UNIVERSITY OF CALIFORNIA,
IRVINE

StaTeS-SQL: Soft Q Learning with State-Dependent Temperature Scheduling

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Computer Science

by

Dailin Hu

Thesis Committee:
Assistant Professor Roy Fox, Chair
Professor Alexander Ihler
Associate Professor Stephan Mandt

2022

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I started off this journey in the hope of exploring the frontiers of artificial intelligence, and in the pursuit of a doctor's degree. Life doesn't always go according to plans, however, and as of now I'm graduating without the privilege of making a joke on Doctor Who every time I introduce myself. Nevertheless, the last two years has definitely not been easy, and I'd like to say thank you to those who offered me help along the way.

I would like to express my deep gratitude and appreciation to my advisor Dr. Roy Fox, whose patience, support and guidance made this work possible and helped me throughout this journey.

I would also like to thank my lab mates for all the inspiring and motivating discussions, and for all the fun and exciting projects we worked on together. My best wishes to them and their future endeavors!

Many thanks to the HPI Research Center in Machine Learning and Data Science for their financial support which helped me through my research.

I'd like to thank my parents and my brother, without whom I will not be who I am today.

Finally, I want to thank my husband and the love of my life, Wesley, for his unwavering and unconditional love and support.

ABSTRACT OF THE THESIS

StaTeS-SQL: Soft Q Learning with State-Dependent Temperature Scheduling

By

Dailin Hu

Master of Science in Computer Science

University of California, Irvine, 2022

Assistant Professor Roy Fox, Chair

Maximum Entropy Reinforcement Learning (MaxEnt RL) algorithms such as Soft Q-Learning (SQL) trade off reward and policy entropy, which has the potential to improve training stability and robustness. Most MaxEnt RL methods, however, use a constant tradeoff coefficient (temperature), contrary to the intuition that the temperature should be high early in training to avoid overfitting to noisy value estimates and decrease later in training as we increasingly trust high value estimates to truly lead to good rewards. Moreover, our confidence in value estimates is state-dependent, increasing every time we use more evidence to update a state's value estimate. In this paper, we present a simple state-based temperature scheduling approach and instantiate it for SQL as StaTeS-SQL. We prove the convergence of this method in the tabular case, describe how to use pseudo-counts generated by a density model to schedule the state-dependent temperature in large state spaces, and propose a combination of our method with advanced techniques collectively known as Rainbow. We evaluate our approach on the Atari Learning Environment benchmark and outperform Rainbow in 18 of 20 domains.

Chapter 1

Introduction

Deep Reinforcement Learning (RL) methods that use neural-network function approximators to learn control policies have shown great performance in domains including games [Mnih et al., 2015, Silver et al., 2017], robot control [Haarnoja et al., 2017], and autonomous driving [Sallab et al., 2017]. The training of such function approximators, however, is often very sensitive to the tuning of the algorithm's hyperparameters in different environments [Henderson et al., 2018]. Recent work in Maximum-Entropy Reinforcement Learning (MaxEnt RL) [Fox et al., 2016, Haarnoja et al., 2018b] trades off maximizing the policy's value with its entropy, producing policies that are more robust to disturbances in the environment dynamics and reward function [Haarnoja et al., 2017, Eysenbach and Levine, 2021]. Despite achieving state-of-the-art performances in a wide range of continuous-control benchmark tasks, these methods have not yet shown similar success in discrete environments [Christodoulou, 2019].

MaxEnt RL maximizes an objective involving two terms: the expected return obtained by the agent's policy and the expected entropy of that policy. Combining the two terms is a coefficient, the inverse-temperature β, serving as a conversion rate from value to entropy. Intuitively, in early stages of training, β should be lower to emphasize entropy maximization in the face of value uncertainty. Later in training, β should be higher as we become increasingly more confident in our value estimates and capable of safely maximizing them. Against this intuition, most MaxEnt RL algorithms use a constant temperature throughout training, preventing it from reflecting the dynamics of our confidence in value estimates as we accumulate evidence for them.

Importantly, our confidence in a reinforcement learner's value estimates is very much state-dependent. Value estimates that have been updated more times, from more empirical evidence, are more reliable. Early in learning, states that are more easily reached by an uninformed exploration policy are experienced more frequently. As the value estimate in these states is updated and improves in accuracy, their β should increase. Later in learning, as exploration reaches novel states, whose value estimates are less reliable, their β should start very low again. In a batch of replayed experience, the hardness of each target value maximization should increase with our confidence in it. We emphasize that this insight pertains to the agent's optimization objective itself in whichever state values it updates, rather than to the prioritization of uncertain states often used in curiosity-driven exploration [Pathak et al., 2017, Bellemare et al., 2016, Ostrovski et al., 2017] and replay experience sampling [Schaul et al., 2015].

In this paper, we describe a simple state-dependent temperature scheduling (abbrev. StaTeS) method for MaxEnt RL based on a pseudo-count of state value updates derived from a CTS density model [Ostrovski et al., 2017], and its specialization to the SQL algorithm. StaTeS-SQL is the first MaxEnt RL method to reliably achieve, in discrete environments, good performance that is comparable to or better than state-of-the-art model-free methods for memoryless policies (Chapter 1). We prove the convergence of tabular StaTeS-SQL, describe how to use pseudo-counts generated by a density model to schedule the state-dependent temperature in large state spaces, and propose a combination of our method with advanced techniques collectively known as Rainbow [Hessel et al., 2018]. Evaluating StaTeS-SQL on

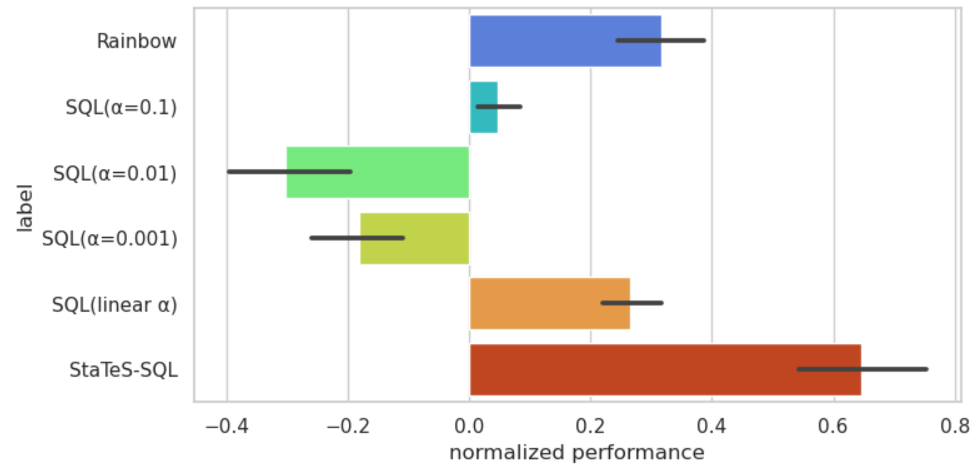

Figure 1.1: StaTeS-SQL average human-normalized performance on 18 domains in the Atari Learning Environment benchmark, compared with Rainbow-DQN and SQL with constant and linearly scheduled temperatures. The normalized performance is evaluated by averaging across 20 games the difference between the trained agent returns and random policy returns (see table B.1) divided by the difference between human performance Mnih et al. [2015] and random policy returns. Rewards are averaged over 5 runs. The black error bar indicates a confidence interval of 80%. SQL with fixed constant temperature can under-perform a random policy if the temperature is not selected properly. The results after 500K interaction steps are averaged over 5 runs, and showing an 80% confidence interval.

20 benchmark domains in the Atari Learning Environment (ALE) platform [Bellemare et al., 2013] suggests that this method outperforms Rainbow-DQN and significantly improves over SQL with a constant temperature [Haarnoja et al., 2017] and a state-independent linearly scheduled temperature [Fox et al., 2016].

Chapter 2

Preliminaries

2.1 Reinforcement Learning

We consider environments modeled as a Markov Decision Process (MDP). We describe this process by a tuple $\langle \mathcal{S}, \mathcal{A}, p, r \rangle$, where $\mathcal{S}$ represents a state space and $\mathcal{A}$ represents an action space. $p(s_{t+1}|s_t, a_t) : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ represents the probability distribution of the next state s_{t+1} given the current state s_t and action a_t. $r_t = r(s_t, a_t) : \mathcal{S} \times \mathcal{A} \to R$ describes the reward for each transition t. Jointly with the MDP, a policy $\pi(a_t|s_t) : \mathcal{S} \to \Delta(\mathcal{A})$ induces a distribution p_π over trajectories $\xi = s_0, a_0, r_0, s_1, a_1, r_1, \ldots$. The discounted return $R(\xi)$, for a discount factor $0 \leq \gamma < 1$, is defined as $R(\xi) = \sum_t \gamma^t r_t$. It is also convenient to define a state distribution $p_\pi^\gamma(s) = (1 - \gamma) \sum_t \gamma^t p_\pi(s_t = s)$, describing the distribution of s_t at a time t distributed geometrically with parameter $1 - \gamma$. The RL objective is to find a policy π maximizing $J_\pi = E_{\xi \sim p_\pi}[R(\xi)] = \frac{1}{1-\gamma} E_{s \sim p_\pi^\gamma}[E_{(a|s) \sim \pi}[r(s, a)]]$.

2.2 Maximum Entropy Reinforcement Learning

The MaxEnt RL objective augments the standard reinforcement learning objective of maximizing expected discounted rewards by adding an entropy term

$$\pi^* = \arg\max_{\pi} E_{s\sim p_{\pi}^{\gamma}} \left[E_{(a|s)\sim\pi}[r(s,a)] + \tfrac{1}{\beta} H[\pi(\cdot|s)] \right], \tag{2.1}$$

where β is an *inverse-temperature* parameter that controls the stochasticity of the optimal policy by determining the relative importance between the reward and policy entropy $H[\pi]$. In early stages of training, β should intuitively be assigned a small value to induce a more stochastic and agnostic objective, and in later stages of training $\beta \to \infty$ to approach the deterministic behavior that optimizes the standard reinforcement learning objective.

2.3 Mellowmax and Soft Q-Learning (SQL)

RL methods often involve the state–action value function $Q(s,a) = E[R|s_0 = s, a_0 = a]$ that is optimally a fixed point of the Bellman operator

$$\mathcal{T}[Q](s,a) = r(s,a) + \gamma E_{(s'|s,a)\sim p}[\max_{a'} Q(s',a')]. \tag{2.2}$$

The MaxEnt RL objective suggests finding a fixed point of a *soft* Bellman operator [Rubin et al., 2012]

$$\mathcal{T}_{\beta}[Q](s,a) = r(s,a) + \gamma E_{(s'|s,a)\sim p}[\max_{\pi}(E_{(a'|s')\sim\pi}[Q(s',a')] + \tfrac{1}{\beta} H[\pi(\cdot|s')])], \tag{2.3}$$

where $\beta > 0$ is an inverse-temperature tradeoff coefficient, and the optimizer in (2.3) is the softmax policy $\pi(a|s) \propto \exp(\beta Q(s,a))$. This can be written in the form of a log-partition function, also called the mellowmax operator [Asadi and Littman, 2017]:

$$\begin{aligned}\mathcal{T}_\beta[Q](s,a) &= r(s,a) + \gamma E_{(s'|s,a)\sim p}[{}_{a';\beta} Q(s',a')] \\ &\overset{\text{def}}{=} r(s,a) + \gamma E_{(s'|s,a)\sim p}[\tfrac{1}{\beta}\log_{a'} \exp(\beta Q(s',a'))]. \end{aligned} \tag{2.4}$$

Mellowmax is non-decreasing in β [Kim et al., 2019] and a non-expansion for a fixed β under the supremum norm [Fox et al., 2016]. As $\beta \to \infty$, it converges to pure maximization, as quantified by the following lemma.

Lemma 2.3.1. *For any $x = [x_1, \dots, x_n] \in R^n$*

$$\max_i(x_i) - \frac{\log n}{\beta} \le_{i;\beta} (x_i) \le \max_i(x_i). \tag{2.5}$$

Proof. Let $m = \max_i(x_i)$, then

$$\exp(\beta m) \le \sum_i \exp(\beta x_i) \le n \exp(\beta m). \tag{2.6}$$

Dividing these terms by n, applying the logarithm, and dividing by β, the lemma follows. □

Soft Q-Learning [Fox et al., 2016, Haarnoja et al., 2017] uses a model-free empirical estimate of the soft Bellman operator (2.4) as the target for learning a Q function. When the Q function has a tabular representation, the contraction property of the soft Bellman operator guarantees convergence to the operator's fixed point, the softmax of which is the optimal policy in equation (2.1) [Fox et al., 2016].

Chapter 3

Related Work

Most MaxEnt RL algorithms, such as SQL [Haarnoja et al., 2017] and some of its variants, use a constant temperature throughout training. The mellowmax operator in SQL's soft Bellman backup operator enables the algorithm to put higher weight on the policy entropy when value uncertainty is high and tend to reward maximization as values become known. Selecting a constant inverse-temperature β may not reflect the changing confidence of the value estimates throughout learning. Furthermore, even when there exists a constant β that works well, it is very much domain-dependent, making it hard to tune as a hyperparameter.

Soft Actor–Critic (SAC) [Haarnoja et al., 2018a] adjusts the temperature β^{-1} automatically with stochastic gradient descent on a Lagrangian that compares the policy entropy to a target entropy. This approach is very successful in continuous control problems, such as in robot learning, but often has poor performance in discrete action spaces [Christodoulou, 2019]. Xu et al. [2021] investigate this issue and suggest using a heuristic target-entropy scheduling method for tuning the temperature in SAC, achieving significant improvement over SAC in several discrete-action benchmark environments. They also apply this method to SQL by approximating a Q-function-based soft-greedy policy, but the agent's learning shows instability throughout training due to the fundamental difference between the structure of SQL and SAC.

Fox et al. [2016] propose a linear inverse-temperature schedule, in which $\beta_i = \kappa i$ in training step i. This corresponds to the intuition that β should increase throughout learning, but the coefficient κ remains a domain-dependent hyperparameter to be tuned. Grau-Moya et al. [2018] adapt this method to Mutual Information RL (MIRL) for tabular Q-Learning and suggest using the adaptive β scheduling from Leibfried et al. [2017] to update β according to the inverse of the empirical square loss between the Q function and its target value.

Entropy-Regularized Q-learning (EQL) [Fox, 2019] uses an ensemble of tabular value estimates that characterizes model uncertainty to compute a value of β that, in theory, eliminates the bias due to overestimation of the max operator. Unbiased Soft Q-learning (UQL) [Liang et al., 2021] applies the same principle to an ensemble of SQL learners, and combines it with ensemble-based exploration [Lee et al., 2021] to reduced the overestimation bias. In comparison, the method we describe in this paper avoids the large memory and computational complexity associated with ensemble methods, and thus needs to infer an efficient temperature from a single value network.

Chapter 4

State-Dependent Temperature Scheduling

4.1 Motivation

4.1.1 MaxEnt RL

MaxEnt RL follows from the maximum entropy principle [Jaynes, 2003], which states that one should prefer maximally uninformed solutions, subject to the available evidence. MaxEnt RL therefore maximizes average policy entropy, subject to an attainable level of policy value:

$$\begin{aligned} \max_{\pi} \quad & E_{s\sim p_\pi^\gamma}[H[\pi(\cdot|s)]] \\ \text{s.t.} \quad & J_\pi \overset{\text{def}}{=} E_{\xi\sim p_\pi}[R(\xi)] \geq \rho. \end{aligned} \tag{4.1}$$

The Lagrangian of this constrained optimization problem, equation (2.1), trades off the traditional RL objective of maximum policy value with the expected policy entropy. The

Lagrange multiplier is an *inverse-temperature* parameter β that determines the relative importance of the value and entropy terms, and corresponds to the level of value ρ that the agent believes it can attain. Intuitively, in early stages of training, β should be lower to encourage a more agnostic policy, as we have low confidence that the current value estimates are attainable. During training, we become increasingly more confident in our value estimates, and we increase β such that the policy stochasticity decreases and eventually approaches a deterministic action selection. We expect this temperature scheduling to lead to more stable optimization, akin to regularization, and may also improve exploration. As $\beta \to \infty$, corresponding to $\rho \to \max_\pi J_\pi$, the conventional RL learning objective is recovered.

4.1.2 State-dependent temperature

The above application of the maximum-entropy principle to the entire policy π is a reasonable coarse-grained view of the RL estimation–optimization problem. A more nuanced view reveals that different components of the policy, namely $\pi(\cdot|s)$ for different states s, can face uncertainty levels that are very different from one another. We should therefore focus our analysis on individual states.

The optimization problem $\max_{a'} Q(s', a')$ inside the Bellman operator (2.2) implies certainty that the maximal value is attainable using the greedy action. Similarly to (4.1), in states s' whose value estimates are uncertain, we should fall back to maximizing $H[\pi(\cdot|s')]$, subject to $E_{(a'|s')\sim\pi}[Q(s', a')] \geq \rho(s')$. Here, $\rho(s')$ is some value level that available evidence suggests is attainable in s'. The Lagrangian of this problem appears inside the soft Bellman operator (2.3), with a state-dependent multiplier (inverse temperature) $\beta(s')$ monotonic in $\rho(s')$, and is optimized by ${}_{a';\beta(s')}Q(s', a')$.

4.1.3 Linear inverse-temperature scheduling

We propose a state-dependent temperature schedule in which $\beta(s)$ grows linearly with the number of times that the algorithm updates $Q(s,a)$, for any action a. Formally, let $n(s,a)$ be the count of sampled data points of the form (s,a,r,s') used thus far in value updates. Then the inverse-temperature is $\beta(s)=\kappa\sum_a n(s,a)$, with $\kappa>0$ a constant hyperparameter.

Prior work has presented empirical and theoretical evidence supporting linear scheduling. Linear scheduling has shown empirical success in tabular MaxEnt RL [Fox et al., 2016, Grau-Moya et al., 2018]. Fox [2019] prove theoretically that, in binary-action environments and under mild modelling assumptions, a linear inverse temperature completely eliminates the target value estimation bias.

Additional insight can be gained by comparing two families of successful RL algorithms. The first family, consisting of such algorithms as G-Learning [Fox et al., 2016], SQL [Haarnoja et al., 2017], Path Consistency Learning (PCL) [Nachum et al., 2017a], and Soft Actor–Critic (SAC) [Haarnoja et al., 2018b], aims to learn a policy that is the softmax of a value function $Q(s,a)$ with a low temperature. In iteration i, the policy target is

$$\pi_i(a|s)\propto\pi_0(a|s)\exp\beta_i(s)Q_i(s,a)\qquad\forall s\in\mathcal{S},a\in\mathcal{A},\tag{4.2}$$

with π_0 the uniform policy, and the policy is either updated toward the target (4.2) in policy-based and actor–critic methods, or set equal to it in value-based methods. Note that in their original formulations, most of these algorithms use a constant state-independent β, rather than a scheduled state-dependent $\beta_i(s)$.

The second family of algorithms, including Relative Entropy Policy Search (REPS) [Peters et al., 2010], Ψ-learning [Rawlik et al., 2010], Trust Region Policy Optimization (TRPO) [Schulman et al., 2015], and Trust-PCL [Nachum et al., 2017b], aims to use value estimates to gradually update the softmax policy. Instead of the policy entropy, which is the Kullback–Leibler (KL) divergence from a uniform prior policy, these algorithms consider a KL term with the current policy as the prior for the update.[1] Moreover, instead of a low temperature β^{-1}, these algorithms place a large coefficient κ^{-1} on the KL term, to induce small updates in a "trust region". In iteration i, the policy update is therefore

$$\begin{aligned}\pi_i(a|s) &\propto \pi_{i-1}(a|s) \exp \kappa_i(s) Q_i(s,a) \qquad (4.3)\\ &\propto \pi_0(a|s) \exp \left(\sum_{j \leq i} \kappa_j(s) Q_j(s,a) \right). \qquad (4.4)\end{aligned}$$

The two types of learning processes (4.2) and (4.4) can have different properties, because Q_i can change significantly between iterations. For the sake of our intuitive argument here, imagine that Q could be approximated by a constant, and that the inverse-temperatures $\kappa_i(s)$ used in trust-region algorithms were also a constant κ. Then combining the above two equations, we have

$$\beta_i(s) \approx \kappa_0(s) + \sum_{1 \leq j \leq i} \kappa_j(s) \approx \epsilon + \kappa i, \qquad (4.5)$$

where ϵ is a small non-zero positive constant ensuring that the temperature parameter is always finite, κ is a small constant, and i is the number of times that the update (4.4) is applied in state s. In practice, updates are not applied to all actions in state s at the same time, leading to the heuristic definition $i = \sum_a n(s,a)$. Throughout this paper, we set $\kappa = 0.01$ and $\epsilon = 10^{-7}$.

[1]Some algorithms in this family, such as TRPO, use the reverse KL, with the updated policy as the prior.

4.2 Tabular Examples

State-dependent temperature scheduling (abbrev. StaTeS) sets the inverse-temperature to $\beta(s) = \epsilon + \kappa n(s)$, where $n(s) = \sum_a n(s,a)$ is the number of times that $Q(s,a)$ has been updated. We propose StaTeS-SQL, a model-free reinforcement learning algorithm that, on experience (s, a, r, s'), uses the SQL update rule

$$Q(s,a) \leftarrow r + \gamma_{a';\beta(s')} Q(s', a')$$

with the state-dependent $\beta(s')$. We present the pseudocode for StaTeS-SQL in Section 5.1 in the supplementary material.

To demonstrate this method's effectiveness, we compare StaTeS-SQL with Q-learning and SQL on toy domains in which the environments are small enough to directly find the optimal policy. We define a noisy chain-walk problem with m states, each state with two possible actions (see Figure 4.1). The agent always starts at state 1 and each episode ends after m steps. The agent receives a reward of +1 when it takes action 1 at state m, and a reward of -0.1 when it takes any action at any other state. The reward that the agent receives is always corrupted with an additive Gaussian noise of $\mathcal{N}(0, 1)$.

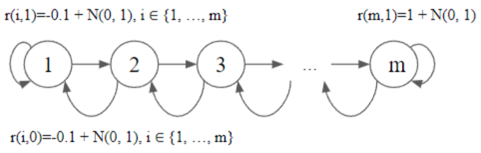
r(i,1)=-0.1 + N(0, 1), i ∈ {1, …, m}
r(m,1)=1 + N(0, 1)
1
2
3
…
m
r(i,0)=-0.1 + N(0, 1), i ∈ {1, …, m}

Figure 4.1: Noisy chain-walk environment

Figure 4.2 show the rewards averaged over 1000 runs comparing four algorithms: (1) Q-learning, (2) SQL with constant inverse-temperature parameters $\beta \in \{10, 100, 1000\}$, (3) SQL with inverse-temperature given by the training timestep times a constant $\kappa \in 1, 0.1, 0.01$, and (4) StaTeS-SQL with a tabular representation, for a chain length of $m \in \{3, 10, 25\}$. These results suggest that StaTeS-SQL can converge significantly faster than Q-learning, soft Q-learning with both constant and linear-scheduled inverse temperatures in this simple domain.

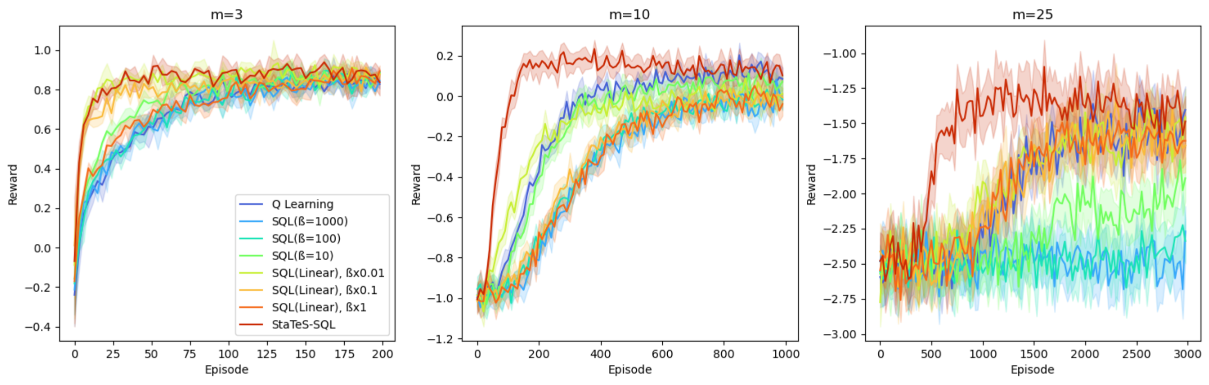

Figure 4.2: Tabular experiment comparing Q-learning, SQL with constant and linear-scheduled inverse temperatures and StaTeS-SQL on a simple chain-walk environment. Rewards are averaged over 1000 runs.

4.2.1 Convergence

In this section, we prove that in the tabular case StaTeS-SQL converges to the same Q value as classic Q-Learning. It is well known that for $\beta = \infty$, i.e. Q-learning, the algorithm converges to the unique fixed point of the Bellman operator. The challenge here is to prove the same result in the limit $\beta \to \infty$.

Lemma 4.2.1. *Let $Q_k = \mathcal{T}[Q_{k-1}]$ be the k-times iterated application of the Bellman operator $\mathcal{T}$ (2.2) to an initial value function $Q_0 : \mathcal{S} \times \mathcal{A} \to R$. Let $\tilde{Q}_k = \mathcal{T}_{\beta_k}[\tilde{Q}_{k-1}]$ be the iterated application of the soft Bellman operator $\mathcal{T}_{\beta_k}$ (2.3) to the same initial value function $\tilde{Q}_0 = Q_0$, with a sequence $\beta_1, \beta_2, \dots$ of inverse temperatures. Then*

$$Q_k - \log |\mathcal{A}| \sum_{i=1}^{k} \frac{\gamma^{k-i}}{\beta_i} \leq \tilde{Q}_k \leq Q_k, \tag{4.6}$$

where inequality is element-wise for each $s \in \mathcal{S}$ and $a \in \mathcal{A}$.

Proof. We prove the lemma by induction on k. When $k = 0$, $\tilde{Q}_0 = Q_0$. Assume that holds for some k. Observe that using Theorem 2.3.1, and by the monotonicity of $\mathcal{T}$, we have for any $s \in \mathcal{S}$ and $a \in \mathcal{A}$

$$\begin{aligned}
\tilde{Q}_{k+1}(s,a) &= r(s,a) + \gamma E_{(s'|s,a)\sim p}[{}_{a';\beta_{k+1}}\tilde{Q}_k(s',a')] \\
&\leq r(s,a) + \gamma E_{(s'|s,a)\sim p}[\max_{a'} \tilde{Q}_k(s',a')] \\
&= \mathcal{T}[\tilde{Q}_k](s,a) \leq \mathcal{T}[Q_k](s,a) = Q_{k+1}(s,a),
\end{aligned} \tag{4.7}$$

and likewise

$$\begin{aligned}
\tilde{Q}_{k+1}(s,a) &\geq \mathcal{T}[\tilde{Q}_k](s,a) - \tfrac{\gamma}{\beta_{k+1}} \log|\mathcal{A}| \\
&\geq \mathcal{T}\left[Q_k - \log|\mathcal{A}| \sum_{i=1}^{k} \frac{\gamma^{k-i}}{\beta_i}\right](s,a) - \tfrac{\gamma}{\beta_{k+1}} \log|\mathcal{A}| \\
&= Q_{k+1}(s,a) - \log|\mathcal{A}| \sum_{i=1}^{k+1} \frac{\gamma^{k+1-i}}{\beta_i}.
\end{aligned} \tag{4.8}$$

□

Lemma 4.2.2. *Repeated application of the soft Bellman update $\mathcal{T}_{\beta_k}$ with a convergent sequence of inverse temperatures $\epsilon \leq \beta_k \xrightarrow[k\to\infty]{} \infty$ converges to the (unique) fixed point of the classic Bellman update $\mathcal{T}$.*

Proof. $\mathcal{T}$ is a γ-contraction, and $Q_k = \mathcal{T}[Q_{k-1}]$ converges to its unique fixed point [Sutton and Barto, 2018]. To have $\tilde{Q}_k$ converge to the same fixed point, we show that the lower and upper bounds in Theorem 4.2.1 converge to the same value, by proving that

$$f(k) = \sum_{i=1}^{k} \frac{\gamma^{k-i}}{\beta_i} \tag{4.9}$$

converges to 0 as $k \to \infty$.

Since $\beta_k \to \infty$, for any $M \in R$, there exists $j(M)$ such that $\forall i \geq j(M)$, $\beta_i \geq M$. For any $\delta > 0$, let $M = \frac{2}{(1-\gamma)\delta}$. Observe that

$$\sum_{i=j(M)}^{k} \frac{\gamma^{k-i}}{\beta_i} \leq \sum_{i=j(M)}^{k} \frac{\gamma^{k-i}}{M} = \frac{1-\gamma^{j(M)}}{(1-\gamma)M} < \frac{\delta}{2}. \tag{4.10}$$

Now for any $k \geq j(M) - 1 + \log_\gamma \frac{(1-\gamma)\epsilon\delta}{2}$

$$\sum_{i=1}^{j(M)-1} \frac{\gamma^{k-i}}{\beta_i} \leq \sum_{i=1}^{j(M)-1} \frac{\gamma^{k-i}}{\epsilon} = \frac{\gamma^{k-j(M)+1}}{(1-\gamma)\epsilon} \leq \frac{\delta}{2}. \tag{4.11}$$

Putting the above together, we find that for any $\delta > 0$ there exists k_0 with

$$k_0 = j\left(\tfrac{2}{(1-\gamma)\delta}\right) - 1 + \log_\gamma \tfrac{(1-\gamma)\epsilon\delta}{2}, \tag{4.12}$$

such that for any $k \geq k_0$ we have $f(k) \leq \delta$, as required for $f(k) \to 0$.

$\square$

There are two differences between the premise of Theorem 4.2.2 and practical StaTeS-SQL. First, in StaTeS-SQL, the soft Bellman operator is not applied in all states–action pairs. Instead, the current value estimate is updated toward the soft Bellman target in sampled states and actions. In the tabular case, with an appropriate scheduling of the learning rate, standard results in stochastic optimization Jaakkola et al. [1994] can be applied to show almost sure (with probability 1) convergence to the same limit.

Second, in StaTeS-SQL, for the temperature to converge to 0, every state–action pair must be updated infinitely often. Under stochastic exploration and sampling of experience from a replay buffer, starvation of some state–action pairs is possible, but with sufficient exploration convergence is almost surely guaranteed. This observation leads to a straightforward extension of the results of Theorem 4.2.2 to state-dependent inverse temperatures.

Chapter 5

Density Model

5.1 Generating pseudo-counts

Directly recording the number of times that the value update has been applied to state s is not useful in practical settings, where the state space is large, and most states will rarely be visited more than once. Moreover, a tabular mapping from s to $n(s) = \sum_a n(s, a)$ would fail to capture the similarity between different states that a Q function approximator does leverage, and would vastly underestimate the effective number of times that $Q(s, a)$ has been updated in states similar to s.

Instead, we use a pseudo-count method derived by Bellemare et al. [2016] from a simplified pixel-level CTS density model [Bellemare et al., 2014]. We denote by $\rho : \mathcal{S} \to R$ a density model on the state space $\mathcal{S}$. Let $\rho_k(s)$ be the probability assigned by the model to $s \in \mathcal{S}$ after k updates of the model. Let $\rho'_k(s)$ be the probability that the model would assign to s if it were updated on s one more time. The pseudo-count [Bellemare et al., 2016] can then

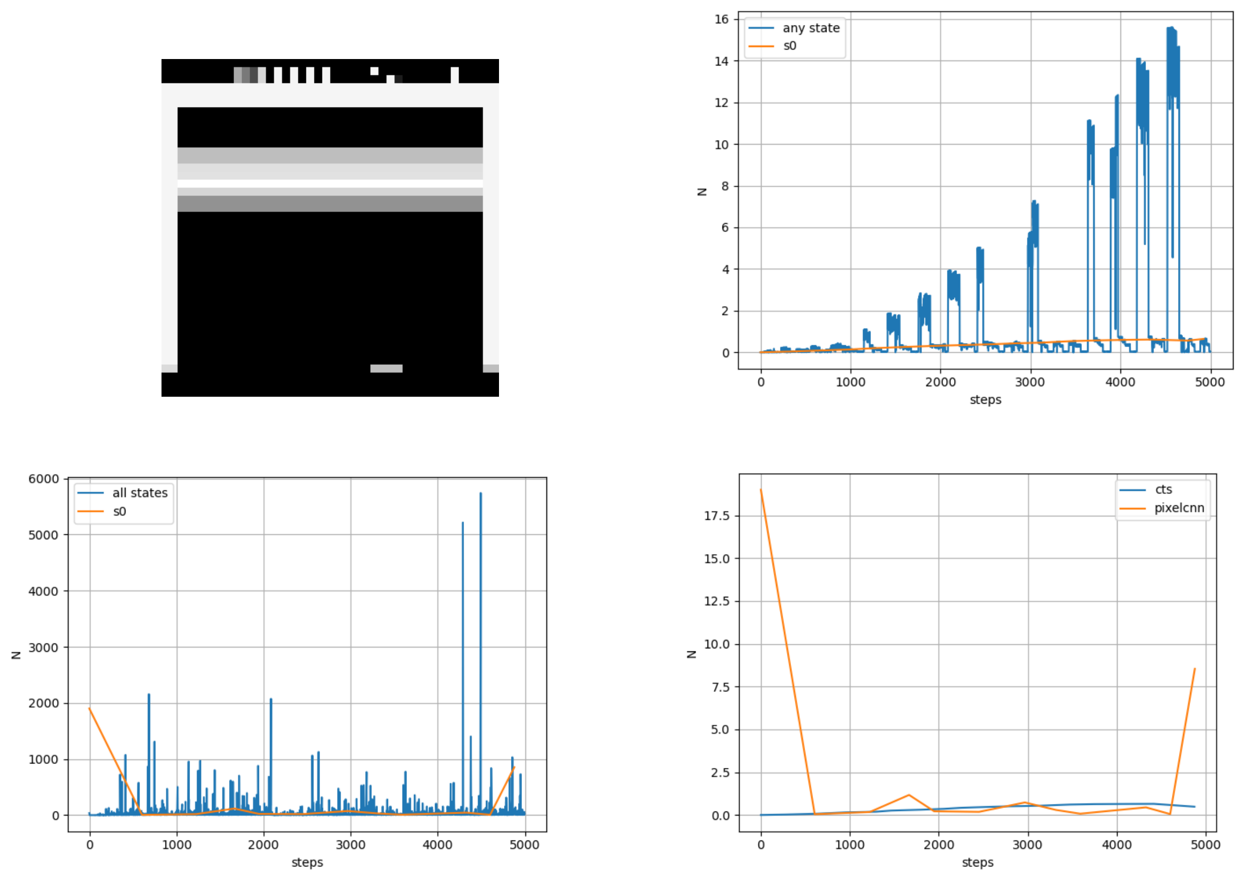

Figure 5.1: Pseudo-counts generated with two density models, CTS and PixelCNN. We generate the pseudo-count for all states in the trajectory under a random policy for 5k steps, and also track (a) the initial state s_0 for the atari game Breakout and monitor the behavior of (b) CTS pseudo-counts, which consistently increase for early states in each episode, and (c) PixelCNN pseudo-counts, which are less consistent. Both density models show an increase in the pseudo-count over time, but (d) PixelCNN displays much more fluctuation than CTS over the pseudo-count of the initial state, suggesting that CTS suffers less from the effects of catastrophic forgetting.

be defined as

$$n_k(s) = \frac{\rho_k(s)(1 - \rho'_k(s))}{\rho'_k(s) - \rho_k(s)}. \tag{5.1}$$

Defining the prediction gain

$$PG_k(s) = \log \rho'_k(s) - \log \rho_k(s), \tag{5.2}$$

the pseudo-count can be approximated by

$$n_k(s) \approx \left(\exp(PG_k(s)) - 1\right)^{-1}. \tag{5.3}$$

Note that k is different from $n(s)$ in the last section and represents the total number of updates in all states. The effective number of times that state s has been updated is its pseudo-count $n_k(s)$, suggesting the state-dependent inverse-temperature

$$\beta(s) = \kappa \cdot n_k(s) + \epsilon. \tag{5.4}$$

Initialize Q network parameters θ Initialize target Q network $\overline{\theta} \leftarrow \theta$ Initialize an empty replay buffer $\mathcal{D} \leftarrow \emptyset$ Initialize a density model ρ

each iteration each step t in the rollout In state s_t, sample action a_t from the ϵ-greedy policy for $Q_\theta(s_t, \cdot)$Execute action a_t and observe reward r_t and new state s_{t+1} Store the transition (s_t, a_t, r_t, s_{t+1}) into the replay buffer $\mathcal{D}$ each gradient step Sample random batch (s, a, r, s') from $\mathcal{D}$ $\beta \leftarrow \left(\exp(PG(s')) - 1\right)^{-1}$ $y = r + \gamma_{a';\beta} Q_{\bar{\theta}}(s', a')$ Perform gradient descent on $(y - Q_\theta(s, a))^2$ Update the density model with state s Every `target_freq` steps, update $\bar{\theta} \leftarrow \theta$

5.2 Modelling pixel observations

Bellemare et al. [2016] suggest using a simplified pixel-level version of the CTS model [Bellemare et al., 2014] to learn a pseudo-count for exploration. The model takes a 2D image input and applies a location-dependent L-shaped filter to calculate the probability. It exhibits fast learning speed and produces pseudo-counts that tend to grow linearly on average with real counts, but its computation time does not scale well.

StaTeS-SQL can be used with any density model. Alternatives to CTS include neural generative models, which have shown promising performance in image processing. One such algorithm is PixelCNN [Oord et al., 2016], a convolutional neural network structure for modelling pixel density. Ostrovski et al. [2017] adapted PixelCNN to generate a pseudo-count for online exploration in DQN. Count-based exploration allows training the density model online, such that pseudo-counts and updates are computed for the same states, and $\rho_{k+1}(s) = \rho'_k(s)$. This removes the need to revert the extra update when calculating the prediction gain, which is crucial for computationally intensive density models such as a PixelCNN.

Two aspects should be considered when selecting a suitable density model for pseudo-count calculation:

Computational efficiency. Most MaxEnt RL methods, including SQL and SAC, are off-policy methods, in which pseudo-counts are needed for the following state s' when updating $Q(s, a)$. Because the density model is updated for s but queried on s', it cannot reuse ρ'_k to more efficiently update ρ_{k+1}, increasing the preference for faster density models, such as CTS.

Reducing catastrophic forgetting. Catastrophic forgetting occurs when function approximation methods lose knowledge of a previously learnt task when training on a new task. As the exploration policy changes throughout training, the density model is trained on a changing state distribution, which can lead it to forget the density of states that decline in frequency. This can lead to instability in the pseudo-count when density model is used to represent the state distribution in value updates, inducing targets whose maximization is too hard or too soft. Because CTS is a shallow autoregressive model, it provides more stable pseudo-counts (See Figure 5.1).

In this paper, we use the CTS density model, which is favored by these considerations over more expressive density models.

Chapter 6

Experiments

DQN represents a principled and powerful approach to RL, and in recent years more extensions to it have been proposed that greatly improve its performance. Interestingly, SQL and StaTeS-SQL can be combined with many of these extensions, allowing us to compare these methods after integrating popular DQN extensions. We integrate StaTeS-SQL with Rainbow DQN [Hessel et al., 2018], a state-of-the-art reinforcement learning algorithm for memoryless agents, which includes multi-step targets [Sutton and Barto, 2018], double Q-learning [Hasselt, 2010], prioritized experience replay [Schaul et al., 2015], dueling networks [Wang et al., 2016], distributional RL [Bellemare et al., 2017], and noisy networks [Fortunato et al., 2017]. All of these methods can be straightforwardly applied to soft Q-learning, with the exception of multi-step targets and distributional RL, which we discuss next.

Multi-step learning. Multi-step targets with a well-tuned number of steps n can lead to faster learning in on-policy RL algorithms [Sutton and Barto, 2018] by trading off the bias and variance of the return estimates [Kearns and Singh, 2000]. The n-step truncated return

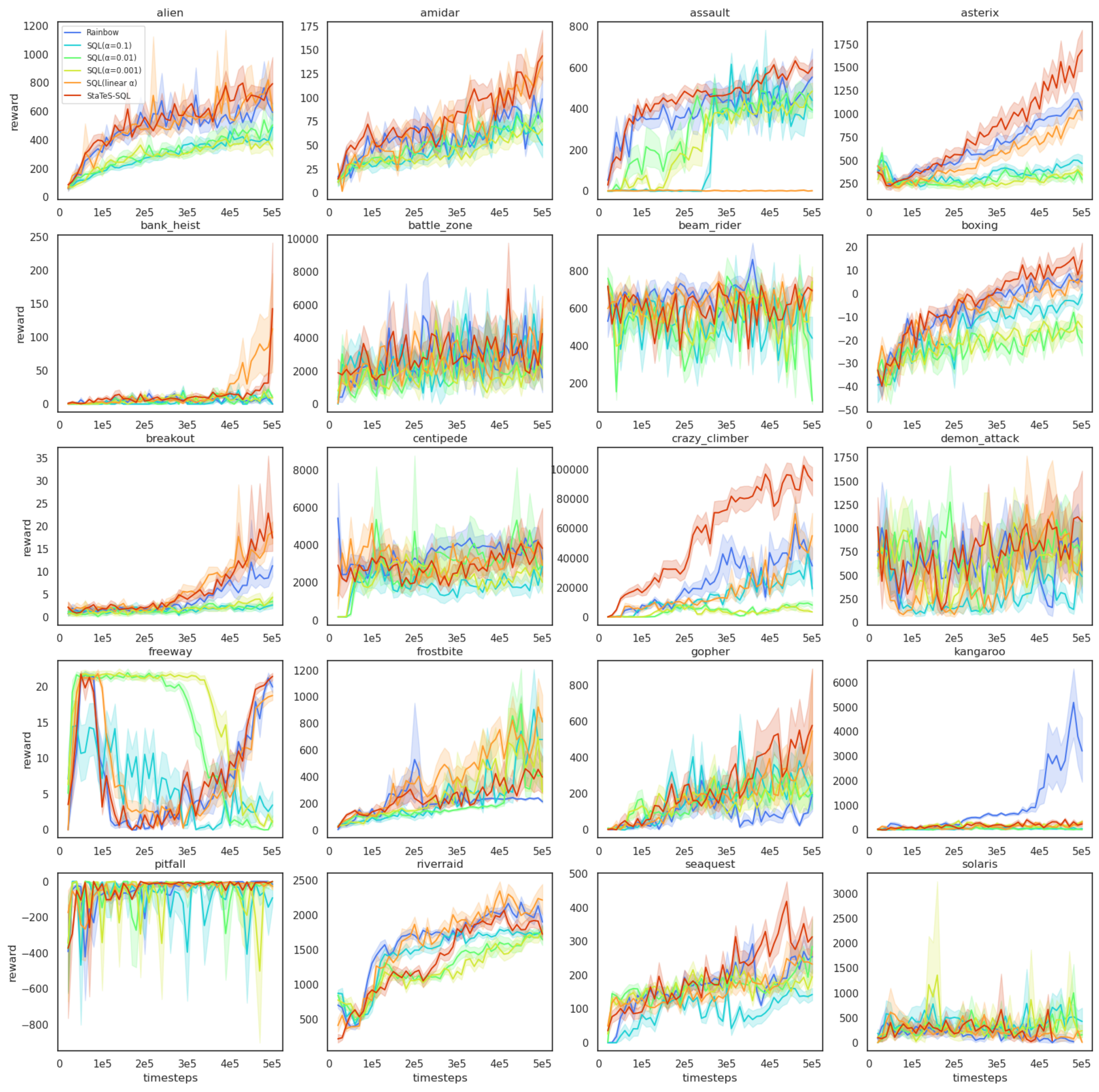

Figure 6.1: Learning curves for Rainbow, SQL with fixed temperatures and Rainbow over 500k timesteps, for each individual game. Every curve is smoothed with a moving average of 50 steps over 5 runs for readability.

at time t is

$$r_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k r(s_{t+k}, a_{t+k}). \tag{6.1}$$

Hessel et al. [2018] define a multi-step variant of DQN by minimizing the alternative loss

$$(r_t^{(n)} + \gamma^n \max_{a \in \mathcal{A}} Q_{\bar{\theta}}(s_{t+n}, a) - Q_\theta(s_t, a_t))^2 \tag{6.2}$$

and demonstrate empirically that n-step targets can outperform single-step targets in DQN, despite the off-policy experience providing biased estimates of the n-step return. In SQL with inverse-temperature β, the n-step truncated return suggested by the soft Bellman operator (2.3) is

$$\tilde{r}_t^{(n)} = r_t^{(n)} + \tfrac{1}{\beta} \sum_{k=1}^{n-1} \gamma^k H[\pi(\cdot|s_{t+k})]. \tag{6.3}$$

Unfortunately, empirical policy entropy estimates are often very noisy, and may degrade beyond any usefulness with off-policy experience. It is also unclear which β should be used in off-policy estimates of (6.3). These considerations call for further study, and in this work we simply use 1-step returns for SQL and StaTeS-SQL.

Distributional RL. Unlike conventional RL, which estimates the expected return, distributional RL estimates the distribution of the return at time t over the stochasticity of the environment and the policy. The estimator uses a fixed N-dimensional vector z of values spaced evenly along the range $[v_{\min}, v_{\max}]$ of possible returns. The distribution of $Q(s_t, a_t)$ is then represented by a categorical distribution $p_\theta(s_t, a_t)$ over the values of z. Distributional DQN updates $p_\theta(s, a)$ by minimizing its KL-divergence from a projected target distribution induced by the categorical distribution $p_{\bar{\theta}}(s', a^*)$ over the values $r + \gamma z$. Here the action a^* is chosen greedily with $a^* = \arg\max_{a'} z_{\bar{\theta}}^p(s', a')$. We adapt distributional RL to soft Q-learning and StaTeS-SQL by defining a policy distribution of

$$\pi(a'|s') = \frac{\exp \beta(s') z_{\bar{\theta}}^p(s', a')}{\sum_{\bar{a}'} \exp \beta(s') z_{\bar{\theta}}^p(s', \bar{a}')}$$

and, on experience (s, a, r, s'), minimize the KL-divergence of $p_\theta(s, a)$ from a projected target distribution induced by the categorical distribution $E_{(a'|s')\sim\pi}[p_{\bar{\theta}}(s', a')]$ over the values $r + \gamma\left(z + \frac{1}{\beta}H[\pi(\cdot|s')]\right)$.

The Atari Learning Environment was proposed as a challenge problem for evaluating general, domain-independent artificial intelligence methods [Bellemare et al., 2013]. We train the Rainbow variations of SQL and StaTeS-SQL as described above over 500K frames for 20 popular Atari games and compare with Rainbow DQN (See Figure 6.1). StaTeS-SQL outperforms Rainbow in 18 out of 20 environments, and outperforms SQL with different inverse temperatures in 14 out of 20 games. We average human-normalized performance on these environments (excluding solaris and riverraid due to lack of related data from Mnih et al. [2015]) and show that StaTeS-SQL's performs significantly better than that of Rainbow DQN**??**.

Among all atari experiments, we follow common modifications to the environments including frame-stacking, reward-clipping and grey-scaling as described in Mnih et al. [2013]. The selections of of our hyper-parameters are consistent with Rainbow DQN [Hessel et al., 2018], and we provide more details on these hyper-parameters in Table A.2.

Chapter 7

Conclusion

In this paper, we present a simple method for temperature scheduling in soft Q-learning which could potentially be applied to other maximum entropy reinforcement learning algorithms. We prove theoretically that our method converges to the optimum in the tabular case and discuss how density models can be used to schedule the temperature in non-tabular cases. Empirical results suggest that StaTeS-SQL can outperform Rainbow DQN and SQL with fixed temperatures on Atari 2600.

A interesting open question is how to efficiently adapt state-dependent temperature scheduling method to high-dimensional continuous control environments. Extending these methods to high-dimensional continuous observation spaces presents more challenges, as CTS models scale poorly to high dimensions, and pixel-based models are not generally adaptable to general observation spaces.

Tang et al. [2016] proposed using an auto-encoder to learn meaningful hash codes to provide desirable semantic similarity and achieved good performance in exploration for continuous control tasks, and a similar approach could benefit state-dependent temperature scheduling in high-dimensional observation spaces. Latent-space generative models, such as normalizing flows Dinh et al. [2017], which have recently been proposed for modelling high-dimensional highly structured data, may also be applicable to value-based reinforcement learning.

Bibliography

Kavosh Asadi and Michael L Littman. An alternative softmax operator for reinforcement learning. In *International Conference on Machine Learning*, pages 243–252. PMLR, 2017.

Marc Bellemare, Joel Veness, and Erik Talvitie. Skip context tree switching. In *International conference on machine learning*, pages 1458–1466. PMLR, 2014.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying count-based exploration and intrinsic motivation. *CoRR*, abs/1606.01868, 2016. URL `http://arxiv.org/abs/1606.01868`.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. *CoRR*, abs/1707.06887, 2017. URL `http://arxiv.org/abs/1707.06887`.

Petros Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp, 2017.

Benjamin Eysenbach and Sergey Levine. Maximum entropy rl (provably) solves some robust rl problems. *arXiv preprint arXiv:2103.06257*, 2021.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

Roy Fox. Toward provably unbiased temporal-difference value estimation. In *Optimization Foundations for Reinforcement Learning Workshop at NeurIPS*, 2019.

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562*, 2016.

Jordi Grau-Moya, Felix Leibfried, and Peter Vrancx. Soft q-learning with mutual-information regularization. In *International conference on learning representations*, 2018.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pages 1352–1361. PMLR, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23: 2613–2621, 2010.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Tommi Jaakkola, Michael I Jordan, and Satinder P Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural computation*, 6(6):1185–1201, 1994.

Edwin T Jaynes. *Probability theory: The logic of science*. Cambridge university press, 2003.

Michael J Kearns and Satinder P Singh. Bias-variance error bounds for temporal difference updates. In *COLT*, pages 142–147. Citeseer, 2000.

Seungchan Kim, Kavosh Asadi, Michael Littman, and George Konidaris. Deepmellow: removing the need for a target network in deep q-learning. In *Proceedings of the Twenty Eighth International Joint Conference on Artificial Intelligence*, 2019.

Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*, pages 6131–6141. PMLR, 2021.

Felix Leibfried, Jordi Grau-Moya, and Haitham Bou-Ammar. An information-theoretic optimality principle for deep reinforcement learning. *arXiv preprint arXiv:1708.01867*, 2017.

Litian Liang, Yaosheng Xu, Stephen McAleer, Dailin Hu, Alexander Ihler, Pieter Abbeel, and Roy Fox. Temporal-difference value estimation via uncertainty-guided soft updates, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. *arXiv preprint arXiv:1702.08892*, 2017a.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Trust-pcl: An off-policy trust region method for continuous control. *arXiv preprint arXiv:1707.01891*, 2017b.

Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*, 2016.

Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *International conference on machine learning*, pages 2721–2730. PMLR, 2017.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. Approximate inference and stochastic optimal control. *arXiv preprint arXiv:1009.3958*, 2010.

Jonathan Rubin, Ohad Shamir, and Naftali Tishby. Trading value and information in mdps. In *Decision Making with Imperfect Decision Makers*, pages 57–74. Springer, 2012.

Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19): 70–76, 2017.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. #exploration: A study of count-based exploration for deep reinforcement learning. *CoRR*, abs/1611.04717, 2016. URL `http://arxiv.org/abs/1611.04717`.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

Yaosheng Xu, Dailin Hu, Litian Liang, Stephen McAleer, Pieter Abbeel, and Roy Fox. Target entropy annealing for discrete soft actor-critic. *arXiv preprint arXiv:2112.02852*, 2021.

Appendix A

Hyperparameters

Hyper-parameter	value
Discount factor γ	0.99
Exploration ϵ	0.001
Learning rate	1

Table A.1: Hyper-parameters for tabular experiments.

Hyper-parameter	value
Grey-scaling	True
Observation down-sampling	(84, 84)
CTS model down-sampling	(42, 42)
Frames stacked	4
Reward clipping	[-1, 1]
Discount factor γ	0.99
Learning rate	0.0000625
Replay buffer size	1000000
Minibatch size	32
Q network channels	[32, 32, 64]
Q network filter size	$8 \times 8, 4 \times 4, 3 \times 3$
Q network stride	4, 2, 1
Q network hidden units	512
Noisy net σ_0	0.5
Multi-step returns n	4 for Rainbow-DQN, 1 for SQL and StaTeS-SQL
Distributional atoms	51
Distributional min/max values	[-10, 10]

Table A.2: Hyper-parameters for Rainbow, SQL and StaTeS-SQL with Rainbow Integration on Atari 2600. The values of all the hyper-parameters are based on the work from Hessel et al. [2018].

Appendix B

Training Data

Game	Human	Random	Rainbow	SQL($\beta = 10$)	SQL($\beta = 100$)	SQL($\beta = 1000$)	SQL(linear β)	StaTeS-SQL
alien	6875	227.8	593.33 (±29.56)	490.25 (±49.46)	505.0 (±64.24)	336.67 (±26.36)	605.0 (±18.99)	**794.0** (±83.48)
amidar	1676	5.8	98.8 (±14.25)	50.35 (±7.0)	74.88 (±5.54)	67.1 (±6.44)	137.4 (±12.17)	**144.0** (±13.9)
assault	1496	222.4	554.4 (±85.2)	440.48 (±31.26)	393.6 (±20.05)	486.0 (±21.97)	2.52 (±1.39)	**601.23** (±23.56)
asterix	8503	210	1048.0 (±53.5)	468.0 (±41.12)	298.0 (±25.08)	344.0 (±42.46)	1038.0 (±56.96)	**1687.5** (±118.15)
bank$_heist$	734.4	14.2	0.0 (±0.0)	0.0 (±0.0)	10.0 (±0.0)	9.33 (±2.67)	112.67 (±39.47)	**143.0** (±48.79)
battle_zone	37800	2360	1600.0 (±476.1)	3400.0 (±686.38)	3500.0 (±772.9)	2150.0 (±715.52)	**5150.0** (±740.82)	4266.67 (±746.21)
beam$_rider$	5775	363.9	477.6 (±38.2)	440.0 (±66.73)	105.6 (±10.78)	**742.4** (±43.29)	700.32 (±25.52)	693.6 (±32.75)
boxing	4.3	0.1	5.1 (±2.38)	-0.2 (±2.06)	-21.16 (±3.02)	-14.44 (±2.6)	8.0 (±2.38)	**14.32** (±3.76)
breakout	31.8	1.7	11.28 (±1.09)	2.62 (±0.4)	3.42 (±0.56)	4.32 (±0.64)	18.37 (±1.33)	**18.7** (±1.48)
centipede	11963	2091	3252.8 (±208.21)	2882.2 (±612.04)	3017.4 (±310.5)	2025.8 (±309.4)	2921.93 (±478.92)	**3836.6** (±1014.18)
crazy$_climber$	35411	10781	34640.0 (±6421.53)	19140.0 (±2940.53)	8070.0 (±1312.59)	3400.0 (±0.0)	55210.0 (±8604.46)	**92568.0** (±5131.56)
demon$_attack$	3401	152.1	554.0 (±158.61)	489.0 (±156.83)	724.0 (±212.52)	862.0 (±176.31)	754.0 (±203.46)	**1071.0** (±320.14)
freeway	29.6	0	20.0 (±0.39)	3.47 (±1.0)	1.25 (±0.38)	1.05 (±0.28)	18.8 (±0.37)	**21.48** (±0.26)
frostbite	4335	65.2	214.0 (±6.78)	680.5 (±84.8)	419.6 (±68.54)	286.4 (±55.8)	**812.8** (±108.85)	398.0 (±77.34)
gopher	2321	257.6	196.0 (±51.92)	184.8 (±34.53)	207.2 (±49.46)	272.0 (±53.27)	548.0 (±169.85)	**578.67** (±161.88)
kangaroo	3035	52	**3210.0** (±687.71)	50.0 (±19.87)	10.0 (±10.0)	330.0 (±39.14)	312.0 (±41.76)	240.0 (±62.34)
pitfall	13513	1339	-15.95 (±7.5)	-91.07 (±21.57)	0.0 (±0.0)	-26.8 (±11.29)	-27.16 (±8.43)	**0.0** (±0.0)
riverraid	-	-	1905.0 (±53.88)	1784.5 (±40.91)	1652.5 (±43.77)	1782.86 (±36.12)	**2220.8** (±114.26)	1729.6 (±48.04)
seaquest	20182	68.4	255.0 (±27.94)	143.0 (±9.43)	285.33 (±40.78)	196.0 (±11.9)	232.8 (±20.42)	**314.0** (±32.71)
solaris	-	-	8.2 (±4.2)	**476.0** (±120.25)	160.0 (±29.21)	240.0 (±69.54)	9.6 (±5.31)	192.0 (±32.52)

Table B.1: The performance of Rainbow, fixed-temperature SQL ($\alpha \in \{0.1, 0.01, 0.001\}$) and StaTeS-SQL on Atari 2600 games, average total rewards over 5 runs after training for 500K timesteps compared with HumanMnih et al. [2015] and random policy performances (averaged over 100 episodes). Bold numbers indicate the best results besides human performance. StaTeS-SQL outperforms Rainbow in 18 out of 20 games and outperforms SQL with different inverse temperatures in 14 out of 20 games.