Interrelating Prediction and Control Objectives in Episodic Actor-Critic Reinforcement Learning

by

Valliappa Chockalingam

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

Master of Science

Department of Computing Science

University of Alberta

Abstract

The reinforcement learning framework provides a simple way to study computational intelligence as the interaction between an agent and an environment. The goal of an agent is to accrue as much reward as possible by intelligently choosing actions given states. This problem of finding a policy that maximizes the expected total reward is called the control problem. Many algorithms that solve the control problem also solve a related problem called the prediction problem. The goal in prediction is to accurately estimate the expected total reward from different states while following a given policy.

Both the prediction and control problems can be formulated as optimization problems with different objectives. In this thesis, we present the first attempt at interrelating the prediction and control objectives. Prediction has often been studied independently in the past, but it is a subproblem subservient to the primary control problem that we generally care about.

There is a need for interrelating the objectives particularly when the number of states is large and agents cannot learn about all states equally well. Agents are forced to use function approximation, and we have to specify how to allocate function approximation resources.

The Emphatic Temporal Difference algorithm is the first prediction algorithm that allows a specification of a degree of caring about value function accuracy at different states. This specification is done through an interest function that changes the prediction objective. We now have a way to communicate to the agent how much we care about different states beyond how often they occur. However, interest is strictly a problem-related concept, defining the objective, in prediction. There were no clear insights about how to set the interest.

We take this opportunity to use interest as a solution-related concept in control and ground the choice of prediction objective in improving control performance. As a running example, we use the episodic discounted control problem where distal rewards are worth exponentially less, and interaction terminates upon reaching a terminal state. In this problem setting, we study the actor-critic class of solution methods that neatly separate the two problems with the actor solving the control problem and the critic solving the prediction subproblem. First, we discuss a recent controversy in discounting and the role it should play in the learning updates of the actor. After concluding how we should update the actor, we move on to the critic. We find the first suggestion of an interest over states for the prediction problem by analyzing the updates made by the actor and hence present a choice of prediction objective motivated by a control objective. Experimental results confirm that control performance is indeed improved when the critic uses the new prediction objective.

Preface

No part of this thesis has been previously published.

To my family

Meanings are not determined by situations, but we determine ourselves by the meanings we give to situations.

– Alfred Adler

Acknowledgements

First and foremost, I would like to thank my supervisor, Richard Sutton. He has taught and motivated me to think clearly and deeply. My numerous discussions with him have been insightful on research, careers, and life, and have undoubtedly left a deep impression on me which I will cherish for a long time to come. I would also like to thank Martha White and Michael Bowling for being on my examining committee and taking the time to understand my work. Finally, I would be remiss if I did not acknowledge the wonderful interactions I had with many RLAI and AMII members under various capacities and I thank them for all their help, encouragement, and fun banter.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis presents ideas and contributions within the Reinforcement Learning (RL) framework (Sutton and Barto, 2018). It is a simple framework for testing computational intelligence, the computational part of the ability to achieve goals in the world (McCarthy, 2007). The core of RL is the interaction between an agent and an environment (the agent's world) over time. In the Markov Decision Process (or MDP) mathematical formulation of RL which we use throughout this thesis, learning is associative. At each timestep, the agent learns and acts with a state describing the agent's current situation. The environment produces a resultant scalar reward and the agent transitions to a new state.

The goal of an RL agent is to find a way of behaving, *i.e.* a policy, that accrues as much reward as possible. This maximization problem over policies is commonly known as the control problem.

RL differs fundamentally from other forms of learning because feedback is evaluative. Upon taking an action, the environment produces a scalar reward that only suggests how good the action the agent just took was. The agent is not instructed as to what the best action was. Moreover, the immediate reward may provide little information about the long term goodness of the action. So, the agent may need numerous samples of cumulative rewards to learn a good policy for control. These characteristics of the reward signal make it difficult and slow to learn from the reward signal alone.

We can now describe two types of feedback for agents to learn from when solving control problems, primary and secondary reinforcers. The rewards that an agent gets while following a policy solely determine whether the policy is good or bad. The reward signal provided by the environment can therefore be called the primary reinforcer. As discussed before, learning with the primary reinforcer alone can be difficult and slow. Secondary reinforcers that have a history of association with the primary reinforcer can be used to aid in solving the primary control problem.

Prediction is the general problem of learning secondary reinforcers. For example, an agent can learn to predict whether a particular property holds about a state some timesteps in the future. Such predictive knowledge can be incorporated into the state to help in maximizing expected return. Prediction can therefore be a useful subproblem for the primary control problem. Such generality in the definition of prediction problems is highlighted in the line of work on General Value Functions (Sutton et al., 2011). However, when we talk about prediction in this thesis we refer to a particular secondary reinforcer, the value function. The value of a state under some policy is simply the expected total reward or expected return from that state when following the policy at hand. We call the problem of estimating the value function for a given policy the prediction problem. Note that this specific prediction problem can be particularly useful for control. The general principle is that, by learning a state-value function, the agent can change the policy to visit states with higher value more.

The prediction and and control problems can both be formalized as optimization problems with different objectives. By the term objective, we refer to the function we optimize when solving an optimization problem. We are generally interested only in how well the control problem is solved. So, we can freely choose among prediction objectives the one that would best aid in solving the control problem. We would thus like a way to find such a prediction objective with a given control objective. In this thesis, we aim to do exactly this and we refer to this process alternatively as interrelating the prediction and control objectives.

A setting in which this need to interrelate the objective arises is when the number of states is very large. If there are only a few states, the agent can maintain a table and learn policies and values for each state independently. However, if there are many states, agents are forced to approximate functions and make tradeoffs between how well functions are approximated at different states. So, there is a need to specify how to allocate function approximation resources. If we have any information about how we should allocate function approximation resources for the prediction problem from the control problem, we can communicate it to the prediction algorithm to improve control performance. However, we did not have a mechanism to do so until recently.

The Emphatic Temporal Difference (ETD) learning algorithm (Mahmood et al., 2015; Sutton et al., 2016) is the first prediction algorithm that allows a user to specify how much we care about prediction error in different states. Prior algorithms learned about states to the extent they were encountered. ETD gives us flexibility through a user-defined interest function that maps each state to a non-negative scalar. If a state has high interest, we care more about reducing its prediction error and vice versa. Thus, changing the interest changes the prediction objective. The fundamental difference between ETD and other algorithms is that ETD calculates an emphasis every timestep that accumulates the interest over time we have in accurately predicting the current state's value. This emphasis is then used to emphasize or de-emphasize updates.

ETD was proposed as a stable prediction algorithm for the off-policy prediction problem and it was later shown to also be convergent as well (Yu, 2015). Off-policy learning entails learning about the value function under a target policy with data from a behaviour policy. However, we will be focusing on the on-policy case of ETD as the control algorithm we study is on-policy.

In this thesis, we study a control algorithm that makes the separation and relation between the prediction and control problems clear. Actor-critic algorithms (Sutton, 1985) are a class of solution methods that solve both the prediction and control problems. The actor maintains a policy, chooses actions, and attempts to solve the control problem. The critic maintains an approximate value function, provides evaluative feedback to the actor, and attempts to solve the prediction problem. Additionally, actor-critic methods use parameterized function approximators for both the actor and critic. We can use an ETD critic instead of, say, the commonly used TD critic. A natural question then arises: "What interest and hence what prediction objective should we use in an actor-critic setup?"

To answer this question, we first have to specify the control objective we care about. We focus on the episodic discounted setting in this thesis. This setting is commonly used in practice and it is perhaps the setting which we understand the most. In this problem setting, agents begin interaction with the environment at a start state and interaction ends when the agent reaches a terminal state. Rewards are downweighted in an exponentially decaying manner by a discount factor γ. The control objective in this setting is simple. The agent's goal is to find a policy that maximizes the start-state value and the objective is therefore the start-state value.

We now discuss the actor and how the control problem is solved by an actor-critic agent. We need to do this before we can seek for an interest as the choice of prediction subproblem depends on the control problem. To solve the control problem, the actor estimates and moves the parameters in the direction of what is known as the policy gradient. The policy gradient is simply the gradient of the parameters of the policy with respect to the objective we would like to maximize, the expected cumulative reward.

With the episodic discounted setting in mind, recent work has shown that the learning updates used by the actor in practice are missing an important γ^t term where t is the timestep. We reiterate that this term arises out of the policy gradient theorem which specifies how the parameters should be moved in order to improve the policy. We also present a simple novel experiment to show that the omission of the term can result in learning poor policies.

Having discussed the issue of how we should incorporate discounting in the actor, we seek for and find an interest over states motivated by an analysis into the learning updates of the actor. Thus, we provide the first suggestion of a particular prediction objective for improving control performance. In the resulting interest, we find that there is a similar decaying term to the γ^t term in actor updates, γ^{2t}. Moreover, we interestingly find that the interest includes the norm of a vector called the characteristic eligibility from the previous timestep. The characteristic eligibility is the gradient of the log probability of taking the action that was chosen by the actor with respect to the parameters of the policy. There is a simple interpretation of this latter term as having a high interest in states that we can get to more with small changes in the policy parameters.

We present empirical results comparing the control performance of actor-critics when the critics use various settings of the interest motivated by the γ^t term in actor updates and the interest that we found from our seeking for an interest. The goal of the experiments is to see whether there is a need for an interest with both the decaying γ^{2t} term and the characteristic eligibility term. First, we present a set of simple experiments for showcasing that using a decaying interest of γ^t or γ^{2t} performs the best compared to other settings of interest such as the one used by TD. Next, we present results on commonly used benchmark tasks of Mountain Car and Puddle World. Similarly, we present results which show that incorporating the characteristic eligibility term helps and a novel algorithm we introduce called Sliding Step MINCE-AC achieves the best control performance across a wide range of settings for the algorithm parameters.

To conclude this chapter, the main contributions of this thesis are that we:

- Show that the commonly used TD(λ) prediction algorithm can be viewed as solving different prediction problems depending on the problem parameter γ and the solution parameter λ when the interest can depend on time,
- Present novel empirical results showing that a γ^t term in actor updates is necessary in actor updates when solving the episodic discounted start-state formulation control problem,
- Seek for and find an interest over states given the updates of episodic actor-critic and hence propose for the first time a way of interrelating the prediction and control objectives,

- Introduce new algorithms that are novel variations of episodic actor-critic algorithms where the critics optimize for different prediction objectives as a result of using an interest inspired by the novel interest we found, and,

- Demonstrate empirically that using the novel algorithms results in improved control performance.

Chapter 2

Background on RL, Prediction, and Control

The aim of this chapter is to present the RL framework and the Markov Decision Process mathematical formulation that we use in the rest of this thesis. We also discuss the prediction and control problems of RL along with commonly used objectives for these problems. This discussion is important as we later will be be seeking for a prediction objective given a control objective. This seeking has become possible due to a recent algorithm called Emphatic TD which is discussed in the following chapter. To understand Emphatic TD and contrast it with algorithms that existed before, a good grasp of prior prediction algorithms is needed and hence relevant prediction algorithms to our discussions are also presented in this background chapter. I delay the presentation of the algorithms for control and the actor-critic algorithm in particular to later chapters where the actor and critic components and related ideas will be introduced as and when necessary.

2.1 Reinforcement Learning

Reinforcement Learning (RL) is an approach to artificial intelligence wherein agents interact with an environment by perceiving states, taking actions, and receiving resultant rewards and new states. States are generally some useful representation of an agent's current situation. At each discrete timestep t, an agent takes an action, denoted A_t, at state S_t. As a result, the environment produces a new state S_{t+1} and reward R_{t+1}. The reward serves as evaluative feedback about how good the action taken was. In the commonly used episodic setting we consider throughout this thesis, agents start in a starting state S_0 and keep interacting with the environment until a terminal state S_T is reached when interaction stops. This agent-environment interaction results in a trajectory or episode containing all the states, actions and rewards, $\{S_0, A_0, R_1, S_1, ..., S_{T-1}, A_{T-1}, R_T, S_T\}$.

The typical mathematical formulation of an environment is known as a Markov Decision Process, often abbreviated as MDP. An MDP can be concisely expressed as a tuple $\langle \mathcal{S}, \mathcal{A}, p, \gamma \rangle$. $\mathcal{S}$ is the set of states and $\mathcal{A}$ is the set of actions. $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathbb{R} \rightarrow [0, 1]$ is the transition model which specifies the probability distribution over next states and rewards given a state and action, $p(r, s'|s, a) \doteq Pr(R_{t+1} = r, S_{t+1} = s'|S_t = s, A_t = a)$. Note that $\doteq$ is used to signify equality by definition throughout this thesis. Also, note that the next state and reward distributions are functions of the current state and action and independent of history. This property of the environment is often called the *Markov* property. The transition model is unknown to the agent. In model-free RL, which we focus on in this thesis, no attempt is explicitly made to learn an approximate model. Finally, the *discount factor* $\gamma \in [0, 1]$ weights later rewards exponentially less. Thus, γ controls how much rewards received earlier are worth as compared to rewards received later.

An agent selects actions according to its *policy* which specifies probability distributions over actions conditioned on states:

$$\pi(a|s) \doteq Pr(A_t = a|S_t = s).$$

One key random variable in the reinforcement learning problem setting is the *return*, defined as follows:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$$

The expected return from a state s following policy π is known as the value of state s under policy π. Concisely, we represent the above statement and define the *value function* v_π at state s as:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s] \tag{2.1}$$

We can similarly define the *action-value function*, q_π as follows:

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \tag{2.2}$$

To distinguish between v_π and q_π, we sometimes explicitly refer to v_π as the *state-value function*.

Finally, there is an important relation between q_π and v_π. $v_\pi(s)$ is simply the average of $q_\pi(s,\cdot)$ weighted by the policy $\pi(\cdot|s)$:

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s,a) \tag{2.3}$$

2.2 Control and the Episodic Discounted Control Objective

The primary problem of reinforcement learning is to find a policy that maximizes expected return. This is often known as the *control* problem. Note that we have to specify which states we wish to maximize expected returns from. The specific control problem we are interested in this thesis is the control problem in the episodic discounted setting. In an episodic problem, we can compare policies simply by comparing their expected returns (or value by (2.1)) from the start state. The episodic discounted *start-state formulation* control problem can be expressed as the following optimization problem:

$$\max_\pi v_\pi^\gamma(s_0) \tag{2.4}$$

The superscript γ above is used to make it explicit that the value function definition uses discounting. We may omit this, but it is good to note here that whenever we discuss about value functions or returns further, we refer to the definitions with discounting. The objective for the episodic discounted control problem is the start-state value, $v_\pi^\gamma(s_0)$. This is because, by the term objective, we mean the function which we try to optimize in an optimization problem.

2.3 Prediction

Apart from the primary control problem, another important problem in reinforcement learning is to estimate the value function for a given policy. This problem is often referred to as the prediction or policy evaluation problem and serves as a useful subproblem to the primary control problem of RL. Note that when talking about prediction, we generally mean the problem of estimating the state-value function, but, without loss of generality, we can define prediction problems with the action-value in mind problem and algorithms can be modified appropriately. Having said that, like in the control problem, we again have to specify how much we care about value prediction error in different states. We will introduce the commonly used prediction objective in this chapter. This objective has a weighting on states such that the agent cares about states to the extent they occur. Before explaining this objective however, we first need to introduce function approximation which follows below.

2.4 Function Approximation

In general, the environment is much larger than the agent. Thus, for example, the state space of the environment is usually very large. Tabular methods that maintain a table of estimates for each state are generally not viable in such scenarios due to memory constraints. Instead, we use function approximation to get approximate value functions and approximate policies.

The general idea of function approximation is to assume that there is a function f which we would like to approximate well. However, we only observe some samples or get noisy estimates from f. We can define a parameterized function approximator, $\hat{f}_{\mathbf{w}}$ or $\hat{f}(\cdot, \mathbf{w})$, parameterized by weights $\mathbf{w}$. The key problem in function approximation is how to find a $\mathbf{w}$, and hence $\hat{f}_{\mathbf{w}}$, that well approximates f.

This thesis focuses on the linear function approximation case of algorithms where theoretical analysis and empirical evaluations are easier.

In linear value function approximation, the function approximator we use is defined by a single set of parameters or weights $\mathbf{w}$ and the approximation to the value of state s under policy π is given as follows:

$$\hat{v}(s, \mathbf{w}) \doteq \mathbf{x}(s)^T \mathbf{w} \tag{2.5}$$

where $\mathbf{x}(s)$ is the *feature vector* of state s, given by the environment, and both $\mathbf{w}$ and $\mathbf{x}(s)$, for all s, are in $\mathbb{R}^d$.

Similarly, we use linear function approximation for parameterizing policies with SoftMax applied on the outputs to generate a probability distribution. We use $\boldsymbol{\theta} \in \mathbb{R}^{d \times |\mathcal{A}|}$ generally to denote the parameters of the policy parameterization. Thus, the policy is defined as follows:

$$\pi(a|s, \boldsymbol{\theta}) \doteq \frac{e^{\mathbf{x}(s)^T \boldsymbol{\theta}[a]}}{\sum_b e^{\mathbf{x}(s)^T \boldsymbol{\theta}[b]}} \tag{2.6}$$

where the notation $[\cdot]$ is used to index the vector $\mathbf{x}(s)^T \boldsymbol{\theta}$.

2.5 Mean Squared Value Error Prediction Objective

Having briefly described function approximation, we come back to define a commonly used prediction objective. Given a policy π, the aim of an agent in the prediction problem is to accurately estimate the values of different states, i.e., estimate v_π with a function approximator $\hat{v}_\mathbf{w}$. Let the probability distribution over states encountered while following policy π be μ, commonly known as the on-policy state distribution. Then, one prediction problem can be defined as follows:

$$\min_{\mathbf{w}} \sum_s \mu(s) \left[v_\pi(s) - \hat{v}(s, \mathbf{w}) \right]^2 \tag{2.7}$$

The objective above is called the Mean Squared Value Error or $\overline{\text{VE}}$ objective. The term 'mean' and the bar in $\overline{\text{VE}}$ indicate that this objective is the average approximation error in the state-values weighted by the distribution of states as they are encountered while following policy π.

Note that the $\overline{\mathrm{VE}}$ cannot be minimized directly as agents do not have access to the value function. Agents generally minimize some surrogate learning objective.

2.6 Solution Methods for Policy Evaluation

In this section we discuss different solution methods for the prediction problem. First, we discuss Monte-Carlo methods, then one-step TD, and finally TD(λ).

2.6.1 Gradient Monte-Carlo for Policy Evaluation

Gradient Monte-Carlo for policy evaluation, like the other methods we study in this thesis, is a stochastic gradient style method where parameters are updated in the direction of the gradient. In this algorithm, we run policy π and wait till the end of an episode. Then, at the end of the episode, we have a sample return from each state visited in the episode. We can then move the parameters in a direction that gets the estimated values of each state closer to the sample returns. The full pseudocode for the algorithm with linear function approximation is given in the box titled Algorithm 1.

Algorithm 1: Gradient Monte-Carlo for Policy Evaluation

```
Input: policy $\pi$
Input: a differentiable state-value function $\hat{v}(s, \mathbf{w})$
Parameters: step-size $\alpha_{\mathbf{w}} > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$
foreach episode do
    Generate trajectory $\{S_0, A_0, R_1, S_1, ..., S_{T-1}, A_{T-1}, R_T, S_T\}$
      following $\pi$
    foreach timestep t in episode do
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha_{\mathbf{w}}[G_t - \hat{v}(S_t, \mathbf{w})]\mathbf{x}(S_t)$
    end foreach
end foreach
```

Here, we see for the first time the step-size parameter. It specifies to what extent we move in the direction of the gradient. Or, in other words, the step-size parameter modulates the step-size. Large step-sizes can result in large fluctuations while small step-sizes can result in slow learning. Gradient Monte-

Carlo uses sample returns which are unbiased estimates of the expected return. So, with small step-sizes and enough samples, in the limit, the value estimates will converge to $\hat{v}(s, \mathbf{w}^*)$ where $\mathbf{w}^* = \min_{\mathbf{w}} \overline{\text{VE}}$. However, sample returns can have high variance as they involve the sum of many random variables and thus convergence to $\mathbf{w}^*$ can be slow.

Now, it would be good to understand how an agent can minimize the $\overline{\text{VE}}$ even without having access to the true value function. The gradient Monte-Carlo algorithm in fact reduced another proxy error called the Mean Squared Return Error. The $\overline{\text{RE}}$ is given by $\overline{\text{RE}}(\mathbf{w}) = \mathbb{E}[(G_t - \hat{v}(S_t, \mathbf{w}))^2] = \overline{\text{VE}}(\mathbf{w}) + \mathbb{E}[(G_t - v_\pi(S_t))^2]$ (Sutton and Barto, 2018). Only the $\overline{\text{VE}}$ term depends on the weight vector and so, if we minimize the $\overline{\text{RE}}$, we minimize the $\overline{\text{VE}}$ and the variance term that does not depend on the parameter vector $\mathbf{w}$ can be ignored.

2.6.2 One Step Temporal Difference Learning

Temporal difference (TD) methods learn incrementally over time within an episode without having to wait till the end of an episode to perform learning. In doing so, TD methods move towards the online end of the online-offline spectrum of solution methods. Namely, they rely on less waiting into the future and make updates sooner than Monte-Carlo methods in which agents must wait till the end of an episode to make learning updates.

A central property of temporal difference methods is the use bootstrapping. They improve a value estimate using another value estimate. Particularly, they rely on important relations between the value function at successive states, often known as the Bellman equations. Recall from (2.1) that the value function $v_\pi(s)$ is the expected return of state s under policy π. The derivation of the Bellman equation for v_π follows below:

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi\left[\sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1} \middle| S_t = s\right] \\ &= \mathbb{E}_\pi\left[R_{t+1} + \sum_{k=1}^{T-t-1} \gamma^k R_{t+k+1} \middle| S_t = s\right] \end{aligned}$$

$$= \mathbb{E}_\pi \left[R_{t+1} + \gamma \sum_{k=0}^{T-(t+1)-1} \gamma^k R_{(t+1)+k+1} \middle| S_t = s \right]$$
$$= \mathbb{E}_\pi \left[R_{t+1} + \gamma G_{t+1} \mid S_t = s \right]$$
$$= \mathbb{E}_\pi \left[R_{t+1} + \gamma \mathbb{E}_\pi[\mathbb{E}_\pi[G_{t+1} \mid S_{t+1}] \mid S_t = s] \mid S_t = s \right] (\because \text{Law of iterated expectations})$$
$$= \mathbb{E}_\pi \left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s \right] (\because \text{Markov assumption}) \tag{2.8}$$

Let $\hat{v}(\cdot, \mathbf{w})$ be an approximation to v_π, then generally there will be some Bellman error, i.e., $|\hat{v}(s, \mathbf{w}) - \mathbb{E}[R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}_t)|S_t = s]| > 0$ where $\mathbf{w}_t$ are the parameters of the value function approximator at time t. While following π, we can consider minimizing the error $[\mathbb{E}_\pi [R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}_t)] - \hat{v}(S_t, \mathbf{w}_t)]^2$. However, while following policy π, we only observe one possible next state S_{t+1}. Since we will observe states S_{t+1} according to the on-policy state distribution, we can minimize with respect to the sample next state we observe. Thus, we can minimize the following error: $[R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t)]^2$. The term inside the square is called the (one-step) TD error or temporal-difference error, often denoted δ:

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

The gradient of the squared TD error with respect to $\mathbf{w}$ is $2\delta_t[\gamma \nabla_{\mathbf{w}} \hat{v}(S_{t+1}, \mathbf{w}_t) - \nabla_w \hat{v}(S_t, \mathbf{w}_t)]$. If we minimize the squared TD error presented above, we get a resulting algorithm known as the naive residual gradient algorithm. For the linear value function approximation case, the update equation is as follows with the 2 folded into the step-size parameter:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_{\mathbf{w}}[R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t)][\mathbf{x}(S_t) - \gamma \mathbf{x}(S_{t+1})]$$

This algorithm is undesirable in a few ways. One of them is that the solution it finds is one which performs temporal smoothing to ensure that the TD errors at all timesteps is small by having gradients flowing forward and backward in time. We are instead looking for accurate predictions. One-step semi-gradient TD methods do something different:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_{\mathbf{w}}[R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t)]\mathbf{x}(S_t)$$

As can be seen above, we ignore the gradient estimation of the next-state's value prediction in the update. We are not following the "full" gradient, and thus we refer to such an algorithm as a semi-gradient method. The one-step term refers to the fact that we are looking one-step ahead for a better estimate of $v_\pi(s)$. Such a technique of moving an estimate towards a better estimate after having seen more data is called bootstrapping. We sometimes refer to $R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w}_t)$ as the TD *target* or the one-step truncated return.

Like with Gradient Monte-Carlo, we do not have access to the true value function. TD(0) minimizes a different error called the Mean Squared Projected Bellman Error. We discussed the Bellman Error above. The project step simply involves projecting the Bellman Error vector which will generally lie outside of the representable space onto the lower dimensional space that is spanned by the parameter vector.

Finally, while TD(0) has lower variance than Gradient Monte-Carlo for prediction due to the inclusion of only one random variable in the target, it has higher bias since the estimate of the value of the next state might be incorrect. Given the advantages and disadvantages of TD(0) and Gradient Monte-Carlo for prediction, we now discuss an algorithm that strides between the two in the online-offline spectrum and allows for controlling this bias-variance tradeoff.

2.6.3 Temporal Difference Learning with Eligibility Traces

Instead of using the one-step truncated return, we can instead use an n-step truncated return as the target. By waiting for n steps while accumulating the rewards and then using our value function approximation, we can update towards an n-step TD target. The learning update for the n-step TD algorithm is given by:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_{\mathbf{w}}[G_{t:t+n} - \mathbf{w}_t^T\mathbf{x}(S_t)]$$

where $G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{n-1}R_{t+n} + \mathbf{w}_t^T\mathbf{x}(S_{t+n})$.

We can also move estimates towards any weighted average of n-step truncated returns as long as the sum of the weightings is 1. Thus, we could for example use $\frac{1}{2}G_{t:t+2} + \frac{1}{2}G_{t:t+4}$ as a target.

TD(λ) uses a decaying weighting over truncated returns. This can be seen from the TD(λ) target which is called the λ-return:

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}G_{t:t+n} + \lambda^{T-t-1}G_t$$

λ is known as the trace decay parameter. If we use $\lambda = 0$, we get the one step TD(0) algorithm. If we use TD(1), we get the gradient Monte-Carlo algorithm. Note that, to calculate the above λ-return for any $\lambda > 0$ we have to wait till the end of an episode. Thus, we have an offline forward-view TD(λ) algorithm where we have to look forward and wait till the end of the episode to perform updates and learn.

Instead, it has been shown that with equivalence at the end of episodes, we can use the concept of an eligibility trace to assign credit backward in time and get the mechanistic online backward view TD(λ) algorithm:

$$\mathbf{e}_t \leftarrow \gamma\lambda\mathbf{e}_{t-1} + \mathbf{x}_t, \text{ with } \mathbf{e}_{-1} = \mathbf{0} \tag{2.9}$$

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_{\mathbf{w}}\delta_t\mathbf{e}_t \tag{2.10}$$

Chapter 3

Interest, Emphasis and Emphatic TD

Most algorithms weight states proportional to the extent they are visited. In this thesis, we challenge this idea and aim to show that learning about states according to a different weighting is more appropriate for prediction when it serves as a subproblem for control. The Emphatic TD (ETD) algorithm (Mahmood et al., 2015; Sutton et al., 2016) is the first prediction algorithm that allows us to flexibly choose such a weighting over states. We make use of this algorithm in this thesis and therefore we naturally need to describe ETD. This chapter describes the algorithm and the important concepts of interest and emphasis that arose as a result of the work. I also present an analysis that suggests that TD(λ), a prior algorithm that does not have an explicit concept of interest, can also be viewed as an algorithm solving different prediction problems depending on the problem parameter γ (the discount factor) and the solution parameter λ (the trace decay).

3.1 Emphatic TD

There has been a line of recent work into off-policy policy evaluation. The goal in such a problem setting is to approximate the value function of a policy with data generated from a different policy. The policy used to collect data for training is called the behaviour policy, often denoted μ. The policy being learned about is the target policy, often denoted π. Thus, the goal in such a problem setting is to solve different prediction problems of the following form:

$$\min_{\mathbf{w}} \sum_{s} d(s) \left[v_\pi(s) - \hat{v}(s, \mathbf{w})\right]^2$$

The difference in the above objective from on-policy policy evaluation objective is the distribution of states that weight the prediction errors. Instead of d_π, we have some other distribution d. Methods differ in what weighting over states they use in their learning objectives. Some use the behaviour policy distribution d_μ. Others correct for the state distribution difference and use d_π in their objective.

Methods can be broadly characterized based on whether they only correct for the difference in behaviour (use posterior corrections) or whether, in addition, they correct for the difference in state distributions (use prior corrections) (Ghiassian et al., 2018).

Let us consider the difference in behaviour first. One assumption commonly made is the coverage assumption. We say that the behaviour policy has coverage if $\pi(a|s) > 0 \implies b(a|s) > 0$. Even with coverage, we have a mismatch between the probabilities of observing transitions when following the behaviour policy versus when following the target policy. We would like to make updates to our value function approximation so that it is as if the actions were taken according to the target policy π. The common approach for such posterior corrections is to use the importance sampling (IS) ratio. The IS ratio is simply the relative probability of taking the action taken, A_t, in the state observed, S_t, under the target and behaviour policies: $\rho_t \doteq \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}$. Off-policy TD(0) applies such a posterior correction to get the following update:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_{\mathbf{w}} \rho_t \delta_t \mathbf{x}(S_t)$$

Some methods also use prior corrections by correcting for the difference between the state distribution under μ and π. One way to correct for the state distribution weighting and effectively solve for the objective using d_π as the distribution d is to use the product of importance sampling ratios from timestep 0. Thus, we can modify off-policy TD(0) to get the following Alternative-life

TD(0) algorithm:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_{\mathbf{w}} \rho_t \left(\prod_{k=0}^{t-1} \rho_k \right) \delta_t \mathbf{x}(S_t)$$

The Alternative-life TD(0) algorithm can have unacceptably large variance since the product of importance sampling ratios can be very large.

Emphatic TD (ETD) performs a different prior correction and as a result has been shown to be stable and convergent with probability 1 (Sutton et al., 2016; Yu, 2015). At each timestep, it is assumed that an excursion away from the behaviour policy is taken. Then, instead of using the product of importance sampling ratios from the beginning of time, ETD uses the product of importance sampling ratios since the beginning of the excursion and tamps down the variance by the discount factor which can be treated as representing a probability of termination.

For the simplest case of one-step updates, ETD(0) can be described using the following two updates:

$$F_t \leftarrow \gamma \rho_{t-1} F_{t-1} + 1, \text{ with } F_0 \doteq 1 \tag{3.1}$$

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_{\mathbf{w}} \delta_t F_t \mathbf{x}(S_t) \tag{3.2}$$

F_t is a scalar known the followon trace which calculates the discounted state visitation under the target policy while doing excursions away from the behaviour policy. With the above updates, we can see that ETD uses a different weighting d over states. Namely, the followon weighting which is given by:

$$f(s) = d_\mu(s) + \gamma \sum_{\bar{s},\bar{a}} d_\mu(\bar{s}) \pi(\bar{a}|\bar{s}) p(s|\bar{s},\bar{a}) + ...$$

The followon captures bootstrapping relationships. Thus, ETD makes bigger updates to a state if many previous states care about its value due to their value estimates relying on its estimate.

Now, let us briefly get back to the mean squared value error objective. The distribution under the behaviour policy alone is limiting. We are bound to care about states based on how often they are visited. However, we may care about the value error in states in a different way and define a more flexible objective with a different optimization problem. Let interest i be a mapping from states to positive scalars so that $i : \mathcal{S} \rightarrow [0, \infty)$. Interest is then defined to be the $i(s)$ weighting on states we have in the following objective:

$$\min_{\mathbf{w}} \sum_{s} d_\mu(s) i(s) \left[v_\pi(s) - \hat{v}(s, \mathbf{w})\right]^2 \tag{3.3}$$

We now present the more general algorithm, ETD(λ) that attempts to minimize the above objective and allows for state-dependent discounting, bootstrapping and user-specified interest. The ETD(λ) algorithm can be fully specified with the following updates:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_{\mathbf{w}} \delta_t \mathbf{e}_t \tag{3.4}$$

$$\mathbf{e}_t \leftarrow \rho_t(\gamma_t \lambda_t \mathbf{e}_{t-1} + M_t \mathbf{x}(S_t)), \text{ with } \mathbf{e}_{-1} \doteq \mathbf{0} \tag{3.5}$$

$$F_t \leftarrow \rho_{t-1} \gamma_t F_{t-1} + i(S_t), \text{ with } F_{-1} \doteq 0 \tag{3.6}$$

$$M_t \leftarrow \lambda_t i(S_t) + (1 - \lambda_t) F_t \tag{3.7}$$

Above, γ_t and λ_t are generalizations of the discount factor and trace decay parameters to be functions of state: $\gamma_t \doteq \gamma(S_t)$ and $\lambda_t \doteq \lambda(S_t)$.

Specific to ETD(λ), when comparing with TD(λ), are the non-negative scalars, F_t and M_t. F_t is the followon trace that was discussed before. However, it now accumulates interest over time (instead of 1 at every timestep) while following an excursion away from the behaviour policy and accounting for termination while following the target policy. M_t is known as the emphasis in the update at timestep t. The emphasis reflects the interest we have in a state arising from the user-specified interest we have on the state as well as the accumulated interest we have on the state due to bootstrapping relationships which is calculated using the follow-on trace.

Thus, we have presented ETD, a simple single parameter set (weight vector) algorithm for off-policy policy evaluation. The ideas of interest and emphasis have also been introduced in this context.

3.2 Interest Specified with Information from the History

In this thesis, we will be comparing algorithms that use different specifications of interest. These interest functions are generally of the form $I_t = i(S_t) = f(\langle \mathbf{x}(S_0), A_0, \mathbf{x}(S_1), ..., \mathbf{x}(S_{t-1}), A_{t-1}, \mathbf{x}(S_t) \rangle)$. In other words, interest is specified as a function of the history of feature vectors and actions till the current timestep. In this section, we explain how interest functions using such information from the experience stream are consistent with the definition of interest: a function mapping from $\mathcal{S}$ to $[0, \infty)$ that serves as a weighting in the MSVE prediction objectve.

In the episodic setting, we are interested in maximizing the start-state value and can compare policies simply by looking at their start-state values. Therefore, a commonplace interest is to have an interest on the start-state alone. That is, an interest of 1 for the first timestep and interest of 0 for later timesteps. There is an apparent violation even in this simple choice of interest. Interest is a function of the timestep and no longer a function mapping from $\mathcal{S}$ to $[0, \infty)$.

Consider any given finite-state MDP. We can always transform the MDP so that the state now contains the timestep. In doing this transformation, state remains Markovian as all we have done in the transformation is to add additional information to a state that already contained all the information necessary for prediction and control. More importantly though, we can now define interest as a function of the timestep, and yet the interest remains a function mapping the (modified) state-space to the non-negative reals and serves as a weighting on the (modified) states in the prediction objective. Therefore, interest is consistently defined.

However, there are some issues that need to be resolved. Notably, the transformed MDP may have a countably infinite state-space due to the timestep being included as part of the state. For example, an MDP may probabilistically terminate, and hence the timestep may grow without bound. Prior proofs of stability and convergence of ETD relied on the MDP having finite state and action spaces. Therefore, we acknowledge that further investigation into the convergence of ETD is needed when we perform such a transformation and, more generally, when the state-space is infinite. Finally, the interest on the states of the original MDP requires investigation. In fact, an analysis into what the interest is on the states of the original MDP may be particularly useful to understand all of these issues.

We can also specify interest as a function of the history of feature vectors and actions up till the current timestep. The question of consistency in the definition of interest arises in this case as well. However, the arguments remain the same. We can transform any given MDP to include the history of previous feature vectors and actions. In other words, S_t is transformed to $\langle S_t, \mathbf{x}(S_0), A_0, \mathbf{x}(S_1), ..., \mathbf{x}(S_{t-1}), A_{t-1}, \mathbf{x}(S_t)\rangle$. Again, the states remain Markovian and interest remains a function from $\mathcal{S}$ to $[0, \infty)$ when specified with information from the history. However, the theoretical concerns about the state-space being countably infinite and the convergence of ETD remain, and we leave to future work an investigation into these issues.

As a first example of where we use a specification of interest that relies on the history, in the next section, we will show that TD(λ) and ETD(λ) are equivalent with a particular setting of interest for ETD which relies on information about the timestep. Later, we will consider algorithms that rely on more than just the timestep and the previous feature vector and action in particular.

3.3 Implicit Interest in TD(λ)

ETD(λ) and TD(λ) differ only in the use of the emphasis term M_t. We can alternatively view on-policy TD(λ) as ETD with an emphasis of $M_t = 1, \forall t$ with constant trace decay parameter $\lambda_t = \lambda$, constant discount factor $\gamma_t = \gamma$, and $\rho_t = 1, \forall t$.

A curious question that arises is then, what is the interest function that results in an emphasis of 1 at all timesteps. We can solve for this with some algebra based on the updates (3.4) - (3.7) for a few steps and then we observe a pattern.

For timestep 0, we have that $M_0 \doteq (1-\lambda)F_0 + \lambda I_0$. Since $F_{-1} = 0$, $F_0 \doteq \gamma F_{-1} + I_0 = \gamma \cdot 0 + I_0 = I_0$. To obtain $M_0 = 1$, we need $(1-\lambda)I_0 + \lambda I_0 = 1$. No matter the choice of λ, we thus have that $I_0 = 1$. Also, the followon $F_0 = I_0 = 1$.

Consider timestep 1, we have $M_1 \doteq (1-\lambda)F_1 + \lambda I_1$. Since $F_0 = 1$, $F_1 \doteq \gamma F_0 + I_1 = \gamma + I_1$. Again, to get $M_1 = 1$, we need $(1-\lambda)(\gamma + I_1) + \lambda I_1 = 1$. Simplifying, we get that $1 - \cancel{\lambda I_1} = \gamma + I_1 - \lambda\gamma - \cancel{\lambda I_1}$ or $I_1 = 1 - \gamma(1-\lambda)$. The followon, F_1, is thus $\gamma + I_1 = 1 + \gamma\lambda$.

Finally, consider timestep 2. If we apply the same steps, we get that $I_2 = 1 - \gamma(1-\lambda) - \gamma^2\lambda(1-\lambda)$ and $F_2 = 1 + \gamma\lambda + \gamma^2\lambda^2$.
Now, we can see a pattern in the interest and followon across timesteps. The implicit interest on states used by TD(λ) is given by:

$$I_t = 1 - \sum_{k=1}^{t} \gamma^k \lambda^{k-1}(1-\lambda)$$

The corresponding followon trace is given by:

$$F_t = \sum_{k=0}^{t} \gamma^k \lambda^k$$

We have thus shown for the first time that we can characterize the prediction problems that TD(λ) solves for as prediction problems with different time-dependent interest weightings. γ being part of the interest is of little issue as it is a parameter of the problem. However, that the solution parameter λ is also part of the interest perhaps raises further questions. The main point to keep in mind however is that, by choosing λ, we are choosing to care about different states differently. Note that TD does not use the follow-on weighting in the error it minimizes while ETD does use the follow-on weighting. Nevertheless, it is interesting to see what interest would result in TD and ETD being equivalent algorithms.

For our purposes, the implicit interest we are concerned with only includes the discount factor, γ. This is because we will be using one-step algorithms, TD(0) and ETD(0), in our experiments. Thus, we note an important conclusion about the interest function used by TD(0):

$$I_t = \begin{cases} 1 & \text{if } t = 0 \\ 1 - \gamma & \text{otherwise} \end{cases} \tag{3.8}$$

As can be seen above, the time-dependent interest on the start state is 1 and for later states is 1 - γ. The extreme cases are of particular importance. When $\gamma = 0$, the interest is 1 in all states, while when $\gamma = 1$, the interest is only on the start state.

Chapter 4

Incorporating Discounting in Actor Updates

In this thesis, we present how one might interrelate the prediction and control problems by starting with a control objective and presenting an algorithm that optimizes for this objective. Then, we seek for an interest based on the learning updates of this algorithm. Thus, we need to present this control algorithm before moving on to seek an interest and interrelate the prediction and control problems. This chapter can be largely read separately from the other chapters and is largely motivated by recent work talking about a subtle point in policy gradient algorithms when using discounting. I first present the policy gradient theorem for the discounted setting and the REINFORCE policy gradient algorithm (Williams, 1992). Then, I reiterate claims from recent work (Nota and Thomas, 2019) that there is a need for a γ^t factor on updates. Finally, with a simple yet illustrative example, I conclude that without the γ^t term, the actor updates can result in agents converging to poor policies.

4.1 Policy Gradient Theorem for the Discounted Setting

Suppose we have a parameterized policy $\pi(s, a, \boldsymbol{\theta})$. To solve the control problem, we would like an expression that tells us how to change the parameters of the policy to maximize expected return from a given state. The policy gradient theorem tells us exactly this. As an agent follows the direction suggested by the policy gradient, the policy improves if small enough steps are taken. The expected return from a state s under a policy π is simply the state-value of state s, $v_\pi(s)$. Thus, we would like an expression for $\nabla_{\boldsymbol{\theta}} v_\pi(s)$ with which we can make updates to $\boldsymbol{\theta}$.

In particular, for the episodic discounted control objective $J(\boldsymbol{\theta}) = v_\pi(s_0)$, the policy gradient theorem tells us that $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_s \mu(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|s) q_\pi(s,a)$.

For the first step of deriving the theorem, recall that the state-value function can be expressed as the expected action-value (2.3). Using this fact, we can continue expanding the expression by using the rules of differentiation and the Bellman equations. This derivation follows (Sutton and Barto, 2018) but crucially uses discounting with the terms related to γ shown in boxes with the notation $Pr(s \rightarrow x, k, \pi)$ denoting the probability of going from state s to state x in k steps while following π.

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} v_\pi(s) &= \nabla_{\boldsymbol{\theta}} \left[\sum_a \pi(a|s) q_\pi(s,a) \right] \\
&= \sum_a \left[\nabla_{\boldsymbol{\theta}} \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla_{\boldsymbol{\theta}} q_\pi(s,a) \right] \text{(Sum \& Product Rule of Differentiation)} \\
&= \sum_a \left[\nabla_{\boldsymbol{\theta}} \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla_{\boldsymbol{\theta}} \sum_{s',r} p(s',r|s,a)[r + \boxed{\gamma} v_\pi(s')] \right] \\
&= \sum_a \left[\nabla_{\boldsymbol{\theta}} \pi(a|s) q_\pi(s,a) \right] + \sum_a \left[\pi(a|s) \nabla_{\boldsymbol{\theta}} \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')] \right]
\end{aligned}$$

Define $\mathbf{g}(s) = \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|s) q_\pi(s,a)$. Then, we have:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} v_\pi(s) &= \mathbf{g}(s) + \sum_a \pi(a|s) \nabla_{\boldsymbol{\theta}} \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')] \\
&= \mathbf{g}(s) + \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \gamma \nabla_{\boldsymbol{\theta}} v_\pi(s') \text{(Derivative of Constant Terms is 0)} \\
&= \mathbf{g}(s) + \gamma \sum_a \sum_{s'} \pi(a|s) p(s'|s,a) \nabla_{\boldsymbol{\theta}} v_\pi(s') \\
&= \mathbf{g}(s) + \gamma \sum_{s'} Pr(s \rightarrow s', 1, \pi) \nabla_{\boldsymbol{\theta}} v_\pi(s')
\end{aligned}$$

Therefore, we now have that:

$$\nabla_{\boldsymbol{\theta}} v_\pi(s) = \mathbf{g}(s) + \gamma \sum_{s'} Pr(s \to s', 1, \pi) \nabla_{\boldsymbol{\theta}} v_\pi(s') \tag{4.1}$$

We can "unroll" once more by using Equation (4.1) in $\nabla_{\boldsymbol{\theta}} v_\pi(s')$ and then we see a pattern:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} v_\pi(s) &= \mathbf{g}(s) + \gamma \sum_{s'} Pr(s \to s', 1, \pi) \left[\mathbf{g}(s') + \gamma \sum_{a'} \sum_{s''} \pi(a'|s') p(s''|s', a') \nabla_{\boldsymbol{\theta}} v_\pi(s'') \right] \\
&= \mathbf{g}(s) + \gamma \sum_{s'} Pr(s \to s', 1, \pi) \mathbf{g}(s') + \gamma^2 \sum_{s''} Pr(s \to s'', 2, \pi) \nabla_{\boldsymbol{\theta}} v_\pi(s'') \\
&= \gamma^0 \sum_{s} Pr(s \to s, 0, \pi) \mathbf{g}(s) + \gamma^1 \sum_{s'} Pr(s \to s', 1, \pi) \mathbf{g}(s') + \dots \\
&= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \boxed{\gamma^k} Pr(s \to x, k, \pi) \mathbf{g}(x) \\
&= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \boxed{\gamma^k} Pr(s \to x, k, \pi) \sum_{a} \nabla_{\boldsymbol{\theta}} \pi(a|x) q_\pi(x, a)
\end{aligned}$$

Assuming the episodic start state formulation objective, we are looking to increase the start-state value. So, we use s_0 in the above equation:

$$\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} v_\pi(s_0) \\
&= \sum_{s \in \mathcal{S}} \sum_{k=0}^{\infty} \boxed{\gamma^k} Pr(s_0 \to s, k, \pi) \sum_{a} \nabla_{\boldsymbol{\theta}} \pi(a|s) q_\pi(s, a)
\end{aligned} \tag{4.2}$$

To proceed, we need to define the on-policy state distribution in episodic settings. First, we define η as the expected number of visits to a state in an episode:

$$\eta(s) = h(s) + \boxed{\gamma} \sum_{\bar{s}} \eta(\bar{s}) \sum_{a} \pi(a|\bar{s}) p(s|\bar{s}, a), \forall s \in \mathcal{S} \tag{4.3}$$

Time is spent in a state s if an episode starts from it (which occurs with probability $h(s)$) or if we enter state s from another state $\bar{s}$. When we use discounting, we treat it as a form of termination and use a factor of γ in the second term.

The on-policy state distribution is then simply the fraction of time spent in each state normalized to sum to one:

$$\mu(s) = \frac{\eta(s)}{\sum_{s'} \eta(s')}, \forall s \in \mathcal{S} \tag{4.4}$$

Now, let us go back to the objective. Since $\eta(s) = \sum_{k=0}^{\infty} \gamma^k Pr(s_0 \rightarrow s, k, \pi)$, we have that:

$$\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} v_\pi(s_0) \\
&= \sum_s \eta(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|s) q_\pi(s,a) \\
&= \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|s) q_\pi(s,a) \\
&= \sum_{s'} \eta(s') \sum_s \mu(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|s) q_\pi(s,a) \\
&\propto \sum_s \mu(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|s) q_\pi(s,a) \qquad (4.5)
\end{aligned}$$

The gradient of our objective being the final expression is the policy gradient theorem which we claimed and have now shown. There are a few subtleties. The constant of proportionality above is the average length of an episode. Recall that when discounting is used, it is treated as a probability of termination in defining η. Thus, the average length should change accordingly.

4.2 The REINFORCE Algorithm

Now, we have established the policy gradient theorem for the discounted setting. We look at how we move on to a practical algorithm called REINFORCE (Williams, 1992).

$$\begin{aligned}
\nabla J(\boldsymbol{\theta}) &\propto \sum_s \mu(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi(a|s) q_\pi(s,a) \\
&= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t, a) \nabla_{\boldsymbol{\theta}} \pi(a|S_t, \boldsymbol{\theta}) \right] \\
&= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t, a) \pi(a|S_t, \boldsymbol{\theta}) \frac{\nabla_{\boldsymbol{\theta}} \pi(a|S_t, \boldsymbol{\theta})}{\pi(a|S_t, \boldsymbol{\theta})} \right] \\
&= \mathbb{E}_\pi \left[q_\pi(S_t, A_t) \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t, \boldsymbol{\theta})}{\pi(A_t|S_t, \boldsymbol{\theta})} \right] \\
&= \mathbb{E}_\pi \left[G_t \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t, \boldsymbol{\theta}) \right]
\end{aligned}$$

Going from the penultimate step to the final expression is possible because we can use a sample of q_π and still maintain the expectation (formally, this would be explained noting that G_t is an unbiased estimate of q_π meaning that they have the same expectation).

Noting the final expression above, the following REINFORCE update is often used:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} G_t \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t, \boldsymbol{\theta}) \tag{4.6}$$

However, this ignores the fact that γ is only used in the multiplicative factor on rewards when defining the return. We do not actually terminate episodes based on the discount factor. In other words, the agent observes states according to the undiscounted state distribution. Thus, we can instead use the definition of η without discounting but then we need to introduce a γ^t factor:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} \gamma^t G_t \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t, \boldsymbol{\theta}) \tag{4.7}$$

We now have two different algorithms based on the two different actor updates. Before we present empirical results showing how the two compare, move on to discuss what the algorithm without the γ^t term is doing, how it might be problematic and how we might interpret the γ^t term.

4.3 Empirical Demonstration for the Need of a γ^t term in Actor Updates

Note that the difference between the updates reduces as $\gamma \to 1$ because γ^t for $\gamma \approx 1$ is near equivalent to using a uniform weighting of 1 on the updates. Thus, the biggest difference between the two updates is when $\gamma = 0$.

We now design an MDP to highlight the difference between the two algorithms. A particular property we are looking for is that the meaning of actions in the first few states is different from the meaning of the same actions many states later.

To simplify MDP design, we use just 2 states and share the feature parameterization for both so that the states appear identical to the agent. Action Right is good in the start state but is bad in the second state. On the other hand, action Left is bad in the first state but good in the second. With $\gamma = 0$, the agent should be myopic and take action right. The full environment specification is given in Figure 4.1.

Now, we compare the two agents: one that does not use a γ^t term and the REINFORCE algorithm that uses the theoretically suggested γ^t term.

The learning curve for actor step-size parameter 0.1 is given in Figure 4.2 with similar results across many different choices for the step-size parameter.

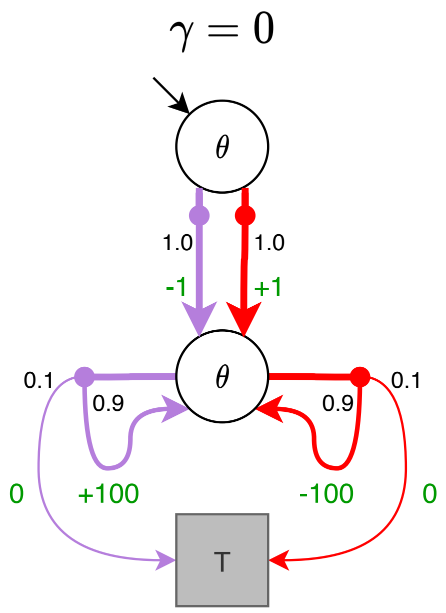


Figure 4.1: Environment illustrating need for γ^t term in actor updates. States appear identical to the agent and hence the agent is forced to take the same policy in both states. There are two actions, Left (lavender coloured transitions) and Right (red coloured transitions). Action Right is good in the starting state while the action Left is good in second state. Since $\gamma = 0$, the horizon is a single timestep and the optimal policy is to take right action.

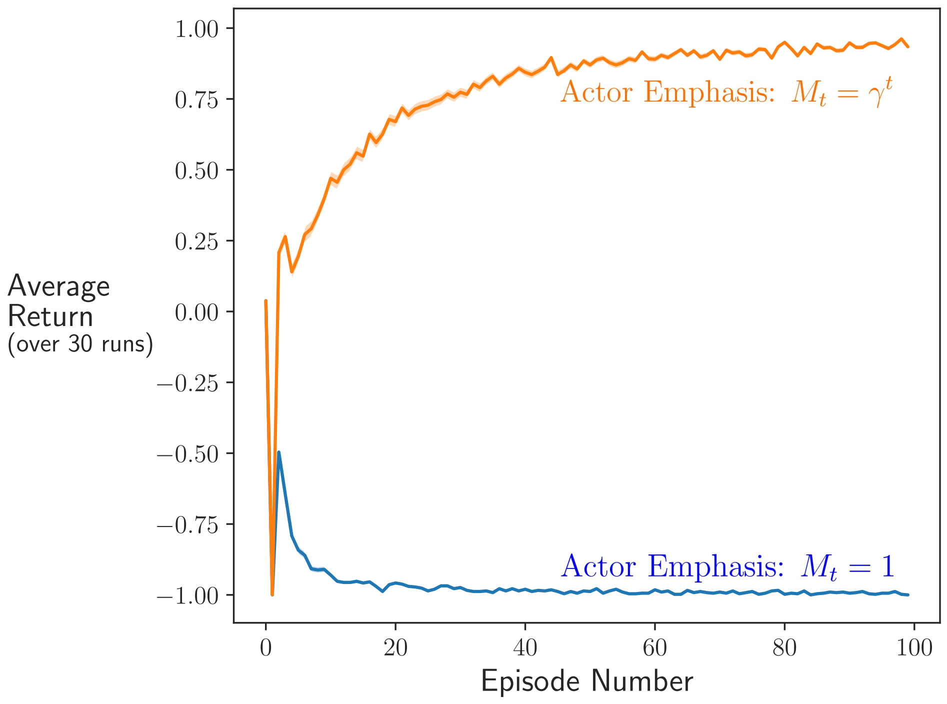


Figure 4.2: Learning curves comparing the algorithm that uses a γ^t term with one that does not use it. Action step-size parameter: $\alpha_{\boldsymbol{\theta}} = 0.1$. We see that the algorithm using a γ^t term converges to a good policy. However, the algorithm that does not use the γ^t term converges to a poor policy.

We see that the agent with the γ^t term does better. It is easy to see why this is the case. When $\gamma = 0$, no updates to the actor take place after the first timestep when using a γ^t term as $0^0 = 1$ but $0^t = 0, \forall t > 0$. However, the algorithm without the γ^t term updates the actor on every timestep.

Since $\gamma = 0$, we only care about the immediate reward from the start state. Thus, naturally, the algorithm with the γ^t term which only updates at the first timestep converges towards to the best policy of always taking the Right action. On the other hand, without the γ^t term, the agent updates on all transitions and the multiple large positive rewards from taking action Left in the second state reinforces taking action Left. This leads to convergence towards a poor policy of always taking action Left since the states appear identical to the agent.

Chapter 5

Seeking Interest

Till now, we largely discussed algorithms for solving the prediction and control problems separately (ETD and REINFORCE respectively) and have only briefly touched upon actor-critic algorithms. In this chapter, I introduce the actor-critic algorithm as an extension to REINFORCE with a component called the critic that solves the prediction problem. Then, I attempt to find a prediction objective motivated by the learning updates of the actor in REINFORCE. I first discuss how the addition of a critic affects control performance in terms of bias and variance of the policy-gradient estimator. Then, the crux of this thesis follows. I seek for and find the first suggestion of an interest over states for the prediction subproblem given a control problem and hence interrelate the prediction and control objectives.

5.1 Introducing Baselines

It is well known that without changing the expected update of the REINFORCE algorithm, any state-dependent function called a baseline can be subtracted from the sample return. We include the proof here for completeness. First, consider the REINFORCE with a baseline update:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha_{\boldsymbol{\theta}} \gamma^t \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t, \boldsymbol{\theta}_t)[G_t - b(S_t)] \tag{5.1}$$

The expected update for REINFORCE is:

$$\mathbb{E}_\pi[\nabla_{\boldsymbol{\theta}} \ln \pi(a|s, \boldsymbol{\theta}) q_\pi(s, a)]$$

The expected update for REINFORCE with a baseline is instead:

$$\mathbb{E}_\pi[\nabla_{\boldsymbol{\theta}} \ln \pi(a|s,\boldsymbol{\theta})[q_\pi(s,a) - b(s)]]$$

By linearity of expectations, we can rewrite the expected update of REINFORCE with a baseline as two terms:

$$\mathbb{E}_\pi[\nabla_{\boldsymbol{\theta}} \ln \pi(a|s,\boldsymbol{\theta}) q_\pi(s,a)] - \mathbb{E}_\pi[\nabla_{\boldsymbol{\theta}} \ln \pi(a|s,\boldsymbol{\theta}) b(s)]$$

Let us define the random variables $A_t = \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t,\boldsymbol{\theta}_t)[G_t]$ and $B_t = \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t,\boldsymbol{\theta}_t)[G_t - b(S_t)]$. Then, to show that introducing a baseline does not introduce any bias, we simply need to show that $\mathbb{E}[A_t] = \mathbb{E}[B_t]$. From looking at the two terms, we can conclude the use of a baseline will not add any bias if the second term is 0. Using $\nabla_{\boldsymbol{\theta}} \ln \pi(a|s,\boldsymbol{\theta}) = \frac{\nabla_{\boldsymbol{\theta}}\pi(a|s,\boldsymbol{\theta})}{\pi(a|s,\boldsymbol{\theta})}$, the sum rule of differentiation, $\sum_a \pi(a|s,\boldsymbol{\theta}) = 1$ and that $\frac{d}{dx}c = 0$, we have:

$$\begin{aligned}
\mathbb{E}_\pi[\nabla_{\boldsymbol{\theta}} \ln \pi(a|s,\boldsymbol{\theta}) b(s)] &= \sum_s d_\pi^\gamma(s) \sum_a \pi(a|s,\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \ln \pi(a|s,\boldsymbol{\theta}) b(s) \\
&= \sum_s d_\pi^\gamma(s) \sum_a \cancel{\pi(a|s,\boldsymbol{\theta})} \frac{\nabla_{\boldsymbol{\theta}}\pi(a|s,\boldsymbol{\theta})}{\cancel{\pi(a|s,\boldsymbol{\theta})}} b(s) \\
&= \sum_s d_\pi^\gamma(s) b(s) \sum_a \nabla_{\boldsymbol{\theta}}\pi(a|s,\boldsymbol{\theta}) \\
&= \sum_s d_\pi^\gamma(s) b(s) \nabla_{\boldsymbol{\theta}} \sum_a \pi(a|s,\boldsymbol{\theta}) \\
&= \sum_s d_\pi^\gamma(s) b(s) \nabla_{\boldsymbol{\theta}} 1 \\
&= 0
\end{aligned}$$

Thus, we have shown that introducing a baseline does not introduce any bias. The addition of a baseline can however affect the variance of the gradient estimator. So, the choice baseline b can affect $\mathbb{E}[B_t - \mathbb{E}_\pi[B_t]]^2$. A common choice for the baseline is the state-value function $\hat{v}(s,\mathbf{w})$:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha_{\boldsymbol{\theta}} \gamma^t \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t,\boldsymbol{\theta}_t)[G_t - \hat{v}(S_t,\mathbf{w}_t)] \tag{5.2}$$

$\hat{v}(s,\mathbf{w})$ can be learned with any prediction algorithm. However, the optimal baseline is in fact not the state-value and is instead the following (Wu et al., 2018):

$$b^*(S_t) = \frac{\mathbb{E}\left[\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(A_t|S_t,\boldsymbol{\theta}_t)^T \nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(A_t|S_t,\boldsymbol{\theta}_t) \hat{q}(S_t,A_t,\mathbf{w}_t)\right]}{\mathbb{E}[\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(A_t|S_t,\boldsymbol{\theta}_t)^T \nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(A_t|S_t,\boldsymbol{\theta}_t)]} \tag{5.3}$$

We do not discuss variance reduction further, but the important takeaway from this discussion about baselines is that learning a state-value and using it as a baseline only does not introduce any bias.

5.2 Actor-Critic Methods

Note that, even with a baseline, we must wait until the end of the episode before making updates as the sample returns G_t are available only then.

Instead of using G_t, we can use bootstrapping and perform updates much sooner. Any target which in expectation is $q_\pi(s,a)$ can be used. Equivalently, we can use any estimate of the advantage which is defined as $q_\pi(s,a) - v_\pi(s)$. In this thesis, we focus on the one-step TD target and the one-step TD error estimate of the advantage in our theoretical analysis and experiments. Using the one-step TD target, we get the following actor update for (TD) actor-critic:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \underbrace{\alpha\gamma^t \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t, \boldsymbol{\theta}_t)[R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t)]}_{\Delta\boldsymbol{\theta}_{\text{AC}}} \tag{5.4}$$

If we use an approximate value function as above (instead of just as a baseline), we call the component maintaining the value function the critic. A common choice of prediction algorithm for the critic is TD(0). The main question we will go on to discuss is what prediction algorithm and objective we should use for the critic.

5.3 Analysis of the Bias in Actor-Critic methods

By replacing the use of estimation for the value of state S_{t+1}, we get the following unbiased update:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \underbrace{\alpha\gamma^t \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t, \boldsymbol{\theta}_t)[R_{t+1} + \gamma v_\pi(S_{t+1}) - \hat{v}(S_t, \mathbf{w}_t)]}_{\Delta\boldsymbol{\theta}_{\text{Unbiased}}} \tag{5.5}$$

Let us consider the difference between $\Delta\boldsymbol{\theta}_{\text{Unbiased}}$ and $\Delta\boldsymbol{\theta}_{\text{AC}}$. We call this error vector $\boldsymbol{\epsilon}$ and define it as follows:

$$\boldsymbol{\epsilon}_t \doteq \alpha\gamma^t \nabla_\theta \ln \pi(A_t|S_t, \boldsymbol{\theta}_t)\,[\gamma v_\pi(S_{t+1}) - \gamma \hat{v}(S_{t+1}, \mathbf{w}_t)] \tag{5.6}$$

Therefore, we now have an error vector that expresses the bias introduced due to our use function approximation and bootstrapping.

5.4 An Interest to Reduce Bias

If we use biased updates, we are not guaranteed to follow the policy gradient and improve the policy. We can improve control performance by reducing the bias of the gradient estimator. So, to get an algorithm that does a better job of reducing the bias, we now think about how we might change the critic parameters so that $\boldsymbol{\epsilon}$ is reduced on average across all timesteps and all states.

The key insight now is that we can do so by using ETD for the critic and choosing a particular interest function. To get a scalar interest, expand the squared norm of the of the error vector. In particular, note that:

$$\begin{aligned}
\|\boldsymbol{\epsilon}_t\|_2^2 &= \|\alpha\gamma^t\nabla_\theta \ln \pi(A_t|S_t,\boldsymbol{\theta}_t)[\gamma v_\pi(S_{t+1}) - \gamma\hat{v}(S_{t+1},\mathbf{w}_t)]\|_2^2 \\
&= \|\alpha\gamma^{t+1}\nabla_\theta \ln \pi(A_t|S_t,\boldsymbol{\theta}_t)[v_\pi(S_{t+1}) - \hat{v}(S_{t+1},\mathbf{w}_t)]\|_2^2 \\
&= \alpha^2\gamma^{2(t+1)}[v_\pi(S_{t+1}) - \hat{v}(S_{t+1},\mathbf{w}_t)]^2\|\nabla_\theta \ln \pi(A_t|S_t,\boldsymbol{\theta}_t)\|_2^2
\end{aligned}$$

The penultimate step takes a common γ factor out. The last step above is due to the absolute homogeneity property of norms which tells us that we can move scalars out of the norm with an absolute sign on them ($\|c\mathbf{v}\| = |c|\|\mathbf{v}\|$). All the terms are squared, so they are non-negative and hence the absolute value is redundant.

Now, let the squared norm of the vector $\boldsymbol{\epsilon}_t$ be the scalar ϵ_t. Transitions into state S_{t+1} occur from may occur from different states taking different actions. In general, the transition to S_{t+1} may also occur at different timesteps. Letting the random variable $S_{t+1} = s$, if we accumulate all such errors, we can calculate the total error in using approximation for the value of state s. Since we care not only about the value function approximation error for the state s, but across all states, we also have to sum over all states. A final expression for the total bias introduced to the policy gradient estimator due to the use of function approximation along all possible transitions and all states at all timesteps is given by:

$$\sum_{s}\sum_{\bar{s}}\sum_{k=0}^{\infty} Pr(s_0 \to \bar{s}, k, \pi) \sum_{a} \pi(a|\bar{s})p(s|\bar{s}, a)[\gamma^{2(k+1)}\|\nabla_{\boldsymbol{\theta}} \ln \pi(a|\bar{s}, \boldsymbol{\theta})\|_2^2(v_\pi(s) - \hat{v}(s, \mathbf{w}))^2]$$

Now, recall the definitions of η and the on-policy distribution μ (Eqns. 4.3 and 4.4) and the manipulations made to get Eqn. 4.5. Note for example that $\eta(\bar{s}) = \sum_{k=0}^{\infty} Pr(s_0 \to \bar{s}, k, \pi)$ when we assume the undiscounted state visitation which is how states are observed in an episodic setting. The next summation over actions of π and p together can be combined to get $\eta(s)$. Then, by multiplying and dividing by $\sum_{x} \eta(x)$, we complete the manipulations to see that the above expression is an expectation with the on-policy distribution. Moreover, recall that the interest weighted mean squared value error objective is of the form $\sum_{s} \mu(s)i(s)[v_\pi(s) - \hat{v}(s, \mathbf{w})]^2$. Thus, we have that the above expression is an interest weighted mean squared value error objective.

The interest is the term beside the on-policy distribution weighting and the squared value error term. We can consider just the error from the one transition we see online, and we thus have that the interest is:

$$i(S_t) = \gamma^{2t}\|\nabla_{\boldsymbol{\theta}} \ln \pi(A_{t-1}|S_{t-1}, \boldsymbol{\theta})\|_2^2 \tag{5.7}$$

Note that the interest is actually however the average of the above right hand side expression across all timesteps and all transitions leading to the state S_t. Also, note that the step-size that was present before is not included. Suppose we have an interest function $i(s)$ across a state space $\mathcal{S}$. Any constant multiplier can be factored out such that a different step-size would give the same interest function. So, any interest function of the form $i'(s) = i(s) \cdot m$ for some constant m can be considered to be the same and we can ignore the multiplier m. The other modification used to get the final interest is a simple shift of timestep to express the interest we have on the state at time $t + 1$ to the interest we have on the state observed at time t.

There are two terms within the interest above. One is an exponential decay similar to the γ^t term in actor updates. The other term is the norm of the characteristic eligibility at the previous timestep.

Consider how we might interpret the interest as a whole, under averaging. The interest is high on any state for which, on average, a small change in the policy parameters leads to a large positive change in the probabilities of taking actions that lead to this state. In other words, if a small change in the policy parameters is sufficient to reach this state more, we should care about it more. This is quite intuitive. The additional modulation by discounting can be viewed as a message from the actor about the γ^t weighting used in its updates.

In the next chapter, we will discuss many algorithms that are motivated by the interest we found above.

Chapter 6

Algorithms for the Critic

The goal of this thesis has been to attempt to interrelate the prediction and control objectives of RL by finding an interest and hence a prediction objective for a given control objective. We found an interest in the previous chapter motivated by reducing bias in actor updates of episodic actor-critic. Now, we want to show that using prediction algorithms that optimize for this novel objective within the actor-critic setup leads to improved control performance. Before we can do this however, we need to discuss the different algorithms we will compare in the empirical study in the next chapter. These algorithms are inspired from the interest we found and the γ^t term in actor updates. While we refer to these algorithms by the interest or emphasis used in prediction, the algorithms all use the actor-critic setup with an ETD(0) critic and differ only in the setting of the interest.

6.1 Algorithms Motivated by Discounting-Dependent Decay Terms

In the previous chapter, the interest we found (Eqn. 5.7) included a discounting-dependent decay term of γ^{2t}. Also, earlier we showed that an unbiased actor update uses a γ^t term. Finally, TD uses an emphasis of 1 at all timesteps. Therefore, the first set of algorithms we choose to compare are all combinations of using 1, γ^t and γ^{2t} as the interest and emphasis. This results in six algorithms.

Before discussing the algorithms, we note that we use discount factor terms as is without incorporating time in the state. Future work can consider the effects of doing this and try incorporating time into state and see how learning is affected.

Now, we get back to discussing the six algorithms. A brief summary of the algorithms with their interests and emphases are given in Table 6.1. Also, plots of the emphasis for the different algorithm with different settings of γ are shown in Figure 6.1.

Name	I_t	M_t	$\max_t M_t$	$\lim_{t\to\infty} M_t$
TD	$\begin{cases}1 & t=0\\ 1-\gamma & t>0\end{cases}$	1	-	1
$M_t = \gamma^t$	$\begin{cases}1 & t=0\\ 0 & t>0\end{cases}$	γ^t	0	0
$M_t = \gamma^{2t}$	$\begin{cases}1 & t=0\\ \gamma^{2t}-\gamma^{2t-1} & t>0\end{cases}$	γ^{2t}	0	0
$I_t = 1$	1	$\frac{\gamma^{t+1}-1}{\gamma-1}$	∞	$\frac{1}{1-\gamma}$
$I_t = \gamma^t$	γ^t	$(t+1)\gamma^t$	$\begin{cases}\frac{-1-\ln\gamma}{\ln\gamma} & \gamma \ge \frac{1}{e}\\ 0 & \gamma < \frac{1}{e}\end{cases}$	0
$I_t = \gamma^{2t}$	γ^{2t}	$\gamma^t\frac{\gamma^{t+1}-1}{\gamma-1}$	$\begin{cases}\frac{-\ln 2-\ln\gamma}{\ln\gamma} & \gamma \ge \frac{1}{2}\\ 0 & \gamma < \frac{1}{2}\end{cases}$	0

Table 6.1: Summary of algorithms motivated by discounting-dependent decay terms with their interests and emphases.

In the first column of the table I have listed the name by which we refer to the algorithm in the plots and discussions. Note again that the algorithms we use in our experiments will be control algorithms using the actor-critic setup with ETD(0) used in the critic. The only difference between the algorithms we compare is the setting of the interest over states for prediction. The second column contains the interest over states used by the algorithm. The third column contains the resulting emphasis. The fourth and fifth columns list some properties about the emphasis. For the algorithm named $I_t = 1$ there is no particular timestep which has the maximum emphasis (as emphasis grows forever) and it is hence omitted. Now, let us discuss the algorithms in more

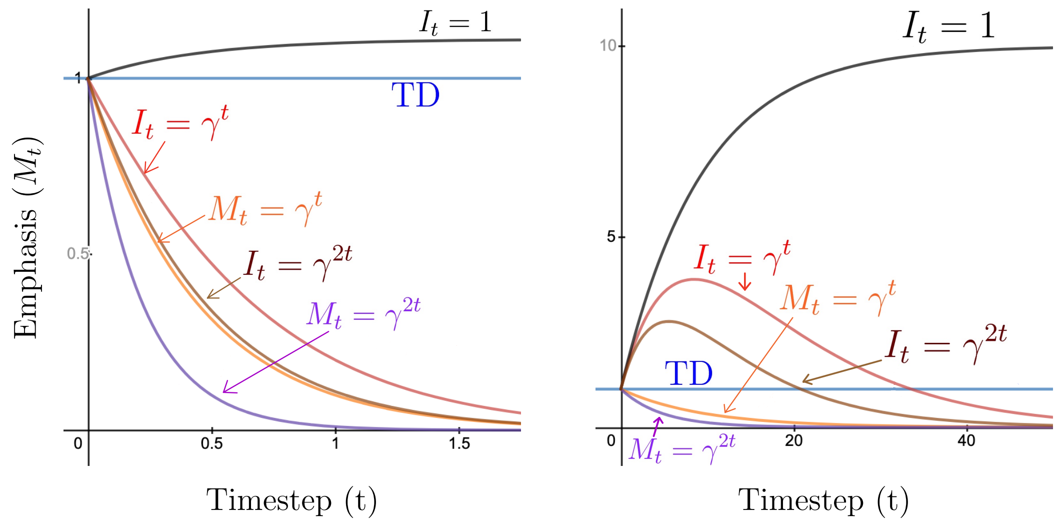


Figure 6.1: Visualization of the emphases in critic updates for algorithms differing in how discounting is incorporated in the interest for two discount factors, $\gamma = 0.1$ on the left and $\gamma = 0.9$ on the right.

detail by row in the table.

The first algorithm, TD, uses a uniform emphasis of 1 at all timesteps. As we showed for TD, the interest is given by (3.8).

Using the second algorithm, $M_t = \gamma^t$ results in the actor and critic having the same emphasis of γ^t. Note that the interest is such that we are only interested in accurately predicting the start-state's value. As can be seen from the plots as well, the emphasis decays to 0 asymptotically and the maximum emphasis of 1 is at timestep 0.

Next, the third algorithm, $M_t = \gamma^{2t}$ uses γ^{2t} term as the emphasis instead of the interest. The properties of the emphasis used by the algorithm are very similar to that of the algorithm using $M_t = \gamma^t$. However, the decay is simply faster.

The fourth algorithm, $I_t = 1$, uses an interest of 1 at all timesteps instead of an emphasis of 1 at all timesteps like TD. The resulting emphasis grows forever but converges asymptotically to the effective horizon, $\frac{1}{1-\gamma}$.

Next, we have $I_t = \gamma^t$ which uses an interest of γ^t which is same as the term used in actor updates. The resulting emphasis is such that, below a threshold discount factor of $\frac{1}{e}$, the emphasis only decays. However, above this threshold, the emphasis grows for a while before it eventually decays toward 0. This can be seen in the plots with the line labeled $I_t = \gamma^t$. The emphasis only decays for the case where $\gamma = 0.1$ but rises and then falls for the case where $\gamma = 0.9$.

Finally, we have the algorithm that is closest to an algorithm motivated by our analysis into what we should set the interest as. Using $I_t = \gamma^{2t}$ means that we use one of the two terms of the interest from the previous chapter. This algorithm behaves similarly to the algorithm using $I_t = \gamma^t$ but the threshold where transition occurs from only decaying emphasis to a rise and then fall is $\frac{1}{2}$.

6.2 Algorithms Motivated by the Characteristic Eligibility Term

Now, we can discuss the more complex term of the interest we found. Stated in full terms, the second term of the interest is the squared norm of the characteristic eligibility at the previous timestep. Unlike depending on just the timestep, the interest involving a characteristic eligibility means that the interest depends on the previous state and action as well. Agents can choose to incorporate information about prior states and actions when constructing state. However, learning state is not the focus of this thesis. We do not do such state construction and use the interests as is. For the characteristic eligibility term, we consider two simple alternatives to incorporating information about previous state and action into state resulting in two interest functions and two corresponding algorithms.

The simpler of the two alternatives uses the norm of the characteristic eligibility seen online. We call this interest Online Norm of the Characteristic Eligibility (ONCE) and specifically call the actor-critic algorithm using this interest in the critic, ONCE-AC. Depending on the trajectory, ONCE-AC may assign different interests to a state based on what the previous state and action were.

The more complex of the two alternatives maintains an additional set of weights, η, and estimates the mean norm of the characteristic eligibility. We call this interest Mean Norm of the Characteristic Eligibility (MINCE) where the 'I' is added for easier pronunciation. Similarly, we refer to the actor-critic algorithm using ETD with MINCE as the interest as MINCE-AC. Note that this algorithm uses an interest closest to the one we found as it approximates the average of the two terms in the interest across timesteps and transitions from different states under different actions leading to the state.

Table 6.2 contains information about these algorithms. Note that information about the emphasis is omitted as there is no simple closed form expression for the emphasis. Also, note that we always use the discounting-dependent decay term γ^{2t} in these algorithms. The γ^{2t} term is motivated by the interest we found but we can largely ignore it as we only focus on the $\gamma = 1$ case in experiments when analyzing the algorithms. Moreover, considering all combinations (such as those with a γ^t or 1 decay) would be computationally expensive and may obfuscate from the central idea of this thesis that using the interest we found in the previous chapter in the prediction objective can improve control performance.

Name	I_t
ONCE	$\begin{cases} 1 & t = 0 \\ \gamma^{2t} \lVert \nabla_{\boldsymbol{\theta}} \ln \pi(A_{t-1} \vert S_{t-1}, \boldsymbol{\theta}) \rVert_2^2 & t > 0 \end{cases}$
MINCE	$\eta_t^T \mathbf{x}(S_t)$

Table 6.2: Summary of algorithms motivated by the characteristic eligibility term with their interests.

Since MINCE-AC is more complex and involves an additional set of parameters, we provide the pseudocode for this algorithm in the box titled Algorithm 2: MINCE-AC.

Algorithm 2: MINCE-AC

```
Input: a differentiable state-value function $\hat{v}(s, \mathbf{w})$
Input: a differentiable policy $\hat{\pi}(s, a, \boldsymbol{\theta})$
Input: a differentiable mean norm of the previous character
  eligibility estimator mince$(s, \eta)$
Parameters: step-sizes $\alpha_{\mathbf{w}}, \alpha_{\boldsymbol{\theta}}, \alpha_\eta > 0$
Initialize $\mathbf{w} \in \mathbb{R}^d$, $\boldsymbol{\theta} \in \mathbb{R}^{d \times |\mathcal{A}|}$, $\eta \in \mathbb{R}^d$
foreach episode do
    $F \leftarrow 0$
    $I \leftarrow \mathbf{x}(S_0)^T \eta$
    foreach timestep t in episode do
        $F \leftarrow \gamma F + I$
        $\delta_t \leftarrow R_{t+1} + \gamma \mathbf{x}(S_{t+1})^T \mathbf{w} - \mathbf{x}(S_t)^T \mathbf{w}$
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha_{\mathbf{w}} F \delta_t \mathbf{x}(S_t)$
        $c \leftarrow \nabla_{\boldsymbol{\theta}} \ln \pi(A_t | S_t, \boldsymbol{\theta})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} c \delta_t$
        $\eta \leftarrow \eta - \alpha_\eta \nabla_\eta [\gamma^{2t+1} c - \mathbf{x}(S_{t+1})^T \eta]^2$
        $I \leftarrow \mathbf{x}(S_{t+1})^T \eta$
    end foreach
end foreach
```

The main takeaway from the pseudocode is that MINCE-AC uses mean squared error loss to move an estimate of the average norm of the previous characteristic eligibility towards the sample we see online.

6.3 Sliding-Step Algorithms

Finally, we note a small modification to certain algorithms that we try. Sliding-step prediction algorithms were introduced as algorithms that bound the size of updates (Tian, 2018) while maintaining convergence.

Consider ETD first. The updates of ETD can have large variance as a result of the follow-on trace and important sampling ratios being too large. Sliding-step ETD was introduced as an algorithm that bounds the size of updates (Tian, 2018):

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \frac{1 - \exp(-\alpha F_t \rho_t \mathbf{x}_t^T \mathbf{x}_t)}{\mathbf{x}_t^T \mathbf{x}_t} \delta_t \mathbf{x}_t \tag{6.1}$$

For our experiments, we focus on the on-policy setting where $\rho_t = 1$. However, F_t may be large and hence ETD may have large updates resulting in wild fluctuations and divergence. With Sliding-step ETD, when the term inside the exponent is large, the effective step-size is $\frac{1}{\mathbf{x}_t^T\mathbf{x}_t}$. So, very large F_t would only move the estimates to the target in one-step. Thus, there will never be over-shooting beyond the target and the updates are bounded to move the estimates some small step towards the target.

Similarly, sliding-step TD was introduced as an algorithm that bounds the size of updates of TD (Tian, 2018):

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \frac{1 - \exp(-\alpha \mathbf{x}_t^T \mathbf{x}_t)}{\mathbf{x}_t^T \mathbf{x}_t} \delta_t \mathbf{x}_t \tag{6.2}$$

Sliding-step algorithms generally perform well across a wide range of step-size parameter. Moreover, one benefit of sliding-step TD shown was that sliding-step TD learns faster than TD when the magnitude of the feature vectors $\mathbf{x}(s)^T\mathbf{x}(s)$ varied significantly.

At this point, we note that, for conciseness, we sometimes refer to sliding-step TD as SSTD and sliding-step ETD with a MINCE interest weighting as SSMINCE.

Chapter 7

Experiments

The aim of this chapter is to empirically compare different variations of actor-critic algorithms that differ only in their choice of prediction objective. In particular, we will be comparing the control performance of different algorithms that all use Emphatic TD for the critic but differ in their settings of the interest function. The main takeaway from this chapter will be that, indeed, using both the discounting-dependent decay and the characteristic eligibility terms improve control performance and that the actor-critic algorithm using a sliding-step ETD critic with the MINCE interest weighting over states (which we refer to as SSMINCE) gives the best control performance across a wide range of settings for the algorithm parameters.

7.1 Effect of the Choice of Prediction Objective when $\gamma = 0$

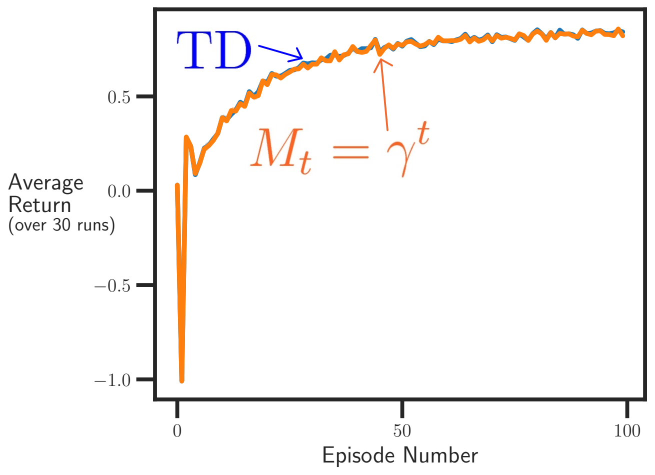
Consider the empirical experiment we conducted for studying the γ^t term in actor updates, i.e., the environment in Figure 4.1. There, we used an actor only algorithm, REINFORCE. We can rerun the experiment with an actor-critic agent.

Let us hypothesize what value function estimate the critic might learn when the emphasis is 1 at all timesteps (as in TD), or, equivalently, when the interest is as defined in Eqn. 3.8. Recall that there are two states that appear identical to the agent and that the first state is only visited once while the second state can be visited multiple times. Since the agent spends more time in the second state, the value function estimate would move towards the true value of the second state.

Consider another natural choice of emphasis for the critic updates. We can use γ^t which arose as a term in actor updates. Equivalently, this is the case for which we have interest only on the start state. Since $\gamma = 0$, updates to the critic will only be done in the first state and the value function estimate would move towards the true value of the first state.

Since γ is zero, the TD error is given by $R_{t+1} - \hat{v}(S_t, \mathbf{w}_t)$ instead of $R_{t+1} + \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t)$. This means that we only really use the value function approximation as a baseline. Thus, the value function approximation does not introduce any bias and we would expect the prediction subproblem to not affect the policy learned by the actor.

Given the above hypothesis that we expect the value functions learned to be different but the policy the same, we compare the algorithm using an emphasis of 1 at all timesteps, i.e., the common choice of emphasis, with the algorithm using γ^t emphasis.



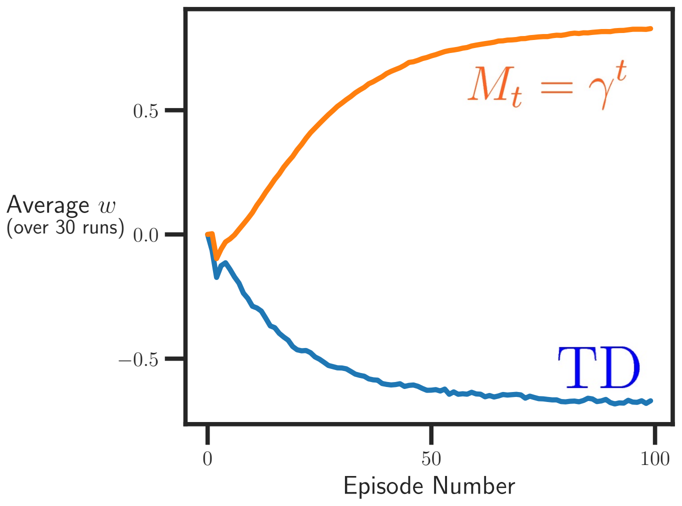


Figure 7.1: Illustration that, when $\gamma = 0$, the choice of critic objective affects the value function while the policy is unaffected. We compare two actor-critic agents differing only in their choice of critic interest in the ETD critic: 1 at all timesteps (leading to $M_t = 1$ and equivalent to TD) vs. 1 only on the start-state (when $M_t = \gamma^t$). The same policies are learned by both the algorithms (Top diagram). However, the value functions learned are drastically different (Bottom Diagram).

As can be seen above, we learn the same near-optimal policy with both algorithms. However, the algorithm using an emphasis of 1 at all timesteps in the critic updates learns a perhaps philosophically strange value estimate. Even though the policy is consistently getting a return of 1 upon convergence, the value function estimate is negative and closer to the value of the second state if γ were bigger than 0.

This experiment sheds light on an obvious but insightful conclusion. When $\gamma = 0$, control performance is unaffected by the choice of prediction objective. This is obvious because there is no need for secondary reinforcers when $\gamma = 0$, the immediate reward suffices for learning. On the other hand however, this means that the choice of prediction objective is likely more important when γ is large. This is in some sense opposite to what we saw with the actor. The bias introduced in actor updates by omitting the γ^t term reduces as $\gamma \to 1$ and increases as $\gamma \to 0$.

7.2 MDP Experiments

In this section, we compare the control performance of the actor-critic algorithms on two simple MDPs in which all states appear the same. We will be investigating which of the six variations of using 1, γ^t or γ^{2t} each as the interest or emphasis gives the best control performance. We leave considering the characteristic eligibility term to the next section where we will consider more complex environments.

First, we test the algorithms on simple environment similar to the one presented in (Nota and Thomas, 2019). The discount factor is small but non-zero $\gamma = 0.1$. As we showed in the previous section, using a discount factor of exactly 0 will result in the choice of prediction objective not having any influence on control performance. But, now, we expect the choice of prediction objective to have some influence, but not as large an influence as when γ is much higher. The MDP is given in Figure 7.2.

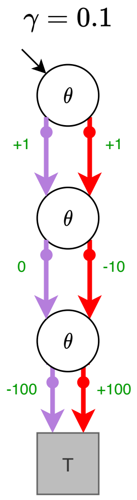


Figure 7.2: MDP environment with a low discount factor for comparing the control performance of agents that differ in how discounting is accounted for in the prediction objective. All states appear the same. Two actions are available to the agent, left (represented as lilac coloured transitions) and right (represented as red coloured transitions). The optimal policy is to take the right action. Under this policy, the start-state value is 1. Under the policy of taking the left action, the start-state value is 0.

There are three states that all appear the same and two actions, Left and Right. The optimal policy is to take action Right. There is a motivation for the above environment in comparing different settings of interest. If too high an interest is given to the second state, the agent may quickly learn that the value of the single feature vector common to all states is close to -10 if the critic step-size is large enough. Then, in the second state, the advantage of taking action Left is much higher than the advantage of taking action Right while the advantages of taking either action are the same in the first state. Therefore, taking action Left is reinforced over time.

The results follow in Figure 7.3 with the parameter studies over different actor step-size parameters for fixed critic step-size parameters.

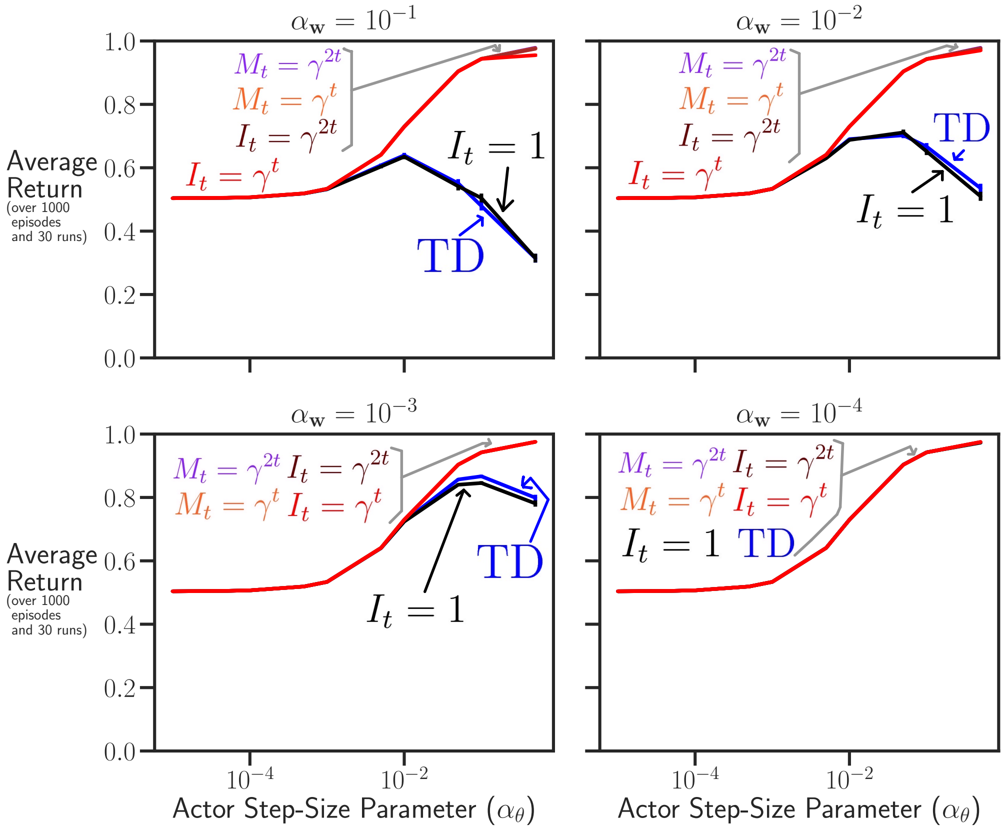


Figure 7.3: Parameter study in the MDP given in Figure 7.2. Each plot presents a parameter study over different actor step-size parameters ($\alpha_{\boldsymbol{\theta}}$) for a fixed critic step-size parameter (α_w). Since the y co-ordinate represents average return which denotes control performance, higher is better in the plots. The error bars denote one standard error. The algorithms compared are actor-critic algorithms that only differ in their of use $1, \gamma^t$ or γ^{2t} each as interest or emphasis in the critic.

To analyze the results, let us refer back to the emphasis visualization plot, Figure 6.1. The emphasis of most algorithms decay very quickly when γ is small. However, there are two exceptions. The algorithm using $M_t = 1, \forall t$ and the algorithm using $I_t = 1$. Note that the former in fact can be viewed as an algorithm using an interest of 1 at the start-state and $1 - \gamma = 1 - 0.1 = 0.9$ at later states (see Eqn. 3.8). So, the two algorithms are very similar in their choice of interest over states. These two algorithms that have a high emphasis over time that does not decay quickly are exactly the ones that perform poorly in some settings of the actor and critic step-size parameters.

Specifically, when the the critic step-size is large, making large actor updates causes learning to suffer. These findings align with the hypothesis we had about what might occur when the critic step-size is large. The results thus demonstrate that using an interest or emphasis of 1 uniformly results in poor control performance sometimes and hence the resulting prediction objectives may not be appropriate to solve with the aim of solving the control problem.

The results on the previous environment indicate that we only seem to have an issue of magnitude due to the actor step-size parameter. However, as we will now see, with a high discount factor, we see that the choice of interest can have a much more drastic effect.

Consider another environment where the discount factor is high $\gamma = 0.99$. This is a discount factor close to 1 which aligns with the discount factors commonly used in practice. The MDP is specified in Figure 7.4. Again, all the states look the same and there are two actions. Here, we call them Continue and Terminate. The former continues along the chain, each time accruing the agent some increasingly large negative reward while the latter transitions the agent to the terminal state with an increasingly large positive reward. The plot that follows the MDP diagram specifies the value of the start-state for different policies given the probability of selecting the continue action.

The optimal policy is stochastic as the agent must continue and then terminate to receive large positive rewards. Moreover, crucially, the discount factor affects the optimal policy. Thus, this problem where the optimal policy is stochastic and depends on the discount factor is a good test for actor-critic agents and particularly, studying the choice of discounting-dependent decay terms in interest and emphasis.

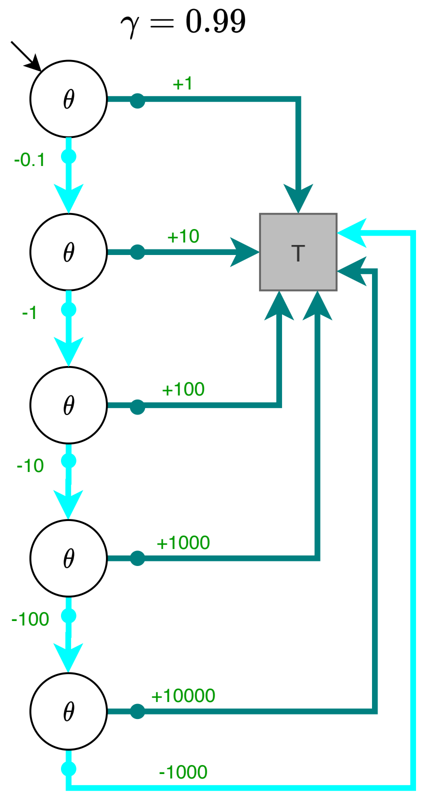


Figure 7.4: MDP environment with a high discount factor for comparing the control performance of agents that differ in how discounting is accounted for in the prediction objective. All states appear the same. Two actions are available to the agent, continue (represented as cyan coloured transitions) and terminate (represented as teal coloured transitions). The optimal policy is stochastic.

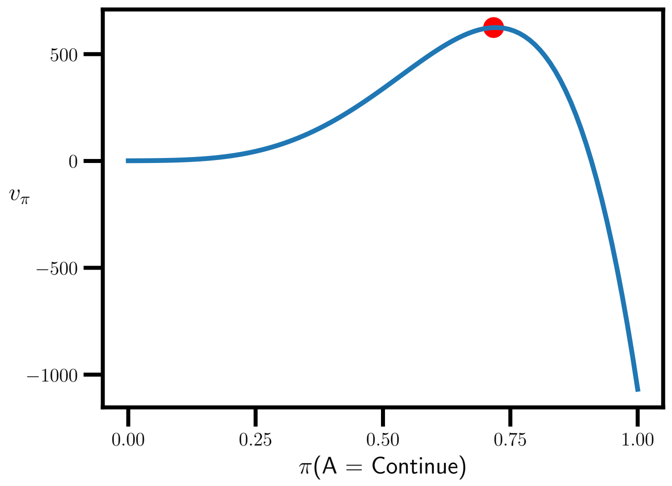


Figure 7.5: Visualization of the value under different policies for the MDP given in Figure 7.4. Recall that all states appear the same, so there is only one value. As we can see, the optimal policy is to take action Continue with probability about 0.7. The state value under this policy is about 600.

Again, the results follow in Figure 7.6 with parameter studies on the actor step-size parameter for fixed critic step-size parameters.

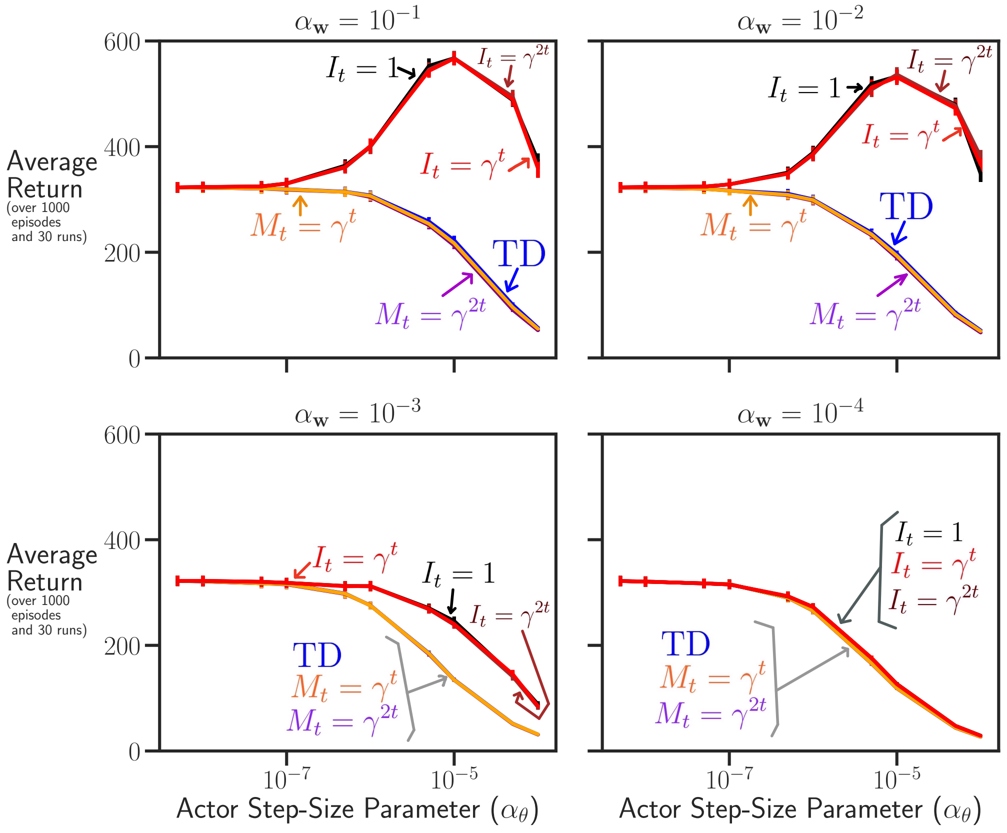


Figure 7.6: Parameter study in the MDP given in Figure 7.4. Each plot presents a parameter study over different actor step-size parameters ($\alpha_{\boldsymbol{\theta}}$) for a fixed critic step-size parameter (α_w). Since the y co-ordinate represents average return which denotes control performance, higher is better in the plots. The error bars denote one standard error. The algorithms compared are actor-critic algorithms that only differ in their of use $1, \gamma^t$ or γ^{2t} each as interest or emphasis in the critic.

We can see that the actor-critic agents using an emphasis of 1, γ^t or γ^{2t} in critic updates do not perform well across all parameter settings. However, using each of them as the interest leads to good control performance in some settings of the actor and critic step-size parameters. Note that since $\gamma \approx 1$, the interests these algorithms use are all approximately the same. However, we saw that using $I_t = 1$ results in poor performance when γ is low. Thus, generally, we can conclude that we should use an interest of γ^t or, agreeing with the interest we presented, γ^{2t}.

7.3 Mountain Car and Puddle World Experiments

We now present experiments and results in common benchmark domains, Mountain Car and Puddle World. The aim of these experiments is to study whether there is any benefit to using the norm of the previous characteristic eligibility term, $\|\nabla_{\boldsymbol{\theta}} \ln \pi(A_{t-1}|S_{t-1}, \boldsymbol{\theta})\|_2^2$ in the interest over states for the prediction subproblem.

Let us first discuss the environments. There are some characteristics shared by both domains. The state-space is continuous. Hence, it is not possible to learn about each state separately and agents likely need to use some form of function approximation. The action-space is discrete. Finally, the discount factor is 1. We now explain each of these domains in more detail.

Puddle World. The goal of an agent in the Puddle World domain is to move from the starting state to a goal state while avoiding a puddle. The puddle world domain has been used in different works with different formulations (Degris et al., 2012; Sutton, 1996). We use the domain presented in Off-Policy Actor Critic work with some small differences. The state-space consists of the 2-dimensional continuous coordinates within $[0, 1]^2$. The action space consists of 4 actions that can be succinctly described as action vectors: $(-0.05, 0.00), (0.00, -0.05), (0.00, 0.05), (0.05, 0.00)$. Uniform noise in [-.025, .025] is added to each component of the action vector. The reward function is defined based on the position the agent is in after taking an action. Suppose this position is (x, y), then the reward is defined as follows: $-1-2(\mathcal{N}(x, .3, .1) \cdot \mathcal{N}(y, .6, .03) + \mathcal{N}(x, .4, .03) \cdot \mathcal{N}(y, .5, .1) + \mathcal{N}(x, .8, .03) \cdot \mathcal{N}(y, .9, .1))$ where $\mathcal{N}$ is the probability density function of the normal distribution. An episode ends when the distance to the goal in L1-norm is less than 0.1 or 1000 steps are completed. The differences are therefore that the no-operation action $(0, 0)$ is not included in the action space and episodes terminate in 1000 steps as opposed to 5000 steps.

Mountain Car. The goal of an agent in the Mountain Car domain is to move along a valley and gain enough velocity to climb to the top of one side of the valley. We use the MountainCar-v0 domain from the Open AI Gym framework (Brockman et al., 2016) with a small modification. The state space is a 2 dimensional continuous space consisting of the 1 dimensional position (between -1.2 and 0.6) and the velocity of the agent (between -0.07 and 0.07). There are three actions available to the agent, decelerate, no-operation and accelerate. The reward function is -1 per step that the agent does not reach the goal. Episodes by default terminate when the agent reaches the goal or at 200 steps in the Open AI Gym environment, we change the termination condition based on timesteps to 1000 steps.

Now, we discuss the algorithms we used in the experiments. We used tile coding function approximation with 8 tilings resulting in binary vectors with exactly 8 components being 1 and the rest being 0. Each tiling consisted of tiles covering a fourth of the state space along each of the two components of the 2 dimensional state-space. The tilings are asymmetrically offset by successive odd numbers. We considered four algorithms: TD, $I_t = 1$, ONCE-AC and MINCE-AC. TD or $M_t = 1$ refers to the commonly used actor-critic algorithm that uses TD(0) for the critic. $I_t = 1$ is equivalent to $I_t = \gamma^{2t}$ since $\gamma = 1$. Therefore, $I_t = 1$ is the algorithm using one of the two terms in the interest we found but not both. Recall that ONCE-AC uses the previous characteristic eligibility seen online and MINCE-AC estimates the mean norm of the previous characteristic eligibility given a state. We tried three different settings for the step-size parameter used to learn the MINCE weighting: $\{10^{-1}, 10^{-3}, 10^{-5}\}$. For the critic step-size parameter, we sweeped over multiples of $\frac{1}{8}$. Using a critic step-size of exactly $\frac{1}{8}$ would result in one-step learning that moves the estimate fully to the target. There is no similar simple way to set the actor step-size parameter and therefore we performed a wide sweep over a logarithmic scale. All the weight vectors were initialized to zero.

The results now follow in the same pattern. We have different plots for different critic step-size parameters and sweep over the actor step-size parameter. Multiple plots are shown for MINCE-AC for different step-size parameters α_η that it uses for for updating η which estimates the mean norm of the previous characteristic eligibility.

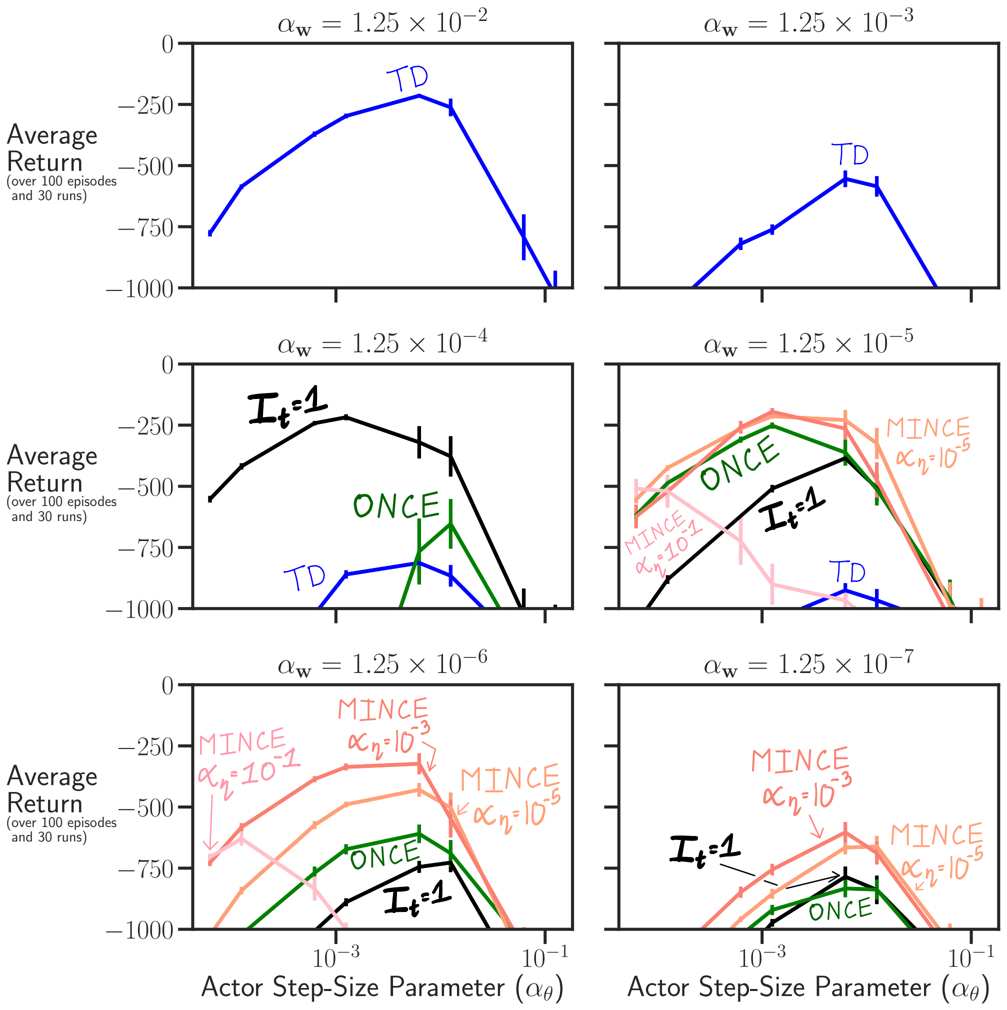


Figure 7.7: Parameter study comparing episodic actor-critic algorithms differing in their choices of interest in the Puddle World Domain. Each plot presents a parameter study over different actor step-size parameters (α_θ) for a fixed critic step-size parameter ($\alpha_{\mathbf{w}}$). Since the y co-ordinate represents the average return, higher is better in these plots. The error bars denote one standard error. The actor-critic algorithm that has been commonly used in practice, *i.e.* with a TD(0) critic, is denoted by the blue lines labeled TD. The rest of the algorithms are novel algorithms we propose based on the interest we found. The actor-critic only accounting for the discounting decay term is denoted by the black lines labeled $I_t = 1$ since $\gamma = 1$. ONCE refers to the algorithm using the online observed squared norm of the previous characteristic eligibility as the interest. Finally, MINCE refers to the algorithm that estimates the mean squared norm of the previous characteristic eligibility and uses it as the interest.

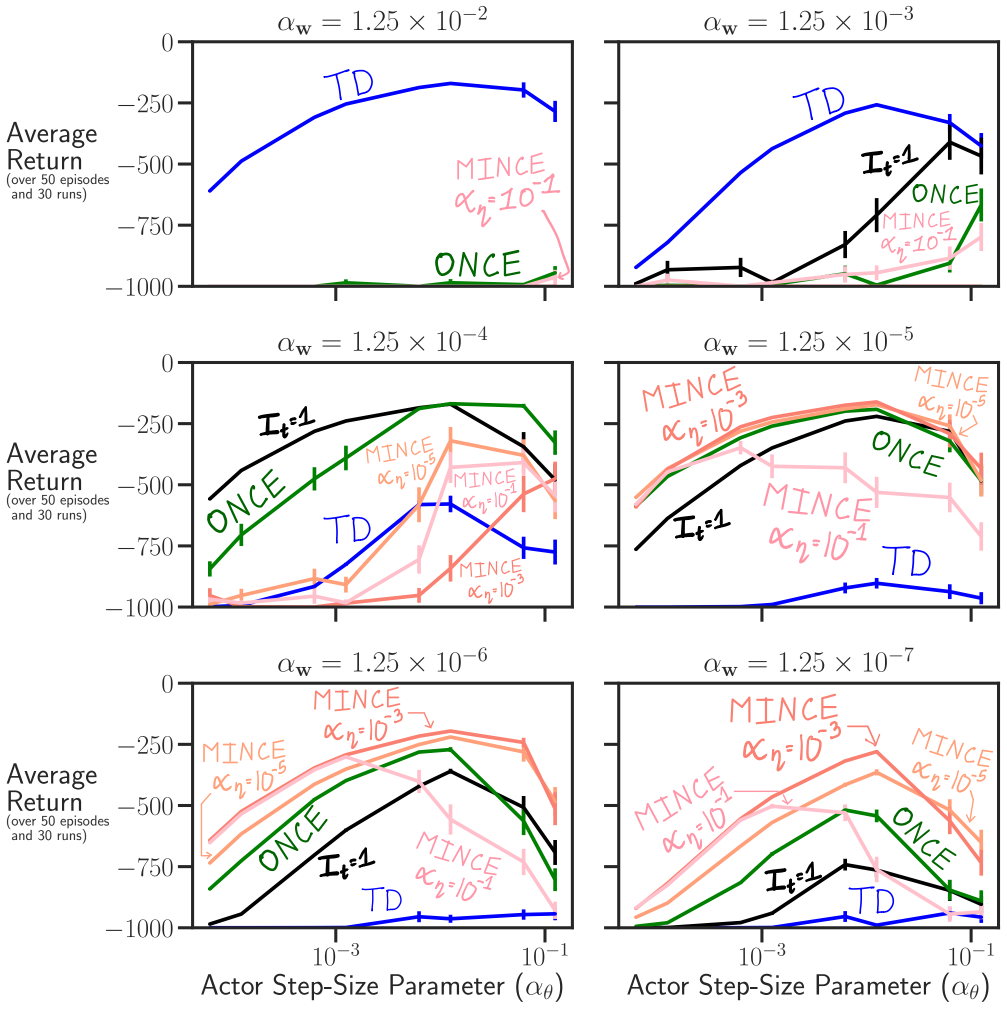


Figure 7.8: Parameter study comparing episodic actor-critic algorithms differing in their choices of interest in the Mountain Car Domain. Each plot presents a parameter study over different actor step-size parameters (α_θ) for a fixed critic step-size parameter ($\alpha_{\mathbf{w}}$). Since the y co-ordinate represents the average return, higher is better in these plots. The error bars denote one standard error. The actor-critic algorithm that has been commonly used in practice, *i.e.* with a TD(0) critic, is denoted by the blue lines labeled TD. The rest of the algorithms are novel algorithms we propose based on the interest we found. The actor-critic only accounting for the discounting decay term is denoted by the black lines labeled $I_t = 1$ since $\gamma = 1$. ONCE refers to the algorithm using the online observed squared norm of the previous characteristic eligibility as the interest. Finally, MINCE refers to the algorithm that estimates the mean squared norm of the previous characteristic eligibility and uses it as the interest.

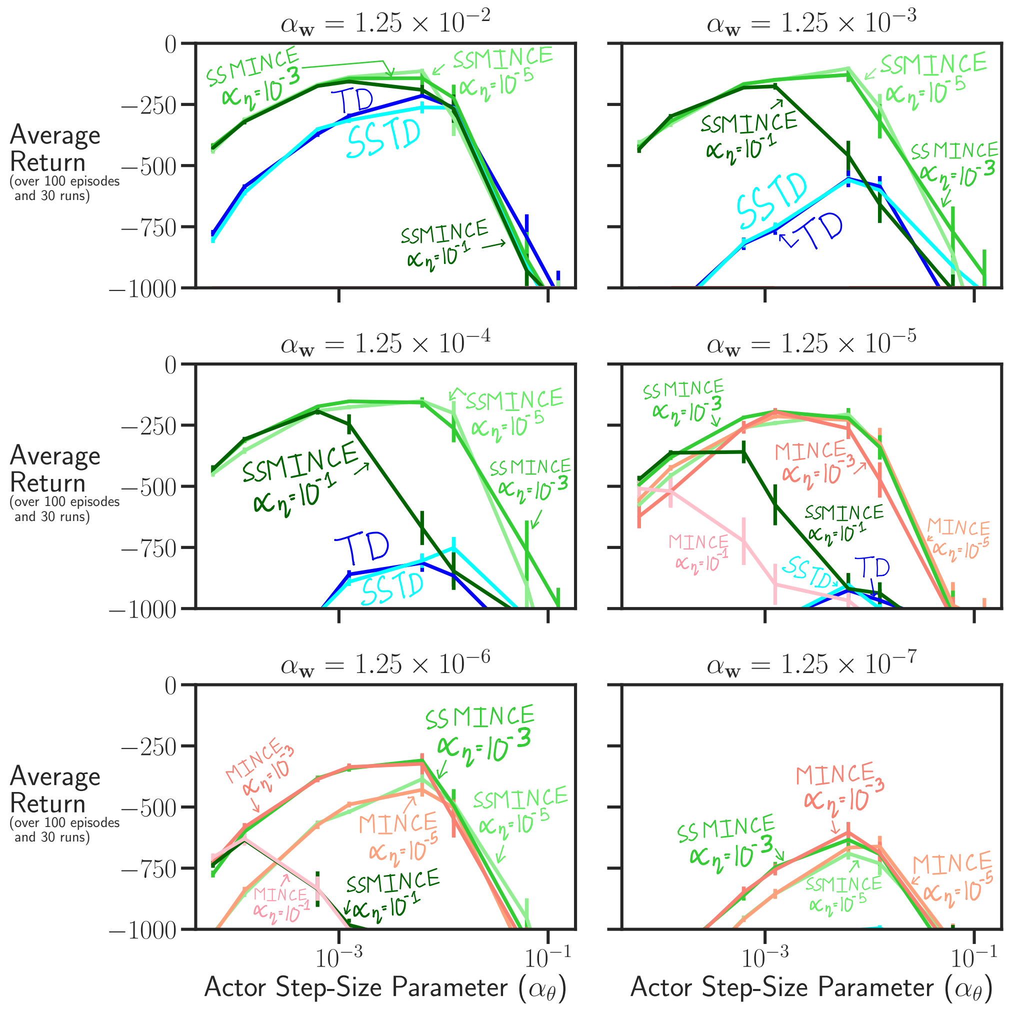


Figure 7.9: Parameter study in the Puddle World Domain with the addition of the Sliding-Step Variants of the TD and ETD algorithms. Each plot presents a parameter study over different actor step-size parameters (α_θ) for a fixed critic step-size parameter ($\alpha_{\mathbf{w}}$). Since the y co-ordinate represents the average return, higher is better in these plots. The error bars denote one standard error. The actor-critic algorithm that has been commonly used in practice, *i.e.* with a TD(0) critic, is denoted by the blue lines labeled TD. SSTD refers to the sliding-step variant of TD(0). MINCE refers to the algorithm that arose as a culmination of our search for a prediction objective given a control objective. It estimates the mean squared norm of the previous characteristic eligibility and uses it in the interest. SSMINCE refers to the algorithm using sliding-step ETD with the MINCE interest weighting.

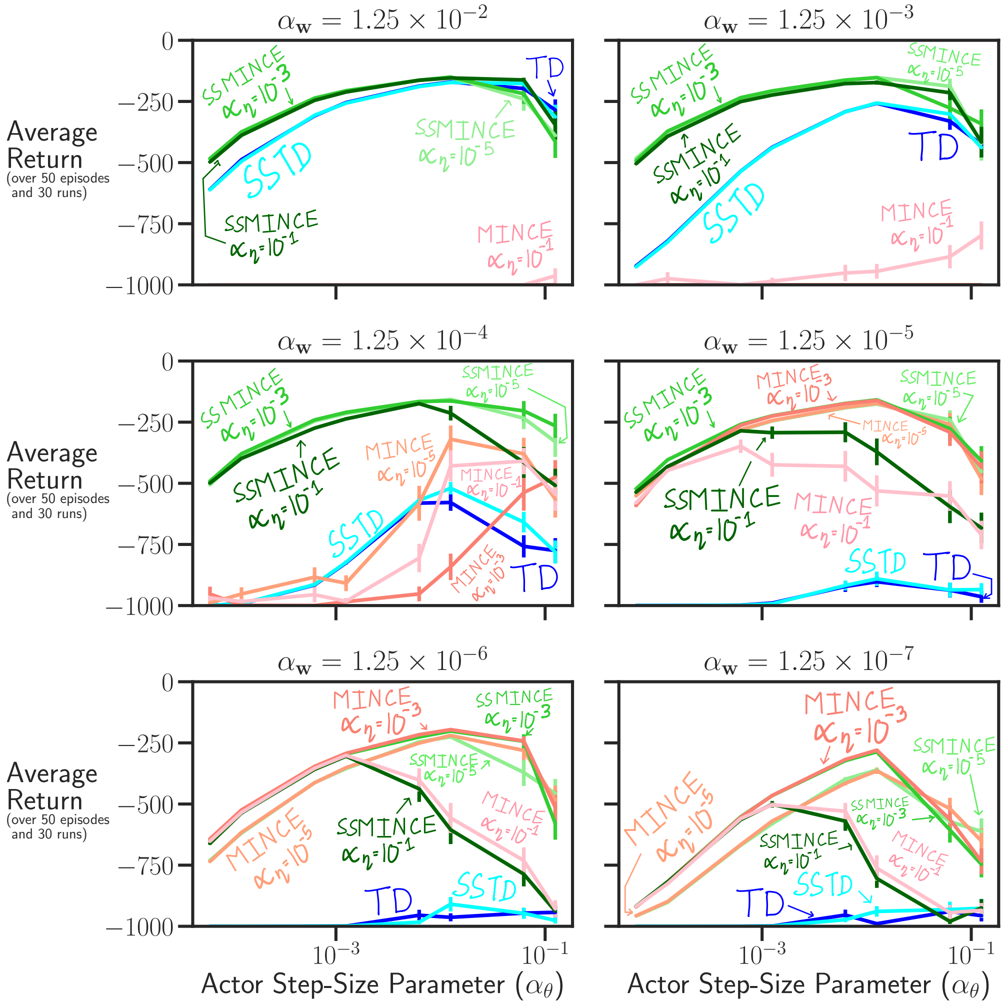


Figure 7.10: Parameter study in the Mountain Car World Domain with the addition of the Sliding-Step Variants of the TD and ETD algorithms. Each plot presents a parameter study over different actor step-size parameters (α_θ) for a fixed critic step-size parameter ($\alpha_{\mathbf{w}}$). Since the y co-ordinate represents the average return, higher is better in these plots. The error bars denote one standard error. The actor-critic algorithm that has been commonly used in practice, *i.e.* with a TD(0) critic, is denoted by the blue lines labeled TD. SSTD refers to the sliding-step variant of TD(0). MINCE refers to the algorithm that arose as a culmination of our search for a prediction objective given a control objective. It estimates the mean squared norm of the previous characteristic eligibility and uses it in the interest. SSMINCE refers to the algorithm using sliding-step ETD with the MINCE interest weighting.

As can be seen from the first two figures, MINCE-AC performed the best across a wide range of actor step-size parameter settings when the critic-size was small (albeit if α_η is small). In the same small critic step-size regime, ONCE and $I_t = 1$ also performed quite well with the episodic actor-critic using TD performing the poorest. These results are promising with the caveat that the the critic step-size should be small for the novel algorithms to perform well. This is however expected as TD and ETD generally learn well in different step-size regimes as the follow-on trace may have high variance.

A more stronger result would be better as the Episodic Actor-Critic using a TD critic performed the best when the critic step-size parameter was large. An ideal result would be that using the MINCE interest weighting results in better control performance across a wide range of actor and critic step-size parameters than the commonly used actor-critic which uses a TD critic. In a final set of experiments, we run sliding-step TD and sliding-step ETD with the MINCE interest weighting. We then get the desired result that using sliding-step ETD with the prediction objective using the MINCE interest weighting on states gives us the best control performance across a wide of parameter settings.

While we have statements above about an algorithm being better than another, we do note that we have to qualify our statements to only be about the environments we used in our experiments. A stronger theoretical argument that presents how control performance is influenced by the choice of prediction subproblem would be needed to make stronger statements about the relative control performance of algorithms differing only in their choice of prediction subproblem.

Chapter 8

Conclusion

In this thesis I took upon the responsibility of finding a way to relate the prediction and control problems. We had well defined objectives for the prediction and control problems and a way to specify how much we care about different states in the prediction problem through the interest function of Emphatic TD. We did not however have an understanding as to how the prediction subproblem should be chosen to improve control performance. Particularly, it was unknown what prediction objective or what interest over states we should have in states when solving the control problem.

We focused on a particular control objective and re-iterated the need for a γ^t in actor updates. We showed that not using this term can result in poor empirical performance.

We then found a particular interest over states by looking at the bias introduced due to function approximation and bootstrapping. Hence, we provided an interest with motivation that its use in the prediction objective or subproblem could result in improved control performance. Thus, we interrelated the prediction and control objectives by seeking for an interest. Our empirical results confirmed that we indeed get improved control performance on commonly used benchmark environments when using the interest we found in an ETD critic. We do not however make a strong theoretical argument that using the suggested interest with ETD will always result in improved control performance. Further theoretical work presenting bounds on the control performance given the use of a particular prediction subproblem could be an interesting direction to pursue.

We have however presented empirical evidence that suggests the choice of prediction subproblem can be an important choice of a control algorithm that affects control performance. Moreover, we also had a theoretical argument that a particular choice of subproblem, the subproblem solved by the MINCE-AC algorithm, can be promising for control and is worthwhile to study further.

Future work can also develop a better understanding of how terms dependent on previous states, actions and time should be incorporated into interest. We also focused on the one-step ETD(0) case in our experiments to simplify analysis. Empirical experiments with the suggested interest with $\lambda > 0$ is an obvious avenue to pursue. Additionally, testing the algorithms with ETD(λ, β) (Hallak et al., 2016) could also be good to study. In this algorithm the follow-on decays by β instead of γ and β trades off bias and variance.

References

Greg Brockman et al. (2016). *OpenAI Gym.* arXiv preprint:1606.01540. 56

Thomas Degris, Martha White, and Richard S Sutton (2012). *Off-policy actor-critic.* arXiv preprint:1205.4839. 55

Sina Ghiassian et al. (2018). *Online off-policy prediction.* arXiv preprint:1811.02597. 18

Assaf Hallak et al. (2016). *Generalized emphatic temporal difference learning: Bias-variance analysis.* In: Thirtieth AAAI Conference on Artificial Intelligence. 63

A. Rupam Mahmood et al. (2015). *Emphatic temporal-difference learning.* arXiv preprint:1507.01569. 3, 17

John McCarthy (2007). *What is artificial intelligence?* Technical report, Stanford University. URL: `www-formal.stanford.edu/jmc/whatisai.html`. 1

Chris Nota and Philip S. Thomas (2019). *Is the policy gradient a gradient?* arXiv preprint:1906.07073. 25, 48

Richard S. Sutton (1985). *Temporal Credit Assignment in Reinforcement Learning.* PhD thesis, University of Massachusetts. 3

Richard S. Sutton (1996). *Generalization in reinforcement learning: Successful examples using sparse coarse coding.* In: Advances in Neural Information Processing Systems, pp:1038–1044. 55

Richard S. Sutton et al. (2011). *Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction.* In: The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2, pp:761–768. 2

Richard S. Sutton and Andrew G. Barto (2018). *Reinforcement learning: An introduction.* MIT press. 1, 13, 26

Richard S. Sutton, A. Rupam Mahmood, and Martha White (2016). *An emphatic approach to the problem of off-policy temporal-difference learning.* In: The Journal of Machine Learning Research, Volume 17, Number 1, pp:2603–2631. 3, 17, 19

Tian Tian (2018). *Extending the Sliding-step Technique of Stochastic Gradient Descent to Temporal Difference Learning.* PhD thesis, University of Alberta. 43, 44

Ronald J. Williams (1992). *Simple statistical gradient-following algorithms for connectionist reinforcement learning.* In: Machine learning, Volume 8, Number 3-4, pp:229–256. 25, 28

Cathy Wu et al. (2018). *Variance reduction for policy gradient with action-dependent factorized baselines*. arXiv preprint:1803.07246. 33

Huizhen Yu (2015). *On convergence of emphatic temporal-difference learning*. In: Conference on Learning Theory, pp:1724–1761. 3, 19