Applications of Sequential Decision Making Frameworks to Intrusion Detection Systems: Progress Report

Dave Ackley and Allen Williams

MSU

November 11th 2021

Recall: Sequential Decision Making

What is Sequential Decision Making?

Sequential Decision making involves an agent repeatedly interacting with the environment. Typically these problems are modeled as *Markov Decision Processes(MDPs)*, or some variant such as *Partially Observable Markov Decision Processes(POMDPs)*.

Recall: Sequential Decision Making

What is Sequential Decision Making?

Sequential Decision making involves an agent repeatedly interacting with the environment. Typically these problems are modeled as *Markov Decision Processes(MDPs)*, or some variant such as *Partially Observable Markov Decision Processes(POMDPs)*. An MDP consists of $(\mathcal{S}, \mathcal{A}, R, P, \gamma)$
An agent follows a policy $\pi : \mathcal{S} \rightarrow prob(A)$ and typically an algorithm should find an *optimal policy*, π^*, which maximizes the discounted sum of expected future rewards, that is

$$\mathbb{E}\left[r_i + \gamma^i v_{\pi^*}(s_{i+1})\right] \geq \mathbb{E}\left[r_i + \gamma^i v_{\pi}(s_{i+1})\right]$$

For all policies π.
If $|\mathcal{S}|$ and $|\mathcal{A}|$ are small and the dynamics function P is known, problems can be solved exactly and efficiently with Dynamic Programming.

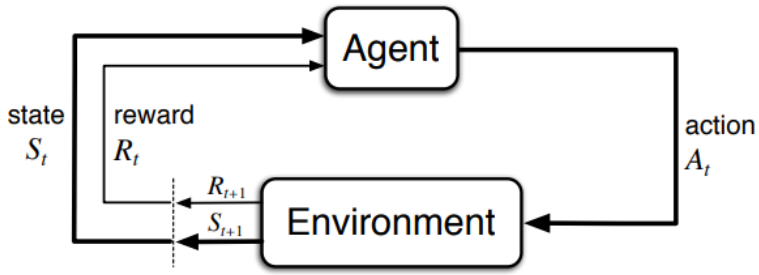

Figure: Image credit [1]

Recall: Reinforcement Learning

What is Reinforcement Learning?

Richard Sutton defines the *reward hypothesis* in [1]

> *That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum or a received scalar signal (called reward).*

Value-Based	*Policy-Based* $\leftrightarrow$ *Actor-Critic*		*Other*
DQN	REINFORCE	A2C	Evolutionary Algorithms
Rainbow	DPG	A3C	
C51	DDPG	SAC	
QR-DQN	TRPO	TD3	
IQN	PPO		

DQN

The beginning of modern deep RL

Reinforcement learning with non-linear function approximation suffers from training instabilities, especially in the case of the *deadly triad*:

- Off-policy training
- Function Approximation
- *Bootstrapping*

Bootstrapping

A large class of reinforcement learning models, including classical Q-learning make use of *temporal difference updates* to estimate values (typically state-action values). These updates make use of an estimated value to update their value function.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma max_a Q_{t+1}(s_{t+1}, a) - Q(s_t, a_t))$$

DQN Continued

DQN: Experience Replay

DQN made use of a few earlier innovations to improve its stability in order to be able to use nonlinear function approximation for the state-action values. First they use an *experience replay buffer* to train the value function networks rather than direct rewards from the environment. This stabilizes training somewhat by reducing the impact of *covariate shift* as the agent's policy changes.

DQN: Target network

The second innovation that allowed DQN to be successful was the use of a separate target network whose weights are updated only occasionally to predict the values in the update rule.

DQN Continued

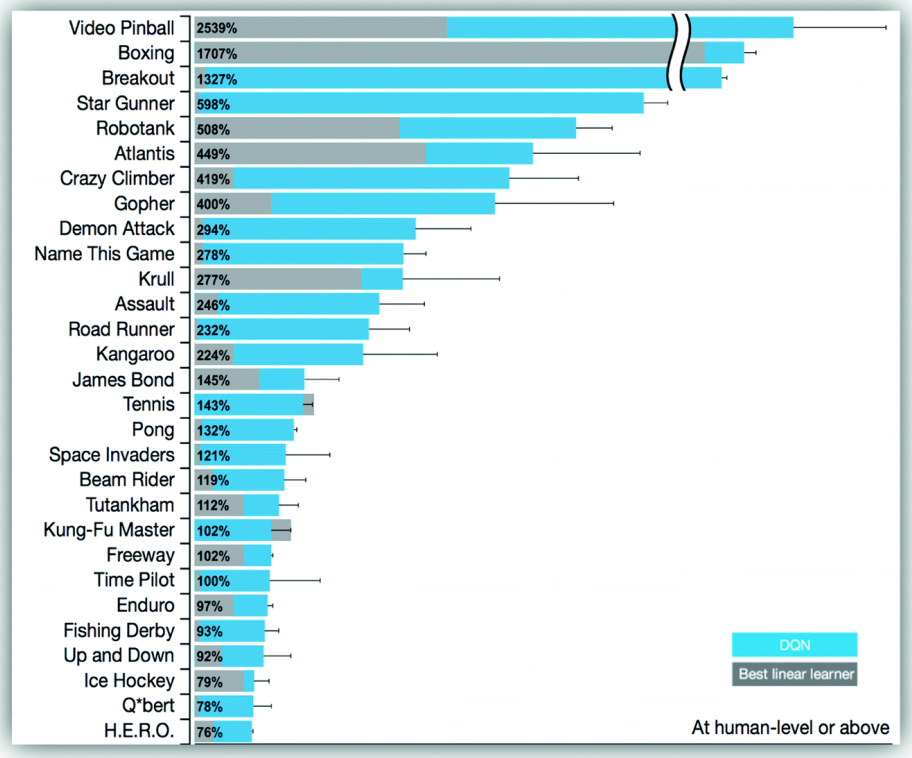

Figure: DQN achieved superhuman performance on several atari games with no fine tuning of hyperparameters.

DQN Hyperparameters

Epsilon Greedy

DQN Estimates Reward-to-go, or *value* for each action at every state it is in. It uses a parameter ε to balance exploration and exploitation. With probability $(1 - \varepsilon)$ the agent chooses the best action, and with probability ε the agent chooses its best action.

Discount Rate and learning rate

Recall the update rule

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma max_a Q_{t+1}(s_{t+1}, a) - Q(s_t, a_t))$$

Here γ is called the discount rate and is important for convergence in infinite horizon environments. In our case the sequential nature of the environment is artificial, we set γ to be very low.

IDS Dataset

KDD CUP 1999

- simulated intrusions in a military network environment
- four main attack categories, DOS, R2L, U2R and probing
- 24 different attack types and 14 more in test set

NSL-KDD

- Aims to improve on KDD99 largely by eliminating redundant rows and offering a standard test-set so algorithms are compared in a fair way
- Includes 21 types of attacks in 5 broad categories.

Environment Setup

Gym

OpenAI Gym provides a general template for RL environments. We setup a custom environment conforming to the OpenAI Gym interface.

We setup an environment for a DQN agent to learn in as if it were a live environment. The environment gives random observations to the agent and the agent tries to predict the attack type. The agent gets a reward of 1 if its action matches the attack type, otherwise 0

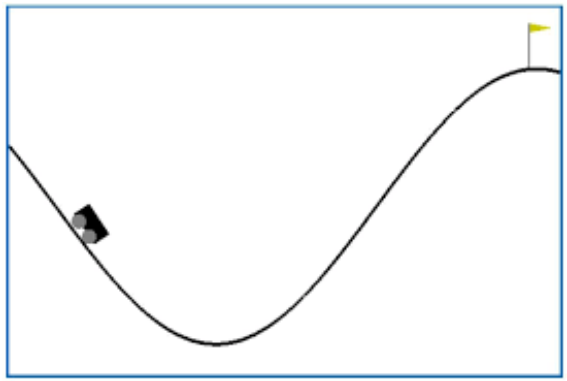

Figure: Mountain Car: A common Gym environment for training RL agents

Initial Results

In our initial test we achieve a test set accuracy of 0.38 on broad category prediction after training for 10000 episodes, which took about 3 minutes, and 0.76 on the binary prediction task.

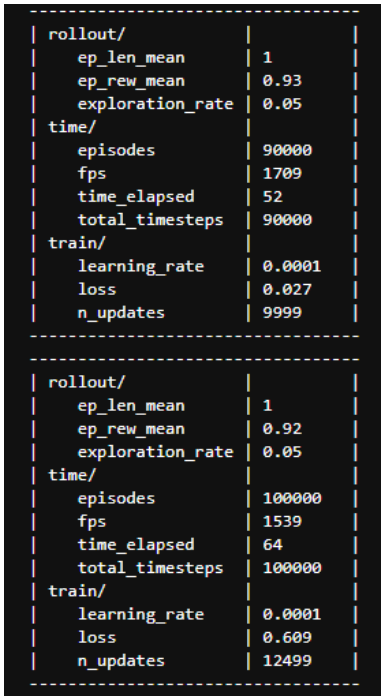

Figure: snippet of training statistics

Final Refinements

- Improve environment: The environment can choose actions deliberately to try to fool the agent in an adversarial setting.
- Collect and compare metrics in the binary and broad label categories against the baseline XGBoost models. This must include comparisons of training time and predicting time, which can be the main benefits of the RL Method.


Richard S Sutton and Andrew G Barto.
Reinforcement learning: An introduction.
MIT press, 2018.