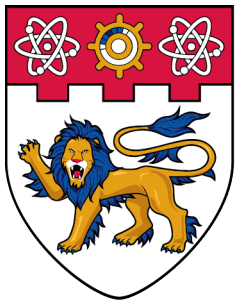

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Final Year Project Report
Reinforcement Learning in Online Principal-Agent Problems

Submitted by

Yue Ming Long

Supervised by

Asst Prof Pun Chi Seng

Ms Nixie Sapphira Lesmana

Submitted to the School of Physical and Mathematical Sciences
in partial fulfillment of the requirements for the degree of
Bachelor of Science at

NANYANG TECHNOLOGICAL UNIVERSITY

Apr 2021

Reinforcement Learning in Online Principal-Agent Problems

Abstract

The principal-agent problem arises when an entity (the agent) acts or makes decisions on behalf of another (the principal) that goes against the best interests of the principal, typically the result of asymmetric information. To address the problem, the principal can align incentives through the use of appropriate contracts. In this paper, we focus on online principal-agent problems. We propose the utilisation of reinforcement learning methods to allow the principal to learn to generate optimal contracts in an end-to-end fashion, solving the principal-agent problem in a model-free manner without the need for prior knowledge of the environment or explicit modelling of the problem.

Contents

1 Introduction

1.1 Problem Statement

The principal-agent problem arises when an entity (the agent) acts or makes decisions on behalf of another (the principal) that potentially goes against the best interests of the principal. This problem is typically the result of conflicting interests and asymmetric information, with the agent possessing greater information. The study of principal-agent problems is closely related to areas in contract theory and mechanism design. There are multiple types of classical principal-agent problems studied in literature [1], such as risk sharing, hidden type and hidden action, and they are typically classified by the nature of the asymmetric information between the principal and the agent. In this paper, we focus on online principal-agent problems, where single-period contracts are used and data becomes available in a sequential order.

1.2 Motivation

To address the problem, the principal can align incentives through the use of appropriate contracts. Contract generation can be viewed as an optimization problem, where the principal aims to maximise his own utility. In certain cases of principal-agent problems, it could be possible to model the problems to derive the optimal contracts analytically [2]. However, it may be prohibitively difficult or even impossible to explicitly model the problem, especially when the principal has little to no prior information about the problem/environment.

Reinforcement learning is an area of machine learning which enables software agents to learn suitable actions through interaction and feedback from the environment. The use of reinforcement learning methods could allow the principal to learn (from experience) to dynamically generate optimal contracts in an end-to-end fashion without the need to explicitly construct such exact mathematical models. This main objective of this paper is to show that reinforcement learning can allow us to solve the principal-agent problem in a model-free manner, without the need for prior knowledge of the environment or explicit modelling of the problem.

1.3 Literature Review

The principal agent problems presented in this paper are based off the moral hazard problem first described by Holmstrom and Milgrom [3] and the sharecropping problem first described by Stiglitz [4]. Holmstrom and Milgrom, and more recently Caroll [5], also show the optimality of linear contracts in such problems.

The reinforcement learning algorithms used in this paper are adapted from pre-existing algorithms. The bandit and REINFORCE algorithms were adapted from the textbook by Sutton and Barto [6]. The TD3 algorithm used was designed by Fujimoto et al. [7] and implemented by the contributors of the stable-baselines repository [8].

There has also been some earlier research on the use of reinforcement learning to solve similar principal agent problems. Ho et al. studied the use of bandit algorithms in crowdsourcing markets [9] and introduced the High-Low problem example which will be elaborated on in later sections. Arifovic and Karaivanov compared to use of social learning and reinforcement learning in a sharecropping problem [10]. Conitzer and Garera studied the use of various learning algorithms in dynamic task pricing, in the context of online auctions [11].

1.4 Problem Definition

In this section, we attempt to define the problem more formally. In this paper, we examine the case of principal-agent problems where the principal is unable to observe both the agent's actions as well as his characteristics/type, known as moral hazard and adverse selection respectively.

We consider a modified version of the single-period moral hazard problem first described by Holmstrom and Milgrom [3]. The state of the environment can be represented with θ which can take some number of finite values. Each individual agent has a type t. The agent chooses to exert some effort level e, which along with the agent type affect the probability distribution over the states, and can be represented as the agent's production function $p(e,t)$. The agent also has a cost function $c(e,t)$. Each state θ_i is associated with some monetary payoff π_i and observable outcome x_i for the principal.

Both parties are assumed to be rational and self-interested, maximising their own utility. Being unable to observe the agent type and effort level, the principal instead rewards the agent based on the observable outcome x by some reward function $f(x)$ outlined by a contract. In other words, the reward function is a mapping from the observable outcome to the actual payment for the agent. The income of the principal is $\pi - f(x)$ and the income of the agent is $f(x) - c(e,t)$. Both the principal and the agent have their own utility functions, $U(\pi - f(x))$ and $V(f(x) - c(e,t))$ respectively, which are a function of their income and can be affected by their risk preferences.

The agent has a minimum required utility w and will only accept the contract if and only if $V(f(x) - c(e,t)) \geq w$. This is known as the participation constraint. Should the participation constraint not be fulfilled, the agent would choose an effort level of 0. This is a special case which deterministically leads to the null outcome, where there is 0 value and 0 payment. All other outcomes are referred to as non-null outcomes.

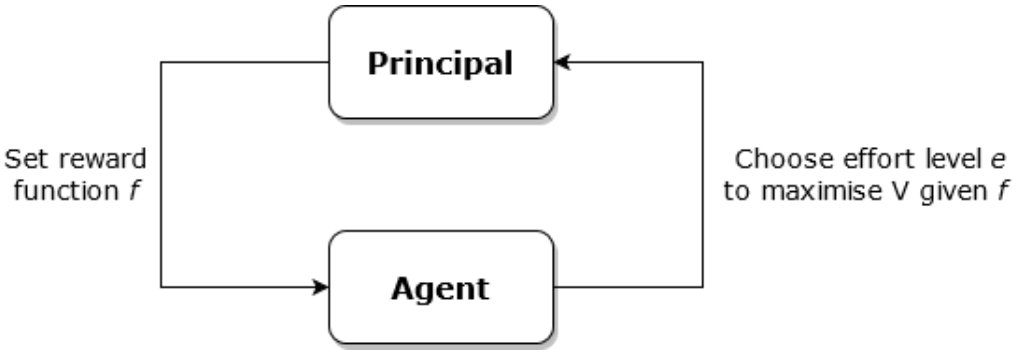

Figure 1: The principal-agent problem expressed as an MDP

A summary of the sequence of events in every round is as follows:

1. The principal decides on the reward function f and offers the contract to the agent.
2. The agent chooses to reject the contract or accept the contract based on the participation constraint.
    - If the agent rejects the contract, the round is terminated. Both the principal and the agent receive 0 payment.
    - If the agent accepts the contract, he chooses between an effort level e that maximises his utility.
3. Quality of the outcome x is realized, which probability is determined by the effort level.
4. The principal pays the agent an amount $f(x)$ as stipulated in the contract.
5. The principal receives the payment π for achieving a certain quality outcome.

Formally, we first solve the agent's problem given reward function f:

$$\max_{e} \sum_{i} p_i(e,t) \cdot V(f(x_i) - c(e,t))$$

Subject to participation constraint:

$$\sum_{i} p_i(e,t) \cdot V(f(x_i) - c(e,t)) \geq w$$

Assuming there is only one optimal effort level e, the principal's problem is then:

$$\max_{f} \sum_{i} p_i(e,t) \cdot U(\pi_i - f(x_i))$$

The above illustrates the outcomes/rewards for a single round. The principal plays multiple rounds in an online manner, with multiple different agents of which type is drawn i.i.d from some arbitrary distribution. Each round is independent of another, and thus the Markov property is satisfied. As seen in Figure 1 above, the problem can be easily represented as an MDP, which will be further elaborated on in Section 2. Thus, the reinforcement learning algorithms described in Section 2 can be used to derive a solution.

This paper aims to show empirically that the use of standard reinforcement learning techniques can allow us to solve the principal's problem as described above in an iterative manner in this online setting where there is little information and no priors. We first examine the case where there are a small number of non-null and possibly qualitative (i.e non-numerical) outcomes in Section 3 followed by the case where there are a large number or possibly infinite (in the case of continuous outcomes) non-null outcomes in Section 4.

2 Reinforcement Learning

In this section, we will detail and define some of the important concepts in Reinforcement Learning (RL) as well as provide a brief overview of the RL algorithms used in this paper.

2.1 Markov Decision Processes

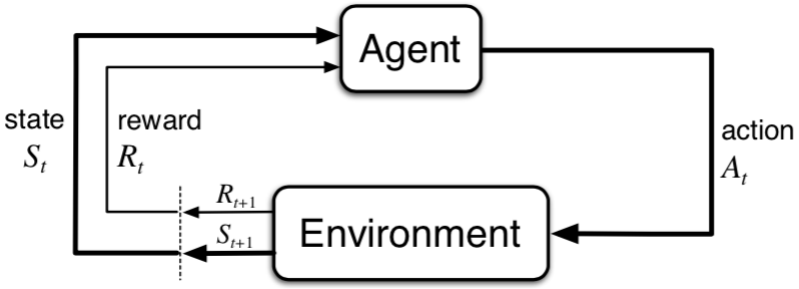

Figure 2: The agent-environment interaction in a Markov Decision Process

RL problems are typically expressed using the Markov Decision Process (MDP) [12] framework and many RL algorithms use dynamic programming methods [13], whereby a problem can be broken down into simpler sub-problems in a recursive manner. The recursive nature of the MDP can be seen in Figure 2.

An MDP can be expressed as a tuple (S, A, P, R, γ)

- S is a finite set of states
- A is a finite set of actions
- P is a state transition probability matrix,

$$P^a_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

- R is a reward function,

$$R^a_s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$$

- γ is a discount factor, $\gamma \in [0, 1]$

An MDP possesses the Markov property, which states that a future is independent of the past given the present. Formally,

$$\mathbb{P}[S_{t+1} | S_t] = \mathbb{P}[S_{t+1} | S_1, ..., S_t]$$

Return of an agent at time t is expressed as G_t, where

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

The sequence of decisions made is a policy. More formally, a policy π defines a probability distribution over actions for each state, where

$$\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$$

Following policy π, we can express the state value function $V_\pi(s)$ and action value function $Q_\pi(s,a)$ as

$$V_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$$

$$Q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$$

2.2 Bellman Equations

The Bellman Equation is one of the central elements of many RL algorithms. It decomposes the value function into two parts, the immediate reward and the discounted future rewards. This simplifies the computation of the value function by breaking the problem into simpler recursive sub-problems and finding their optimal solutions [14]. The derivation of the equations are omitted.

The Bellman equation for the state value function defined recursively is

$$V_\pi(s) \ = \ \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}(R^a_s + \gamma V_\pi(s'))$$

The relation between the state value function $V_\pi(s)$ and action value function $Q_\pi(s,a)$ is

$$V_\pi(s) \ = \ \sum_a \pi(s,a) Q_\pi(s,a)$$

Similarly, the Bellman equation for the action value function is

$$Q_\pi(s,a) \ = \ \sum_{s'} P^a_{ss'}(R^a_s + \gamma \cdot \sum_{a'} \pi(a'|s') \cdot Q_\pi(s',a'))$$

The optimal policy is one that maximises the total cumulative reward. Bellman then proved that the Bellman optimality equation is able to provide the best reward and the optimal policy.

The Bellman optimality equation for the state value function is

$$V_*(s) = \sum_{s'} P^a_{ss'}(R^a_s + \gamma V_*(s'))$$

Similarly, the Bellman optimality equation for the action value function is

$$Q_*(s,a) = \sum_{s'} P^a_{ss'}(R^a_s + \gamma \cdot \max_{a'} Q_*(s',a'))$$

2.3 Exploration-Exploitation Trade-off

The trade-off between exploration and exploitation is crucial to understand in reinforcement learning [15]. During the learning process, before an action is taken, an agent has to choose between gathering more information that might lead to better decisions in the future (exploration) or making the best decision given the current information (exploitation). Clearly, pure exploration or exploitation strategies are flawed. In this paper, we consider two common policies, ϵ-greedy and Boltzmann Selection.

ϵ-greedy: With probability ϵ, a random action is chosen uniformly over all possible actions. With probability $1-\epsilon$, the greedy action is chosen. More formally, at state s, an action a is chosen by $\underset{a}{\operatorname{argmax}}\, Q(s,a)$. There are also other variants of this policy, such as ϵ-decay where the value of ϵ falls over time [16].

Boltzmann Selection: This policy is based off a Boltzmann (or softmax) distribution. At state s, an action a is chosen with probability $p = \frac{\exp \frac{Q(s,a)-\max_b Q(s,b)}{T}}{\sum_a \exp \frac{Q(s,a)-\max_b Q(s,b)}{T}}$. T is the temperature parameter which increases the exploration rate as it increases.

2.4 Multi-Armed Bandits

Figure 3: An agent choosing between multiple levers

The multi-armed bandit problem is formally equivalent to a one-state MDP. In a bandit setting [17], an agent has to choose between multiple levers (like in a slot machine) as can be seen in Figure 3. The agent plays one lever per round and observes the associated reward. The rewards associated with the other chosen levers are unknown, unlike in a experts setting [18]. This is called bandit feedback. The objective would be to maximise

the total accumulated reward. The ϵ-greedy exploration strategy is used.

Algorithm 1 ϵ-greedy Bandit Algorithm

Inputs:
Number of rounds N
Exploration parameter ϵ
Set of bounded and monotone contracts A
Initialize:
$Q(a) \leftarrow 0$ for all $a \in A$
for $n \leftarrow 1$ to N **do**
Draw action a using ϵ-greedy policy π derived from Q
$Q(a) \leftarrow Q(a) + \frac{r - Q(a)}{n}$
end for

2.5 Policy Gradient Methods

2.5.1 Vanilla Methods

While the bandit algorithm and other popular RL methods such as Q-Learning focuses on learning action-value functions Q, policy gradient methods learn a parameterized policy directly based on the gradient of some scalar performance measure $J(\boldsymbol{\theta})$ with respect to the policy parameters $\boldsymbol{\theta}$. This allows us handle high dimensional action spaces as well as continuous actions relatively easily. The Boltzmann Selection exploration strategy is used here. We first look at a vanilla policy gradient method named REINFORCE. The use of baselines helps reduce variance and speed up learning.

Algorithm 2 REINFORCE with Baselines: Monte Carlo Policy Gradient Control

Inputs:
Number of episodes N
Differentiable policy parameterization $\pi(a|\boldsymbol{\theta})$
Set of bounded and monotone contracts A
Step sizes $a_\theta > 0, a_w > 0$
Initialize:
Policy parameters $\boldsymbol{\theta}$ to 0
Baseline Q to 0
for $n \leftarrow 1$ to N **do**
Draw action a following π to obtain reward r
$Q(a) \leftarrow Q(a) + \frac{r - Q(a)}{n}$
$\delta \leftarrow r - Q(a)$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + a_\theta \delta \nabla \ln \pi(a|\boldsymbol{\theta})$
end for

2.5.2 Actor-Critic Methods

Actor-Critic methods aim to improve upon vanilla policy-gradient methods such as REINFORCE described above by incorporating a "Critic", which estimates the action-value function Q. The "Actor" then updates the policy distribution in the direction suggested by the "Critic", similar to the vanilla policy gradient methods. TD3 is an actor-critic method that improves upon DDPG through clipped double Q-Learning, delayed policy update and target policy smoothing [7].

Algorithm 3 Twin Delayed DDPG (TD3)

Initialize:
Critic networks $Q_{\theta_1}, Q_{\theta_2}$ and actor network π_ϕ with random parameters θ_1, θ_2, ϕ
Target networks $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2, \phi' \leftarrow \phi$
Replay buffer β
for $t \leftarrow 1$ to T **do**
Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon, \epsilon \sim N(0, \sigma)$ and observe reward r and new state s'
Store transition tuple (s, a, r, s') in β
Sample mini-batch of N transitions (s, a, r, s') from β
$\tilde{a} \sim \pi_\phi(s) + \epsilon, \epsilon \sim \text{clip}(N(0, \tilde{\sigma}), -c, c)$
$y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a}))$
Update critics $\theta_i \leftarrow \text{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$
if t mod d **then**
Update ϕ by the deterministic policy gradient: $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
Update target networks:
$\theta'_i \leftarrow r\theta_i + (1 - r)\theta'_i$
$\phi' \leftarrow r\phi + (1 - r)\phi'$
end if
end for

3 High-Low Problem

In this section, we examine the case where there are 2 non-null outcomes, named as the high-low problem [9].

3.1 Problem Description

Each agent has 3 potential actions: to reject the contract, to put in low effort or to put in high effort. Correspondingly, there are 3 possible outcomes: the task is not completed, low quality or high quality. The task is not completed if and only if the contract is rejected by the agent. Otherwise, the probability distribution of the outcome is determined by the effort level of the agent. Let e_l and e_h represent low and high effort levels respectively.

Each agent has a type t which is drawn from an arbitrary distribution. The type is also a parameter for each agent's production function (probability of achieving a certain outcome level given a certain effort level) and cost function C (cost incurred for a certain effort level). Let p_l and p_h represent the probability of achieving a high quality outcome with a low effort level and a high effort level respectively. We assume that $p_l \leq p_h$.

Let θ_l and θ_h represent the payment the principal receives for achieving a low quality and high quality outcome respectively, where $\theta_l \leq \theta_h$. Let r_l and r_h represent the payment the principal awards the agent upon achieving a low quality and high quality outcome respectively. This forms the reward function that is outlined in the contract.

For simplicity, we assume both the principal and agent are risk neutral and their utility functions are linear as follows: $U = \theta, V = r$. The objective of the principal is to maximise his own expected utility (or payments equivalently). To do so, the principal controls the reward function, specifically r_l and r_h. The principal has no prior information about the environment and is unable to observe the agent type t nor the effort level e used by the agent. The principal is only able to observe the quality of the outcome achieved.

The expected utility function for the agent can be expressed as follows:

$$EV(0,t) = 0$$
$$EV(e_l,t|r_l,r_h) = p_l r_h + (1-p_l)r_l - C(e_l,t)$$
$$EV(e_h,t|r_l,r_h) = p_h r_h + (1-p_h)r_l - C(e_h,t)$$

The agent accepts the contract (i.e puts in a non-zero effort level) if and only if the expected payment is greater than the opportunity cost, expressed as $EV \geq 0$. This is known as the participation constraint. If the partici-

pation constraint is fulfilled, the agent simply chooses the effort level that maximises his own utility.

The expected utility function for the principal should the agent accept the contract:

$$EU(r_l, r_h) = P(e_h)(p_h(U_h - r_h) + (1 - p_h)(U_l - r_l)) + (1 - P(e_h))(p_l(U_h - r_h) + (1 - p_l)(U_l - r_l))$$

3.2 Real-World Example

An example of the High-Low problem can be found in crowdwork/crowdsourcing market, or otherwise known as the gig economy. which is one of the most important developments in the labour economy as of late. Employers are able to utilise digital platforms such as Upwork and Fiverr to outsource certain tasks to contractors/freelancers on these platforms. In order to do so, employers post these tasks and the respective amounts they are willing to pay, after which workers can choose to accept these tasks and receive payments after completion. These tasks are likely to be repetitive and periodic, removing the need to hire full-time workers.

It is clear that the principal-agent problem can easily occur due to the lack of information possessed by the employer (playing the role of the principal) about the workers (playing the role of the agent) that accept their tasks. Employers are likely unable to observe the workers' types nor their actions, and can only easily observe the quality of their output. The usage of additional bonuses for higher quality outcomes can be useful for employers to align incentives. Reinforcement learning in this online setting can thus be used to help dynamically adjust the quality-contingent payments for these tasks.

3.3 Analytical Solution

We attempt to derive an analytical solution (optimal r_l^* and r_h^*) for a specific version of the environment described above. We assume that the contracts are bounded and monotone. Formally, $0 \leq r_l \leq r_h \leq 1$.

Let $t \sim U(0, 1)$. We assume that agent type does not affect ability in production, and thus p_l and p_h are constants across all agent types. Instead, t can be interpreted as an inversely proportional indicator to the skill level of the agent, where agents with higher t have higher costs. The cost function is as follows: $C(e_l) = 0$, $C(e_h, t) = t$.

It is obvious from the cost function that the participation constraint is always fulfilled since using a low effort does not incur any cost. Thus, in the optimal solution, $r_l^* = 0$ since the principal does not need to incentivise the agent to select a low effort level.

The agent will only choose a high effort level if and only if:

$$EV(e_h,t) \geq EV(e_l,t)$$
$$p_h r_h + (1-p_h) r_l - C(e_h,t) \geq p_l r_h + (1-p_l) r_l - C(e_l,t)$$
$$p_h r_h - t \geq p_l r_h$$
$$t \leq r_h(p_h - p_l)$$

We can use the condition above to derive the probability of high effort $P(e_h|r_l,r_h)$. The optimal expected principal utility given $r_l^* = 0$ is thus as follows:

$$\begin{aligned} EU(r_l^*,r_h^*) &= P(e_h|r_h^*,r_l^*)(p_h(U_h - r_h^*) + (1-p_h)(U_l - r_l^*)) \\ &\quad + (1 - P(e_h|r_h^*,r_l^*))(p_l(U_h - r_h^*) + (1-p_l)(U_l - r_l^*)) \\ &= r_h^*(p_h - p_l)(p_h(U_h - r_h^*) + (1-p_h)U_l) + (1 - r_h^*(p_h - p_l))(p_l(U_h - r_h^*) + (1-p_l)U_l) \end{aligned}$$

Taking the first order condition w.r.t. r_h,

$$\begin{aligned} EU'(r_l^*,r_h^*) &= (p_h - p_l)(p_h(U_h - r_h^*) + (1-p_h)U_l) + r_h^*(p_h - p_l)(-p_h) \\ &\quad - (p_h - p_l)(p_l(U_h - r_h^*) + (1-p_l)U_l) + (1 - r_h^*(p_h - p_l))(-p_l) \\ &= (p_h - p_l)(p_h U_h - p_h r_h^* + (1-p_h)U_l - p_l U_h + p_l r_h^* - (1-p_l)U_l) \\ &\quad - r_h^*(p_h - p_l)^2 - p_l \\ &= (p_h - p_l)^2(U_h - U_l) - 2r_h^*(p_h - p_l)^2 - p_l \\ &= 0 \end{aligned}$$

Solving for the optimal r_h^*,

$$r_h^* = \frac{U_h - U_l}{2} - \frac{p_l}{2(p_h - p_l)^2}$$

3.4 Experimental Results

Using the analytical solution derived above, we are able to compare our learned solution to the optimal. For the experiments, we will use the following environment parameters: $p_l = 0.1, p_h = 0.8, \pi_l = 0.3, \pi_h = 1$. The optimal reward function would then be: $r_l^* = 0, r_h^* = 0.248$. The optimal expected utility would be: $EU(r_l^*,r_h^*) = 0.4$.

For the experiments conducted using the bandit algorithm and REINFORCE, the reward function is discretized into intervals of 0.05. On the other hand, TD3 permits the use of continuous action spaces, allowing the reward function to remain continuous.

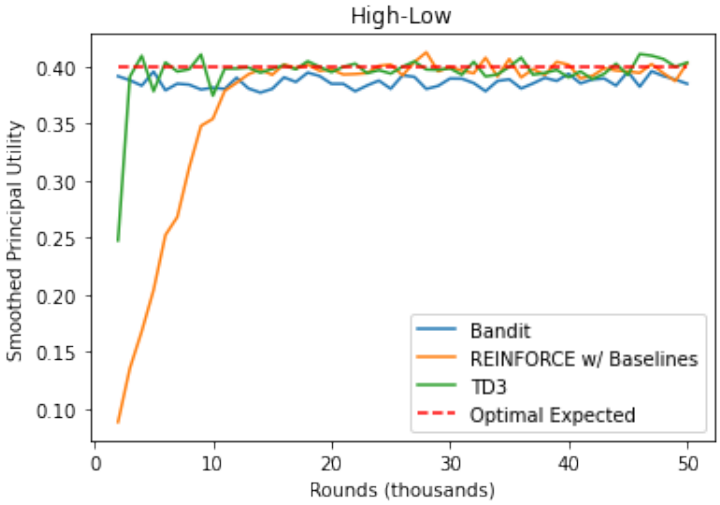

Figure 4: Convergence to the optimal expected utility

Figure 4 above shows the performance of the 3 algorithms compared against each other and the optimal. We can observe that all 3 algorithms performed well, managing to converge to the optimal solution to achieve near optimal average utility. Empirically, we can observe that TD3 and the bandit algorithm achieved relatively better sample efficiency, converging to near optimum utility in roughly 5000 rounds or less. On the other hand, the convergence using REINFORCE was significantly slower, taking roughly 10000 rounds. The utility achieved by the bandit algorithm seems to consistently fall slightly short of the optimal, which may be attributed to the constant epsilon epsilon-greedy exploration strategy resulting in sub-optimal actions. In contrast, the Boltzmann exploration strategy employed by the policy gradient methods is likely to concentrate the policy weights, leading to less exploration over time. Otherwise, the overall performance of all 3 algorithms in terms of principal utility are mostly similar.

4 Large Outcome Space

In this section, we examine the case where there are an infinite number of non-null outcomes with a continuous outcome variable.

4.1 Problem Description

Many aspects of this problem is similar to that in the High-Low problem described above in Section 3. The key difference is that both the payments received by the principal θ and effort levels e of the agent are now continuous instead of discrete. It would be impossible for the principal to manually determine the reward for each individual outcome as before. Instead, we assume that that the principal sets a linear reward function to achieve a generalized mapping from outcome to reward. A linear contract is not only easily interpretable for the parties involved, but it has also been shown that linear contracts are optimal under certain conditions [3].

The linear reward function f has the structure $f(\theta|m,a) = m\theta + a$, where θ is payment the principal receives (the observable output) and $0 \leq m \leq 1$. The parameter a can be interpreted as the percentage of the profits that the principal shares with the agent and a can be interpreted as the fixed payment that the principal pays the agent. It is of note that a can be a negative number, which would indicate that the agent would have to pay the principal instead to enter the contract. The learning objective would be to learn the parameters of the linear contract so as to maximise expected utility.

As before, each agent has a type t which is drawn from an arbitrary distribution. t acts a parameter for the agent's production function $\theta(e,t)$ and cost function $C(e,t)$. It is of note that the production function and/or the cost function may be stochastic. An effort level of 0 from the agent is equivalent to the rejection of the contract, and deterministically results in 0 payment for both the agent and the principal.

For simplicity, we assume both the principal and agent are risk neutral and their utility functions are linear as follows: $U = \theta, V = f(\theta|m,a)$. The objective of the principal is to maximise his own expected utility (or payments equivalently). The learning objective for the principal would be to learn the parameters of the linear contract so as to maximise expected utility. The principal has no prior information about the environment and is unable to observe the agent type t nor the effort level e used by the agent. The principal is only able to observe the output θ achieved.

The expected utility function for the agent can be expressed as follows:

$$EV(0,t) = 0$$
$$EV(e,t|m,a) = mE\theta_A(e,t) + a - C(e,t)$$

$E\theta_A(e,t)$ represents the expected output from the perspective of the agent, taking into account his effort level and type as inputs. The agent accepts the contract (i.e puts in a non-zero effort level) if and only if the expected payment is greater than the opportunity cost, expressed as $EV \geq 0$. This is known as the participation constraint. If the participation constraint is fulfilled, the agent simply chooses the effort level that maximises his own utility.

The expected utility function for the principal should the agent accept the contract:

$$EU(m,a) = (1-m) \cdot E\theta_P(m,a) - a$$

$E\theta_P(m,a)$ represents the expected output from the perspective of the principal, taking into account the parameters of the linear contract. Expected utility of the principal is 0 should the agent reject the contract.

4.2 Real-World Example

An example of the this problem can be found in sharecropping. Landowners (the principal) can rent land to tenants (the agent) in exchange for a share of the crops/profits in addition to possible fixed payments. This profit/risk sharing can be used to help align incentives in light of the problem of asymmetric information [4]. The online setting presented would however be more applicable for sharecropping with short-term/seasonal contracts.

4.3 Analytical Solution

We attempt to derive an analytical solution (optimal m^* and a^*) for a specific version of the environment described above.

Let $t \sim U(1,3)$. The production function of the agents is as follows: $\theta(e,t) = e + \epsilon$, where $\epsilon \sim N(0, \frac{t}{100})$. The cost function of the agents is as follows: $C(e,t) = \frac{e^2}{t}$. Agents with greater skills, indicated by higher t, have a lower variation in their production and also have lower costs.

Assuming that the contract is accepted, the agent's expected reward function is as follows: $ER(e,t) =$

$f(E\theta_A(e,t)|m,a) = me + a$. The agent's expected utility function can then be constructed as: $EV(e,t) = ER(e,t) - C(e,t) = me + a - \frac{e^2}{t}$. We can then derive the optimal effort level e^* for the agent using the first order condition.

Taking the first order condition w.r.t. e,

$$\begin{aligned} EV'(e,t) &= m - 2et \\ &= 0 \\ e^* &= \frac{mt}{2} \end{aligned}$$

Participation Constraint:

$$\begin{aligned} EV(e,t) &\geq 0 \\ me + a - \frac{e^2}{t} &\geq 0 \\ \frac{m^2 t}{2} + a - \frac{m^2 t}{4} &\geq 0 \\ \frac{m^2 t}{4} &\geq -a \\ t &\geq -\frac{4a}{m^2} \end{aligned}$$

The participation constraint is met and the agent puts in a positive effort level if and only if the above inequality holds for the agent. We can thus derive the probability that the agent accepts the contract, denoted as $P(e > 0)$. $P(e > 0) = P(t \geq -\frac{4a}{m^2}) = \frac{2a}{m^2} + \frac{3}{2}$, where $0 \leq m \leq 1$ and $-\frac{3m^2}{4} \leq a \leq -\frac{2m^2}{4}$.

Assuming that the agent accepts the contract, we can construct the principal's expected output function as follows: $E\theta_P(m,a) = E(e^*|m,a) = E(\frac{mt}{2}) = m$.

The principal's expected utility function can then be expressed as:

$$\begin{aligned} EU(m,a) &= P(e > 0)[(1-m) \cdot E\theta_P(m,a) - a] \\ &= (\frac{2a}{m^2} + \frac{3}{2})[(1-m)m - a] \end{aligned}$$

Because the probability is discontinuous, we are unable to utilize partial derivatives to obtain the optimal contract parameters. Instead, we use an optimizer to obtain the optimal parameters $m^* = \frac{2}{3}$ and $a^* = -\frac{1}{9}$. The optimal expected principal utility would then be $\frac{1}{3}$. This tells us that it would be optimal for the principal to share $\frac{2}{3}$ of his profits and charge a $\frac{1}{9}$ fixed payment for agents to enter the contract.

4.4 Experimental Results

For the experiments conducted using the bandit algorithm and REINFORCE, the reward function is discretized into intervals of 0.05. On the other hand, TD3 permits the use of continuous action spaces, allowing the reward function to remain continuous.

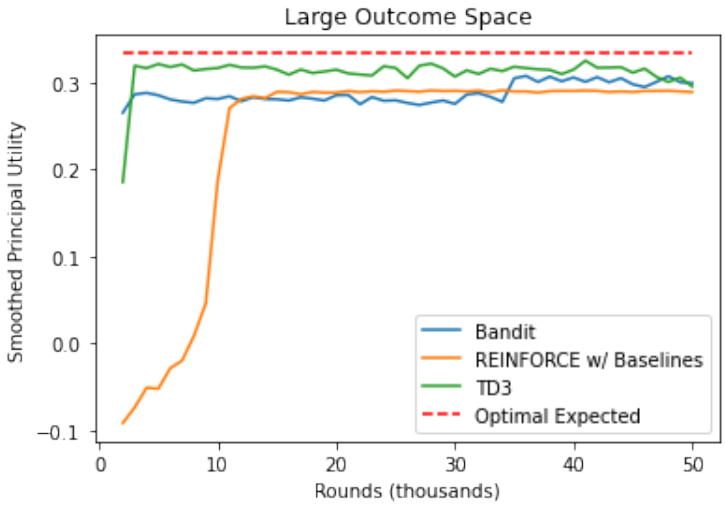

Figure 5: Convergence to the optimal expected utility

Figure 5 above shows the performance of the 3 algorithms compared against each other and the optimal. The comparison between the algorithms also remain largely the same, with the bandit algorithm and TD3 having superior empirical sample efficiency and converging at a substantially faster rate than REINFORCE. The advantages of TD3 is more prominent here, achieving the best performance in terms of principal utility. Unlike the performance shown in the High-Low problem, all 3 algorithms fell slightly short of the optimal. The failure to achieve the optimal expected utility could be attributed to the increased difficulty of the problem due to the large number of non-null outcomes compared to the 2 of the High-Low problem.

5 Conclusion and Future Research

In this paper, we explored the application of reinforcement learning algorithms in online principal agent problems which feature characteristics of moral hazard (hidden action) as well as adverse selection (hidden type). These problems were split into two main categories. The key difference between the two is the size of the outcome space, with one having a small number of non-null outcomes while the other having a large number or possibly infinite number of non-null outcomes, which in turn affects the potential structure of the contracts.

We first modelled these problems and produced analytical solutions to obtain theoretical optimums. We then expressed these problems as MDPs, allowing us to apply reinforcement learning to solve the problem in a model-free manner. The performance obtained using reinforcement learning were then compared to the theoretical optimums. The results showed that near optimal performance can be achieved even with the use of mostly standard reinforcement learning algorithms, showing that the use of reinforcement learning could prove useful in real-world principal-agent problems where explicit modelling is difficult. An interesting observation that we can also see is that the bandit algorithm achieved relatively decently performance in terms of the empirical utility obtained and sample efficiency, despite its simplicity and low computational cost.

Some simplifying assumptions were made to limit the scope of this paper. Thus, there is definitely much room for further research and the outline for some potential avenues will be detailed below.

A key assumption, though made implicitly, is that there is possibly an infinite number of agents drawn from the chosen distribution and it would be difficult for the principal to do better than the average/expected utility. Thus, the MDP constructed for our problem did not include the presence of state and state transitions. However, in reality, the number of agents a principal encounters would be limited and it would be highly probable for a principal to interact with the same agent multiple times. The information gained from previous interactions should then be able to help the principal make a more informed decision. A possible method to achieve this would be to use the agent's trajectory/previous interactions as a means of agent identification [19]. The trajectory can then be used as an observable to construct a mapping from the agent to a reward function/contract.

The risk profiles of both the principal and the agent were assumed to be risk-neutral, but this assumption can also be relaxed to introduce other risk profiles, potentially altering the utility functions of both parties. We also assumed the agent to have full information and to be fully rational, which may not hold true in reality as explored by Bihary and Kerényi [20] who introduced the idea of dynamic participation constraints. Lastly, the use of social learning, where principals can learn from each other and their past experiences, may offer additional advantages to pure reinforcement learning [10].

6 Appendix

More information regarding the code and implementation details can be found in the public GitHub repository for this project which can be found here.

References

[1] J.-J. Laffont and D. Martimort, *The Theory of Incentives: The Principal-Agent Model*. Princeton, NJ, USA: Princeton University Press, 2001.

[2] J. Cvitanic and J. Zhang, *Contract Theory in Continuous-Time Models*. Springer Science & Business Media, 01 2013.

[3] B. Holmstrom and P. Milgrom, "Aggregation and linearity in the provision of intertemporal incentives," *Econometrica*, vol. 55, no. 2, pp. 303–328, 1987. [Online]. Available: http://www.jstor.org/stable/1913238

[4] J. E. Stiglitz, "Incentives and risk sharing in sharecropping," *The Review of Economic Studies*, vol. 41, no. 2, pp. 219–255, 1974. [Online]. Available: http://www.jstor.org/stable/2296714

[5] G. D. Carroll, "Robustness and linear contracts," *The American Economic Review*, vol. 105, pp. 536–563, 2015.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[7] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018.

[8] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

[9] C.-J. Ho, A. Slivkins, and J. W. Vaughan, "Adaptive contract design for crowdsourcing markets: Bandit algorithms for repeated principal-agent problems," 2015.

[10] J. Arifovic and A. Karaivanov, "Learning by doing vs. learning from others in a principal-agent model," *Journal of Economic Dynamics and Control*, vol. 34, no. 10, pp. 1967–1992, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0165188910000874

[11] V. Conitzer and N. Garera, "Learning algorithms for online principal-agent problems (and selling goods online)," *ACM International Conference Proceeding Series*, vol. 148, pp. 209–216, 01 2006.

[12] R. Bellman, "A markovian decision process," *Indiana Univ. Math. J.*, vol. 6, pp. 679–684, 1957.

[13] M. Otterlo and M. Wiering, "Reinforcement learning and markov decision processes," *Reinforcement Learning: State of the Art*, pp. 3–42, 01 2012.

[14] R. Bellman, *Dynamic Programming*, ser. Dover Books on Computer Science Series. Dover Publications, 2003.

[15] M. Coggan, "Exploration and exploitation in reinforcement learning," *Fourth International Conference on Computational Intelligence and Multimedia Applications*, pp. 1–44, 2001.

[16] A. Maroti, "RBED: reward based epsilon decay," *CoRR*, vol. abs/1910.13701, 2019. [Online]. Available: http://arxiv.org/abs/1910.13701

[17] A. Slivkins, "Introduction to multi-armed bandits," *CoRR*, vol. abs/1904.07272, 2019. [Online]. Available: http://arxiv.org/abs/1904.07272

[18] D. P. de Farias and N. Megiddo, "How to combine expert (and novice) advice when actions impact the environment?" in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. K. Saul, and B. Schölkopf, Eds. MIT Press, 2004, pp. 815–822.

[19] N. C. Rabinowitz, F. Perbet, H. F. Song, C. Zhang, S. M. A. Eslami, and M. Botvinick, "Machine theory of mind," 2018.

[20] Z. Bihary and P. Kerényi, "Gig economy: A dynamic principal-agent model," 2019.