Sensitivity Analysis on DQN Variants E0270 - Project Presentation

Renga Bashyam K G   Arun Govind M

Dept. of CDS, IISc

27/04/2019

RL Basics

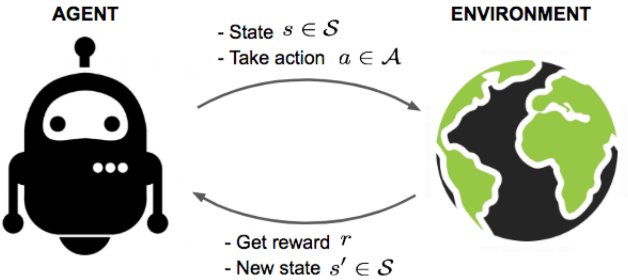

Figure 1: Reinforcement Learning Overview[1]

[1] Weng Lilian. https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html.

Markov Decision Process

(S, A, R, P, γ) where

- S is the set of states of an agent can be in
- A is set of actions an agent can take
- $R(s, a)$ is the reward an agent gets for its action
- $P(S_{t+1} = s' | S_t = s)$ is the transition probabilities
- γ is the discounting factor

Policy : $\pi(a|s)$ - distribution over all possible actions from s

Cumulative Reward at time t :

$G_t = (R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...)$

Value function : $V(s) = \mathbb{E}_\pi[G_t | S_t = s]$

Action-value function : $Q(s, a) = \mathbb{E}_\pi[G_t | S(t) = s, A(t) = a]$

For a given π : $V(s) = \Sigma_{a \in A} Q(s, a)\pi(a|s)$.

Weak ordering of π wrt $V(s)$, atleast one deterministic optimal policy π^* exists

Bellman Expectation Equations

$$
\begin{aligned}
V_\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] \\
&= \mathbb{E}_\pi[r_{t+1} + \gamma G_{t+1} | S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r + \gamma V_\pi(s')])
\end{aligned}
$$

$$
\begin{aligned}
Q_\pi(s,a) &= \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \\
&= \mathbb{E}_\pi[r_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \\
&= \sum_{s'} P(s'|s,a)[r + \gamma V_\pi(s')]) \\
&= \sum_{s'} P(s'|s,a)[r + \gamma \sum_{a'} \pi(a'|s') Q(s',a')])
\end{aligned}
$$

Bellman Optimality Equations

$$V_{\pi}^{*}(s) = max_{a} \sum_{s'} P(s'|s,a)[r + \gamma V_{\pi}^{*}(s')])$$

$$Q_{\pi}^{*}(s,a) = \sum_{s'} P(s'|s,a)[r + \gamma max_{a'} Q^{*}(s',a')])$$

Q-learning

- Model-free, Off-policy
- A form of Temporal Difference Learning
  - $V(S_t) = (1-\alpha)V(S_t) + \alpha G_t$
  - $V(S_t) = V(S_t) + \alpha(G_t - V(S_t))$
  - $V(S_t) = V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$
  - $Q(S_t, A_t) = Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$
- Steps
  - At step t, we pick the action by using an ϵ-greedy method, where we choose a random action with probability ϵ and select the action $A_t = max_{a \in A} Q(S_t, a)$ with probability $1 - \epsilon$
  - With action A_t, we observe reward R_{t+1} and get into the next state S_{t+1}
  - $Q(S_t, A_t) = Q(S_t, A_t) + \alpha(R_{t+1} + \gamma max_{a \in A} Q(S_{t+1}, a) - Q(S_t, A_t))$

Deep Q Networks

- As state and action spaces grow, managing tables becomes intractable
- Use deep network for compact representation - $Q(s, a; \theta)$
- $L(\theta) = E_{(s,a,r,s') \sim U(D)}[(Y(s', a'; \theta^-) - Q(s, a; \theta)^2]$
- $Y(s', a, \theta^-) = r + \gamma max_{a'} Q(s', a'; \theta^-)$
- $U(D)$ is the uniform distribution over the replay memory
- θ "frozen" as θ^- every T iterations
- Follow ϵ-greedy method for action selection

DQN Architecture

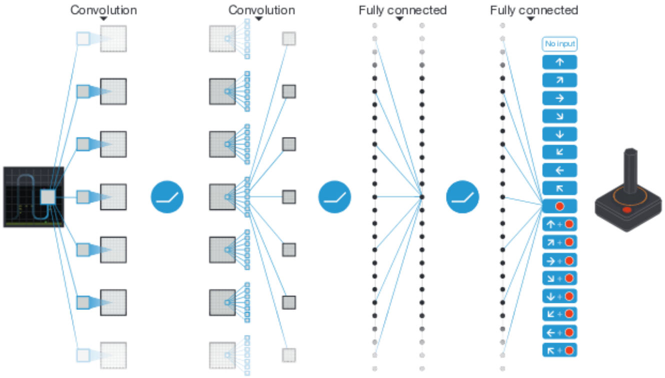

Figure 2: First ever DQN (for Atari games)[2]

[2] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), p. 529.

Double DQNs

- In the vanilla DQN, the target network i.e the network with the frozen parameter θ^{-} is used to both select the next optimal action and evaluate its score $Y(s', a'; \theta^{-})$
- A double DQN [5] decouples the selection and evaluation of the next action a' taken by the agent. Here, the online DQN network. i.e the network with parameter θ is used to select the action a' as $a' = argmax_a Q(s, a)$ and the quality of that action a' given by $Q(s', a')$ is evaluated by the target network.
- This seemingly simple step helps double DQN overcome the problem of overestimation that DQN suffers from.

Double DQNs

- Vanilla DQN
  - $L(\theta) = E_{(s,a,r,s')\sim U(D)}[(Y(s',a';\theta^-) - Q_o(s,a;\theta)^2]$
  - $Y(s',a,\theta^-) = r + \gamma max_{a'} Q_t(s',a';\theta^-)$
- equivalently,
  - $Y(s',a,\theta^-) = r + \gamma Q_t(s', argmax_{a'} Q_t(s',a';\theta^-)$
- Double DQN
  - $L(\theta) = E_{(s,a,r,s')\sim U(D)}[(Y(s',a';\theta^-) - Q(s,a;\theta)^2]$
  - $Y(s',a,\theta^-) = r + \gamma Q_t(s', argmax_{a'} Q_o(s',a';\theta^-)$

Modification to DQN architecture

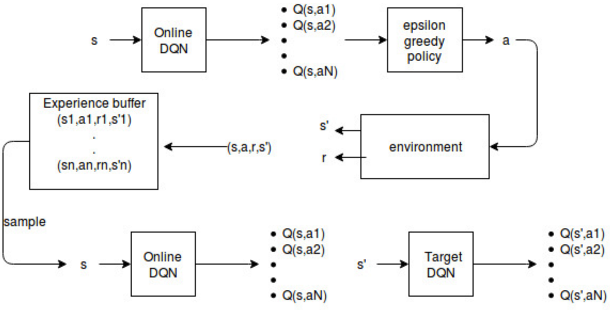

Figure 3: a flow chart explaining DQN

Dueling Networks - Architecture

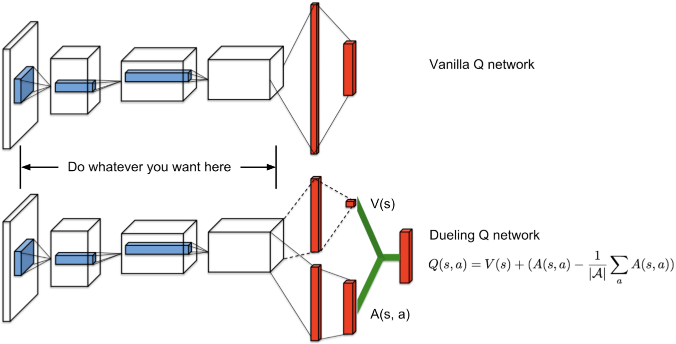

Figure 4: Dueling Network[3]

[3] Ziyu Wang et al. "Dueling network architectures for deep reinforcement learning". In: *arXiv preprint arXiv:1511.06581* (2015).

Dueling Networks

- Advantage function : $A(s,a) = Q(s,a) - V(s)$
- $Q(s,a) = V(s) + (A(s,a) - \frac{1}{|A|}\sum_a A(s,a))$
- Advantages:
  - Better performance and faster convergence in environments with a large action space
  - When actions with similar Q-values for the same state are present, robust to noise

Experimental Setup

- Using an open-source implementation using TensorFlow[4]
- "Cartpole-V1" environment of OpenAI Gym[5]
  - State - represented using four reals
  - Action - binary

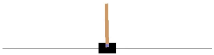

Figure 5: Cartpole-V1

- "p2.xlarge" Amazon EC2 machine (with a Tesla K80 GPU)

[4] Weng Lilian. *Deep Reinforcement Learning Gym's Github repository*. https://github.com/lilianweng/deep-reinforcement-learning-gym.

[5] Greg Brockman et al. *OpenAI Gym*. 2016. eprint: arXiv:1606.01540.

Default HyperParameters

- *Hidden layer dimensions*: $32 * 32$
- *Batch size*: 512
- *Learning rate* α: 0.01
- ϵ *in* ϵ*-greedy (start)*: 1
- ϵ *in* ϵ*-greedy (end)*: 0.02
- *Target update every* T *steps,* T: 10
- *Total number of episodes*: 50

Effect of Learning Rate

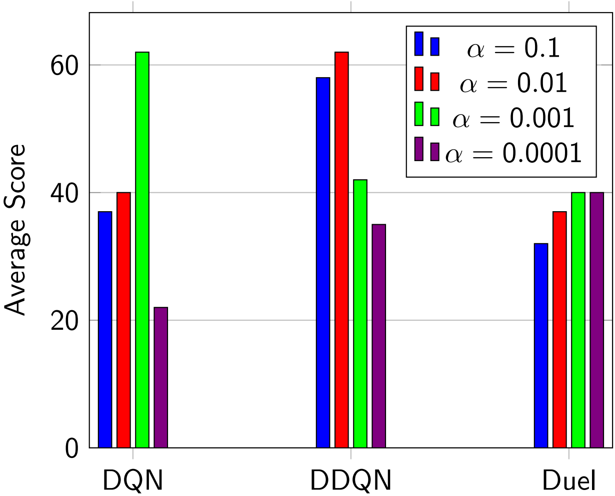

Figure 6: Average Score vs Learning Rates

Effect of Batch Size

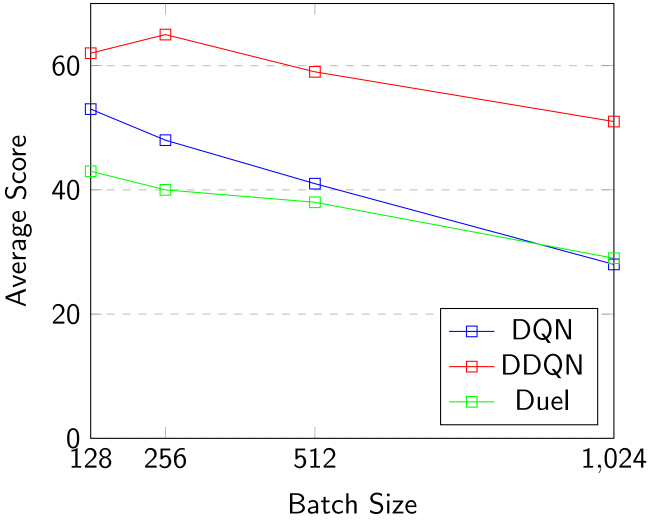

Figure 7: Average Score vs Batch Sizes

Effect of Target Update Steps

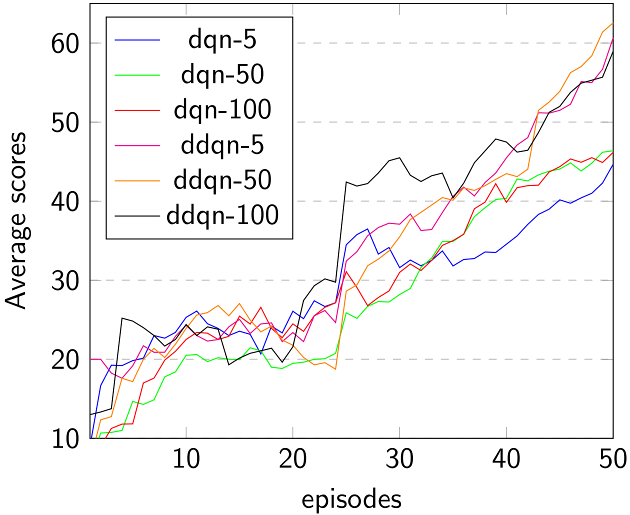

Figure 8: Average score in previous 5 episodes

Effect of Target Update Steps

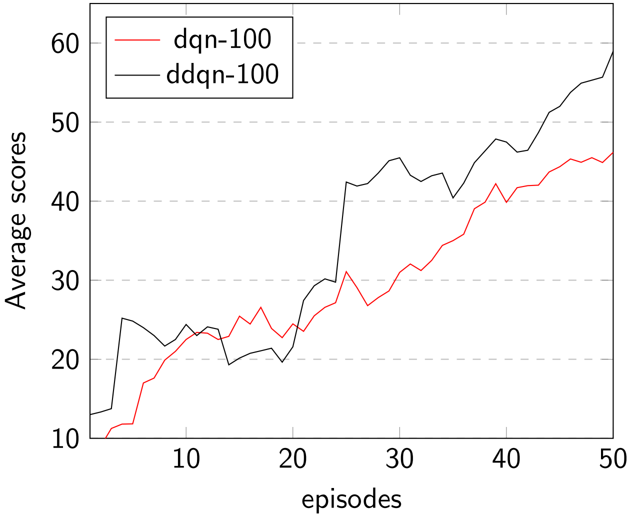

Figure 9: Average score in previous 5 episodes

Effect of Target Update Steps

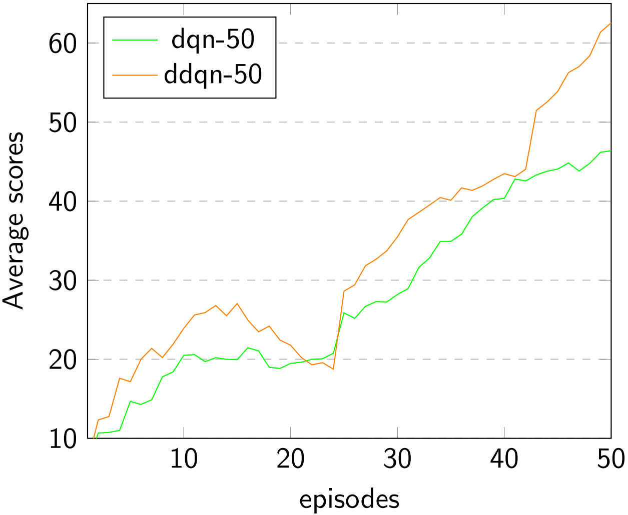

Figure 10: Average score in previous 5 episodes

Effect of Target Update Steps

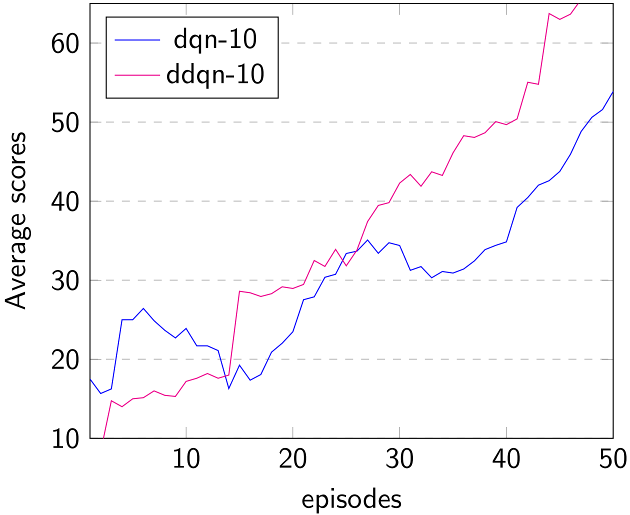

Figure 11: Average score in previous 5 episodes

Effect of Target Update Steps

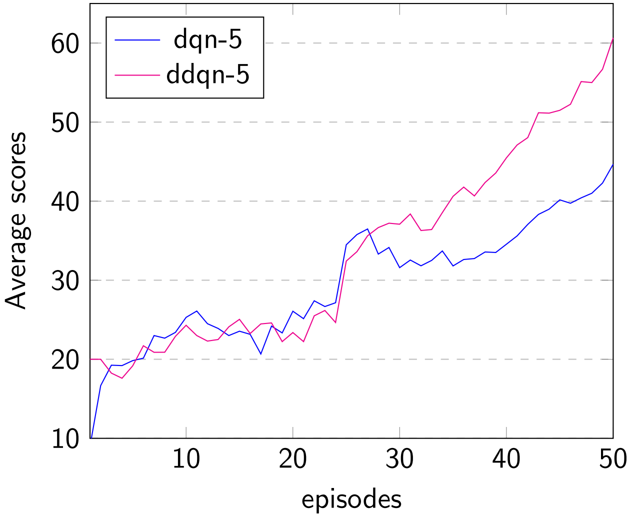

Figure 12: Average score in previous 5 episodes