Combining Recent Advances in Reinforcement Learning in Super Mario Bros. : Appendix

Student Name: Max Woolterton

Supervisor Name: Stefan Dantchev

Submitted as part of the degree of Msci Natural Sciences to the

Board of Examiners in the Department of Computer Sciences, Durham University

VII APPENDIX

A Q-Learning

Q-Learning (Watkins and Dayan, 1992), a form of temporal difference learning (Sutton, 1988), is a model-free RL algorithm used to estimate and learn state-action values, equivalently the value of an action in a state, and in doing so learn an optimal policy π^* in any finite MDP. The value of an action a in a state s following a policy π is defined as $Q_\pi(s,a) = \mathbb{E}[G_t \,|\, s_t = s, a_t = a, \pi]$, the expectation of G_t over the distribution of admissible trajectories from time-step $t+1$ onwards following policy π (NB that, throughout this text, γ is always set to 0 for a terminal state). The optimal value of an action a in a state s is then $Q^*(s,a) = \max_\pi Q_\pi(s,a)$ and the optimal policy $\pi^* = \operatorname{argmax}_a Q^*(s,a)$. It is of note that Q^* obeys an identity known as the Bellman equation by the time homogeneity property of the Markov chains that MDPs are built upon:

$$Q^*(s,a) = \mathbb{E}_{s' \sim T(\cdot|s,a)}[r + \gamma \max_{a' \in \mathcal{A}} Q^*(s',a') \,|\, s,a]$$

By constructing estimates Q_i of Q^*, the use of this identity allows various value-iteration based RL algorithms, such as Q-Learning, to converge to the optimal action-value function, $Q_i \to Q^*$ as $i \to \infty$ (Sutton, 1988). In traditional Q-Learning, an estimate for the values of all state-actions are stored in a large Q-table, Q_i, which is updated via using the Bellman equation as an iterative update on individual transitions of experience (s,a,r,s'), with learning rate $\alpha \in [0,1]$:

$$Q_{i+1}(s,a) \leftarrow Q_i(s,a) + \alpha\big(r + \gamma \max_{a' \in \mathcal{A}} Q_i(s',a') - Q_i(s,a)\big) \tag{1}$$

Q-Learning is also off-policy, learning about the greedy policy $a = \operatorname{argmax}_{a \in \mathcal{A}} Q(s,a)$, while following a policy that enforces adequate exploration of the state space, often an ϵ-greedy policy. The feasibility and convergence of traditional tabular Q-Learning is however highly dependent upon the size of the state-action space, as there is no generalization over states nor entries and all values must be updated individually, thus this method is often impractical for many applications.

B Deep Q-Learning

In Deep Q-Learning, a parameterized value function approximator $Q(s,a;\boldsymbol{\theta}) : \mathcal{S} \to \mathbb{R}$ with parameters $\boldsymbol{\theta}$ is instead used to estimate Q^*, specifically a multi-layered neural network, replacing the Q-tables used above. This approximator is called a Deep Q-Network (Mnih et al., 2013) (DQN). This network is trained and its parameters updated towards target $Y_i^{(tag)}$ (3) by using stochastic gradient descent to minimize the loss (2) (this update mimicking the function of (1)):

$$\big(Y_i - Q(s_t, a_t; \boldsymbol{\theta}_i)\big)^2 \qquad (2) \qquad\qquad Y_i^Q = r_t + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a'; \boldsymbol{\theta}_i) \qquad (3)$$

where, in an attempt to ensure samples learned from remain i.i.d, by reducing the correlation between subsequent samples, and thus reducing the variance of updates (while also reducing unwanted feedback loops), the transition (s_t, a_t, r_t, s_{t+1}) is sampled from an Experience Replay buffer containing the last N $(= 10^6)$ transitions.

However, Mnih et al. (2013) also introduced the idea of a "target network" $Q(s, a; \boldsymbol{\theta}^-)$, in addition to the "online network" $Q(s, a; \boldsymbol{\theta})$ referred to in the above paragraph. The target network is identical to the online network, except for its weights $\boldsymbol{\theta}^-$, which are copied over from the online network every τ network updates i, and kept constant in between, while gradients are only ever back-propagated into the online network. This is done to increase training stability by providing a stable "target" for the network to learn against (avoiding the "catastrophic forgetting" effect), and dramatically improves the performance of the DQN algorithm (Mnih et al., 2013). The resulting targets from (3) used in the loss (2) for the DQN algorithm become:

$$Y_i^{DQN} = r_t + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a'; \boldsymbol{\theta}_i^-) \qquad (4)$$

C Double Q-Learning

Double Q-Learning (van Hasselt, 2010) increases the stability of Q-Learning via reducing value over-estimation (see §III-B) by decoupling the selection and evaluation of state-action values within the max operator in (1, 3) into two separate networks Q and Q', which are then trained alternately by swapping their roles each iteration. This transforms the operator into: $Q'(s_{t+1}, \operatorname{argmax}_{a' \in \mathcal{A}} Q(s_{t+1}, a'))$. van Hasselt et al. (2015) then used the ideas of Double Q-Learning to create a new variant of the DQN algorithm, namely DDQN, where the pre-existing target network replaces the role of Q', however the networks are no longer swapped over during training as they are in Double Q-Learning. The targets from (4) are hence transformed from their equivalent form $Y_i^{DQN} = r_t + \gamma Q(s_{t+1}, \operatorname{argmax}_{a' \in \mathcal{A}} Q(s_{t+1}, a'; \boldsymbol{\theta}_i^-); \boldsymbol{\theta}_i^-)$ into:

$$Y_i^{DDQN} = r_t + \gamma Q(s_{t+1}, \operatorname*{argmax}_{a' \in \mathcal{A}} Q(s_{t+1}, a'; \boldsymbol{\theta}_i); \boldsymbol{\theta}_i^-) \qquad (5)$$

D Multi Step Returns

Sutton and Barto (1998) showed that using forward-view multi-step targets (Sutton, 1988) leads to faster convergence and learning. Using n-step returns $\sum_{k=0}^{n-1} \gamma^k r_{t+k}$, the single-step targets in (5) become:

$$Y_i^{nStepDDQN} = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n Q(s_{t+n}, \operatorname*{argmax}_{a' \in \mathcal{A}} Q(s_{t+n}, a'; \boldsymbol{\theta}_i); \boldsymbol{\theta}_i^-) \qquad (6)$$

E Value Function Rescaling

Traditionally in DQN, for the sake of stability, individual rewards are clipped to a narrow range such as $[-1, 1]$, however Pohlen et al. (2018) showed that performance could be improved by

instead applying an invertible value function rescaling $h(x) = \text{sign}(x)\left(\sqrt{|x|+1}-1\right) + \epsilon x$ with its equivalent inverse $h^{-1}(x)$, transforming the targets from (6) into:

$$Y_i^{h(nStepDDQN)} = h\left(\sum_{k=0}^{n-1}\gamma^k r_{t+k} + \gamma^n h^{-1}\left(Q(s_{t+n}, \underset{a'\in\mathcal{A}}{\text{argmax}}\, Q(s_{t+n}, a'; \boldsymbol{\theta}_i); \boldsymbol{\theta}_i^-)\right)\right), \quad (7)$$

$$\text{with:} \quad h^{-1}(x) = \text{sign}(x)\left(\left(\frac{\sqrt{1+4\epsilon(|x|+1+\epsilon)}-1}{2\epsilon}\right)^2 - 1\right) \quad (8)$$

References

Mnih, V. et al. (2013), 'Playing atari with deep reinforcement learning', *arXiv preprint arXiv:1312.5602* .
Pohlen, T. et al. (2018), 'Observe and look further: Achieving consistent performance on atari'.
Sutton, R. S. (1988), 'Learning to predict by the methods of temporal differences', *Machine learning* **3**(1), 9–44.
Sutton, R. S. and Barto, A. G. (1998), *Reinforcement learning: An introduction*, MIT press.
van Hasselt, H. (2010), 'Double q-learning', *Advances in neural information processing systems* **23**, 2613–2621.
van Hasselt, H., Guez, A. and Silver, D. (2015), 'Deep reinforcement learning with double q-learning'.
Watkins, C. J. and Dayan, P. (1992), 'Q-learning', *Machine learning* **8**(3-4), 279–292.