Continuous Deep Q Learning

Anirudh Goyal

Outline

- DQN => Robotics
- Needs to be sample efficient.
- Standard/Model free RL: experience -> policy
- Model based RL -> model -> policy

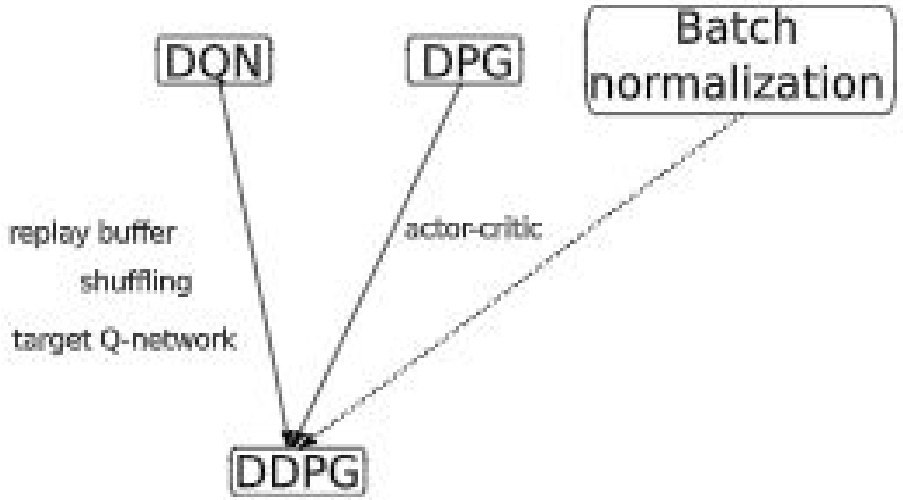
Model based can be very sample efficient (if it works)

Practically, Model is never good enough for many tasks.

Reinforcement Learning: Model free ?

- When system dynamics, p(x_t+1 | x_t, u_t) are not known.
- We define the Q function, corresponding to a policy as the expected return from x_t after taking u_t and following policy thereafter.
- Q learning learns a greedy deterministic policy.
  - **NOTE! Adversarial examples can exploit these!**
- Learning objective is to minimize the bellman error.

DDPG: ancestors



- Most of the actor-critic theory for continuous problem is for stochastic policies (policy gradient theorem, compatible features, etc.)
- DPG: an efficient gradient computation for deterministic policies, with proof of convergence

Continuous Q learning with Normalized Advantage function

- Using a value/advantage representation of Q(s,a), restricting A(s,a) to be quadratic w.r.t. 'a' and parametrized by a neural net.
- The Representation of the advantage function using a quadratic whose mean and covariance are the outputs of a neural network. This makes practical deep Q-learning possible in the continuous state/action setting.

NAF: Approximate the advantage function

- Reminder in Q learning, high cost to select best action.
- Here set a specific form to Q function, so as to find the best action easily.

Advantage function: $A(s_i, a_i|\theta) = Q(s_i, a_i|\theta) - max_a Q(s_i, a|\theta)$

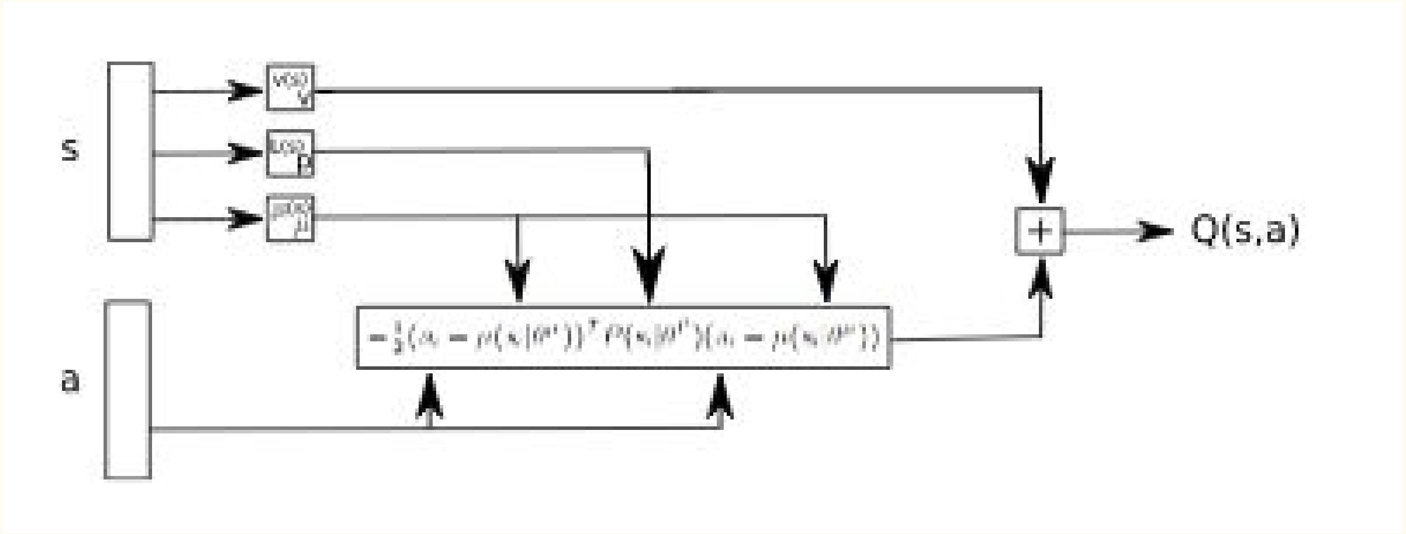
$$V(s_i) = max_a Q(s_i, a|\theta)$$

$$Q(s_i, a_i|\theta^Q) = A(s_i, a_i|\theta^A) + V(s_i|\theta^V)$$

$$A_\theta(s_i, a_i|\theta^A) = -\frac{1}{2}(a_i - \mu(s_i|\theta^\mu))^T P(s_i|\theta^P)(a_i - \mu(s_i|\theta^\mu))$$

NAF: The Network

- All neural nets are dim(s) X dim(a)
- Implemented with 2 layers of 200 relu units
- The μ network is the actor
- Outperforms DDPG on some benchmarks.



s
V(s)
L(s)
μ(s)
a
$-\frac{1}{2}(a_t - \mu(x_t|\theta^\mu))^T P(x_t|\theta^P)(a_t - \mu(x_t|\theta^\mu))$
+
Q(s,a)

Advantage over Actor Critic

- Quadratic assumption acts as a regularizer.
- Learning signal for all of the estimators comes directly from the environment.
- In actor-critic methods, the learning signal for the policy comes indirectly through Q(s,a), which is unlikely to be a good approximation early in training and may have wacky gradients if it's a big deep network (see, e.g., papers on adversarial examples for convnets).
- Defining the advantage function as being quadratic assumes that the state only has one "optimal" action (or, that all good actions are centred around the same mode), but for many tasks several different actions may be "optimal". One simple extension would be to define the advantage function as a sum of k quadratic functions. For Q-learning this would require k evaluations.

Experiments

- Black Curve - DDPG with

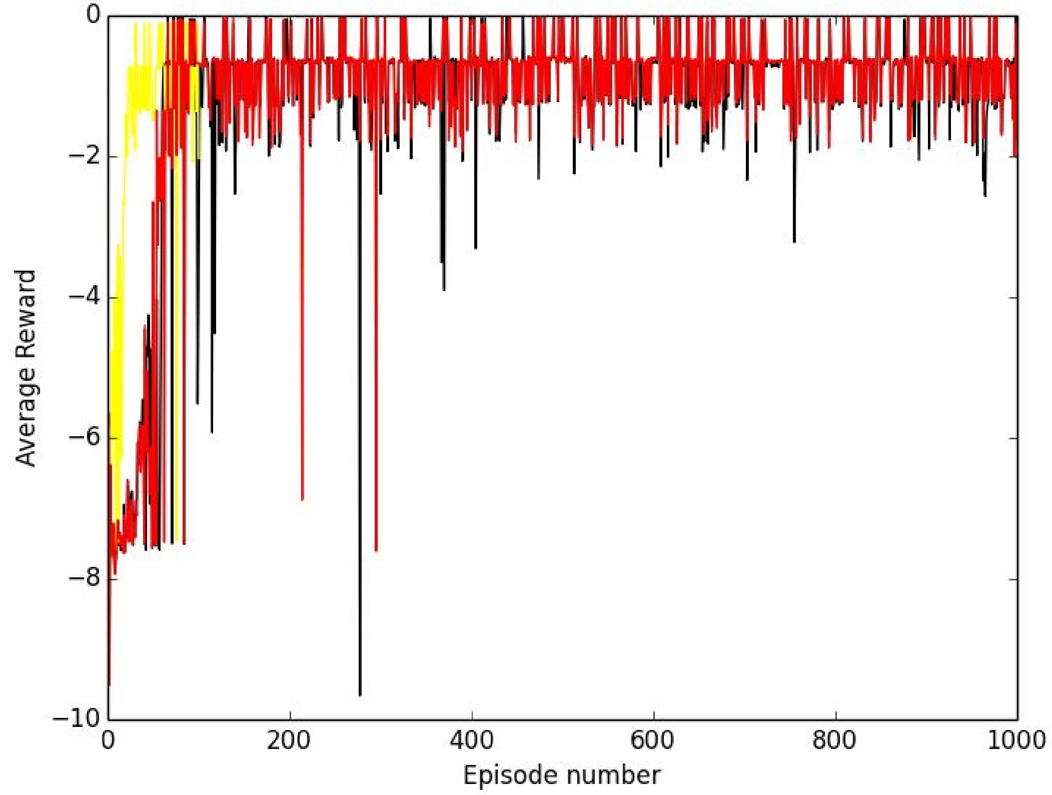
1000 steps/episode.

- Red Curve - DDPG with

200 steps/episode.

- Yellow Curve - Normalized

Advantage function, 100 steps/episode.



Conclusion!

- Applying Q-learning to problems with continuous states and actions while using neural networks for function approximation.
- Choosing to represent the Q-function in terms of a value function and advantage function.
- Parametrizing the value function so that it's quadratic w.r.t. the action.
- Directly estimating the mean and covariance of the quadratic advantage function using a deep network makes it possible to evaluate the max of the Q-function using a simple forward pass through the value and advantage function networks.

Questions ?