Optimal Swarm RL: An Improved Deep Exploration Strategy

A B. Tech Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Rishav Chourasia
(140101086)

under the guidance of

Dr. Rashmi Dutta Baruah

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**Optimal Swarm RL: An Improved Deep Exploration Strategy**" is a bonafide work of **Rishav Chourasia** (**Roll No. 140101086**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Dr. Rashmi Dutta Baruah**

Assistant Professor,

Apr, 2018

Department of Computer Science & Engineering,

Guwahati.

Indian Institute of Technology Guwahati, Assam.

Acknowledgements

Foremost I would like to thank Prof. Rashmi Dutta Baruah for her patience, motivation and time. Her guidance helped me in my research and in writing this thesis. My sincere thanks goes to my department professors for getting me interested in research. I thank my fellow batch mates for indulging me in insightful and stimulating discussions, for working together before deadlines and for all the fun we had. Last but not the least, I would like to thank my family for supporting me spiritually throughout my life.

Contents

List of Figures

List of Tables

Abstract

The proper balance between exploration and exploitation has been a deeply researched topic in literature, and some interesting ways of achieving it [McF] have been proposed. Some of the recently established technique achieves it via use of an Ensemble of Q-function approximators [OBPVR16]. Instead of taking exploratory actions by using Boltzman or Gibbs strategy, manifesting exploratory behaviour via model parameters directly has rightly gained considerable attention recently [SOR+10]. For Bootstrapped DQN, deep exploration is achieved based on the random initialization of an ensemble of function approximators' weights. Based on similar lines, other kinds of ensemble algorithms exists [WvH08] that attempt to improve exploration-exploitation balance. We attempt to come up with an optimal exploration strategy in terms of expected regret, for a class of RL algorithms we call Swarm RL. Some notable algorithms including Bootstrapped DQN and Ensemble Voting DQN belong to this class, implying that the optimal exploration strategy effectively achieves better exploration-exploitation balance. For the empirical comparison, we have presented a comparison on standard 49 ALE environments.

Chapter 1

Introduction

Various exploration strategy like parameter perturbation, Boltzmann strategy, ϵ-greedy strategy are unable to cope up with reinforcement learning environments with considerably sparse reward distribution. Techniques to enable effective explorations in these conditions have been proposed, such as maintaining a probability distribution on actions for handling uncertainty like Bayesian Q-learning[DFR98] or maintaining a probability distribution on MDPs itself like in PSRL[OVRR13]. These however suffer form the curse of dimentionality, and lead to intractable solutions for most real life RL environments. One efective way of handling intractability in maintaining probabilistic distribution over MDPs is to maintain a sample(ensemble) of Q-function approximators instead. Bootstrapped DQN uses this methodology and have shown significant increase in performance which is tantamount to a decrease in net expected regret of the algorithm.

The reason for an improvement, as stated in their work, was due to information exchange between ensemble heads, via a shared memory buffer. Other ensemble algorithms also achieve sharing via some form of communication or another. Averaged DQN[ABS16], for instance proposes sharing of learning via averaging of ensemble head estimates for error calculation. Voting ensemble DQN[WvH08] suggests sound action decision with majority consensus as the mode of learning exchange. An important question arises here is how to measure benefit of

shared learning. In our opinion, the best way to measure benefit due to information sharing is via estimating the net ensemble regret accrued in the whole duration of training for the RL algorithm.

Based on method for error calculation and action decision for training an ensemble using heads' estimates, a class of algorithms can be formulated with guaranteed convergence as well as preserved optimality[BOG+16] provided the update strategy has some improvement property. Ensemble algorithms such as Voting ensemble DQN and Bootstrapped DQN can be verified to belong in this class of RL algorithms, we call Swarm RL. All such algorithm employ sharing of knowledge for deep exploration[MR17].

Among Swarm RL algorithms, we are interested in is Optimal Swarm RL algorithm, the one that achieves best exploration, and consequently has the least expected regret overall. By developing regret bounds on an arbitrary Swarm RL algorithm, important insights on what constitutes as the best algorithm in this class can be developed. Moreover, it can help in determining the number of ensemble heads to employ for best performance guarantees. Following the results, ensembles can be trained effectively to perform considerably better than some known Ensemble learning algorithms.

Our intuition is that in Optimal Swarm RL, an effective communication between heads would be established by colluding similar agents into sets, and training them using in-sync next-actions for error computation. Moreover, these collusions would be different in each training step and will depend on the agents' net returns estimates.

1.1 Organization of The Report

After a brief background on notation 2.1, Exact Set Cover problem 2.2 and bootstrapped DQN 2.3(we use same architecture for efficiency), we introduce a swarm update strategy 3 for a set of agents with a provably convenient property of being asymptotically not worse than each of the present agent policies. Then using the constraints of the update strategy, we formalize a class of Swarm RL algorithms 4, and prove their asymptotic Q-function's convergence to Bellman update strategy's converged Q-function. Next, we compute a bound on step-wise ensemble-regret bound

and show it's dependence 5 on sizes of agent agglomeration generated. Using it we develop a clustering mechanism which reduces to optimal set cover 5. Using an approximate optimal greedy set-cover, we present our Optimal Swarm DQN algorithm 6 and show it's results for 49 standard ALE Atari environments 7.

Chapter 2

Notation and Background Concepts

2.1 Notation

We consider a typical RL environment as a Markov Decision Process(MDP) $M := (S, A, T, R, \gamma)$, where S is the state space, A is the action space and γ is the discount factor belonging in the range $[0, 1)$. For any $s, s' \in S$ and $a \in A$, $R(s, a)$ gives the finite reward of taking action a at state s and $T(s'|s, a)$ gives the environment's transition probability of going to state s' on taking action a at state s. Without any loss in generality, we assume that the range of the bounded reward function R is $[0, 1 - \gamma]$ to make the maximum discounted returns independent of γ. Any other bounded reward function can always be linearly scaled down to this range without affecting the action decisions. We also denote the space for all bounded functions with range $[0, 1]$ on $S \times A$ and on S as $\mathcal{Q} := \mathcal{Q}_{S,A}$ and $\mathcal{V} := \mathcal{V}_S$ respectively. This bound on range reflects the bound on net returns any policy can get when facing R reward function.

A mapping $\pi : S \to A$ is referred to as a deterministic policy. Every policy π has a corresponding Q function $Q^\pi \in \mathcal{Q}$ computed from the Bellman equation

$$Q^\pi(s, a) = R(s, a) + \gamma E_{s' \sim T(\cdot|s,a)}[Q^\pi(s', \pi(s')]. \tag{2.1}$$

From now on, we'll represent $E_{s' \sim T(\cdot|s,a)}$ as E_T for convenience. This $Q^\pi(s, a)$ function is interpreted

as the expected discounted net returns an agent receives on starting with action a on state$s$ and following policy π thereafter. The value function

$$V^{\pi}(s) = Q^{\pi}(s, \pi(s)) \tag{2.2}$$

can similarly be interpreted as the discounted net returns on following policy π right from the first state s. An optimal policy π^* is defined as a policy mapping that accrues maximum expected discounted returns on MDP M, whose Q-table Q^* was shown (Bertsekas and Tsitsiklis 1996) to be the unique identity solution of the Bellman operator $\mathcal{T}|\mathcal{Q} \rightarrow \mathcal{Q}$ defined point-wise as

$$\mathcal{T}Q(s, a) = R(s, a) + \gamma E_T[\max_b Q(s', b)]. \tag{2.3}$$

Starting from any Q function $Q_0 \in \mathcal{Q}$, a sequence of application of $\mathcal{T}$ leads to convergence to the optimal Q function Q^*[K$^+$12] i.e.

$$Q^*(s, a) = \lim_{t \rightarrow \infty} \mathcal{T}^t Q_0(s, a). \tag{2.4}$$

and corresponding value function i.e.

$$V^*(s) = \max_a Q^*(s, a). \tag{2.5}$$

Dealing with an ensemble of K agents, we denote the initial Q-functions as $Q_0^{(i)} \in \mathcal{Q} \quad \forall i \in \mathcal{K} := \{1, 2..K\}$. Let $H_t = (s_0, a_0, R(s_0, a_0), s_1, a_1, ..., s_{t-1}, a_{t-1}, R(s_{t-1}, a_{t-1}))$ denote the history of observations made before t^{th} update step. Depending on the RL algorithm, say represented by the operator $\mathcal{T}'$, the returns estimates $Q_t^{(i)}, \forall i \in \mathcal{K}$ at t^{th} step for a history H_t, could be said to have been sampled from the posterior distribution over $\mathcal{Q}$[OVRR13][AJ17], i.e.

$$Q_t^{(i)}|H_t \sim P_{\mathcal{T}'}(Q|H_t) \quad \forall i \in \mathcal{K}. \tag{2.6}$$

For a distribution on MDPs, respective Q^* and V^* are stochastic and follow some distribution depending upon M's distribution. For such a stochastic environment, we denote ensemble regret $Regret_{\pi^{(\cdot)}}$ for an ensemble policy $\pi^{(\cdot)}$ as

$$Regret_{\pi^{(\cdot)}}(s) = \sum_{i \in \mathcal{K}} \sum_{a \in A} \mathbb{1}\{\pi^{(i)}(s) = a\} \Delta_{s,a}, \tag{2.7}$$

where $\pi^{(i)}$ is the policy for agent i and $\Delta_{s,a}$ is the action regret given by

$$\Delta_{s,a} = E[V^*(s) - Q^*(s,a)], \tag{2.8}$$

and $\Delta_{s,a} \in [-1, 1]$ because of range of any $Q \in \mathcal{Q}$.

2.2 Exact Set Cover

Given a collection $\mathcal{S}$ of subsets of a universe set $\mathcal{U}$, a subset $C \subset \mathcal{S}$ that covers all the elements of the universe $\mathcal{U}$ is called an exact cover of the collection $\mathcal{S}$. Mathematically, a valid cover is expressed as

$$\mathcal{U} = \bigcup_{X \in C} X, \quad C \subset \mathcal{S}. \tag{2.9}$$

An optimal set cover is defined as a cover C^* such that for any other cover C, $|C^*| <= |C|$. The problem of finding optimal exact set cover has been shown to be in NP-complete. However, many polynomial time approximate algorithms exists that can be used when finding a close to optimal cover suffices the requirements.

In our algorithm presented later, the task of clustering of agents at each time step essentially reduces to finding exact cover for universe $\mathcal{U} = \mathcal{K}$, and collection $\mathcal{S}$ consisting of a set of agents for all action $a \in A$, i.e. $|\mathcal{S}| = |A|$. Though for our examples, the sizes are conveniently small to use exponential time solutions to the set cover problem, we nonetheless incorporate an efficient implementation of $log(|\mathcal{U}|)$-approximate set cover algorithm, adapted from greedy set cover.

2.3 Bootstrapped DQN

An adaptation of Posterior Sampling for Reinforcement Learning(PSRL) for practical environments is Bootstrapped DQN. A sample of $|\mathcal{K}|$ Q-function estimates $Q_t^{(i)}$, parameterized in form of neural network heads as shown in 2.3, are maintained in place of a distribution over Q-functions like in Bayesian Q-Learning [DFR98]. A common experience buffer is used for all the heads, and experience tuples of the form $(s_t, a_t, R(s_t, a_t), s_{t+1})$ are shared among a random subset of agents by storing a probability p Bernoulli mask vector $M = (m^{(1)}, m^{(2)}.., m^{(|\mathcal{K}|)})$, where $m^{(i)} \sim Ber(p), \forall i \in \mathcal{K}$.

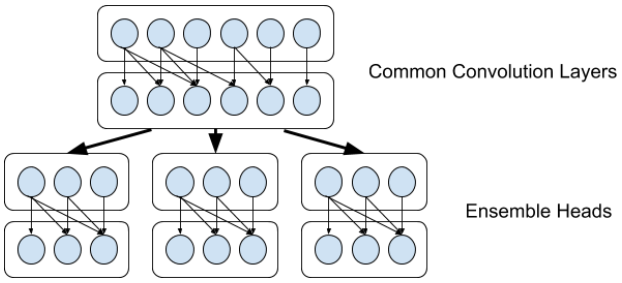

Fig. 2.1 Neural Network Architecture with $|\mathcal{K}| = 3$ heads.

In the beginning of an episode, a random neural network head is sampled and the corresponding greedy policy if followed throughout the episode to generate experience tuples, which are then randomly shared with a subset of heads using the sampled mask M. Frequently enough, batches of experience tuples are sampled from the replay and used to train the network optimizing for the following loss

$$L = E[((Y_t^{(.)} - Q_t^{(.)}(s, a)) \cdot M)^2], \tag{2.10}$$

where

$$Y_t^{(.)} = R(s, a) + \gamma \times Q_t^{(.)}(s', \pi_t^{(.)}(s')), \tag{2.11}$$

and $(s, a, R(s, a), s', M)$ is sampled from experience buffer. The major deviation of proposed Optimal Swarm DQN from bootstrapped DQN is in the selection of $\pi_t^{(.)}(s')$ for the loss term

calculation. For bootstrapped DQN,

$$\pi_t^{(i)}(s') = \underset{b}{\operatorname{argmax}}\, Q_t^{(i)}(s', b), \quad i \in \mathcal{K}. \tag{2.12}$$

Similar to Double DQN's policy update, Bootstrapped DQN's update can be stabilized in terms of overestimation by using a target and a current network, with swiping on regular intervals.

Chapter 3

Swarm Update Strategy

Instead of greedily selecting $\pi_t^{(i)}(s')$ using equation (2.12), we propose a Swarm Update Strategy for ensembles that considers the expected returns $Q_t^{(j)}$ of other agents $j \in \mathcal{K}$. By accounting for the the expectations of other agents, Swarm Update Strategy enjoys robustness due to lesser uncertainty. Any update strategy qualifies as a Swarm Update Strategy if for any state s and agent i, a subset of agents $\mathcal{I}_i(s)$ including i deems it's update policy's action $\omega_i(s)$ preferable to current policy action π_i, i.e.

$$Q_{\pi_j}(s, \omega_i(s)) \geq Q_{\pi_j}(s, \pi_j(s)), \quad \forall j \in \mathcal{I}_i(s). \tag{3.1}$$

For a valid Swarm Update strategy, it's update policy must choose from a set of preferable actions, denoted as

$$\mathcal{A}_i(s) = \{a | Q_{\pi_i}(s, a) \geq Q_{\pi_i}(s, \pi_i(s))\}. \tag{3.2}$$

Such an update strategy improves upon the current policy, as established by the following theorem.

Theorem 3.0.1 (Policy Improvement Theorem)**.** *If $\pi_i, \forall i \in \mathcal{K}$ be agent policies, then a set of swarm update policies $\Omega_i, \forall i \in \mathcal{K}$ defined such that for any $\omega_i \in \Omega_i$ and state $s \in S$, an agent*

subset $\mathcal{I}_i(s) \subset \mathcal{K}$ containing agent i exists such that

$$Q_{\pi_j}(s, \omega_i(s)) \geq Q_{\pi_j}(s, \pi_j(s)) \; \forall s \in S, j \in \mathcal{I}_i(s), \tag{3.3}$$

then it has the property that

$$Q_{\omega_i}(s, a) \geq Q_{\pi_i}(s, a) \; \forall s \in S, a \in A, i \in \mathcal{K}. \tag{3.4}$$

Proof. We will prove this using induction on the horizon depth h. For every h, we have a corresponding Q-function for following policy π_i deterministically on each step of the horizon. This is given by the equation

$$\begin{aligned} Q^h_{\pi_i}(s, a) &= R(s, a) + \gamma E_T[Q^{h-1}_{\pi_i}(s', \pi_i(s'))], \\ Q^1_{\pi_i}(s, a) &= R(s, a). \end{aligned} \tag{3.5}$$

where $s' \sim T(\cdot|s, a)$. For agent i, we represent the depth-h arbitary swarm policy ω^h_i on a subset of $\mathcal{K}$, $\mathcal{I}^h_i$ constructed such that for all $s \in S$, $i \in \mathcal{I}^h_i(s)$ and

$$Q^h_{\pi_j}(s, \omega^h_i(s)) \geq Q^h_{\pi_j}(s, \pi_j(s)), \forall j \in \mathcal{I}^h_i(s). \tag{3.6}$$

The Q-function of agent i for following the optimal policy ω^h_i on corresponding horizon depth h is represented as $Q^h_{\omega^h_i}$ and given by the equation

$$\begin{aligned} Q^h_{\omega^h_i}(s, a) &= R(s, a) + \gamma E_T[Q^{h-1}_{\omega^{h-1}_i}(s', \omega^{h-1}_i(s'))], \\ Q^h_{\omega^1_i}(s, a) &= R(s, a). \end{aligned} \tag{3.7}$$

By subtracting (3.5) from (3.7), we get

$$Q^h_{\omega^h_i}(s, a) - Q^h_{\pi_i}(s, a) = \gamma E_T[Q^{h-1}_{\omega^{h-1}_i}(s', \omega^{h-1}_i(s')) - Q^{h-1}_{\pi_i}(s', \pi_i(s'))]. \tag{3.8}$$

Induction Hypothesis: For all horizon depth h, $Q^h_{\omega_i^h}(s,a) \geq Q^h_{\pi_i}(s,a), \forall s \in S, a \in A$.

Limiting step: For h=1 case, it is clear that $Q^1_{\omega_i^1}(s,a) \geq Q^1_{\pi_i}(s,a)$, as both are equal to $R(s,a)$.

Inductive step: Assuming the induction inequality is true for depth $h-1$, we need to show that it is also true for depth h. From (3.8), the inequality for depth h is true if

$$Q^{h-1}_{\omega_i^{h-1}}(s', \omega_i^{h-1}(s')) - Q^{h-1}_{\pi_i}(s', \pi_i(s')) \geq 0, \forall s' \in S. \tag{3.9}$$

Consider two cases.

Case 1. *If $\omega_i^{h-1}(s') = \pi_i(s') = a'(say)$, then the above inequality follows from the induction hypothesis on $h-1$.*

Case 2. *If $\omega_i^{h-1}(s') \neq \pi_i(s')$, then we have*

$$Q^{h-1}_{\omega_i^{h-1}}(s', \omega_i^{h-1}(s')) \geq Q^{h-1}_{\pi_i}(s', \omega_i^{h-1}(s')), \forall s' \in S, \tag{3.10}$$

from the induction hypothesis and

$$Q^{h-1}_{\pi_i}(s', \omega_i^{h-1}(s')) \geq Q^{h-1}_{\pi_i}(s', \pi_i(s')), \forall s' \in S, \tag{3.11}$$

from the definition of agent i swarm policy ω_i^{h-1} for depth $h-1$. Combining both (3.10) *and* (3.11) *we get the required inequity* (3.9)*.*

Thus the induction hypothesis is true for all depth h. On applying $\lim_{h\to\infty}$ to both the sides of the induction hypothesis, we get

$$\lim_{h\to\infty} Q^h_{\omega_i^h}(s,a) \geq \lim_{h\to\infty} Q^h_{\pi_i}(s,a), \tag{3.12}$$

which is same as

$$Q_{\omega_i}(s,a) \geq Q_{\pi_i}(s,a). \tag{3.13}$$

□

The Swarm Update strategy introduces a family of policies and their corresponding clustering $\mathcal{I}_i(s)$ of agents based on expected estimates of net returns.

This strategy when applied on ensemble DQN adds one more level of communication among heads in comparison to Bootstrapped version, i.e. in addition to experience sharing, taking group consensus actions instead of greedy best actions allows for an improved sharing of learning.

Chapter 4

Swarm RL Algorithms

The manifestation of the Swarm update strategy as a class of RL algorithms can be expressed as a transformation Operator similar to Bellman's operator in (2.3). Following the notation in 2.1, we define Swarm operator $\mathcal{T}_t^{(i)}$ and the corresponding update for agent $i \in \mathcal{K}$ and time t as

$$Q_{t+1}^{(i)}(s,a) = \mathcal{T}_t^{(i)} Q_t^{(i)}(s,a) = R(s,a) + \gamma E_T[Q_t^{(i)}(s', \pi_t^{(i)}(s')], \tag{4.1}$$

where the $\pi_t^{(i)}$ is analogous to ω_i in sense that for a subset of agents $\mathcal{I}_t^{(i)}(s') \subset \mathcal{K}$ it satisfies $i \in \mathcal{I}_t^{(i)}(s')$ and

$$Q_t^{(j)}(s', \pi_t^{(i)}(s')) \geq Q_t^{(j)}(s', g_{t-1}^{(j)}(s')), \quad \forall j \in \mathcal{I}_t^{(i)}(s'), \tag{4.2}$$

$g_{t-1}^{(j)}$ being the greedy strategy,

$$g_{t-1}^{(j)}(s') = \underset{b}{\operatorname{argmax}}\, Q_{t-1}^{(j)}(s', b). \tag{4.3}$$

Consistent with the notation, the set of preferable actions similar to (3.2) would be

$$\mathcal{A}_t^{(i)}(s) = \{a | Q_t^{(i)}(s,a) \geq Q_t^{(i)}(s, g_{t-1}^{(i)}(s))\}. \tag{4.4}$$

It is clear from comparing Y^t in (2.11) and definition of $\mathcal{T}_t^{(i)}$ that Bootstrapped RL belongs to the set of Swarm RL algorithms because $\pi_t^{(i)}$ if defined as (2.12) satisfies the condition (4.2) and has a corresponding cluster $\mathcal{I}_t^{(i)}(s') = \{i\}$. Implication of this observation is that the best Swarm RL algorithm in terms of regret bound would be not worse than Bootstrapped RL.

Using the swarm operator definition (4.1) it can be shown that all Swarm RL algorithm converges and the asymptotic Q-function is optimal.

Theorem 4.0.1. *Swarm RL algorithms are optimal, i.e. for any random initialization $Q_0^i \in \mathcal{Q}$ and subsequent swarm operators $\mathcal{T}_t^{(i)}$, the asymptotic Q-function $\widetilde{Q}^{(i)}$ is optimal, i.e. $Q^* = \widetilde{Q}^{(i)}$, $\forall i \in \mathcal{K}$,*

$$\widetilde{Q}^{(i)}(s,a) = \lim_{t\to\infty} \mathcal{T}_t^{(i)} Q_t^{(i)}(s,a) = \lim_{t\to\infty} \mathcal{T}_t^{(i)} \mathcal{T}_{t-1}^{(i)} ... \mathcal{T}_0^{(i)} Q_0^{(i)}(s,a). \tag{4.5}$$

Proof. On subtracting (2.3) from (4.1), we get

$$\mathcal{T}_t^{(i)} Q_t^{(i)}(s,a) - \mathcal{T} Q_t^{(i)}(s,a) = \gamma E_T[Q_t^{(i)}(s', \pi_t^{(i)}(s')) - \max_b Q_t^{(i)}(s', b)]. \tag{4.6}$$

Applying limiting infimum conditions to L.H.S,

$$\begin{aligned} L.H.S &= \liminf_{t\to\infty} \mathcal{T}_t^{(i)} Q_t^{(i)}(s,a) - \liminf_{t\to\infty} \mathcal{T} Q_t^{(i)}(s,a) \\ &= \underline{\widetilde{Q}}^{(i)}(s,a) - \mathcal{T} \liminf_{t\to\infty} Q_t^{(i)}(s,a) \\ &= \underline{\widetilde{Q}}^{(i)}(s,a) - \mathcal{T} \underline{\widetilde{Q}}^{(i)} \end{aligned} \tag{4.7}$$

Applying limiting infimum conditions to R.H.S

$$\begin{aligned} R.H.S &= \liminf_{t\to\infty} \gamma E_T[Q_t^{(i)}(s', \pi_t^{(i)}(s')) - \max_b Q_t^{(i)}(s', b)] \\ &\geq \gamma E_T[\liminf_{t\to\infty} (Q_t^{(i)}(s', \pi_t^{(i)}(s'))) - \liminf_{t\to\infty} (\max_b Q_t^{(i)}(s', b))] \quad \text{(from Fatou's lemma)} \\ &= \gamma E_T[\underline{\widetilde{Q}}^{(i)}(s', \liminf_{t\to\infty} \pi_t^{(i)}(s')) - \max_b \underline{\widetilde{Q}}^{(i)}(s', b)] \\ &= \gamma E_T[\underline{\widetilde{Q}}^{(i)}(s', \operatorname*{argmax}_b \underline{\widetilde{Q}}^{(i)}(s', b)) - \max_b \underline{\widetilde{Q}}^{(i)}(s', b)] \quad \text{(from } t \to \infty \text{ on (4.2))} \\ &= 0. \end{aligned} \tag{4.8}$$

From (4.1) and (4.5) we have

$$
\begin{aligned}
\overline{\widetilde{Q}}^{(i)}(s,a) &= \limsup_{t\to\infty} \mathcal{T}_t^{(i)} Q_t^{(i)}(s,a) \\
&\leq \limsup_{t\to\infty} \mathcal{T} Q_t^{(i)}(s,a) && \text{(from comparing (2.3) and (4.1))} \\
&= R(s,a) + \gamma[\limsup_{t\to\infty} E_T[\max_b Q_t^{(i)}(s',b)]] \\
&\leq R(s,a) + \gamma E_T[\limsup_{t\to\infty} \max_b Q_t^{(i)}(s',b)] && \text{(from Jensen's inequality)} \\
&= R(s,a) + \gamma E_T[\max_b \limsup_{t\to\infty} Q_t^{(i)}(s',b)] \\
&= R(s,a) + \gamma E_T[\max_b \overline{\widetilde{Q}}^{(i)}(s',b)] \\
&= \mathcal{T}\overline{\widetilde{Q}}^{(i)}(s,a).
\end{aligned}
\tag{4.9}
$$

So we get

$$
\begin{aligned}
\underline{\widetilde{Q}}^{(i)}(s,a) &\geq \mathcal{T}\underline{\widetilde{Q}}^{(i)}, \\
\overline{\widetilde{Q}}^{(i)}(s,a) &\leq \mathcal{T}\overline{\widetilde{Q}}^{(i)}.
\end{aligned}
\tag{4.10}
$$

Now we will see that all swarm operators $\mathcal{T}_t^{(i)}$ are contractions and therefore $\widetilde{Q}^{(i)} = \underline{\widetilde{Q}}^{(i)} = \overline{\widetilde{Q}}^{(i)}$ [K+12]. We show contraction for infinity norm that is

$$
\|\mathcal{T}_t^{(i)} Q - \mathcal{T}_t^{(i)} Q'\|_\infty \leq \alpha \|Q - Q'\|_\infty, \tag{4.11}
$$

for some $\alpha \in [0, 1]$ and any $Q, Q' \in \mathcal{Q}$.

$$
\begin{aligned}
\|\mathcal{T}_t^{(i)} Q - \mathcal{T}_t^{(i)} Q'\|_\infty &= \max_{s \in S, a \in A} |\mathcal{T}_t^{(i)} Q(s,a) - \mathcal{T}_t^{(i)} Q'(s,a)| \\
&= \max_{s \in S, a \in A} |\gamma E_T(Q(s', \pi_t^{(i)}(s')) - Q(s', \pi_t^{(i)}(s')))| \\
&= \max_{s \in S, a \in A} \gamma |\sum_{s' \in S} T(s'|s,a)(Q(s', \pi_t^{(i)}(s')) - Q(s', \pi_t^{(i)}(s')))| \\
&\leq \max_{s \in S, a \in A} \gamma |\sum_{s' \in S} T(s'|s,a) \max_{s' \in S} (Q(s', \pi_t^{(i)}(s')) - Q(s', \pi_t^{(i)}(s')))| \\
&= \max_{s \in S, a \in A} \gamma |\max_{s' \in S} (Q(s', \pi^{(i)})_t(s') - Q(s', \pi_t^{(i)}(s'))) \sum_{s' \in S} T(s'|s,a)| \\
&= \max_{s' \in S} \gamma |(Q(s', \pi_t^{(i)}(s')) - Q(s', \pi_t^{(i)}(s'))) \times 1| \\
&\leq \max_{s' \in S, a' \in A} \gamma |(Q(s', a') - Q(s', a'))| \\
&= \gamma \|Q - Q'\|_\infty
\end{aligned}
\tag{4.12}
$$

Therefore from (4.10) and $\underline{\widetilde{Q}}^{(i)} = \overline{\widetilde{Q}}^{(i)}$ (convergence due to contraction property), we get

$$
\widetilde{Q}^{(i)}(s,a) = \mathcal{T} \widetilde{Q}^{(i)}. \tag{4.13}
$$

Since the fixed point value for Bellman operator is unique and equal to Q^*, the above results to

$$
\widetilde{Q}^{(i)} = Q^*. \tag{4.14}
$$

□

Chapter 5

Regret Bounds and Optimal Swarm RL

Among all Swarm RL algorithms, we are interested in the regret-optimal Swarm RL algorithm. By trying to upper-bound (2.7) for Swarm RL, we hope to derive an update policy $\pi_t^{(i)}$ and clustering $\mathcal{I}_t^{(i)}$ of agents. From the definition of ensemble-regret, based on the action recommended by $\pi_t^{(i)}$ for a state s, agglomerations of agents based on unique actions can be created. We add on to the notations defined in 2.1 by adding

$$\mathcal{J}_t^{(a)}(s) = \{i | \pi_t^{(i)}(s) = a\}, \forall a \in A, \tag{5.1}$$

which represents these agglomerations.

Theorem 5.0.1. *For a swarm RL algorithm, ensemble regret $Regret_{\pi_t^{(\cdot)}}(s)$ at step t for state s is upper-bounded as*

$$E[Regret_{\pi_t^{(\cdot)}}(s)] \leq \sum_{\mathcal{J} \in \mathcal{J}_t^{(\cdot)}(s)} \sum_{a \in A} \frac{|\mathcal{J}| \Delta_{s,a}}{e^{\frac{1}{2}|\mathcal{J}|\Delta_{s,a}^2}}. \tag{5.2}$$

Proof. Similar to PSRL[OVRR13], an important observation to note is that for Swarm RL, Q^* and $Q_t^{(i)}$ are identically distributed. Therefore, from the posterior sampling lemma 5.0.2 that states

Lemma 5.0.2 (posterior sampling)**.** *If f is the distribution of Q^*, then for any $\sigma(H_t)$ measurable function g,*

$$E[g(Q^*)|H_t] = E[g(Q_t^{(i)})|H_t]. \tag{5.3}$$

On taking expectation over H_t shows $Eg([Q^*)] = E[g(Q_t^{(i)})]$ through the tower property. Using this property for a given state s and action a on R.H.S of 4.2 gives

$$E[Q_t^{(j)}(s, g_{t-1}^{(j)}(s))] = E[\max_{b \in A} Q^*(s, b)], \tag{5.4}$$

and on L.H.S gives

$$E[Q_t^{(j)}(s, a)] = E[Q^*(s, a)]. \tag{5.5}$$

From the definition of ensemble regret,

$$\begin{aligned}
Regret_{\pi^{(\cdot)}}(s) &= \sum_{i \in \mathcal{K}} \sum_{a \in A} \mathbb{1}\{\pi^{(i)}(s) = a\} \Delta_{s,a} \\
&= \sum_{\mathcal{J} \in \mathcal{J}_t^{(\cdot)}(s)} \sum_{a \in A} \mathbb{1}\{\pi_t^{(\mathcal{J})}(s) = a\} |\mathcal{J}| \Delta_{s,a} \qquad \text{(As per (5.1))} \\
&\leq \sum_{\mathcal{J} \in \mathcal{J}_t^{(\cdot)}(s)} \sum_{a \in A} \mathbb{1}\{\sum_{j \in \mathcal{J}} (Q_t^{(j)}(s, a) - Q_t^{(j)}(s, g_{t-1}^{(j)}(s))) \geq 0\} |\mathcal{J}| \Delta_{s,a} \qquad (5.6) \\
&\quad \text{(From the condition (4.2)) on swarm update strategy)} \\
&= \sum_{\mathcal{J} \in \mathcal{J}_t^{(\cdot)}(s)} \sum_{a \in A} \mathbb{1}\{\sum_{j \in \mathcal{J}} \chi_t^{(j)}(s, a) \geq 0\} |\mathcal{J}| \Delta_{s,a},
\end{aligned}$$

where $\chi_t^{(j)}$ is a random variable defined as

$$\chi_t^{(j)}(s, a) = Q_t^{(j)}(s, a) - Q_t^{(j)}(s, g_{t-1}^{(j)}(s)). \tag{5.7}$$

From (5.4) and (5.5), expectation of $\chi_t^{(j)}(s, a)$ is given by

$$\begin{aligned}
E[\chi_t^{(j)}(s, a)] &= E[Q_t^{(j)}(s, a) - Q_t^{(j)}(s, g_{t-1}^{(j)}(s))] \\
&= E[Q^*(s, a)] - E[\max_{b \in A} Q^*(s, b)] \\
&= E[Q^*(s, a) - V^*(s)] \\
&= -\Delta_{s,a}.
\end{aligned} \tag{5.8}$$

Lemma 5.0.3 (Chernoff-Hoffending Bound)**.** *Let $Z_1, Z_2, ..., Z_n$ be independent random variables on $\mathcal{R}$ such that $a_i \leq Z_i \leq b_i$ with probability one. If $S_n = \sum_{i=1}^{n}$, then for all $t \geq 0$*

$$Pr(S_n - E[S_n] \geq t) \leq e^{-2t^2 / \sum(b_i - a_i)^2} \tag{5.9}$$

and

$$Pr(S_n - E[S_n] \leq -t) \leq e^{-2t^2 / \sum(b_i - a_i)^2}. \tag{5.10}$$

Probability $Pr(\sum_{j \in \mathcal{J}} \chi_t^{(j)}(s,a) \geq 0)$ can now be bound using the above stated Chernoff-Hoeffding Inequality as follows

$$\begin{aligned} Pr(\sum_{j \in \mathcal{J}} \chi_t^{(j)}(s,a) \geq 0) &= Pr(\sum_{j \in \mathcal{J}} \chi_t^{(j)}(s,a) + |\mathcal{J}|\Delta_{s,a} \geq |\mathcal{J}|\Delta_{s,a}) \\ &\text{(On subtracting } E[\sum_{j \in \mathcal{J}} \chi_t^{(j)}(s,a)] \text{ on both sides and using (5.8))} \\ &\leq e^{\frac{-2(|\mathcal{J}|\Delta_{s,a})^2}{|\mathcal{J}| \times 2^2}} \\ &\text{(Using (5.9). Range of any } \Delta_{s,a} \text{ is } [-1, 1]) \\ &= e^{-\frac{1}{2}|\mathcal{J}|\Delta_{s,a}^2} \end{aligned} \tag{5.11}$$

Using the above bound, we can now bound $E[Regret_{\pi^{(\cdot)}(s)}]$ as

$$\begin{aligned} E[Regret_{\pi^{(\cdot)}}(s)] &\leq \sum_{\mathcal{J} \in \mathcal{J}_t^{(\cdot)}(s)} \sum_{a \in A} E[\mathbb{1}\{\sum_{j \in \mathcal{J}} \chi_t^{(j)}(s,a) \geq 0\}]|\mathcal{J}|\Delta_{s,a} \\ &\text{(Taking } E \text{ inside summation)} \\ &= \sum_{\mathcal{J} \in \mathcal{J}_t^{(\cdot)}(s)} \sum_{a \in A} Pr(\sum_{j \in \mathcal{J}} \chi_t^{(j)}(s,a) \geq 0)|\mathcal{J}|\Delta_{s,a} \\ &\leq \sum_{\mathcal{J} \in \mathcal{J}_t^{(\cdot)}(s)} \sum_{a \in A} e^{-\frac{1}{2}|\mathcal{J}|\Delta_{s,a}^2}|\mathcal{J}|\Delta_{s,a} \\ &\text{(From (5.11))} \\ &= \sum_{\mathcal{J} \in \mathcal{J}_t^{(\cdot)}(s)} \sum_{a \in A} \frac{|\mathcal{J}|\Delta_{s,a}}{e^{\frac{1}{2}|\mathcal{J}|\Delta_{s,a}^2}} \end{aligned} \tag{5.12}$$

□

From the regret bound, it is clear that the regret-optimal Swarm RL algorithm's update strategy must be such that minimizes the number of agglomerations $|\mathcal{J}^{(\cdot)_t(s)}|$, in-turn maximizing individual agglomeration sizes $|\mathcal{J}_t^{(i)}(s)|$. In other words, our $\pi_t^{(.)}(s)$ must be such that number of unique actions are minimized. The restriction imposed by (4.2) coupled with the above observation induces the requirements

$$\pi_t^{(j)}(s) = \pi_t^{(i)}(s), \forall\, j \in \mathcal{I}_t^{(i)}(s), \tag{5.13}$$

$$\mathcal{I}_t^{(i)}(s) = \{j | \pi_t^{(i)}(s) \in \mathcal{A}_t^{(i)}\}, \tag{5.14}$$

for all agent $i \in \mathcal{K}$ and state $s \in S$. As mentioned in section 2.2, it can be seen that the above requirements can be satisfied by applying optimal set cover to a collection $S = \bigcup_{i \in \mathcal{K}} \mathbb{I}_t^{(i)}(s)$ to obtain a cover $C^* \subset S$ on a universe $\mathcal{U} = \mathcal{K}$. The action corresponding to each set in C^* is the action decision for all agents in that set.

Chapter 6

Optimal Swarm DQN algorithm

Double DQN utilizes a target and a current policy network for removing positive bias. We use the target network to do away with need of maintaining previous iteration's Q-function estimate. Adaptation of the Optimal Swarm RL as a DQN is summarized by the following pseudo code. For a pictorial representation of the Agglomerate function, refer to 6.

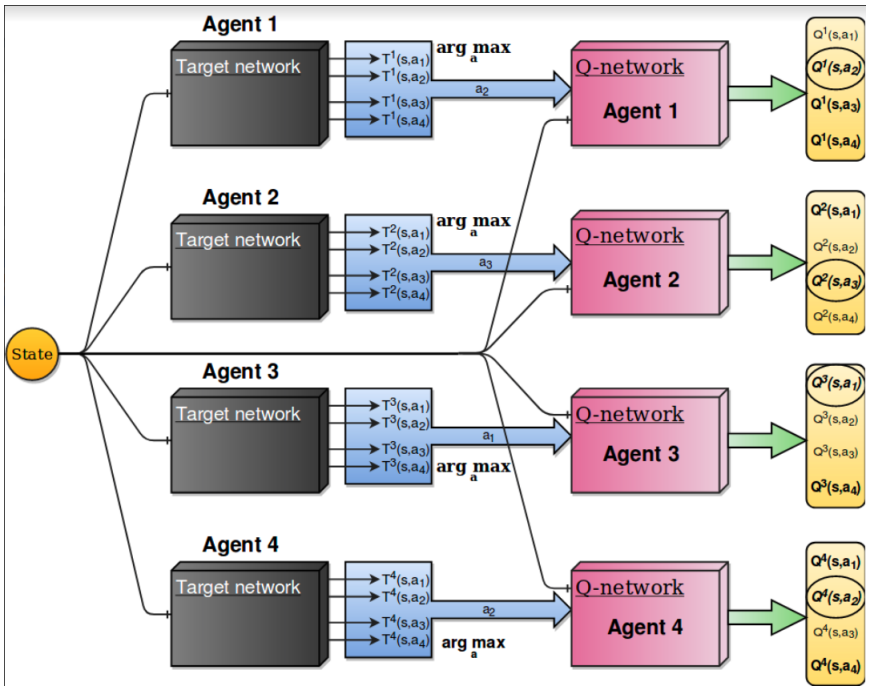

Fig. 6.1 Pictorial flow of Agglomerate function computation.

The $Agglomerate(s, T^{(\cdot)}, Q^{(\cdot)})$ procedure is used both while making action decision for an

Algorithm 1 Optimal Swarm DQN algorithm.

procedure AGGLOMERATE(state s, Target Network $T^{(\cdot)}$, Current Network $Q^{(\cdot)}$)
  Find Target best actions $g^{(i)} = \text{argmax}_{a \in A}\, T^{(i)}(s_t, a)$
  Construct preferred action sets $\mathcal{A}^{(j)} = \{a | Q^{(j)}_{s_t}(s_t, a) \geq Q^{(j)}_{s_t}(s_t, g^{(i)})\}$
  Revert preferred actions sets to get $\mathcal{J}^{(a)} = \{j | a \in \mathcal{A}^{(j)}\}$
  Get optimal set cover of $\mathcal{K}$ from $\mathcal{J}^{(\cdot)}$. Delete sets in $\mathcal{J}^{(\cdot)}$ that were not in cover
  Return $\mathcal{J}^{(\cdot)}$
Initialize $\mathcal{K}$ current $Q^{(i)}_0$ and target $T^{(i)}$ heads $T^{(i)} = Q^{(i)}_0$ with random weights.
Initialize step $t \rightarrow 0$
for Each episode **do**
  Pick an agent at random using $i \sim Uniform(\mathcal{K})$
  for Each step observation s_t of episode **do**
    $\mathcal{J}^{(\cdot)} \leftarrow$ Agglomerate(s, $T^{(\cdot)}$,$Q^{(i)}_t$)
    Take action $a_t = a$, s.t. $i \in \mathcal{J}^{(a)}$, and observe r_t and s_{t+1}.
    Store experience (s_t, a_t, r_t, s_{t+1}) in replay memory D.
    for $(s_{t'}, a_{t'}, r_{t'}, s_{t'+1})$ in *SampleBatch(D)* **do**
      $\mathcal{J}^{(\cdot)} \leftarrow$ Agglomerate(s, $T^{(\cdot)}$,$Q^{(i)}_t$)
      Compute $A^{(j)} = a$, s.t. $j \in \mathcal{J}^{(a)}, \forall\, j \in \mathcal{K}$
      Train using error = $Y^{(\cdot)} \rightarrow r_{t'} + \gamma \times T^{(\cdot)}(s_{t'+1}, A^{(\cdot)}) - Q^{(\cdot)}(s_{t'}, a_{t'})$
    if t % *UpdateFreq* = 0 **then** $T^{(\cdot)} \leftarrow Q^{(\cdot)}_t$
    $t \leftarrow t + 1$.

observation, as well as error calculation while training on a sampled batch. This procedure uses exponential time Optimal Set Cover algorithm for generating a cover of heads, which isn't computationally restrictive as long as the number of heads in the ensemble are small. As increasing number of heads directly increase neural network architecture complexity, usually the ensemble head count is kept small. For large number of heads, $log(N)$ approximation greedy algorithm for set cover can be used, without affecting performance significantly.

Chapter 7

Experiments and Results

In this section we list the simulation result compression for 49 ALE environments. Table 7.3 lists the scores of Double DQN[VHGS16], Ensemble Voting DQN[WvH08], Bootstrap DQN[OBPVR16] and Optimal Swarm DQN on these environments. Table 7.1 list the parameters used for Optimal Swarm DQN, and table 7.3 compares the performance in different ALE environments. The exact Neural Network Architecture is shown in figure 7. Source code is available at `github.com/Rishav1/Atari`.

Parameter Name	Value
Number of Frames	20,000,000
Experience Replay Size	200,000
Training Begin Step	50,000
Target Update Frequency	10,000
Sample Batch Size	32
Frames per observation	4
Memory Sample Frequency	4
Optimizer	RMSprop
RMS Epsilon	0.1
RMS Momentum	0.95
RMS Learning Rate	0.0000625
Ensemble heads	5

Table 7.1 Parameter Table for Optimal Swarm DQN implementation on ALE environments.

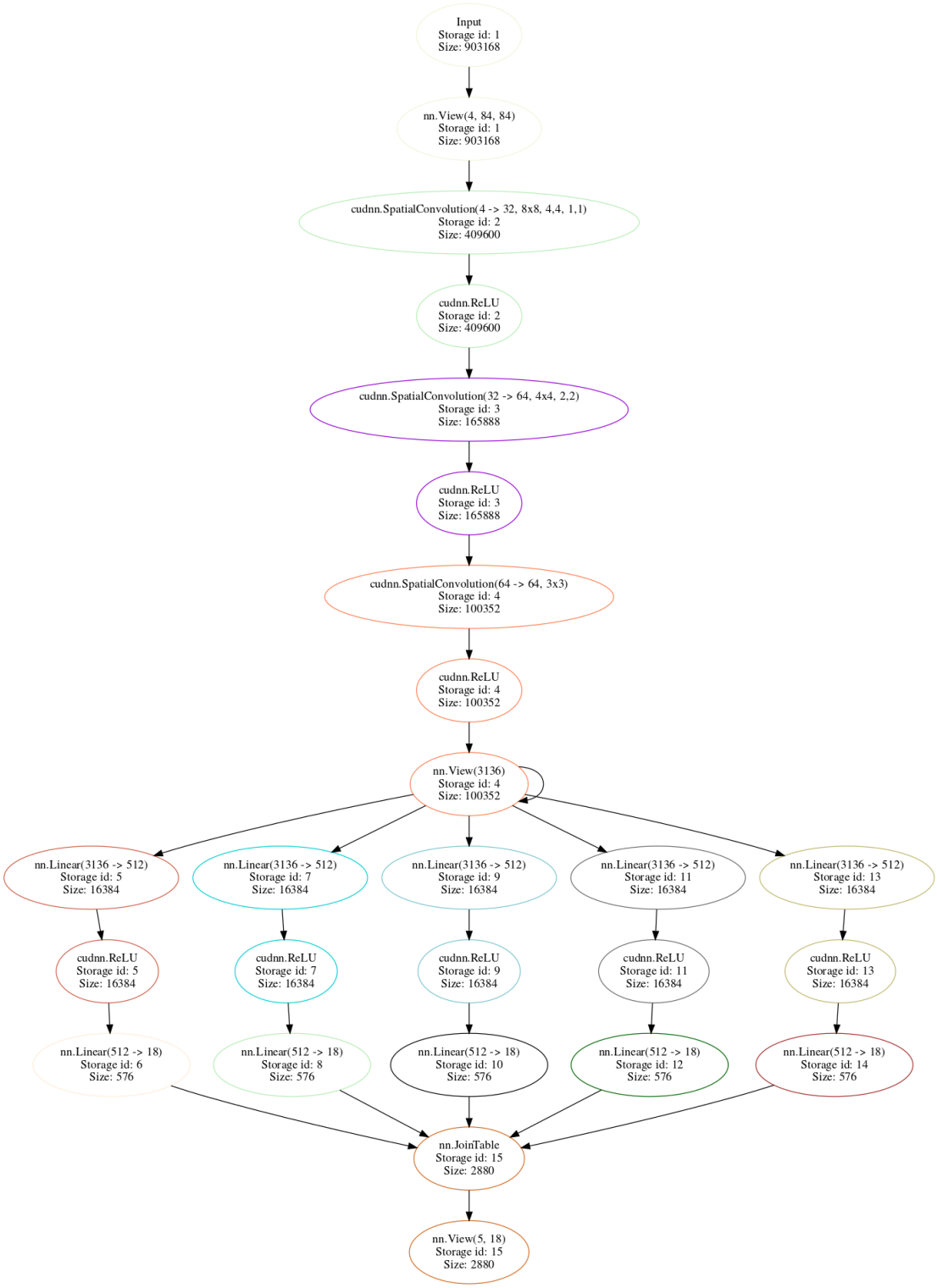

Fig. 7.1 Neural Network Architecture with $|\mathcal{K}| = 5$ heads.

	Bootstrapped DQN	Double DQN	Ensemble Voting	Optimal RL DQN
Alien	1445.1	2059.7	2282.8	720.0
Amidar	430.58	667.5	683.72	128.0
Assault	2519.06	2820.61	3213.58	399.2
Asterix	3829.0	7639.5	8740.0	1050.1
Asteroids	1009.5	1002.3	1149.3	1410.6
Atlantis	1314058.0	1982677.0	1786305.0	18100.0
Bank Heist	795.1	789.9	869.4	52.1
Battle Zone	26230.0	24880.0	27430.0	14380.0
Beam Rider	8006.58	7743.74	7991.9	801.4
Bowling	28.62	30.92	32.92	70.4
Boxing	85.91	94.07	94.47	18.2
Breakout	400.22	467.45	426.78	3.72
Centipede	5328.77	5177.51	6153.28	6749.23
Chopper Command	2153.0	3260.0	3544.0	1483.0
Crazy Climber	110926.0	124456.0	126677.0	49500.0
Demon Attack	9811.45	23562.55	30004.4	925.2
Double Dunk	-10.82	-14.58	-11.94	-12.22
Enduro	1314.31	1439.59	1999.88	697.31
Fishing Derby	21.89	23.69	30.02	-61.96
Freeway	33.57	32.93	33.92	27.53

Table 7.3 Comparison of maximal mean rewards achieved by agents. Maximal mean reward is calculated in a window of 100 consecutive episodes. Bold denotes the highest value in each row. (Table lists incomplete training results. Optima RL DQN was trained only for 200,000/20,000,000 steps)

Chapter 8

Conclusion

We introduced a class of ensemble RL algorithms, namely Swarm RL, and analyzed it's convergence and optimality properties. We formulated a regret bound for an arbitrary Swarm RL algorithm and used it to come up with regret-optimal Swarm RL algorithm. The adaptation of Optimal Swarm RL as a DQN was presented and an empirical analysis of it's performance in comparison to Double DQN, Ensemble Voting DQN and Bootstrapped DQN was shown to establish it's efficacy.

References

[ABS16] O. Anschel, N. Baram, and N. Shimkin. Averaged-DQN: Variance Reduction and Stabilization for Deep Reinforcement Learning. *ArXiv e-prints*, November 2016.

[AJ17] S. Agrawal and R. Jia. Posterior sampling for reinforcement learning: worst-case regret bounds. *ArXiv e-prints*, May 2017.

[BOG+16] M.G. Bellemare, G. Ostrovski, A. Guez, P. Thomas, and R. Munos. Increasing the action gap: New operators for reinforcement learning. pages 1476–1483, 2016. cited By 8.

[DFR98] Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian q-learning. pages 761–768, 1998. cited By 157.

[K+12] Takashi Kamihigashi et al. Existence and uniqueness of a fixed point for the bellman operator in deterministic dynamic programming. Technical report, 2012.

[McF] Roger McFarlane. A survey of exploration strategies in reinforcement learning.

[MR17] Rakesh R. Menon and Balaraman Ravindran. Shared learning : Enhancing reinforcement in q-ensembles. *CoRR*, abs/1709.04909, 2017.

[OBPVR16] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4026–4034. Curran Associates, Inc., 2016.

[OVRR13] I. Osband, B. Van Roy, and D. Russo. (more) efficient reinforcement learning via posterior sampling. 2013. cited By 24.

[SOR+10] Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.

[VHGS16] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. pages 2094–2100, 2016. cited By 38.

[WvH08] M. A. Wiering and H. van Hasselt. Ensemble algorithms in reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(4):930–936, Aug 2008.