POLITECNICO DI MILANO

Facoltà di Ingegneria

Scuola di Ingegneria Industriale e dell'Informazione

Dipartimento di Elettronica, Informazione e Bioingegneria

Master of Science in

Computer Science and Engineering – Ingegneria Informatica

Adaptive Batch Size for Policy Gradient Methods

Supervisor:

MARCELLO RESTELLI

Assistant Supervisor:

MATTEO PIROTTA

Master Graduation Thesis by:

MATTEO PAPINI

Student Id n. 850421

Academic Year 2016-2017

ACKNOWLEDGMENTS

Todo

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

MDP	Markov Decision Process
RL	Reinforcement Learning

ABSTRACT

Todo

Todo

1 INTRODUCTION

This is an introduction.

2

PRELIMINARIES ON POLICY GRADIENT

In this chapter we provide basic knowledge on the reinforcement learning framework and on the policy gradient approach. The aim is to introduce concepts that will be used extensively in the following chapters. For a complete treatment of reinforcement learning, see [2].

2.1 CONTINUOUS MARKOV DECISION PROCESSES

In this section we provide a brief treatment on the model at the root of most Reinforcement Learning (RL) algorithms, including policy gradient methods: the Markov Decision Process (MDP). Given the scope of our work, we will focus on continuous MDPs.

2.1.1 *The model*

Markov decision processes allow to model the very general situation in which an agent interacts with a stochastic, partially observable environment in order to reach a goal. At each time step t, the agent receives observations from the environment. From the current observations and previous interactions, the agent builds a representation of the current state of the environment, s_t. Then, basing its decision on s_t and any other source of knowledge, it takes action a_t. Finally, it is given a (possibly negative) reward depending on how much its behavior is compliant with the goal, in the form of a scalar signal r_t. The dynamics of the environment, as represented by the agent, must satisfy the Markov property: s_{t+1} must depend only on s_t and a_t. Under this framework, the goal of the agent can be restated as the problem of collecting as much reward as possible. For now, and every time it is not specified otherwise, we will refer to continuing (i.e. never-ending) tasks, in which future rewards are exponentially discounted.

Formally, a discrete-time continuous Markov decision process is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mu \rangle$, where:

- $\mathcal{S} \subseteq \mathbb{R}^n$ is an n-dimensional continuous state space. Elements of this set are the possible states of the environment as observed by the agent.
- $\mathcal{A} \subseteq \mathbb{R}^m$ is an $m-$dimensional continuous action space, from which the agent can draw its actions.

- $\mathcal{P}\colon \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$ is a Markovian transition model, where $\mathcal{P}(s' \mid s, a)$ defines the transition density between states s and s' under action a, i.e. the probability distribution over the next state s' when the current state is s and the taken action is a.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to [-R, R]$ is the reward function, such that $\mathcal{R}(s, a)$ is the expected value of the reward received by taking action a in state s and $R \in \mathbb{R}^+$ is the maximum absolute-value reward.
- $\gamma \in [0, 1)$ is the discount factor for future rewards.
- $\mu \in \Delta(\mathcal{S})$ is the initial state distribution, from which the initial state of the environment is drawn.

The behavior of the agent can be defined as a stationary, possibly stochastic policy:

$$\pi\colon \mathcal{S} \mapsto \Delta(\mathcal{A}),$$

such that $\pi(s)$ is the probability distribution over the action a to take in state s.

2.1.2 *Problem formulation*

The goal of RL is to find an optimal policy π^* by exploiting the information obtained through the interaction with the environment. The objective is to maximize the total discounted reward, called return v:

$$v = \sum_{t=0}^{\infty} \gamma^t r_t$$

To evaluate how good a policy π is, we need to compute the expected value of the return over π and the initial state distribution μ:

$$J_\mu^\pi = \mathbf{E}_{\mu,\pi}[v].$$

It is convenient to introduce a new distribution, called stationary distribution or future-state distribution d_μ^π:

$$d_\mu^\pi(s) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \Pr(s_t = s \mid \pi, \mu),$$

which allows to define the expected return J as:

$$J_\mu^\pi = \int_{\mathcal{S}} d_\mu^\pi(s) \int_{\mathcal{A}} \pi(a \mid s) \mathcal{R}(s, a) \mathrm{d}a\mathrm{d}s.$$

The RL problem can then be formulated as a stochastic optimization problem:

$$\pi^* \in \arg\max_\pi J(\pi)$$

2.1.3 *Value functions*

Value functions provide a measure of how good a state, or a state-action pair, is under a policy π. They are powerful theoretical tools and are employed in many practical RL algorithms. The state value function $V^\pi\colon \mathcal{S} \mapsto \mathbb{R}$ assigns to each state the expected return that is obtained by following π starting from state s, and can be defined recursively with the following equation:

$$V^\pi(s) = \int_{\mathcal{A}} \pi(a|s) \left(\mathcal{R}(s,a) + \gamma \int_{\mathcal{S}} \mathcal{P}(s'|s,a) V^\pi(s') ds' \right) da,$$

which is known as Bellman's expectation equation. The expected return $J_\mu(\pi)$ can be expressed in terms of the state-value function as:

$$J_\mu^\pi = \int_{\mathcal{S}} \mu(s) V^\pi(s) ds.$$

For control purposes, it is more useful to define an action-value function $Q^\pi\colon \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, such that $Q^\pi(s,a)$ is the return obtained starting from state s, taking action a and following policy π from then on. Also the action-value function is characterized by a Bellman equation:

$$Q^\pi(s,a) = \mathcal{R}(s,a) + \gamma \int_{\mathcal{S}} \mathcal{P}(s'|s,a) \int_{\mathcal{A}} \pi(a'|s') Q^\pi(s',a') da' ds'.$$

2.2 POLICY GRADIENT METHODS

In this section we describe in general terms a special class of RL algorithms known as policy gradient methods, and we report some theoretical results that are of key importance for any treatment of such methods.

2.2.1 *Definition*

Direct policy search is an approach to RL consisting in exploring a subset of policy space directly to find an optimal policy. It is opposed to the value function approach, in which the optimal policy is derived from an estimate of the action value function Q^π. The advantages of direct policy search over value function methods for continuous MDPs are discussed in the following chapter. For a recent survey on policy search methods refer to [1].

Policy gradient is a kind of policy search in which the search is restricted to a class of parametrized policies:

$$\Pi_\theta = \{\pi_\theta : \theta \in \mathbb{R}^m\},$$

where θ is the parameter vector. The parametric policy π_θ can have any shape, but is required to be differentiable in θ. Like all gradient

descent methods, the parameter vector is updated in the direction of the gradient of a performance measure, which is guaranteed to be the direction of maximum improvement. In our case the performance measure is the expected return $J_\mu^{\pi_\theta}$, which we rename $J_\mu(\theta)$ for readability. The gradient of the expected return w.r.t. the parameter vector, $\nabla_\theta J_\mu(\theta)$, is called policy gradient. Starting from an arbitrary initial parametrization θ_0, policy gradient methods performs updates of the kind:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J_\mu(\theta),$$

where $\alpha \geqslant 0$ is the learning rate, also called step size. The update is performed at each learning iteration. Convergence to at least a local optimum is guaranteed by the gradient descent update.

2.2.2 *Stochastic gradient descent*

In many practical tasks it is impossible, or unfeasible, to compute the exact policy gradient $\nabla_\theta J_\mu(\theta)$, but a gradient estimate $\hat{\nabla}_\theta J_\mu(\theta)$ can be estimated from sample trajectories. With the term trajectory we mean an episode of the task starting from an initial state s_0, drawn from μ, and stopping after a fixed number H of time steps. The gradient estimate is often averaged over a number N of such trajectories, also called a batch. The size N of the batch is called batch size. The stochastic gradient descent update is:

$$\theta \leftarrow \theta + \alpha \hat{\nabla}_\theta J_\mu(\theta),$$

where, at each learning iteration, N trajectories need to be performed. Convergence to a local optimum is still guaranteed, provided that the angular difference between $\nabla_\theta J_\mu(\theta)$ and $\hat{\nabla}_\theta J_\mu(\theta)$ is less than $\pi/2$.

2.2.3 *Policy Gradient Theorem*

The Policy Gradient Theorem is a result by Sutton [3] that relates the policy gradient to the value function Q^π:

Theorem 2.1 (Continuous version of Theorem 1 from [3])**.** For any MDP

$$\nabla_\theta J_\mu(\theta) = \int_S d_\mu^\pi(s) \int_A \nabla_\theta \pi_\theta(a \mid s) Q^\pi(s,a) \, da \, ds.$$

This theorem is of key importance for most policy gradient algorithms.

3

SAFE POLICY GRADIENT METHODS: STATE OF THE ART

4

JOINT STEP-SIZE AND BATCH-SIZE OPTIMIZATION

5 NUMERICAL SIMULATIONS

6 CONCLUSION

This is a conclusion.

BIBLIOGRAPHY

[1] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. "A Survey on Policy Search for Robotics." In: *Foundations and Trends in Robotics* 2.1-2 (2013), pp. 1–142 (cit. on p. 5).

[2] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. 1st. Cambridge, MA, USA: MIT Press, 1998. ISBN: 0262193981 (cit. on p. 3).

[3] Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. "Policy Gradient Methods for Reinforcement Learning with Function Approximation." In: *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*. Ed. by Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller. The MIT Press, 1999, pp. 1057–1063. ISBN: 0-262-19450-3 (cit. on p. 6).