Toward Emphatic Reinforcement Learning

by

Jingjiao Ni

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Emphatic-Temporal-Difference (Emphatic-TD) learning algorithms were recently proposed based on the most central and widely used reinforcement learning algorithms, Temporal-Difference (TD) methods. Emphatic-TD learning algorithms were originally designed to solve the divergence problem of conventional TD methods when they are applied to off-policy training. However, recent studies on Emphatic-TD learning have shown that Emphatic-TD methods can outperform conventional TD methods even in the on-policy case on prediction problems. Thus we are interested in how Emphatic-TD methods can in general be extended for on-policy control in this thesis. Also, Emphatic-TD methods are sensitive to the step-size parameter, and inappropriate step-size parameters will lead to divergence, which is called "the sensitivity problem" in this thesis. We encountered this problem during the empirical studies on Emphatic-TD methods, thus we provide a solution to this sensitivity problem.

In this thesis, we will make contributions in two separate but correlated areas. First, we proposed new heuristics for reliably adapting step sizes to emphatic methods for the sensitivity problem of the step-size parameter. Second, we extended the idea of emphatic methods to the on-policy control methods and proposed the new n-step Emphatic-Sarsa method and Emphatic-Sarsa(λ) method. We also conducted some empirical studies for them and our empirical results showed that both the step-size heuristics and the new on-policy emphatic control methods worked, and the new on-policy emphatic control methods outperformed the corresponding non-emphatic methods in some particular cases. A limitation of our work is that our empirical studies on the on-policy control methods is only designed in the equal-interest case.

Preface

No part of this thesis has been published before.

To my parents

without them I would not be here.

The best time to plant a tree is 20 years ago. The second-best time is now.

– Chinese Proverb.

Acknowledgements

I want to give my foremost thank to my supervisor Dr. Richard S. Sutton for his help and contribution in completing this thesis. His wisdom and patience not only help me in deepening my knowledge of reinforcement learning, but also benefit me all my life.

I also want to thank the PhD student Sina Ghiassian for sharing his understand of the emphatic methods with me and helping revise my thesis. He is an expert in the field of the emphatic methods, and I can learn a lot after each talk with him.

Lastly, I would like to thank my parents Yan Liu and Xiaodong Ni for their continuous support.

Contents

Chapter 1

Introduction

In this chapter, an overview of our studies is presented. We will briefly introduce the purpose, evaluation, contribution and scope of this thesis.

1.1 Temporal-Difference Learning

In recent years, the field of reinforcement learning is fast developed because of the rapid growth of computational power, and Temporal-Difference (TD) learning is undoubtedly the heart of reinforcement learning. Like most reinforcement learning methods, TD learning can be used to solve both prediction problems and control problems. Given a dynamical system, a prediction problem is to compute the state-value function based on an arbitrary policy while a control problem is to find an optimal policy. TD prediction methods use experiments following the policy to update the estimate value of the state-value for each non-terminal state. The update is based on the TD error, which is the difference between the previous estimated state-value and a more precise estimate based on the reward and the estimate of the next state-value. Because the update of TD methods estimate is based on a previous estimate, we call this kind of methods the bootstrapping methods. Two popular TD methods we used in this thesis are the n-step TD method (Sutton & Barto, 1998) and the TD(λ) method (Sutton, 1988). TD methods can also be used to solve control problems by following the pattern of generalized policy iteration (GPI). Instead of learning the state-value function in prediction problems, TD control methods learn action-value functions. Depending on if the behavior policy is the same as the target policy, control methods can be classified into on-policy methods and off-policy methods. Sarsa (Rummery & Niranjan, 1994) is a common on-policy control algorithm while Q-Learning (Watkins, 1989) is a common off-policy control algorithm. Because we focus on the on-policy emphatic control methods in this thesis, we will introduce the Sarsa methods which are on-policy in more detail in the background chapter.

1.2 Emphatic Methods

In conventional reinforcement learning algorithms such as conventional TD methods, all the states experienced are usually treated equally. This is based on all the states experienced have same importance. However, in practice there are situations where some states are more important than others or we have more interests in some particular states. We can improve the performance of algorithms by focusing on more valued parts due to the limitation of computational resources when using function approximation. Another reason for treating all the states experienced equally in conventional TD methods is that the efficiency and stability of linear semi-gradient TD methods are only guaranteed under on-policy distribution. In off-policy learning, we use importance sampling to reweight the state transition, but the state distribution remains unchanged in conventional TD methods so there is a mismatch in the state distribution. Emphatic methods are designed to solve this problem by emphasizing or de-emphasizing the updates with a scalar measure called emphasis. The first and typical emphatic method is the Emphatic-TD(λ) method which is introduced as an off-policy method by Sutton, Mahmood, and White (2016). The convergence proofs and some other studies on the Emphatic-TD methods are developed by Yu (2015), Yu (2016), Mahmood, Yu, White, and Sutton (2015), Hallak, Tamar, and Mannor (2015) and Hallak, Tamar, Munos, and Mannor (2016). Then some studies such as Ghiassian, Rafiee, and Sutton (2017) and Gu, Ghiassian, and Sutton (2019) showed that Emphatic-TD methods also outperformed conventional TD methods as on-policy prediction methods. This makes us interested in the performance of the possible emphatic on-policy control methods and their performance comparing to the corresponding non-emphatic methods. One of the few studies on emphatic control methods such as the Actor Critic with Emphatic weightings (ACE) method introduced by Imani, Graves, and White (2018) as an off-policy actor-critic algorithm. Though the original ACE paper focuses on the off-policy learning, the ACE method can be considered as an on-policy algorithm if the importance sampling ratio is set to 1, thus we would like to focus on extending the idea of emphatic methods to the on-policy action-value methods. In particular, we extended the idea of emphatic methods to the conventional on-policy action-value methods Sarsa and compared their performance empirically in this thesis.

1.3 The Sensitivity Problem of the Step-size Parameter in Emphatic Methods

The emphasis is the core of the emphatic methods, but the emphasis with inappropriate step-size parameters can together make the update too large which leads to divergence. Thus it is hard for emphatic methods to find appropriate ranges of step-size parameters, and we call it "the sensitivity problem" of the step-size parameters in emphatic methods in this thesis. Karampatziakis and Langford (2011) first demonstrated a problem of machine learning algorithms when importance weights were large and developed the technique of sliding step for dealing with it. The importance weights play a similar role as the emphasis of emphatic methods which makes these two problems correlated, but Karampatziakis and Langford (2011)'s work covered nothing about reinforcement learning or emphatic methods. Some previous studies of emphatic methods provided hints of the existence of the sensitivity problem of the step-size parameter. The weak convergence properties of the Emphatic-TD(λ) method with constant and slowly diminishing step-size parameters is presented by Yu (2016) with some constraints and assumptions. Ghiassian, Patterson, White, Sutton, and White (2018)'s work also showed that it is hard to find Emphatic-TD(0)'s step size for which it converges to its best final performance. And we found that the step-size parameter of the Emphatic-TD method is more sensitive than the corresponding non-emphatic method during our empirical studies so we would like to find a solution to this sensitivity problem. Tian and Sutton (2019) extended Karampatziakis and Langford (2011)'s idea of sliding step to the original Emphatic-TD method and formed an interesting new algorithm without the sensitivity problem, the sliding-step Emphatic-TD method. Because we would like to study the original Emphatic-TD method in the on-policy control cases, we would like to find some techniques to help find appropriate ranges of step-size parameters without changing the original Emphatic-TD method. Thus in this thesis, we proposed new heuristics for reliably adapting step sizes to emphatic methods, and they are used in our empirical studies of emphatic on-policy control methods afterwards.

1.4 The Contribution of this Thesis

In this thesis, we will make contributions in two separate but correlated areas. First, we showed the sensitivity problem specifically for emphatic methods empirically, and proposed new heuristics for reliably adapting step sizes to emphatic methods for the sensitivity problem of the step-size parameter. Our step-size heuristics reveal the relationship between conventional TD methods and Emphatic-TD methods which can help to find an appropriate range of the step-size parameter. We analysed our step-size heuristics theoretically, and some empirical studies were also conducted to evaluate them. Second, we extended the idea of emphatic methods to the on-policy control action-value methods and proposed the new n-step Emphatic-Sarsa method and Emphatic-Sarsa(λ) method. We also conducted some empirical studies for them. Our empirical results showed that the two new on-policy emphatic control methods worked, and they outperformed the corresponding non-emphatic methods in some special cases. A limitation of our work is that our empirical studies on the on-policy control methods are only designed in the equal-interest case.

Chapter 2

Background

In this chapter, we will introduce some background information related to this thesis including some terminologies and equations. Readers who are familiar with these subjects can skip this chapter without any loss of continuity because the following information are standard. Most content of this chapter is based on Sutton and Barto (2018)'s book Reinforcement Learning: An Introduction.

The notations and terminologies in this thesis are also aligned with the book Reinforcement Learning: An Introduction (Sutton & Barto, 2018).

2.1 Markov Decision Processes (MDPs)

Markov decision processes (MDPs) are discrete time stochastic control processes (Bellman, 1957). MDPs provide an ideal mathematical framework for the reinforcement learning problems because we can make strong theoretical statements. In MDPs, we call the learner who makes decisions the *agent*. We call anything outside the agent the *environment*. The agent interacts with the environment at each of a sequence of discrete time steps $t = 0, 1, 2....$ The agent gets the representation of the environment which is called the *state* $S_t \in \mathcal{S}$ and then it chooses a action $A_t \in \mathcal{A}(s)$ based on the state it observed. At the time step $t + 1$, the agent will receive the numerical feedback from the environment which is called the *reward* $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$. At the same time, the agent will be in a new state S_{t+1}.

2.2 Value Functions and Prediction Problems

In reinforcement learning, the goal of the agent is to find a policy to receive the maximum cumulative total rewards which is defined as the expected *return* G_t. To reach this goal, the agent needs to know how good for it to be in a given state. We use *value functions* as a estimator of the quality of a state. The value function of a state s under the policy π $v_\pi(s)$ is defined as the expected return the agent will receive in a given state s if it follows a particular policy π. In MDPs, the value function can be defined as

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s\right], \text{ for all } s \in \mathcal{S}, \tag{2.1}$$

where $\mathbb{E}_\pi[\cdot]$ is the expected value of $\cdot$ if the agent follows the policy π. The discount-rate parameter γ determines the current value of the future rewards. And how to compute such *state-value function for policy* π is referred as the *policy evaluation* or the *prediction problem.*

There is also a similar *action-value function for policy* π which is defined as the expected return the agent will receive after it takes action a in state s if it follows a particular policy π afterwards. This is denoted as $q_\pi(s, a)$:

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a\right]. \tag{2.2}$$

2.3 Generalized Policy Iteration (GPI) and Control Problems

The goal of the agent is to find the optimal policy π_* which can maximize the expected return, and how to find such optimal policy is considered as the *control problem* in reinforcement learning. First we can use the policy evaluation to compute the value functions v_π under some arbitrary policy π. Selecting the actions greedily based on the value functions can lead to a better policy π'. This process is called *policy improvement.* After the policy improvement, we can do policy evaluation again under the new policy π' to get the new value functions $v_{\pi'}$, and then improve the policy again to get a new better policy π''. By alternately doing policy evaluation and policy improvement, the policy is guaranteed to be better than the previous one until the policy converged to the optimal policy π_*. The policy evaluation and policy improvement can be done asynchronously and still guarantee the convergence. And this general idea is called Generalized Policy Iteration which is used for all control problems in reinforcement learning.

2.4 Temporal-Difference (TD) Learning

There are three basic classes of methods to solve the problems in MDPs: Dynamic Programming (DP) methods, Monte Carlo (MC) methods, and Temporal-Difference (TD) methods. If we are given a perfect model of MDP, DP methods can find the optimal policies. But due to the requirements of the perfect model of the environment and it is also computationally expensive, DP methods have lots of limitations in practice. Unlike DP methods, MC methods don't require the model of the environment, it only rely on experiments. The experiments provide sequences of states, actions and rewards so that MC methods can use them to compute the average sample returns to solve the problem. But classical MC methods can only work for episodic tasks because it can only get the complete average sample returns until the end of the episode. MC methods are thus considered as episode-by-episode incremental computation instead of step-by-step incremental computation, which makes it unsuited for online learning. TD methods are the combination of DP methods and MC methods. Although they are more complex to analyse, TD methods retain some advantages of both DP and MC methods. TD methods make estimates based on some previous estimates just like DP methods, which is called *bootstrapping*, thus TD methods do not need to wait for the completion of an episode like MC methods. TD methods can learn form experiments like MC methods, so they do not rely on the knowledge of the environment like DP methods.

2.5 TD Prediction

First we consider using TD methods for prediction problems. TD methods use experiments to update the estimates V of state-value function v_π for any non-terminal state S_t appeared in the the experiments. The updates of TD methods are based on the *error* which is the difference between the *target* and the old estimate. The target determines the desirable direction to move so the TD methods can reduce the error. The simplest TD method waits just one step to perform the update, and it uses the next reward R_{t+1} and the estimate $V_t(S_{t+1})$ of the state S_{t+1} at time step t. In this case, the update target is called the *one-step return*:

$$G_{t:t+1} \doteq R_{t+1} + \gamma V_t(S_{t+1}). \tag{2.3}$$

The error here is called the *TD error*:

$$\begin{aligned} \delta_t &\doteq G_{t:t+1} - V_t(S_t) \\ &\doteq R_{t+1} + \gamma V_t(S_{t+1}) - V_t(S_t). \end{aligned} \tag{2.4}$$

And the update will be:

$$\begin{aligned} V_{t+1}(S_t) &\doteq V_t(S_t) + \alpha\delta_t \\ &\doteq V_t(S_t) + \alpha[G_{t:t+1} - V_t(S_t)] && (2.5) \\ &\doteq V_t(S_t) + \alpha[R_{t+1} + \gamma V_t(S_{t+1}) - V_t(S_t)]. && (2.6) \end{aligned}$$

This simplest TD method is called *one-step TD* or *TD(0)* because it is the base case of the *n-step TD* method and the *TD(λ)* method.

If the TD method waits n steps instead of one step to make the update, it becomes the n-step TD method. The target of the n-step TD update should be the *n-step return*:

$$\begin{aligned} &G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n}), \quad t+n < T, \\ &G_{t:t+n} \doteq G_t, \qquad t+n \geq T. \end{aligned} \tag{2.7}$$

Similarly, the update then becomes:

$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha[G_{t:t+n} - V_{t+n-1}(S_t)], \quad 0 \leq t < T. \tag{2.8}$$

Note that if we use the ∞-step TD method, the ∞-step return will be the complete return which is used in the Monte Carlo methods and the updates will be the same likewise. The ∞-step TD method is just the Monte Carlo method so we can also say that n-step TD method is a way to unify and generalize TD methods and Monte Carlo methods. The Monte Carlo method and the one-step TD method are two extreme cases of n-step TD method, and intermediate methods are usually better than extreme methods.

The above one-step TD method and n-step TD method are introduced in tabular cases which has limitations on the state spaces. In practice, the state spaces are usually very large so we need a way called *function approximation* to generalize the states. Function approximation can estimate the state-value function v_π from policy π by using a weight vector $\mathbf{w} \in \mathbb{R}^d$. The approximate value of state s given weight vector $\mathbf{w}$ is denoted by $\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$, and the approximate value $\hat{v}$ can be a wide range of functions such as linear functions, neural networks, decision trees, etc. In this thesis, we only consider the simple linear function approximation. In this case, each state s is represented as a real-valued vector called *feature vector* $\mathbf{x}(s) \doteq (x_1(s), x_2(s), \ldots, x_d(s))^\top$, where the length of the feature vector d is the same as the length of the weight vector $\mathbf{w}$. And the estimate $\hat{v}(s, \mathbf{w})$ is just the inner product of the feature vector and the weight vector:

$$\hat{v}(s, \mathbf{w}) \doteq \mathbf{w}^\top \mathbf{x}(s) \doteq \sum_{i=1}^{d} w_i x_i(s). \tag{2.9}$$

By combining the linear function approximation and the tabular n-step TD method with *semi-gradient descent*, we can get n-step semi-gradient TD method which has the n-step return as:

$$\begin{aligned} G_{t:t+n} &\doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{v}(S_{t+n}, \mathbf{w}_{t+n-1}), \quad t+n < T, \\ G_{t:t+n} &\doteq G_t, \qquad t+n \geq T. \end{aligned} \tag{2.10}$$

The update is:

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha[G_{t:t+n} - \hat{v}(S_t, \mathbf{w}_{t+n-1})]\nabla\hat{v}(S_t, \mathbf{w}_{t+n-1}), \quad 0 \leq t < T, \tag{2.11}$$

where $\nabla\hat{v}(S_t, \mathbf{w}_{t+n-1})$ is the column vector of partial derivatives of $\hat{v}(S_t, \mathbf{w}_{t+n-1})$ with respect to $\mathbf{w}_{t+n-1}$. And in the simple linear function approximation, the gradient of the approximate value function with respect to the weight vector is just the feature vector:

$$\nabla\hat{v}(S_t, \mathbf{w}_{t+n-1}) = \mathbf{x}(S_t). \tag{2.12}$$

In addition to the n-step TD method, there is another way to unify and generalize TD methods and Monte Carlo methods which uses *eligibility trace*, and we call it the TD(λ) method. The TD(λ) method is computationally better than the n-step TD method because it just needs to store the eligibility trace $\mathbf{z}_t \in \mathbb{R}^d$ which is a short-term memory vector parallels to $\mathbf{w}_t \in \mathbb{R}^d$ instead of the last n feature vectors in the n-step TD method. In the semi-gradient TD(λ) method, the eligibility trace $\mathbf{z}_t$ will be:

$$\begin{aligned} \mathbf{z}_{-1} &\doteq 0, \\ \mathbf{z}_t &\doteq \gamma\lambda\mathbf{z}_{t-1} + \nabla\hat{v}(S_t, \mathbf{w}_t), \quad 0 \leq t \leq T, \end{aligned} \tag{2.13}$$

where λ is the *trace-decay* parameter which determines the speed of the decay of the trace. If $\lambda = 0$, it is the algorithm TD(0) which uses the one-step semi-gradient TD update. And if $\lambda = 1$, it is the algorithm TD(1) which uses the update of Monte Carlo methods. The TD error of the TD(λ) method using function approximation is:

$$\delta_t \doteq R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t). \tag{2.14}$$

And the update of the TD(λ) method is based on the eligibility trace and TD error:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\delta_t\mathbf{z}_t. \tag{2.15}$$

2.6 On-policy TD Control: Sarsa

There are two kinds of methods to solve the control problems in reinforcement learning: on-policy and off-policy methods. On-policy methods use just one policy to generate behaviours and evaluate that policy, whereas off-policy methods use two different policies. Off-policy methods use one *behavior policy* to generate behaviours but evaluate another *target policy*. The on-policy TD control method is called *Sarsa*. Based on the pattern of GPI, we need to first learn the action-value function $q_\pi(s,a)$ instead of the state-value function $v_\pi(s)$ for the policy evaluation. This can easily be derived from Equation 2.5 as:

$$\begin{aligned} Q_{t+1}(S_t, A_t) &\doteq Q_t(S_t, A_t) + \alpha[G_{t:t+1} - Q_t(S_t, A_t)] && (2.16) \\ &\doteq Q_t(S_t, A_t) + \alpha[R_{t+1} + \gamma Q_t(S_{t+1}, A_{t+1}) - Q_t(S_t, A_t)]. && (2.17) \end{aligned}$$

The name of this algorithm is Sarsa because the above update uses all the elements in the quintuple $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$. After the policy evaluation, we can improve the policy π by making it greedy with respect to the action-value function q_π using methods such as ε-greedy or ε-soft. The complete algorithm is formed by continually alternating these two methods until the policy π converges to the optimal policy.

Obviously this is the tabular one-step Sarsa method, and it can be extended to the n-step version. We just need to change the state-value functions in the n-step TD method to action-value functions and add the ε-greedy method for policy improvement. Then the n-step return will be defined using the action-value function as:

$$\begin{aligned} &G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n}), \quad t+n < T, \\ &G_{t:t+n} \doteq G_t, \qquad t+n \geq T. \end{aligned} \tag{2.18}$$

The update of the action-value function is:

$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha[G_{t:t+n} - Q_{t+n-1}(S_t, A_t)], \quad 0 \leq t < T, \tag{2.19}$$

and all other action-value functions will not change.

It is straightforward to extend the Sarsa methods with function approximation in episodic cases as we did for TD methods. The n-step return is redefined in the function approximation form as:

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1}), \quad t+n < T,$$
$$G_{t:t+n} \doteq G_t, \qquad t+n \geq T. \tag{2.20}$$

The n-step update is redefined as:

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha[G_{t:t+n} - \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1})]\nabla \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1}), \quad 0 \leq t < T. \tag{2.21}$$

By combining the Sarsa method with eligibility traces, we will have the Sarsa(λ) method (Rummery & Niranjan, 1994), and it is only slightly different from the TD(λ) method. The Sarsa(λ) method has the same update as Equation 2.15, but the eligibility trace for the Sarsa(λ) method should be:

$$\begin{aligned} \mathbf{z}_{-1} &\doteq 0, \\ \mathbf{z}_t &\doteq \gamma\lambda \mathbf{z}_{t-1} + \nabla \hat{q}(S_t, A_t, \mathbf{w}_t), \quad 0 \leq t \leq T, \end{aligned} \tag{2.22}$$

also the TD error is redefined in terms of the action-value function for the Sarsa(λ) method as:

$$\delta_t \doteq R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t). \tag{2.23}$$

2.7 Tile Coding

We have already described how to estimate the value of a state in linear function approximation case (2.9), and the feature vector is needed for that equation. There are a lot of ways to construct the feature vector, and we will introduce *Tile Coding* here which is the only one used in this thesis. Tile coding is a computationally efficient and flexible feature representation which is widely used in practice.

Let us consider an example where the state can be naturally represented in a two-dimensional space. There are some receptive fields of features called *tilings* which are larger than the state space. All the tilings are of the same size but lay in different places to fully cover the state space. And each tiling is partitioned into some small pieces called *tiles*. If the state is inside a tile of one tiling, the corresponding feature of that tile in that tiling will be 1 which is said to be active or present, and the features of other tiles in that tiling will all be 0 which is said to be inactive or absent. And a state will be active in exact one tile for each tiling. If we have n tilings and each tiling has $x \times y$ tiles, there will be n components having value 1 which are the corresponding active tiles among the total of $n \times x \times y$ components.

The above two-dimensional state space example is just the case of the mountain car example in Section 4.1, but it can also be extended to other dimensions using the same idea such as the one-dimensional state space random walk example in Section 5.3.

2.8 The Objective for Prediction: RMSVE

In this thesis, we will consider the function approximation cases instead of the tabular cases, thus we need a continues measure of the quality of the estimate in the prediction problems. Unlike the tabular cases, function approximation cannot get the exact true value function. We can not just update for one particular state, an update on one state will always influence other states. In most cases, the size of the weights will be smaller than the size of the state space which limits the total accuracy. Making one state more accurate leads to other states being less accurate so we need to find a balance based on how we care about each state. The objective function *Root Mean Squared Value Error* weighted the error for each state by how we care about the error in that state, which is denoted as RMSVE or $\sqrt{\overline{\mathrm{VE}}}$:

$$\sqrt{\overline{\mathrm{VE}}(\mathbf{w})} \doteq \sqrt{\sum_{s \in \mathcal{S}} \mu(s)[v_\pi(s) - \hat{v}(s, \mathbf{w})]^2}, \tag{2.24}$$

where the error in each state is the square of difference between the true value and the estimate value which is the square bracket part. And $\mu(s)$ is the *state distribution* which is how we care about the error in the state s. The state distribution is non-negative $\mu(s) \geq 0$ and sum to 1 $\sum_s \mu(s) = 1$. In general, the state distribution is the fraction of time spend in state s, but because we consider the interest and emphasis in this thesis, the state distribution needs to be reweighted based on the interest of states. Also, we only discuss the on-policy episodic cases in this thesis so we only care about the state distribution under on-policy training which is also called *on-policy distribution.* And the episodic on-policy distribution is defined as:

$$\mu(s) = \frac{I(s)\eta(s)}{\sum_{s'} I(s')\eta(s')}, \qquad \text{for all } s \in \mathcal{S}, \tag{2.25}$$

where $I(s)$ is the interest of state s. The expected number of visits to state s per episode is denoted as $\eta(s)$, and it is counted when the episodes start in the state s or there is a transition to the state s from some preceding state $\bar{s}$, which is formally defined as:

$$\eta(s) = h(s) + \gamma \sum_{\bar{s}} \eta(\bar{s}) \sum_{a} \pi(a|\bar{s}) p(s|\bar{s}, a), \quad \text{for all } s \in \mathcal{S}, \tag{2.26}$$

where $h(s)$ is the probability that an episode will start in the state s.

Chapter 3

Emphatic Methods and New Step-size Heuristics

In this chapter, we introduce the theoretical part of the first area of contribution of this thesis, the new heuristics for reliably adapting step sizes to emphatic methods. We start this chapter by introducing the emphatic methods, then we analyse them and study the relationship between the emphatic methods and the corresponding non-emphatic methods. The relationship helped us propose different sub-heuristics for different emphatic methods and situations.

3.1 Emphatic Methods

For all the algorithms we introduced in the background chapter, each state is treated with equal importance. If we have different interests in different states, the *interest* and the *emphasis* are two variables that can help. The interest $I_t \in [0, 1]$ is a scalar measuring the degree of interest to compute the value functions precisely in time step t. When $I_t = 0$, it means that we have no interest in the state at time t, and $I_t = 1$ indicates that we have full interest in the state at time t. The emphasis M_t is a scalar used as a multiplication rate of the update which can emphasize or de-emphasize the learning of the algorithm in time step t. The general n-step update will be changed from Equation 2.11 to:

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha M_t [G_{t:t+n} - \hat{v}(S_t, \mathbf{w}_{t+n-1})] \nabla \hat{v}(S_t, \mathbf{w}_{t+n-1}), \quad 0 \leq t < T, \tag{3.1}$$

where the n-step return $G_{t:t+n}$ remains the same as Equation 2.10. And the emphasis is defined as:

$$\begin{aligned} &M_t = I_t + \gamma^n M_{t-n}, \qquad 0 \leq t < T, \\ &M_t = 0, \qquad t < 0. \end{aligned} \tag{3.2}$$

Similarly, we can apply the emphasis to the Monte Carlo method and consider it as a variation of the above equations:

$$\begin{aligned} \mathbf{w}_{t+1} &\doteq \mathbf{w}_t + \alpha M_t [G_t - \hat{v}(S_t, \mathbf{w}_t)] \nabla \hat{v}(S_t, \mathbf{w}_t), \quad 0 \leq t < T, \\ M_t &= I_t, \end{aligned} \tag{3.3}$$

where the n-step return $G_{t:t+n}$ is replaced by the complete return G_t. And $n = T - t$ in this case because Monte Carlo methods only update all at the end of each episode. The emphasis is just the interest of the state because Monte Carlo is not a bootstrapping method, and we don't need to consider the discounted emphasis of the states where the current state bootstraps from.

Now we will consider combining the idea of interest and emphasis with the TD(λ) method. Recall that the TD(λ) method uses eligibility traces to unify and generalize TD methods and Monte Carlo methods. When $\lambda = 0$, it is TD(0) which is also the one-step TD method. When $\lambda = 1$, it is TD(1) which behaves just like Monte Carlo methods. Turning to the Emphatic-TD(λ) method, we still want such a method to lie between the one-step Emphatic-TD method and the emphatic Monte Carlo method. Because the emphasis M_t is a scalar to emphasize or de-emphasize the learning update and the update in the TD(λ) method is affected by the degree of bootstrapping controlled by λ, the emphasis in the Emphatic-TD(λ) method also needs to be adjusted by λ. We then introduce another scalar variable, the *followon trace* $F_t \geq 0$. It keeps tracking the contribution of the interest of the visited states. And the emphasis is the sum of the followon trace and the interest of the current state but reweighted by λ. The complete algorithm of the Emphatic-TD(λ) method will then be:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\delta_t\mathbf{z}_t, \tag{3.4}$$

$$\delta_t \doteq R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t), \tag{3.5}$$

$$\mathbf{z}_{-1} \doteq 0,$$

$$\mathbf{z}_t \doteq \gamma\lambda\mathbf{z}_{t-1} + M_t\nabla\hat{v}(S_t, \mathbf{w}_t), \tag{3.6}$$

$$M_t \doteq \lambda I_t + (1-\lambda)F_t, \tag{3.7}$$

$$F_0 \doteq i(S_0),$$

$$F_t \doteq \gamma F_{t-1} + I_t. \tag{3.8}$$

3.2 The Sensitivity Problem in Emphatic Methods and our Solution

As we discussed in Chapter 1, some previous work gave hints about the existence of the sensitivity problem in emphatic methods and we also encountered this problem in Section 4.2. The interest I_t and the emphasis M_t together play an important role in emphatic methods, but they also lead to a potential problem. When the interest of most states are big (close to 1), the emphasis M_t will increase really fast, especially when the discount rate γ is also big (close to 1). The rapid growth of the emphasis will make the absolute value of the update really large after just several time steps which usually leads to divergence if the step-size parameter is inappropriate. Because one of the most common settings is to set the interest of all states equally (usually set the interest of all states to 1) [1], divergence caused by large update is not rare in practice. Moreover, the large absolute value of the update also makes the appropriate range of the step-size parameter narrower which makes finding an appropriate step-size parameter a more difficult task. To solve this problem, a natural idea is to find the relationship between the emphatic methods and the corresponding non-emphatic methods. Thus we try to find the appropriate

[1]Technically this common setting is to set the interest of all states except the terminal states to 1 and the interest of terminal states are set to 0. The value of terminal states is always 0 so we do not care about them. And this applies to the rest of this thesis when we say setting the interest of all states to 1.

range of the step-size parameter for emphatic methods by adapting the step sizes of the corresponding non-emphatic methods.

We can also find some hints of the sensitivity problem in on-policy emphatic methods by studying the fixed point of the emphatic methods and non-emphatic methods. The TD fixed point [2] is defined as:

$$\mathbf{w}_{\text{TD}} = \mathbf{A}^{-1}\mathbf{b}. \tag{3.9}$$

In on-policy TD(0), the $\mathbf{A}$ matrix is:

$$\mathbf{A} = \mathbf{X}^\top \mathbf{D}_\pi (\mathbf{I} - \gamma \mathbf{P}_\pi)\mathbf{X}, \tag{3.10}$$

where $\mathbf{X}$ is the $|\mathcal{S}| \times d$ matrix with the $\mathbf{x}(s)$ as its rows, $\mathbf{D}_\pi$ is the $|\mathcal{S}| \times |\mathcal{S}|$ diagonal matrix with diagonal elements $\mathbf{d}_\pi$ which is the steady-state distribution under π, $\mathbf{P}_\pi$ is the $|\mathcal{S}| \times |\mathcal{S}|$ matrix of state-transition probabilities under π. Because the behavior policy is the same as the target policy under on-policy training, both $\mathbf{D}$ and $\mathbf{P}$ are under the same policy π, which makes on-policy TD(0) stable. While in off-policy TD(0), the $\mathbf{A}$ matrix becomes:

$$\mathbf{A} = \mathbf{X}^\top \mathbf{D}_\mu (\mathbf{I} - \gamma \mathbf{P}_\pi)\mathbf{X}, \tag{3.11}$$

where $\mathbf{P}_\pi$ is under the target policy π but $\mathbf{D}_\mu$ is under the behavior policy μ. Thus there is a mismatch, which makes off-policy TD(0) unstable. And emphatic methods were designed to solve this mismatch. Afterwards let us consider the most simple and common case of the Emphatic-TD methods where the interest of all states are 1. The Emphatic-TD fixed point is the same as the TD fixed point (3.9), but the $\mathbf{A}$ matrix is different. In off-policy Emphatic TD(0), the $\mathbf{A}$ matrix is:

$$\mathbf{A} = \mathbf{X}^\top \mathbf{F} (\mathbf{I} - \gamma \mathbf{P}_\pi)\mathbf{X}, \tag{3.12}$$

where $\mathbf{F}$ is a diagonal matrix with $f(s) \doteq d_\mu \lim_{t\to\infty} \mathbb{E}[F_t | S_t = s]$ on its diagonal. And the vector $\mathbf{f} \in \mathbb{R}^{|\mathcal{S}|}$ with components $[\mathbf{f}]_s \doteq f(s)$ is:

$$\mathbf{f} = \mathbf{d}_\mu + \gamma \mathbf{P}_\pi^\top \mathbf{d}_\mu + (\gamma \mathbf{P}_\pi^\top)^2 \mathbf{d}_\mu + \cdots \tag{3.13}$$

$$= (\mathbf{I} - \gamma \mathbf{P}_\pi^\top)^{-1} \mathbf{d}_\mu, \tag{3.14}$$

[2] The more detailed derivation and explanation of the TD fixed point (3.9), the $\mathbf{A}$ matrix of on-policy TD(0) (3.10), off-policy TD(0) (3.11), off-policy Emphatic TD(0) (3.12) can be found in Sutton et al., 2016.

where $(\mathbf{I} - \gamma \mathbf{P}_\pi^\top)^{-1}$ help to make the state distribution $\mathbf{d}_\mu$ under behavior policy μ match the target policy. And off-policy Emphatic TD(0) is stable. However, the behavior policy and the target policy are the same for on-policy cases, and the on-policy Emphatic TD(0)'s $\mathbf{A}$ matrix remains unchanged as Equation 3.12, but the vector $\mathbf{f}$ of on-policy Emphatic TD(0) will be:

$$\mathbf{f} = (\mathbf{I} - \gamma \mathbf{P}_\pi^\top)^{-1} \mathbf{d}_\pi, \tag{3.15}$$

where the state distribution $\mathbf{d}_\mu$ is already under π. Then $(\mathbf{I} - \gamma \mathbf{P}_\pi^\top)^{-1}$ is kind of redundant, and we think that is the cause of the sensitivity problem in on-policy emphatic methods.

Note that if the interest I, the decay-rate parameter λ, the discount-rate parameter γ and the importance sampling ratio ρ are all constant, the Emphatic-TD fixed point is actually the same as the TD fixed point. But if any of the four variables is not constant, the Emphatic-TD fixed point and the TD fixed point will be different.

3.3 The Step-size Heuristics in n-step Case

Let us first consider the typical n-step TD case when the interest equals to 1 for all states. Let integer $a = t$ **div** n and integer k be any integer smaller than a, then the emphasis equation (3.2) will be:

$$\begin{aligned}
M_t &= 0, \qquad t < 0, \\
M_t &= \gamma^n M_{t-n} + 1 \\
&= \gamma^n(\gamma^n M_{t-2n} + 1) + 1 \\
&= \gamma^{2n} M_{t-2n} + \gamma^n + 1 \\
&= \gamma^{2n}(\gamma^n M_{t-3n} + 1) + \gamma^n + 1 \\
&= \gamma^{3n} M_{t-3n} + \gamma^{2n} + \gamma^n + 1 \\
&\vdots \\
&= \gamma^{kn} M_{t-kn} + \gamma^{(k-1)n} + \cdots + \gamma^n + 1 \\
&\vdots \\
&= \gamma^{an} M_{t-an} + \gamma^{(a-1)n} + \cdots + \gamma^n + 1 \\
&= \gamma^{an}(\gamma^n M_{t-(a+1)n} + 1) + \gamma^{(a-1)n} + \cdots + \gamma^n + 1 \\
&= \gamma^{an} + \gamma^{(a-1)n} + \cdots + \gamma^n + 1 \\
&= (\gamma^n)^a + (\gamma^n)^{a-1} + \cdots + (\gamma^n)^1 + (\gamma^n)^0,
\end{aligned} \tag{3.16}$$

and it is a geometric series so it can be simplified as:

$$M_t = \frac{1 - (\gamma^n)^{a+1}}{1 - \gamma^n}. \tag{3.17}$$

When t is approaching infinity, a will also be approaching infinity, so the above equation can be more simpler:

$$M_{\text{inf}} = \frac{1}{1 - \gamma^n}, \tag{3.18}$$

$$M_t \leq M_{\text{inf}}. \tag{3.19}$$

Because we are studying the episodic emphatic methods in this thesis, the time step t will not reach infinity, but we still use Equation 3.18 instead of Equation 3.17 for several reasons. One characteristic of geometric series is that it increases fast at the beginning and more slowly as t approaches infinity, so the result of Equation 3.18 is usually very close to the result of Equation 3.17 except when the episode is very short which will not make the emphasis and update too large. Also, we are making a larger estimate than the true value as shown in Equation 3.19. When we adapt the step sizes to make the update smaller, a larger estimate will make the adapted step sizes even smaller which is good for preventing divergence. Note that both Equation 3.17 and Equation 3.18 hold when $-1 < \gamma < 1$, but the range for the discount rate γ in reinforcement learning is $0 \leq \gamma \leq 1$. Hence when $\gamma = 1$, we consider it as a special case and will be discussed later.

Because M_t is the only difference between the n-step TD update (2.11) and the n-step Emphatic-TD update (3.1), we can multiply the step size α by $\frac{1}{M}$ to limit the growth rate of the update to prevent divergence. Let us denote the step size of the n-step TD method as α_{TD} and the step size of the n-step Emphatic-TD method as α_{ETD}. We introduce a new variable *adapting rate* χ which is used to adapt the step-size parameter α as:

$$\begin{aligned} \alpha_{ETD} &= \chi \alpha_{TD}, \\ \chi &= \frac{1}{M_{\text{inf}}} \\ &= 1 - \gamma^n. \end{aligned} \tag{3.20}$$

3.4 The Step-size Heuristics in Eligibility Traces Case

The step-size heuristics work for not only the n-step Emphatic-TD method but also other emphatic methods such as the Emphatic-TD(λ) method. We still consider the case when the interest of all states equal to 1 first. The followon trace equation (3.8) in this case will be:

$$\begin{aligned} F_0 &= 1, \\ F_t &= \gamma F_{t-1} + 1 \\ &= \gamma(\gamma F_{t-2} + 1) + 1 \\ &= \gamma^2 F_{t-2} + \gamma + 1 \\ &\vdots \\ &= \gamma^t F_0 + \gamma^{t-1} + \cdots + \gamma + 1 \\ &= \gamma^t + \gamma^{t-1} + \cdots + \gamma + 1. \end{aligned} \tag{3.21}$$

This geometric series can be simplified as:

$$F_t = \frac{1-\gamma^{t+1}}{1-\gamma}, \tag{3.22}$$

and when $t = \infty$, it will be:

$$F_{\text{inf}} = \frac{1}{1-\gamma}, \tag{3.23}$$

$$F_t \le F_{\text{inf}}. \tag{3.24}$$

In addition to the reasons used in analyzing the n-step Emphatic-TD method, we have one more reason to use Equation 3.23 instead of Equation 3.22. Using the infinite version makes the afterward analysis simpler and easier, and it also makes the heuristic independent of the time step t. The heuristic then only depends on λ and γ, which makes it convenient for us to analyse the eligibility trace equations.

Let us substitute Equation 3.23 into the emphasis equations (3.7), the equations will become:

$$M_t = \lambda + (1-\lambda)F_t$$

$$M_{\text{inf}} = \lambda + (1-\lambda)F_{\text{inf}} \tag{3.25}$$

$$M_{\text{inf}} = \lambda + \frac{1-\lambda}{1-\gamma}, \tag{3.26}$$

$$M_t \le M_{\text{inf}}. \tag{3.27}$$

We will then analyse the equations of eligibility traces, and first let $C = \gamma\lambda$ for simplification. We will use M_{inf} instead of M_t because it is independent of t from Equation 3.26. Then Equation 3.6 will be:

$$\begin{aligned}
\mathbf{z}_0 &= M_{\text{inf}} \nabla\hat{v}(S_0, \mathbf{w}_0) \\
\mathbf{z}_1 &= C M_{\text{inf}} \nabla\hat{v}(S_0, \mathbf{w}_0) + M_{\text{inf}} \nabla\hat{v}(S_1, \mathbf{w}_1) \\
&= M_{\text{inf}}(C\nabla\hat{v}(S_0, \mathbf{w}_0) + \nabla\hat{v}(S_1, \mathbf{w}_1)) \\
\mathbf{z}_2 &= C M_{\text{inf}}(C\nabla\hat{v}(S_0, \mathbf{w}_0) + \nabla\hat{v}(S_1, \mathbf{w}_1)) + M_{\text{inf}} \nabla\hat{v}(S_2, \mathbf{w}_2) \\
&= M_{\text{inf}}(C^2\nabla\hat{v}(S_0, \mathbf{w}_0) + C\nabla\hat{v}(S_1, \mathbf{w}_1) + \nabla\hat{v}(S_2, \mathbf{w}_2)) \\
&\vdots \\
\mathbf{z}_t &= M_{\text{inf}}(C^t\nabla\hat{v}(S_0, \mathbf{w}_0) + C^{t-1}\nabla\hat{v}(S_1, \mathbf{w}_1) + \cdots + \nabla\hat{v}(S_t, \mathbf{w}_t)). \qquad (3.28)
\end{aligned}$$

The part in the parentheses of the result is just the same as what we can get from the equation of the eligibility trace of TD(λ) (2.13). Because the other parts of the update (Equation 2.14 and Equation 3.5, Equation 2.15 and Equation 3.4) are the same, we can multiply the step-size parameter of the emphatic method by $\frac{1}{M}$ to limit the growth rate of the update to prevent divergence. Then the adapting rate χ can be formulated as:

$$\begin{aligned}\alpha_{ETD} &= \chi\alpha_{TD},\\ \chi &= \frac{1}{M_{\text{inf}}}\\ &= \frac{1}{\lambda + \dfrac{1-\lambda}{1-\gamma}}.\end{aligned} \tag{3.29}$$

3.5 The Step-size Heuristics in Unequal-interest Case

Now we will consider the case where the interest of states are not all equal. Actually we can use the previous results because they can still guarantee that the update is not too large to diverge. For the n-step Emphatic-TD method:

$$\begin{aligned} i(s) &\le 1, \forall s \in \mathcal{S}\\ M \le M_t &\le M_{\text{inf}},\end{aligned} \tag{3.30}$$

and similarly for the Emphatic-TD(λ) method:

$$\begin{aligned} i(s) &\le 1, \forall s \in \mathcal{S}\\ F \le F_t &\le F_{\text{inf}}\\ M \le M_t &\le M_{\text{inf}}\\ \mathbf{z} \le \mathbf{z}_t &\le \mathbf{z}_{\text{inf}},\end{aligned} \tag{3.31}$$

where $F, M, \mathbf{z}$ denotes the case when not all interest are equal to 1. The true value is still smaller than the estimate in this case so it can prevent divergence. The only problem it may cause is that if the true value is too small, the learning will become much slower. This only happens when the interest of most states are close to 0, which is not very common. Notwithstanding, we can still solve this by redefining a more accurate adapting rate χ as:

$$\chi = \frac{|\mathcal{S}|(1-\gamma^n)}{\sum i(s)}, \forall s \in \mathcal{S}, \tag{3.32}$$

$$\chi = \frac{1}{\frac{\sum i(s)}{|\mathcal{S}|}(\lambda + \frac{1-\lambda}{1-\gamma})}, \forall s \in \mathcal{S}, \tag{3.33}$$

where $\sum i(s)$ is the sum of the interest of all states and $|\mathcal{S}|$ is the number of all the states. Equation 3.32 is for the n-step Emphatic-TD method and Equation 3.33 is for the Emphatic-TD(λ) method.

In most situations, we recommend using the general versions (3.20) (3.29) instead of the accurate versions (3.32) (3.33). We are trying to keep the characteristics of emphatic methods on the basis of preventing divergence, and the general version is enough for that goal. We first use the step-size heuristics to find an appropriate range of the step sizes, then we still need to use other classical parameter tuning methods to find the optimal results so the accuracy is not really important for this heuristic at the beginning.

3.6 The Step-size Heuristics in Other Special Cases

Then we will discuss the special case when it is an undiscounted task in which $\gamma = 1$. When $\gamma = 1$, the adapting rate χ in Equation 3.20 will be 0 which makes the algorithm does not learn, and Equation 3.29 does not work because $1-\gamma$ is a denominator of Equation 3.29 and denominators cannot be 0. Also, because the previous interest will not be discounted, the emphasis will continually increase as the time step t increases, which makes the probability of divergence really high. In this case, it is a better choice to use the sequence of step-size parameters instead of constant step-size parameters. Let us consider the n-step

Emphatic-TD method first, from Equation 3.16 we can get:

$$\begin{aligned} M_t &= (\gamma^n)^a + (\gamma^n)^{a-1} + \cdots + (\gamma^n)^1 + (\gamma^n)^0 \\ &= 1^a + 1^{a-1} + \cdots + 1^1 + 1^0 \\ &= a + 1 \\ &= t \ \mathbf{div}\ n + 1 \\ &\approx \frac{t}{n}. \end{aligned} \tag{3.34}$$

The emphasis equation now depends on t which makes the adapting rate χ not constant:

$$\begin{aligned} \chi &= \frac{1}{M_t} \\ &= \frac{n}{t}. \end{aligned} \tag{3.35}$$

Similarly, the followon trace equation of the Emphatic-TD(λ) method (3.21) will become:

$$\begin{aligned} F_t &= \gamma^t + \gamma^{t-1} + \cdots + \gamma + 1 \\ &= t + 1. \end{aligned} \tag{3.36}$$

Then the emphasis equation will be:

$$\begin{aligned} M_t &= \lambda + (1-\lambda)F_t \\ &= \lambda + (1-\lambda)(t+1) \\ &= t - \lambda t + 1. \end{aligned} \tag{3.37}$$

Because the eligibility trace equations (3.28) are not affected, the adapting rate χ is:

$$\begin{aligned} \chi &= \frac{1}{M_t} \\ &= \frac{1}{t - \lambda t + 1}. \end{aligned} \tag{3.38}$$

Note that because the one-step Emphatic-TD method and the Emphatic-TD(0) method are identical, their corresponding adapting rates are the same as well:

$$\chi = 1 - \gamma^n = 1 - \gamma \qquad \text{(from Equation 3.20)}$$

$$\chi = \frac{1}{\lambda + \dfrac{1-\lambda}{1-\gamma}} = 1 - \gamma. \qquad \text{(from Equation 3.29)}$$

For the Emphatic-TD(λ) method, when $\lambda = 1$, the adapting rate χ will be:

$$\chi = \frac{1}{\lambda + \dfrac{1-\lambda}{1-\gamma}} = 1, \tag{3.39}$$

which means that the step-size parameter does not need to be changed. This is because when there is no bootstrapping and all the interest are equal, the Emphatic-TD(λ) method is exactly the same as the TD(λ) method.

We have introduced and discussed our step-size heuristics in different cases theoretically in this chapter. In the next chapter, we will design some experiments to verify our step-size heuristics.

Chapter 4

Experiments for Step-size Heuristics

In this chapter, we conduct some empirical studies on the new step-size heuristics we introduced in the previous chapter. This chapter and the previous chapter together compose the first area of contribution of this thesis. We start this chapter by introducing the testbed we used throughout this whole chapter, then we show the empirical results of the sensitivity problem of the step-size parameter in emphatic methods. After that, we verify the step-size heuristics empirically in different cases.

4.1 The Random Walk Example

For both the empirical studies of the sensitivity problem and the step-size heuristics, we used a variation of the random walk example from Sutton (1988) as the testbed. Let us consider a *Markov reward process* MRP with 100 states. An MRP is a Markov decision process that we introduced in Section 2.1 without actions, and we use MRP here because the discussions of the previous chapter are all based on the prediction problems. In this MRP, the 100 states are numbered from 1 to 100 in order from left to right. There are two terminal states: state 0 on the extreme left and state 101 on the extreme right. If an episode terminates on the left state 0, a reward of -1 will occur. If an episode terminates on the right state 101, a reward of +1 will occur. And all other rewards are 0. All episodes start in the state 50 which is close to the center, then proceed to one of the 10 neighbouring states to the left, or to one of the 10 neighbouring states to the right, with equal probability. If the current state is close to the terminal state, it is possible that that side has less than 10 neighbouring states. Then the probability of transiting to the missing states will be added to the probability of transiting to that terminal state in this case. For example, there is a probability of 0.5 for state 1 and state 100 to transit to the corresponding terminal states, and there is a probability of 0.25 for state 6 and state 95 to transit to the corresponding terminal states. We used this version of random walk example throughout this whole chapter.

We used the tile coding introduced in Section 2.7 to construct the function approximation feature vectors. More specifically, we had 5 tilings with 6×1 tiles for each tiling. Note for the above version random walk example, the state is only represented by a number from 1 to 100, which is the one-dimensional state space. That is the reason why the number of tiles for each tiling is in $n \times 1$ form. The weights and eligibility traces (just for the TD(λ) method) are all initialized to 0. Because we are comparing different methods or different parameter settings for most experiments, we sample state transitions with the corresponding rewards just once and apply different methods on that to eliminate the bias.

4.2 Experiments of the Sensitivity Problem in Emphatic Methods

First we would like to show why we want such heuristics for adapting step sizes to emphatic methods. As we have discussed before, some existing studies gave hints of the existence of the sensitivity problem, and we also encountered this problem during our empirical studies in emphatic methods. Moreover, we found that the main problem of the sensitivity of the step-size parameters is that divergence would arise if inappropriate step-size parameters are used.

To study the sensitivity of the step-size parameters, we did the parameter studies over the step-size parameter α between TD(0) and Emphatic-TD(0) on our version of the random walk example. We also compared the two methods

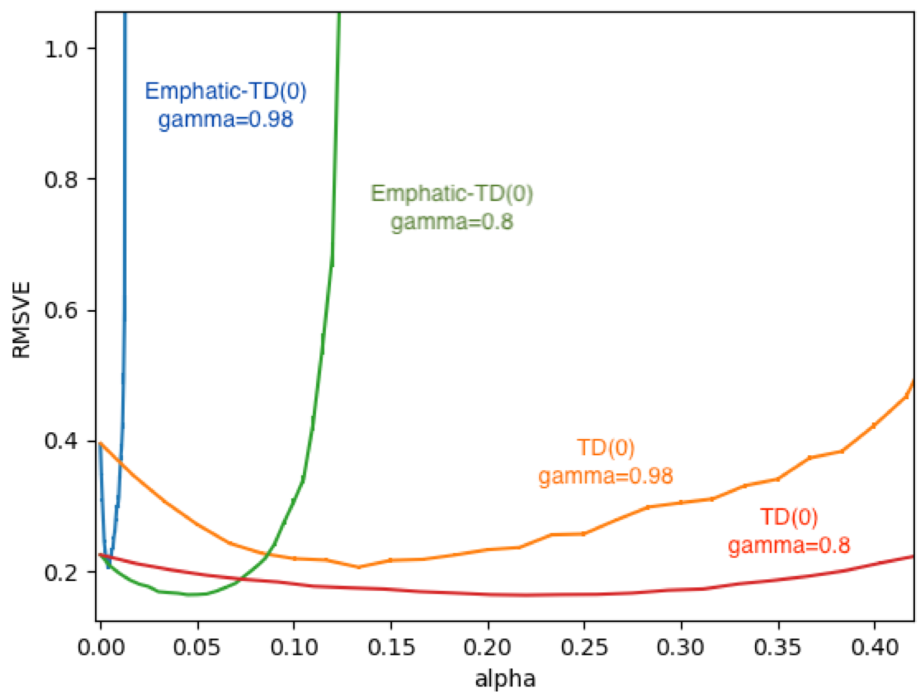


Figure 4.1: Parameter studies of the performance over the step-size parameter α between TD(0) and Emphatic-TD(0) for different discount-rate parameters γ. The bars show the ranges of the standard errors but are not obvious because of their small sizes.

in two situations where $\gamma = 0.98$ and $\gamma = 0.8$. We ran the experiments for 10 episodes for each run and had 100 such runs. The result of the experiment is shown in Figure 4.1.

The y-axis of Figure 4.1 is the averaged RMSVE over the episodes and runs. In the figure, it is obvious that the averaged RMSVE of Emphatic-TD(0) grew too rapidly which made the lines almost vertical near $\alpha = 0.015$ when $\gamma = 0.98$ and near $\alpha = 0.125$ when $\gamma = 0.8$. We considered that divergence arose at those two points so we conducted another experiment to verify that. We ran the experiment using Emphatic-TD(0) with $\alpha = 0.015$ and $\gamma = 0.98$ for 100 episodes and got Figure 4.2 after 100 such runs. The learning curve of Emphatic-TD(0) is in an upward trend so divergence indeed arose when $\alpha = 0.015$ and $\gamma = 0.98$.

Let us refocus on Figure 4.1, we could find that TD(0) had different optimal

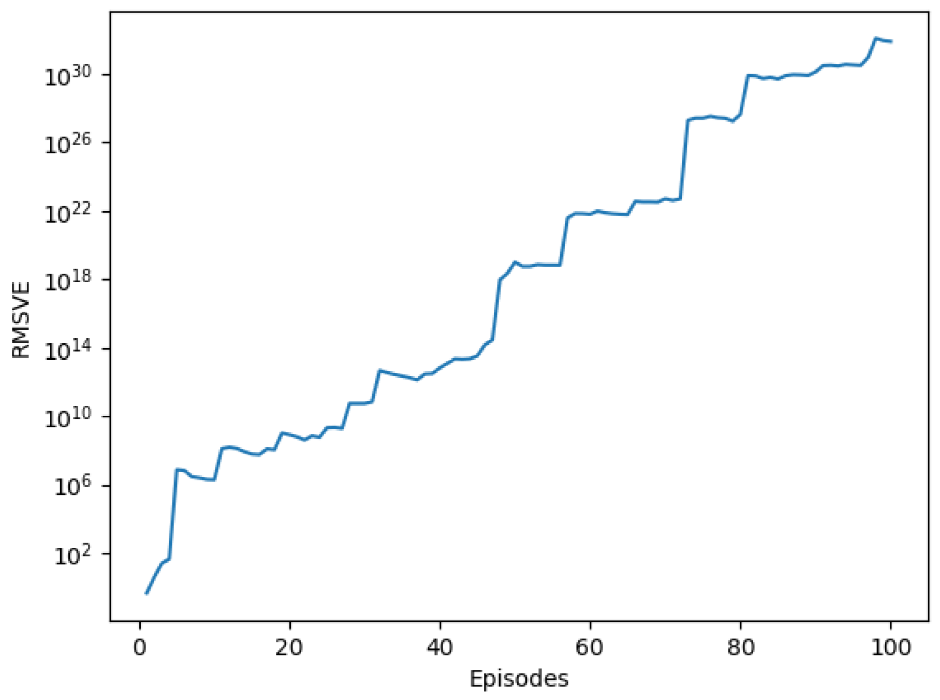


Figure 4.2: Learning curve of Emphatic-TD(0) using the inappropriate step-size parameter on our version of random walk example. Note that the y-axis is on a log scale.

points for different γ, but it had relatively low averaged RMSVE over a wide range of the step-size parameter α for both γ. This makes it easy for TD(0) to find the optimal step-size parameter because even bad step-size parameters will not lead to divergence. However, Emphatic-TD(0) only had relatively low averaged RMSVE over a narrow range of the step-size parameter α. Even worse, the ranges of relatively low averaged RMSVE are different for different γ which makes it harder to find the optimal step-size parameter in emphatic methods. For example, the optimal step-size parameter of Emphatic-TD(0) was near 0.05 when $\gamma = 0.8$. However, in the setting of $\gamma = 0.98$, the step-size parameter $\alpha = 0.05$ would have a really large averaged RMSVE and actually divergence would arise at that point. Therefore, we would like to study the relationship between the emphatic methods and the corresponding non-emphatic methods for some heuristics.

4.3 The Equal-interest Setting

In this section, we present some experiments to support our results in the previous chapter. In particular, we show that the Emphatic-TD methods using our adapting step-size heuristics have very similar results as the corresponding non-emphatic methods if the interest of all states are equal.

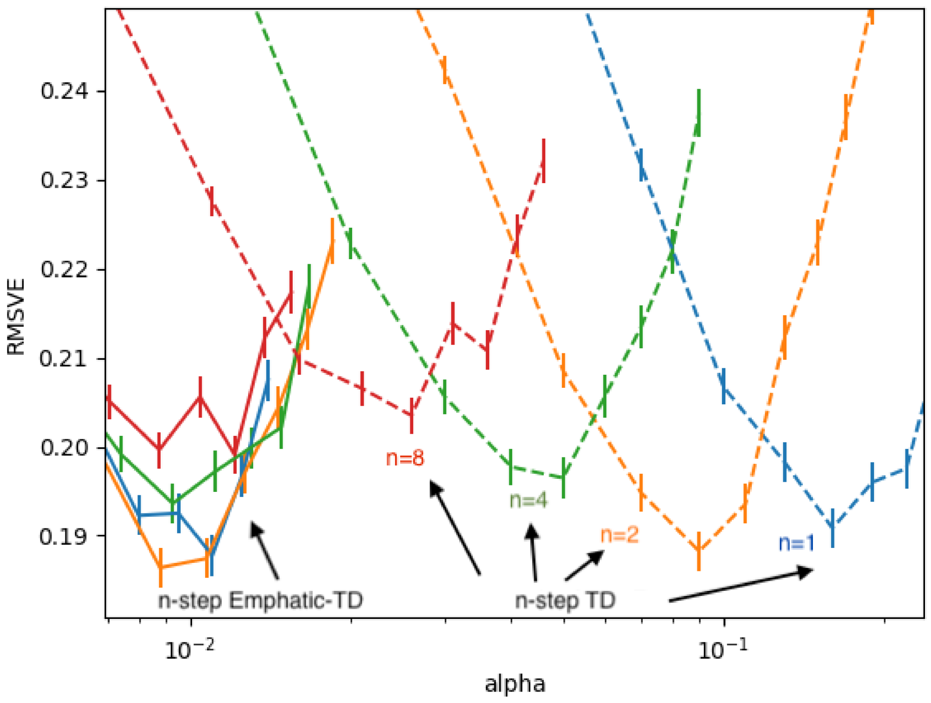


Figure 4.3: Parameter studies of the performance over the step-size parameter α between n-step TD and n-step Emphatic-TD for different n. The solid line representing n-step Emphatic-TD has the same n as the corresponding dashed line with the same color. The bars show the ranges of the standard errors and the x-axis is on a log scale.

Let us first discuss the step-size heuristics for the n-step Emphatic-TD method. We applied the n-step TD method and the n-step Emphatic-TD method using different n and different step-size parameters α on our version of the random walk task. It is a discounted task ($\gamma = 0.95$) so we could have a rough idea of whether Equation 3.20 holds. We ran the experiment for 10 episodes for each run and got Figure 4.3 after 100 such runs.

The y-axis of the figure is the averaged RMSVE over the episodes and runs. For the same n, the n-step TD method and the n-step Emphatic-TD method both had an optimal range of α which have similar RMSVE, and the two ranges accorded with the step-size heuristics. For example, when $n = 2$, the n-step TD method had the lowest RMSVE near $\alpha = 0.09$. By using the step-size heuristics (3.20), we could compute the corresponding proper step-size parameter to be 0.008775 for the n-step Emphatic-TD method when $n = 2$, and we could find that the local minimum for the n-step Emphatic-TD method was near $\alpha = 0.009$ indeed. When we know the suitable range of the step-size parameter for the n-step TD method, we could use the step-size heuristics to find the corresponding range of the step-size parameter for the n-step Emphatic-TD method.

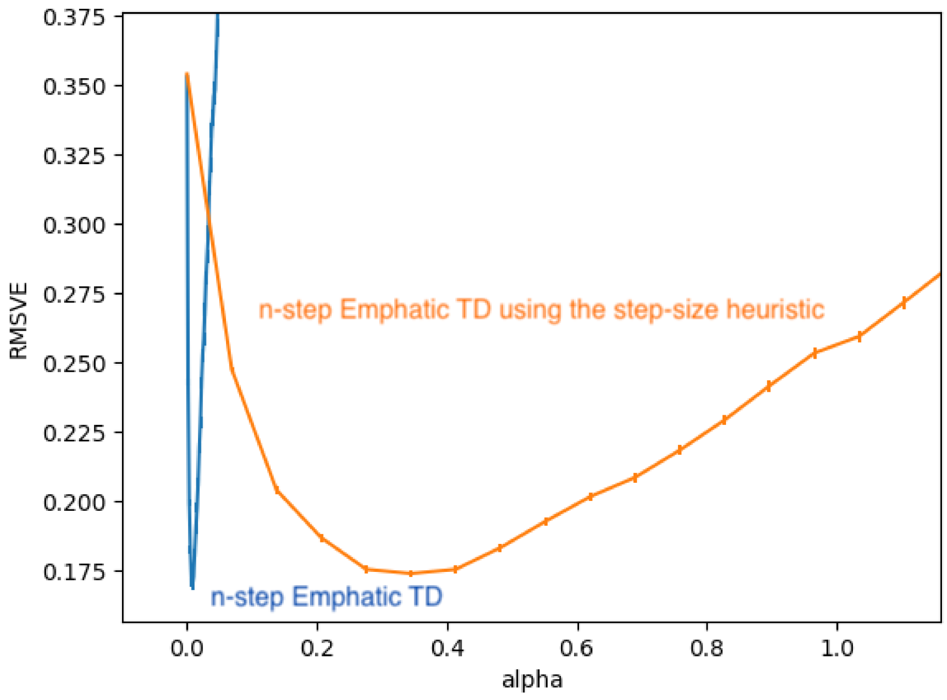
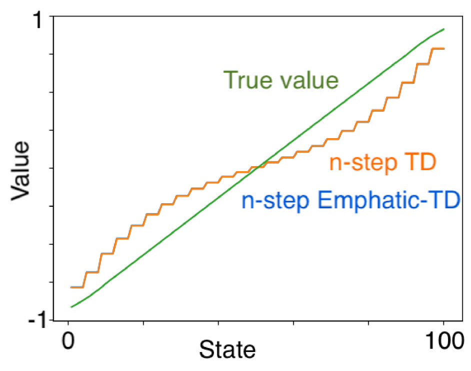
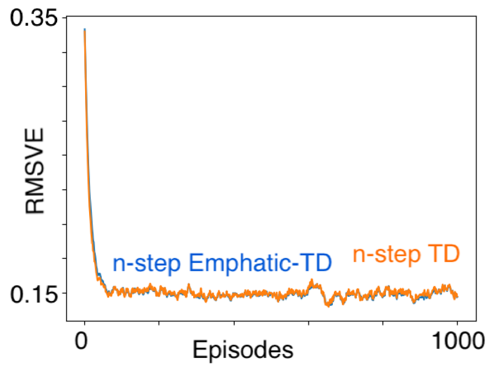


Figure 4.4: Parameter studies of the performance over the step-size parameter α for n-step Emphatic TD with and without the step-size heuristic. The bars show the ranges of the standard errors but are not obvious because of their small sizes.

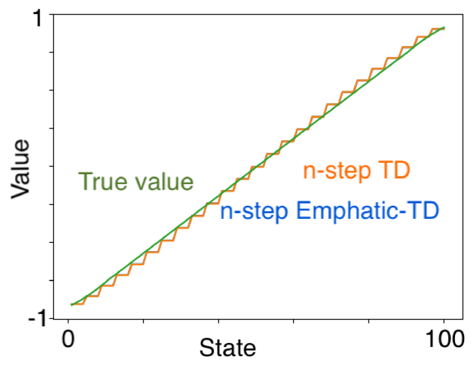
Then we ran a parameter study experiment to show the effect of the step-size heuristics for the n-step Emphatic-TD method when $n = 3$ and $\gamma = 0.95$. We ran the experiments for 20 episodes for each run and had 100 such runs. The result is shown in Figure 4.4. The y-axis of the figure is the averaged RMSVE over the episodes and runs. We can see that the local minimum of the n-step Emphatic-TD method did not change after using the step-size heuristic, they were near the level where RMSVE is 0.175 for both cases. But by using the step-size heuristic, the n-step Emphatic-TD method had a wider range of low RMSVE comparing to the original method without the step-size heuristic, which means the step-size heuristic can indeed help the n-step Emphatic-TD method to be less sensitive to the step-size parameter α.



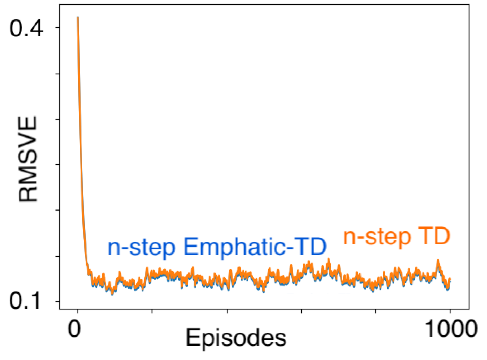
(a) State value of discounted task



(b) Learning curve of discounted task



(c) State value of undiscounted task



(d) Learning curve of undiscounted task

Figure 4.5: Learning curves between n-step TD and n-step Emphatic-TD using step-size heuristics and the corresponding approximate state values in equal-interest setting. The lines of n-step TD and n-step Emphatic-TD using step-size heuristics are almost overlapped.

We have also run some experiments between the n-step TD method and the n-step Emphatic-TD method using step-size heuristics when $n = 3$. The experiments all had 1000 episodes as one run and had 100 such runs, and the results are shown in Figure 4.5. The right two sub-figures demonstrate the complete learning curve of the averaged RMSVE, and the left two sub-figures demonstrate the learned state value after 1000 episodes. And both the RMSVE and the learned state value are averaged over 100 runs. The top two sub-figures are under the discounted task where $\gamma = 0.95$. We used the n-step TD method with the step-size parameter $\alpha = 0.01$ and the n-step Emphatic-TD method with $\alpha \approx 0.0014$ based on Equation 3.20. The bottom two sub-figures are under the undiscounted task where $\gamma = 1$. The step-size parameter α of the n-step TD method was also 0.01, while the n-step Emphatic-TD method used the sequences of step-size parameters by Equation 3.35.

For both the state-value sub-figures and the learning curve sub-figures, we could find that the lines for the n-step TD method and the n-step Emphatic-TD method were overlapped, which means they were almost the same after using the step-size heuristics.

Next we would run the experiments to see the effect of the step-size heuristics on the Emphatic-TD(λ) method. And we started with comparing the TD(λ) method with the Emphatic-TD(λ) method using different decay-rate parameters for eligibility traces over different step-size parameters α on our version of the random walk task. Again it is a discounted task ($\gamma = 0.95$) so we could have a rough idea if Equation 3.29 holds. We only ran the experiments for 10 episodes but repeated the process for 100 times, then we would have Figure 4.6 as the result.

The y-axis of the figure is the average RMSVE over the episodes and runs. Note that there is only one purple line in the figure representing Emphatic-TD(1) and TD(1), this is because the TD(λ) method and the Emphatic-TD(λ) method are exactly the same when $\lambda = 1$ and the interest of all states are equal. For the same decay-rate parameter λ, the TD(λ) method and the Emphatic-TD(λ) method both had an optimal range of α which had similar RMSVE, and the relation between these two ranges followed the step-size heuristics. For example, when the decay-rate parameter for eligibility trace λ was 0.4,

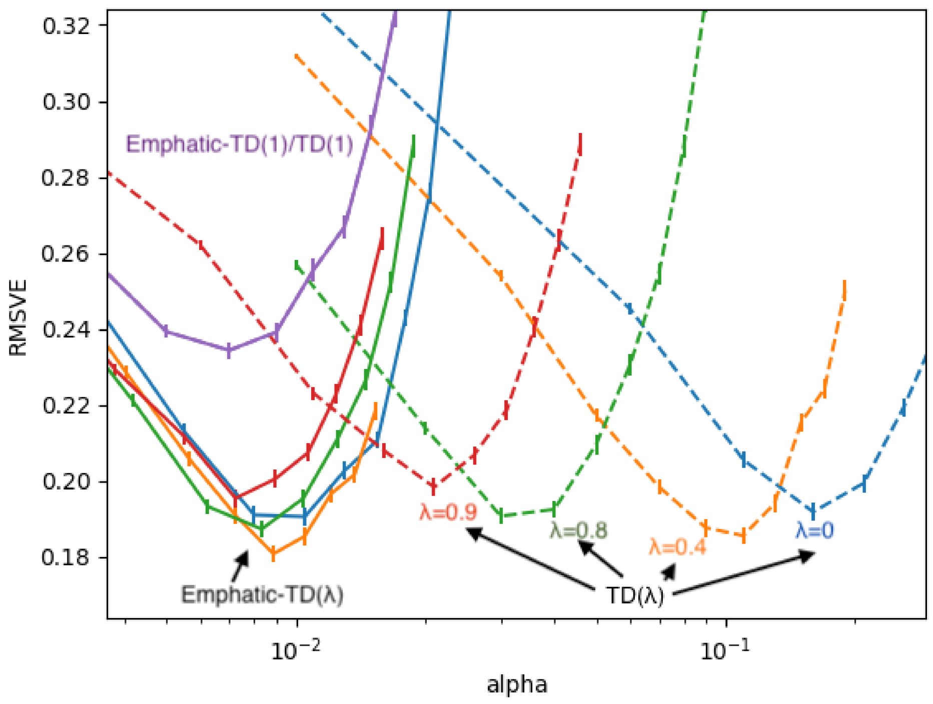


Figure 4.6: Parameter studies of the performance over the step-size parameter α between TD(λ) and Emphatic-TD(λ) for different λ. The solid line representing Emphatic-TD(λ) has the same λ as the corresponding dashed line with the same color. The bars show the ranges of the standard errors and the x-axis is on a log scale.

the TD(λ) method had the lowest RMSVE near $\alpha = 0.11$. By using the step-size heuristics (3.29), we could calculate that the corresponding step-size parameter for the Emphatic-TD(λ) method should be around 0.0089, and we could find that the smallest RMSVE for the Emphatic-TD(λ) method was near the point $\alpha = 0.009$ indeed. We could also use the step-size heuristics to find an appropriate range of the step-size parameter for the Emphatic-TD(λ) method if we know the suitable range of the step-size parameter for the TD(λ) method.

Similarly, we ran another parameter study experiment to show the effect of the step-size heuristics for the Emphatic-TD(λ) method when $\lambda = 0.3$ and $\gamma = 0.95$. We ran the experiments for 20 episodes for each run and had 100 such runs. The result is shown in Figure 4.4. The y-axis of the figure is

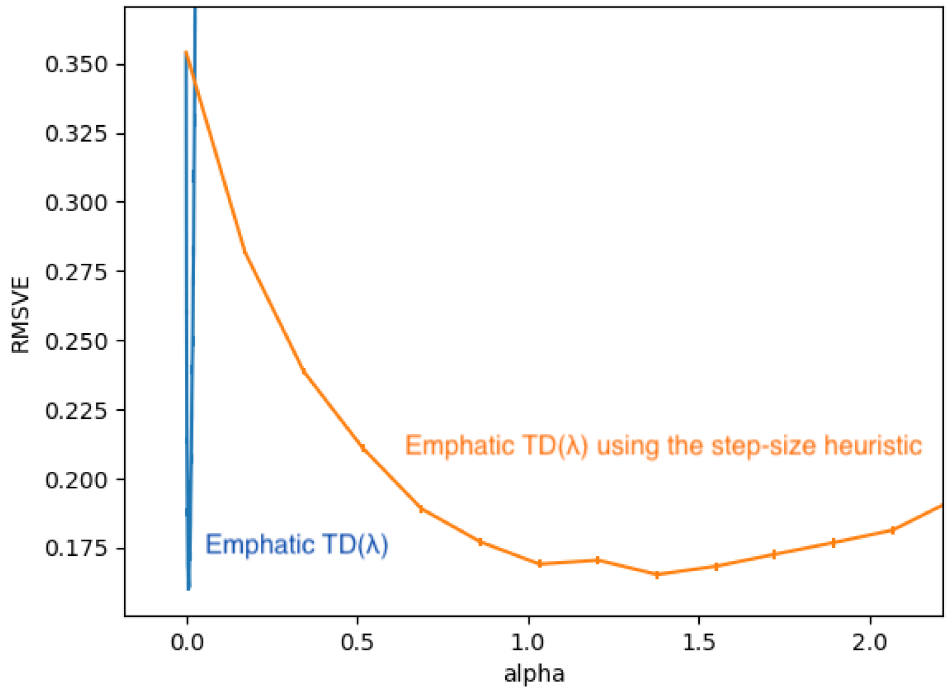
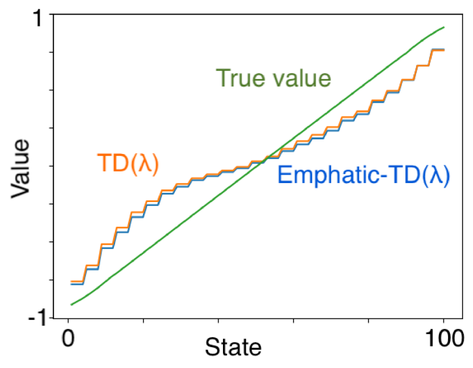


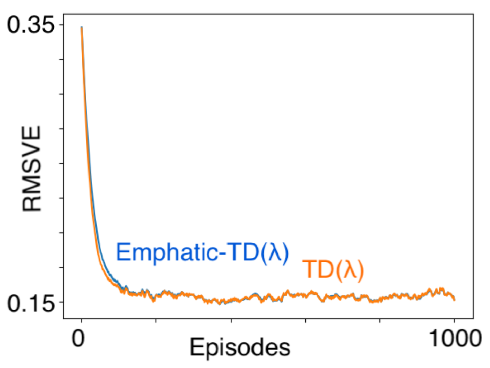
Figure 4.7: Parameter studies of the performance over the step-size parameter α for Emphatic-TD(λ) with and without the step-size heuristic. The bars show the ranges of the standard errors but are not obvious because of their small sizes.

the averaged RMSVE over the episodes and runs. The result is also similar to the n-step case. The local minimum of the n-step Emphatic-TD method did not change after using the step-size heuristic, they were near the level where RMSVE is 0.15 for both cases. And by using the step-size heuristic, the Emphatic-TD(λ) method also had a wider range of low RMSVE comparing to the original method without the step-size heuristic, which means the step-size heuristic can also help the Emphatic-TD(λ) method to be less sensitive to the step-size parameter α.

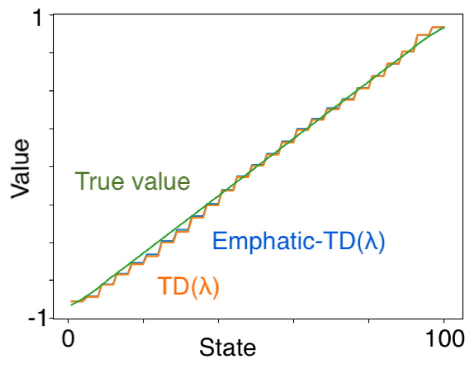
Then we would run more specific experiments between the TD(λ) method and the Emphatic-TD(λ) method using the step-size heuristics, and take a look at the result Figure 4.8. The decay-rate parameter for eligibility traces λ was 0.3, and the experiments were ran for 100 runs and each run had 1000 episodes. The two sub-figures on the right-hand side are of the learning curve



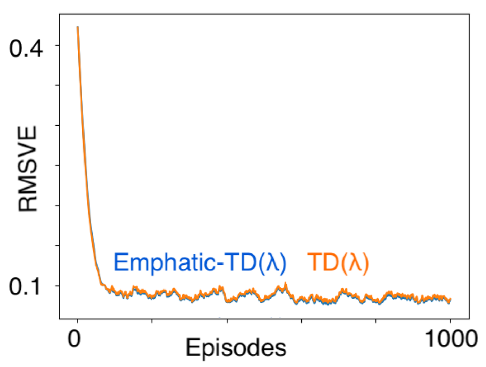
(a) State value of discounted task



(b) Learning curve of discounted task



(c) State value of undiscounted task



(d) Learning curve of undiscounted task

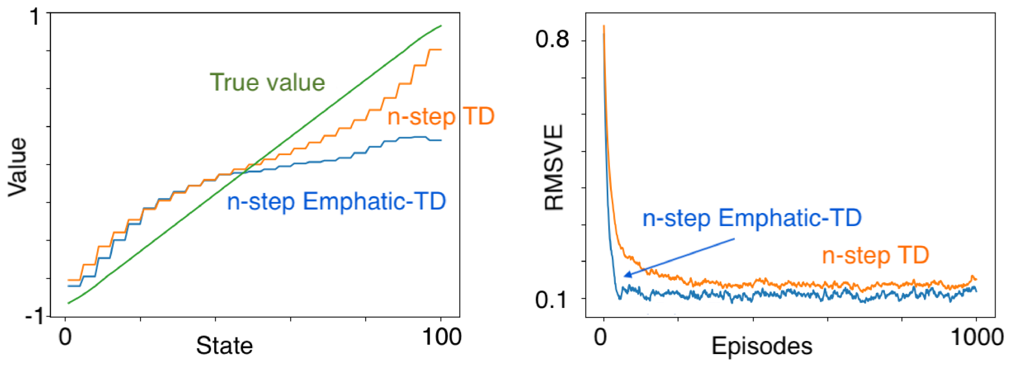
Figure 4.8: Learning curves between TD(λ) and Emphatic-TD(λ) using step-size heuristics and the corresponding approximate state values in equal-interest setting. The lines of TD(λ) and Emphatic-TD(λ) using step-size heuristics are almost overlapped.

of the averaged RMSVE and sub-figures on the left-hand side are of the true state values versus the approximate state values after 1000 episodes. The top two sub-figures are under the discounted task where $\gamma = 0.95$. We used the TD(λ) method with step-size parameter $\alpha = 0.01$ and the Emphatic-TD(λ) method with $\alpha \approx 0.0007$ based on Equation 3.29. The bottom two sub-figures are under the undiscounted task where $\gamma = 1$. The step-size parameter α of the TD(λ) method was still 0.01, while the Emphatic-TD(λ) method used the sequences of step-size parameters by Equation 3.38.

In all these four sub-figures, we could find that the lines for the TD(λ) method and the lines for the Emphatic-TD(λ) method are almost overlapped. Thus the step-size heuristics could also make the Emphatic-TD(λ) method to have almost the same performance as the TD(λ) method by adapting the step-size parameter when the interest of all states are equal.

4.4 The Unequal-interest Setting

In the previous section, we showed that the step-size heuristics would make the Emphatic-TD methods almost the same as the corresponding non-emphatic methods when the interest of all states are equal. Although it is a popular setting where the interest of all states equal to 1, emphatic methods will highlight the advantage when specific interest of states are set. In this section, we would compare the Emphatic-TD methods using step-size heuristics and the corresponding non-emphatic methods under our version of the random walk task with unequal interest of states. In particular, our version of the random walk example had 100 states and assumed that we are only interested in the 5 leftmost states. In that case the interest of state 1 to 5 equal to 1, and the interest of the rest states all equal to 0. And we would run the experiments for 1000 episodes and repeat this for 100 runs under the discounted task where $\gamma = 0.95$.

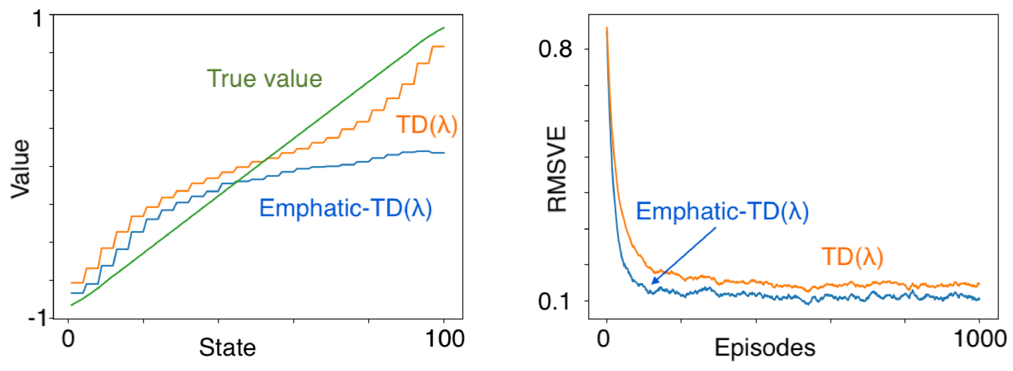


(a) State value (b) Learning curve

Figure 4.9: Learning curves between n-step TD and n-step Emphatic-TD using step-size heuristics and the corresponding approximate state values in equal-interest setting. Only the approximate values of the leftmost 5 states are important in Figure 4.9a.

Let us first compare the n-step TD method with the n-step Emphatic-TD method using the step-size heuristics. When $n = 3$, we could plot the results in Figure 4.9. We still plotted the complete learning curve of the averaged RMSVE over 100 runs and compared the state value of the two methods with the true state value as we have done in the equal-interest cases.

From the learning curves in Figure 4.9b, we could find that the averaged RMSVE of the n-step Emphatic-TD method using the step-size heuristics dropped much faster at the beginning and kept at a lower level than the averaged RMSVE of the n-step TD method afterwards. Also, from Figure 4.9a we could see that the approximate values of the first 5 states of the n-step Emphatic-TD method using the step-size heuristics were closer to the true state values than those of the n-step TD method. Note that the right-hand part (around state 50 to state 100) of the approximate values of the n-step Emphatic-TD method using the step-size heuristics were farther from the true state values than those of the n-step TD method, but because we only care about the first 5 states, it does not matter if the method leads to poor results in other states.



(a) State value (b) Learning curve

Figure 4.10: Learning curves between TD(λ) and Emphatic-TD(λ) using step-size heuristics and the corresponding approximate state values in equal-interest setting. Only the approximate values of the leftmost 5 states are important in Figure 4.10a.

The results of the comparison between the TD(λ) method and the Emphatic-TD(λ) method using the step-size heuristics were quite similar. When the decay-rate parameter for eligibility traces λ was 0.3, we ran the experiments and got Figure 4.10.

In Figure 4.10b, the averaged RMSVE of the Emphatic-TD(λ) method using the step-size heuristics also decreased more rapidly at the starting episodes than the averaged RMSVE of the TD(λ) method and kept at a lower level afterwards. Comparing to the n-step experiments in Figure 4.9b, the averaged RMSVE of the Emphatic-TD(λ) method using the step-size heuristics did not drop that fast at the beginning, but there was a larger difference between the emphatic methods and the corresponding non-emphatic methods after the starting episodes in Figure 4.10b. Likewise, in Figure 4.10a, the approximate values of the first 5 states of the Emphatic-TD(λ) method using the step-size heuristics were also closer to the true state values than those of the TD(λ) method in Figure 4.9a.

In this chapter, we demonstrated the effect of our step-size heuristics empirically. We first conducted experiments to show the sensitivity problem in emphatic methods and the significance of our step-size heuristics. Then our experiments showed that both the n-step Emphatic-TD method and the Emphatic-TD(λ) method using the step-size heuristics had almost the same performance as the corresponding non-emphatic methods when the interest of all states are equal. When the interest of states are not all equal, both the n-step Emphatic-TD method and the Emphatic-TD(λ) method using the step-size heuristics had better performance than the corresponding non-emphatic methods which means our step-size heuristics can retain the advantages of the emphatic methods. The empirical results are in agreement with our theoretical results in the previous chapter.

Chapter 5

The On-policy Control Emphatic Methods

This chapter concerns the second area of contribution of this thesis, and we extend the idea of emphatic methods to the on-policy control methods Sarsa. We first extended the idea of emphatic methods to Sarsa and introduced the n-step Emphatic-Sarsa method and the Emphatic-Sarsa(λ) method. Then we conducted some empirical studies to compare the performance of the two new on-policy emphatic control methods with the corresponding non-emphatic methods in different cases.

5.1 The n-step Emphatic-Sarsa Method

We would like to extend the idea of emphatic methods to the on-policy action-value methods, and Sarsa should be the most fundamental and widely used on-policy TD action-value method which is a good choice to see if emphasis works in on-policy control cases. Thus we would develop the emphatic version of the n-step Sarsa method and the Sarsa(λ) method. Let us first consider the n-step version which is named the *n-step Emphatic-Sarsa* method. We have introduced the n-step Sarsa method in Section 2.6. In the Emphatic-TD prediction methods, the interest at time t is the measurement of how much we care about accurately valuing the state. Proceeding to control methods, the actions are another concern for us because action values are learned. Thus for the Emphatic-Sarsa methods, the non-negative scalar variable interest I_t

Algorithm 1: Episodic semi-gradient n-step Emphatic-Sarsa for estimating $\hat{q} \approx q_*$ (or q_π)

```
1  Input: a differential action-value function parameterization
     $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
2  Input: a function parameterization $i : \mathcal{S} \times \mathcal{A} \to [0, 1]$ returning the
     interest for each state-action pair
3  (Input: a policy $\pi$ if estimating $q_\pi$)
4  Algorithm parameters: step-size parameter $\alpha > 0$, probability of
     taking a random action parameter $\varepsilon > 0$, an integer $n > 0$
5  Initialize the weight vector $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = 0$)
6  Loop for each episode:
7    Initialize and store the starting state $S_0$
8    Use $\varepsilon$-greedy with respect to $\hat{q}(S_0, \cdot, \mathbf{w})$ to choose an action $A_0$ (or
     use the policy $\pi$ to choose an action $A_0 \sim \pi(\cdot|S_0)$) to store
9    $T \leftarrow \infty$
10   $M_0 \leftarrow i(S_0, A_0)$
11   Loop for $t = 0, 1, 2, ...$:
12     If $t < T$:
13       Take action $A_t$
14       Observe the next reward $R_{t+1}$ and the next state $S_{t+1}$ and
     store them
15       If $S_{t+1}$ is the terminal state:
16         $T \leftarrow t + 1$
17       else:
18         Use $\varepsilon$-greedy with respect to $\hat{q}(S_{t+1}, \cdot, \mathbf{w})$ to choose an action
     $A_{t+1}$ (or use the policy $\pi$ to choose an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$) to
     store
19         If $t + 1 < n$:
20           $M_{t+1} = i(S_{t+1}, A_{t+1})$
21         else:
22           $M_{t+1} = i(S_{t+1}, A_{t+1}) + \gamma^n M_{t-n+1}$
23     $\tau \leftarrow t - n + 1$
24     If $\tau \geq 0$:
25       $G \leftarrow \sum_{i=\tau+1}^{min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
26       If $\tau + n < T$:
27         $G \leftarrow G + \gamma^n \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w})$
28       $\mathbf{w} \leftarrow \mathbf{w} + \alpha M_\tau [G - \hat{q}(S_\tau, A_\tau, \mathbf{w})] \nabla \hat{q}(S_\tau, A_\tau, \mathbf{w})$
29   Until $\tau = T - 1$
```

indicates the degree of our interest in accurately valuing the state-action pair at time t. The other non-negative scalar variable emphasis M_t which can emphasize or de-emphasize the learning by multiplying the update at time t

is still defined as:

$$M_t = I_t + \gamma^n M_{t-n}, \qquad 0 \le t < T,$$
$$M_t = 0, \qquad t < 0. \tag{5.1}$$

Then we add the emphasis to the n-step Sarsa update (2.21) to define the new update as:

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha M_t [G_{t:t+n} - \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1})] \nabla \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1}), \quad 0 \le t < T. \tag{5.2}$$

And the n-step return is also redefined as:

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1}), \quad t+n < T,$$
$$G_{t:t+n} \doteq G_t, \qquad t+n \ge T. \tag{5.3}$$

The complete pseudocode of the n-step Emphatic-Sarsa method is given in the box 1.

5.2 The Emphatic-Sarsa(λ) Method

Then let us take a look at the Emphatic-Sarsa method with eligibility traces, which is named the *Emphatic-Sarsa(λ)* method by us. The interest here is the same as that of the n-step Emphatic-Sarsa method which depends on both the state and the action. And there is an additional non-negative scalar variable which is the followon trace F_t:

$$F_0 \doteq i(S_0),$$
$$F_t \doteq \gamma F_{t-1} + I_t, \tag{5.4}$$

the emphasis M_t now depends on both the interest and the followon trace:

$$M_t \doteq \lambda I_t + (1-\lambda) F_t. \tag{5.5}$$

The eligibility trace of the Emphatic-Sarsa(λ) method can be modified from the eligibility trace of the Sarsa(λ) method (2.22). We can multiply the gradient part of the eligibility trace by the emphatic parameter, then the eligibility trace is defined as:

$$\mathbf{z}_{-1} \doteq 0,$$
$$\mathbf{z}_t \doteq \gamma\lambda \mathbf{z}_{t-1} + M_t \nabla \hat{q}(S_t, A_t, \mathbf{w}_t), \quad 0 \le t \le T, \tag{5.6}$$

Algorithm 2: Episodic semi-gradient Emphatic-Sarsa(λ) for estimating $\hat{q} \approx q_*$ (or q_π)

```
1  Input: a differential action-value function parameterization
     $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
2  Input: a function parameterization $i : \mathcal{S} \times \mathcal{A} \to [0, 1]$ returning the
     interest for each state-action pair
3  (Input: a policy $\pi$ if estimating $q_\pi$)
4  Algorithm parameters: step-size parameter $\alpha > 0$, probability of
     taking a random action parameter $\varepsilon > 0$, decay-rate parameter
     $\lambda \in [0, 1]$
5  Initialize the weight vector $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = 0$)
6  Loop for each episode:
7      Initialize the state $S$
8      Use $\varepsilon$-greedy with respect to $\hat{q}(S, \cdot, \mathbf{w})$ to choose an action $A$ (or use
     the policy $\pi$ to choose an action $A \sim \pi(\cdot|S)$)
9      $F \leftarrow 0$
10     $M \leftarrow 0$
11     $\mathbf{z} \leftarrow 0$
12     Loop for each time step in the episode:
13         Take action $A$, observe the next reward $R$ and the next state $S'$
14         $F \leftarrow i(S, A) + \gamma F$
15         $M \leftarrow \lambda i(S, A) + (1 - \lambda)F$
16         $\mathbf{z} \leftarrow \gamma\lambda\mathbf{z} + M\nabla\hat{q}(S, A, \mathbf{w})$
17         If $S'$ is the terminal state:
18             $\delta \leftarrow R - \hat{q}(S, A, \mathbf{w})$
19             $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\mathbf{z}$
20             Go to the next episode
21         Use $\varepsilon$-greedy with respect to $\hat{q}(S', \cdot, \mathbf{w})$ to choose an action $A'$ (or
     use the policy $\pi$ to choose an action $A' \sim \pi(\cdot|S')$)
22         $\delta \leftarrow R + \gamma\hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})$
23         $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\mathbf{z}$
24         $S \leftarrow S'$
25         $A \leftarrow A'$
```

and the TD error is now defined as:

$$\delta_t \doteq R_{t+1} + \gamma\hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t), \tag{5.7}$$

the update is just the product of the step-size parameter, the above eligibility trace vector and the TD error:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\delta_t\mathbf{z}_t. \tag{5.8}$$

The complete pseudocode of the Emphatic-Sarsa(λ) method is given in the box 2.

Note that the above equation of the eligibility trace (5.6) is based on the eligibility trace of Sarsa(λ) (2.22), and the trace of (2.22) is called the *accumulating trace*. Accumulating traces are the most commonly used traces because they are available in almost all situations. There is another kind of traces called the *replacing trace* (Singh & Sutton, 1996), which usually has a better performance than the accumulating trace. But replacing traces have some limitations, they can only be used for tabular cases or binary feature vectors. The replacing trace is defined based on whether each component of the binary feature vectors is 0 or 1:

$$z_{i,t} \doteq \begin{cases} \gamma\lambda z_{i,t-1} & \text{if } x_{i,t} = 0, \\ 1 & \text{otherwise,} \end{cases} \tag{5.9}$$

where $z_{i,t}$ is the ith component of the eligibility trace $\mathbf{z}$ at time t, and $x_{i,t}$ represents the ith component of the binary feature vector $\mathbf{x}$ at time t.

If the constraint of the binary feature vector is met, an Emphatic-Sarsa(λ) method using replacing traces is possible. And we come up with the emphatic version of the replacing trace equation as:

$$z_{i,t} \doteq \begin{cases} \gamma\lambda z_{i,t-1} & \text{if } x_{i,t} = 0, \\ M_t & \text{otherwise.} \end{cases} \tag{5.10}$$

The complete algorithm of the Emphatic-Sarsa(λ) method using replacing traces is similar to Algorithm 2. We only need to modify the line 16 of Algorithm 2 to:

```
1 $\mathbf{z} \leftarrow \gamma\lambda\mathbf{z}$
2 Loop for $x_i$ in $\mathcal{X}(S, A)$:
3     If $x_i = 1$:
4         $z_i \leftarrow M$
```

where $\mathcal{X}(S, A)$ is a function to return the binary feature vectors for the given state S and action A.

5.3 The Mountain Car Example

We used the mountain car control problem[1] as a testbed to compare the Emphatic-Sarsa methods with the corresponding non-emphatic methods. Consider the task of driving a car in a U-shaped mountain valley road. The position and the velocity of the car are two state-related variables of this task. The position of the car at time t is denoted as x_t between -1.2 and 0.5, and the velocity of the car at time t is denoted as $\dot{x}_t$ between -0.07 and 0.07. The action of the car at time t is denoted as A_t which can possibly be full throttle froward (+1), zero throttle (0) and full throttle backward (-1). The relation between the position x_t, the velocity $\dot{x}_t$ and the action A_t follows a simplified physical model:

$$\begin{aligned} x_{t+1} &\doteq \mathit{bound}[x_t + \dot{x}_{t+1}] \\ \dot{x}_{t+1} &\doteq \mathit{bound}[\dot{x}_t + 0.001A_t - 0.0025\cos(3x_t)], \end{aligned}$$

where the operation of *bound* limits the position within $-1.2 \leq x_t \leq 0.5$ and the velocity within $-0.07 \leq \dot{x}_t \leq 0.07$.

In each episode, the car will start from a random position $x_0 \in [-0.6, -0.4)$ which is near the bottom of the mountain valley at the velocity $\dot{x}_0 = 0$. When the car reaches the left bound $x_t = -1.2$, the velocity $\dot{x}_t$ will be reset to 0. When the car reaches the right bound $x_t = 0.5$, the goal is reached and the episode will terminate. The reward of the task is -1 for each time step until the goal is reached. If we take a good look at the physical model, we can find that the gravity is larger than the accelerating ability which makes the car unable to reach the goal directly even if it always chooses the action of full throttle forward. To reach the goal, the car needs to first move toward the opposite direction (left) to build up enough gravitational potential energy so it can then take the action of full throttle toward right to reach the goal because of the extra kinetic energy transformed from the gravitational potential energy. The difficulty of this task is that something "bad" needs to be done before things get better, which makes the mountain car example a great control problem.

[1]The mountain car example we used here is from Sutton (1996) which is based on a similar example from Moore (1990).

We still used the tile coding introduced in Section 2.7 and used in Chapter 4 to construct the function approximation feature vectors. But unlike in the random walk example, we have two state-related variables and the additional actions in the mountain car task so the construction of the tile coding were different. We still had 5 tilings, but we had different $n \times n$ tiles for each tiling. The two state-related variables form a two-dimensional space which makes the numbers of tiles for each tiling in $n \times n$ forms. Then the size of the feature vectors is $5 \times n \times n$, and the size trebles to $3 \times 5 \times n \times n$ when we consider the three actions. The first $5 \times n \times n$ feature vectors represent the action of full throttle forward, the second $5 \times n \times n$ feature vectors represent the action of zero throttle and the third $5 \times n \times n$ feature vectors represent the action of full throttle backward. The weights and the eligibility traces (just for the Sarsa(λ) methods) are all initialized to 0. The interest for all states and actions are equal to 1 and the probability of taking a random action parameter ε is 0.01 throughout this whole chapter.

5.4 Step-size Heuristics Used for On-policy Control Emphatic Methods

In the following sections, we compare our Emphatic-Sarsa methods with the corresponding non-emphatic methods empirically on the mountain car example. Because the mountain car example is an undiscounted task, divergence will arise if a fixed step size is used as in the prediction problems. Thus we multiplied the step-size parameter α by the adapting rate χ from Equation 3.35 for the n-step Emphatic-Sarsa method or from Equation 3.38 for the Emphatic-Sarsa(λ) method. In this whole chapter, all the step-size parameters of the emphatic methods actually used in the experiments are the step-size parameters shown in the figure times the corresponding adapting rates, and we will not explain this again.

5.5 Experiments of n-step Sarsa versus n-step Emphatic-Sarsa

Let us first compare the early performance of the n-step Sarsa method with that of the n-step Emphatic-Sarsa method. For the tile coding, the setting of 5 tilings with 8×8 tiles for each tiling would be used. We applied the n-step Sarsa method and the n-step Emphatic-Sarsa method with different n over step-size parameters α on the mountain car example to get a rough idea of the performance. And this also helped us to find some reasonable good parameter settings for the later learning curve experiments. Here 20 episodes are considered as one run and we had 100 such runs. After averaging over the episodes and runs, we got the result as shown in Figure 5.1.

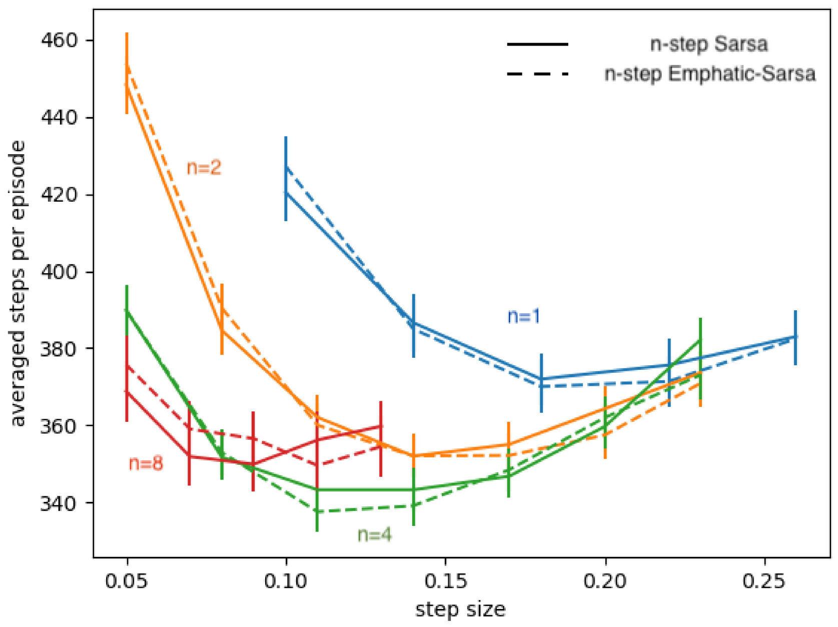


Figure 5.1: Parameter studies of the performance over the step size α between n-step Sarsa and n-step Emphatic-Sarsa for different n. The bars show the ranges of the standard errors.

The y-axis of the figure is the average steps the car spent to reach the goal over the first 20 episodes and 100 runs. We could find that the early performance of the two methods (the dash lines and the solid lines) under this setting were quite similar. The local optimums of the two methods were both near the point where the step-size parameter $\alpha = 0.125$ and $n = 4$, and we could use these parameters for the following complete learning curve experiment.

With $\alpha = 0.125$ and $n = 4$, we ran 100 episodes for each run, then the complete learning curve of the two methods after such 100 runs is shown in Figure 5.2.

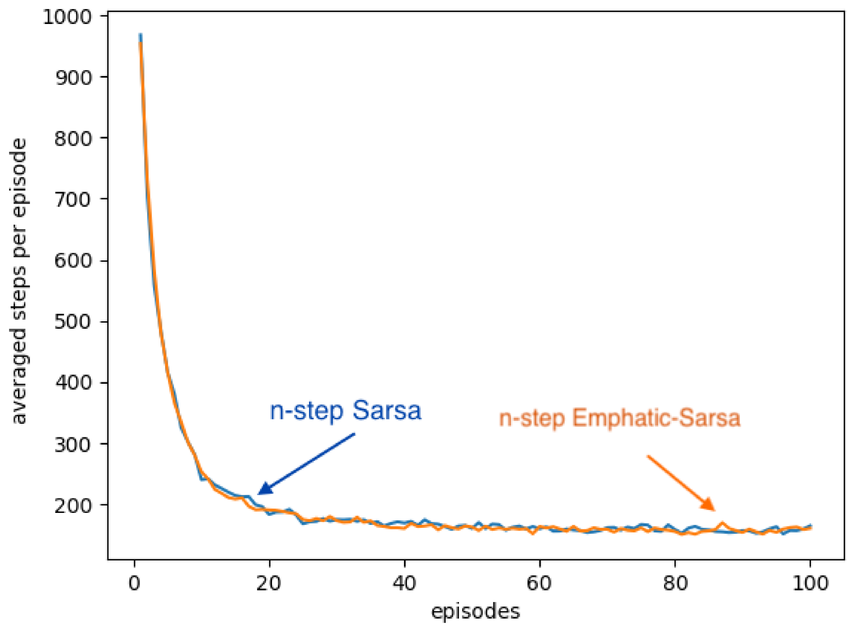


Figure 5.2: Learning curves between n-step Sarsa and n-step Emphatic-Sarsa both using high resolution tile coding on the mountain car example.

The y-axis of the figure is the average steps the car spent to reach the goal over 100 runs. From the figure we could see that the two methods had similar learning curves, and the average steps of both methods dropped below 200 steps and remained there with low variances after the early episodes. The tile coding used in this experiment is 5 tilings with 8×8 tiles for each tiling which could be called as the high resolution tile coding.

Then we reran the experiments with the same setting except using a low resolution tile coding. The low resolution tile coding is 5 tilings with 4×4 tiles for each tiling and the corresponding result is shown in Figure 5.3.

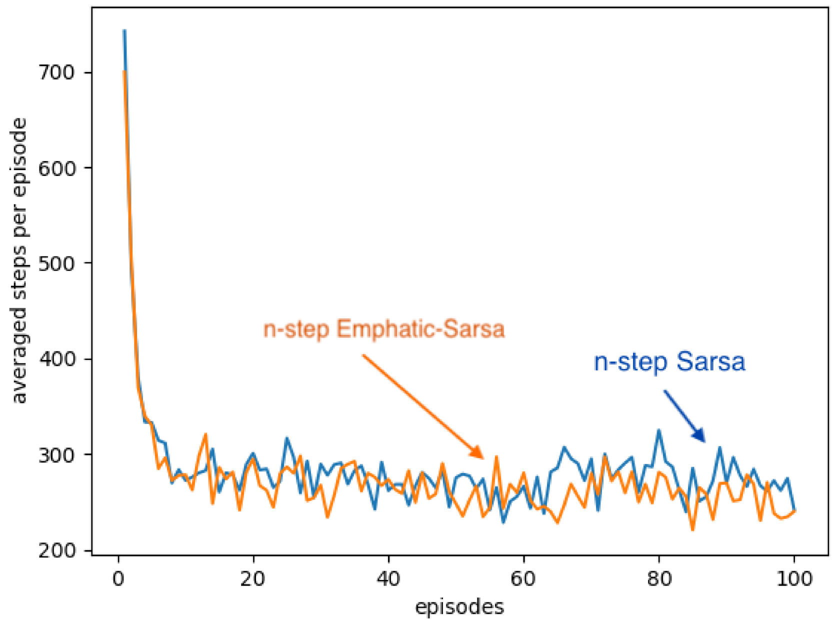


Figure 5.3: Learning curves between n-step Sarsa and n-step Emphatic-Sarsa both using low resolution tile coding on the mountain car example.

By using a low resolution tile coding, the length of the feature vectors is smaller which means less computational resources are needed, but the performance is sacrificed at the same time. We could find that the learning curves of the two methods stopped dropping near the level of 300 steps and had much larger variances comparing to using high resolution tile coding. But the trends of the learning curves of the n-step Sarsa method and the n-step Emphatic-Sarsa method were still very similar when using the low resolution tile coding.

5.6 Experiments of Sarsa(λ) versus Emphatic-Sarsa(λ)

Then we did some experiments to compare the Sarsa(λ) method with the Emphatic-Sarsa(λ) method on the mountain car example. Due to the fact that we use the tile coding as the function approximation method and tile coding produces binary feature vectors, we can use replacing traces in this situation.

Thus we used both the accumulating traces and the replacing traces as the eligibility traces for both the Sarsa(λ) method and the Emphatic-Sarsa(λ) method.

Let us still start with comparing the early performance of the two methods using the two kinds of eligibility traces. And we also used the setting of 5 tilings with 8×8 tiles for each tiling for tile coding. The two methods using the two kinds of eligibility traces were applied to the mountain car example for 100 runs with different decay-rate parameters for eligibility traces λ over step-size parameters α, and there were only 20 episodes for one run by which the early performance of those methods was presented. The result of the experiments is shown in Figure 5.4.

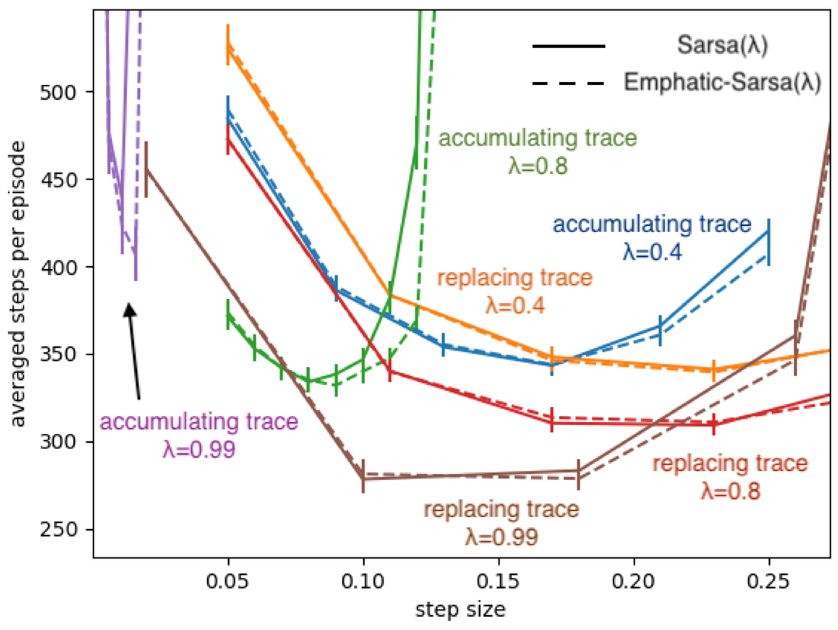


Figure 5.4: Parameter studies of the performance over the step size α between Sarsa(λ) and Emphatic-Sarsa(λ) both using accumulating traces and replacing traces for different λ. The bars show the ranges of the standard errors.

Note that we did not use $\lambda = 1$ in the experiments because the Emphatic-Sarsa(λ) method would be exactly the same as the Sarsa(λ) method in that case (Note the interest of all state-action pairs are 1 throughout this whole chapter). The y-axis of the figure is the average steps over the episodes and runs, and we could find that the early performance of the Emphatic-Sarsa(λ) method was also similar to the corresponding non-emphatic methods. From

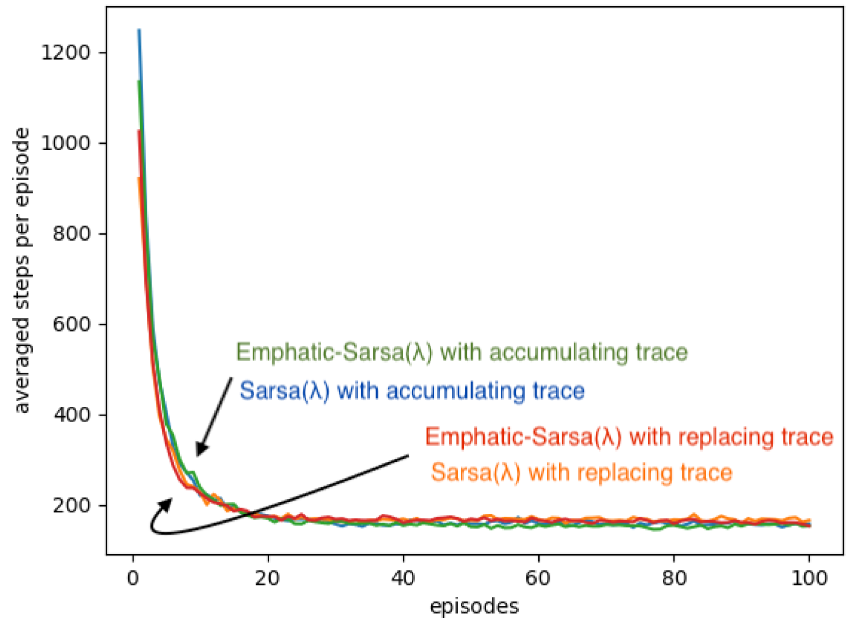


Figure 5.5: Learning curves between Sarsa(λ) and Emphatic-Sarsa(λ) with accumulating traces and replacing traces using high resolution tile coding on the mountain car example.

Figure 5.4 we could also see that the methods using replacing traces had better performance than the methods using accumulating traces, and the difference was in a wide range depending on the parameters. The methods using accumulating traces had the best performance when $\lambda = 0.8$, and the methods using replacing traces also had good performance when $\lambda = 0.8$. Although the methods using replacing traces had the best performance when $\lambda = 0.99$, the methods using accumulating traces had relatively bad performance when $\lambda = 0.99$, so we chose to use $\lambda = 0.8$ for the subsequent complete learning curve experiments.

When the decay-rate parameter for eligibility traces $\lambda = 0.8$, the local optimum for the accumulating traces was near the point where the step-size parameter $\alpha = 0.075$, and the local optimum for the replacing traces was near the point where $\alpha = 0.2$. By using this parameter setting, we could get Figure

5.5 as the complete learning curve of the two methods using the two kinds of eligibility traces. The result was averaged out over 100 runs, and the tile

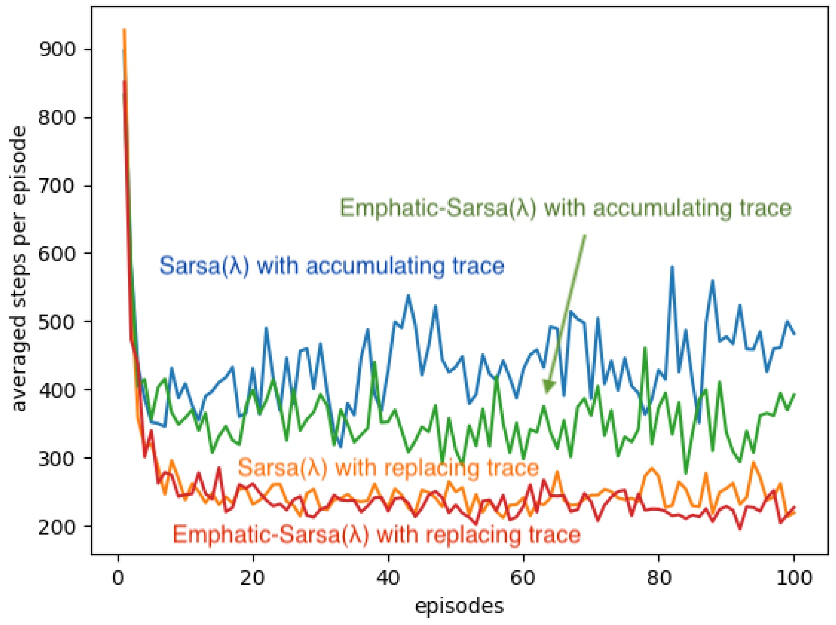


Figure 5.6: Learning curve between Sarsa(λ) and Emphatic-Sarsa(λ) with accumulating traces and replacing traces using low resolution tile coding on the mountain car example.

coding was the high resolution one with 5 tilings and 8×8 tiles for each tiling.

From Figure 5.5 we could see that the learning curves of the two methods using the two kinds of eligibility traces were almost overlapped, only the methods using replacing traces had slightly more rapid drop at the early episodes (around episode 10).

Again, we reran the experiments with the same parameters setting but using the low resolution tile coding with 5 tilings and 4×4 tiles for each tiling, and we got the result shown in Figure 5.6.

Comparing with using the high resolution tile coding, the low resolution tile coding made the result worse with larger variances. The average steps to reach the goal over the 100 runs of the methods using replacing traces were smaller than the methods using accumulating traces, and the variances of the methods using replacing traces were also smaller than the methods using accumulating traces. The performance of the Emphatic-Sarsa(λ) method using replacing traces was similar to the Sarsa(λ) method using replacing traces, but the performance of the Emphatic-Sarsa(λ) method using accumulating traces was obviously better than the Sarsa(λ) method using accumulating traces.

The above experiments showed that the on-policy emphatic methods worked in control problems. More specifically, the Emphatic-Sarsa methods performed at least as well as the corresponding non-emphatic methods in all our experiments. And in some special cases such as when using low resolution function approximation and accumulating traces as eligibility traces, the Emphatic-Sarsa(λ) method were better than the corresponding non-emphatic method.

Chapter 6

Conclusion

In this thesis, we did some studies on emphatic reinforcement learning and made contributions in two separate but correlated areas.

First, we have designed new heuristics for reliably adapting step sizes to emphatic methods. More specifically, our step-size heuristics provide a way to transform the Emphatic-TD methods to the corresponding non-emphatic methods by adjusting the step-size parameters when the interest of all states are equal. Based on the same idea, we can use the step-size heuristics to find the appropriate range for the step-size parameters of the emphatic methods. When the interest of states are not all equal, the Emphatic-TD methods and the corresponding non-emphatic methods can not be the same, but there still exists sub-heuristics of our step-size heuristics work in this case. In the empirical evaluations of the step-size heuristics, the results showed that the Emphatic-TD methods using the step-size heuristics and the corresponding non-emphatic methods had almost the same performance when the interest of all states are equal. When the interest of states are not all equal, the Emphatic-TD methods using the step-size heuristics outperformed the corresponding non-emphatic methods because our step-size heuristics can retain the advantages of the emphatic methods.

Second, we also extended the idea of emphatic methods to the on-policy control methods, and presented the new algorithms as the n-step Emphatic-Sarsa method and the Emphatic-Sarsa(λ) method. We conducted some experiments to compare the performance of the Emphatic-Sarsa methods with the corresponding non-emphatic methods on the mountain car task. Because our empirical studies did not cover all aspects, we are not able to give a definite answer to the question if the new Emphatic-Sarsa methods are better than the corresponding non-emphatic methods. In most cases we studied, we found no significant difference between the performances of the Emphatic-Sarsa methods and the corresponding non-emphatic methods. In one particular case when using low resolution tile coding and accumulating traces on the mountain car task, we found that the Emphatic-Sarsa(λ) method performs better than the corresponding non-emphatic method. In our opinion, the new Emphatic-Sarsa methods may also be better than the corresponding non-emphatic methods in other situations. This suggests that further empirical studies of the new Emphatic-Sarsa methods in other situations such as using unequal interest can be interesting future works.

References

Bellman, R. (1957). A Markovian decision process. *Journal of Mathematics and Mechanics*, *6*(5), 679–684.

Ghiassian, S., Patterson, A., White, M., Sutton, R. S., & White, A. (2018). Online off-policy prediction. arXiv: 1811.02597

Ghiassian, S., Rafiee, B., & Sutton, R. S. (2017). A first empirical study of emphatic temporal difference learning. arXiv: 1705.04185

Gu, X., Ghiassian, S., & Sutton, R. S. (2019). Should all temporal difference learning use emphasis? arXiv: 1903.00194

Hallak, A., Tamar, A., & Mannor, S. (2015). Emphatic TD Bellman operator is a contraction. arXiv: 1508.03411

Hallak, A., Tamar, A., Munos, R., & Mannor, S. (2016). Generalized emphatic temporal difference learning: Bias-variance analysis. In D. Schuurmans & M. P. Wellman (Eds.), *Proceedings of the thirtieth AAAI conference on artificial intelligence* (pp. 1631–1637). AAAI Press.

Imani, E., Graves, E., & White, M. (2018). An off-policy policy gradient theorem using emphatic weightings. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems 31: Annual conference on neural information processing systems 2018* (pp. 96–106).

Karampatziakis, N., & Langford, J. (2011). Online importance weight aware updates. In F. G. Cozman & A. Pfeffer (Eds.), *UAI 2011, Proceedings of the twenty-seventh conference on uncertainty in artificial intelligence* (pp. 392–399). AUAI Press.

Mahmood, A. R., Yu, H., White, M., & Sutton, R. S. (2015). Emphatic temporal-difference learning. arXiv: 1507.01569

Moore, A. W. (1990). *Efficient memory-based learning for robot control* (tech. rep. No. UCAM-CL-TR-209). University of Cambridge, Computer Laboratory.

Rummery, G. A., & Niranjan, M. (1994). *On-line Q-learning using connectionist systems* (tech. rep. No. TR 166). Cambridge University Engineering Department. Cambridge, England.

Singh, S. P., & Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning*, *22*(1–3), 123–158. doi:10.1007/BF00114726

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*(1), 9–44. doi:10.1007/BF00115009

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in neural information processing systems 8* (pp. 1038–1044). Cambridge, MA, USA: MIT Press.

Sutton, R. S., & Barto, A. G. (1998). *Introduction to reinforcement learning* (1st ed.). Cambridge, MA, USA: MIT Press.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). Cambridge, MA, USA: MIT Press.

Sutton, R. S., Mahmood, A. R., & White, M. (2016). An emphatic approach to the problem of off-policy temporal-difference learning. *Journal of Machine Learning Research*, *17*(73), 1–29.

Tian, T., & Sutton, R. S. (2019). Extending sliding-step importance weighting from supervised learning to reinforcement learning. In A. E. F. Seghrouchni & D. Sarne (Eds.), *Artificial intelligence. IJCAI 2019 international workshops* (Vol. 12158, pp. 67–82). doi:10.1007/978-3-030-56150-5_4

Watkins, C. J. C. H. (1989). *Learning from delayed rewards* (Doctoral dissertation, King's College, Cambridge, UK).

Yu, H. (2015). On convergence of emphatic temporal-difference learning. In P. Grünwald, E. Hazan, & S. Kale (Eds.), *Proceedings of the 28th conference on learning theory* (Vol. 40, pp. 1724–1751). JMLR.org.

Yu, H. (2016). Weak convergence properties of constrained emphatic temporal-difference learning with constant and slowly diminishing stepsize. *Journal of Machine Learning Research*, *17*(220), 1–58.