (Kothari et al., 2019). To summarize: symmetric auctions are known to be optimal in many settings of interest (even those which are not themselves symmetric). Even in settings where they are not optimal, they are known to yield near-optimal auctions. And even when they are only approximately optimal, seminal work has identified them as important objects of study owing to their simplicity. In modern discussion of auctions, they are also desirable due to fairness considerations.

While applying existing feed-forward architectures as RegretNet to symmetric auctions is possible, we show in Section 4.3 that RegretNet struggles to find symmetric auctions, *even when the optimum is symmetric*. To be clear, the architecture's performance is indeed quite close to optimal, but the resulting auction is not "close to symmetric". This chapter proposes an architecture that outputs a symmetric auction symmetry by design.

Contributions

This chapter identifies three drawbacks from using the RegretNet architecture when learning with symmetric auctions. First, RegretNet is incapable of finding symmetric auctions when the optimal mechanism is known to be symmetric. Second, RegretNet is sample inefficient, which is not surprising since the architecture does not incorporate any inductive bias. Third, RegretNet is incapable of generalizing to settings with a different number of bidders of objects. In fact, by construction, the solution found by RegretNet can only be evaluated on settings with exactly the same number of bidders and objects of the setting it was trained on.

We address these limitations by proposing a new architecture EquivariantNet, that outputs symmetric auctions. EquivariantNet is an adaption of the deep sets architecture (Hartford et al., 2018) to symmetric auctions. This architecture is parameter-efficient and is able to recover some of the optimal results in the symmetric auctions literature. Our approach outlines three important benefits:

- *Symmetry*: our architecture outputs a symmetric auction by design. It is immune to permutation-sensitivity as defined in Section 4.3.1 which is related to fairness.
- *Sample generalization*: Because we use domain knowledge, our architecture converges to the optimum with fewer valuation samples.
- *Out-of-setting generalization*: Our architecture does not require hard-coding the number of bidders or items during training — training our architecture on instances with n bidders and m items produces a well-defined auction even for instances with n' bidders and m' items. Somewhat surprisingly, we show in 4.4 some examples where our architecture trained on 1 bidder with 5 items generalizes well even to 1 bidder and m items, for any $m \in \{2, 10\}$.

We highlight that the novelty of this work is not to show that a new architecture is a viable alternative to RegretNet. Instead we are solving three fundamental limitations

we identified for the RegretNet architecture. These three problems are not easy to solve in principle, it is surprising that a change of architecture solves all of them in the context of symmetric auctions. We would also like to emphasize that both RegretNet and EquivariantNet are capable of learning auction with near optimal revenue and negligible regret. It is not possible to significantly outperform RegretNet on these aspects. The way we improve over RegretNet is by having better sample efficiency, out-of-setting generalization and by ensuring that our solutions are exactly equivariant.

The chapter decomposes as follows. Section 4.2 introduces the standard notions of auction design. Section 4.3 presents our permutation-equivariant architecture to encode symmetric auctions. Finally, Section 4.4 presents numerical evidence for the effectiveness of our approach.

Related work

Auction design and machine learning. Machine learning and computational learning theory have been used in several ways to design auctions from samples of bidder valuations. Some works have focused sample complexity results for designing optimal revenue-maximizing auctions. This has been established in single-parameter settings (Dhangwatnotai et al., 2015; Cole and Roughgarden, 2014; Morgenstern and Roughgarden, 2015; Medina and Mohri, 2014; Huang et al., 2018; Devanur et al., 2016; Hartline and Taggart, 2019; Roughgarden and Schrijvers, 2016; Gonczarowski and Nisan, 2017; Guo et al., 2019), multi-item auctions (Dughmi et al., 2014; Gonczarowski and Weinberg, 2018), combinatorial auctions (Balcan et al., 2016; Morgenstern and Roughgarden, 2016; Syrgkanis, 2017) and allocation mechanisms (Narasimhan and Parkes, 2016). Machine learning has also been used to optimize different aspects of mechanisms (Lahaie, 2011; Dütting et al., 2015). All these aforementioned differ from ours as we resort to deep learning for finding optimal auctions.

Auction design and deep learning. While Duetting et al. (2019) is the first paper to design auctions through deep learning, several other paper followed-up this work. Feng et al. (2018) extended it to budget constrained bidders, Golowich et al. (2018) to the facility location problem. Tacchetti et al. (2019) built architectures based on the Vickrey-Clarke-Groves auctions. Recently, Shen et al. (2019) and Duetting et al. (2019) proposed architectures that *exactly* satisfy incentive compatibility but are specific to *single-bidder* settings. In this work, we aim at *multi-bidder* settings and build permutation-equivariant networks that return nearly incentive compatibility symmetric auctions.

4.2 Symmetries and learning problem in auction design

We review the framework of auction design and the problem of finding truthful mechanisms. We then present symmetric auctions and similarly to Duetting et al. (2019), frame auction design as a learning problem.

4.2.1 Auction design and symmetries

Auction design. We consider the setting of additive auctions with n bidders with $N = \{1, \dots, n\}$ and m items with $M = \{1, \dots, m\}$. Each bidder i is has value v_{ij} for item j, and values the set S of items at $\sum_{j \in S} v_{ij}$. Such valuations are called *additive*, and are perhaps the most well-studied valuations in multi-item auction design (Hart and Nisan, 2012, 2013; Li and Yao, 2013; Babaioff et al., 2014; Daskalakis et al., 2014a; Hart and Reny, 2015; Cai et al., 2016; Daskalakis et al., 2017; Beyhaghi and Weinberg, 2019).

The designer does not know the full valuation profile $V = (v_{ij})_{i \in N, j \in M}$, but just a distribution from which they are drawn. Specifically, the valuation vector of bidder

i for each of the m items $\vec{v}_i = (v_{i1}, \ldots, v_{im})$ is drawn from a distribution D_i over $\mathbb{R}^m$ (and then, V is drawn from $D := \times_i D_i$). The designer asks the bidders to report their valuations (potentially untruthfully), then decides on an allocation of items to the bidders and charges a payment to them.

Definition 3. *An auction is a pair* (g, p) *consisting of a randomized allocation rule* $g = (g_1, \ldots, g_n)$ *where* $g_i \colon \mathbb{R}^{n \times m} \to [0,1]^m$ *such that for all* V*, and all* j*,* $\sum_i (g_i(V))_j \leqslant 1$ *and payment rules* $p = (p_1, \ldots, p_n)$ *where* $p_i \colon \mathbb{R}^{n \times m} \to \mathbb{R}_{\geqslant 0}$ *.*

Given reported bids $B = (b_{ij})_{i \in N, j \in M}$, the auction computes an allocation probability $g(B)$ and payments $p(B)$. $[g_i(B)]_j$ is the probability that bidder i received object j and $p_i(B)$ is the price bidder i has to pay to the mechanism. In what follows, $\mathcal{M}$ denotes the class of all possible auctions.

Definition 4. *The utility of bidder* i *is defined by* $u_i(\vec{v}_i, B) = \sum_{j=1}^{m} [g_i(B)]_j v_{ij} - p_i(B)$.

Bidders seek to maximize their utility and may report bids that are different from their valuations. Let V_{-i} be the valuation profile without element $\vec{v}_i$, similarly for B_{-i} and $D_{-i} = \times_{j \neq i} D_j$. We aim at auctions that invite bidders to bid their true valuations through the notion of incentive compatibility.

Definition 5. *An auction* (g, p) *is dominant strategy incentive compatible (DSIC) if each bidder's utility is maximized by reporting truthfully no matter what the other bidders report. For every bidder* i*, valuation* $\vec{v}_i \in D_i$*, bid* $\vec{b}_i{}' \in D_i$ *and bids* $B_{-i} \in D_{-i}$*,* $u_i(\vec{v}_i, (\vec{v}_i, B_{-i})) \geqslant u_i(\vec{v}_i, (\vec{b}_i{}', B_{-i}))$.

Additionally, we aim at auctions where each bidder receives a non-negative utility.

Definition 6. *An auction is individually rational (IR) if for all* $i \in N$*,* $\vec{v}_i \in D_i$ *and* $B_{-i} \in D_{-i}$*,*

$$u_i(\vec{v}_i, (\vec{v}_i, B_{-i})) \geqslant 0. \tag{IR}$$

In a DSIC auction, the bidders have the incentive to truthfully report their valuations and therefore, the revenue on valuation profile V is defined as $\sum_{i=1}^{n} p_i(V)$. Optimal auction design aims at finding a DSIC auction that maximizes the expected revenue $rev := \mathbb{E}_{V\sim D}[\sum_{i=1}^{n} p_i(V)]$.

Linear program. We frame the problem of optimal auction design as an optimization problem where we seek an auction that minimizes the negated expected revenue among all IR and DSIC auctions. Since there is no known characterization of DSIC mechanisms in the multi-bidder setting, we resort to the relaxed notion of *ex-post regret*. It measures the extent to which an auction violates DSIC, for each bidder.

Definition 7. *The ex-post regret for a bidder i is the maximum increase in his utility when considering all his possible bids and fixing the bids of others. For a valuation profile V, the ex-post regret for a bidder i is $rgt_i(V) = \max_{\vec{v}_i{}'\in\mathbb{R}^m} u_i(\vec{v}_i; (\vec{v}_i{}', V_{-i})) - u_i(\vec{v}_i; (\vec{v}_i, V_{-i}))$. In particular, DSIC is equivalent to*

$$rgt_i(V) = 0,\ \forall i \in N. \tag{IC}$$

Therefore, by setting equation IC and equation IR as constraints, finding an optimal auction is equivalent to the following linear program

$$\min_{(g,p)\in\mathcal{M}} -\mathbb{E}_{V\sim D}\left[\sum_{i=1}^{n} p_i(V)\right] \quad \text{s.t.} \quad rgt_i(V) = 0, \qquad \forall i \in N,\ \forall V \in D,$$
$$u_i(\vec{v}_i, (\vec{v}_i, B_{-i})) \geqslant 0, \quad \forall i \in N,\ \vec{v}_i \in D_i, B_{-i} \in D_{-i}. \tag{LP}$$

Symmetric auctions. Equation LP is intractable due to the exponential number of constraints. However, in the setting of *symmetric* auctions, it is possible to reduce the search space of the problem as shown in Theorem 1. We first define the notions of bidder- and item-symmetries.

Definition 8. *The valuation distribution $\boldsymbol{D}$ is bidder-symmetric if for any permutation of the bidders $\varphi_b \colon N \to N$, the permuted distribution $D_{\varphi_b} := D_{\varphi_b(1)} \times \cdots \times D_{\varphi_b(n)}$ satisfies: $D_{\varphi_b} = D$.*

Bidder-symmetry intuitively means that the bidders are a priori indistinguishable (although individual bidders will be different). This holds for instance in auctions where the identity of the bidders is anonymous, or if $D_i = D_j$ for all i, j (bidders are i.i.d.).

Definition 9. *Bidder i's valuation distribution D_i is item-symmetric if for any items $x_1, \ldots, x_m$ and any permutation $\varphi_o \colon M \to M$, $D_i(x_{\varphi_o(1)}, \ldots, x_{\varphi_o(m)}) = D_i(x_1, \ldots, x_m)$.*

Intuitively, item-symmetry means that the items are also indistinguishable but not identical. It holds when the distributions over the items are i.i.d. but this is not a necessary condition. Indeed, the distribution $\{(a, b, c) \in \mathcal{U}(0,1)^{\otimes 3} : a + b + c = 1\}$ is not i.i.d. but is item-symmetric.

Definition 10. *An auction is symmetric if its valuation distributions are bidder- and item-symmetric.*

We now define the notion of permutation-equivariance that is important in symmetric auctions.

Definition 11. *The functions g and p are permutation-equivariant if for any two permutation matrices $\Pi_n \in \{0,1\}^{n \times n}$ and $\Pi_m \in \{0,1\}^{m \times m}$, and any valuation matrix V, we have $g(\Pi_n V \Pi_m) = \Pi_n g(V) \Pi_m$ and $p(\Pi_n V \Pi_m) = \Pi_n p(V)$.*

Theorem 1. *When the auction is symmetric, there exists an optimal solution to equation LP that is permutation-equivariant.*

Theorem 1 is originally proved in Daskalakis and Weinberg (2012) and its proof is reminded in section 4.B for completeness. It encourages to reduce the search

space in equation LP by only optimizing over permutation-equivariant allocations and payments. We implement this idea in Section 4.3 where we build equivariant neural network architectures. Before, we frame auction design as a learning problem.

4.2.2 Auction design as a learning problem

Similarly to Duetting et al. (2019), we formulate auction design as a learning problem. We learn a parametric set of auctions (g^w, p^w) where $w \in \mathbb{R}^d$ parameters and $d \in \mathbb{N}$. Directly solving equation LP is challenging in practice. Indeed, the auctioneer must have access to the bidder valuations which are unavailable to her. Since she has access to the valuation distribution, we relax equation LP and replace the IC constraint for all $V \in D$ by the expected constraint $\mathbb{E}_{V\sim D}[rgt_i(V)] = 0$ for all $i \in N$.. In practice, the expectation terms are computed by sampling L bidder valuation profiles drawn i.i.d. from D. The empirical ex-post regret for bidder i is

$$\widehat{rgt}_i(w) = \frac{1}{L}\sum_{\ell=1}^{L} \max_{\vec{v}_i{}' \in \mathbb{R}^m} u_i^w(\vec{v}_i^{(\ell)}; (\vec{v}_i{}', V_{-i}^{(\ell)})) - u_i(\vec{v}_i^{(\ell)}; (\vec{v}_i^{(\ell)}, V_{-i}^{(\ell)})), \qquad (\hat{R})$$

where $u_i^w(\vec{v}_i, B) := \sum_{j=1}^m [g_i^w(B)]_j v_{ij} - p_i^w(B)$ is the utility of bidder i under the parametric set of auctions (g^w, p^w). Therefore, the learning formulation of equation LP is

$$\min_{w\in\mathbb{R}^d} -\frac{1}{L}\sum_{\ell=1}^{L}\sum_{i=1}^{n} p_i^w(V^{(\ell)}) \quad \text{s.t.} \quad \widehat{rgt}_i(w) = 0,\ \forall i \in N. \qquad (\widehat{\text{LP}})$$

Duetting et al. (2019) justify the validity of this reduction from equation LP to equation $\widehat{\text{LP}}$ by showing that the gap between the expected regret and the empirical regret is small as the number of samples increases. Additionally to being DSIC, the auction must satisfy IR. The learning problem equation $\widehat{\text{LP}}$ does not ensure this but we will show how to include this requirement in the architecture in section 4.3.

4.3 A Permutation-equivariant neural network architecture

We first show that feed-forward architectures as RegretNet (Duetting et al., 2019) may struggle to find a symmetric solution in auctions where the optimal solution is known to be symmetric. We then describe our neural network architecture, EquivariantNet that learns symmetric auctions. EquivariantNet is build using exchangeable matrix layers (Hartford et al., 2018).

4.3.1 Feed-forward nets and permutation-equivariance

In the following experiments we use the RegretNet architecture with the exact same training procedure and parameters as found in Duetting et al. (2019) .

Permutation-sensitivity. Given L bidders valuation samples $\{B^{(1)}, \dots, B^{(L)}\} \in \mathbb{R}^{n \times m}$, we generate for each bid matrix $B^{(\ell)}$ all its possible permutations $B^{(\ell)}_{\Pi_n, \Pi_m} := \Pi_n B^{(\ell)} \Pi_m$, where $\Pi_n \in \{0,1\}^{n \times n}$ and $\Pi_m \in \{0,1\}^{m \times m}$ are permutation matrices. We then compute the revenue for each one of these bid matrices and obtain a revenue matrix $R \in \mathbb{R}^{n!m! \times L}$. Finally, we compute $h_R \in \mathbb{R}^L$ where $[h_R]_j = \max_{i \in [n!m!]} R_{ij} - \min_{i \in [n!m!]} R_{ij}$. The distribution given by the entries of h_R is a measure of how close the auction is to permutation-equivariance. A symmetric mechanism satisfies $h_R = (0, \dots, 0)^\top$. Our numerical investigation considers the following auction settings:

- (I) One bidder and two items, the item values are drawn from $\mathcal{U}[0,1]$. Optimal revenue: 0.55 Manelli and Vincent (2006).
- (II) Four bidders and five items, the item values are drawn from $\mathcal{U}[0,1]$.

Figure 4.3.1 (a)-(b) presents the distribution of h_R of the optimal auction learned for setting (I) when varying the number of samples L. When L is large, the distribution

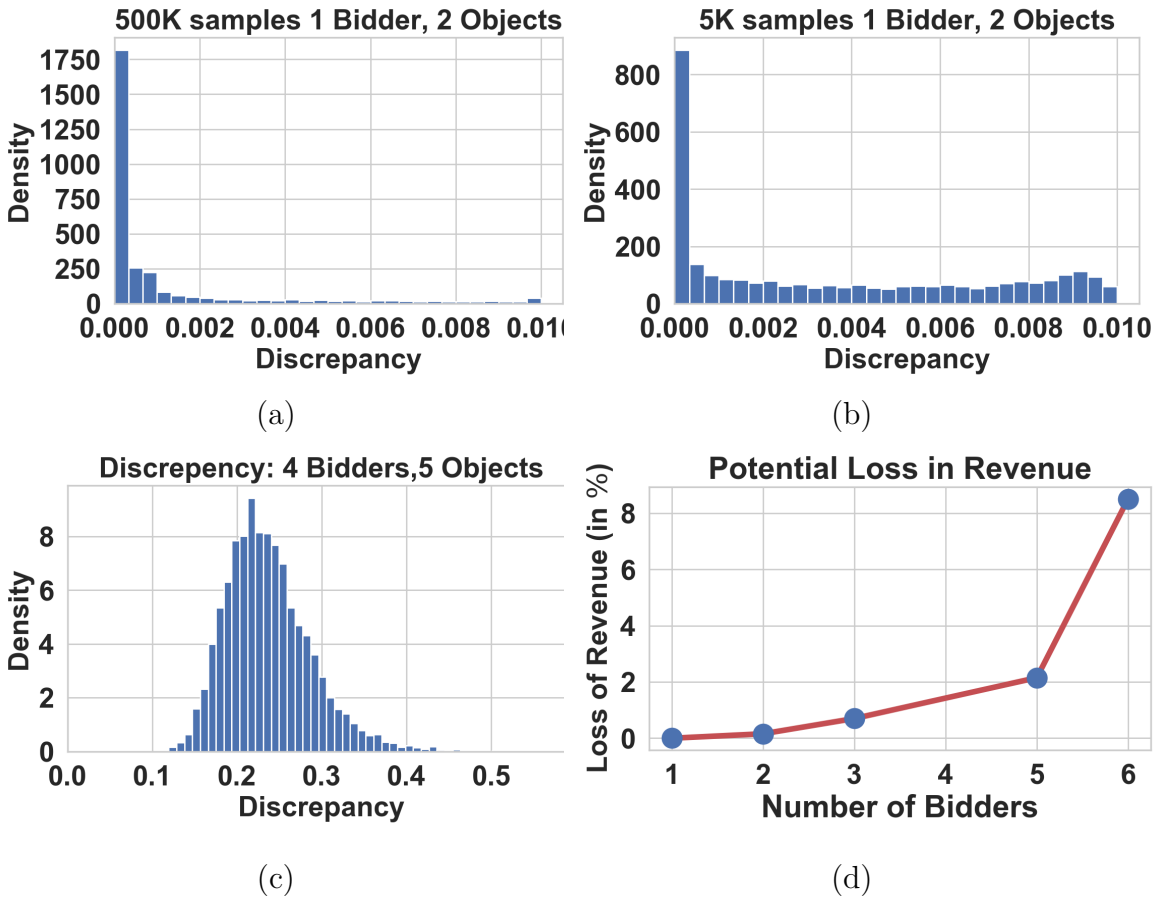


Figure 4.3.1: (a)-(b): Distribution h_R when varying the number of training samples (a) 500 000 (b) 5000 samples. (c): Histogram of the distribution h_R for setting (II). (d): Maximum revenue loss when varying the number of bidders for setting (III$_n$)
.

is almost concentrated at zero and therefore the network is almost able to recover the permutation-equivariant solution. When L is small, h_R is less concentrated around zero and therefore, the solution obtained is non permutation-equivariant.

As the problem's dimensions increase, this lack of permutation-invariance becomes more dramatic. Figure 4.3.1 (c) shows h_R for the optimal auction mechanism learned for setting (II) when trained with $5 \cdot 10^5$ samples. Contrary to (I), almost no entry of h_R is located around zero, they are concentrated around between 0.1 and 0.4 i.e. between 3.8% and 15% of the estimated optimal revenue.

Exploitability. Finally, to highlight how important equivariant solutions are, we analyze the worst-revenue loss that the auctioneer can incur when the bidders act adversarially. Indeed, since different permutations can result in different revenues for the auction, cooperative bidders could pick among the $n!$ possible permutations of their labels the one that minimized the revenue of the mechanism and present themselves in that order. Instead of getting a revenue of $R_{opt} = \mathbb{E}_{V\sim D}\left[\sum_{i=1}^{n} p_i(V)\right]$, the auctioneer would get a revenue of $R_{adv} = \mathbb{E}_{V\sim D}\left[\min_{\Pi_n}\{\sum_{i=1}^{n} p_i(\Pi_n V)\}\right]$. The percentage of revenue loss is given by $l = 100 \times \frac{R_{opt}-R_{adv}}{R_{opt}}$. We compute l in in the following family of settings:

– (III_n) n additive bidders and ten item where the item values are drawn from $\mathcal{U}[0, 1]$.

In Figure 4.3.1 (d) we plot $l(n)$ the loss in revenue as a function of n. As the number of bidders increases, the loss becomes more substantial getting over the 8% with only 6 bidders.

While it is unlikely that all the bidders will collide and exploit the bidding mechanism in real life, these investigations of permutation sensitivity and exploitability give us a sense of how far the solutions found by RegretNet are from being bidder-symmetric. The underlying real problem with non bidder-symmetric solution has to do with fairness. RegretNet finds mechanisms that do not treat all bidders equally. Their row number in the bid matrix matters, two bidders with the same bids will not get the same treatment. If the mechanism is equivariant however, all bidders will be treated equally by design, there are no biases or special treatments. Aiming for symmetric auctions is important and to this end, we design a permutation-equivariant architecture.

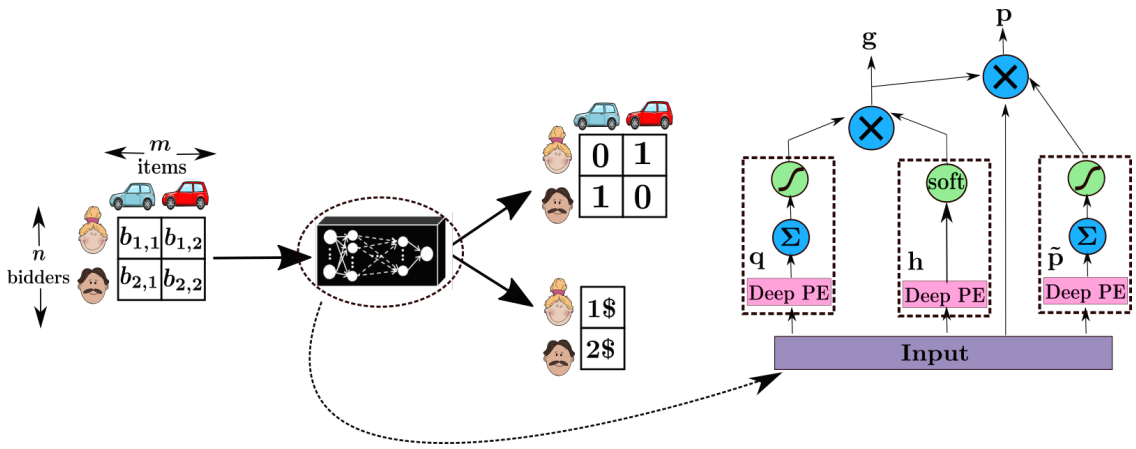


Figure 4.3.2: *Left:* Auction design setting. *Right:* EquivariantNet: Deep permutation-equivariant architecture for auction design. Deep PE denotes the deep permutation-equivariant architecture described in subsection 4.3.2, $\sum$ the sum over rows/columns operations, $\times$ the multiplication operations, soft stands for soft-max and the curve for sigmoid. The network outputs an allocation g and a payment p.

4.3.2 Architecture for symmetric auctions (EquivariantNet)

Our input is a bid matrix $B = (b_{i,j}) \in \mathbb{R}^{n\times m}$ drawn from a bidder-symmetric and item-symmetric distribution. We aim at learning a randomized allocation neural network $g^w : \mathbb{R}^{n\times m} \to [0,1]^{n\times m}$ and a payment network $p^w : \mathbb{R}^{n\times m} \to \mathbb{R}^n_{\geqslant 0}$. The symmetries of the distribution from which B is drawn and Theorem 1 motivates us to model g^w and p^w as permutation-equivariant functions. To this end, we use *exchangeable matrix layers* (Hartford et al., 2018) and their definition is reminded in Section 4.A. We now describe the three modules of the allocation and payment networks Figure 4.3.2.

The first network outputs a vector $q^w(B) \in [0,1]^m$ such that entry $q^w_j(B)$ is the probability that item j is allocated to any of the n bidders. The architecture consists of three modules. The first one is a deep permutation-equivariant network with tanh activation functions. The output of that module is a matrix $Q \in \mathbb{R}^{n\times m}$. The second module transforms Q into a vector $\mathbb{R}^m$ by taking the average over the rows of Q. We finally apply the sigmoid function to the result to ensure that $q^w(B) \in [0,1]^m$. This

architecture ensures that $q^w(B)$ is invariant with respect to bidder permutations and equivariant with respect to items permutations.

The second network outputs a matrix $h(B) \in [0,1]^{n\times m}$ where h^w_{ij} is the probability that item j is allocated to bidder i conditioned on item j being allocated. The architecture consists of a deep permutation-equivariant network with tanh activation functions followed by softmax activation function so that $\sum_{i=1}^{n} h^w_{ij}(B) = 1$. This architecture ensures that q^w equivariant with respect to object and bidder permutations.

By combining the outputs of q^w and h^w, we compute the allocation function $g^w : \mathbb{R}^{n\times m} \to [0,1]^{n\times m}$ where $g_{ij}(B)$ is the probability that the allocated item j is given to bidder i. Indeed, using conditional probabilities, we have $g^w_{ij}(B) = q^w_j(B)h^w_{ij}(B)$. Note that g^w is a permutation-equivariant function.

The third network outputs a vector $p(B) \in \mathbb{R}^n_{\geqslant 0}$ where $\tilde{p}^w_i$ is the fraction of bidder's i utility that she has to pay to the mechanism. Given the allocation function g^w, bidder i has to pay an amount $p_i = \tilde{p}_i(B) \sum_{j=1}^{m} g^w_{ij}(B)B_{ij}$. Individual rationality is ensured by having $\tilde{p}_i \in [0,1]$. The architecture of $\tilde{p}^w$ is almost similar to the one of q^w. Instead of averaging over the rows of the matrix output by the permutation-equivariant architecture, we average over the columns.

4.3.3 Optimization and training

The optimization and training procedure of EquivariantNet is similar to Duetting et al. (2019). For this reason, we briefly mention the outline of this procedure and remind the details in Section 4.C. We apply the augmented Lagrangian method to equation $\hat{R}$. The Lagrangian with a quadratic penalty is:

$$\mathcal{L}_\rho(w;\lambda) = -\frac{1}{L}\sum_{\ell=1}^{L}\sum_{i\in N} p_i^w(V^{(\ell)}) + \sum_{i\in N}\lambda_i \widehat{rgt}_i(w) + \frac{\rho}{2}\sum_{i\in N}\left(\widehat{rgt}_i(w)\right)^2,$$

where $\lambda \in \mathbb{R}^n$ is a vector of Lagrange multipliers and $\rho > 0$ is a fixed parameter controlling the weight of the quadratic penalty. The solver alternates between the updates on model parameters and Lagrange multipliers: $w^{new} \in \text{argmax}_w \mathcal{L}_\rho(w^{old}, \lambda^{old})$ and $\lambda_i^{new} = \lambda_i^{old} + \rho \cdot \widehat{rgt}_i(w^{new})$, $\forall i \in N$.

4.4 Experimental Results

We start by showing the effectiveness of our architecture in symmetric and asymmetric auctions. We then highlight its sample-efficiency for training and its ability to extrapolate to other settings. More details about the setup and training can be found in Section 4.C and Section 4.D.

Evaluation. In addition to the revenue of the learned auction on a test set, we also evaluate the corresponding empirical average regret over bidders $\widehat{rgt} = \frac{1}{n}\sum_{i=1}^{n}\widehat{rgt}_i$. We evaluate these terms by running gradient ascent on v_i' with a step-size of 0.001 for $\{300, 500\}$ iterations (we test $\{100, 300\}$ different random initial v_i' and report the one achieves the largest regret).

Known optimal solution. We first consider instances of single bidder multi-item auctions where the optimal mechanism is known to be symmetric. While independent private value auction as (I) fall in this category, the following item-asymmetric auction has surprisingly an optimal symmetric solution.

- (IV) One bidder and two items where the item values are independently drawn according to the probability densities $f_1(x) = 5/(1+x)^6$ and $f_2(y) = 6/(1+y)^7$. Optimal solution in Daskalakis et al. (2017).

Dist.	rev	rgt	OPT
(I)	0.551	0.00013	0.550
(IV)	0.173	0.00003	0.1706
(V)	0.873	0.001	0.860

(a)

	EquivariantNet		RegretNet	
λ_2	rev	rgt	rev_F	rgt_F
0.01	0.37	0.0006	0.39	0.0003
0.1	0.41	0.0004	0.41	0.0007
1	0.86	0.0005	0.84	0.0012
10	3.98	0.0081	3.96	0.0056

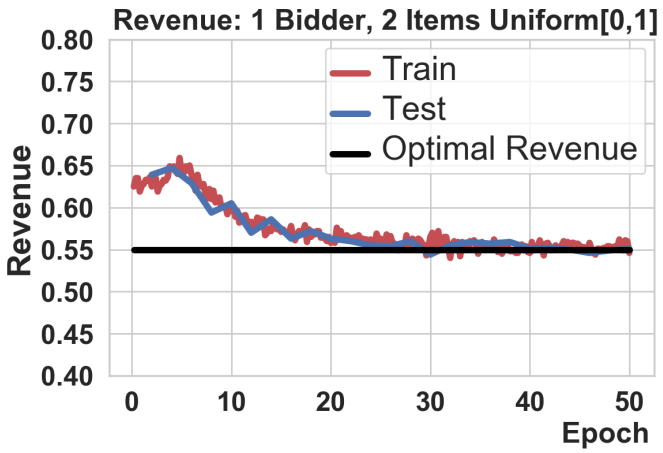
(b)

Figure 4.4.1: (a): Test revenue and regret found by EquivariantNet for settings (I), (IV) and (V). For seeting (V) OPT is the optimal revenue from VVCA and AMA_{bsym} families of auctions (Sandholm and Likhodedov, 2015). For settings (I) and (IV), OPT is the theoretical optimal revenue. (b): Test revenue/regret for setting (VI) when varying λ_2 ($\lambda_1 = 1$). rev_F and rgt_F are computed with RegretNet.

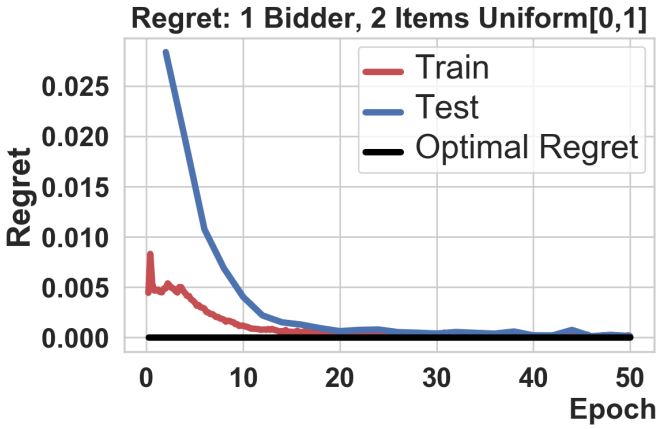
The two first lines in Figure 4.4.1(a) report the revenue and regret of the mechanism learned by our model. The revenue is very close to the optimal one, and the regret is negligible. Remark that the learned auction may achieve a revenue slightly above the optimal incentive compatible auction. This is possible because although small, the regret is non-zero. Figure 4.4.2(a)-(b) presents a plot of revenue and regret as a function of training epochs for the setting (I).

Unknown optimal solution. Our architecture is also able to recover a permutation-equivariant solution in settings for which the optimum is not known analytically such as:

– (V) Two additive bidders and two items where bidders draw their value for each item from $\mathcal{U}[0, 1]$.



(a) Train/test revenue as a function of epochs for setting (I) for EquivariantNet. The revenue converges to the theoretical optimum (0.55).



(b) Train/test regret as a function of epochs for setting (I) for EquivariantNet. The regret converges to 0.

Figure 4.4.2: EquivariantNet learn the optimal auction for the setting (I).

We compare our solution to the optimal auctions from the VVCA and AMA_{bsym} families of incentive compatible auctions from (Sandholm and Likhodedov, 2015). The last line of Figure 4.4.1(a) summarizes our results.

Non-symmetric optimal solution. Our architecture returns satisfactory results in asymmetric auctions. (VI) is a setting where there may not be permutation-equivariant solutions.

– (VI) Two bidders and two items where the item values are independently drawn according to the probability densities $f_1(x) = \lambda_1^{-1} e^{-\lambda_1 x}$ and $f_2(y) = \lambda_2^{-1} e^{-\lambda_2 y}$, where $\lambda_1, \lambda_2 > 0$.

Figure 4.4.1(b) shows the revenue and regret of the final auctions learned for setting (VI). When $\lambda_1 = \lambda_2$, the auction is symmetric and so, the revenue of the learned auction is very close to the optimal revenue, with negligibly small regret. However, as we increase the gap between λ_1 and λ_2, the asymmetry becomes dominant and the optimal auction does not satisfy permutation-equivariance. We remark that our architecture does output a solution with near-optimal revenue and small regret.

Sample-efficiency. Our permutation-equivariant architecture exhibits solid generalization properties when compared to the feed-forward architecture RegretNet. When enough data is available at training, both architectures generalize well to unseen data and the gap between the training and test losses goes to zero. However, when fewer training samples are available, our equivariant architecture generalizes while RegretNet struggles to. This may be explained by the inductive bias in our architecture.

We demonstrate this for auction (V) with a training set of 20 samples and plot the training and test losses as a function of time (measures in epochs) for both architectures in Figure 4.4.3(a).

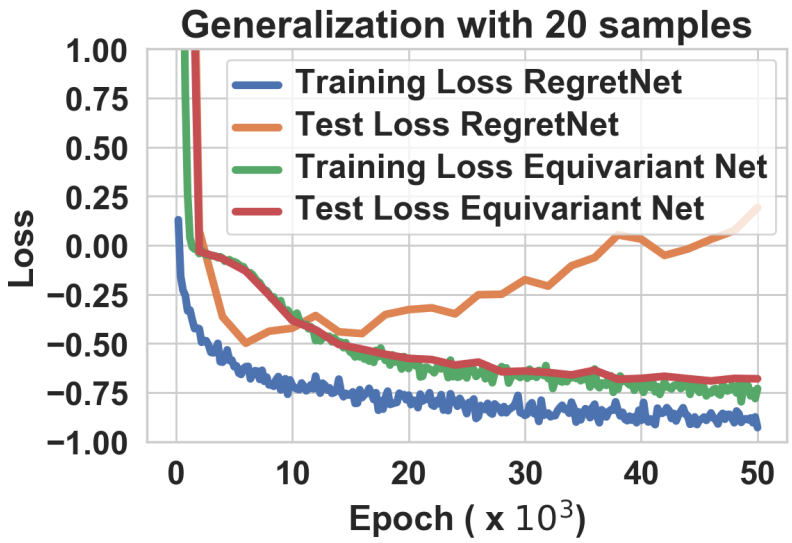


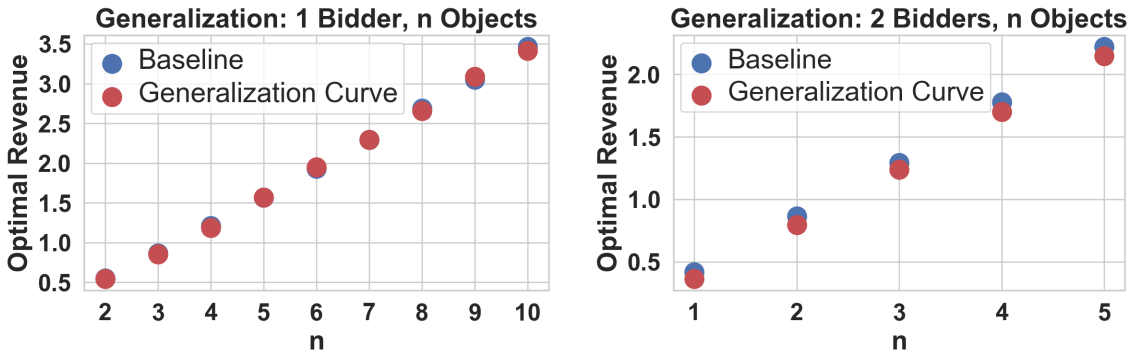
Figure 4.4.3: Train and test losses (the Lagrangian) for setting (V) with 20 training samples. RegretNet and EquivariantNet both achieve small losses on the training set, only EquivariantNet is generalizes to the testing set.

Out-of-setting generalization. The number of parameters in our permutation equivariant architecture does not depend on the size of the input. Given an architecture that was trained on samples of size (n, m), it is possible to evaluate it on samples of any size (n', m') (More details in Section 4.A). This evaluation is not well defined for feed-forward architectures where the dimension of the weights depends on the input size. We use this advantage to check whether models trained in a fixed setting perform well in totally different ones.

- (α) Train an equivariant architecture on 1 bidder, 5 items and test it on 1 bidder, n items for $n = 2 \cdots 10$. All the items values are sampled independently from $\mathcal{U}[0, 1]$.

- (β) Train an equivariant architecture on 2 bidders, 3 objects and test it on 2 bidders, n objects for $n = 2 \cdots 6$. All the items values are sampled independently from $\mathcal{U}[0, 1]$.

Figure 4.4.4(a)-(b) reports the test revenue that we get for different values of n in (α) and (β) and compares it to the empirical optimal revenue. Our baseline for that is RegretNet. Surprisingly, our model does generalize well. It is worth mentioning that knowing how to solve a larger problem such as 1×5 does not automatically result in a capacity to solve a smaller one such as 1×2; the generalization does happen on both ends. Our approach looks promising regarding out of setting generalization. It generalizes well when the number of objects varies and the number of bidders remain constants. However, generalization to settings where the number of bidders varies is more difficult due to the complex interactions between bidders. We do not observe good generalization with our current method.



(a) (b)

Figure 4.4.4: Generalization revenue of EquivariantNet in experiment (α) and (β). Each baseline point is computed using a RegretNet architecture trained from scratch.

Conclusion

We have explored the effect of adding domain knowledge in neural network architectures for auction design. We built a permutation-equivariant architecture to design symmetric auctions and highlighted its multiple advantages. It recovers several known optimal results and provides competitive results in asymmetric auctions. Compared to fully connected architectures, it is more sample efficient and is able to

generalize to settings it was not trained on. In a nutshell, this chapter insists on the importance of bringing domain-knowledge to the deep learning approaches for auctions.

Our architecture presents some limitations. It assumes that all the bidders and items are permutation-equivariant. However, in some real-world auctions, the item/bidder-symmetry only holds for a group of bidders/items. More advanced architectures such as Equivariant Graph Networks (Maron et al., 2018) may solve this issue. Another limitation is that we only consider additive valuations. An interesting direction would be to extend our approach to other settings as unit-demand or combinatorial auctions.

Appendix

4.A Permutation-equivariant network

In this section, we remind the *exchangeable matrix layers* introduced by Hartford et al. (2018). These layers are are a generalization of the deep sets architecture by Zaheer et al. (2017). We briefly describe this architecture here and invite the reader to look at the original paper for details.

The architecture consists in several layers and each of them is constituted of multiple channels. Each layer is specified by the number of input channels K channels and the number of outputs channels O. The input of such a layer is a tensor B of size (K, n, m) and the output is another tensor Y of size (O, n, m). The first element of these tensor is the channel number. In the following we will denote by $B_{i,j}^{(k)}$ the element of B of index (k, i, j) and similarly for $Y_{i,j}^{(o)}$.

In addition to the K and O, an exchangeable layer is defined by a set of five weights $w_1, w_2, w_3, w_4 \in \mathbb{R}^{K \times O}$ and $w_5 \in \mathbb{R}$. Given these weights, the element (i, j) of the o-th output channel $Y_{i,j}^{(o)}$ is given by:

$$Y_{i,j}^{(o)} = \sigma\Bigg(\sum_{k=1}^{K} w_1^{(k,o)} B_{i,j}^{(k)} + \frac{w_2^{(k,o)}}{n} \sum_{i'} B_{i',j}^{(k)} + \frac{w_3^{(k,o)}}{m} \sum_{j'} B_{i,j'}^{(k)} + \frac{w_4^{(k,o)}}{nm} \sum_{i',j'} B_{i',j'}^{(k)} + w_5^{(o)} \Bigg) \tag{4.1}$$

This layer preserved permutation-equivariance. This was first proven in Hartford et al. (2018). Additionally, the number of parameters of each layer only depends on K and O is does not depend on the dimension of the input (i.e. m and n). In particular, we can apply this exchangeable layer to any tensor of size (K, n', m') for any value of n' and m' and the resulting output will be a tensor of size (O, n', m').

We can compose these exchangeable layers as long as the number of channels of the output of one layer is equal to the number of input channels required by the following layer. By composing many such layer of this form we get a deep exchangeable neural network. This deep network preserved permutation-equivariance since this property is preserved by every layer. In addition, this network can be evaluated on an input of any dimension n and m. We use this property of the network to test our mechanisms on settings with different number of bidders and objects. Without this property out of setting generalization not possible.

4.B Proof of Theorem 1

Notation: For a matrix $B \in \mathbb{R}^{nm}$ we will denote the ith line by $B_i \in \mathbb{R}^m$ or $[B]_i \in \mathbb{R}^m$.Let D denote an equivariant distribution on $\mathbb{R}^{nm}$. Let $g : \mathbb{R}^{nm} \to \mathbb{R}^{nm}$ and $p : \mathbb{R}^{nm} \to \mathbb{R}^n$ be solutions to the following problem:

$$p = \text{argmax } \mathbb{E}_{B\sim D}\left[\sum_{i=1}^{n} p_i(B)\right]$$

subject to:

$$\langle\, [g(B)]_i\,,\, B_i\,\rangle \geqslant p_i(B),$$

and

$$\langle\, [g(B_i, B_{-i})]_i\,,\, B_i\,\rangle - p_i(B_i, B_{-i}) \geqslant \langle\, [g(B'_i, B_{-i})]_i\,,\, B_i\,\rangle - p_i(B'_i, B_{-i}), \quad \forall B'_i \in \mathbb{R}^m.$$

Let Π_n and Π_m be two permutation matrices of sizes n and m. In particular Π_n and Π_m are orthogonal matrices and in the following we use that $\Pi_n^{-1} = \Pi_n^T$ and $\Pi_m^{-1} = \Pi_m^T$. Let's define:

$$g^{\Pi_n,\Pi_m}(B) = \Pi_n^{-1}\, g(\Pi_n\, B\, \Pi_m)\, \Pi_m^{-1}$$
$$p^{\Pi_n,\Pi_m}(B) = \Pi_n^{-1}\, p(\Pi_n\, B\, \Pi_m).$$

Let's prove that if (g, p) is a solution to the problem then so is $(g^{\Pi_n,\Pi_m}, p^{\Pi_n,\Pi_m})$. First we show that $(g^{\Pi_n,\Pi_m}, p^{\Pi_n,\Pi_m})$ still satisfy the previous constraints.

$$\begin{aligned}
\langle [g^{\Pi_n,\Pi_m}(B)]_i\,,\, B_i\,\rangle &= \langle [\Pi_n^{-1} g(\Pi_n\, B\, \Pi_m)\Pi_m^{-1}]_i\,,\, B_i\,\rangle \\
&= \langle [\Pi_n^{-1} g(\Pi_n\, B\, \Pi_m)]_i \Pi_m^{-1}\,,\, B_i\,\rangle \\
&= \langle [\Pi_n^{-1} g(\Pi_n\, B\, \Pi_m)]_i\,,\, B_i \Pi_m\,\rangle \\
&= \langle [\Pi_n^{-1} g(\Pi_n\, B\, \Pi_m)]_i\,,\, [B\Pi_m]_i\,\rangle \\
&= \langle [\Pi_n^{-1} g(\Pi_n\, B\, \Pi_m)]_i\,,\, [B\Pi_m]_i\,\rangle.
\end{aligned}$$

Let's denote by φ the permutation on the indices corresponding to the Π_n permutation. then we have:

$$[\Pi_n^{-1} g(\Pi_n \, B \, \Pi_m)]_i = [g(\Pi_n \, B \, \Pi_m)]_{\varphi^{-1}(i)}$$
$$[B\Pi_m]_i = [\Pi_n B \Pi_m]_{\varphi^{-1}(i)}.$$

This gives us that:

$$\begin{aligned}
\langle [g^{\Pi_n,\Pi_m}(B)]_i \, , \, B_i \rangle &= \langle [\Pi_n^{-1} g(\Pi_n \, B \, \Pi_m)]_i \, , \, [B\Pi_m]_i \rangle \\
&= \langle [g(\Pi_n \, B \, \Pi_m)]_{\varphi^{-1}(i)} \, , \, [\Pi_n B \Pi_m]_{\varphi^{-1}(i)} \rangle \\
&\geqslant [p(\Pi_n B \Pi_m)]_{\varphi^{-1}(i)} \\
&= [\Pi_n^{-1} p(\Pi_n B \Pi_m)]_i \\
&= [p^{\Pi_n,\Pi_m}(B)]_i.
\end{aligned}$$

This shows that $(g^{\Pi_n,\Pi_m}, p^{\Pi_n,\Pi_m})$ satisfies the first constraint. We now move to the second constraint.

Let's write $\tilde{B} = (B'_i, B_{-i})$. As a reminder, this is the matrix B where the ith line has been replaced with B'_i. We need to show that:

$$\langle \, [g^{\Pi_n,\Pi_m}(B)]_i \, , \, B_i \, \rangle - p_i^{\Pi_n,\Pi_m}(B) \geqslant \langle \, [g^{\Pi_n,\Pi_m}(\tilde{B})]_i \, , \, B_i \, \rangle - p_i^{\Pi_n,\Pi_m}(\tilde{B}).$$

Using the previous computations we find that:

$$\langle \, [g^{\Pi_n,\Pi_m}(B)]_i \, , \, B_i \, \rangle - p_i^{\Pi_n,\Pi_m}(B) = \langle [g(\Pi_n \, B \, \Pi_m)]_{\varphi^{-1}(i)} \, , \, [\Pi_n B \Pi_m]_{\varphi^{-1}(i)} \rangle - [p(\Pi_n B \Pi_m)]_{\varphi^{-1}(i)},$$

where φ is the permutation associated with Π_n. Since g and p satisfy the second constraint we have:

$$\begin{aligned}
\langle\, [g^{\Pi_n,\Pi_m}(B)]_i\,,\, B_i\,\rangle - p_i^{\Pi_n,\Pi_m}(B) &= \langle [g(\Pi_n\, B\, \Pi_m)]_{\varphi^{-1}(i)}\,,\, [\Pi_n B \Pi_m]_{\varphi^{-1}(i)}\,\rangle - [p(\Pi_n B \Pi_m)]_{\varphi^{-1}(i)} \\
&\geqslant \langle [g(\Pi_n\, \tilde{B}\, \Pi_m)]_{\varphi^{-1}(i)}\,,\, [\Pi_n \tilde{B} \Pi_m]_{\varphi^{-1}(i)}\,\rangle - [p(\Pi_n \tilde{B} \Pi_m)]_{\varphi^{-1}(i)} \\
&= \langle\, [g^{\Pi_n,\Pi_m}(\tilde{B})]_i\,,\, B_i\,\rangle - p_i^{\Pi_n,\Pi_m}(\tilde{B}).
\end{aligned}$$

This concludes the proof that $(g^{\Pi_n,\Pi_m}, p^{\Pi_n,\Pi_m})$ satisfy the constraints. Now we have to show that p^{Π_n,Π_m} is optimal.

$$\begin{aligned}
\mathbb{E}_{B\sim D}\left[\sum_{i=1}^{n} p^{\Pi_n,\Pi_m}(B)\right] &= \mathbb{E}_{B\sim D}\left[\langle\, p^{\Pi_n,\Pi_m}(B)\,,\mathbf{1}\,\rangle\right] \\
&= \mathbb{E}_{B\sim D}\left[\langle\, \Pi_n^{-1}\, p(\Pi_n\, B\, \Pi_m)\,,\mathbf{1}\,\rangle\right] \\
&= \mathbb{E}_{B\sim D}\left[\langle\, p(\Pi_n\, B\, \Pi_m)\,,\mathbf{1}\,\rangle\right] \\
&= \mathbb{E}_{B\sim D}\left[\langle\, p(B)\,,\mathbf{1}\,\rangle\right] \\
&= \mathbb{E}_{B\sim D}\left[\sum_{i=1}^{n} p_i(B)\right],
\end{aligned}$$

where we used that $\Pi_n^{-1} = \Pi_n^T$, $\Pi_n \mathbf{1} = \mathbf{1}$ and that $\Pi_n\, B\, \Pi_m \sim D$ since D is an equivariant distribution. This shows that if p is optimal then p^{Π_n,Π_m} is also optimal since they have the same expectation. We conclude that $(g^{\Pi_n,\Pi_m}, p^{\Pi_n,\Pi_m})$ is an optimal solution. Let's define

$$\begin{aligned}
\tilde{g}(B) &= \mathbb{E}_{\Pi_n,\Pi_m}\left[g^{\Pi_n,\Pi_m}(B)\right] \\
\tilde{p}(B) &= \mathbb{E}_{\Pi_n,\Pi_m}\left[p^{\Pi_n,\Pi_m}(B)\right].
\end{aligned}$$

Here, in the expectation, Π_n and Π_m are drawn uniformly at random. Since the problem and constraints are convex, $(\tilde{g}, \tilde{p})$ is also an optimal solution to the problem as a convex combination of optimal solutions. Let's prove that $\tilde{g}$ and $\tilde{p}$ are equivariant functions.

$$\begin{aligned}\tilde{g}(\Pi_n B \Pi_m) &= \mathbb{E}_{\Pi'_n,\Pi'_m}\left[g^{\Pi'_n,\Pi'_m}(\Pi_n B \Pi_m)\right] \\ &= \mathbb{E}_{\Pi'_n,\Pi'_m}\left[{\Pi'_n}^{-1} g(\Pi'_m \Pi_n B \Pi_m \Pi'_m){\Pi'_m}^{-1}\right] \\ &= {\Pi_n}^{-1}\ \mathbb{E}_{\Pi'_n,\Pi'_m}\left[(\Pi'_n\Pi_n)^{-1} g(\Pi'_n \Pi_n B \Pi_m \Pi'_m)(\Pi_m\Pi'_m)^{-1}\right]\ {\Pi_m}^{-1}.\end{aligned}$$

If Π'_n and Π'_m are uniform among permutation then so is $\Pi'_n\Pi_n$ and $\Pi'_m\Pi_m$. So through a change of variable we find that:

$$\begin{aligned}\tilde{g}(\Pi_n B \Pi_m) &= {\Pi_n}^{-1}\ \mathbb{E}_{\Pi'_n,\Pi'_m}\left[{\Pi'_n}^{-1} g(\Pi'_n B \Pi'_m){\Pi'_m}^{-1}\right]\ {\Pi_m}^{-1} \\ &= {\Pi_n}^{-1}\ \tilde{g}(B)\ {\Pi_m}^{-1}.\end{aligned}$$

This shows that $\tilde{g}$ is equivariant. The proof that $\tilde{p}$ is equivariant is similar.

$$\begin{aligned}\tilde{p}(\Pi_n B \Pi_m) &= \mathbb{E}_{\Pi'_n,\Pi'_m}\left[p^{\Pi'_n,\Pi'_m}(\Pi_n B \Pi_m)\right] \\ &= \mathbb{E}_{\Pi'_n,\Pi'_m}\left[{\Pi'_n}^{-1} p(\Pi'_m \Pi_n B \Pi_m \Pi'_m)\right] \\ &= {\Pi_n}^{-1}\ \mathbb{E}_{\Pi'_n,\Pi'_m}\left[(\Pi'_n\Pi_n)^{-1} p(\Pi'_n \Pi_n B \Pi_m \Pi'_m)\right].\end{aligned}$$

By doing a change of variable as before we find:

$$\begin{aligned}\tilde{p}(\Pi_n B \Pi_m) &= {\Pi_n}^{-1}\ \mathbb{E}_{\Pi'_n,\Pi'_m}\left[{\Pi'_n}^{-1} p(\Pi'_n B \Pi'_m)\right] \\ &= {\Pi_n}^{-1}\ \tilde{p}(B),\end{aligned}$$

$(\tilde{g}, \tilde{p})$ is an equivariant optimal solution, this concludes the proof.

4.C Optimization and training procedures

Our training algorithm is the same as the one found in Duetting et al. (2019). We made that choice to better illustrate the intrinsic advantages of our permutation equivariant architecture. We include implementation details here for completeness and additional details can be found in the original paper.

Algorithm 4.1 Training Algorithm

Input: Minibatches $\mathcal{S}_1, \ldots, \mathcal{S}_T$ of size B
Parameters: $\gamma > 0$, $\eta > 0$, $c > 0$, $R \in \mathbb{N}$, $T \in \mathbb{N}$, $T_\rho \in \mathbb{N}$, $T_\lambda \in \mathbb{N}$.
Initialize Parameters: $\rho^0 \in \mathbb{R}$, $w^0 \in \mathbb{R}^d$, $\lambda^0 \in \mathbb{R}^n$,
Initialize Misreports: $v_i'^{(\ell)} \in V_i,\ \forall \ell \in [B],\ i \in N$.
for $t = 0, \ldots, T$ **do**
    Receive minibatch $\mathcal{S}_t = \{V^{(1)}, \ldots, V^{(B)}\}$.
    for $r = 0, \ldots, R$ **do**

$$\forall \ell \in [B],\ i \in n:$$
$$v_i'^{(\ell)} \leftarrow v_i'^{(\ell)} + \gamma \nabla_{v_i'} u_i^{w_t}(v_i{}^{(\ell)}; (v_i'^{(\ell)}, V_{-i}^{(\ell)}))$$

    Get Lagrangian gradient using equation 4.2 and update w^t:
        $w^{t+1} \leftarrow w^t - \eta \nabla_w \mathcal{L}_{\rho^t}(w^t)$.
    Update ρ once in T_ρ iterations:
    if t is a multiple of T_ρ **then**
        $\rho^{t+1} \leftarrow \rho^t + c$
    else
        $\rho^{t+1} \leftarrow \rho^t$
    Update Lagrange multipliers once in T_λ iterations:
    if t is a multiple of T_λ **then**
        $\lambda_i^{t+1} \leftarrow \lambda_i^t + \rho^t \widehat{rgt}_i(w^t), \forall i \in N$
    else
        $\lambda^{t+1} \leftarrow \lambda^t$

We generate a training dataset of valuation profiles $\mathcal{S}$ that we then divide into mini-batches of size B. Typical sizes for $\mathcal{S}$ are $\{5000, 50000, 500000\}$ and typical batch sizes are $\{50, 500, 50000\}$. We train our networks over for several epochs (typically $\{50, 80\}$) and we apply a random shuffling of the training data for each new epoch. We denote the minibatch received at iteration t by $\mathcal{S}_t = \{V^{(1)}, \ldots, V^{(B)}\}$. The update

on model parameters involves an unconstrained optimization of $\mathcal{L}_\rho$ over w and is performed using a gradient-based optimizer. Let $\widehat{rgt}_i(w)$ be the empirical regret in equation $\hat{R}$ computed on mini-batch $\mathcal{S}_t$. The gradient of $\mathcal{L}_\rho$ with respect to w is given by:

$$\begin{aligned}\nabla_w \mathcal{L}_\rho(w) = -\frac{1}{B}\sum_{\ell=1}^{B}\sum_{i\in N}\nabla_w p_i^w(V^{(\ell)}) \\ + \sum_{i\in N}\sum_{\ell=1}^{B}\lambda_i^t g_{\ell,i} + \rho_t \sum_{i\in N}\sum_{\ell=1}^{B}\widehat{rgt}_i(w) g_{\ell,i},\end{aligned} \tag{4.2}$$

where

$$g_{\ell,i} = \nabla_w \left[\max_{v_i'\in V_i} u_i^w(v_i^{(\ell)}; (v_i', V_{-i}^{(\ell)})) - u_i^w(v_i^{(\ell)}; (v_i^{(\ell)}, V_{-i}^{(\ell)}))\right].$$

The terms $\widehat{rgt}_i$ and $g_{\ell,i}$ requires us to compute the maximum over misreports for each bidder i and valuation profile ℓ. To compute this maximum we optimize the function $v_i' \to u_i^w(v_i^{(\ell)}; (v_i', V_{-i}^{(\ell)}))$ using another gradient based optimizer.

For each i and valuation profile ℓ, we maintain a misreports valuation $v_i'^{(\ell)}$. For every update on the model parameters w^t, we perform R gradient updates to compute the optimal misreports: $v_i'^{(\ell)} = v_i'^{(\ell)} + \gamma \nabla_{v_i'^{(\ell)}} u_i^w(v_i^{(\ell)}; (v_i'^{(\ell)}, V_{-i}^{(\ell)}))$, for some $\gamma > 0$. In our experiments, we use the Adam optimizer (Kingma and Ba, 2014) for updates on model w and $v_i'^{(\ell)}$. Typical values are $R = 25$ and $\gamma = 0.001$ for the training phase. During testing, we use a larger number of step sizes R_{test} to compute these optimal misreports and we try bigger number initialization, N_{init}, that are drawn from the same distribution of the valuations. Typical values are $R_{test} = \{200, 300\}$ and $N_{init} = \{100, 300\}$. When the valuations are constrained to an interval (for instance $[0, 1]$), this optimization inner loop becomes constrained and we make sure that the values we get for v_i' are realistic by projecting them to their domain after each

gradient step.

The parameters λ^t and ρ_t in the Lagrangian are not constant but they are updated over time. ρ_t is initialized at a value ρ_0 is incremented every T_ρ iterations, $\rho_{t+1} \leftarrow \rho_t + c$. Typical values are $\rho_0 = \{0.25, 1\}$, $c = \{0.25, 1, 5\}$ and $T_\rho = \{2, 5\}$ epochs. λ_t is initialized at a value λ_0 is updates every T_λ iterations according to $\lambda_i^{t+1} \leftarrow \lambda_i^t + \rho_t \widehat{rgt}_i(w^t), \forall i \in N$. Typical values are $\lambda_i^0 = \{0.25, 1, 5\}$ and $T_\lambda = \{2\}$ iterations.

4.D Setup

We implemented our experiments using PyTorch. A typical deep exchangeable network consists of 3 hidden layers of 25 channels each. Depending on the experiment, we generated a dataset of $\{5000, 50000, 500000\}$ valuation profiles and chose mini batches of sizes $\{50, 500, 5000\}$ for training. The optimization of the augmented Lagrangian was typically run for $\{50, 80\}$ epochs. The value of ρ in the augmented Lagrangian was set to 1.0 and incremented every 2 epochs. An update on w^t was performed for every mini-batch using the Adam optimizer with a learning rate of 0.001. For each update w^t, we ran $R = 25$ misreport update steps with a learning rate of 0.001. An update on λ^t was performed once every 100 minibatches.

Chapter 5

Auction Learning as a Two Player Game

Designing an incentive compatible auction that maximizes expected revenue is a central problem in Auction Design. While theoretical approaches to the problem have hit some limits, a recent research direction initiated by Duetting et al. (2019) consists in building neural network architectures to find optimal auctions. We propose two conceptual deviations from their approach which result in enhanced performance. First, we use recent results in theoretical auction design to introduce a *time-independent* Lagrangian. This not only circumvents the need for an expensive hyper-parameter search (as in prior work), but also provides a single metric to compare the performance of two auctions (absent from prior work). Second,the optimization procedure in previous work uses an inner maximization loop to compute optimal misreports. We amortize this process through the introduction of an additional neural network. We demonstrate the effectiveness of our approach by learning competitive or strictly improved auctions compared to prior work. Both results together further imply a novel formulation of Auction Design as a two-player game with stationary utility functions.

5.1 Introduction

Efficiently designing truthful auctions is a core problem in Mathematical Economics. Concrete examples include the sponsored search auctions conducted by companies as Google or auctions run on platforms as eBay. Following seminal work of Vickrey (Vickrey, 1961) and Myerson (Myerson, 1981), auctions are typically studied in the *independent private valuations* model: each bidder has a valuation function over items, and their payoff depends only on the items they receive. Moreover, the auctioneer knows aggregate information about the population that each bidder comes from, modeled as a distribution over valuation functions, but does not know precisely each bidder's valuation (outside of any information in this Bayesian prior). A major difficulty in designing auctions is that valuations are private and bidders need to be incentivized to report their valuations truthfully. The goal of the auctioneer is to design an incentive compatible auction which maximizes expected revenue.

Auction Design has existed as a rigorous mathematical field for several decades and yet, complete characterizations of the optimal auction only exist for a few settings. While Myerson's Nobel prize-winning work provides a clean characterization of the single-item optimum (Myerson, 1981), optimal *multi-item* auctions provably suffer from numerous formal measures of intractability (including computational intractability, high description complexity, non-monotonicity, and others) (Daskalakis et al., 2014b; Chen et al., 2014, 2015, 2018; Hart and Reny, 2015; Thanassoulis, 2004).

An orthogonal line of work instead develops deep learning architectures to find the optimal auction. Duetting et al. (2019) initiated this direction by proposing RegretNet, a feed-forward architecture. They frame the auction design problem as a constrained learning problem and lift the constraints into the objective via the augmented Lagrangian method. Training RegretNet involves optimizing this Lagrangian-penalized objective, while simultaneously updating network parameters and the Lagrangian multipliers themselves. This architecture produces impressive

results: recovering near-optimal auctions in several known multi-item settings, and discovering new mechanisms when a theoretical optimum is unknown.

Yet, this approach presents several limitations. On the conceptual front, our main insight is a connection to an exciting line of recent works (Hartline and Lucier, 2010; Hartline et al., 2011; Bei and Huang, 2011; Daskalakis and Weinberg, 2012; Rubinstein and Weinberg, 2018; Dughmi et al., 2017; Cai et al., 2019) on ε-truthful-to-truthful reductions.[1] On the technical front, we identify three areas for improvement. First, their architecture is difficult to train in practice as the objective is non-stationary. Specifically, the Lagrangian multipliers are time-dependent and they increase following a pre-defined schedule, which requires careful hyperparameter tuning (see subsection 5.3.1 for experiments illustrating this). Leveraging the aforementioned works in Auction Theory, we propose a *stationary* Lagrangian objective. Second, all prior work inevitably finds auctions which are not *precisely* incentive compatible, and does not provide a metric to compare, say, an auction with revenue 1.01 which is 0.002-truthful, or one with revenue 1 which is 0.001-truthful. We argue that our stationary Lagrangian objective serves as a good metric (and that the second auction of our short example is "better" for our metric). Finally, their training procedure requires an inner-loop optimization (essentially, this inner loop is the bidders trying to maximize utility in the current auction), which is itself computationally expensive. We use amortized optimization to make this process more efficient.

Contributions

This chapter leverages recent work in Auction Theory to formulate the learning of revenue-optimal auctions as a two-player game. We develop a new algorithm ALGnet (Auction Learning Game network) that produces competitive or better results compared

[1] By ε-truthful, we mean the expected total regret $\boldsymbol{R}$ is bounded by ε. See Proposition 5.3.1 for a definition of $\boldsymbol{R}$.

to Duetting et al. (2019)'s RegretNet. In addition to the conceptual contributions, our approach yields the following improvements (as RegretNet is already learning near-optimal auctions, our improvement over RegretNet is not due to significantly higher optimal revenues).

- *Easier hyper-parameter tuning*: By constructing a time-independent loss function, we circumvent the need to search for an adequate parameter scheduling. Our formulation also involves less hyperparameters, which makes it more robust.

- *A metric to compare auctions*: We propose a metric to compare the quality of two auctions which are not incentive compatible.

- *More efficient training*: We replace the inner-loop optimization of prior work with a neural network, which makes training more efficient.

- *Online auctions*: Since the learning formulation is time-invariant, ALGnet is able to quickly adapt in auctions where the bidders' valuation distributions varies over time. Such setting appears for instance in the online posted pricing problem studied in Bubeck et al. (2017).

Furthermore, these technical contributions together now imply a novel formulation of auction learning as a two-player game (not zero-sum) between an auctioneer and a misreporter. The auctioneer is trying to design an incentive compatible auction that maximizes revenue while the misreporter is trying to identify breaches in the truthfulness of these auctions.

The chapter decomposes as follows. Section 5.2 introduces the standard notions of auction design. Section 5.3 presents our game formulation for auction learning. Section 5.4 provides a description of ALGnet and its training procedure. Finally, Section 5.5 presents numerical evidence for the effectiveness of our approach.

Related work

Auction design and machine learning. Machine learning and computational learning theory have been used in several ways to design auctions from samples of bidder valuations. Machine learning has been used to analyze the sample complexity of designing optimal revenue-maximizing auctions. This includes the framework of single-parameter settings(Morgenstern and Roughgarden, 2015; Huang et al., 2018; Hartline and Taggart, 2019; Roughgarden and Schrijvers, 2016; Gonczarowski and Nisan, 2017; Guo et al., 2019), multi-item auctions (Dughmi et al., 2014; Gonczarowski and Weinberg, 2018), combinatorial auctions (Balcan et al., 2016; Morgenstern and Roughgarden, 2016; Syrgkanis, 2017) and allocation mechanisms (Narasimhan and Parkes, 2016). Other works have leveraged machine learning to optimize different aspects of mechanisms (Lahaie, 2011; Dütting et al., 2015). Our approach is different as we build a deep learning architecture for auction design.

Auction design and deep learning. While Duetting et al. (2019) is the first paper to design auctions through deep learning, several other paper followed-up this work. Feng et al. (2018) extended it to budget constrained bidders, Golowich et al. (2018) to the facility location problem. Tacchetti et al. (2019) built architectures based on the Vickrey- Clarke-Groves mechanism. Rahme et al. (2021b) used permutation-equivariant networks to design symmetric auctions. Shen et al. (2019) and Duetting et al. (2019) proposed architectures that *exactly* satisfy incentive compatibility but are specific to *single-bidder* settings. While all the previously mentioned papers consider a non-stationary objective function, we formulate a time-invariant objective that is easier to train and that makes comparisons between mechanisms possible.

5.2 Auction design as a time-varying learning problem

We first review the framework of auction design and the problem of finding truthful mechanisms. We then recall the learning problem proposed by Duetting et al. (2019) to find optimal auctions.

5.2.1 Auction design and linear program

Auction design. We consider an auction with n bidders and m items. We will denote by $N = \{1, \dots, n\}$ and $M = \{1, \dots, m\}$ the set of bidders and items. Each bidder i values item j at a valuation denoted v_{ij}. We will focus on *additive* auctions. These are auctions where the value of a set S of items is equal to the sum of the values of the elements in that set at $\sum_{j \in S} v_{ij}$. Additive auctions are perhaps the most well-studied setting in multi-item auction design (Hart and Nisan, 2012; Li and Yao, 2013; Daskalakis et al., 2014b; Cai et al., 2016; Daskalakis et al., 2017).

The auctioneer does not know the exact valuation profile $V = (v_{ij})_{i \in N, j \in M}$ of the bidders in advance but he does know the distribution from which they are drawn: the valuation vector of bidder i, $\vec{v}_i = (v_{i1}, \dots, v_{im})$ is drawn from a distribution D_i over $\mathbb{R}^m$. We will further assume that all bidders are independent and that $D_1 = \cdots = D_n$. As a result V is drawn from $D := \otimes_{i=1}^{n} D_i = D_1^{\otimes^n}$.

Definition 12. *An auction is defined by a randomized allocation rule $g = (g_1, \dots, g_n)$ and a payment rule $p = (p_1, \dots, p_n)$ where $g_i \colon \mathbb{R}^{n \times m} \to [0,1]^m$ and $p_i \colon \mathbb{R}^{n \times m} \to \mathbb{R}_{\geqslant 0}$. Additionally for all items j and valuation profiles V, the g_i must satisfy $\sum_i [g_i(V)]_j \leqslant 1$.*

Given a bid matrix $B = (b_{ij})_{i \in N, j \in M}$, $[g_i(B)]_j$ is the probability that bidder i receives object j and $p_i(B)$ is the price bidder i has to pay to the auction. The condition $\sum_i [g_i(V)]_j \leqslant 1$ allows the possibility for an item to be not allocated.

Definition 13. *The utility of bidder i is defined by* $u_i(\vec{v}_i, B) = \sum_{j=1}^{m} [g_i(B)]_j v_{ij} - p_i(B)$.

Bidders seek to maximize their utility and may report bids that are different from their true valuations. In the following, we will denote by B_{-i} the $(n-1)\times m$ bid matrix without bidder i, and by $(\vec{b}_i', B_{-i})$ the $n \times m$ bid matrix that inserts $\vec{b}_i'$ into row i of B_{-i} (for example: $B := (\vec{b}_i, B_{-i})$. We aim at auctions that incentivize bidders to bid their true valuations.

Definition 14. *An auction (g, p) is dominant strategy incentive compatible (DSIC) if each bidder's utility is maximized by reporting truthfully no matter what the other bidders report. For every bidder i, valuation $\vec{v}_i \in D_i$, bid $\vec{b}_i{}' \in D_i$ and bids $B_{-i} \in D_{-i}$,* $u_i(\vec{v}_i, (\vec{v}_i, B_{-i})) \geqslant u_i(\vec{v}_i, (\vec{b}_i{}', B_{-i}))$.

Definition 15. *An auction is individually rational (IR) if for all $i \in N$, $\vec{v}_i \in D_i$ and $B_{-i} \in D_{-i}$,*

$$u_i(\vec{v}_i, (\vec{v}_i, B_{-i})) \geqslant 0. \tag{IR}$$

In a DSIC auction, the bidders have the incentive to truthfully report their valuations and therefore, the revenue on valuation profile V is $\sum_{i=1}^{n} p_i(V)$. Optimal auction design aims at finding a DSIC and IR auction that maximizes the expected revenue $rev := \mathbb{E}_{V\sim D}[\sum_{i=1}^{n} p_i(V)]$. Since there is no known characterization of DSIC mechanisms in the multi-item setting, we resort to the relaxed notion of *ex-post regret*. It measures the extent to which an auction violates DSIC.

Definition 16. *The ex-post regret for a bidder i is the maximum increase in his utility when considering all his possible bids and fixing the bids of others. For a valuation profile V, it is given by* $r_i(V) = \max_{\vec{b}_i{}' \in \mathbb{R}^m} u_i(\vec{v}_i, (\vec{b}_i{}', V_{-i})) - u_i(\vec{v}_i, (\vec{v}_i, V_{-i}))$. *In particular, DSIC is equivalent to*

$$r_i(V) = 0, \ \forall i \in N, \forall V \in D. \tag{IC}$$

The bid $\vec{b}'_i$ that achieves $r_i(V)$ is called the optimal misreport of bidder i for valuation profile V.

Therefore, finding an optimal auction is equivalent to the following linear program:

$$\min_{(g,p)\in\mathcal{M}} -\mathbb{E}_{V\sim D}\left[\sum_{i=1}^{n} p_i(V)\right] \quad \text{s.t.} \quad \begin{aligned} & r_i(V) = 0, && \forall\ i \in N,\ \forall\ V \in D, \\ & u_i(\vec{v}_i, (\vec{v}_i, B_{-i})) \geqslant 0,\ \forall i \in N,\ \vec{v}_i \in D_i, B_{-i} \in D_{-i}. \end{aligned} \tag{LP}$$

5.2.2 Auction design as a learning problem

As the space of auctions $\mathcal{M}$ may be large, we will set a parametric model. In what follows, we consider the class of auctions (g^w, p^w) encoded by a neural network of parameter $w \in \mathbb{R}^d$. The corresponding utility and regret function will be denoted by u_i^w and r_i^w.

Following Duetting et al. (2019), the formulation equation LP is relaxed: the IC constraint for all $V \in D$ is replaced by the expected constraint $\mathbb{E}_{V\sim D}[r_i^w(V)] = 0$ for all $i \in N$. The justification for this relaxation can be found in Duetting et al. (2019). By replacing expectations with empirical averages, the learning problem becomes:

$$\min_{w\in\mathbb{R}^d} -\frac{1}{L}\sum_{\ell=1}^{L}\sum_{i=1}^{n} p_i^w(V^{(\ell)}) \quad \text{s.t.} \quad \widehat{r}_i^w := \frac{1}{L}\sum_{\ell=1}^{L} r_i^w(V^{(\ell)}) = 0,\ \forall i \in N. \tag{$\widehat{\text{LP}}$}$$

The learning problem equation $\widehat{\text{LP}}$ does not ensure equation IR. However, this constraint is usually built into the parametrization (architecture) of the model: by design, the only auction mechanism considered satisfy equation IR.

Implementation details can be found in Duetting et al. (2019); Rahme et al. (2021b) or in Sec 5.4.

5.3 Auction learning as a two-player game

We first present the optimization and the training procedures for equation $\widehat{\text{LP}}$ proposed by Duetting et al. (2019). We then demonstrate with numerical evidence that this approach presents two limitations: hyperparameter sensitivity and lack of interpretability. Using the concept of ε-truthful to truthful reductions, we construct a new loss function that circumvents these two aspects. Lastly, we resort to amortized optimization and reframe the auction learning problem as a two-player game.

5.3.1 The augmented Lagrangian method and its shortcomings

Optimization and training. We briefly review the training procedure proposed by Duetting et al. (2019) to learn optimal auctions. The authors apply the augmented Lagrangian method to solve the constrained problem equation $\widehat{\text{LP}}$ and consider the loss:

$$\mathcal{L}(w;\lambda;\rho) = -\frac{1}{L}\sum_{\ell=1}^{L}\sum_{i\in N} p_i^w(V^{(\ell)}) + \sum_{i\in N}\lambda_i r_i^w(V^{(\ell)}) + \frac{\rho}{2}\sum_{i\in N}\left(r_i^w(V^{(\ell)})\right)^2,$$

where $\lambda \in \mathbb{R}^n$ is a vector of Lagrange multipliers and $\rho > 0$ is a parameter controlling the weight of the quadratic penalty. More details about the training procedure can be found in Appendix A.

Scheduling consistency problem. The parameters λ and ρ are time-varying. Indeed, their value changes according to a pre-defined scheduling of the following form: 1) Initialize λ and ρ with respectively λ^0 and ρ^0, 2) Update ρ every T_ρ iterations : $\rho^{t+1} \leftarrow \rho^t + c$, where c is a pre-defined constant, 3) Update λ every T_λ iterations according to $\lambda_i^t \leftarrow \lambda_i^t + \rho^t \widehat{r}_i^{w^t}$.

Therefore, this scheduling requires to set up five hyper parameters $(\lambda^0, \rho^0, c, T_\lambda, T_\rho)$. Some of the experiments found Duetting et al. (2019) were about learning an optimal

mechanism for an n-bidder m-item auction ($n \times m$) where the valuations are iid $\mathcal{U}[0,1]$. Different scheduling parameters were used for different values of n and m. We report the values of the hyper parameters used for the 1×2, 3×10 and 5×10 settings in Table 5.3.1(a). A natural question is whether the choice of parameters heavily affects the performance. We proceed to a numerical investigation of this questions by trying different schedulings (columns) for different settings (rows) and report our the results in Table 5.3.1(b).

Table 5.3.1: (a): Scheduling parameters values set in Duetting et al. (2019) to reach optimal auctions in $n \times m$ settings with n bidders, m objects and i.i.d. valuations sampled from $\mathcal{U}[0,1]$. (b): Revenue $rev := \mathbb{E}_{V \sim D}[\sum_{i=1}^{n} p_i(V)]$ and average regret per bidder $reg := 1/n\, \mathbb{E}_{V \in D}\left[\sum_{i=1}^{n} r_i(V)\right]$ for $n \times m$ settings when using the different parameters values set reported in (a).

	1×2	3×10	5×10
λ^0	5	5	1
ρ^0	1	1	0.25
c	50	1	0.25
T_λ	10^2	10^2	10^2
T_ρ	10^4	10^4	10^5

(a)

	Scheduling					
	1×2		3×10		5×10	
Setting	*rev*	*rgt*	*rev*	*rgt*	*rev*	*rgt*
1×2	0.552	0.0001	0.573	0.0012	0.332	0.0179
3×10	4.825	0.0007	5.527	0.0017	5.880	0.0047
5×10	4.768	0.0006	5.424	0.0033	6.749	0.0047

(b)

The auction returned by the network dramatically varies with the choice of scheduling parameters. When applying the parameters of 1×2 to 5×10, we obtain a revenue that is lower by 30%! The performance of the learning algorithm strongly depends on the specific values of the hyperparameters. Finding an adequate scheduling requires an extensive and time consuming hyperparameter search.

Lack of interpretability. How should one compare two mechanisms with different expected revenue and regret? Is a mechanism M_1 with revenue $P_1 = 1.01$ and an

average total regret $R_1 = 0.02$ better than a mechanism M_2 with $P_2 = 1.0$ and $R_2 = 0.01$? The approach in Duetting et al. (2019) cannot answer this question. To see that, notice that when $\lambda_1 = \cdots = \lambda_n = \lambda$ we can rewrite $\mathcal{L}(w; \lambda; \rho) = -P + \lambda R + \frac{\rho}{2} R^2$. Which mechanism is better depends on the values of λ and ρ. For example if $\rho = 1$ and $\lambda = 0.1$ we find that M_1 is better, but if $\rho = 1$ and $\lambda = 10$ then M_2 is better. Since the values of λ and ρ change with time, the Lagrangian approach in Duetting et al. (2019) cannot provide metric to compare two mechanisms.

5.3.2 A time-independent and interpretable loss function for auction learning

Our first contribution consists in introducing a new loss function for auction learning that addresses the two first limitations of Duetting et al. (2019) mentioned in Section 5.3.1. We first motivate this loss in the one bidder case and then extend it to auctions with many bidders.

Mechanisms with one bidder

Proposition 5.3.1. *[Balcan et al. (2005), attributed to Nisan] Let $\mathcal{M}$ be an additive auction with 1 bidder and m items. Let P and R denote the expected revenue and regret, $P = \mathbb{E}_{V \in D}[p(V)]$ and $R = \mathbb{E}_{V \in D}[r(V)]$. There exists a mechanism $\mathcal{M}^*$ with expected revenue $P^* = (\sqrt{P} - \sqrt{R})^2$ and zero regret $R^* = 0$.*

A proof of this proposition can be found in Appendix C. Comparing two mechanisms is straightforward when both of them have zero-regret: the best one achieves the highest revenue. Proposition 5.3.1 allows a natural and simple extension of this criteria for non zero-regret mechanism with one bidder: we will say that M_1 is better than M_2 if and only if M_1^* is better than M_2^*:

$$M_1 \succcurlyeq M_2 \iff P^*(M_1) \geqslant P^*(M_2) \iff \sqrt{P_1} - \sqrt{R_1} \geqslant \sqrt{P_2} - \sqrt{R_2} \tag{5.1}$$

Using our metric, we find that a one bidder mechanism with revenue of 1.00 and regret of 0.01 is "better" than one with revenue 1.01 and regret 0.02.

Mechanisms with multiple bidders

Let M_1 and M_2 be two mechanisms with n bidders and m objects. Let P_i and R_i denote their total expected revenue and regret, $P_i = \mathbb{E}_{V \in D}\left[\sum_{j=1}^{n} p_j(V)\right]$ and $R_i = \mathbb{E}_{V \in D}\left[\sum_{j=1}^{n} r_j(V)\right]$. We can extend our metric derived in Section 5.3.2 to the multiple bidder by the following:

$$M_1 \text{ is "better" than } M_2 \iff M_1 \succcurlyeq M_2 \iff \sqrt{P_1} - \sqrt{R_1} \geqslant \sqrt{P_2} - \sqrt{R_2} \tag{5.2}$$

When $n = 1$ we recover the criteria from Section 5.3.2 that is backed by Proposition 5.3.1. When $n > 1$, it is considered a major open problem whether the extension of Proposition 5.3.1 still holds. Note that a multi-bidder variant of Proposition 5.3.1 *does* hold under a different solution concept termed "Bayesian Incentive Compatible" (Rubinstein and Weinberg, 2018; Cai et al., 2019), supporting the conjecture that Proposition 5.3.1 indeed extends.[2] Independently of whether or not Proposition 5.3.1 holds, this reasoning implies a candidate loss function for the multi-bidder setting which we can evaluate empirically.

This way of comparing mechanisms motivates the use of loss function: $\mathcal{L}(P, R) = -(\sqrt{P} - \sqrt{R})$ instead of the Lagrangian from Section 5.3, and indeed this loss function

[2]An auction is *Bayesian Incentive Compatible* if every bidder maximizes their expected utility by truthful reporting *in expectation over the other bidders' truthful bids.* Compare this to Dominant Strategy Incentive Compatible (our work), where every bidder maximizes their expected utility by truthful reporting *for all realizations of the other bidders' bids.*

works well in practice. We empirically find the loss function $\mathcal{L}_m(P, R) = -(\sqrt{P} - \sqrt{R}) + R$ further accelerates training, as it further (slightly) biases towards mechanisms with low regret. Both of these loss function are time-independent and hyperparameter-free.

5.3.3 Amortized misreport optimization

To compute the regret $r_i^w(V)$ one has to solve the optimization problem:

$$\max_{\vec{v}_i{}' \in \mathbb{R}^m} u_i^w(\vec{v}_i, (\vec{v}_i{}', V_{-i})) - u_i^w(\vec{v}_i, (\vec{v}_i, V_{-i}))$$

. In Duetting et al. (2019), this optimization problem is solved with an inner optimization loop for each valuation profile. In other words, computing the regret of each valuation profile is solved separately and independently, from scratch.

If two valuation profiles are very close to each other, one should expect that the resulting optimization problems to have close results. We leverage this to improve training efficiency.

We propose to amortize this inner loop optimization. Instead of solving all these optimization problems independently, we will instead learn one neural network M^φ that tries to predict the solution of all of them. M^φ takes as entry a valuation profile and maps it to the optimal misreport:

$$M^\varphi : \begin{cases} \mathbb{R}^{n \times m} & \to \mathbb{R}^{n \times m} \\ V = [\vec{v}_i]_{i \in N} & \to [\text{argmax}_{\vec{v}' \in D} u_i(\vec{v}_i, (\vec{v}', V_{-i}))]_{i \in N} \end{cases} \tag{5.3}$$

The loss $\mathcal{L}_r$ that M^φ is trying to minimize follows naturally from that definition and is then given by: $\mathcal{L}_r(\varphi, w) = -\mathbb{E}_{V \in D}\left[\sum_{i=1}^n u_i^w(\vec{v}_i, ([M^\varphi(V)]_i, V_{-i}))\right]$.

5.3.4 Auction learning as a two-player game

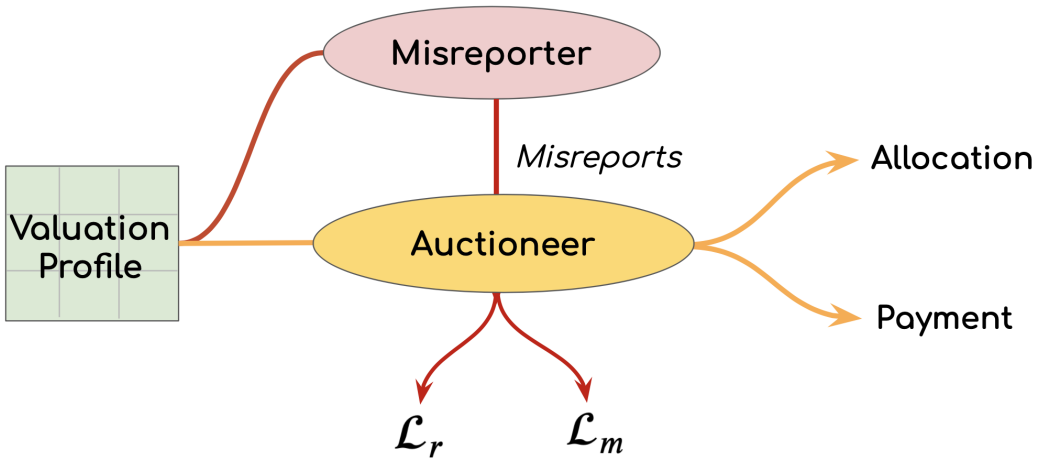
In this section, we combine the ideas from Sections 5.3.2 and 5.3.3 to obtain a new formulation for the auction learning problem as a two-player game between an Auctioneer with parameter w and a Misreporter with parameter φ. The optimal parameters for the auction learning problem (w^*, φ^*) are a Nash Equilibrium for this game.

The Auctioneer is trying to design a truthful equation IC and rational equation IR auction that maximizes revenue. The Misreporter is trying to maximize the bidders' utility, for the current auction selected by Auctioneer, w. This is achieved by minimizing the loss function $\mathcal{L}_r(\varphi, w)$ wrt to φ (as discussed in Sec 5.3.3). The Auctioneer in turn maximizes expected revenue, for the current misreports as chosen by Misreporter. This is achieved by minimizing $\mathcal{L}_m(w, \varphi) = -(\sqrt{P^w} + \sqrt{R^{w,\varphi}}) + R^{w,\varphi}$ with respect to w (as discussed in Sec 5.3.2). Here, $R^{w,\varphi}$ is an estimate of the total regret that auctioneer computes for the current Misreporter φ, $R^{w,\varphi} = \frac{1}{L}\sum_{\ell=1}^{L}\sum_{i\in N}\left(u_i^w(\vec{v}_i, ([M^\varphi(V)]_i, V_{-i})) - u_i^w(\vec{v}_i, (\vec{v}_i, V_{-i}))\right)$. This game formulation can be summarized in Figure 5.3.1.

Remark 5.3.1. *The game formulation equation G reminds us of Generative Adversarial Networks (Goodfellow et al., 2014). Contrary to GANs, it is not a zero-sum game.*

5.4 Architecture and training procedure

We describe ALGnet, a feed-forward architecture solving for the game formulation equation G and then provide a training procedure. ALGnet consists in two modules that are the auctioneer's module and the misreporter's module. These components take as input a bid matrix $B = (b_{i,j}) \in \mathbb{R}^{n\times m}$ and are trained jointly. Their outputs are used to compute the regret and revenue of the auction.



$$\text{Misreporter: } \begin{cases} \textbf{loss: } \mathcal{L}_r(\varphi, w) \\ \textbf{parameter: } \varphi \end{cases} \qquad \text{Auctioneer: } \begin{cases} \textbf{loss: } \mathcal{L}_m(w, \varphi) \\ \textbf{parameter: } w \end{cases} \tag{G}$$

Figure 5.3.1: Diagrammatic representation of the two-play auction learning game.

Notation. We use $\mathrm{MLP}(d_{\text{in}}, n_l, h, d_{\text{out}})$ to refer to a fully-connected neural network with input dimension d_{in}, output dimension d_{out} and n_l hidden layers of width h and tanh activation function. sig denotes the sigmoid activation function. Given a matrix $B = [\vec{b}_1, \dots, \vec{b}_n]^\top \in \mathbb{R}^{n \times m}$, we define for a fixed $i \in N$, the matrix $B_{(i)} := [\vec{b}_i, \vec{b}_1, \dots, \vec{b}_{i-1}, \vec{b}_{i+1}, \dots, \vec{b}_n]$.

5.4.1 The Auctioneer's module

It is composed of an allocation network that encodes a randomized allocation $g^w : \mathbb{R}^{nm} \to [0,1]^{nm}$ and a payment network that encodes a payment rule $p^w : \mathbb{R}^{nm} \to \mathbb{R}^n$.

Allocation network. It computes the allocation probabily of item j to bidder i $[g^w(B)]_{ij}$ as $[g^w(B)]_{ij} = [f_1(B)]_j \cdot [f_2(B)]_{ij}$ where $f_1 : \mathbb{R}^{n \times m} \to [0,1]^m$ and $f_2 : \mathbb{R}^{n \times m} \to [0,1]^{m \times n}$ are functions computed by two feed-forward neural networks.

– $[f_1(B)]_j$ is the probability that object $j \in M$ is allocated and is given by

$[f_1(B)]_j = \text{sig}\left(\text{MLP}(nm, n_a, h_a, n)\right)$.

– $[f_2(B)]_{ij}$ is the probability that item $j \in M$ is allocated to bidder $i \in N$ conditioned on object j being allocated. A first MLP computes $l_j := \text{MLP}(nm, n_a, h_a, m)(B_{(j)})$ for all $j \in M$. The network then concatenates all these vectors l_j into a matrix $L \in \mathbb{R}^{n \times m}$. A softmax activation function is finally applied to L to ensure feasibility i.e. for all $j \in M, \sum_{i \in N} L_{ij} = 1$.

Payment network. It computes the payment $[p^w(B)]_i$ for bidder i as $[p^w(B)]_i = \tilde{p}_i \sum_{j=1}^{m} B_{ij}[g^w(B)]_{ij}$, where $\tilde{p}\colon \mathbb{R}^{n \times m} \to [0,1]^n$. $\tilde{p}_i$ is the fraction of bidder's i utility that she has to pay to the mechanism. We compute $\tilde{p}_i = \text{sig}\left(\text{MLP}(nm, n_p, h_p, 1)\right)(B_{(i)})$. Finally, notice that by construction $[p^w(B)]_i \leqslant \sum_{j=1}^{m} B_{ij} g^w(B)_{ij}$ which ensures that equation IR is respected.

5.4.2 The Misreporter's module

The module consists in an $\text{MLP}(nm, n_M, h_M, m)$ followed by a projection layer Proj that ensure that the output of the network is in the domain D of the valuation. For example when the valuations are restricted to $[0,1]$, we can take Proj = sig, if they are non negative number,we can take Proj = SoftPlus. The optimal misreport for bidder i is then given by $\text{Proj} \circ \text{MLP}(nm, n_M, h_M, m)(B_{(i)}) \in \mathbb{R}^m$. Stacking these vectors gives us the misreport matrix $M^\varphi(B)$.

5.4.3 Training procedure and optimization

We optimize the game equation G over the space of neural networks parameters (w, φ). The algorithm is easy to implement (Algorithm 5.1). At each time t, we sample a batch of valuation profiles of size B. The algorithm performs τ updates for the Misreporter's network (line 9) and one update on the Auctioneer's network (line 10). Moreover, we often reinitialize the Misreporter's network every T_{init} steps in the early phases of the

training ($t \leqslant T_{limit}$). This step is not necessary but we found empirically that it speeds up training.

Algorithm 5.1 ALGnet training

Input: number of agents, number of objects.

Parameter: $\gamma > 0$; $B, T, T_{init}, T_{limit}, \tau \in \mathbb{N}$.

Initialize misreport's and auctioneer's nets.

for $t = 1, \ldots, T$ **do**

  if $t \equiv 0 \bmod T_{init}$ and $t < T_{Limit}$ **then**:

    Reinitialize Misreport Network

  Sample valuation batch S of size B.

  for $s = 1, \ldots, \tau$ **do**

    $\varphi^{s+1} \leftarrow \varphi^s - \gamma \nabla_\varphi \mathcal{L}_r(\varphi^s, w^t)(S)$.

  $w^{t+1} \leftarrow w^t - \gamma \nabla_w \mathcal{L}_m(w^t, \varphi)(S)$.

5.5 Experimental Results

We show that ALGnet can recover near-optimal auctions for settings where the optimal solution is known and that it can find new auctions for settings where analytical solution are not known. Since RegretNet is already capable of discovering near optimal auctions, one cannot expect ALGnet to achieve significantly higher optimal revenue than RegretNet. The results obtained are competitive or better than the ones obtained in Duetting et al. (2019) while requiring much less hyperparameters (Section 5.3).

We also evaluate ALGnet in online auctions and compare it to RegretNet.

For each experiment, we compute the total revenue $rev := \mathbb{E}_{V \sim D}[\sum_{i \in N} p_i^w(V)]$ and average regret $rgt := {}^1/_n \, \mathbb{E}_{V \sim D}[\sum_{i \in N} r_i^w(V)]$ on a test set of $10,000$ valuation profiles. We run each experiment 5 times with different random seeds and report the average

and standard deviation of these runs. In our comparisons we make sure that ALGnet and RegretNet have similar sizes for fairness (Appendix D).

5.5.1 Auctions with known and unknown optima

Known settings. We show that ALGnet is capable of recovering near optimal auction in different well-studied auctions that have an analytical solution. These are one bidder and two items auctions where the valuations of the two items v_1 and v_2 are independent. We consider the following settings:

- (A): v_1 and v_2 are i.i.d. from $\mathcal{U}[0, 1]$
- (B): $v_1 \sim \mathcal{U}[4, 16]$ and $v_2 \sim \mathcal{U}[4, 7]$
- (C): v_1 has density $f_1(x) = 5/(1 + x)^6$ and v_2 has density $f_2(y) = 6/(1 + y)^7$.

(A) is the celebrated Manelli-Vincent auction (Manelli and Vincent, 2006); (B) is a non-i.i.d. auction and (C) is a non-i.i.d. heavy-tail auction and both of them are studied in Daskalakis et al. (2017). We compare our results to the theoretical optimal auction (Table 5.5.1). (Duetting et al. (2019) does not evaluate RegretNet on settings (B) & (C)). During the training process, *reg* decreases to 0 while *rev* and $\boldsymbol{P}^*$ converge to the optimal revenue. For (A), we also plot *rev*, *rgt* and $\boldsymbol{P}^*$ as function of the number of epochs and we compare it to RegretNet (Figure 5.5.1).

Contrary to ALGnet, we observe that RegretNet overestimates the revenue in the early stages of training at the expense of a higher regret. As a consequence, ALGnet learns the optimal auction faster than RegretNet while being schedule-free and requiring less hyperparameters.

Table 5.5.1: Revenue & regret of ALGnet for settings (A)-(C).

	Optimal		ALGnet (Ours)	
	rev	*rgt*	*rev*	*rgt* ($\times 10^{-3}$)
(A)	0.550	0	0.555 ($\pm$0.0019)	0.55 ($\pm$0.14)
(B)	9.781	0	9.737 ($\pm$0.0443)	0.75 ($\pm$0.17)
(C)	0.1706	0	0.1712 ($\pm$0.0012)	0.14 ($\pm$0.07)

Unknown and large-scale auctions. We now consider settings where the optimal auction is unknown. We look at n-bidder m-item additive settings where the valuations are sampled i.i.d from $\mathcal{U}[0,1]$ which we will denote by $n \times m$. In addition to "reasonable"-scale auctions (1×10 and 2×2), we investigate large-scale auctions (3×10 and 5×10) that are much more complex. Only deep learning methods are able to solve them efficiently. Table 5.5.2 shows that ALGnet is able to discover auctions that yield comparable or better results than RegretNet.

Table 5.5.2: Comparison of RegretNet and ALGnet. The values reported for RegretNet are found in Duetting et al. (2019), the numerical values for *rgt* and standard deviations are not available.

Setting	RegretNet		ALGnet (Ours)	
	rev	*rgt*	*rev*	*rgt*
1×2	0.554	$< 1.0 \cdot 10^{-3}$	0.555 ($\pm$0.0019)	$0.55 \cdot 10^{-3} (\pm 0.14 \cdot 10^{-3})$
1×10	3.461	$< 3.0 \cdot 10^{-3}$	3.487 ($\pm$0.0135)	$1.65 \cdot 10^{-3} (\pm 0.57 \cdot 10^{-3})$
2×2	0.878	$< 1.0 \cdot 10^{-3}$	0.879 ($\pm$0.0024)	$0.58 \cdot 10^{-3} (\pm 0.23 \cdot 10^{-3})$
3×10	5.541	$< 2.0 \cdot 10^{-3}$	5.562 ($\pm$0.0308)	$1.93 \cdot 10^{-3} (\pm 0.33 \cdot 10^{-3})$
5×10	6.778	$< 5.0 \cdot 10^{-3}$	6.781 ($\pm$0.0504)	$3.85 \cdot 10^{-3} (\pm 0.43 \cdot 10^{-3})$

5.5.2 Online auctions

ALGnet is an online algorithm with a time-independent loss function. We would expect it to perform well in settings where the underlying distribution of the valuations changes over time. We consider a one bidder and two items additive auction with valuations v_1 and v_2 sampled i.i.d from $\mathcal{U}[0, 1+t]$ where t in increased from 0 to 1 at a steady rate. The optimal auction at time t has revenue $0.55 \times (1+t)$.

We use ALGnet and two versions of RegretNet, the original offline version (Appendix A) and our own online version (Appendix B) and plot $rev(t)$, $rgt(t)$ and $P^*(t)$ (Figure 5.5.1). The offline version learns from a fixed dataset of valuations sampled at $t = 0$ (i.e. with $V \sim \mathcal{U}[0,1]^{nm}$) while the online versions (as ALGnet) learns from a stream of data at each time t.

Overall, ALGnet performs better than the other methods. It learns an optimal auction faster at the initial (especially compared to RegretNet Online) and keep adapting to the distributional shift (contrary to vanilla RegretNet).

5.6 Conclusion

We identified two inefficiencies in previous approaches to deep auction design and propose solutions, building upon recent trends and results from machine learning (amortization) and theoretical auction design (stationary Lagrangian). This resulted in a novel formulation of auction learning as a two-player game between an Auctioneer and a Misreporter and a new architecture ALGnet. ALGnet requires significantly fewer hyperparameters than previous Lagrangian approaches. We demonstrated the effectiveness of ALGnet on a variety of examples by comparing it to the theoretical optimal auction when it is known, and to RegretNet when the optimal solution is not known.

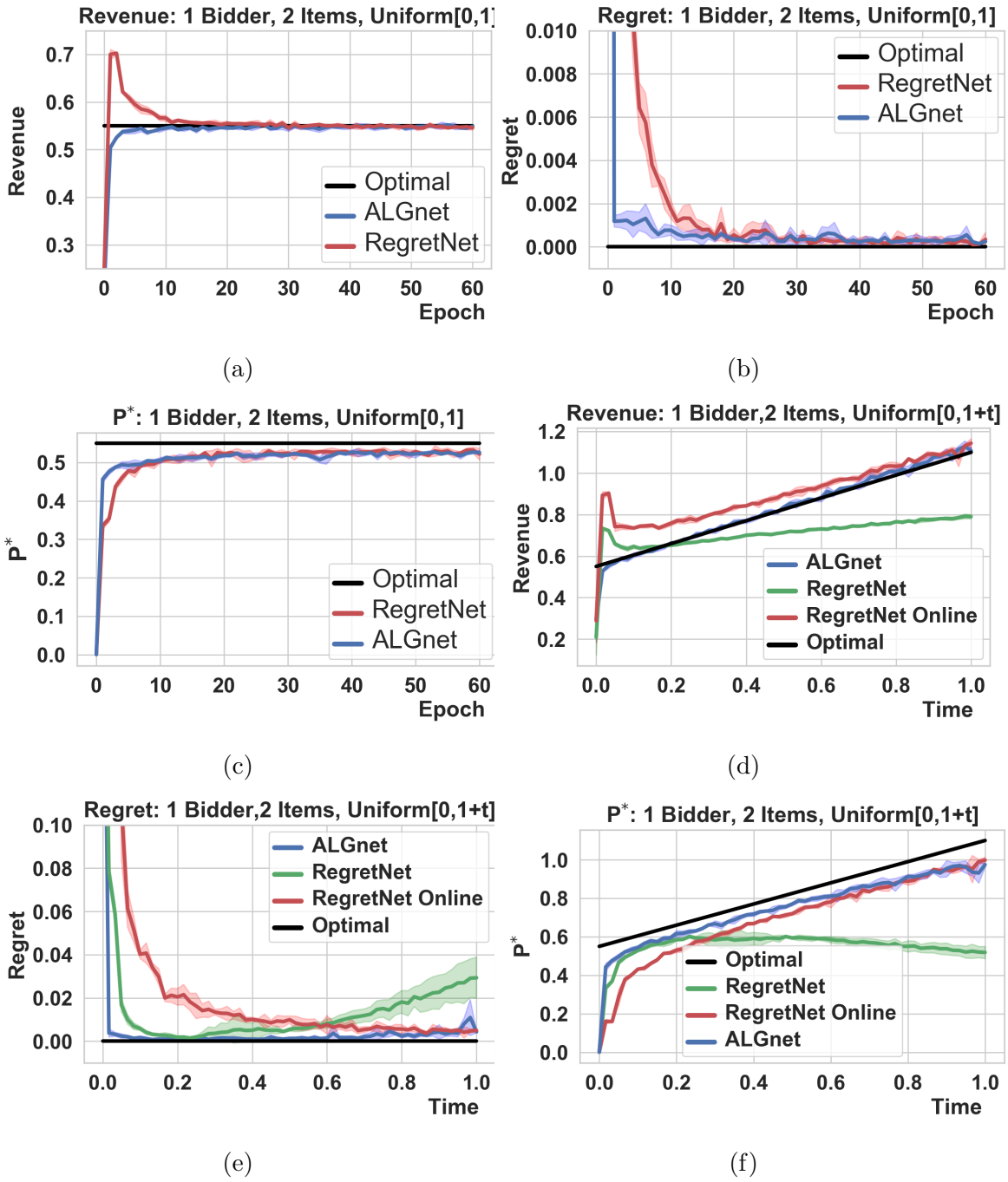


Figure 5.5.1: (a-b-c) compares the evolution of the revenue, regret and P^* as a function of the number of epoch for RegretNet and ALGnet for setting (A). (d-e-f) plots the the revenue, regret and P^* as a function of time for ALGnet and (offline & online) RegretNet for an online auction (Section 5.5.2).

Appendix

5.A Proof of Proposition 5.3.1

Lemma 5.A.1. *Let M be a one bidder m item mechanism with expected revenue P and expected regret R, then $\forall \varepsilon > 0$, there exists a mechanism M' with expected revenue $P' = (1-\varepsilon)P - \frac{1-\varepsilon}{\varepsilon}R$ and zero expected regret, $R' = 0$.*

Proof. For every valuation vector $v \in D$, let $g(v)$ and $p(v)$ denote the allocation vector and price that M assigns to v.

We now consider the mechanism M' that does the following:

- $g'(v) = g(v')$
- $p'(v) = (1-\varepsilon)\, p(v')$

Where v' is given by : $v' = \text{argmax}_{\tilde{v} \in D} \langle v\,,\, g(\tilde{v})\rangle - (1-\varepsilon)\, p(\tilde{v})$. By construction, the mechanism M' has zero regret, all we have to do now is bound its revenue. If we denote by $R(v)$ the regret of the profile v in the mechanism M, $R(v) = \max_{\tilde{v} \in D} \langle v\,,\, g(\tilde{v}) - g(v)\rangle - (p(\tilde{v}) - p(v))$ we have.

$$\langle v\,,\, g(v')\rangle - p(v') = \langle v\,,\, g(v)\rangle - p(v) + \langle v\,,\, g(v') - g(v)\rangle - (p(v') - p(v)) \tag{5.4}$$

$$\leqslant \langle v\,,\, g(v)\rangle - p(v) + R(v) \tag{5.5}$$

Which we will write as:

$$\langle v\,,\, g(v)\rangle - p(v) \geqslant \langle v\,,\, g(v')\rangle - p(v') - R(v) \tag{5.6}$$

Second, we have by construction:

$$\langle v\,,\, g(v')\rangle - (1-\varepsilon)p(v') \geqslant \langle v\,,\, g(v)\rangle - (1-\varepsilon)p(v) \tag{5.7}$$

By summing these two relations we find :

$$p(v') \geqslant p(v) - \frac{R(v)}{\varepsilon} \tag{5.8}$$

Finally we get that:

$$p'(v) \geqslant (1-\varepsilon)\, p(v) - \frac{1-\varepsilon}{\varepsilon}\, R(v) \tag{5.9}$$

Taking the expectation we get:

$$P' \geqslant (1-\varepsilon)\, P - \frac{1-\varepsilon}{\varepsilon}\, R \tag{5.10}$$

□

Proposition 1. *Let $\mathcal{M}$ be an additive auction with 1 bidders and m items. Let P and R denote the total expected revenue and regret, $P = \mathbb{E}_{V\in D}\left[p(V)\right]$ and $R = \mathbb{E}_{V\in D}\left[r(V)\right]$. There exists a mechanism $\mathcal{M}^*$ with expected revenue $P^* = \left(\sqrt{P} - \sqrt{R}\right)^2$ and zero regret $R^* = 0$.*

Proof. From Lemma 5.A.1 we know that $\forall \varepsilon > 0$, we can find a zero regret mechanism with revenue $P' = (1-\varepsilon)\, P - \frac{1-\varepsilon}{\varepsilon}\, R$. By optimizing over ε we find that the best mechanism is the one correspond to $\varepsilon = \sqrt{\frac{R}{P}}$. The resulting optimal revenue is given

by:

$$P^* = (1 - \sqrt{\frac{R}{P}})P - \frac{\sqrt{\frac{R}{P}}}{\sqrt{\frac{R}{P}}}R = P - 2\sqrt{PR} + R = \left(\sqrt{P} - \sqrt{R}\right)^2 \tag{5.11}$$

□

5.B Training Algorithm for Regret Net

We present the training algorithm for RegretNet, more details can be found in Duetting et al. (2019).

Algorithm 5.2 Training Algorithm.

Input: Minibatches $\mathcal{S}_1, \ldots, \mathcal{S}_T$ of size B
Parameters: $\gamma > 0$, $\eta > 0$, $c > 0$, $R \in \mathbb{N}$, $T \in \mathbb{N}$, $T_\rho \in \mathbb{N}$, $T_\lambda \in \mathbb{N}$.
Initialize Parameters: $\rho^0 \in \mathbb{R}$, $w^0 \in \mathbb{R}^d$, $\lambda^0 \in \mathbb{R}^n$,
Initialize Misreports: $v_i'^{(\ell)} \in \mathcal{D}_i,\ \forall \ell \in [B],\ i \in N$.

for $t = 0, \ldots, T$ **do**
    Receive minibatch $\mathcal{S}_t = \{V^{(1)}, \ldots, V^{(B)}\}$.
    for $r = 0, \ldots, R$ **do**

$$\forall \ell \in [B],\ i \in n :$$
$$v_i'^{(\ell)} \leftarrow v_i'^{(\ell)} + \gamma \nabla_{v_i'} u_i^{w^t}(v_i{}^{(\ell)}; (v_i'^{(\ell)}, V_{-i}^{(\ell)}))$$

    Get Lagrangian gradient and update w^t:
        $w^{t+1} \leftarrow w^t - \eta \nabla_w \mathcal{L}(w^t; \lambda^t; \rho^t)$.

    Update ρ once in T_ρ iterations:
    if t is a multiple of T_ρ **then**
        $\rho^{t+1} \leftarrow \rho^t + c$
    else
        $\rho^{t+1} \leftarrow \rho^t$

    Update Lagrange multipliers once in T_λ iterations:
    if t is a multiple of T_λ **then**
        $\lambda_i^{t+1} \leftarrow \lambda_i^t + \rho^t \widehat{r}_i(w^t), \forall i \in N$
    else
        $\lambda^{t+1} \leftarrow \lambda^t$

5.C Training algorithm for Online Regret Net

We present an online version of the training algorithm for RegretNet, more details can be found in Duetting et al. (2019). This version in mentionned in the orginal paper but the algorithm is not explicitly written there. The following code is our own adaptation of the original RegretNet algorithm for online settings.

Algorithm 5.3 Training Algorithm.

Input: Valuation's Distribution $\mathcal{D}$
Parameters: $\gamma > 0$, $\eta > 0$, $c > 0$, $R \in \mathbb{N}$, $T \in \mathbb{N}$, $T_\rho \in \mathbb{N}$, $T_\lambda \in \mathbb{N}, B \in \mathbb{N}$
Initialize Parameters: $\rho^0 \in \mathbb{R}$, $w^0 \in \mathbb{R}^d$, $\lambda^0 \in \mathbb{R}^n$,
for $t = 0, \ldots, T$ **do**
    Sample minibatch $\mathcal{S}_t = \{V^{(1)}, \ldots, V^{(B)}\}$ from distribution $\mathcal{D}$.
    Initialize Misreports: $v_i'^{(\ell)} \in \mathcal{D}_i, \ \forall \ell \in [B], i \in N$.

    for $r = 0, \ldots, R$ **do**

$$\forall \ell \in [B], i \in n :$$
$$v_i'^{(\ell)} \leftarrow v_i'^{(\ell)} + \gamma \nabla_{v_i'} u_i^{w^t}(v_i{}^{(\ell)}; (v_i'^{(\ell)}, V_{-i}^{(\ell)}))$$

    Get Lagrangian gradient and update w^t:
        $w^{t+1} \leftarrow w^t - \eta \nabla_w \mathcal{L}(w^t; \lambda^t; \rho^t)$.

    Update ρ once in T_ρ iterations:
    if t is a multiple of T_ρ **then**
        $\rho^{t+1} \leftarrow \rho^t + c$
    else
        $\rho^{t+1} \leftarrow \rho^t$

    Update Lagrange multipliers once in T_λ iterations:
    if t is a multiple of T_λ **then**
        $\lambda_i^{t+1} \leftarrow \lambda_i^t + \rho^t \widehat{r}_i(w^t), \forall i \in N$
    else
        $\lambda^{t+1} \leftarrow \lambda^t$

5.D Implementation and Setup

We implemented ALGnet in PyTorch and all our experiments can be run on Google's Colab plateform (with GPU). In 5.1, we used batches of valuation profiles of size $B \in \{500\}$ and set $T \in \{160000, 240000\}$, $T_{limit} \in \{40000, 60000\}$, $T_{init} \in \{800, 1600\}$ and $\tau \in \{100\}$.

We used the AdamW optimizer (Loshchilov and Hutter, 2017) to train the Auctioneer's and the Misreporter's networks with learning rate $\gamma \in \{0.0005, 0.001\}$. Typical values for the architecture's parameters are $n_a = n_p = n_m \in [3, 7]$ and $h_p = h_n = h_m \in \{50, 100, 200\}$. These networks are similar in size to the ones used for RegretNet in Duetting et al. (2019).

For each experiment, we compute the total revenue $rev := \mathbb{E}_{V\sim D}[\sum_{i\in N} p_i^w(V)]$ and average regret $rgt := {}^1\!/_n\, \mathbb{E}_{V\sim D}[\sum_{i\in N} r_i^w(V)]$ using a test set of $10,000$ valuation profiles. We run each experiment 5 times with different random seeds and report the average and standard deviation of these runs.

Bibliography

D Ackley, G Hinton, and T Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9(1):147–169, 1985.

Rishabh Agarwal, Marlos C. Machado, P. S. Castro, and Marc G. Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. *ArXiv*, abs/2101.05265, 2021.

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Saeed Alaei. Bayesian Combinatorial Auctions: Expanding Single Buyer Mechanisms to Many Buyers. In *the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2011.

Saeed Alaei, Hu Fu, Nima Haghpanah, Jason Hartline, and Azarakhsh Malekian. Bayesian Optimal Auctions via Multi- to Single-agent Reduction. In *the 13th ACM Conference on Electronic Commerce (EC)*, 2012.

Saeed Alaei, Hu Fu, Nima Haghpanah, and Jason Hartline. The Simple Economics of Approximately Optimal Auctions. In *the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2013.

Saeed Alaei, Jason D. Hartline, Rad Niazadeh, Emmanouil Pountourakis, and Yang Yuan. Optimal auctions vs. anonymous pricing. *CoRR*, abs/1507.02615, 2015. URL `http://arxiv.org/abs/1507.02615`.

Francisco Albarrán-Arriagada, Juan C Retamal, Enrique Solano, and Lucas Lamata. Measurement-based adaptation protocol with quantum reinforcement learning. *Physical Review A*, 98(4):042315, 2018.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Karl J Astrom. Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174–205, 1965.

Hagai Attias. Planning by probabilistic inference. In *International Conference on Artificial Intelligence and Statistics*, 2003.

Moshe Babaioff, Nicole Immorlica, Brendan Lucier, and S. Matthew Weinberg. A simple and approximately optimal mechanism for an additive buyer. *SIGecom Exchanges*, 13(2):31–35, 2014. doi: 10.1145/2728732.2728736. URL `http://doi.acm.org/10.1145/2728732.2728736`.

Maria-Florina Balcan, Avrim Blum, Jason D. Hartline, and Yishay Mansour. Mechanism design via machine learning. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 605–614. IEEE Computer Society, 2005. doi: 10.1109/SFCS.2005.50. URL `https://doi.org/10.1109/SFCS.2005.50`.

Maria-Florina F Balcan, Tuomas Sandholm, and Ellen Vitercik. Sample complexity of automated mechanism design. In *Advances in Neural Information Processing Systems*, pages 2083–2091, 2016.

Xiaohui Bei and Zhiyi Huang. Bayesian Incentive Compatibility via Fractional Assignments. In *the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2011.

Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.

Michael Betancourt. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*, 2017.

Hedyeh Beyhaghi and S. Matthew Weinberg. Optimal (and benchmark-optimal) competition complexity for additive buyers over independent items. In *Proceedings of the 51st ACM Symposium on Theory of Computing Conference (STOC)*, 2019.

Sebastien Bubeck, Nikhil R Devanur, Zhiyi Huang, and Rad Niazadeh. Online auctions and multi-scale online learning. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 497–514, 2017.

Marin Bukov. Reinforcement learning for autonomous preparation of floquet-engineered states: Inverting the quantum kapitza oscillator. *Physical Review B*, 98(22):224305, 2018.

Marin Bukov, Alexandre GR Day, Dries Sels, Phillip Weinberg, Anatoli Polkovnikov, and Pankaj Mehta. Reinforcement learning in different phases of quantum control. *Physical Review X*, 8(3):031086, 2018.

Yang Cai, Constantinos Daskalakis, and S. Matthew Weinberg. An algorithmic characterization of multi-dimensional mechanisms. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA,*

May 19 - 22, 2012, pages 459–478, 2012a. doi: 10.1145/2213977.2214021. URL https://doi.org/10.1145/2213977.2214021.

Yang Cai, Constantinos Daskalakis, and S. Matthew Weinberg. Optimal multi-dimensional mechanism design: Reducing revenue to welfare maximization. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 130–139, 2012b. doi: 10.1109/FOCS.2012.88. URL https://doi.org/10.1109/FOCS.2012.88.

Yang Cai, Nikhil Devanur, and S. Matthew Weinberg. A duality based unified approach to bayesian mechanism design. In *Proceedings of the 48th ACM Conference on Theory of Computation(STOC)*, 2016.

Yang Cai, Argyris Oikonomou, Grigoris Velegkas, and Mingfei Zhao. An efficient ε-bic to BIC transformation and its application to black-box reduction in revenue maximization. *CoRR*, abs/1911.10172, 2019. URL http://arxiv.org/abs/1911.10172.

Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences, 2019.

Stefano Carrazza and Frédéric A Dreyer. Jet grooming through reinforcement learning. *Physical Review D*, 100(1):014014, 2019.

Xi Chen, Ilias Diakonikolas, Dimitris Paparas, Xiaorui Sun, and Mihalis Yannakakis. The complexity of optimal multidimensional pricing. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1319–1328, 2014. doi: 10.1137/1.9781611973402.97. URL http://dx.doi.org/10.1137/1.9781611973402.97.

Xi Chen, Ilias Diakonikolas, Anthi Orfanou, Dimitris Paparas, Xiaorui Sun, and Mihalis Yannakakis. On the complexity of optimal lottery pricing and randomized mechanisms. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1464–1479. IEEE, 2015.

Xi Chen, George Matikas, Dimitris Paparas, and Mihalis Yannakakis. On the complexity of simple and optimal deterministic mechanisms for an additive buyer. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2036–2049. SIAM, 2018.

Jack Clark and Dario Amodei. Faulty reward functions in the wild. *Internet: https://blog. openai. com/faulty-reward-functions*, 2016.

K. Cobbe, Oleg Klimov, Christopher Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *ICML*, 2019.

K. Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. *ArXiv*, abs/1912.01588, 2020.

Richard Cole and Tim Roughgarden. The sample complexity of revenue maximization. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 243–252, 2014.

Vincent Conitzer and Tuomas Sandholm. Complexity of mechanism design. *arXiv preprint cs/0205075*, 2002.

Vincent Conitzer and Tuomas Sandholm. Self-interested automated mechanism design and implications for optimal combinatorial auctions. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 132–141, 2004.

Constantinos Daskalakis and Seth Matthew Weinberg. Symmetries and optimal multi-dimensional mechanism design. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 370–387, 2012.

Constantinos Daskalakis, Alan Deckelbaum, and Christos Tzamos. The Complexity of Optimal Mechanism Design. In *the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2014a.

Constantinos Daskalakis, Alan Deckelbaum, and Christos Tzamos. The complexity of optimal mechanism design. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1302–1318. SIAM, 2014b.

Constantinos Daskalakis, Alan Deckelbaum, and Christos Tzamos. Strong duality for a multiple-good monopolist. *Econometrica*, 85(3):735–767, 2017.

R. Dearden, N. Friedman, and Stuart J. Russell. Bayesian q-learning. In *AAAI/IAAI*, 1998.

R. Dearden, N. Friedman, and D. Andre. Model based bayesian exploration. In *UAI*, 1999.

Ofer Dekel and Elad Hazan. Better rates for any adversarial deterministic MDP. In *International Conference on Machine Learning*, pages 675–683, 2013.

Nikhil R. Devanur, Zhiyi Huang, and Christos-Alexandros Psomas. The sample complexity of auctions with side information. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 426–439, 2016. doi: 10.1145/2897518.2897553. URL `http://doi.acm.org/10.1145/2897518.2897553`.

Peerapong Dhangwatnotai, Tim Roughgarden, and Qiqi Yan. Revenue maximization with a single sample. *Games and Economic Behavior*, 91:318–333, 2015. doi: 10.1016/j.geb.2014.03.011. URL `https://doi.org/10.1016/j.geb.2014.03.011`.

Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

Paul Duetting, Zhe Feng, Harikrishna Narasimhan, David Parkes, and Sai Srivatsa Ravindranath. Optimal auctions through deep learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1706–1715, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/duetting19a.html.

M. Duff and A. Barto. Optimal learning: computational procedures for bayes-adaptive markov decision processes. 2002.

Shaddin Dughmi, Li Han, and Noam Nisan. Sampling and representation complexity of revenue maximization. In *International Conference on Web and Internet Economics*, pages 277–291. Springer, 2014.

Shaddin Dughmi, Jason D. Hartline, Robert Kleinberg, and Rad Niazadeh. Bernoulli factories and black-box reductions in mechanism design. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 158–169, 2017. doi: 10.1145/3055399.3055492. URL http://doi.acm.org/10.1145/3055399.3055492.

Paul Dütting, Felix Fischer, Pichayut Jirapinyo, John K Lai, Benjamin Lubin, and David C Parkes. Payment rules through discriminant-based classifiers. *ACM Transactions on Economics and Computation (TEAC)*, 3(1):1–41, 2015.

Benjamin Eysenbach and Sergey Levine. If maxent rl is the answer, what is the question? *arXiv preprint arXiv:1910.01913*, 2019.

Jesse Farebrother, Marlos C. Machado, and Michael H. Bowling. Generalization and regularization in dqn. *ArXiv*, abs/1810.00123, 2018.

Zhe Feng, Harikrishna Narasimhan, and David C Parkes. Deep learning for revenue-optimal auctions with budgets. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*, pages 354–362. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

Thomas Fösel, Petru Tighineanu, Talitha Weiss, and Florian Marquardt. Reinforcement learning with neural networks for quantum feedback. *Physical Review X*, 8(3):031084, 2018.

Scott Fujimoto, H. V. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. *ArXiv*, abs/1802.09477, 2018.

M. Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *Found. Trends Mach. Learn.*, 8:359–483, 2015.

Dibya Ghosh, Avi Singh, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. Divide-and-conquer reinforcement learning. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJwelMbR-.

Dibya Ghosh*, Jad Rahme*, Aviral Kumar, Amy Zhang, Ryan P. Adams, and Sergey Levine. Why generalization in rl is difficult: Epistemic pomdps and implicit partial observability. *Advances in Neural Information Processing Systems*, 34, 2021.

Noah Golowich, Harikrishna Narasimhan, and David C. Parkes. Deep learning for multi-facility location mechanism design. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2018)*, pages 261–267, 2018. URL `https://econcs.seas.harvard.edu/files/econcs/files/golowich_ijcai18.pdf`.

Yannai A. Gonczarowski and Noam Nisan. Efficient empirical revenue maximization in single-parameter auction environments. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 856–868, 2017. doi: 10.1145/3055399.3055427. URL `http://doi.acm.org/10.1145/3055399.3055427`.

Yannai A. Gonczarowski and S. Matthew Weinberg. The sample complexity of up-to-ε multidimensional revenue maximization. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(9), 2004.

Chenghao Guo, Zhiyi Huang, and Xinzhi Zhang. Settling the sample complexity of single-parameter revenue maximization. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 662–673, 2019. doi: 10.1145/3313276.3316325. URL `https://doi.org/10.1145/3313276.3316325`.

Mingyu Guo and Vincent Conitzer. Computationally feasible automated mechanism design: General approach and case studies. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

Sergiu Hart and Noam Nisan. Approximate Revenue Maximization with Multiple Items. In *the 13th ACM Conference on Electronic Commerce (EC)*, 2012.

Sergiu Hart and Noam Nisan. The menu-size complexity of auctions. In *the 14th ACM Conference on Electronic Commerce (EC)*, 2013.

Sergiu Hart and Philip J. Reny. Maximizing Revenue with Multiple Goods: Nonmonotonicity and Other Observations. *Theoretical Economics*, 10(3):893–922, 2015.

Jason Hartford, Devon R Graham, Kevin Leyton-Brown, and Siamak Ravanbakhsh. Deep models of interactions across sets. *arXiv preprint arXiv:1803.02879*, 2018.

Jason D. Hartline and Brendan Lucier. Bayesian Algorithmic Mechanism Design. In *the 42nd ACM Symposium on Theory of Computing (STOC)*, 2010.

Jason D. Hartline and Tim Roughgarden. Simple versus optimal mechanisms. In *ACM Conference on Electronic Commerce*, pages 225–234, 2009.

Jason D. Hartline and Samuel Taggart. Sample complexity for non-truthful mechanisms. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019.*, pages 399–416, 2019. doi: 10.1145/3328526.3329632. URL `https://doi.org/10.1145/3328526.3329632`.

Jason D. Hartline, Robert Kleinberg, and Azarakhsh Malekian. Bayesian Incentive Compatibility via Matchings. In *the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2011.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8): 2554–2558, 1982.

Zhiyi Huang, Yishay Mansour, and Tim Roughgarden. Making the most of your samples. *SIAM Journal on Computing*, 47(3):651–674, 2018.

Maximilian Igl, K. Ciosek, Yingzhen Li, Sebastian Tschiatschek, C. Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. In *NeurIPS*, 2019.

Max Jaderberg, V. Mnih, Wojciech Czarnecki, Tom Schaul, Joel Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *ArXiv*, abs/1611.05397, 2017.

Wonseok Jeon, Seokin Seo, and Kee-Eung Kim. A bayesian approach to generative adversarial imitation learning. In *NeurIPS*, 2018.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. *ArXiv*, abs/2010.03934, 2020.

Yaonan Jin, Pinyan Lu, Qi Qi, Zhihao Gavin Tang, and Tao Xiao. Tight approximation ratio of anonymous pricing. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 674–685. ACM, 2019a. doi: 10.1145/3313276.3316331. URL `https://doi.org/10.1145/3313276.3316331`.

Yaonan Jin, Pinyan Lu, Zhihao Gavin Tang, and Tao Xiao. Tight revenue gaps among simple mechanisms. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 209–228. SIAM, 2019b.

Niels Justesen, R. Torrado, Philip Bontrager, Ahmed Khalifa, J. Togelius, and S. Risi. Illuminating generalization in deep reinforcement learning through procedural level generation. *arXiv: Learning*, 2018.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2): 99–134, 1998.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *In Proc. 19th International Conference on Machine Learning*. Citeseer, 2002.

Sham M Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

Katie Kang, Suneel Belkhale, Gregory Kahn, Pieter Abbeel, and Sergey Levine. Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight. In *2019 international conference on robotics and automation (ICRA)*, pages 6008–6014. IEEE, 2019.

Hilbert J Kappen. Optimal control theory and the linear Bellman equation. In D Barber, A T Cemgil, and S Chiappa, editors, *Bayesian Time Series Models*. Cambridge University Press, 2011.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail`, 2018.

Ilya Kostrikov, Denis Yarats, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *ArXiv*, abs/2004.13649, 2020.

Pravesh Kothari, Divyarthi Mohan, Ariel Schvartzman, Sahil Singla, and S Matthew Weinberg. Approximation schemes for a buyer with independent items via symmetries. *arXiv preprint arXiv:1905.05231*, 2019.

Heinrich Kuttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment. *ArXiv*, abs/2006.13760, 2020.

Sébastien Lahaie. A kernel-based iterative combinatorial auction. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

A. Lazaric and M. Ghavamzadeh. Bayesian multi-task reinforcement learning. In *ICML*, 2010.

Nevena Lazic, Craig Boutilier, Tyler Lu, Eehern Wong, Binz Roy, MK Ryu, and Greg Imwalle. Data center cooling using model-predictive control. In *Advances in Neural Information Processing Systems*, pages 3814–3823, 2018.

Kimin Lee, Kibok Lee, Jinwoo Shin, and H. Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. In *ICLR*, 2020a.

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. *ArXiv*, abs/2005.06800, 2020b.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review, 2018.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Xinye Li and Andrew Chi-Chih Yao. On revenue maximization for selling multiple independently distributed items. *Proceedings of the National Academy of Sciences*, 110(28):11232–11237, 2013.

Zhuang Liu, Xuanlin Li, Bingyi Kang, and Trevor Darrell. Regularization matters in policy optimization – an empirical study on continuous control. *arXiv: Learning*, 2020.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

X. Lu, Kimin Lee, P. Abbeel, and Stas Tiomkin. Dynamics generalization via information bottleneck in deep reinforcement learning. *ArXiv*, abs/2008.00614, 2020.

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Omid Madani. Polynomial value iteration algorithms for deterministic MDPs. In *Proceedings of the Eighteenth conference on Uncertainty in Artificial Intelligence*, pages 311–318. Morgan Kaufmann Publishers Inc., 2002.

Alejandro Manelli and Daniel Vincent. Bundling as an optimal selling mechanism for a multiple-good monopolist. *Journal of Economic Theory*, 127(1):1–35, 2006.

Alejandro M Manelli and Daniel R Vincent. Bayesian and dominant-strategy implementation in the independent private-values model. *Econometrica*, 78(6): 1905–1938, 2010.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

Andres M Medina and Mehryar Mohri. Learning theory and algorithms for revenue optimization in second price auctions with reserve. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 262–270, 2014.

Marc Mezard and Andrea Montanari. *Information, physics, and computation*. Oxford University Press, 2009.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015. doi: 10.1038/nature14236. URL `https://doi.org/10.1038/nature14236`.

George E Monahan. State of the art—a survey of partially observable markov decision processes: theory, models, and algorithms. *Management science*, 28(1):1–16, 1982.

Guido Montúfar, K. Zahedi, and N. Ay. Geometry and determinism of optimal stationary control in partially observable markov decision processes. *ArXiv*, abs/1503.07206, 2015.

Jamie Morgenstern and Tim Roughgarden. Learning simple auctions. In *Conference on Learning Theory*, pages 1298–1318, 2016.

Jamie H Morgenstern and Tim Roughgarden. On the pseudo-dimension of nearly optimal auctions. In *Advances in Neural Information Processing Systems*, pages 136–144, 2015.

Iain Murray and Zoubin Ghahramani. Bayesian learning in undirected graphical models: approximate MCMC algorithms. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 392–399. AUAI Press, 2004.

Roger B Myerson. Optimal auction design. *Mathematics of operations research*, 6(1): 58–73, 1981.

Harikrishna Narasimhan and David C Parkes. A general statistical framework for designing strategy-proof assignment mechanisms. In *UAI'16 Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, 2016.

Radford M Neal et al. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2, 2011.

Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta learn fast: A new benchmark for generalization in rl. *ArXiv*, abs/1804.03720, 2018.

Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 2007.

Murphy Yuezhen Niu, Sergio Boixo, Vadim N Smelyanskiy, and Hartmut Neven. Universal quantum control through deep reinforcement learning. *npj Quantum Information*, 5(1):1–8, 2019.

Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. In *NIPS*, 2013.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL `https://proceedings.neurips.cc/paper/2016/file/8d8818c8e140c64c743113f563cf750f-Paper.pdf`.

Gregory Pavlov. Optimal mechanism for selling two goods. *The B.E. Journal of Theoretical Economics*, 11(3), 2011.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *CoRR*, abs/1710.06537, 2017. URL `http://arxiv.org/abs/1710.06537`.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, May 2018. doi: 10.1109/ICRA.2018.8460528.

P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete bayesian reinforcement learning. *Proceedings of the 23rd international conference on Machine learning*, 2006.

Jad Rahme and Ryan P. Adams. A theoretical connection between statistical physics and reinforcement learning. *arXiv preprint arXiv:1906.10228*, 2019.

Jad Rahme, Samy Jelassi, Joan Bruna, and S. Matthew Weinberg. A permutation-equivariant neural network architecture for auction design. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(6):5664–5672, May 2021a. URL `https://ojs.aaai.org/index.php/AAAI/article/view/16711`.

Jad Rahme, Samy Jelassi, Joan Bruna, and S. Matthew Weinberg. A permutation-equivariant neural network architecture for auction design. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021b.

Jad Rahme, Samy Jelassi, and S. Matthew Weinberg. Auction learning as a two-player game. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021c. URL `https://openreview.net/forum?id=YHdeAO6116T`.

Roberta Raileanu, Maxwell Goldstein, Denis Yarats, Ilya Kostrikov, and R. Fergus. Automatic data augmentation for generalization in deep reinforcement learning. *ArXiv*, abs/2006.12862, 2020.

A. Rajeswaran, Sarvjeet Ghotra, Sergey Levine, and Balaraman Ravindran. Epopt: Learning robust neural network policies using model ensembles. *ArXiv*, abs/1610.01283, 2017.

Deepak Ramachandran and E. Amir. Bayesian inverse reinforcement learning. In *IJCAI*, 2007.

Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayes-adaptive pomdps. In *NIPS*, pages 1225–1232, 2007.

T. Roughgarden. *Twenty Lectures on Algorithmic Game Theory*. Cambridge University Press, 2016. ISBN 9781316782095. URL `https://books.google.fr/books?id=5hjCDwAAQBAJ`.

Tim Roughgarden and Okke Schrijvers. Ironing in the dark. In *Proceedings of the 2016 ACM Conference on Economics and Computation, EC '16, Maastricht, The Netherlands, July 24-28, 2016*, pages 1–18, 2016. doi: 10.1145/2940716.2940723. URL `http://doi.acm.org/10.1145/2940716.2940723`.

Aviad Rubinstein and S Matthew Weinberg. Simple mechanisms for a subadditive buyer and applications to revenue monotonicity. *ACM Transactions on Economics and Computation (TEAC)*, 6(3-4):1–25, 2018.

Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Math. Oper. Res.*, 39:1221–1243, 2014.

Fereshteh Sadeghi and Sergey Levine. (cad)2rl: Real single-image flight without a single real image. *ArXiv*, abs/1611.04201, 2017.

Tuomas Sandholm and Anton Likhodedov. Automated design of revenue-maximizing combinatorial auctions. *Operations Research*, 63(5):1000–1025, 2015.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

John Schulman, F. Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

Weiran Shen, Pingzhong Tang, and Song Zuo. Automated mechanism design via neural networks. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*, pages 215–223. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

Horst Siebert. *Der Kobra-Effekt: Wie man Irrwege der Wirtschaftspolitik vermeidet*. Dt. Verlag-Anst., 2001.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Satinder P. Singh, Tommi S. Jaakkola, and Michael I. Jordan. Learning without state-estimation in partially observable markovian decision processes. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, page 284–292, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. ISBN 1558603352.

Edward J Sondik. The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26(2):282–304, 1978.

Edward Jay Sondik. *The optimal control of partially observable Markov processes*. Stanford University, 1971.

Xingyou Song, Yiding Jiang, Stephen Tu, Yilun Du, and Behnam Neyshabur. Observational overfitting in reinforcement learning. *ArXiv*, abs/1912.02975, 2020.

Austin Stone, Oscar Ramirez, K. Konolige, and Rico Jonschkowski. The distracting control suite - a challenging benchmark for reinforcement learning from pixels. *ArXiv*, abs/2101.02722, 2021.

Adam Stooke, Kimin Lee, P. Abbeel, and M. Laskin. Decoupling representation learning from reinforcement learning. *ArXiv*, abs/2009.08319, 2020.

Edwin Stoudenmire and David J Schwab. Supervised learning with tensor networks. In *Advances in Neural Information Processing Systems*, pages 4799–4807, 2016.

M. Strens. A bayesian framework for reinforcement learning. In *ICML*, 2000.

F. Stulp, E. Theodorou, J. Buchli, and S. Schaal. Learning to grasp under uncertainty. *2011 IEEE International Conference on Robotics and Automation*, pages 5703–5708, 2011.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Vasilis Syrgkanis. A sample complexity measure with applications to learning optimal auctions. In *Advances in Neural Information Processing Systems*, pages 5352–5359, 2017.

Andrea Tacchetti, DJ Strouse, Marta Garnelo, Thore Graepel, and Yoram Bachrach. A neural architecture for designing truthful and efficient auctions. *arXiv preprint arXiv:1907.05181*, 2019.

Yee Teh, Victor Bapst, Wojciech M. Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/0abdc563a06105aee3c6136871c9f4d1-Paper.pdf`.

John Thanassoulis. Haggling over substitutes. *Journal of Economic Theory*, 117: 217–245, 2004.

Joshua Tobin, Rachel Fong, Alex Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

Emanuel Todorov. Linearly-solvable Markov decision problems. In *Advances in Neural Information Processing Systems*, pages 1369–1376, 2007.

Harm Van Seijen, Hado Van Hasselt, Shimon Whiteson, and Marco Wiering. A theoretical and empirical analysis of expected SARSA. In *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 177–184. IEEE, 2009.

William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 16(1):8–37, 1961.

Zihe Wang and Pingzhong Tang. Optimal mechanisms with simple menus. In *ACM Conference on Economics and Computation, EC '14, Stanford , CA, USA, June 8-12, 2014*, pages 227–240, 2014. doi: 10.1145/2600057.2602863. URL `https://doi.org/10.1145/2600057.2602863`.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3): 279–292, 1992.

Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

R. Weber. On the gittins index for multiarmed bandits. *Annals of Applied Probability*, 2:1024–1033, 1992.

Zheng Wen and Benjamin Van Roy. Efficient exploration and value function generalization in deterministic systems. In *Advances in Neural Information Processing Systems*, pages 3021–3029, 2013.

S. Whiteson, B. Tanner, Matthew E. Taylor, and P. Stone. Protecting against evaluation overfitting in empirical reinforcement learning. *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 120–127, 2011.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. *Advances in neural information processing systems*, 30, 2017.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Jonathan S Yedidia, William T Freeman, and Yair Weiss. Generalized belief propagation. In *Advances in Neural Information Processing Systems*, pages 689–695, 2001.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

Lenka Zdeborová and Florent Krzakala. Statistical physics of inference: Thresholds and algorithms. *Advances in Physics*, 65(5):453–552, 2016.

A. Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *ArXiv*, abs/1806.07937, 2018a.

A. Zhang, Yuxin Wu, and Joelle Pineau. Natural environment benchmarks for reinforcement learning. *ArXiv*, abs/1811.06032, 2018b.

A. Zhang, Rowan McAllister, R. Calandra, Y. Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *ArXiv*, abs/2006.10742, 2020.

C. Zhang, Oriol Vinyals, R. Munos, and S. Bengio. A study on overfitting in deep reinforcement learning. *ArXiv*, abs/1804.06893, 2018c.

Zhenpeng Zhou, Xiaocheng Li, and Richard N Zare. Optimizing chemical reactions with deep reinforcement learning. *ACS central science*, 3(12):1337–1344, 2017.

Brian D Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, University of Washington, 2010.

Luisa M. Zintgraf, K. Shiarlis, Maximilian Igl, Sebastian Schulze, Y. Gal, Katja Hofmann, and S. Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. *ArXiv*, abs/1910.08348, 2020.