Deep Reinforcement Learning with POMDPs

Maxim Egorov

December 11, 2015

1 Introduction

Recent work has shown that Deep Q-Networks (DQNs) are capable of learning human-level control policies on a variety of different Atari 2600 games [1]. Other work has looked at treating the Atari problem as a partially observable Markov decision process (POMDP) by adding imperfect state information through image flickering [2]. However, these approaches leverage a convolutional network structure [3] for the DQN, and require the state space to be represented as a two dimensional grid. This approach works well on problems where the state space is naturally two dimensional (i.e. Atari screen), but does not generalize to other problems. This project aims to extend DQNs to reinforcement learning with POMDPs without the limitations of a two dimensional state-space structure. In this project we develop a novel approach to solving POMDPs that can learn policies from a model based representation by using a DQN to map POMDP beliefs to an optimal action. We also introduce a reinforcement learning approach for POMDPs that maps an action-observation history to an optimal action using a DQN.

2 Partial Observability

In many real-world applications, the complete state of the environment is not known to the agent. In these cases, the agent receives an observation that is conditioned on the current state of the system, and must make a decision based on the observations it receives as it acts in the environment. These types of problems can be modeled as POMDPs. A POMDP can be formalized as a 6-tuple $(\mathcal{S}, \mathcal{A}, T, R, \mathcal{Z}, O)$, where $\mathcal{S}, \mathcal{A}, T, R$ are the states, actions, transitions and rewards as in a Markov decision process. The $\mathcal{Z}, O$ are the observation space and the observation model respectively. When the POMDP model is known, the agent can update its belief $b(s)$ as it interacts with the environments The belief defines the probability of being in state s according to its history of actions and observations. Systems with discrete beliefs can be updated exactly using

$$b'(s') \propto O(s', a, o) \sum_{s \in \mathcal{S}} T(s, a, s') b(s). \tag{1}$$

Many approaches exist for approximating optimal POMDP policies when the model of the environment is known. The state-of-the-art solver known as SARSOP [4] attempts to sample the optimally reachable belief space in order to efficiently compute a POMDP policy. In this work, SARSOP will be used as a benchmark for testing the DQN policies. A POMDP policy can be represented as a collection of alpha-vectors denoted Γ. Associated with each of the alpha-vectors is one of the actions. The optimal value function for a POMDP can be approximated by a piecewise-linear convex function that takes the form

$$V(b) = \max_{\alpha \in \Gamma} (\alpha \cdot b). \tag{2}$$

If an action associated with an alpha vector α maximizes the inner product $\alpha \cdot b$, then that action is optimal. Reinforcement learning has also been used in the context of POMDPs by using function approximation to represent a stochastic policy [5]. We use a similar approach to compare against the DQN approach to reinforcement learning with POMDPs and refer to it as POMDP RL.

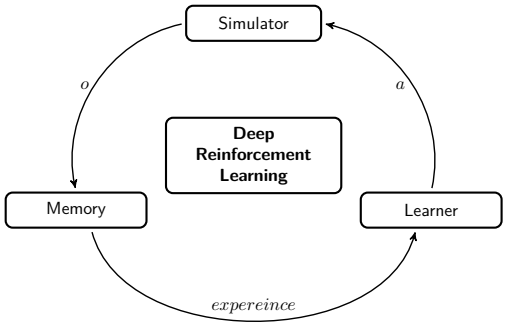

Figure 1: Information flow in deep reinforcement learning with a generalized problem simulator

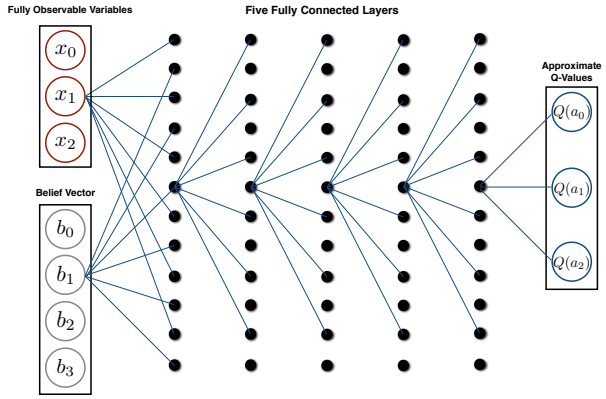

Figure 2: Five layer fully connected network that maps the concatenated fully observable variable and belief vectors to Q-values

3 Deep Reinforcement Learning

In reinforcement learning, an agent interacting with its environment is attempting to learn an optimal control policy. At each time step, the agent observes a state s, chooses an action a, receives a reward r, and transitions to a new state s'. Q-Learning estimates the utility values of executing an action from a given state by continuously updating the Q-values using the following rule:

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)). \tag{3}$$

Learning the Q-values for POMDPs using the approach above is intractable, because a Q-value would be needed for each possible belief or for arbitrary long action-observation histories. However, a function approximator, such as a neural network, can be used to approximate the POMDP Q-values. In Deep Q-Learning, a neural network is used to approximate the Q-values in a decision process. For a POMDP, the Q-values are parameterized by either the belief and the action $Q(b,a)$ or an action-observation history h and an action $Q(h,a)$. These modified Q-values can be learned by a neural network that is characterized by weights and biases collectively denoted as θ. These Q-values can be denoted by $Q(b,a|\theta)$ and $Q(h,a|\theta)$. Instead of updating each Q-value, we can now update the parameters of the neural network by minimizing the loss function:

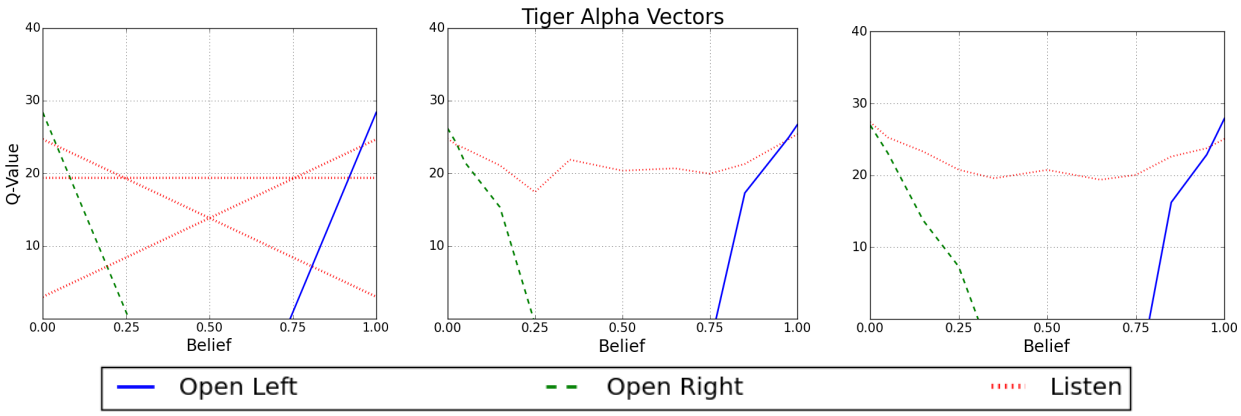

Figure 3: The value function surfaces for the Tiger problem for the SARSOP alpha-vectors (left), DQN converged policy (middle), and DQN non-converged policy (right)

$$\mathcal{L}(b, a|\theta_i) = (r + \gamma \max_a Q(b', a|\theta_i) - Q(b, a|\theta_i))^2, \tag{4}$$

where $\theta_{i+1} = \theta_i + \alpha \nabla_\theta \mathcal{L}(\theta_i)$. However, this approach to updating the Q-values can lead to divergence. There are three techniques used to stabilize the learning. First, experience replay tuples (b, a, r, b') are recorded in a reply memory, and are then sampled uniformly. Second, a separate target network is used to provide state update targets to the main network. Lastly, an adaptive learning method known as RMSProp is used to regulate the parameter adjustment rate of the network. This framework is decomposed into three components: the simulator, memory and learner (Fig. 1). The simulator can be a POMDP model or an Atari emulator, while the learner contains a function approximator which is a DQN in this project.

The architecture of the DQN is shown in Fig. 2. We adopt a similar framework to [1], where a simulator is used to populate and experience replay dataset. In the figure the input layer to the network consists of the belief of the agent, and the fully observable state variables. The fully observable state variables generalize this representation to problems where the agent may have some knowledge of the system state, known as mixed observability MDPs (MDOMPs). A similar architecture was used for training the DQN on action observation histories as well. Once again, if the problem has fully observable state variables, those are also used as inputs to the network. The current formulation uses a fully connected network that either takes the fully observable state variables x and the belief b or just the belief b and outputs a value approximation for each action available to the agent. The DQN hyper-parameters used in this project can be found in Section 7.

4 Evaluation and Results

This famework was tested on two popular benchmark problems found in the literature: Tiger and Rock Sample. In the tiger problem, the agent has to choose between opening a door on the left or on the right. If the agent opens the door with tiger, it receives a reward of -100, if it opens the other door it receives a reward of 10 and escapes. The agent can listen, and receive a noisy measurement of where the tiger may be located. The true alpha-vectors for this problem were obtained using SARSOP, and are shown in Figure 3. The DQN for this problem was evaluated at various belief points to construct a value function surface. The value function surface for the converged DQN closely resembles the surface formed by the alpha-vectors. The value surface for a non-converged

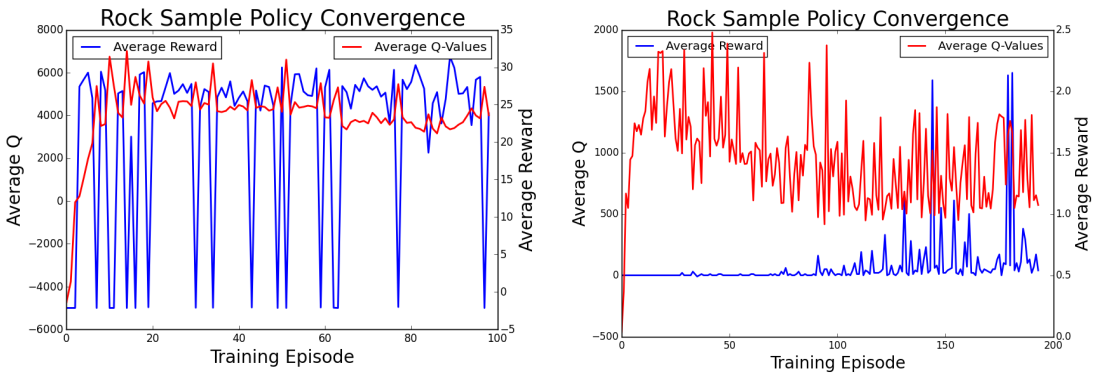

Figure 4: Policy and Q-value convergence for the tiger (left) and rock sample (right) problems

DQN is shown for reference. While the difference between the converged and non-converged value surfaces are small, the primary difference in policy comes from the listen action dominating the open-right action at small belief values.

In the rock sample problem, a rover is tasked with sampling a set of eight rocks on a 7×7 gridded map. The goal is to sample rocks that are good and to avoid sampling bad rocks. The rover can make a noisy measurement of the rock quality, with the accuracy of the measurement parametrized by the distance between the rock and the rover. Rock sample is a MOMDP, because the rover knows its own position exactly, but does not know the state of the rocks. This is a fairly large problem with $\sim 12,000$ states, 13 actions, and 2 observations. Nearly optimal policies took about 5 minutes of training for the tiger problem and about 6 hours of training for the rock sample problem. Note that it takes SARSOP less than 1 second to compute the tiger policy and about 5 minutes to compute the rock sample policy. However, SARSOP needs explicit knowledge of the model to determine the alpha vectors, while the DQN can be trained using just action-observation histories.

One of the more significant results from this project was identifying the differences in convergence behavior of the Q-values and the policies. These are demonstrated in Fig. 4, where the average Q-values of the network are evaluated at each episode, and the policy is evaluated as well (and is scored using an average reward). The figure shows that for both of the problems the Q-values converge, while the policies do not. For the tiger problem, the jumps in the policy evaluations can be attributed to the small difference in value function surfaces between two actions as seen in Fig. 3. Despite the lack of policy convergence, the DQN can still learn very good policies as indicated by the upward spikes in the average reward values. For both the belief approach and the history approach, these policies were evaluated and compared to the SARSOP policies and the POMDP RL approach from [5]. The results of the evaluations are shown in Table 1. For the tiger problem, the DQN approaches are able to perform just as well as the optimal SARSOP policy. For the rock sample problem, the belief DQN performs better than the history DQN. Both the DQN approaches outperform the POMDP RL approach.

5 Conclusion and Future Work

This project introduced a novel approach of solving POMDPs using a DQN. We demonstrated that DQNs can learn good policies, but require significantly more computational power. We also showed that while the Q-values converge, the policies are sensitive to small perturbations, and do not converge even after long training cycles.

Table 1: Average rewards per time-step for the Tiger and the Rock Sample problems using the SARSOP, DQN and POMDP RL policies. The rewards ere averaged over 100 trials.

	Belief DQN	History DQN	SARSOP	POMDP RL
Tiger	1.08 ± 0.1	1.07 ± 0.1	1.12 ± 0.1	1.06 ± 0.1
Rock Sample	1.65 ± 0.1	1.43 ± 0.2	1.75 ± 0.1	1.14 ± 0.2

The first area of future work would be to determine if the two problems examined in this project have structure that leads to policy instabilities, or if these instabilities are present in all POMDP problems. An approach to dealing with unstable policies is to try and stabilize the alpha-vectors of the POMDP. This can be done by using a policy gradient approach as opposed to using Q-learning. Another area of future work would attempt to improve the Q-value convergence rate by first creating a two-dimensional representation of the belief space or of the action observation history space using a self-organizing map (SOM). The output of the SOM would be used in a convolutional neural network, which would take advantage of locality information in belief space.

6 Acknowledgments

I would like to thank Yegor Tkachenko without whom this project would not have been possible. He implemented the deep learning backend for this project and ran many of the experiments.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 02 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[2] M. Hausknecht and P. Stone, "Deep Recurrent Q-Learning for Partially Observable MDPs," *ArXiv e-prints*, Jul. 2015. [Online]. Available: http://arxiv.org/pdf/1507.06527v3.pdf

[3] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015, published online 2014; based on TR arXiv:1404.7828 [cs.NE].

[4] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *Robotics: Science and Systems*, Zurich, Switzerland, June 2008.

[5] J. Baxter and P. L. Bartlett, "Reinforcement learning in pomdp's via direct gradient ascent," in *In Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, 2000, pp. 41–48.

7 Appendix: Deep Q-Network Hyperparameters

The hyperparameters used for training the DQN can be found in Table 2

Table 2: Deep Q-Network hyperparameters

Hyperparameter	Value	Decription
Max train iterations	500000	The maximum number of training samples generated
Minimbatch size	32	Nnumber of training samples per update in stochastic gradient descent
Target network update	1000	Frequency of updating the target network
Replay size	100000	Size of the experience replay dataset
Learning rate	0.001	Rate used by RMSProp
Initial exploration	1.0	Initial ϵ value in ϵ greedy exploration policy
ϵ decay	0.0001	Rate at which ϵ decreases
Max history	20	The maximum number of samples kept for action-observation histories