Learning to manage a system of bridges subject to seismic hazard using deep Q-networks

Gitanjali Bhattacharjee, gjee@stanford.edu

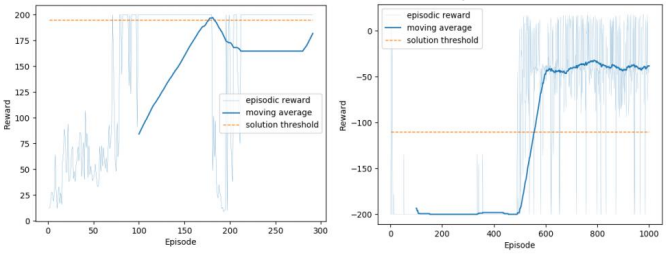
Learning to manage bridges

This project investigates the application of (double) deep Q-learning to an open problem in civil engineering: the (optimal) management of a network of bridges subject to seismic hazard. The reinforcement learning agent's goal is to manage each bridge in network such that some network performance objective – for example, total travel time across the network – is met over a planning horizon.

Deep Q-learning (DQN) is a reinforcement learning technique in which a neural network (NN) approximates the action-value function, Q(*s*, *a*) [1]. Double deep Q-learning (DDQN) partially decouples action selection and evaluation, reducing error in Q-value estimates [2].

Classic verification problems

The DDQN agent solved CartPole-v0 and MountainCar-v0. For CartPole, the agent used an NN with two hidden layers (HLs) of 8 and 4 nodes. For MountainCar, the agent used an NN with three HLs of 256, 128, and 64 nodes. All HLs had ReLU activation functions with He uniform variance scaling initialization, while the output layer used a linear activation function.



The DDQN agent solves classic OpenAI Gym verification problems: CartPole (left) after 195 episodes and MountainCar (right) after 457 episodes.

Deep Q-network inputs and outputs

The NN input is a state vector with *b* elements, each a fragility function parameter *f* of a bridge (from CalTrans).

$$\vec{s} = [f_1, \ldots, f_b]$$

Unless damaged, a bridge has a non-zero *f*. Larger values of *f* indicate greater resistance to ground shaking. The agent can do nothing to, retrofit (double the *f* of), or repair (restore the original *f* of) each bridge. The size of the action space $|A| = 3^b$, so the NN outputs a vector with 3^b elements, each corresponding to a predicted Q(*s*, *a*).

$$\vec{Q}(s,a) = [Q(s,a_1), \ldots, Q(s,a_{3^b})]$$

For this project, $b = 3$ and $|A| = 27$.

One-step episode returns

In a multi-step episode,

$$Q(s,a) = r(s,a) + \gamma \arg\max_{a'} Q(s,a')$$

but for a one-step episode, this reduces to

$$Q(s,a) = r(s,a)$$

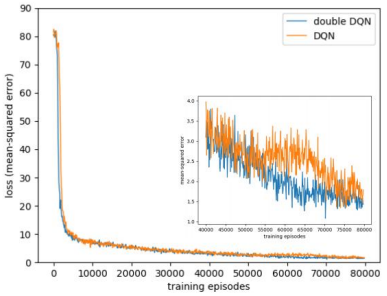
which makes evaluation of an NN's predictive power straightforward. Since there are no standard NN architectures for this problem, multiple NNs were tested on their ability to predict one-step episode returns.

Predicting one-step episode returns

The NNs' predictive power was measured by the loss recorded on a batch of 64 one-step episodes randomly selected from memory and periodically evaluated during training. The following rewards function was used as a proxy for evaluating system performance with an iterative traffic model. All actions were free.

$$r(s,a) = \sum_{i=1}^{b} F_i \frac{1}{b}, \text{ where } F_i = \begin{cases} 1 \text{ if bridge } i \text{ is undamaged} \\ -1 \text{ if bridge } i \text{ is damaged} \end{cases}$$

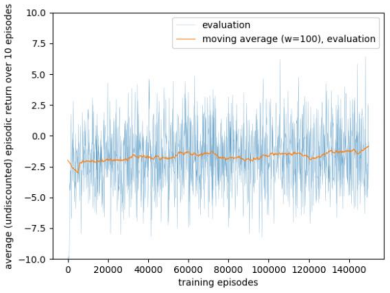
Multiple NNs were tested, with two to five HLs, each with five to 500 hidden units. All HLs had ReLU activation functions with He uniform variance scaling initialization, while the output layer used a linear activation function. All NNs used the Adam optimization algorithm and a mean-squared-error loss metric.



Loss over time of one-step return predictions, with zoom inset.

The best-performing NN had two hidden layers, each with 300 nodes. The DDQN agent outperformed the DQN agent, with a minimum loss of 1.1 after 80000 episodes, compared to 2.7 for the DQN agent.

Longer-term planning results



The DDQN agent's average return over ten evaluation episodes for 150000 episodes.

The undiscounted return for a ten-year episode ranges from -10 to 10. The agent was evaluated every 100 training episodes; its average return over 10 decade-long evaluation episodes (with random starting states) is plotted at left.

Over 150000 training episodes, the DDQN agent achieved a maximum return of 6.4 during evaluation. However, this maximum was not consistently achieved.

Discussion

The DDQN agent learns to repair and retrofit bridges, improving its average episodic return with training. However, the improvement is slight. As repairs and retrofits are free, the agent was expected to achieve the near-optimal return during evaluation, but did not. The agent trained for a maximum of 150000 episodes in an action space of size 27. Mnih et al. trained their agent for 10 million episodes, though it acted in much smaller action spaces (of size 4 to 18) [1]. This suggests that training was too short. Increasing the NN depth from 2 to 5 HLs did not improve results over 50k training episodes; under-fitting may not be the primary concern.

Future work

Immediate future work must include more closely examining the policy and how it changes with training, to better understand what the agent is (and is not) learning. Other opportunities for future work include increasing training time, considering $b > 3$, incorporating action costs, extending episodes to 30 years, and adding prioritized experience replay.

References

1. Mnih, Volodymyr, et al. Playing Atari with deep reinforcement learning. *NIPS Deep Learning Workshop*, 2013.
2. van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double Q-learning. arXiv preprint arXiv:1509.06461, 2015.