CS4910: Deep Learning for Robotics

David Klee
klee.d@northeastern.edu

T/F, 3:25-5:05pm
Behrakis Room 204

https://www.ccs.neu.edu/home/dmklee/cs4910_s22/index.html

https://piazza.com/northeastern/spring2022/cs4910a/home

CONTINUOUS CONTROL WITH DEEP REINFORCEMENT LEARNING

Timothy P. Lillicrap[*], **Jonathan J. Hunt**[*], **Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver & Daan Wierstra**
Google Deepmind
London, UK
{countzero, jjhunt, apritzel, heess, etom, tassa, davidsilver, wierstra} @ google.com

An obvious approach to adapting deep reinforcement learning methods such as DQN to continuous domains is to to simply discretize the action space. However, this has many limitations, most notably the curse of dimensionality: the number of actions increases exponentially with the number of degrees of freedom. For example, a 7 degree of freedom system (as in the human arm) with the coarsest discretization $a_i \in \{-k, 0, k\}$ for each joint leads to an action space with dimensionality: $3^7 = 2187$. The situation is even worse for tasks that require fine control of actions as they require a correspondingly finer grained discretization, leading to an explosion of the number of discrete actions. Such large action spaces are difficult to explore efficiently, and thus successfully training DQN-like networks in this context is likely intractable. Additionally, naive discretization of action spaces needlessly throws away information about the structure of the action domain, which may be essential for solving many problems.

Page 1: Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." *arXiv preprint arXiv:1509.02971* (2015).

In this work we present a model-free, off-policy actor-critic algorithm using deep function approximators that can learn policies in high-dimensional, continuous action spaces. Our work is based on the deterministic policy gradient (DPG) algorithm (Silver et al., 2014) (itself similar to NFQCA (Hafner & Riedmiller, 2011), and similar ideas can be found in (Prokhorov et al., 1997)). However, as we show below, a naive application of this actor-critic method with neural function approximators is unstable for challenging problems.

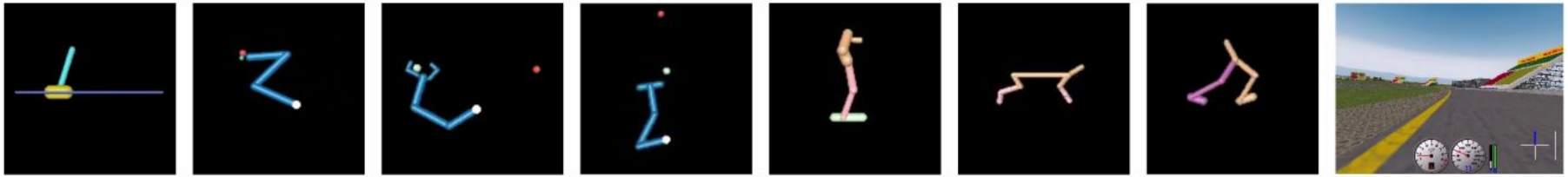
Page 1&2: Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." *arXiv preprint arXiv:1509.02971* (2015).

Background

Algorithm

https://medium.com/geekculture/introduction-to-deterministic-policy-gradient-dpg-e7229d5248e2

Tasks



Comparison to baselines and ablations

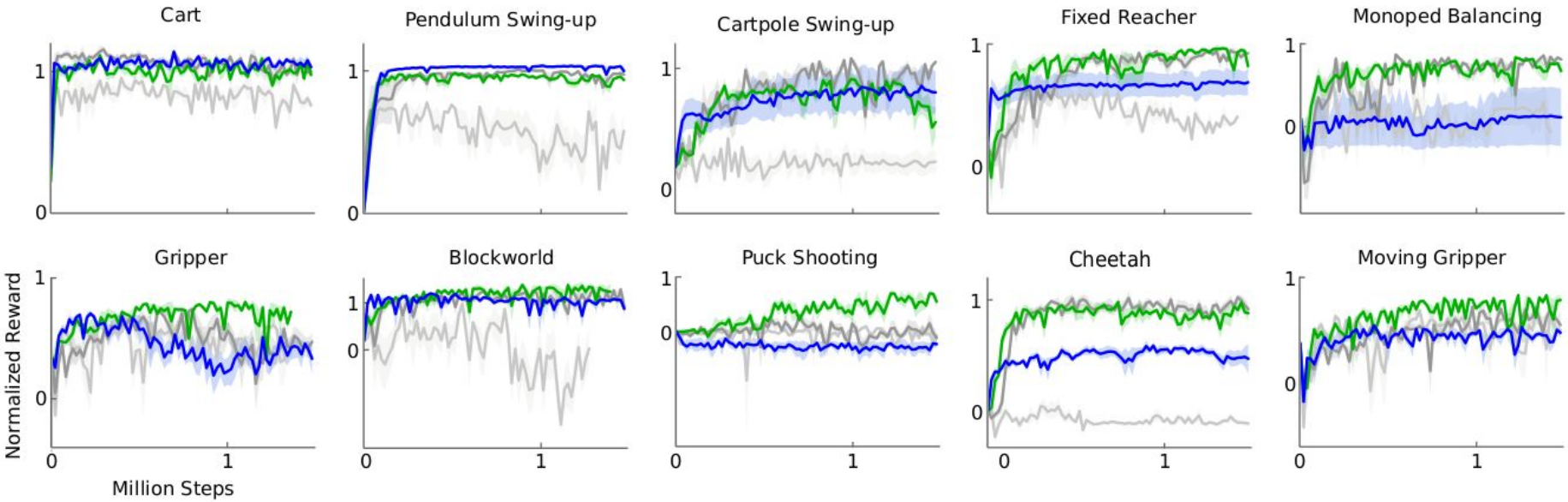


Figure 2: Performance curves for a selection of domains using variants of DPG: original DPG algorithm (minibatch NFQCA) with batch normalization (light grey), with target network (dark grey), with target networks and batch normalization (green), with target networks from pixel-only inputs (blue). Target networks are crucial.

Investigation into Q-value estimates

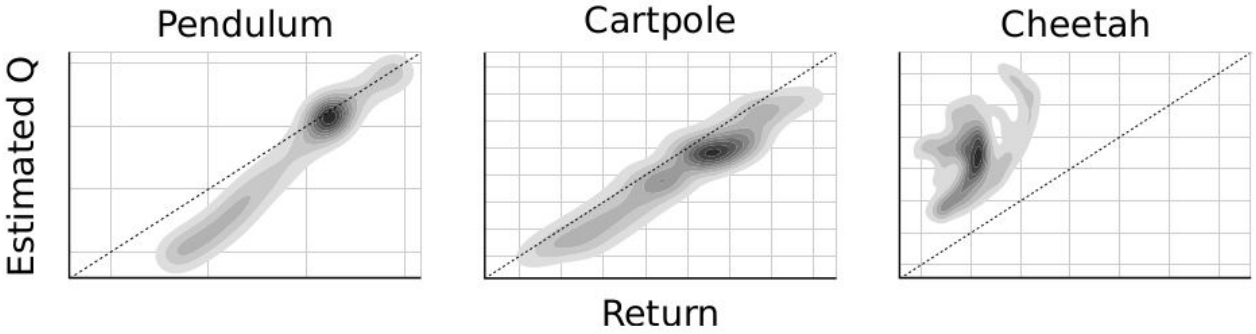


Figure 3: Density plot showing estimated Q values versus observed returns sampled from test episodes on 5 replicas. In simple domains such as pendulum and cartpole the Q values are quite accurate. In more complex tasks, the Q estimates are less accurate, but can still be used to learn competent policies. Dotted line indicates unity, units are arbitrary.

Takeaways about paper organization

Introduction: motivates the work by situating it in relation to literature

Background: formulates the problem (e.g. describes MDP, whether it addresses partial observability, reward scheme, etc.)

Algorithm/Method: proposes new way to approach said problem; provides new loss function, or training algorithm; often includes description of task

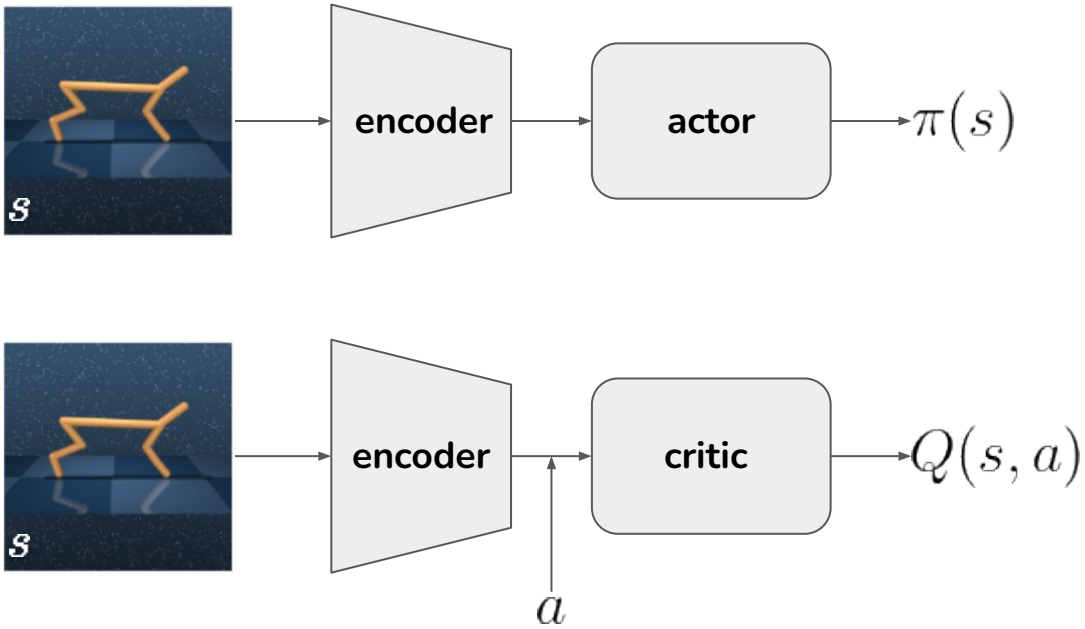
Results: compares proposed method to state of the art baselines; highlights nuances/limitations of method; performs ablations

Conclusion: summarize paper, provides suggestions for next steps

Appendix: includes details needed to replicate results or lengthy proofs; anything that would interrupt the flow of the paper should go here

DDPG for xArm Reacher

Extending to Image Observations...



- Common to share encoder weights (only send critic loss to encoder)
- Use techniques to pretrain encoder

State of the Art for Continuous Control RL

Soft Actor Critic:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi(\cdot | \mathbf{s}_t)) \right]$$

For PyTorch implementations: see stable-baselines3

Soft target update in PyTorch

```
def soft_update_params(net: nn.Module, target_net: nn.Module, tau: float):
    for param, target_param in zip(net.parameters(), target_net.parameters()):
        target_param.data.copy_(
            tau * param.data + (1 - tau) * target_param.data
        )
```

Next Class (3.4.22)

- General guidance on how to approach a deep learning problem
- Discussion about Project Proposal Feedback

Survey (3.1.22)

https://forms.gle/9JLYeDdrbX8yJoP76