Improving Convergence of Deterministic Policy Gradient Algorithms in Reinforcement Learning

Final Report

Riashat Islam

Supervisor: Professor John Shawe-Taylor
Second Assessor: Professor Miguel Rodrigues

March 2015
DECLARATION

I have read and understood the College and Department’s statements and guidelines concerning plagiarism.

I declare that all material described in this report is all my own work except where explicitly and individually indicated in the text. This includes ideas described in the text, figures and computer programs.

Name: ....................................

Signature: ..............................

Date: .................................
Improving Convergence of Deterministic Policy Gradient Algorithms in Reinforcement Learning

by

Riashat Islam

Submitted to the Department of Electronic and Electrical Engineering in partial fulfillment of the requirements for the degree of

Bachelor of Engineering

at the

March 2015

Author

Department of Electronic and Electrical Engineering

March 27, 2015

Accepted by

John Shawe-Taylor

Director, Centre for Computational Statistics and Machine Learning, Department of Computer Science
Improving Convergence of Deterministic Policy Gradient Algorithms in Reinforcement Learning

by

Riashat Islam

Submitted to the Department of Electronic and Electrical Engineering on March 27, 2015, in partial fulfillment of the requirements for the degree of Bachelor of Engineering

Abstract

Policy gradient methods in reinforcement learning directly optimize a parameterized control policy with respect to the long-term cumulative reward. Stochastic policy gradients for solving large problems with continuous state and action spaces have been extensively studied before. Recent work also showed existence of deterministic policy gradients, which has a model-free form that follows the gradient of the action-value function. The simple form of the deterministic gradient means it can be estimated more efficiently. We consider the convergence of deterministic policy gradient algorithms on practical tasks.

In our work, we consider the issue of local optima convergence of deterministic policy gradient algorithms. We propose a framework of different methods to improve and speed up convergence to a good locally optimal policy. We use the off-policy actor-critic algorithm to learn a deterministic policy from an exploratory stochastic policy. We analyze the effect of stochastic exploration in off-policy deterministic gradients to improve convergence to a good local optima. We also consider the problem of fine tuning of parameters in policy gradient algorithms to ensure optimal performance. Our work attempts to eliminate the need for the systematic search over learning rate parameters that affects the speed of convergence. We propose an adaptive step size policy gradient algorithms to automatically adjust learning rate parameters in deterministic policy gradient algorithms. Inspired from work in deep neural networks, we also introduce momentum-based optimization techniques in deterministic policy gradient algorithms. Our work considers combinations of these different methods to address the issue of improving local optima convergence and also to speed up the convergence rates in deterministic policy gradient algorithms. We show results of our algorithms on standard reinforcement learning benchmark tasks, such as Toy, Grid World and Cart Pole MDP.

We demonstrate the effect of off-policy exploration on improving convergence in deterministic policy gradients. We also achieve optimal performance of our algorithms with careful fine tuning of parameters, and then illustrate that using automatically adjusted learning rates, we can also obtain optimal performance of these algorithms. Our results are the first to consider deterministic natural gradients in an experimental framework. We demonstrate that using optimization techniques that can eliminate the need for fine tuning of parameters, and combining approaches that can speed up convergence rates, we can improve local optimal convergence of deterministic policy gradient algorithms on reinforcement learning benchmark tasks.
Acknowledgments

First and foremost, I would like to thank my supervisor Professor John Shawe-Taylor, who has been an unswerving source of inspiration for me. His knowledgeable advice helped me to explore exhilarating areas of machine learning, and helped me work on a project of my interest. I would also like to thank my mentor Dr Guy Lever, without whose support and patience, I would not have this project see this day. Thank you Guy, for suggesting me to work in this direction of reinforcement learning, for giving me your valuable time while conducting our discussions and for fuelling my avid curiosity in this sphere of machine learning. During the stretch of this project, there were so many times I felt incensed due to the obscure results and so many botched attempts to get a proper output, and every time I would find Guy assist me and help me regain the momentum of the work. I am also grateful to Professor Miguel Rodrigues for being my supervisor acting on behalf of the UCL EEE Department. I also thank Professor Paul Brennan and Professor Hugh Griffiths who provided me useful advice acting as my tutors during the first two years of my undergraduate degree.

I am indebted for being able to spend such an amazing time at UCL, working on this project. I am lucky to have met all the amazing friends and colleagues during my time at UCL. My friends Baton, Francesco, Omer, Temi and Timothy have always given me their enormous support during my undergraduate degree.

I would like to thank my parents Siraj and Shameem - the most important people in my life who have always put my wellbeing and academic interests over everything else, I owe you two everything. My life here in UK would not have felt like home without the amazing relatives that I have here - thank you, Amirul, Salma, Rafsan and Tisha for always being there and supporting me through all these years of my undergraduate degree. I would like to thank Tasnova for her amazing support and care, and for bringing joy and balance to my life. Thanks to my friends Rashik, Sadat, Riyasat, Mustafa, Raihan, Intiaz and Mahir for standing by my side over the long years.

I am always thankful to Almighty Allah for the opportunities and successes He has given me in this life, I would not be what I am today without His blessings.
This undergraduate project has been supervised and examined by a Committee of the Department of Computer Science and Department of Electronic and Electrical Engineering as follows:

Professor John Shawe-Taylor .......................................................... Project Supervisor
Professor of Computational Statistics and Machine Learning, Department of Computer Science

Dr Guy Lever .......................................................... Project Mentor
Postdoctoral Researcher, Centre for Computational Statistics and Machine Learning, Department of Computer Science

Professor Miguel Rodigues .......................................................... Second Assessor,
Professor of Electronic and Electrical Engineering
## Contents

1 Introduction
  1.1 Aims ......................................................... 10  
  1.2 Reinforcement Learning ..................................... 10  
  1.3 Reinforcement Learning In Practice ......................... 11  
  1.4 Overview ................................................ 12  

2 Background
  2.1 Basic Concept ............................................. 13  
  2.2 Markov Decision Processes .................................. 13  
  2.3 Learning Framework ....................................... 14  
  2.4 Value-Based Reinforcement Learning ....................... 15  
  2.5 Policy Gradient Reinforcement Learning ................... 15  
    2.5.1 Stochastic Policy Gradient ............................. 16  
    2.5.2 Deterministic Policy Gradient ......................... 17  
  2.6 Function Approximation .................................... 17  
    2.6.1 Temporal Difference Learning ......................... 18  
    2.6.2 Least-Squares Temporal Difference Learning .......... 18  
  2.7 Off-Policy Actor-Critic Algorithms ....................... 19  
  2.8 Exploration in Deterministic Policy Gradient Algorithms 19  
  2.9 Optimization and Parameter Tuning In Policy Gradient Methods 20  
  2.10 Adaptive Step-Size for Policy Gradients .................. 21  
    2.10.1 Adaptive Step Size With Average Reward Metric .... 21  
  2.11 Improving Convergence of Deterministic Policy Gradients 22  
  2.12 Natural Gradients ...................................... 23  
    2.12.1 Stochastic Natural Gradient .......................... 23  
    2.12.2 Deterministic Natural Gradient ....................... 24  
  2.13 Momentum-based Gradient Ascent .......................... 24  
    2.13.1 Classical Momentum ................................... 24  
    2.13.2 Nesterov Accelerated Gradient ....................... 24  

3 Learning System Structural Framework ....................... 26  
  3.1 Parameterized Agent Class ................................ 26
### 3.2 MDP Class
- 3.2.1 Toy MDP .................................................. 27
- 3.2.2 Grid World MDP ....................................... 28
- 3.2.3 Cart Pole MDP .......................................... 28

### 3.3 Other Implementations for Developing Software ........................................ 28

### 3.4 Experimental Details ................................................................. 29

### 4 Experimental Results and Analysis .................................................. 30
- 4.1 Policy Gradients ......................................................... 30
  - 4.1.1 Stochastic and Deterministic Policy Gradient .................. 30
  - 4.1.2 Local Optima Convergence .................................... 31
- 4.2 Exploratory Stochastic Off-Policy in Deterministic Gradients ............ 32
  - 4.2.1 Effect of Exploration ........................................ 33
- 4.3 Resolving Issue of Fine Tuning Parameters in Policy Gradient Algorithms .... 34
  - 4.3.1 Choosing Optimal Step Size parameters ...................... 34
  - 4.3.2 Adaptive Policy Gradient Methods With Average Reward Metric .... 35
- 4.4 Natural Policy Gradients To Speed Up Convergence ......................... 36
  - 4.4.1 Natural Stochastic Policy Gradient .......................... 36
  - 4.4.2 Natural Deterministic Policy Gradient ..................... 36
  - 4.4.3 Comparing Convergence of Natural and Vanilla Policy Gradients ... 37
  - 4.4.4 Natural Deterministic Gradient With Exploration ............ 38
- 4.5 Momentum-based Policy Gradients To Speed Up Convergence ................. 39
  - 4.5.1 Classical Momentum-based Policy Gradient .................. 39
  - 4.5.2 Nesterov’s Accelerated Gradient-based Policy Gradients ....... 40
  - 4.5.3 Comparing Vanilla, Natural and Momentum-based Policy Gradients . 41
- 4.6 Improving Convergence of Deterministic Policy Gradients .................. 43
  - 4.6.1 Vanilla, Natural and Classical Momentum .................... 43
  - 4.6.2 Vanilla, Natural and Nesterov’s Accelerated Gradient ......... 44
  - 4.6.3 Adaptive Step Size Policy Gradients ........................ 44
  - 4.6.4 Improving Convergence ....................................... 45
  - 4.6.5 Convergence of Stochastic and Deterministic Policy Gradients .... 47
- 4.7 Extending To Other Benchmark Tasks .......................................... 48
  - 4.7.1 Grid World MDP ............................................. 48
  - 4.7.2 Cart Pole MDP ................................................ 49
- 4.8 Modified and Adaptive Initial Step Size ...................................... 49

### 5 Discussion .................................................
- 5.1 Effect of Exploration ............................................. 52
- 5.2 Resolving Issue of Learning Rate ..................................... 53
- 5.3 Improving Convergence Rates ....................................... 54
- 5.4 Generalizing To Other Benchmark MDPs .................................. 55
- 5.5 Future Work and Open Problems ....................................... 55
# List of Figures

1-1 Reinforcement Learning in Practice: Playing Atari Games [Mnih et al., 2013] 11
1-2 Reinforcement Learning in Practice: Autonomous Inverted Helicopter Flight via Reinforcement Learning [Ng et al., 2004] 11

4-1 Stochastic Policy Gradient on Toy MDP 31
4-2 Deterministic Policy Gradient on Toy MDP 31
4-3 Stochastic and Deterministic Policy Gradient on Grid World MDP 32
4-4 Convergence to Local Optima - Deterministic Policy Gradient Toy MDP 33
4-5 Effect of Exploration on Vanilla Deterministic Policy Gradient - Toy MDP With Multiple Local Reward Blobs 34
4-6 Effect of Step Size parameters on convergence rates 34
4-7 Effect of ε parameter using Adaptive Vanilla Deterministic Policy Gradient 36
4-8 Natural Stochastic Policy Gradient on Toy MDP 37
4-9 Natural Deterministic Policy Gradient on Toy MDP 37
4-10 Comparing Natural and Vanilla Stochastic and Deterministic Policy Gradients 38
4-11 Effect of Exploration on Natural Deterministic Policy Gradient - Toy MDP With Multiple Local Reward Blobs 38
4-12 Stochastic Policy Gradient With Classical Momentum based gradient ascent - Effect of µ and ε parameters 40
4-13 Deterministic Policy Gradient With Classical Momentum Gradient Ascent - Effect of µ and ε parameters 40
4-14 Deterministic Policy Gradient With Nesterov’s Accelerated Gradient 41
4-15 Comparing Vanilla, Natural and Classical Momentum-based Policy Gradients 42
4-16 Comparing Vanilla, Natural and Nesterov’s Accelerated Gradient-based policy gradients 42
4-17 Comparing Deterministic Vanilla, Natural and Classical Momentum policy gradients 43
4-18 Comparing DPG Vanilla, Natural and Nesterov Accelerated Gradient policy gradients 44
4-19 Comparing Adaptive Deterministic Policy Gradient Algorithms 45
4-20 Effect of Adaptive Step Size on Convergence of Stochastic and Deterministic Policy Gradients 46
Chapter 1

Introduction

This undergraduate thesis (project report) investigates deterministic policy gradient algorithms in reinforcement learning.

1.1 Aims

In this project, we analyze local optimal convergence of deterministic policy gradient algorithms on practical reinforcement learning tasks. We address how the convergence to an optimal policy is dependent on stochastic behaviour exploration in the off-policy actor-critic algorithm. We consider using different optimization techniques, to analyze the dependence of the speed of convergence on learning rates in stochastic gradient ascent. Our work also suggests an automatically adjustable learning rate method to eliminate fine tuning of parameters on practical reinforcement learning benchmark tasks; we show that using adjustable learning rate parameters, we can eliminate the problem of fine tuning in policy gradient algorithms on practical reinforcement learning tasks. We demonstrate the performance of natural deterministic policy gradients in an experimental framework for the first time. Our work also proposes using a momentum-based optimization technique in the policy gradient framework for the first time. Throughout this project, we address the issue of speed of convergence and how we can improve local optimal convergence of deterministic policy gradient algorithms.

In this chapter, we first give an overall review of the reinforcement learning framework considered in our work and then give an outline for the rest of the undergraduate thesis.

1.2 Reinforcement Learning

Reinforcement learning (RL) is concerned with the optimal sequential decision making problem in biological and artificial systems. In reinforcement learning, the agent makes its decisions and takes actions based on observations from the world (such as data from robotic sen-
sors) and receives a reward that determines the agent’s success or failure due to the actions taken. The fundamental question in reinforcement learning is how can an agent improve its behaviour given its observations, actions and the rewards it receives [Kaelbling et al., 1996], [Sutton and Barto, 1998]. The goal in reinforcement learning is for the agent to improve its behaviour policy by interacting with its environment and learn from its own experience.

1.3 Reinforcement Learning In Practice

Figure 1-1: Reinforcement Learning in Practice: Playing Atari Games [Mnih et al., 2013]

Figure 1-2: Reinforcement Learning in Practice: Autonomous Inverted Helicopter Flight via Reinforcement Learning [Ng et al., 2004]

Reinforcement learning (RL) has had success in application areas ranging from computer games to neuroscience to economics. [Tesauro, 1994] has used RL for learning games of skill, while recent work from [Mnih et al., 2013] considered using RL to play Atari 2600 games from the Arcade Learning Environment and showed that RL agent surpasses human experts in playing some of the Atari games having no prior knowledge. [Silver, 2009] considered RL algorithms in the Game of Go, and achieved human master level. Reinforcement learning has also been widely used in robotics for robot controlled helicopters that can fly stunt manoeuvres [Abbeel et al., 2007] [Bagnell and Schneider, 2001]; in neuroscience, RL has been used to model the human brain [Schultz et al., 1997], and in psychology, it has been used to predict human behaviour [Sutton and Barto, 1990].

[Mnih et al., 2013] from figure 1-1 has exceeded human level game playing performance by using reinforcement learning to play Atari games, while [Ng et al., 2004] from figure 1-2 has
demonstrated the use of apprenticeship and reinforcement learning for making a helicopter learn to do stunt manoeuvres by itself. Our inspiration to work in this undergraduate project comes from such practical examples of reinforcement learning.

1.4 Overview

We present the outline of this project report in this section. Detailed explanation is provided in a later section of background literature.

• Chapter 2
  
  In Chapter 2, we provide a literature review of the key concepts in reinforcement learning. We discuss the basic approaches to policy gradient reinforcement learning, including discussion of function approximators, exploration and off-policy actor-critic architecture that we considered for our policy gradient algorithms. We include a brief background of the different optimization techniques that we used, including momentum-based gradient ascent approaches. In Chapter 2, we also provide an explanation of the importance of step size or learning rate parameter tuning in policy gradient methods. We then give a brief introduction of the adaptive step size policy gradient methods that we consider in our work.

• Chapter 3
  
  In Chapter 3, we provide an overview of the learning system experimental framework that we consider. We describe the software modules that implements the learning algorithms. In this chapter we provide an overview of the Markov Decision Processes (MDPs) that are standard benchmark reinforcement learning tasks that we considered in this project.

• Chapter 4
  
  Chapter 4 presents our experimental results that we obtained using the methods and approaches considered in Chapters 2 and 3. In this section, we provide the results we obtain using each of the policy gradient methods we considered; results include analysing the differences in performance using different optimization techniques, and the importance of different learning rate methods towards improving convergence to a locally optimal policy.

• Chapter 5
  
  Chapter 5 discusses the experimental results that we obtain, and whether the results are compatible with our expected theoretical background. Furthermore, we summarize our accomplishments in this project in this section.

• Chapter 6
  
  Finally, we provide a conclusion to the project with general discussion, our contributions and achievements in our work.
Chapter 2

Background

2.1 Basic Concept

We recall some of the basic concepts associated with reinforcement learning and control problems in which an agent acts in a stochastic environment. Reinforcement learning is a sequential decision making problem in which an agent chooses actions in a sequence of time steps in order to maximize the cumulative reward. The decision making entity is the agent and everything outside the agent is its environment. At each step, the agent receives observations from the environment and executes an action according to its behaviour policy. Given this action, the environment then provides a reward signal to indicate how well the agent has performed. The general purpose goal of reinforcement learning is to maximize the agent’s future reward given its past experience. The problem is modelled as a Markov Decision Process (MDP) that comprises of a state and action space, an initial state distribution that satisfies the Markov property that $P(s_{t+1}, r_{t+1}|s_1, a_1, r_1, \ldots s_t, a_t, r_t) = P(s_{t+1}, r_{t+1}|s_t, a_t)$, and a reward function.

2.2 Markov Decision Processes

Markov Decision Processes (MDPs) in reinforcement learning are used to describe the environment with which the agent interacts. In our framework, we consider fully observable MDPs. The MDP comprises of state and action spaces and is defined as a tuple consisting of $(S, D, A, P_{sa}, \gamma, R)$ consisting of :

- $S$: Set of possible states of the world
- $D$: An initial state distribution
- $A$: Set of possible actions
- $P_{sa}$: The state transition distributions
- $\gamma$: A constant in $[0, 1]$ that is called the discount factor
• R: A reward function that considers $S \times A \rightarrow R$

In an MDP, the agent starts from an initial state $s_0$ that is drawn from the initial state distribution. The agent moves around in the environment, and at each time step, it takes an action for which it moves to the next successor state $s_{t+1}$. By taking actions at each step, the agent traverses a series of states $s_0, s_1, s_2, \ldots$ in the environment such that the sum of discounted rewards as the agent moves is given by:

$$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$$ (2.1)

The goal of the agent is to choose actions $a_0, a_1, \ldots$ over time such that such that it can maximize the expected value of the rewards as given in equation 2.1. The agent’s goal is to learn a good policy such that during each time step in a given state, the agent can take a stochastic or deterministic action $\pi(s)$ which will obtain a large expected sum of rewards:

$$E_{\pi}[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots]$$ (2.2)

### 2.3 Learning Framework

In our work, we consider model-free reinforcement learning methods where the agent learns directly from experience and do not have any prior knowledge of the environment dynamics. A policy is used to select actions in a MDP, and we consider both stochastic $a \sim \pi_\theta(a|s)$ and deterministic $a = \pi_\theta(s)$ policies, where the basic idea is to consider parametric policies, and we can choose an action $a$ (both stochastically and deterministically) according to a parameter vector $\theta$, where $\theta \in \mathbb{R}^n$ is a vector of $n$ parameters. Using its policy, the agent can interact with the MDP to give a trajectory of states, actions and rewards $h_{1:T} = s_1, a_1, r_1, \ldots s_T, a_T, r_T$. We use the return $r^T_t$ which is the total discounted reward from time step $t$ onwards, $r^T_t = \sum_{k=t}^{\infty} \gamma^{k-t} r(s_k, a_k)$ where $0 < \gamma < 1$. The agent performance objective is given by $J(\pi) = E[r^T_1|\pi]$. In our work, we consider continuous state and action spaces, and so we use the frequently used Gaussian policies $\pi_\theta(a|s) = N(\phi(a, s)^T \theta_1, \theta_2$ with an exploration parameter $\theta_2$.

**Expected Return**

The goal of algorithms in reinforcement learning is to maximize the long-term expected return of a policy $\pi_\theta$ with respect to the expected return.

$$J(\theta) = Z_\gamma E \sum_{k=0}^H \gamma^k r_k$$ (2.3)

where $H$ is the planning horizon, $\gamma \in [0, 1]$ is the discount factor and $Z_\gamma$ is a normalization factor. In our work we consider policy gradient algorithms that are advantageous for dealing with large continuous state and action spaces. The idea behind policy gradient algorithms is to adjust the parameters $\theta$ of the policy in the direction of the performance gradient.
\[ \nabla_{\theta} J(\pi_{\theta}) \] so as to maximize \[ J(\pi_{\theta}) \]. We include a detailed discussion of policy gradient algorithms in later sections.

### 2.4 Value-Based Reinforcement Learning

Reinforcement learning uses value functions that can measure the long-term value of following a particular decision making policy. When the agent follows a policy \( \pi(s) \) and is currently at state \( s \), the value function \( V^\pi(s) \) is given by:

\[
V^\pi(s) = E_{\pi}[R_t | s_t = s] \tag{2.4}
\]

and the action value function \( Q^\pi(s, a) \) is the reward that the agent should expect when it selects action \( a \) in state \( s \), and follows a policy \( \pi \) afterwards such that:

\[
Q^\pi(s, a) = E_{\pi}[R_t | s_t = s, a_t = a] \tag{2.5}
\]

where \( E_{\pi} \) denotes the expectation over the episodes of the agent’s experiences that are gathered by following policy \( \pi \). We consider model-free reinforcement learning such that the value or action-value function is updated using a sample backup. This means, that at each time step of the agent’s experience, an action is sampled from the agent’s policy, while we get the successor state and the reward sampled from the environment. The action-value function of the agent is updated using the agent’s experience as it interacts with the environment.

### 2.5 Policy Gradient Reinforcement Learning

Policy gradient algorithms are used in reinforcement learning problems, and are techniques that rely upon optimizing parameterized policies with respect to the long-term cumulative reward. These methods directly optimize a parametrized control policy by gradient ascent such that the optimal policy will maximize the agent’s average reward per time-step.

Policy gradient methods are particularly useful because they are guaranteed to converge at least to a locally optimal policy and can handle high dimensional continuous states and actions, unlike value-based methods where local optima convergence is not guaranteed. However, one of the major concerns is that global optima are not easily obtained by policy gradient algorithms [Peters and Bagnell, 2010].

In this project, we consider both stochastic [Sutton et al., 2000] and deterministic [Silver et al., 2014] policy gradient algorithms for analyzing convergence of deterministic policy gradient algorithms on benchmark MDPs.
Policy Gradient Theorem
The fundamental result underlying all our algorithms is the policy gradient theorem. We consider the stochastic policy gradient theorem [Sutton et al., 2000], where for stochastic policies, the policy gradient can be written as:
\[ \nabla_\theta J(\theta) = E_{\rho^\pi, a \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(s, a)Q^{\pi_\theta}(s, a)] \] (2.6)
and the deterministic policy gradient theorem [Silver et al., 2014]:
\[ \nabla_\theta J(\theta) = E_{s \sim \rho^\mu}[\nabla_\theta \mu_\theta(s)\nabla_a Q^\mu(s, a)|a = \mu(s)] \] (2.7)

Gradient Ascent in Policy Space
Policy gradient methods follow the gradient of the expected return
\[ \theta_{k+1} = \theta_k + \alpha_k \nabla_\theta J(\pi_\theta)|_{\theta = \theta_k} \] (2.8)
where \( \theta_k \) denotes the parameters after update \( k \) with initial policy parameters \( \theta_0 \) and \( \alpha_k \) denotes the learning rate or step-size.

2.5.1 Stochastic Policy Gradient
As mentioned previously, the goal of policy gradient algorithms is to search for the local maximum in \( J(\theta) \) by ascending the gradient of \( J(\theta) \) of the policy, such that, \( \Delta \theta = \alpha \nabla_\theta J(\theta) \), and \( \alpha \) is the step size parameter for gradient ascent. We consider a Gaussian policy where the mean of the Gaussian policy is a linear combination of state features \( \mu(s) = \phi(s)^T \theta \).

In the expression of stochastic policy gradient theorem defined previously in equation 2.6, the expression for the \( \nabla_\theta \log \pi_\theta(s, a) \), which is called the score function is given as:
\[ \nabla_\theta \log \pi_\theta(s, a) = \frac{(a - \mu(s))\phi(s)}{\sigma^2} \]
From equation 2.6, the \( Q^{\pi_\theta}(s, a) \) is then approximated using a linear function approximator, where the critic estimates \( Q^w(s, a) \), and in the stochastic policy gradient, the function approximator is compatible such that
\[ Q^w(s, a) = \nabla_\theta \log \pi_\theta(a|s)^T w \] (2.9)
where the \( w \) parameters are chosen to minimize the mean squared error between the true and the approximated action-value function.
\[ \nabla_\theta J(\theta) = E_{\rho^\pi, a \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(s, a)Q^w(s, a)] \] (2.10)
As discussed later in section 2.7, we use the off-policy stochastic actor-critic in the stochastic gradients setting. We get the expression for the stochastic off-policy policy gradient.
[Degris et al., 2012b] as shown in equation 2.11.

\[
\nabla_{\theta} J(\pi_{\theta}) = E_{s \sim \rho, a \sim \beta}[\pi_{\theta}(a|s)\nabla_{\theta} \log \pi_{\theta}(a|s) Q^\pi(s, a)]
\]

(2.11)

where \( \beta (a|s) \) is the behaviour off-policy that is distinct from the parameterized stochastic policy \( \beta (a|s) = \pi_{\theta}(a|s) \).

### 2.5.2 Deterministic Policy Gradient

[Silver et al., 2014] considered extending the stochastic gradient framework into the deterministic policy gradients using a similar approach, that is analogous to the stochastic policy gradient theorem. In case of deterministic gradients, considering continuous action spaces, the idea is to move the policy in the direction of the gradient of \( Q \) rather than globally maximising \( Q \). In other words, for deterministic policy gradients, the policy parameters are updated as follows:

\[
\theta^{k+1} = \theta^k + \alpha E_{s \sim \rho, a \sim \mu_{\theta}}[\nabla_{\theta} Q^\mu_{\theta}(s, \mu_{\theta}(s))] 
\]

(2.12)

The function approximator \( Q^w(s, a) \) is again compatible with the deterministic policy \( \mu_{\theta}(s) \) if :

\[
\nabla_{a} Q^w(s, a)|_{a=\mu_{\theta}(s)} = \nabla_{\theta} \mu_{\theta}(s)^T w
\]

(2.13)

We get the final expression for the deterministic policy gradient theorem using the function approximator as shown in equation 2.14

\[
\nabla_{\theta} J(\theta) = E_{s \sim \rho^T}[\nabla_{\theta} \mu_{\theta}(s)\nabla_{a} Q^w(s, a)|_{a=\mu_{\theta}(s)}]
\]

(2.14)

We again consider the off-policy deterministic actor-critic, but this time we consider learning a deterministic target policy \( \mu_{\theta}(s) \) from trajectories to estimate the gradient using an arbitrary stochastic behaviour policy \( \pi(s, a) \).

\[
\nabla_{\theta} J_{\beta}(\mu_{\theta}) = E_{s \sim \rho^T}[\nabla_{\theta} \mu_{\theta}(s)\nabla_{a} Q^w(s, a)|_{a=\mu_{\theta}(s)}]
\]

(2.15)

### 2.6 Function Approximation

From equation 2.6 and equation 2.7, the action-value Q function is unknown. Hence, we consider approximating the Q function in policy gradient algorithm. Policy gradient methods use function approximation in which the policy is explicitly represented by its own function approximator and is updated with respect to the policy parameters. Using function approximation, it is possible to show that the gradient can be written in a suitable form for estimation by an approximate action-value function [Sutton et al., 2000].
In our work, we consider linear function approximators, such that both the stochastic and the deterministic policies can be approximated using an independent linear function approximator with its own parameters, that we call $w$. The main purpose of using function approximator in policy gradient methods is to approximate the Q function by a learned function approximator. To learn the function approximator parameters $w$ we use policy evaluation methods as discussed below.

### 2.6.1 Temporal Difference Learning

To estimate the Q function, we have used the policy evaluation algorithm known as temporal difference (TD) learning. Policy evaluation methods give a value function that assesses the return or quality of states for a given policy. Temporal Difference methods have widely dominated the line of research along policy evaluation algorithms [Bhatnagar et al., 2007], [Degris et al., 2012a], [Peters et al., 2005a]. TD learning can be considered as the case when the reward function $r^\pi$ and the transition probability $P^\pi$ are not known and we simulate the system with a policy $\pi$.

The off-policy TD control algorithm known as Q-learning [Watkins and Dayan, 1992] was one of the most important breakthroughs in reinforcement learning, in which the learnt action value function directly approximates the optimal action-value function $Q^*$, independent of the policy being followed. In particular we have implemented the least-squares temporal difference learning algorithm (LSTD) [Lagoudakis et al., 2002] that we discuss in the next section.

### 2.6.2 Least-Squares Temporal Difference Learning

The least squares temporal difference algorithm (LSTD) makes efficient use of data and converges faster than the conventional temporal difference learning methods [Bradtke and Barto, 1996]. [Lagoudakis et al., 2002], [Boyan, 1999] showed LSTD techniques for learning control problems by considering Least Squares Q Learning which is an extension of Q learning, that learns a state-action value function instead of the state value function. For large state and action spaces, for both continuous and discrete MDPs, we approximate the Q function with a parametric function approximator:

$$
\hat{Q}^\pi = \phi^T w
$$

where $w$ is the set of weights or parameters, $\phi$ is a $(|S||A| \times k)$ matrix where row i is the vector $\phi_i(s, a)^T$, and we find the set of weights $w$ that can yield a fixed point in the value function space such that the approximate Q action-value function becomes:

$$
\hat{Q}^\pi = \phi w^\pi
$$

18
Assuming that the columns of $\phi$ are independent, we require that $A\pi = b$ where $b = \phi^T R$ and $A$ is such that

$$A = \phi^T (\phi - \gamma P^\pi \phi)$$

(2.18)

where $w$ is a square matrix of size $k \times k$, and the feature map $\phi$ is a $(|S||A| \times k)$ matrix where row $i$ is the vector $\phi_i(s, a)^T$ and we find the set of weights that can yield a fixed point in the value function space. We can find the function approximator parameters $w$ from:

$$w^\pi = A^{-1} b$$

(2.19)

Using these desired set of weights for our function approximator, our approximated Q function therefore becomes

$$\hat{Q}^\pi = \sum_{i=1}^{k} \phi_i(s, a) w_i = \phi(s, a)^T w$$

(2.20)

### 2.7 Off-Policy Actor-Critic Algorithms

In actor-critic, for both stochastic and deterministic gradients, the actor adjusts the policy parameters $\theta$ by gradient ascent. Using the critic, it learns the $w$ parameters of the LSTD used as function approximator to estimate the action-value function $Q^w(s, a) \approx Q^\pi(s, a)$. Additionally, when using actor-critic, we must ensure that the function approximator is compatible for both stochastic and deterministic gradients. Otherwise, the estimated $Q^w(s, a)$ that we substitute instead of the true $Q^\pi(s, a)$ may introduce bias in our estimates.

For the stochastic policy gradient, the compatible function approximator is $Q^w(s, a) = \nabla_\theta \log \pi_\theta(a|s)^T w$. For the deterministic policy gradient, the compatible function approximator is given by $Q^w(s, a) = w^T a^T \nabla_a \mu_\theta(s)$. For both the stochastic and the deterministic policy gradients, we considered off-policy actor-critic instead of on-policy actor-critic algorithm. Using an off-policy setting means that, to estimate the policy gradient, we use a different behaviour policy to sample the trajectories. We used the off-policy actor critic algorithm (OffPAC) [Degris et al., 2012c] to pull the trajectories for each gradient estimate using a behaviour stochastic policy. Further using these trajectories, the critic then estimates the action-value function off-policy by using gradient temporal difference learning [Peters and Schaal, 2008].

### 2.8 Exploration in Deterministic Policy Gradient Algorithms

In our work, we analyze the effect of off-policy exploration to improve the convergence of deterministic policy gradient algorithms. Using a stochastic off-policy in deterministic gradients will therefore ensure sufficient exploration in the state and action space. The basic idea of using off-policy exploration is that the agent will choose its actions according
to a stochastic behaviour off-policy but will learn a deterministic policy in deterministic policy gradients. From our experimental results, we show that, by fine tuning the amount of exploration (\(\sigma\) parameters), we can make the agent learn to reach a better optimal policy, as it can explore more of the environment.

[Degris et al., 2012b] first considered off-policy actor-critic algorithms in reinforcement learning where they demonstrated that it is possible to learn about a target policy while obtaining trajectory states and actions from another behaviour policy; consequently, by having a stochastic behaviour policy, we can ensure a larger action space. The most well-known method of off-policy method in reinforcement learning is Q-learning [Watkins and Dayan, 1992].

The Off-PAC algorithm considered by [Degris et al., 2012b] demonstrates that the actor updates the policy weights while the critic learns an off-policy estimate of the action-value function for the current actor policy [Sutton et al., 2009]. In our work in deterministic gradients, therefore, we consider obtaining the episode or trajectory of states and actions using a stochastic off-policy, and learn a deterministic policy. Our work considers examining the effect of stochasticity in off-policy exploration to learn a better deterministic policy.

### 2.9 Optimization and Parameter Tuning In Policy Gradient Methods

Optimizing policy parameters in policy gradient methods is conventionally done using stochastic gradient ascent. Under unbiased gradient estimates and following certain conditions of learning rate, the learning process is guaranteed to converge to at least a local optima. [Baird and Moore, 1999] has largely considered the use of gradient ascent methods for general reinforcement learning framework.

Speed of convergence to local optima using gradient ascent is dependent on the learning rate or step size. After each learning trial, the gradient ascent learning rate is further reduced, and this helps in the convergence to a locally optimal solution. For our work, we consider a decaying learning parameter such that \(\alpha = \frac{a}{t+b}\) where \(a\) and \(b\) are the parameters that we run a grid search over to find optimal values, and \(t\) is the number of learning trials on each task. It has been shown that convergence results are dependent on decreasing learning rates or step sizes that also satisfy the conditions that \(\sum \gamma_t^2 < \infty\) and \(\sum \gamma_t = \infty\).

In this project, we take into account the problem of parameter tuning in gradient ascent for policy gradient methods. Parameter tuning is fundamental to presenting the results of RL algorithms on benchmark tasks. The choice of the learning rate affects the convergence speed of policy gradient methods. The general approach considered in reinforcement learning is to run a cluster of experiments over a collection of parameter values, and then report the best performing parameters on each of the benchmark MDP tasks.

The approach considered to find best parameter values for optimization becomes complex
for more difficult MDPs such as the Inverted Pendulum or Cart Pole. This is because, for example, on the Toy MDP, our work included running over a grid of 100 pairs of a and b values for the step size parameters, results averaged over 25 experiments for each pair of values. Running this cluster of experiments takes almost two days in practice. For performing a similar procedure to find best parameters for complex MDPs like the Inverted Pendulum or Cart Pole MDP will therefore be computationally very expensive.

2.10 Adaptive Step-Size for Policy Gradients

To resolve and eliminate the problem of parameter tuning in policy gradient methods, we consider the adaptive step size approach from [Matsubara et al., ] and we consider it in the framework of deterministic policy gradient algorithms. From our own literature search to find a solution to choosing learning rates in optimization techniques, we found this method (details of which is considered in next section) and applied it in the deterministic setting. We also did a literature search over [Pirotta et al., 2013] that considers a learning rate by maximizing a lower bound on the expected performance gain. However, experimental results using this approach was not successful, and so in this report, we will only consider results using the adaptive step size method from [Matsubara et al., ].

2.10.1 Adaptive Step Size With Average Reward Metric

The adaptive step size approach considered in [Matsubara et al., ] defines a metric for policy gradients that measures the effect of changes on average reward with respect to the policy parameters \( \theta \).

**Average Reward Metric Vanilla Gradient**

The average reward metric policy gradient method (APG) considers updating the policy parameters such that

\[
\theta = \theta + \alpha(\theta) R(\theta)^{-1} \nabla_{\theta} J(\theta)
\]

(2.21)

where \( R(\theta) \) is a positive definite metric that defines the properties of the metric, \( \epsilon \) is again a decaying parameter and \( \alpha(\theta) \) is given as:

\[
\alpha(\theta) = \frac{1}{\sqrt{2} \nabla_{\theta} \eta(\theta)^T R(\theta)^{-1} \nabla_{\theta} J(\theta)}
\]

(2.22)

Following this average reward metric vanilla policy gradient scheme, we implement our adaptive step size vanilla policy gradient algorithms, such that:

\[
\theta_{k+1} = \theta_k + \epsilon J \alpha^*(\theta) \nabla_{\theta} J(\theta)
\]

(2.23)
where in case of vanilla policy gradients, the step size will therefore be given as:

$$\alpha^*(\theta) = \frac{1}{\nabla_\theta J(\theta)^T \nabla_\theta J(\theta)}$$  \hspace{1cm} (2.24)

**Average Reward Metric With Natural Gradient**

From [Matsubara et al.,], we also propose our adaptive natural deterministic policy gradient for the first time, that is a variation of natural deterministic gradients considered in [Silver et al., 2014].

The update of the policy parameters $\theta$ and the adaptive step size $\alpha(\theta)$ in case of natural deterministic gradients is given by:

$$\theta_{k+1} = \theta_k + \epsilon \alpha^*(\theta) F(\theta)^{-1} \nabla_\theta J(\theta)$$ \hspace{1cm} (2.25)

$$\alpha^*(\theta) = \frac{1}{\nabla_\theta J(\theta)^T F(\theta)^{-1} \nabla_\theta J(\theta)}$$ \hspace{1cm} (2.26)

In our experimental results in chapter 4, we will demonstrate our results using both the adaptive vanilla (APG) and adaptive natural (ANPG) stochastic and deterministic policy gradient algorithms, and analyze improvements in convergence using these methods.

### 2.11 Improving Convergence of Deterministic Policy Gradients

In our work, we then considered different techniques to improve the convergence of deterministic policy gradient algorithms. In the sections below, at first we consider natural gradients that are well-known methods to speed up convergence of policy gradient methods. Previously, stochastic natural gradients have been implemented in an experimental framework. [Silver et al., 2014] recently proved the existence of deterministic natural gradients, but has not shown any experimental solutions to this approach. We consider the experimental framework of deterministic natural gradients for the first time, and use this algorithm on our benchmark RL tasks. Section 2.12 below discusses the background for natural stochastic and deterministic gradients.

In section 2.13 below, we also consider a novel approach to speed up convergence of deterministic policy gradients for the first time. We used momentum-based gradient ascent approach, inspired from use in deep neural networks. We are the first to consider applying momentum-based optimization techniques in policy gradient approach in reinforcement learning. Our goal for using these techniques is to analyze whether such gradient ascent approaches can have a significant effect on the convergence rate of deterministic policy gradients. Our work is inspired by recent work from [Sutskever et al., 2013] who considered
training both deep and recurrent neural networks (DNNs and RNNs) by following stochastic gradient descent in neural networks with momentum-based approach.

2.12 Natural Gradients

2.12.1 Stochastic Natural Gradient

The approach is to replace the gradient with the natural gradient [Amari, 1998], which leads to the natural policy gradient [Kakade, 2001] [Peters et al., 2005b]. The natural policy gradient provides the intuition that with any change in the policy parameterisation, there should not be any effect on the result of the policy update.

[Peters et al., 2005b] considers the actor-critic architecture of the natural policy gradient where the actor updates are based on stochastic policy gradients and the critic obtains the natural policy gradient. [Kakade, 2001] has demonstrated that vanilla policy gradients often gets stuck in plateaus, while the natural gradients do not. This is particularly because the natural gradient methods do not follow the steepest direction in the parameter space but rather the steepest direction with respect to the Fisher metric which is given by:

$$\nabla_\theta J(\theta)^{\text{nat}} = G^{-1}(\theta)\nabla_\theta J(\theta)$$ (2.27)

where $G(\theta)$ denotes the Fisher information matrix. It can be derived that an estimate of the natural policy gradient is given by:

$$\nabla_\theta J(\theta) = F_\theta w$$ (2.28)

Furthermore, by combining equation 2.27 and 2.28, it can be derived that the natural gradient can be computed as:

$$\nabla_\theta J(\theta)^{\text{nat}} = G^{-1}(\theta)F_\theta w = w$$ (2.29)

since $F_\theta = G(\theta)$ as shown in [Peters et al., 2005b]. This proves that for natural gradients, we only need an estimate of the $w$ parameter and not $G(\theta)$, and as previously discussed, we estimate the $w$ parameters using LSTD. Therefore, the policy improvement step in natural gradients is

$$\theta_{k+1} = \theta_k + \alpha w$$ (2.30)

where $\alpha$ again denotes the learning step size. Additionally, [Peters et al., 2005b] also discusses how similar to the vanilla gradients, convergence of natural gradients is also guaranteed in the local optimum. Additionally, their work discusses that, by considering the parameter space, and by choosing a more direct path to the global optimal solution, the natural gradients have a faster convergence compared to the vanilla gradients.
2.12.2 Deterministic Natural Gradient

Further to that, [Silver et al., 2014] stated the deterministic version of the natural gradient, and proved that the natural policy gradient can be extended to deterministic policies. [Silver et al., 2014] states that, even for deterministic policies, the natural gradient is the steepest ascent direction. For deterministic policies, they used the metric

\[ M_\mu(\theta) = \mathbb{E}_{s \sim \rho^\theta}[\nabla_\theta \mu_\theta(s) \nabla_\theta \mu_\theta(s)^T] \]  

(2.31)

and by combining deterministic policy gradient theorem with compatible function approximation, they proved that

\[ \nabla_\theta J(\mu_\theta) = \mathbb{E}_{s \sim \rho^\theta}[\nabla_\theta \mu_\theta(s) \nabla_\theta \mu_\theta(s)^T w] \]  

(2.32)

and so the steepest direction in case of natural deterministic gradients is again simply given by

\[ M_\mu(\theta)^{-1} \nabla_\theta J_\beta(\mu_\theta) = w \]  

(2.33)

For the purposes of our work, we implement the natural deterministic policy gradients, and include it in our comparison of policy gradient methods.

2.13 Momentum-based Gradient Ascent

2.13.1 Classical Momentum

Classical momentum (CM), as discussed in [Sutskever et al., 2013] for optimizing neural networks, is a technique for accelerating gradient ascent such that it can accumulative a velocity vector in directions of persistent improvement in the objective function. Considering our objective function \( J(\theta) \) that needs to be maximized, the classical momentum approach is given by:

\[ v_{t+1} = \mu v_t + \epsilon \nabla_\theta J(\theta_t) \]  

(2.34)

\[ \theta_{t+1} = \theta_t + v_{t+1} \]  

(2.35)

where again, \( \theta \) is our policy parameters, \( \epsilon > 0 \) is the learning rate, and we use a decaying learning rate of \( \epsilon = \frac{\alpha}{\beta+\theta} \) and \( \mu \) is the momentum parameter \( \mu \in [0, 1] \).

2.13.2 Nesterov Accelerated Gradient

We also consider Nesterov Accelerated Gradient (NAG), as discussed in [Sutskever et al., 2013] for optimizing neural networks, in our policy gradient algorithms, that have been of recent interest in the convex optimization community [Cotter et al., 2011]. [Sutskever et al., 2013] discusses how NAG is a first order optimization method that has a better convergence rate
guarantee than certain gradient ascent situations. NAG method considers the following optimization technique:

\[ v_{t+1} = \mu v_t + \epsilon \nabla_{\theta} J(\theta_t + \mu v_t) \quad (2.36) \]

\[ \theta_{t+1} = \theta_t + v_{t+1} \quad (2.37) \]

**Relationship between Classical Momentum and Nesterov’s Accelerated Gradient**

Both the CM and NAG method computes a velocity vector by applying a gradient-based correction to the previous velocity vector; the policy parameters \( \theta_t \) are then updated with the new velocity vector. The difference between CM and NAG method is that CM computes the gradient update from the current position \( \theta_t \) while NAG computes a partial update to \( \theta_t \) and computes \( \theta_t + \mu v_t \) which is similar to \( \theta_{t+1} \).
Chapter 3

Learning System Structural Framework

In this section, we give a short description of the fundamental building blocks that have been made for our experimental framework. All our building blocks are implemented such that it can be integrated within an existing code base to run policy gradient experiments. The algorithms considered in our work are implemented from scratch by ourselves using MATLAB object-oriented programming, and is made modular such that additional techniques can be implemented in this existing framework.

This section introduces the experimental details and the software approach that are considered to obtain results for comparison and analysis of convergence of policy gradient methods. Code for our implementation of stochastic and deterministic policy gradient algorithms on the benchmark RL tasks are available online.

3.1 Parameterized Agent Class

We used a parametric agent controller in which we consider a stochastic Gaussian policy

$$\pi_{h, \Sigma}(a|s) = \frac{1}{Z} \exp\left(-\frac{1}{2} (h(s)-a)^T \Sigma^{-1} (h(s)-a)\right)$$  \hspace{1cm} (3.1)

where the function $h$ is given by a linear function of features $\phi_i(.)$ where

$$h(s) = \sum_{i=1}^{n} \theta_i \phi_i(s)$$  \hspace{1cm} (3.2)

$$\phi_i(s) = K(s, c_i)$$  \hspace{1cm} (3.3)

where $c$ is the centres and we optimize the parameters $\theta$.

For our experimental framework, we have made an Agent class (that can also be ex-
tended for different controllers) in which we initialize the policy parameters, the policy function that implements a Gaussian policy by interacting with our kernel functions class, and also have defined the gradient of our log policy required for our derivation of policy gradients. For our parametric controller, we also define methods for updating policy parameters and functions that pick up centres to initialize our policy. The agent controller implements the feature maps that are used for compatible function approximation, in case of stochastic and deterministic gradients. In this class, we have also defined the off-policy stochastic policy for exploration in deterministic gradients. The agent class interacts with the MDP class; we implemented a function that takes the functional product of the components used to compute the gradient and also implemented the deterministic and stochastic gradient functions that gives an estimate of the gradient vector or the direction of steepest ascent.

3.2 MDP Class

Our work focused on implementing the Toy, Grid World and Mountain Car MDPs, while the other two (Cart Pole and Pendulum) MDPs were adopted from an existing MDP library.

We have implemented MDP classes on which the agent can interact. The environment contains functions that can describe the state transitions when an action is executed. These transition functions are used for the internal state transitions. The transition functions take the current state and action and evaluate the next state for our MDP class. Our MDP classes also contain the reward functions, the initialized start states and actions. Below we give a short description for the MDP classes that were implemented for building our learning system framework. As a first step, a Pendulum MDP was adopted, from which we extended into other MDP classes such as a Toy MDP with multiple states and action, and a GridWorld MDP. We also implemented a simpler version of the Mountain Car MDP, but do not include details here since our experiments were not carried out with the Mountain Car MDP.

3.2.1 Toy MDP

We considered the Toy benchmark MDP, which is a Markov chain on the interval \( S \in (-1, 1) \) where we consider continuous actions \( A \in (-1, 1) \) and the reward function defined as \( r(s, a) = \exp(-|s-3|) \). The dynamics of the MDP are defined as \( s' = s + a + \epsilon \) where \( \epsilon \) is a Gaussian noise added to the successor state. In our Toy MDP, we also included multiple local reward blobs such that \( r(s, a) = 0.4 \exp(-|s-1|) \) for local blob 1, \( r(s, a) = 0.4 \exp(-|s-2|) \) for local blob 2 and \( r(s, a) = 0.4 \exp(-|s+3|) \) for local blob 3. We introduced these Gaussian local reward blobs such that we are guaranteed to have multiple local optima in the policy space of the Toy MDP. Our setting considers the case where the agent cannot take an action more than \([-1, 1]\) and cannot exceed the state space, ie, the agent is reset to \( s = 0 \) if it tries to leave the Markov chain. We considered the Toy MDP over a horizon
of $H = 20$ and the optimal cumulative reward found (by simply acting optimally on the MDP since it is a small setting) is around 18. For our controllers, we used 20 centres and obtained our results averaged over a large set (typically 25) of experiments.

### 3.2.2 Grid World MDP

Another benchmark MDP that we considered for our work is the Grid World MDP. The Grid World is in a continuous state space of a $4 \times 4$ grid of states, such that $S \in (-4, 4)$ and we consider continuous actions of still, left, right, up and down, ie $A = [0, 1, 2, 3, 4]$. For instance, any action value between 3 and 4 is considered as action up and so on. Similar to the Toy MDP, we also considered having multiple local reward blobs in our Grid World MDP. In the Grid World example, the goal state is at position $(3, 3)$ and our reward is defined as $r(s, a) = \exp(-\text{dist}/2)$ where "dist" is the distance between the current state of the agent and the goal state. The other local blob is at position $(1, 1)$ such that $r(s, a) = 0.5\exp(-\text{localdist}/2)$ where again, "localdist" is the distance between the current state of the agent, and the local reward blob.

### 3.2.3 Cart Pole MDP

The Cart Pole MDP is concerned with balancing a pole on the top of the cart as considered in [Sutton and Barto, 1998]. We consider the more difficult Cart Pole MDP for analyzing convergence of our implemented algorithms. This is an MDP code that we used from the research group, and we do not implement the Cart Pole MDP by ourselves. For our experimental purposes, we show the performance of our algorithms on this cart pole task. Since implementation of the Cart Pole MDP is not our own work, we do not include the details of this MDP in this report.

### 3.3 Other Implementations for Developing Software

We implemented a function that can approximate the Q function. This function interacts with another function that implements the least squares temporal difference learning (LSTD) algorithm. Using the learned weights $w$ for function approximation, the Q function is then implemented with the feature maps $\phi(s, a)$ (which is the score function and is different for stochastic and deterministic gradients to ensure function approximation) and the weights $w$.

We also implemented a "Rho Integrator" class which can compute the sum or integral over the trajectory of states and actions used in computing the gradient. This function samples the states and actions for policy gradients, and computes the "Expectation" over the score function and the approximated Q function over the sampled states as shown in the deterministic gradient theorem.
We use the same software framework (since it is modular) for implementing variations of deterministic gradient algorithms that we consider in our work (such as the adaptive step size, natural and momentum-based policy gradients).

## 3.4 Experimental Details

For the Toy MDP, for each of the stochastic and deterministic policy gradient algorithms, including variations in the optimization technique, the experiments are averaged over 25 different experiments, each containing 200 learning trials. Similarly, for the Grid World MDP, we used 500 learning trials or episodes of experience, and the results are averaged over 5 experiments. For the Cart Pole MDP, our results are only averaged over 3 experiments for 1800 learning trials, due to the computation time involved in running our algorithms on complex MDPs.

For fine tuning learning rate parameters, we ran a grid of experiments over 10 values of $a$ and $b$ each (grid of 100 parameters) to find optimal learning rates for the Toy MDP. When considering the momentum-based gradient ascent approach, we also had to fine tune the momentum and learning rate parameters using a similar approach.
Chapter 4

Experimental Results and Analysis

In this section, we present our experimental results based on our own implementation of stochastic and deterministic policy gradient algorithms. At first, we illustrate results of the performance of stochastic and deterministic policy gradients. We then present our analysis towards improving convergence by fine tuning off-policy exploration. Our results then illustrates the need for grid search over learning rate parameters in gradient ascent, and the approach we considered using adaptive step size. Finally, we include results of our algorithms using natural and momentum-based policy gradients to improve convergence of deterministic policy gradient algorithms on benchmark reinforcement learning tasks.

4.1 Policy Gradients

4.1.1 Stochastic and Deterministic Policy Gradient

Our first experiment focuses on stochastic policy gradient algorithms on the Toy MDP. We show that over the number of learning trials, the parameterized agent learns a near optimal policy, ie, it learns to reach the goal state. The results show that, for the stochastic policy gradient, we can reach a cumulative reward close to the optimal reward.

In our results here and further in this section, we often include a horizontal line on the graph which denotes the optimal cumulative reward or the local cumulative reward that we get by acting out in the environment. These lines show the cumulative rewards if we act out and get stuck in the local reward blobs.

Figure 4-1 shows the averaged results of our stochastic policy gradient (SPG) and figure 4-2 shows results for the deterministic policy gradient (DPG) algorithm on the simple Toy MDP considered in section 3.2.1. The x-axis showing the number of learning trials (or episodes) of experience, and the y-axis showing the cumulative reward. Both results show that our algorithm is working effectively to maximize the cumulative reward on the simple Toy problem.
Grid World MDP
Figure 4-3 shows the performance of our stochastic and deterministic policy gradient algorithm on the simple Grid World MDP considered in section 3.2.2. We here present our results on Grid World averaged over 5 experiments, and using 500 learning trials, and compare the two vanilla (steepest ascent) policy gradient algorithms together. Our Grid World MDP contains multiple local reward blobs (i.e., with goal distractors or local gaussian reward blobs) in the policy space. Results in figure 4-3 show that the stochastic gradient performs better than the deterministic gradient.

4.1.2 Local Optima Convergence
Toyd MDP With Multiple Local Reward Blobs

We then consider including multiple local reward blobs (goal distractors) on the Toy MDP. The inclusion of local reward blobs (Gaussian reward blobs that are less than the goal blob) means that we are guaranteed to have local optima in the Toy MDP setting. The results in this section show that our stochastic and deterministic policy gradient algorithm sometimes gets stuck in a local optima, i.e., the agent reaches the local reward blobs or distractors and stays there, instead of reaching the goal state. Figure 4-4 here and Figure B-1 in Appendix B compares local optima convergence of the stochastic and the deterministic policy gradient algorithms. Our results suggest that the deterministic policy gradient is more likely to get stuck in a local reward blob. All results here are for one run of DPG only, and not averaged. In later section, we consider our analysis to always avoid the worst local optima, and fine tune exploration to converge to the best possible locally optimal policy.

4.2 Exploratory Stochastic Off-Policy in Deterministic Gradients

Considering local reward blobs in the Toy MDP, the deterministic policy gradient having no exploration showed that the agent gets stuck in the worst possible local optima. Our analysis considers the effect of off-policy stochastic exploration in the deterministic setting to make these algorithms converge to a better optimal policy in practice.
To resolve the issue of getting stuck in worst possible local optima in the MDP, we ran a grid of experiments over sigma (\(\sigma\)) values ranging form 0 to 0.5, and took average over 25 experiments for each sigma value. Our results show that, with variations in the ”noise” or ”stochasticity” parameter \(\sigma\) in the stochastic behaviour policy that is used to generate the trajectories at each episode, the convergence of the deterministic gradient changes. Here, we show our results for the Toy MDP having multiple reward blobs, and the effect of exploration in deterministic gradients.

### 4.2.1 Effect of Exploration

Results in figure 4-5 suggest that having no exploratory stochastic policy, but simply using the deterministic policy to generate trajectories (no exploration, ie \(\sigma = 0\)), the deterministic gradient performs the worst. Results also show that with variations in the sigma parameter, the performance of the deterministic gradient improves. At an optimal value of sigma (\(\sigma = 0.4\)), the deterministic gradient performs the best, converging more towards a globally optimal policy. Hence, result here illustrates that, by finding an optimal value for exploration, we can make the agent avoid getting stuck in the worst local blob, and therefore can converge to a better locally optimal policy.

For later experiments, we use this \(\sigma = 0.4\) parameter forward in case of vanilla or steepest ascent policy gradients.
4.3 Resolving Issue of Fine Tuning Parameters in Policy Gradient Algorithms

4.3.1 Choosing Optimal Step Size parameters

Our analysis in section 2.9 suggested that the convergence rate of the policy gradient al-
algorithms is dependent on the learning rate of the stochastic gradient ascent. Our step size is dependent on two parameters such that step size $\alpha = a/(p + b)$ where $a$, $b$ are the two parameters that we need to find the optimal values for, and $p$ denotes the number of iterations or learning trials.

At first we ran our experiments over a grid of step size parameters, to choose the optimal parameters that would ensure a good convergence rate. However, choosing the optimal step size even for the Toy MDP required running over clusters of experiments for a grid of 100 pairs of $a$ and $b$ values, each time having 200 iterations and averaging over 25 experiments. Our experience finding the optimal learning rate parameters showed that this process will be very time consuming for other complex MDPs.

Experimental results in figure 4-6 is considered for the deterministic gradient with changing step size parameters $b$. As expected, our results show that having a very small step size for $b = 1000$ leads to a very slow convergence to a local optima. In contrast, for $b = 100$ or $b = 50$, ie, for a larger step size, the deterministic policy gradient has a faster convergence rate and outperforms the results for other step size parameters.

4.3.2 Adaptive Policy Gradient Methods With Average Reward Metric

However, as discussed previously, the conventional approach of choosing the learning rate parameters over a grid search of values would become more complex for larger MDPs such as the Cart Pole. To this date, there are no approaches to automatically choosing the learning rates in stochastic gradient ascent based optimization techniques in the framework of policy gradient methods.

Therefore, we present results for the adaptive step size approach that we proposed in deterministic policy gradients. We made an attempt to partially eliminate the need for fine tuning of learning rate parameters. Here, we only present the results for the adaptive deterministic policy gradients. In later sections, when we include other methods to speed up convergence, we show the effect of using adaptive step size in those methods, and include a comparison of all methods, to show the best performing algorithm.

Adaptive Deterministic Policy Gradients

Using the adaptive step size, we also need to find the optimal $\epsilon$ parameter such that $\epsilon = c/\sqrt{\text{iterations}}$. However, fine tuning this $c$ parameter is not a problem (even for larger MDPs), since we only need to ensure that the decaying $\epsilon$ parameter does not become too small. Experimental results in figure 4-7 shows the effect of changing the $\epsilon$ parameter in the adaptive vanilla deterministic policy gradient setting. Results show that the $c$ parameter can be chosen to be high arbitrarily. In later sections, we further demonstrate how our
adaptive step size approach can eliminate the need for doing a grid search to find optimal learning rate parameters, which is time expensive for complex and larger MDPs.

**4.4 Natural Policy Gradients To Speed Up Convergence**

From this section onwards, we present some of the approaches we used to speed up convergence of deterministic policy gradients. We consider natural stochastic and deterministic gradients as discussed in section 2.12, to further analyse whether the natural gradients can speed up convergence of our algorithms. At first, we show the performance of our implemented natural stochastic (NSPG) and natural deterministic (NDPG) policy gradient algorithms on the Toy MDP.

**4.4.1 Natural Stochastic Policy Gradient**

Figure 4-8 shows results for the natural stochastic gradient, for two different experiments, on the Toy MDP averaged over 25 experiments.

**4.4.2 Natural Deterministic Policy Gradient**

We then present our results with natural deterministic policy gradient (NDPG) that are implemented and presented for the first time based on the theoretical proof that [Silver et al., 2014] showed. Our results, again for two different experiments, are the first to show performance
guarantees of the natural deterministic policy gradient on the simple Toy MDP (having multiple reward blobs) that we consider.

4.4.3 Comparing Convergence of Natural and Vanilla Policy Gradients

Our results from figure 4-10 shows that our vanilla (steepest ascent policy gradients) and natural gradients perform similarly when compared. However, from the plot, the deterministic natural gradients seem to perform worse than the vanilla gradients, while the stochastic gradients perform the best. For both vanilla and natural gradients, the stochastic algorithm outperforms the deterministic algorithm. Results in this section, although shows that the stochastic gradients outperforms, but this is also because the performance of the deterministic gradients are dependent on the parameter $\sigma$ which determines the stochasticity in exploration as we discussed in the previous section.
4.4.4 Natural Deterministic Gradient With Exploration

Figure 4-10: Comparing Natural and Vanilla Stochastic and Deterministic Policy Gradients

Figure 4-11: Effect of Exploration on Natural Deterministic Policy Gradient - Toy MDP With Multiple Local Reward Blobs
In the section above, we showed that our implemented natural gradient works for both the stochastic and deterministic setting. In order to further improve convergence of natural deterministic gradient, we considered the effect of the exploration parameter on the natural deterministic policy gradient, to see whether the exploration parameter is consistent for all our algorithms on the Toy MDP. Our results again illustrates that with variations in the exploration parameter $\sigma$, the convergence rate of the natural DPG also varies significantly.

In both settings, having no exploration in the off-policy stochastic policy means that the deterministic policy gradient performs poorly, and significantly gets stuck to the worst of the reward blobs present in the policy space. Our results are interesting because both figure 4-5 and figure 4-11 shows an almost similar exploration parameter $\sigma = 0.3$, which demonstrates our hypothesis that the effect of exploration must be the same on all the algorithms we consider for the Toy MDP. Our results therefore show that an optimal exploration parameter can improve convergence rate in case of both vanilla and natural policy gradients, and this exploration parameter is further used for other algorithms that we consider in our work.

4.5 Momentum-based Policy Gradients To Speed Up Convergence

We then considered momentum-based gradient ascent approach to speed up convergence of deterministic policy gradients. Momentum-based gradient ascent approaches are known to speed up convergence involving stochastic gradient ascent based optimization techniques. We include results with changing values of momentum $\mu$ and learning rate $\epsilon$ parameters, to find the optimal parameters with which our algorithms perform the best in practice.

4.5.1 Classical Momentum-based Policy Gradient

At first, we ran experiments over variations of momentum and learning rate parameters to analyze the performance of stochastic policy gradients based on the classical momentum approach. Our results show that, with variations of $\mu$, the resulting convergence speed of the SPG varies significantly. Figure 4-12 considers the effect of changing the $\mu$ and $\epsilon$ parameters on the convergence of the stochastic policy gradients to learn a locally optimal policy.

We then consider the effect of classical momentum on the deterministic policy gradients. We show our results here for the Toy MDP, for different pairs of momentum ($\mu$) and learning rate ($\epsilon$) parameters. In figure 4-13 our results show that at a certain value of $\mu$ and $\epsilon$ for the momentum parameters, the deterministic gradient with classical momentum can achieve a significantly better convergence rate. We use the best performing $\mu$ and $\epsilon$ for later experiments with classical momentum-based approach.
Figure 4-12: Stochastic Policy Gradient With Classical Momentum based gradient ascent - Effect of $\mu$ and $\epsilon$ parameters

Figure 4-13: Deterministic Policy Gradient With Classical Momentum Gradient Ascent - Effect of $\mu$ and $\epsilon$ parameters

4.5.2 Nesterov’s Accelerated Gradient-based Policy Gradients

As shown in figure 4-14 our results again illustrate that by choosing optimal $\mu$ and $\epsilon$ momentum parameters, we can obtain a fast convergence rate of the deterministic gradients with Nesterov’s momentum-based gradient ascent.
4.5.3 Comparing Vanilla, Natural and Momentum-based Policy Gradients

In this section, we compare the convergence of the vanilla and natural policy gradients with the momentum-based approaches, to further analyze whether the momentum-based gradient ascent technique can improve the convergence rate of our algorithms.

Comparing With Classical Momentum

Our results in figure 4-15 shows that even though the vanilla and natural SPG tends to avoid the local optima and converges to a near global optima, but it is the classical momentum approach that has a faster convergence rate compared to the other methods we consider. Even though the classical momentum approach gets stuck in worse locally optimal policy, but it has a faster convergence rate compared to the other methods.

Comparing With Nesterov’s Accelerated Gradient

We then compare the performance of the vanilla, natural and Nesterov’s accelerated gradient methods for convergence to a locally optimal policy. Our results are interesting since it shows that, similar to previous result, even though the natural SPG outperforms and avoids the local optima in the MDP, the Nesterov’s based gradient has a fast convergence rate compared to the other methods, even though it converges to a worse local optimal policy.
Comparison of SPG and DPG - Vanilla, Natural and Classical Momentum on Toy MDP with Multiple Reward Blobs

Figure 4-15: Comparing Vanilla, Natural and Classical Momentum-based Policy Gradients

Comparison of Vanilla, Natural and Nesterov’s Accelerated Gradient-based policy gradients

Figure 4-16: Comparing Vanilla, Natural and Nesterov’s Accelerated Gradient-based policy gradients
4.6 Improving Convergence of Deterministic Policy Gradients

In this section, we combine our approaches together to analyze the effect on improving convergence of deterministic policy gradient algorithms. At first we consider comparing the vanilla deterministic gradients with the approaches (natural and momentum-based) we considered to speed up convergence.

4.6.1 Vanilla, Natural and Classical Momentum

![Comparison of DPG – Vanilla, Natural and Classical Momentum Gradient on Toy MDP With Multiple Reward Blobs](image)

Figure 4-17: Comparing Deterministic Vanilla, Natural and Classical Momentum policy gradients

We consider four different settings (with changing the $\mu$ and $\epsilon$ step size parameters) of the classical momentum deterministic gradient, and compare their performance with two runs of the natural deterministic gradient (each averaged over 25 experiments) and the vanilla deterministic gradient. Comparison of these algorithms are shown in figure 4-17.

Results here show that the classical momentum-based approach converges to a worse local optimal policy compared to the other methods considered. Our results once again proved that if we can find an optimal exploration $\sigma$ parameter for our MDP, then we can ensure a fast convergence to a good local optimal policy of the vanilla deterministic policy gradient algorithm.
4.6.2 Vanilla, Natural and Nesterov’s Accelerated Gradient

We further considered comparing the vanilla, natural and Nesterov’s accelerated based gradient on the deterministic policy gradient setting. We include two average graphs for the natural gradient, two runs for the Nesterov’s based gradient with changing $\mu$ and $\epsilon$ values, and compare them with the vanilla gradient with optimal exploratory $\sigma$ parameter.

Our results in figure 4-18 again shows that the deterministic vanilla gradient with optimal exploration outperforms the other methods having a high convergence rate and converging to a better local optima. The Nesterov’s gradient based method, even though has a high convergence rate, but it gets stuck to a worse local optima compared to the vanilla gradient.

However, one drawback is that, for all these algorithms, careful fine tuning of parameter is required, and hence our results are biased and dependent on fine tuning of learning rate and momentum parameters.

4.6.3 Adaptive Step Size Policy Gradients

To eliminate the need for fine tuning of parameters in policy gradient algorithms, in this section, we take a deeper look into our proposed adaptive policy gradients, the results of which we briefly introduced in section 4.3.2. This method that can partially eliminate the need for fine tuning, is then combined and compared with the other algorithms to find the best performing algorithm on the benchmark tasks.
At first we present our results of comparing all the adaptive deterministic policy gradient methods together as shown in figure 4-19. Our results comparing the DPG methods shows that our adaptive natural DPG outperforms all other methods by converging to a better local optimal policy, having a similar fast convergence rate compared to the vanilla DPG with optimal exploration parameter.

The results show that, using our approach of adaptive policy gradients to eliminate fine tuning, combined with our method of natural deterministic gradients to speed up convergence, the adaptive natural deterministic gradient (ANDPG) algorithm outperforms other adaptive deterministic gradient algorithms.

### 4.6.4 Improving Convergence

In this section, we therefore compare our adaptive policy gradients, with the conventional approach that requires fine tuning of parameters.

We include our experimental results for comparing all the step size methods together for comparison in figure 4-20. Our results show that the adaptive step size methods performs as good as the grid search step size method, and this illustrates that we can use the adaptive step size approach for more complex MDPs later in our work. For complex MDPs, we can therefore avoid performing a grid search over parameters over clusters of experiments, which would have involved running experiments for long days.
Figure 4-20: Effect of Adaptive Step Size on Convergence of Stochastic and Deterministic Policy Gradients

Figure 4-21: Comparing Deterministic Policy Gradients - Adaptive Natural DPG Outperforming Other Algorithms
Our results illustrate that the adaptive deterministic gradients (with fine tuned exploration parameter) can perform equally good as the adaptive stochastic gradients. Furthermore, results from figure 4-21 shows that the adaptive natural DPG can perform equally good as the vanilla and natural DPG (that are fine tuned for exploration and learning rate parameters). This result is significant to us as it clearly shows that our adaptive natural DPG is an ideal algorithm that can eliminate fine tuning, and also improve convergence to a good locally optimal policy in deterministic policy gradient algorithms.

4.6.5 Convergence of Stochastic and Deterministic Policy Gradients

In this section, we consider combining all our algorithms together that are considered in this project.

Figure 4-22 illustrates our results comparing convergence of all the algorithms. For convenience, we list down the methods that are considered in the results in this section:

- Vanilla stochastic and deterministic gradients (with optimal and no exploration)
- Natural stochastic and deterministic gradients (with optimal and no exploration)
- Adaptive vanilla stochastic and deterministic policy gradient
• Adaptive natural stochastic and deterministic policy gradient
• Classical momentum-based stochastic and deterministic gradient
• Nesterov’s accelerated gradient-based stochastic and deterministic policy gradient

Our results illustrate the performance of all our algorithms, in terms of their convergence rates and whether they can converge to a better locally optimal policy, that is close to the global optimum. Results show that the adaptive deterministic policy gradients (ADPG), can perform equally well compared to the stochastic gradient algorithms on the benchmark MDP. Furthermore, as stated previously, the ANDPG outperforms and converges to a better local optimal policy compared to the other deterministic counterparts.

4.7 Extending To Other Benchmark Tasks

Our results in section 4.6.5 illustrates that all our adaptive step size algorithms performs quite well when compared with other algorithms. In this section, we first show a similar result for our Grid World MDP, using the deterministic policy gradient algorithms to see the effect of using adaptive step size methods on more complex MDPs. Taking the best performing DPG algorithm, we will then illustrate our results for a more complicated Cart Pole MDP.

4.7.1 Grid World MDP

![Figure 4-23: Comparing Algorithms for Grid World MDP](image)

Results from figure 4-23 illustrates that the adaptive step size vanilla and natural deterministic gradients outperforms the other algorithms (Nesterov’s, Vanilla and Natural). The
results on the Grid World MDP shows that similar to the Toy MDP, even for more complex MDPs, the adaptive step size DPG method can converge to a better policy, while also eliminating the need for fine tuning of learning rate parameters.

4.7.2 Cart Pole MDP

![Figure 4-24: Comparison of Algorithms on the Cart Pole MDP](image)

Finally, in this section we show the performance of our natural deterministic policy gradient algorithm on the complex Cart Pole MDP task. Our results here are averaged over 5 experiments, and figure 4-24 shows that using this algorithm, the Cart Pole converges to a good local optimal policy that is close to the optimal cumulative reward of 52. Here, we do not consider fine tuning learning parameters for the vanilla and natural DPG to show the significance of it on complex MDPs. Results from figure 4-24 shows that the adaptive natural DPG can converge to a very good locally optimal policy, without the need to fine tune this algorithm, which is a significant improvement in the practical aspects of policy gradient algorithms.

4.8 Modified and Adaptive Initial Step Size

We have shown our experimental results for different step size methods on different approaches considered in stochastic and deterministic policy gradients. Our experimental results from previous section have shown that the adaptive step size method performs well on our benchmark MDPs. However, after careful analysis of our work, we realized that one major problem with the adaptive step size method is that, when the algorithms converge to an the maximum point in the local optima, the gradient at that point is zero. This
makes the step size at the local optima become infinitely large. Hence, our algorithm works such that, it never converges to an absolute local optima, but previously, all results are for reaching close to a local optima. This illustrates that, even though the adaptive step size can eliminate problem of fine tuning parameters, but it is not a sound method that can be applied across all domains of policy gradient research.

In this section, we consider a different step size method, that we call the ”Modified Step Size”, such that we can ensure convergence to the maximum point in the local optima. In particular, we use a step size such as:

\[ \alpha = \frac{A}{1 + 10t/\text{iterations}} \]

Here, we only need to fine tune the initial step size parameter A, while keeping the decay rate \( \frac{10t}{\text{iterations}} \) fixed. We use a small step size that even though has a slow convergence rate, but can guarantee convergence to a better locally optimal policy compared to the other algorithms considered previously. For our experiments, we consider a larger number of learning trials now, with a slower convergence rate, in order to ensure that our algorithms can converge to a good local optimal policy.

![Figure 4-25: Vanilla and Natural DPG - Using Modified Step Size on Grid World MDP](image)

When considering the Grid World MDP, we find that using this modified step size method, with a slow convergence rate, we can ensure good local optima convergence. Experimental results in figure 4-25 show that using this method, we can ensure much better local convergence for the Grid World MDP. Results for the vanilla deterministic gradient have significantly improved compared to previously shown result for Grid World MDP in fig-
This is because, using the modified step size, our aim is to converge to a very good local optimal policy, even though we have a slow convergence rate.

Our research analysis then made us realize that we can consider the initial step size $A$ as an adaptive step size, while keeping the decay rate fixed. We call this method the "Adaptive Initial Step Size", since our initial step in the algorithm is considered using an adaptive approach. The learning rate $\alpha$ is such that:

$$\alpha = \frac{A}{1 + (10t)/\text{iterations}}$$

where $A$ is the initial step size. Rather than having to fine tune and grid search the $A$ parameter, we can use the initial $A$ step size as the adaptive step size such that:

$$A = \frac{1}{\nabla_\theta J(\theta)^T \nabla_\theta J(\theta)}$$

Using the initial adaptive step size, and a constant decay rate, we can therefore completely eliminate the problem of fine tuning in policy gradient methods. This "Adaptive Initial Step Size" method has been recently applied on policy gradients in the RKHS [Lever et al., 2015], that are not part of our work in this project, and has shown significant improvements in their experimental results, eliminating the need to run cluster of experiments to fine tune learning rate parameters. This is also part of our future work as discussed in section 5.5.
Chapter 5

Discussion

In this section, we include a discussion of our results, and our work towards analyzing and improving convergence rates of deterministic policy gradient algorithms. In each of the following sections in this chapter, we discuss the achievements from our work, and the problems we resolved in deterministic policy gradient methods in reinforcement learning.

5.1 Effect of Exploration

In section 4.2 we presented our results for the effect of exploration in deterministic policy gradients. Our results in figure 4-5 and figure 4-11 showed that in both vanilla and natural deterministic gradients, there is an optimal value of exploration that can achieve the highest convergence rate and converges to a good local optimal policy. In contrast, using a deterministic off-policy exploration, we found that our algorithms get stuck in the worst local optima present in the policy space of our Toy MDP.

Our results have been shown for the Toy benchmark MDP having multiple local reward blobs. Our technique can be extended to more complicated MDPs such as the Mountain Car or Cart Pole, to consider the effect of exploration. However, this is computationally inconvenient since it would require running our experiments over a grid of sigma exploration values, to find the optimal exploration parameter.

However, for the algorithms considered in our work, using the Toy MDP, we demonstrated that the exploration parameter is dependent only on the MDP (since it is used to obtain the state, action trajectories on which the gradient is computed) and is independent of the different methods of deterministic policy gradient algorithms (vanilla, natural or momentum-based algorithms all used the same σ parameter that achieves the optimal state and action space exploration). Results have shown that with no exploration, the agent gets stuck in the worst local blob, while fine tuning to find optimal value of exploration (σ) can significantly improve local optimal convergence.
Conclusion

Our achievement in this work therefore has been to prove that, in deterministic gradients, for each MDP, we can fine tune the off-policy stochastic exploration parameter to improve the convergence of deterministic policy gradient algorithms. For future directions of work, to find the best performing vanilla and natural deterministic policy gradient algorithms, a useful extension would be to have an automatic selection of exploration parameter that can ensure good performance for the deterministic policy gradient algorithms on any domain. It would be ideal to have an automatic way to learn an optimal value of this exploration parameter for any MDPs.

5.2 Resolving Issue of Learning Rate

We showed our results for comparison of our deterministic policy gradient algorithms using the grid search of step size parameter values. In section 4.3.1 we then demonstrated that all our algorithms of comparison are also dependent on the learning rate of gradient ascent for convergence rate to a locally optimal policy. Then in section 4.3.2 we considered the adaptive step size policy gradient algorithms, that defines a new metric for policy gradients, and uses a method that can measure the effect of the change in average reward with respect to the policy parameters. Our results from section 4.6.4 figure 4-20 demonstrated the difference in performance of using the adaptive step size approach compared to the conventional approach considered in policy gradient literature.

Our results from figure 4-20 illustrated that the adaptive step size approach can achieve a similar or even better performance (in case of Natural DPG specifically) compared to the conventional step size policy gradient approach, by avoiding the large cluster of grid search to find optimal learning rate/step size parameters. We therefore demonstrated from our experimental results that the adaptive step size policy gradient approach can significantly improve the performance (both in terms of convergence rate and better local optima convergence) of our deterministic policy gradient algorithm. Our results are the first to take this approach for deterministic policy gradients (both for vanilla and natural gradients), and we have demonstrated an improvement in convergence of the deterministic policy gradient algorithms, while also eliminating the need for fine tuning of learning rate parameters.

Issues With Adaptive Step Size Policy Gradients

The adaptive step size approach to policy gradients ensures good performance and can eliminate the need for grid search over learning rate parameter. However, one drawback with this approach is that it only converges to a near optimal policy, and not to the exact local optima. This is because, as discussed earlier, at the local optima, the gradient becomes zero (hence adaptive step size becomes infinite), and so the algorithm only converges at a near optima. We then showed that using an adaptive initial step of gradient ascent, and having a fixed parameter for the decay rate, we can also eliminate the need for grid search,
and also avoid the disadvantages with the adaptive step size method.

We have therefore partially solved the problem of having an automatically adjustable learning rate, and therefore leaves room for further improvement. This indeed, is not only a problem for reinforcement learning policy gradient literature, but a generic problem to deep neural networks and other methods involving optimization techniques with stochastic gradient descent in general. Recent work from [Schaul et al., 2013] considered finding an automatic selection of learning rate parameters in stochastic gradient ascent. These methods are not yet extendable for policy gradient algorithms, since their work is focused on second-order optimization techniques involving the Hessian matrix. However, recent research from [Tom and Lever et al.,] is focused on deriving a policy Hessian theorem, to derive the Hessian matrix to be used in policy gradient literature, such that second-order optimization techniques can also be applied in this line of research. This, indeed is part of our future work, that we have started working on since the end of this undergraduate project.

**Conclusion**

Our achievement in this work therefore has been to first highlight the effect of learning rate parameters on convergence of policy gradient algorithms. Using a thorough literature search, we then found that we can use an adaptive step size method to automatically adjust learning rate parameters in gradient ascent. Our analysis showed that, even though the adaptive method improves performance, there are considerable disadvantages of using this method, and we have only semi-solved this open problem. Finally, however, we concluded that, in order to eliminate the need for fine tuning the learning rate parameters, we can alternatively use an adaptive initial step size to choose our learning rate of stochastic gradient ascent in policy gradient methods, and this method has so far shown significant improvements in experimental results, that are not part of our work in this project.

### 5.3 Improving Convergence Rates

In section 4.6.3 figure 4-21 we compared our results for the adaptive step size approach, and illustrated that the adaptive step size DPG algorithms performs as good as the stochastic counterparts. Our results comparing all our algorithms in section 4.6.5 demonstrated that we can achieve high convergence rate to a good local optimal policy using the adaptive step size deterministic policy gradients. Additionally, we are the first to show experimenal results of the natural deterministic, and the momentum-based deterministic policy gradients.

In section 4.8 we considered using a modified step size parameter, that slightly can reduce the need for grid search over learning rate parameters. Our "modified step size" method involves changing the initial step size only, while keeping the decay rate of the step size fixed. Using such a technique, in section 4.8 we have shown that the modified step size
method can ensure better local optimal convergence for the Grid World benchmark MDP compared to all the other algorithms (as shown in figure 4-25), even though the convergence rate can be slow.

**Conclusion**

From our results, we can conclude that the adaptive natural deterministic policy gradient is our best performing DPG algorithm on the Toy MDP having multiple local reward blobs on the MDP. This algorithm can achieve a high convergence rate converging to a good local optima, and outperforms all other deterministic policy gradient algorithms that we considered. However, we also discussed the problems using the adaptive step size method, and then considered using the "modified step size" approach to improve local optima convergence of the DPG algorithms. Our analysis showed that, by automatically adjusting the learning rates, and by considering the natural version of the deterministic gradient, we can improve the convergence of deterministic policy gradient algorithms on benchmark MDPs.

### 5.4 Generalizing To Other Benchmark MDPs

Our adaptive natural deterministic policy gradient outperformed all other algorithms on the three MDPs we considered in our work. We then considered the Grid World MDP to compare all our DPG algorithms. Results in section 4.7.1 again demonstrated that our adaptive natural deterministic gradient algorithm outperformed all other algorithms even for the more complex 2D Grid World MDP. This convinced us, that among all the best performing algorithms with optimal parameters, our adaptive NDPG outperforms both in the Toy and Grid World MDP having multiple local reward blobs. We then considered using our best performing algorithm on a more complex Cart Pole MDP, as shown in figure 4-24. Experimental results showed that ANDPG algorithm even performs well on the complex Cart Pole MDP achieving close to a good local optimal policy.

In section 4.8, we then showed how using the modified step size approach, we can reduce the need for grid search, while also ensuring better policy convergence. Finally, in figure 4-24, we compared our algorithms on the more difficult Cart Pole MDP task, and showed good performance on the Cart Pole MDP. To conclude, our results have therefore shown that the adaptive natural deterministic gradient also performs well on other complex benchmark MDPs, and so our algorithm is sound over several domains.

### 5.5 Future Work and Open Problems

**Global Optimal Convergence of Policy Gradient Methods**

Convergence of policy gradient algorithms to a global optimal policy still remains an open problem. For this, a natural future direction of work can be to try using a different optimization technique that can ensure global convergence of these algorithms. Simulated
Annealing based global optimization techniques has recently been used in the context of reinforcement learning [Atiya et al., 2003], [Stefan, 2003]. Work on deterministic policy gradients can be extended using a simulated annealing based global optimization techniques to see the performance of deterministic gradient algorithms ensuring global optimal policy convergence on benchmark MDPs.

**Automatic Selection of Learning Rate Parameters**

A natural open problem to policy gradient methods is to find the optimal learning rate parameters for optimization. To this date, there is still no automatic way to find a good learning rate for policy gradient optimization techniques. Recent work considered using adaptive step size methods that we adopted in our work for deterministic policy gradients. For future work, it would be ideal to have a better method for automatic learning rate parameter selection, since this problem scales up for more complex tasks like learning to play Atari games. Even recent work from Google DeepMind [Mnih et al., 2013] considered fine tuning of parameters to achieve optimal performance on reinforcement learning gaming tasks.

**Policies parameterized by neural networks**

Another extension to our work is to consider using stochastic and deterministic policies that are parameterized by a neural network. This method has been demonstrated in the Octopus Arm experiment in [Silver et al., 2014] [Heess et al., ] using a multi-layer perceptron to parametrize deterministic policies. An interesting future direction of our work will be to consider natural actor-critic deterministic policy gradients parameterized by neural networks, and compare its performance with the other deterministic policy gradients that we considered here.

**Policies in the Reproducing Kernel Hilbert Space (RKHS)**

Our work can be extended by modelling policies in MDPs in vector valued reproducing kernel Hilbert spaces (RKHS) [Lever, et al., 2015], and we can extend our work by considering deterministic policy gradients in the RKHS space. We can also consider our adaptive natural deterministic actor-critic policy gradient in the RKHS space, such that we can work non-parametrically in a richer function class. [Lever et al., 2015] presented a framework where they can work on performing gradient based policy optimization in the RKHS such that the policy representation can adapt to the complexity of the policy. Future work can be extended for considering natural deterministic gradients in the RKHS for benchmark tasks.
Chapter 6

Conclusions

In this undergraduate thesis, we considered the effect of off-policy stochastic exploration on the convergence rate in deterministic policy gradients. We analyzed our algorithms on MDPs that have multiple local optima in the policy space, and studied the effect of exploration in converging to a good local optimal policy. With no exploration, our results showed convergence to the worst local optima, while fine tuning the exploration parameter can ensure fast convergence rate to a good optimal policy.

We considered the problem of fine tuning learning rate parameters in policy gradient methods, and how this fine tuning affects the convergence rate of our algorithms. We then showed our attempts towards using an adaptive step size policy gradient method that can automatically adjust the learning rates. Our results showed that, using the adaptive natural deterministic gradients, we could ensure optimal performance on our benchmark tasks. Our results are interesting since the adaptive step size method can eliminate the need for grid search over learning parameters, while also achieving good local policy convergence. We also considered the disadvantages of adaptive step size methods, and finally showed that we can modify the learning rate by having an adaptive initial step size, and therefore have partially solved the problem of fine tuning learning rate parameters.

We considered algorithms to improve the convergence rates of deterministic policy gradients. Our analysis used natural and momentum-based stochastic gradient ascent approach in the deterministic policy gradient methods, and compared our vanilla, natural, momentum-based and adaptive step size policy gradient algorithms together for the benchmark tasks. In our project, we are the first to consider experimental framework of the natural deterministic policy gradients on benchmark tasks. We showed that using our adaptive learning rate approach combined with natural deterministic gradients, we can achieve good local optimal convergence of this algorithm on several benchmark domains. By providing a unifying review of policy gradient methods, and considering the practical aspects of these algorithms, we showed that we can improve and speed up the local optimal convergence of deterministic policy gradient algorithms in reinforcement learning.
Appendix A

Bibliography
Bibliography


Appendix B

Extra Figures

In Appendix B, we include some of the extra figures that we have from some of our experiments.

We mainly include figures for some of the results using the stochastic policy gradient, and compare this with results from deterministic gradients shown in the report. This is mainly because stochastic gradient algorithms are not the main concern of our work - but we had to implement this algorithm for comparison with the deterministic policy gradients.
Figure B-1: Convergence to Local Optima - Stochastic Policy Gradient Toy MDP

Figure B-2: Stochastic Policy Gradient on Toy MDP using Line Search Optimization
Figure B-3: Effect of Exploration on Vanilla Deterministic Policy Gradient - Toy MDP with multiple local reward blobs

Figure B-4: Stochastic Policy Gradient With Nesterov’s Accelerated Gradient
Figure B-5: Effect of $\epsilon$ parameter using Adaptive Vanilla Stochastic Policy Gradient

Figure B-6: Effect of Step Size parameters on convergence rates on Vanilla DPG
Appendix C

Policy Gradient Theorem

We provide the derivation of the stochastic policy gradient theorem, which was fundamental to our understanding of implementing the stochastic policy gradient, and then the deterministic policy gradient theorem.

\[
\frac{\partial V^\pi(s)}{\partial \theta} = \frac{\partial}{\partial \theta} \sum_a \pi(s,a)Q^\pi(s,a)
\]

\[
= \sum_a \left[ \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a) + \pi(s,a) \frac{\partial}{\partial \theta} Q^\pi(s,a) \right]
\]

\[
= \sum_a \left[ \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a) + \pi(s,a) \frac{\partial}{\partial \theta} [R^\pi(s) - \rho(\pi) + \sum_{s'} P_{ss'}^\pi V^\pi(s')] \right]
\]

\[
= \sum_a \left[ \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a) + \pi(s,a) [\frac{\partial}{\partial \theta} - \sum_{s'} \frac{P_{ss'}^\pi}{\theta} \frac{\partial V^\pi(s')}{\partial \theta} - \sum_s d^\pi(s) \frac{\partial V^\pi(s)}{\partial \theta}] \right]
\]

and from there we can find \( \frac{\partial \rho}{\partial \theta} \)

\[
\frac{\partial \rho}{\partial \theta} = \sum_a [\frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a) + \pi(s,a) \sum_{s'} \frac{P_{ss'}^\pi}{\theta} \frac{\partial V^\pi(s')}{\partial \theta} - \sum_s d^\pi(s) \frac{\partial V^\pi(s)}{\partial \theta}]
\]

\[
\sum_s d^\pi(s) \frac{\partial \rho}{\partial \theta} = \sum_s d^\pi(s) \sum_a [\frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a) + \sum_s \pi(s,a) \sum_{s'} \frac{P_{ss'}^\pi}{\theta} \frac{\partial V^\pi(s')}{\partial \theta} - \sum_s d^\pi(s) \frac{\partial V^\pi(s)}{\partial \theta}]
\]

and since \( d^\pi \) is stationary, we eventually get the expression for policy gradient as follows

\[
\frac{\partial \rho}{\partial \theta} = \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a)
\]

Policy Gradient with Function Approximation
Function Approximators are very important in the analysis of our work, and it is also vitally required for future work when deriving the deterministic policy gradient theorem. We include the proof here for policy gradient with function approximation.

$f_w$ is our approximation to $Q^\pi$ with parameter $w$ and we learn $f_w$ by following $\pi$ and updating $w$. On convergence to a local optimum, we get:

$$\sum_s d^\pi(s) \sum_a \pi(s,a)[Q^\pi(s,a) - f_w(s,a)] \frac{\partial f_w(s,a)}{\partial w} = 0$$

If $f_w$ satisfies the above equation, and if the equation below satisfies

$$\frac{\partial f_w(s,a)}{\partial w} = \frac{\partial \pi(s,a)}{\partial \theta} \frac{1}{\pi(s,a)}$$

then we can say that $f_w$ is compatible with policy parameterization and we can write

$$\frac{\partial \rho}{\partial \theta} = \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} f_w(s,a)$$

$$\sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} [Q^\pi(s,a) - f_w(s,a)] = 0$$

and this tells us that the error in $f_w(s,a)$ is orthogonal to the gradient of the policy parameterization. Eventually, we can write

$$\frac{\partial \rho}{\partial \theta} = \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} f_w(s,a)$$

where $f_w(s,a)$ is our approximated $Q$ function, and we have evaluated our function approximator using least squares temporal difference learning.