CENG 783
Special topics in Deep Learning

Week 14
Deep Reinforcement Learning
Sinan Kalkan
now

- Reinforcement Learning

- Deep Reinforcement Learning
  - Value Networks
  - Policy Networks
Reinforcement Learning

The agent receives reward $r_t$ for its actions.
More formally

• An agent’s behavior is defined by a policy, $\pi$:
  $$\pi: \mathcal{S} \to \mathcal{P}(\mathcal{A})$$
  $\mathcal{S}$: The space of states.
  $\mathcal{A}$: The space of actions.

• The “return” from a state is usually:
  $$R_t = \sum_{i=t}^{T} \gamma^{(i-t)}r(s_i, a_i)$$
  $r(s_i, a_i)$: the reward for action $a_i$ in state $s_i$.
  $\gamma$: discount factor.

• Goal: Learn a policy that maximizes the expected return from the starting position:
  $$\mathbb{E}_{r_i, s_i \sim E, a_i \sim \pi}[R_1]$$

http://www.cs.ubc.ca/~murphyk/Bayes/pomdp.html
More formally

- We can define an expected return for taking action $a_t$ at state $s_t$:
  \[
  Q^\pi(s_t, a_t) = \mathbb{E}_{r_{i \geq t}, s_{i > t} \sim E, a_{i > t} \sim \pi} [R_t | s_t, a_t]
  \]

- This can be rewritten as (called the Bellman equation):
  \[
  Q^\pi(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E} \left[ r(s_t, a_t) + \gamma \mathbb{E}_{a_{t+1} \sim \pi} [Q^\pi(s_{t+1}, a_{t+1})] \right]
  \]
Reinforcement Learning in/with Deep Networks

- Two general approaches:
  - Value gradients
  - Policy gradients
Figure 1 | Schematic illustration of the convolutional neural network. The details of the architecture are explained in the Methods. The input to the neural network consists of an $84 \times 84 \times 4$ image produced by the preprocessing map $\phi$, followed by three convolutional layers (note: snaking blue line symbolizes sliding of each filter across input image) and two fully connected layers with a single output for each valid action. Each hidden layer is followed by a rectifier nonlinearity (that is, $\max(0,x)$).

LETTER

Human-level control through deep reinforcement learning

Volodymyr Mnih*, Koray Kavukcuoglu*, David Silver†, Andrei A. Rusu†, Joel Veness†, Marc G. Bellemare†, Alex Graves†, Martin Riedmiller†, Andreas K. Fidjeland†, Georg Ostrovski†, Stig Petersen†, Charles Beattie†, Amir Sadik†, Ioannis Antonoglou†, Helen King†, Dharshan Kumaran†, Daan Wierstra†, Shane Legg† & Demis Hassabis†
network. We refer to a neural network function approximator with weights \( \theta \) as a Q-network. A Q-network can be trained by adjusting the parameters \( \theta_i \) at iteration \( i \) to reduce the mean-squared error in the Bellman equation, where the optimal target values \( r + \gamma \max_{a'} Q^*(s',a') \) are substituted with approximate target values \( y = r + \gamma \max_{a'} Q(s',a'; \theta_i^-) \), using parameters \( \theta_i^- \) from some previous iteration. This leads to a sequence of loss functions \( L_i(\theta_i) \) that changes at each iteration \( i \),

\[
L_i(\theta_i) = \mathbb{E}_{s,a,r} \left[ (\mathbb{E}_s[y|s,a] - Q(s,a; \theta_i))^2 \right]
\]

**LETTER**

**Human-level control through deep reinforcement learning**

Volodymyr Mnih\(^1\)*, Koray Kavukcuoglu\(^1\)*, David Silver\(^1\)*, Andrei A. Rusu\(^1\), Joel Veness\(^1\), Marc G. Bellemare\(^1\), Alex Graves\(^1\), Martin Riedmiller\(^1\), Andreas K. Fidjeland\(^1\), Georg Ostrovski\(^1\), Stig Petersen\(^1\), Charles Beattie\(^1\), Amir Sadik\(^1\), Ioannis Antonoglou\(^1\), Helen King\(^1\), Dharshan Kumaran\(^1\), Daan Wierstra\(^1\), Shane Legg\(^1\) & Demis Hassabis\(^1\)
Policy gradients

http://karpathy.github.io/2016/05/31/rl/
Policy gradients

http://karpathy.github.io/2016/05/31/rl/