University of Stuttgart

Deep Reinforcement Learning for Control

Daniel Hennes

24.11.2017

University Stuttgart - IPVS - Machine Learning & Robotics

Deep (*supervised*) learning

- **Deep representation** is a composition of *many* function

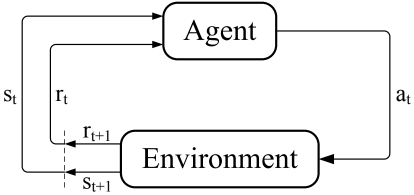
$$x \xrightarrow[w_1]{} h_1 \xrightarrow[w_2]{} h_2 \xrightarrow[w_3]{} \ldots \xrightarrow[w_n]{} h_n \xrightarrow[w_{n+1}]{} y$$

- Linear transformation and non-linear **activation functions** h_k
- Weight sharing
  - **Recurrent neural networks**: across time steps
  - **Convolutional neural networks**: across spatial (or temporal) regions
- Stochastic gradient descent (SGD)
  - **loss-function**, e.g., $l(y) = ||y^* - y||^2$
  - objective is to minimize expected *loss*: $\mathcal{L} = \mathbb{E}_x\left[l(y)\right]$
  - adjust weights in direction of gradient: $\Delta w_i = -\alpha \frac{\delta l(y)}{\delta w_i}$
- Powerful **function approximation** and **representation learning**
  - finds compact low-dimensional representation (*features*)
- State-of-the-art for image, text and audio

Reinforcement learning

- General purpose framework for artificial intelligence
- Autonomous agent that *interacts* with its environments

  Learning through interaction
- Learns *optimal* behaviors
- Improving over time through *trial & error*
- Scaling reinforcement learning requires powerful representations
  - domains with high-dimensional state (or observation) spaces
  - continuous action spaces



Many flavours of reinforcement learning

model-based $s \sim T,\ r \sim R \rightarrow \boxed{T_{s'}(s,a), R(s,a)} \rightarrow \boxed{V(s)} \rightarrow \pi(s)$

model-free

value-based $s \sim T,\ r \sim R \rightarrow \boxed{Q(s,a)} \rightarrow \pi(s)$

policy-based $s \sim T,\ r \sim R \rightarrow \boxed{\pi(s)}$

actor-critic $s \sim T,\ r \sim R \rightarrow \boxed{Q(s,a), \pi(s)}$

imitation learn. $\left\{(s_{1:t}, a_{1:t}, r_{1:t})^i\right\}_{i=1}^{n} \rightarrow \boxed{\pi(s)}$

inverse RL $\left\{(s_{1:t}, a_{1:t}, r_{1:t})^i\right\}_{i=1}^{n} \rightarrow \boxed{R(s,a)} \rightarrow \boxed{V(s)} \rightarrow \pi(s)$

learning | dynamic programming

Value-based reinforcement learning

- Bellman expectation equation:

$$
\begin{aligned}
Q^{\pi}(s,a) &= \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \ldots \mid s_t = a, a_t = a\right] \\
&= \mathbb{E}_{s_{t+1} \sim T, a_{t+1} \sim \pi}\left[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) \mid s_t = a, a_t = a\right]
\end{aligned}
$$

- Bellman optimality equation:

$$
Q^{*}(s,a) = \mathbb{E}_{s_{t+1}}\left[r_{t+1} + \gamma \max_{a_{t+1}} Q^{*}(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a\right]
$$

- Value iteration algorithm:

$$
Q_{i+1}(s,a) = \mathbb{E}_{s_{t+1}}\left[r_{t+1} + \gamma \max_{a_{t+1}} Q_{i}(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a\right]
$$

- Transition model $s_{t+1} \sim T(s_t, a_t)$ is unknown

Naive deep Q-learning

- Q-learning update rule:

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

- Q is represented by a (deep) neural network with weights w: $Q(s, a, w)$

- Loss is the mean-squared TD-error:

$$\mathcal{L}(w) = \mathbb{E}\left[\left(r_{t+1} + \gamma \max_a Q(s_{t+1}, a, w) - Q(s_t, a_t, w) \right)^2 \right]$$

- Minimize loss with SGD: $\frac{\delta l(w)}{\delta w}$

Stability

Naive Q-learning with neural networks oscillates or diverges:

1. Data is non i.i.d!
   - trajectories
   - samples are correlated (generated by interaction)
2. Policy changes rapidly with slight changes to Q-values
   - policy may oscillate
3. Reward range is unknown
   - gradients can be large
   - instabilities during back-propagation

Deep Q-networks (DQN)

Deep Q-networks (DQN) address instabilities through:

- **Experience replay**
  - store transitions $\langle s_t, a_t, r_{t+1}, s_{t+1} \rangle$
  - sample random mini-batches
  - removes correlation, restores i.i.d. property
- **Target network**
  - second Q network
  - fixed parameters in target network
  - periodically update target network parameters
- **Reward clipping/normalization**
  - clip rewards to $r \in [0, 1]$
  - batch normalization

DQN in Atari

"End-to-end" learning:

- *state:* stack of 4 frames, raw pixels
- *action:* joystick commands (18 discrete actions)
- *reward:* change in score

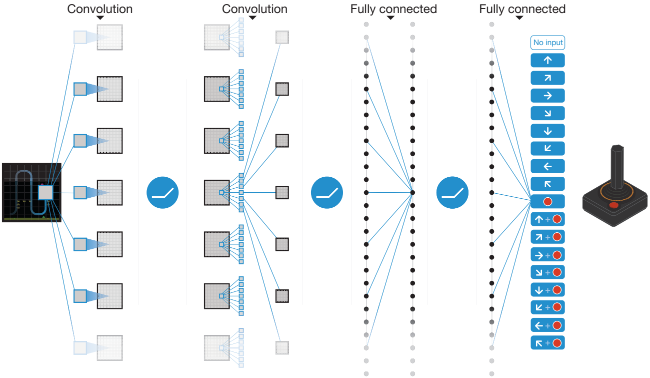


image from *Human-level control through deep reinforcement learning* (Google Deepmind / Nature)

DQN in Atari

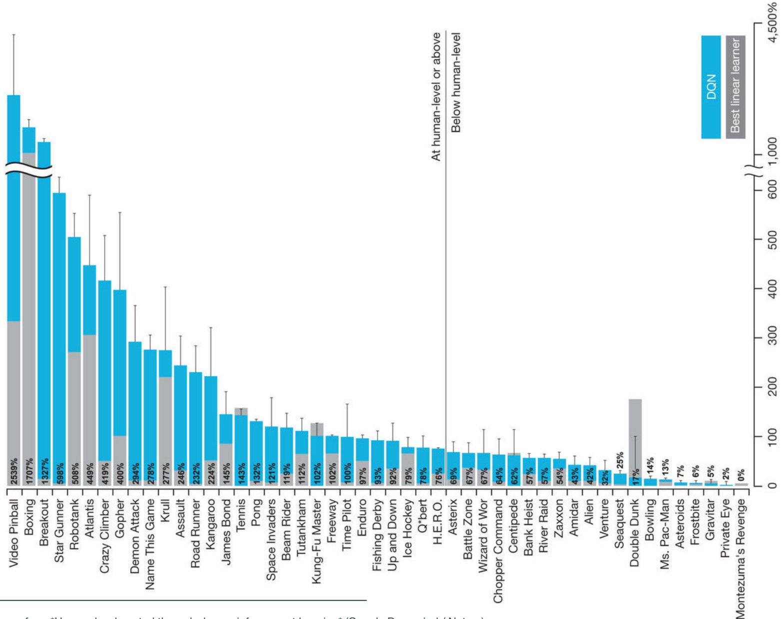


image from *Human-level control through deep reinforcement learning* (Google Deepmind / Nature)

Policy gradient for continuous actions

- Value-based with continuous actions: $\arg\max_a Q(\cdot)$ is an optimization problem in itself
- Represent policy directly by a deep network: $\pi(s, \theta)$
- Objective:
  - discounted reward: $J(\theta) = \mathbb{E}\left[r_{t_0} + \gamma r_{t_1} + \gamma^2 r_{t_2} \ldots\right]$
  - episodic reward: $J(\theta) = \mathbb{E}\left[\sum_{t=t_0}^{T} r_t\right]$
- Optimize with SGD

Policy gradient for continuous actions

- Value-based with continuous actions: $\arg\max_a Q(\cdot)$ is an optimization problem in itself
- Represent policy directly by a deep network: $\pi(s, \theta)$
- Objective:
  - discounted reward: $J(\theta) = \mathbb{E}\left[r_{t_0} + \gamma r_{t_1} + \gamma^2 r_{t_2} \ldots\right]$
  - episodic reward: $J(\theta) = \mathbb{E}\left[\sum_{t=t_0}^{T} r_t\right]$
- Optimize with SGD
- Problems:
  - relies on empirical return of a trajectory $\rightarrow$ high variance
  - introducing unbiased estimates $\rightarrow$ reduce variance
  - *substracting a baseline* (e.g., average over several MC rollouts)
  - weighting updates by an **advantage** instead of pure reward

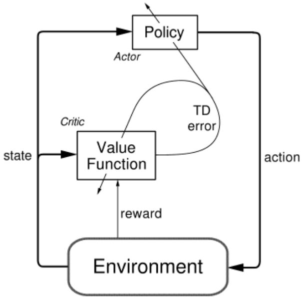
Actor-critic

- The gradient of the policy is the direction that *most improves Q*:

$$\frac{\delta J(\theta)}{\delta\theta} = \mathbb{E}_{\mathcal{S}}\left[\frac{\delta Q^{\pi}(s,a,w)}{\delta a}\frac{\delta\pi(s,\theta)}{\delta\theta}\right]$$

- Actor-critic methods use the value function as a baseline for policy gradients

- Trade off between *variance reduction* of policy gradients with *bias introduction* from value function methods



Deep deterministic policy gradient (DDPG)

Deterministic policy gradient uses the $critic$ as the loss function for $actor$

DDGP also addresses instabilities by:

- **Experience replay** for *actor* and *critic*
- **Target network** to *freeze* parameters for Q, periodically updated

"End-to-end" learning with continuous actions

- *state:* stack of 4 frames, raw pixels
- *action:* continuous (up to 12-dimensional)
- *reward:* various objectives

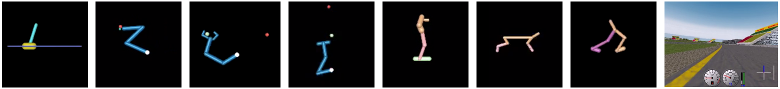


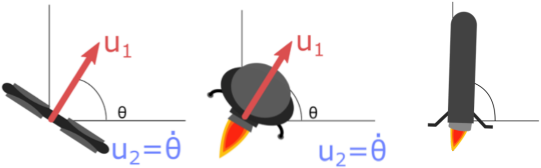
image from *Continuous control with deep reinforcement learning* (Google Deepmind / ICRL)

Asynchronous advantage actor critic (A3C)

- Speedup through parallel computing
- Parameters are read/updated asynchronously by multiple *agents*
- Agents are situated in *independent* environments
    - stabalizes gradients
    - allows for more exploration

Learning to imitate optimal control

- Pre-compute (off-line) many optimal trajectories
- Train a deep artificial neural architecture to learn the optimal policy $\pi^*(s)$ (or $u^*(x)$)
- Use the learned policy to drive the system in *real-time*



Training

- Millions of optimal state-action pairs

  100 pairs per optimal trajectory

- Optimal control profiles

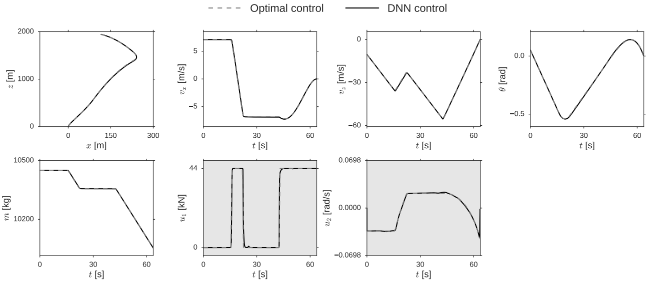
  - Continuous control (quadratic control)
  - Bang-off-bang control, saturated control (time or mass optimal)

- Stochastic gradient descent with mini-batches

- Tested diverse architectures (depth & non-linearities)

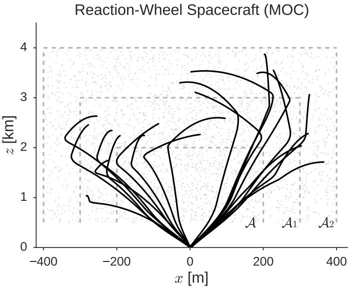
Evaluation

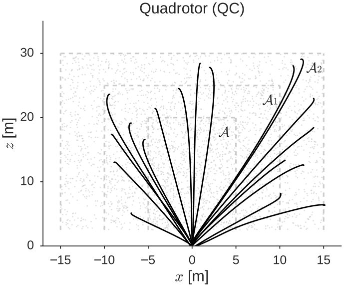
	Success rate	Distance to target r [m]	v [m/s]	Optimality
Multicopter [QC]	100.0%	0.014	0.027	98.18%
Multicopter [TOC]	100.0%	0.016	0.028	98.88%
Spacecraft [QC]	100.0%	0.29	0.044	99.60%
Spacecraft [MOC]	98.3%	2.90	0.073	99.28%
Rocket [QC]	99.0%	1.10	0.066	99.62%
Rocket [MOC]	95.0%	1.95	0.094	99.67%

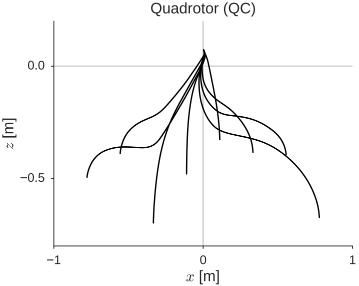


NOTE: unsuccessful landings are not catastrophic, they only miss the pinpoint by more than a strictly defined tolerance.

Powerful generalization: initial state

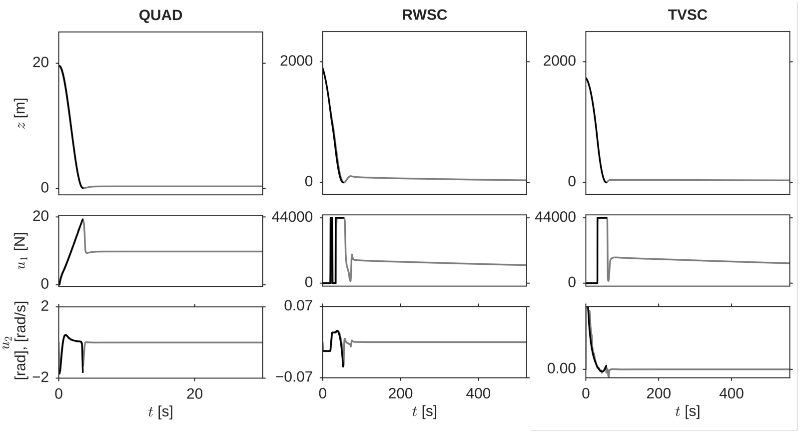




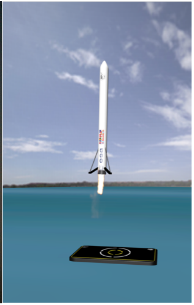


Powerful generalization: after episode termination

- In MOC the target position is always reached with either maximum or minimum thrust
- Hovering: $u_1 = mg, u_2 = 0$
- **Never seen in training!**



Visual landing with CNNs



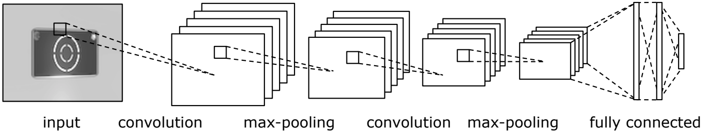
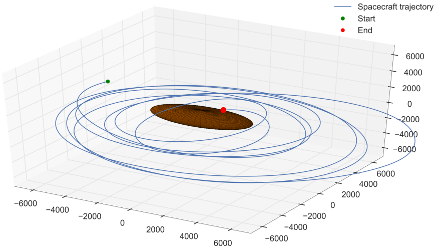


image credit (top left): Space X

Learning to hover near small bodies

- Highly dynamic environments
- Elementary motion detectors
  - Provide optical flow
- Reinforcement learning
  - Evolutionary policy search
  - LSPI
- Future directions
  - Deep reinforcement learning
  - “end-to-end” learning



Discussion

- *Powerful representations for powerful reinforcement learning*
- Learning by *trail and error* from **interaction**
  - *advantage:* general-purpose
  - *disadvantage:* sample efficiency
- Sample efficiency can be addressed by *imitation learning*
  - e.g. with trajectories generated by optimal control methods
  - supervised-learning in combination with importance sampling (correcting for off-policy samples)
  - Guided policy search (GPS)
  - Trust region policy optimization (TRPO)
- Deep representations allow for powerful reinforcement learning
- *Interpretability* remains a big challenge in deep (reinforcement) learning

Thank you!

Daniel Hennes | `daniel.hennes@gmail.com`

Collaborators:

Dario Izzo (ESA | ACT)

Carlos Sanchez-Sanchez

Stefan Willi