IE 534/CS 598 Deep Learning

University of Illinois at Urbana-Champaign

Fall 2018

Lecture 12

Topics we will cover this week:

- Actor-critic methods.
- Deep exploration.
- AlphaGo algorithm.

- How to efficiently explore?
- Exploration versus greedy strategy.
- Bayesian approach can be efficient in simple cases such as multi-armed bandits (commonly referred to as “Thompson sampling”).

- Let $Q(x, a; \theta)$ be the approximation for the value function.
- Suppose we have a prior $p(\theta)$.
- We observe a sequence $(A_t, X_t, R_t)_t$.
- Bayesian posterior $p_t(\theta)$ given the data $(A_\tau, X_\tau, R_\tau)_{\tau=1:t}$.

Thompson algorithm:

- Sample θ_t from $p_{t-1}(\theta)$.
- Take action

$$A_t = \arg\max_{a \in \mathcal{A}} Q(X_t, a; \theta_t). \tag{1}$$

- Observe reward R_t.
- Update posterior distribution to produce $p_t(\theta)$.

Given the prior distribution $p_{t-1}(\theta)$ and the observations $H_t = (A_\tau, X_\tau, R_\tau)_{\tau=1:t}$, use Bayes' theorem to compute the posterior distribution

$$p_t(\theta) = \frac{\mathbb{P}[H_t|\theta]p_{t-1}(\theta)}{\int_\Theta \mathbb{P}[H_t|\theta']p_{t-1}(\theta')d\theta'}, \tag{2}$$

where $\theta \in \Theta$.

Given the prior distribution $p_{t-1}(\theta)$ and the observations $H_t = (A_\tau, X_\tau, R_\tau)_{\tau=1:t}$, use Bayes' theorem to compute the posterior distribution

$$p_t(\theta) = \frac{\mathbb{P}[H_t|\theta]p_{t-1}(\theta)}{\int_\Theta \mathbb{P}[H_t|\theta']p_{t-1}(\theta')d\theta'}, \tag{3}$$

where $\theta \in \Theta$.

- Note that θ is high-dimensional, so we have to compute a high-dimensional integral.
- $\mathbb{P}[H_t|\theta]$ may also be challenging to compute.
- Only feasible in very simple settings (e.g., linear models, multi-armed bandits).

- In general, Bayesian estimation for neural networks is computationally infeasible.
- Various approximations have been proposed for the Bayesian posterior update, but don't necessarily perform better than SGD.
- There is also a claim that dropout produces a Bayesian posterior distribution (which is incorrect!). See "Risk versus Uncertainty in Deep Learning: Bayes, Bootstrap and the Dangers of Dropout" by Osband.

Thompson algorithm:

- Very good at exploration.
- Computationally efficient and has optimal mathematical properties in simple settings.
- Computationally impractical for more complex settings (such as RL) and models (such as deep RL).

- Develop a "bootstrap deep Q-learning network" which is similar in spirit to Thompson sampling.
- Ensemble of neural networks $\theta^1, \ldots, \theta^K$.
- Each network has a different random initialization.
- Each model corresponds to a different value function $Q(x, a; \theta^k)$.
- Therefore, a variety of strategies will be explored.

Bootstrap deep Q-learning network:

- Select k at random from $\{1, 2, \ldots, K\}$.
- Take action

$$A_t = \arg\max_{a \in \mathcal{A}} Q(X_t, a; \theta_t^k). \tag{4}$$

- Observe reward R_t and X_{t+1}.
- Sample bootstrap mask m_t (a binary mask of length K)
- Save $(X_t, A_t, R_t, X_{t+1}, m_t)$ in replay buffer.
- Update models $\theta_t^1, \ldots, \theta_t^K$.

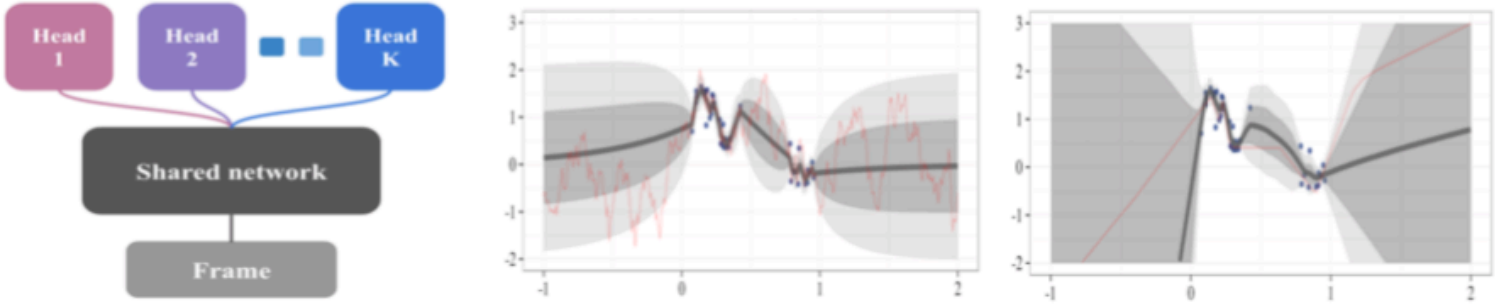
Suppose we sample $(X_\tau, A_\tau, R_\tau, X_{\tau+1}, m_\tau)$ from the replay buffer. The update to the k-th model is

$$\begin{aligned} \theta_t^k &= \theta_t^k + g_\tau^k, \\ g_\tau^k &= m_\tau^k \Big(y_\tau - Q(X_\tau, A_\tau; \theta_t^k) \Big) \nabla_\theta Q(X_\tau, A_\tau; \theta_t^k), \\ y_\tau &= R_\tau + \gamma \max_{a \in \mathcal{A}} Q(X_{\tau+1}, a; \theta_t^k). \end{aligned} \tag{5}$$

Of course, in practice we sample a mini-batch from the replay buffer.

Some important implementation details:

- Replay buffer
- Shared network between the different models $1, 2, \ldots, M$.
- Gradient normalization
- Replace objective function (5) with Double Q-learning.
- See Appendix of NIPS paper.



(a) Shared network architecture (b) Gaussian process posterior (c) Bootstrapped neural nets

Figure 1: Bootstrapped neural nets can produce reasonable posterior estimates for regression.

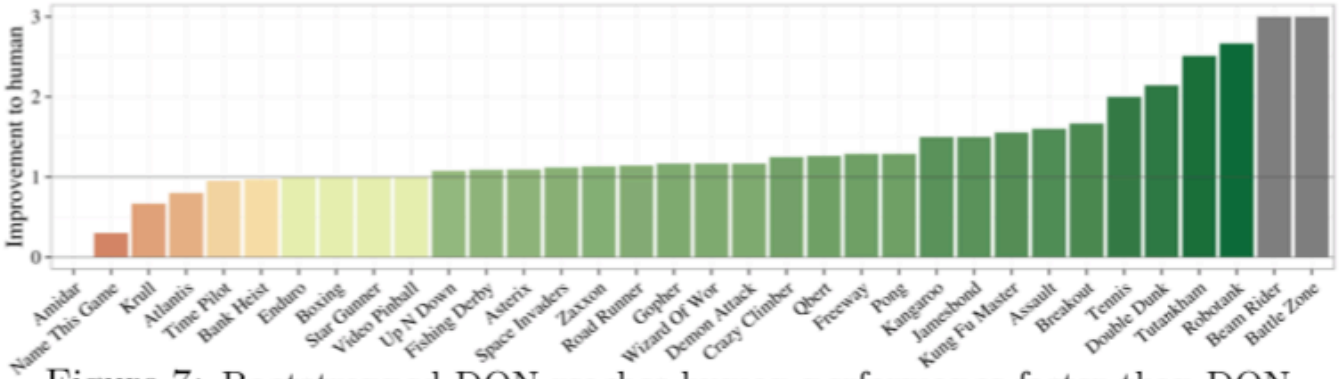


Figure 7: Bootstrapped DQN reaches human performance faster than DQN.

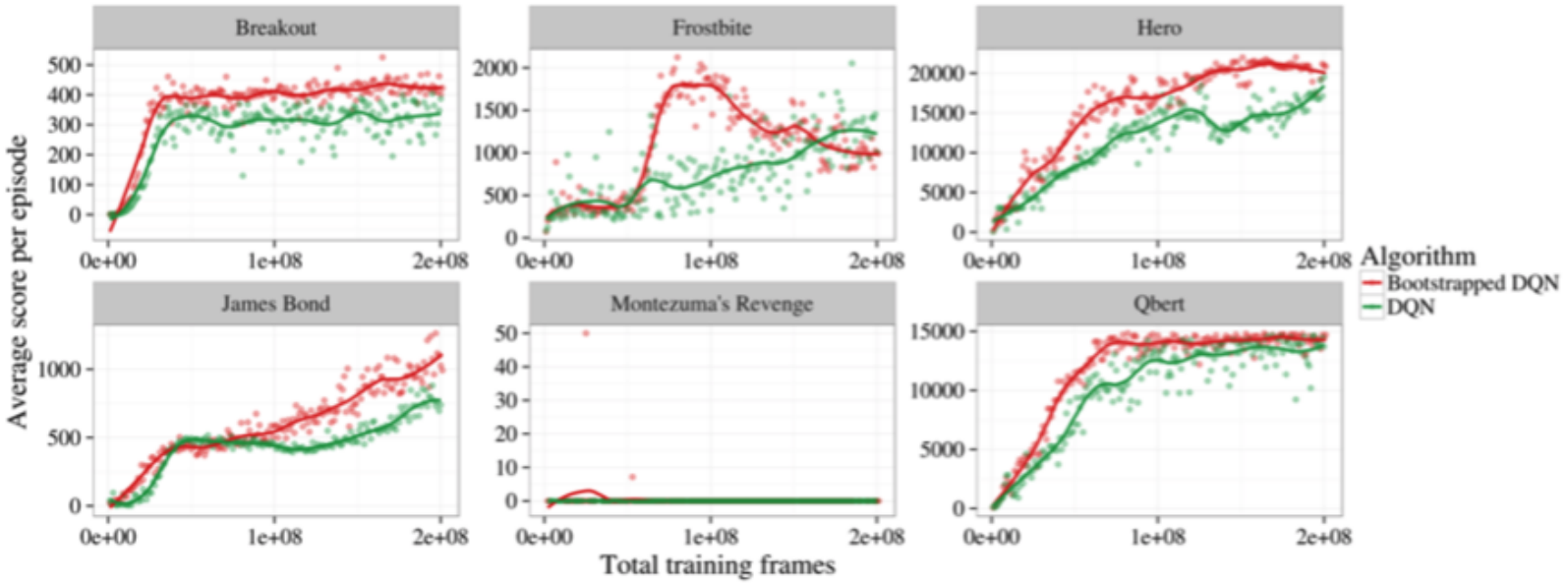


Figure 6: Bootstrapped DQN drives more efficient exploration.

Source: "Deep Exploration via Bootstrapped DQN" by Osband et al. (2015).