Reinforcement Learning on Nim

Marten Thompson
Department of Statistics, University of Minnesota

December 17, 2020

Introduction

Traditional statistics and machine learning primarily focus on supervised and unsupervised learning tasks. While these methods find great success in a rich set of problems, they are not applicable to many scenarios we might otherwise understand to be common learning experiences: bargaining prices, operating machinery, and playing strategy games. For such tasks we require methods comprising the field of *reinforcement learning* where the objective is to understand an environment and maximize our reward when acting on it.

Strategy games provide an attractive environment to examine reinforcement learning methodologies. They are fixed in the sense of unchanging rules, legal moves, and possible outcomes, even if these are quite large in number. Also, they offer clear rewards: winning and losing. Finally, it is often easy to simulate game play thereby affording ample training.

In particular, we investigate the game of Nim, a simple paper-and-pencil game with a known winning strategy [1]. Two players share a board consisting of K piles of $\{n_1, \dots, n_k\}$ $0 < n_i < \infty$ pieces (arbitrary coins, dots, tokens, etc.). Games proceed in a typical turn-by-turn fashion where each player is allowed to take one or more pieces from one pile only. The objective is to take the final piece(s) thus clearing the board. For example, a game with $K = 3$ piles containing $\{1, 2, 3\}$ pieces may proceed like:

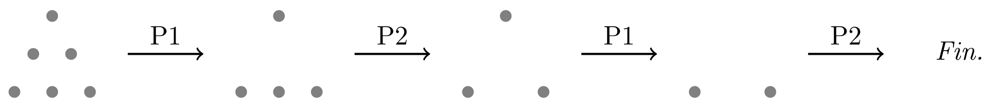


Figure 1: Player 2 wins a game of Nim

In this example we see each player's ability to take one piece, all, or any amount in between from a single pile.

The Game of Nim

While reinforcement learning algorithms can best human players in many games based on heuristics and their own experience, mathematicians have also completely solved many strategy games. That is, they have developed a strategy that is proven to result in a win (or draw) when followed exactly. You yourself may have organically come to this in tic-tac-toe where you can always win or draw against your opponent. Nim's winning strategy was developed early last century [1]. Whether or not reinforcement learning algorithms learn this strategy will be the focus of our report.

Representing a Nim board state as $\{s_1, s_2, \dots, s_K\}$, an optimal move results in a board whose term-wise bit sum is zero. For example, if player 1 removes pieces resulting in the board $\{5, 4, 1\}$ (bit sum = 0), player 2 is unable to win if player 1 continues to take pieces according to this policy. Given two perfect players then, the outcome of the game is determined as soon as the board is revealed. Of course, we don't have perfect players, and will investigate various RL algorithms' ability to recover this strategy.

Reinforcement Learning

Reinforcement learning abstracts learning as a Markov Decision Process, in our case marked by discrete turns [3]. We say *process* because at each turn t a player (also called an agent) at state s_t takes an action

a_t and receives reward r_t forming the sequence of play

$$(s_0, a_0, r_0) \rightarrow (s_1, a_1, r_1) \rightarrow \dots$$

We also assume all of the information of the environment at turn t is encoded in s_t, including the effects of past moves. That is, s_{t+1} and r_{t+1} depend only on s_t, a_t, called the *Markov property.*

Q-Learning

A Q-learning agent estimates the value of an action a at a given state s. These estimates are stored in a *Q-table* $Q(s,a)$ which is updated each turn $t \rightarrow t+1$ according to

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a \in A_t} Q(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{1}$$

where $0 \leq \alpha_t < 1$ is the learning rate, r_{t+1} is the reward received for having taken action a_t, $0 \leq \gamma \leq 1$ controls the back propagation of learning, and A_t is the set of available actions from state s_t [3]. Note that $r_{t+1} = 0$ until the final move of a game at which point $r_{t+1} = 1$ if the agent won, otherwise $r_{t+1} = -1$. Under mild additional conditions this has been proved to converge to the true value-action function q^* as $t \rightarrow \infty$ with probability 1. They are: $|r_t|$ bounded, $\sum_{t=1}^{\infty} \alpha_t < \infty, \sum_{t=1}^{\infty} \alpha_t^2 = \infty$, and the guiding policy must visit all state-action pairs[2].

Note that this learning occurs irrespective of the policy guiding the Q-agent's actions. Such learners are called *off-policy* and require the guiding policy be specified separately. An ϵ-greedy policy is commonly chosen and accommodates both leveraging what is learned and exploration of new moves. Under such a policy, an agent chooses

$$a_t = \begin{cases} \arg\max_a Q(s_t, a) & \text{wp } 1 - \epsilon \\ a \sim \text{Uniform}(A_t) & \text{wp } \epsilon \end{cases}$$

In the next section, we examine both ϵ fixed and $\epsilon_t \rightarrow 0$ cases (referred to as Q- and Q_t-learning, respectively). The latter case attempts to reduce exploration as learning progresses, eventually choosing valuable moves with high probability. Specifically, such an agent moves randomly with probability $\epsilon_t = \epsilon e^{-\eta t}$.

Quasi-Bayesian Learning

This need to both leverage experience and to change exploration behavior over time motivated our creation of an *on-policy* quasi-Bayesian approach to reinforcement-learning. This approach allowed us to treat the estimated value of a given action $V_t(s,a)$ as a random variable whose distribution is initially broad but concentrates over time. More specifically, we assume values $V_t(s,a)$ are independent normal random variables with conjugate priors $\pi_{\mu(s,a)}^{(t)}$ and $\pi_{\tau(s,a)}^{(t)}$ on the mean and precision:

$$V_t(s,a) \sim \mathcal{N}\big(\mu_t(s,a), \tau_t(s,a)^{-1}\big); \qquad \text{where } \mu_t(s,a) \sim \pi_{\mu(s,a)}^{(t)} \text{ and } \tau_t(s,a) \sim \pi_{\tau(s,a)}^{(t)}$$

When playing a game of Nim, the agent acts on-policy by comparing realizations of $V_t(s_t, a)$ for $a \in A_t$ and choosing the action corresponding to the largest realized value.

Learning proceeds as follows. After a complete game and given sequence of T turns:

$$(s_0, a_0, r_0 = 0) \rightarrow (s_1, a_1, r_1 = 0) \rightarrow \dots \rightarrow (s_T, a_T, r_T = \pm 1)$$

we apply discount rate $d \in (0, 1)$ to the final reward

$$(s_0, a_0, r'_0 = d^T \cdot r_T) \rightarrow (s_1, a_1, r'_1 = d^{T-1} \cdot r_T) \rightarrow \ldots \rightarrow (s_T, a_T, r'_T = r_T) \rightarrow$$

Then, we update our priors based on the the discounted reward r'_t received to their usual Bayesian posteriors. Initially at $t = 0$ we set conjugate priors $\pi^{(0)}_{\mu(s,a)} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \frac{1}{2})$ and $\pi^{(0)}_{\tau(s,a)} \overset{\text{i.i.d.}}{\sim} \text{Gamma}(\frac{1}{2}, 1)$ independent $\forall s, a$. The updates to the priors are thus:

$$\pi^{(t)}_{\mu(s,a)} = \mathcal{N}(m, \nu^{-1}) \xrightarrow{assign} \pi^{(t+1)}_{\mu(s,a)} = \mathcal{N}\left(\frac{m\nu + r'_t}{\nu + 1},\ \nu^{-1} + 1\right) \tag{2}$$

$$\pi^{(t)}_{\tau(s,a)} = \text{Gamma}(a, b) \xrightarrow{assign} \pi^{(t+1)}_{\tau(s,a)} = \text{Gamma}\left(a + \frac{1}{2},\ b + \frac{1}{2} + \frac{\nu}{\nu + 1}\frac{(r'_t - m)^2}{2}\right) \tag{3}$$

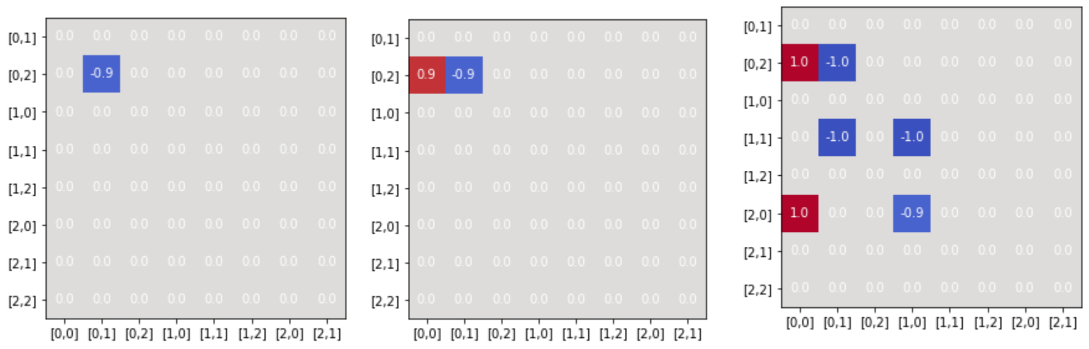
In this way, exploration is always possible as all $V_t(s, a)$ have the same support and any may realize the largest value on a given turn. But as priors are updated, more valuable actions are favored: the location of of $\pi^{(\cdot)}_{\mu(s,a)}$ increases or decreases to reflect historical rewards, and its precision increases. Similarly, the mean of $\pi^{(\cdot)}_{\tau(s,a)}$ decreases with each passing game.

We qualify this approach as *quasi*-Bayesian because the full Bayesian treatment would be to use the posterior predictive distribution of $V_t(s, a)$ when choosing the next action. However, this is a t-distribution which leads to an analytic dead end: there are no conjugate priors for its parameters, and our update steps above would have to resort to sampling approximations e.g. MCMC. Hence, we recycle the normal likelihood and conjugate priors described above.

Results

Learning Mechanisms

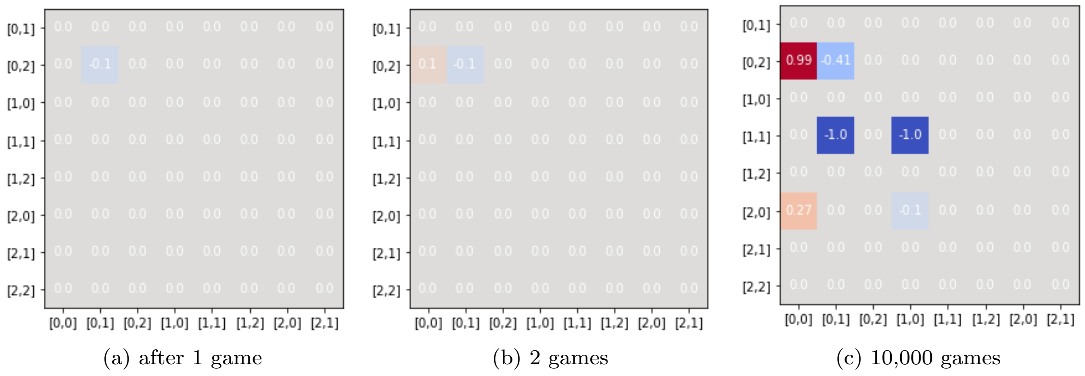
Before reporting how each agent performed, we first illustrate the learning processes described symbolically above. We consider a simple Nim board for illustration purposes - $\{2, 2\}$ - and examine how a Q-learner updates its Q-table via equation 1. Below, we compare 'aggressive' agent and 'conservative' agent. The former uses $\alpha = \gamma = 0.9$ to promote faster back propagation of results. The conservative learner ($\alpha = \gamma = 0.1$) instead makes updates to its Q-table in smaller increments.



(a) after 1 game (b) 2 games (c) 1,000 games

Figure 2: Aggressive Q-learner ($\epsilon : 0.1, \alpha : 0.9, \gamma : 0.9$)

The vertical axis lists starting positions, and the horizontal axis represents the state of the board after the agent acts. Note that zero values are not legal or just not visited. After 1,000 turns, the aggressive Q-agent has visited and learned winning and losing moves almost completely. Below in figure 3, the slower learning agent has significantly smaller values after 10,000 games. Eventually it learned a similar Q-table, but only after approximately 50,000 games.



(a) after 1 game (b) 2 games (c) 10,000 games

Figure 3: Conservative Q-learner ($\epsilon : 0.1, \alpha : 0.1, \gamma : 0.1$)

There are strengths and weaknesses to both approaches. An aggressive learner might begin to favor what it perceives to be valuable moves sooner, but a non-optimal move might have resulted in a win due to the randomness of an ϵ-greedy opponent. More incremental updates might make it easier to correct poor estimates.

We witnessed a similar phenomenon with the Bayes agents with aggressive and conservative learning behavior. Recall that this model treated the values in figures 2 and 3 as normal random variables $V_t(s,a) \sim \mathcal{N}(\mu_t(s,a),\ \tau_t(s,a))$ instead of fixed points. A discount rate d controlled the degree to which the final reward r_T was back-propagated when updating the priors (equations 2 and 3). We compare an aggressive learning strategy $(d = 0.9)$ to a more conservative one $(d = 0.5)$ in figure 4 below. Each line represents $\mathbb{E}[\mu_t(\cdot,\cdot)]$ for a given legal pair (s,a) pairs on the same $\{2,2\}$ game considered above. Line colors refer to the same (s,a) pair in all plots.

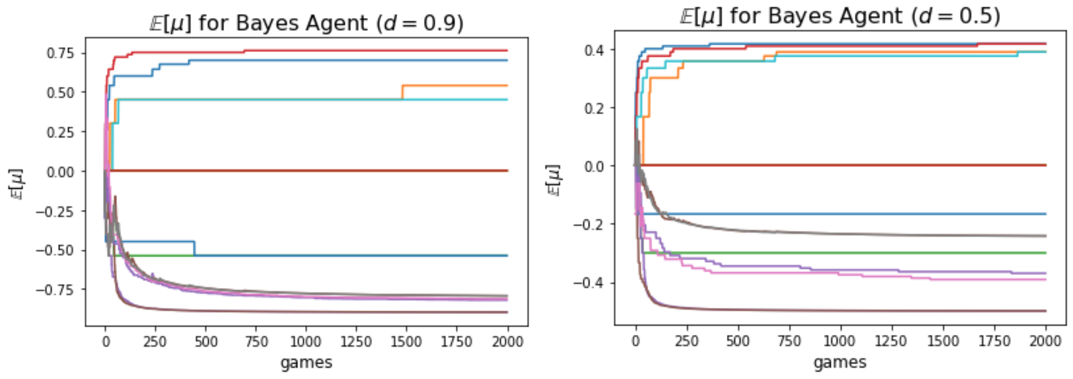


Figure 4: Hyperparameter Moments for Bayes Agents

First, we see that both learners quickly differentiate between advantageous and disadvantageous moves. Like the Q-agents above, the larger discount rate increases the rate at which each moment converges. Comparing the drift of each mean, we see the flexibility of this variable learning strategy. Unlike a strict $\epsilon_t \to 0$

guiding behavior all states, we see how the exploration is attenuated at different rates for different board positions under this approach.

Learning Performance

Armed with a more complete understanding of each algorithm's behavior with respect to its parameterization, we can now examine the rate at which different agents learn optimal play. We measure the proportion of optimal plays an agent made per game over a sequence of many thousands of games; both this and win rate are comparable (see figure 6.a in the appendix). Each game started from a random board of three piles $\{n_1, n_2, n_3\}$ where $n_i \in \{0, 1, \ldots, 9\}$ not all zero.

All simulations were repeated ten times and averaged. Agents played an independent copy of themselves for 75,000 games. We considered all 75 combinations of Q-agents with parameters $\epsilon \in \{0.1, 0.3, 0.5\}$, $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, and $\gamma \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. Choosing the strongest of these learners, we investigated $\epsilon_t \to 0$ with the following 36 parameterizations of a Q_t agent: $\epsilon \in \{0.1, 0.5, 0.9\}, \alpha \in \{0.1, 0.3\}, \gamma \in \{0.5, 0.9\}, \eta \in \{0.001, 0.0001, 1\text{e-}5\}$. Finally, we considered Bayes agents with discounts $d \in \{0.1, 0.5, 0.9, 0.99\}$. See figure 6 in the appendix for an overview of their learning progress.

Finally, in the likely-anticipated conclusion to these trials, we pitted the three highest performing learners against each other, copies of themselves, and a randomly-moving opponent, all initialized to an untrained state. The results in figure 5 displays each learner's win rate when they are the first player, averaged across five games with each of its four opponents. We see that the Bayes agent clearly outperforms the Q-agents. Furthermore, it appears as ig the Q_t agent never gets an opportunity to learn, winning only half of its games, those against its copy and the random agent.

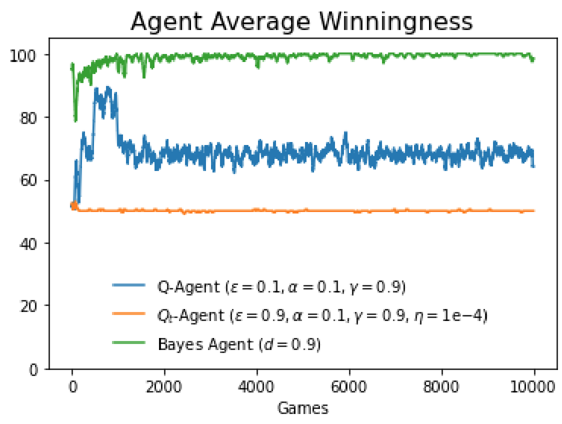


Figure 5: Average wins across different opponents

Conclusion

Through our examination of the Q-learning algorithm we saw the effects of conservative and aggressive learning strategies. A quasi-Bayesian formulation of a Q-agent was a natural on-policy extension that balanced early exploration with choices refined by experience. We believe part of its strength lay in it doing so in a manner specific to each state-action pair. Ultimately, we saw how effective approach was against two variants of Q-learning whereby it went largely undefeated.

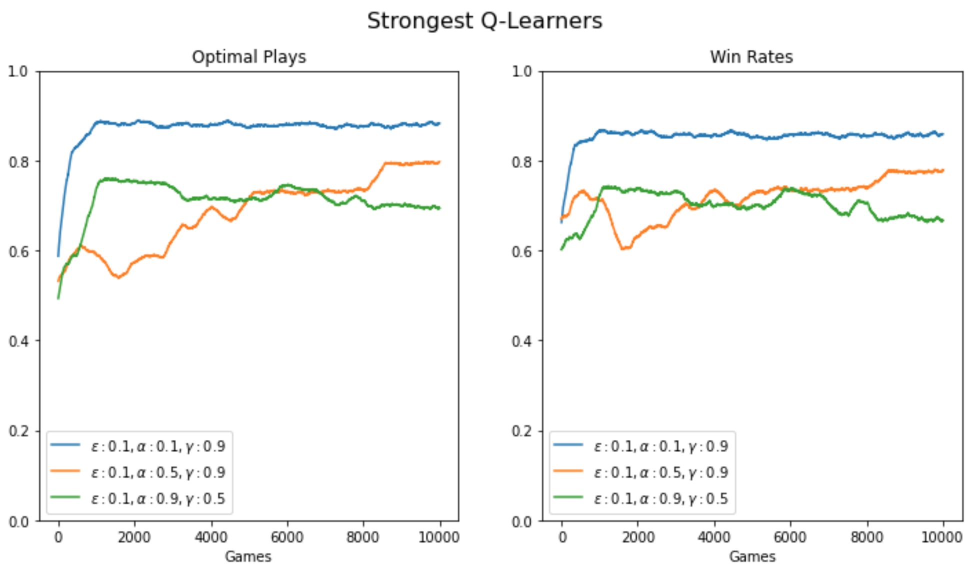
References

[1] Charles L Bouton. Nim, a game with a complete mathematical theory. *The Annals of Mathematics*, 3(1/4):35–39, 1901.

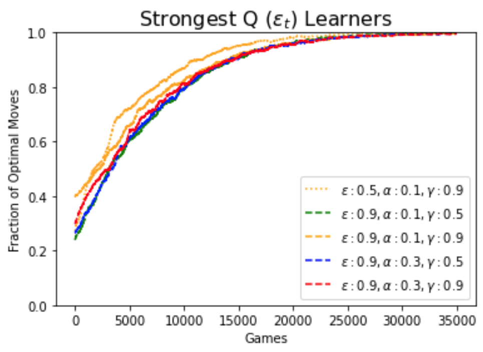
[2] Francisco S Melo. Convergence of q-learning: A simple proof. *Institute Of Systems and Robotics, Tech. Rep*, pages 1–4, 2001.

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

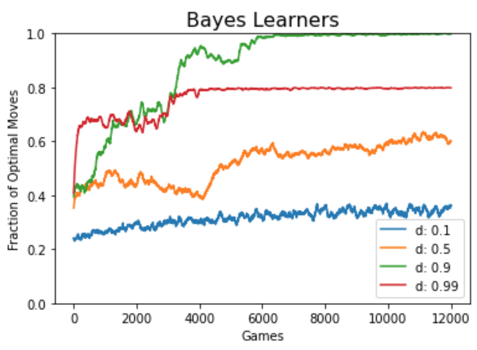
Appendix



(a) three best Q-learners



(b) five best Q -learners with $\epsilon_t \to 0$



(c) all Bayes learners

Figure 6

In figure 6.a we see the unsurprising similarity between optimal plays and win rates and chose to only report one in the body of our report. In (c), note the time to flawless play is much shorter for the best Bayes agent than for the Q_t agents in (b).