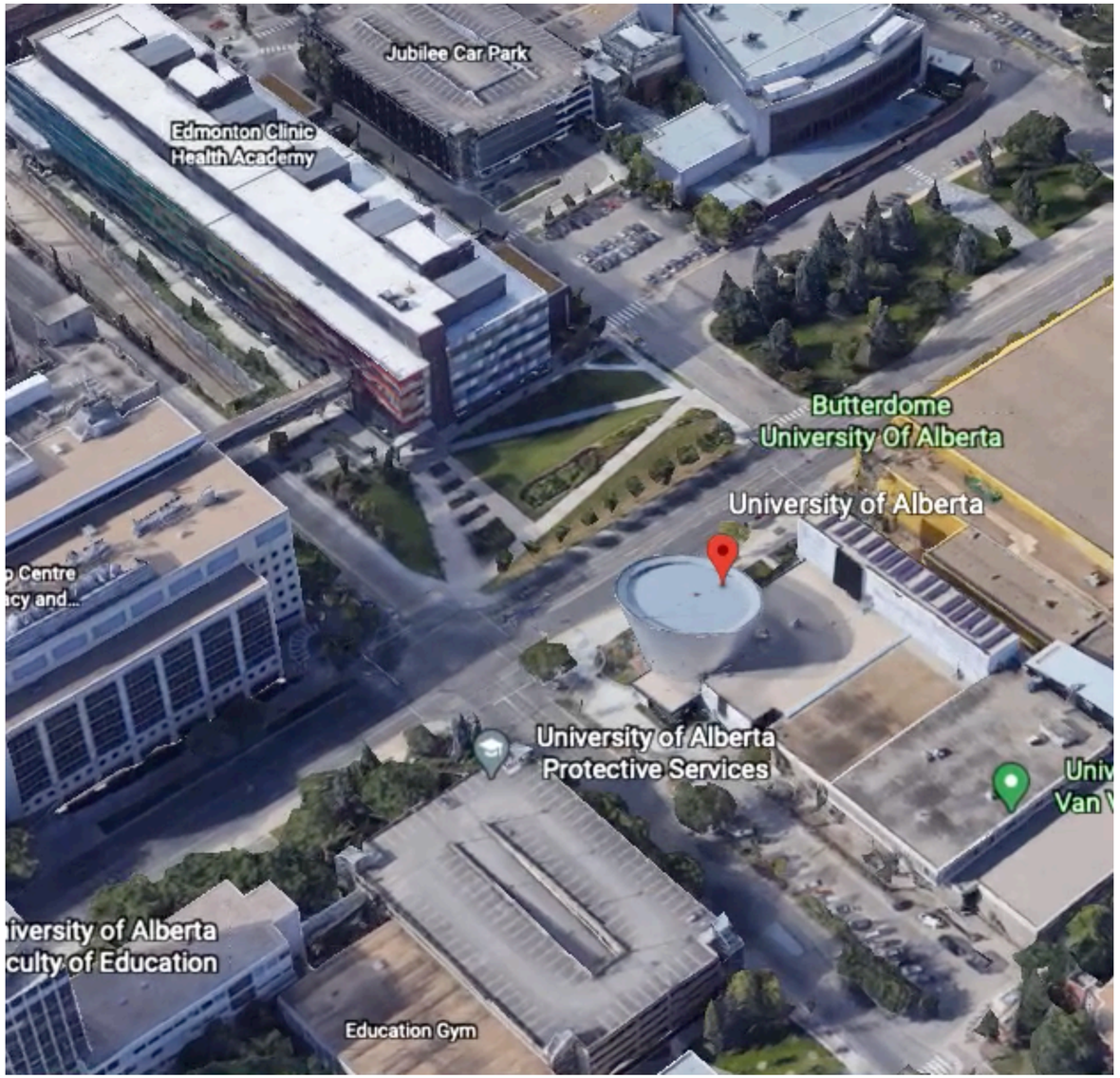
Reinforcement Learning

An Introduction

second edition

Richard S. Sutton and Andrew G. Barto



Smoothness in RL with Large State and Action Spaces

Kavosh Asadi

Degree of Doctor of Philosophy
Brown University

amazon web services

- fundamental research on RL
- RL applications to AWS

Lecture 33:

Advanced Reinforcement Learning

Kavosh Asadi, April 21st 2021

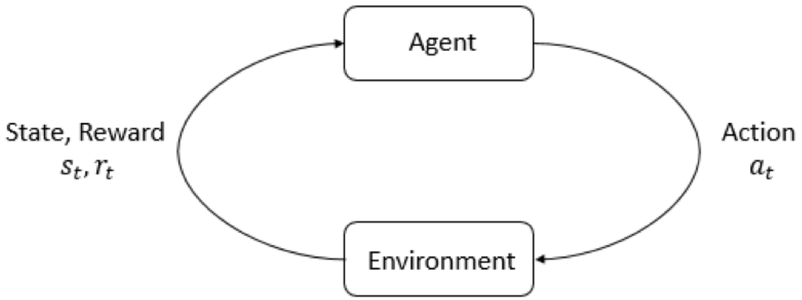
Mnih et. al, 2015, “Human-level Control Through Deep Reinforcement Learning”, Nature

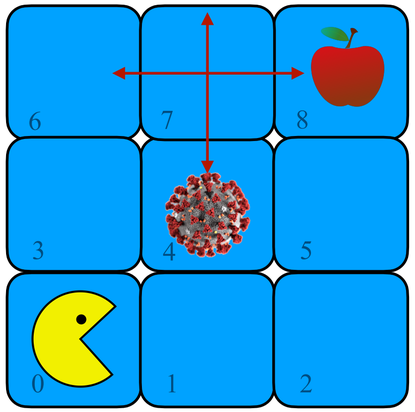
no training

~2 hours of training

~4 hours of training

what is RL?





$$\langle \mathcal{S}, \mathcal{A}, R, T, \gamma \rangle$$

$$\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}r(\mathcal{S})$$

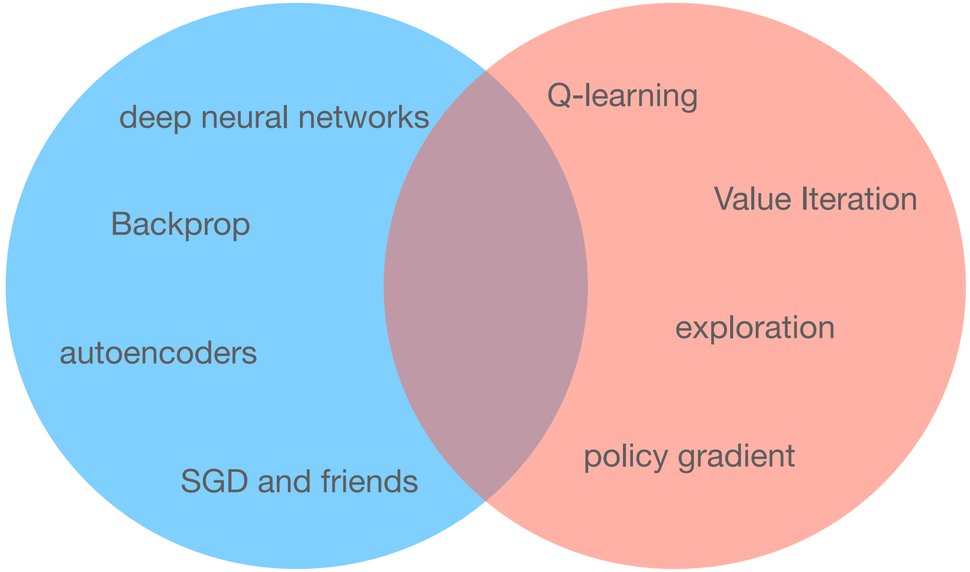
$$\mathcal{R}(s' = 7, |s = down, up) = 0.8$$

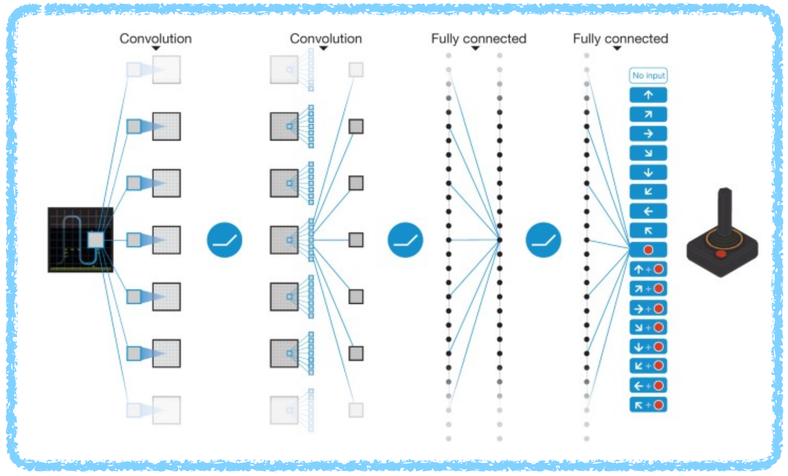
$$\mathcal{R}(s' = 6, |s = right, up) = 0.1$$

$$T(s' = 8 | s = 7, a = up) = 0.1$$

what is deep RL?

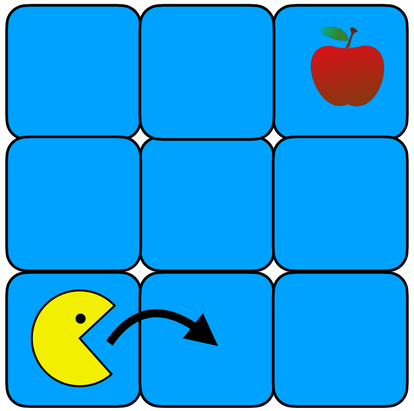
- integration of RL algorithms with deep networks
- applied to large-scale problems

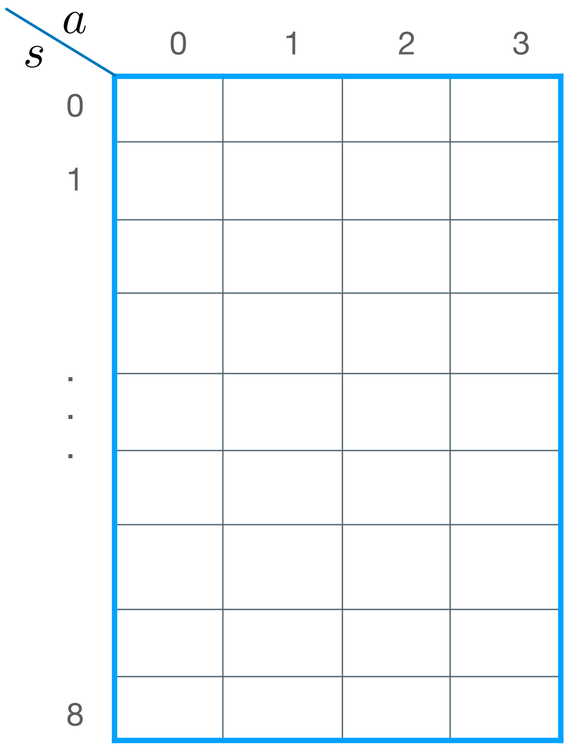




trained using Q-learning

Mnih et. al, 2015, “Human-level Control Through Deep Reinforcement Learning”, Nature





what is Q-learning?

estimate the long-term goodness of a state-action pair

bootstrapping — update your estimate based on another estimate

given $\langle s, a, r, s' \rangle$

$$Q(s,a) \leftarrow \underbrace{(1-\alpha)Q(s,a)}_{\text{current guess}} + \alpha\underbrace{\left(r + \gamma \max_{a'} Q(s',a')\right)}_{\text{another guess}}$$

$$= Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

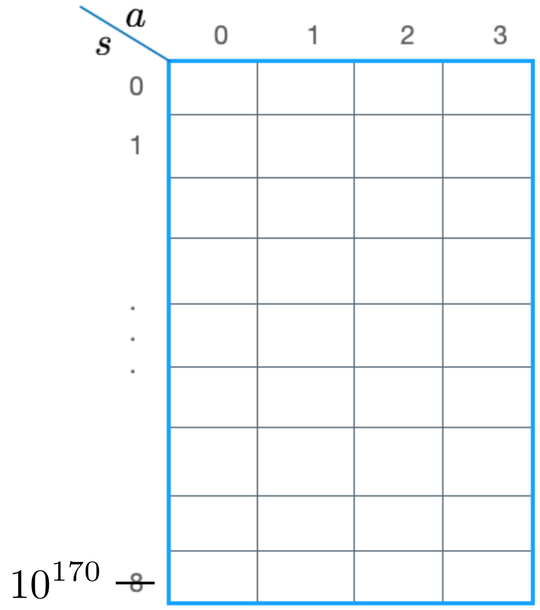
chicken an egg, what if the other estimate is wrong?

TD error

this is called “tabular” Q-learning

Silver et. al, 2016, “Mastering the game of Go with deep neural networks and tree search”, Nature

$|\mathcal{S}| = \sim 10^{170} \gg$ # of atoms in the observable universe, $\sim 10^{80}$

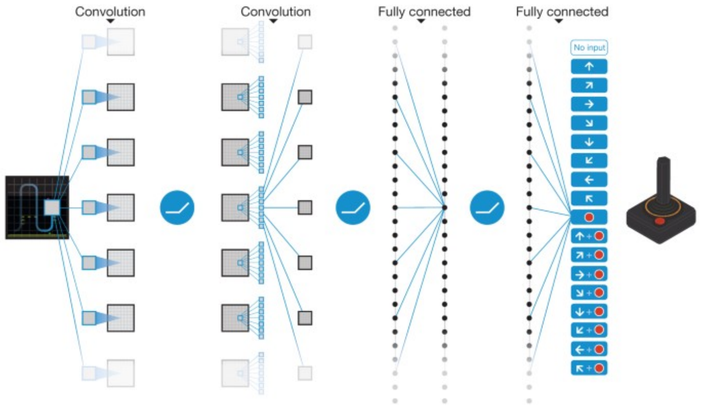


Q-learning with Neural Networks

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\Big)$$

$$\theta \leftarrow \theta + \alpha\Big(r + \gamma \max_{a'} Q(s',a';\theta) - Q(s,a;\theta)\Big)\nabla_\theta Q(s,a;\theta)$$

in practice we use advanced SGD (Adam, RMSProp etc)



$Q(s,a;\theta)$ parameterized by θ

action selection:

our Q values are inaccurate
so don't starve any action

ϵ-greedy: select the maximizing action with some probability, otherwise select randomly

$$\text{if } \text{random}() > \epsilon \quad \text{take} \quad \arg\max_a Q(s,a)$$

else choose randomly

non-maximizing actions have same probability

Boltzmann softmax:

$$\max_{p(a)} \sum_a p(a)Q(s,a) - \beta \sum_a p(a) \log\big(p(a)\big)$$

$$p(a) = \frac{e^{\beta Q(s,a)}}{\sum_a e^{\beta Q(s,a)}}$$

target network:

$$\theta \leftarrow \theta + \alpha\big(\underbrace{r + \gamma \max_{a'} Q(s', a'; \theta)}_{\text{target}} - Q(s, a; \theta)\big)\nabla_\theta Q(s, a; \theta)$$

notice that, unlike SGD, updating the parameters changes the target, this can cause instabilities during learning

simple hack, keep the target fixed for a while, synchronize once in a while

$$\theta \leftarrow \theta + \alpha\big(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\big)\nabla_\theta Q(s, a; \theta)$$

after a few updates $\theta^- \leftarrow \theta$

Congrats:

you now understand how to implement DQN !!!

don’t get too excited just yet! RL training can sometimes be hard

many issues exist:

- statistical problems
- optimization problems
- sample efficiency
- …

we will see two examples: the over-estimation problem, and optimization with continuous actions

don’t get too excited just yet! RL training can sometimes be hard

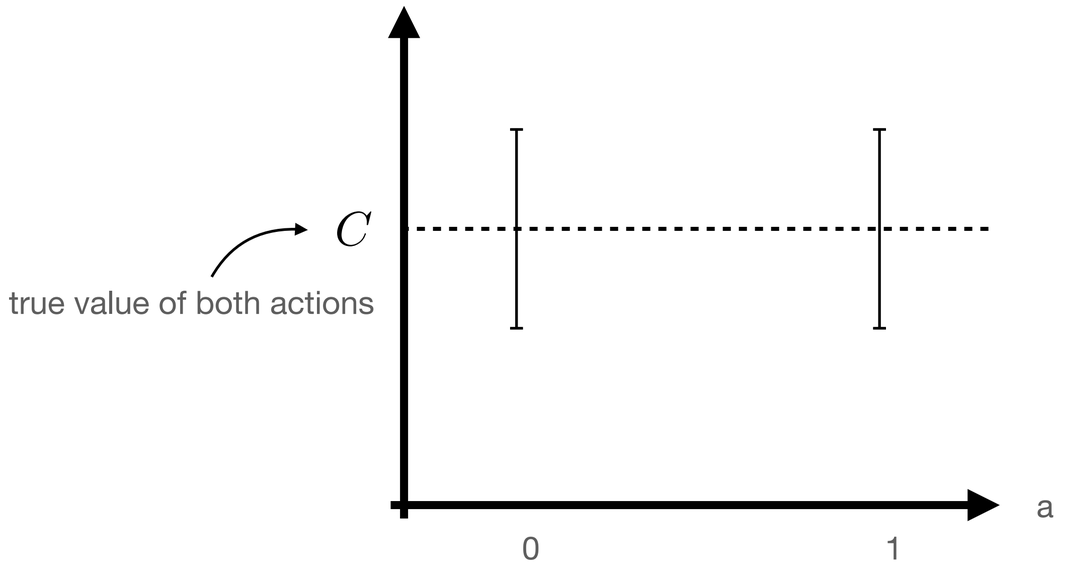
many issues exist:

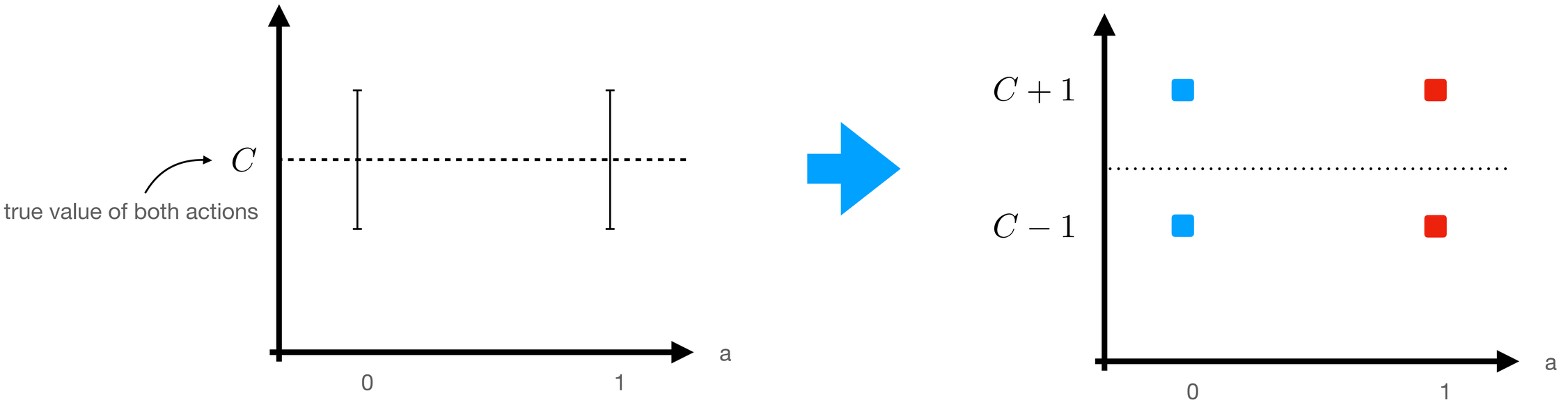
- statistical problems
- optimization problems
- sample efficiency
- …

we will see two examples: the **over-estimation problem**, and optimization with continuous actions

$$\theta \leftarrow \theta + \alpha\big(r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta)\big)\nabla_\theta Q(s, a; \theta)$$

$$\max_{a'} Q(s', a') = \{\max Q(s', 0), \max Q(s', 1)\}$$





pr 1/4 max(C+1,C+1) =C+1

pr 1/4 max(C+1,C-1) =C+1

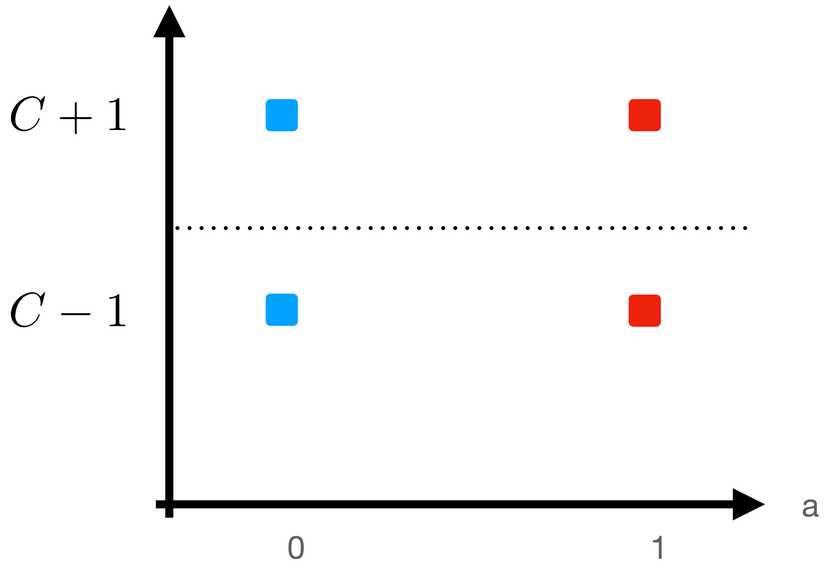
pr 1/4 max(C-1,C+1) =C+1

pr 1/4 max(C-1,C-1) =C-1

$$\mathbf{E}[\max Q(s', 0), \max Q(s', 1)] = C + \frac{1}{2}$$

we want $\max\{E[Q(s,0)], E[Q(s,1)]\}$

but we took $E[\max\{Q(s,0), Q(s,1)\}]$



define $f(x) = \max(x)$

we have $E[f(x)] \geq f(E[x])$

turns out that the inequality holds for any function that is convex, maximum included. This is called Jensen's inequality

Maximum is Convex

a function f is convex, if and only if: $\forall \alpha \in [0,1] \quad f\big(\alpha x_1 + (1-\alpha)x_2\big) \leq \alpha f(x_1) + (1-\alpha) f(x_2)$

prove maximum is convex:

$$\begin{aligned} f\big(\alpha x + (1-\alpha)y\big) = \max\big(\alpha x + (1-\alpha)y\big) &= \alpha x_0 + (1-\alpha)y_0 \\ &\leq \alpha \max(x) + (1-\alpha)\max(y) \\ &= \alpha f(x) + (1-\alpha) f(y) \end{aligned}$$

$$\theta \leftarrow \theta + \alpha\big(r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta)\big)\nabla_\theta Q(s, a; \theta)$$

Q-learning indeed suffers from an overestimation bias
this is addressed by either keeping more than one Q function **[van Hasselt, 2009]**, and/or using softmax operators **[Asadi & Littman 2017]**

van Hasselt, 2009, “Double Q-learning”, in Advances in Neural Information Processing (Neurips)
Asadi, & Littman, 2017, “An Alternative Softmax Operator for Reinforcement Learning, In International Conference on Machine Learning (ICML)

don’t get too excited just yet! RL training can sometimes be hard

many issues exist:

- statistical problems
- optimization problems
- sample efficiency
- …

we will see two examples: the **over-estimation problem**, and optimization with continuous actions

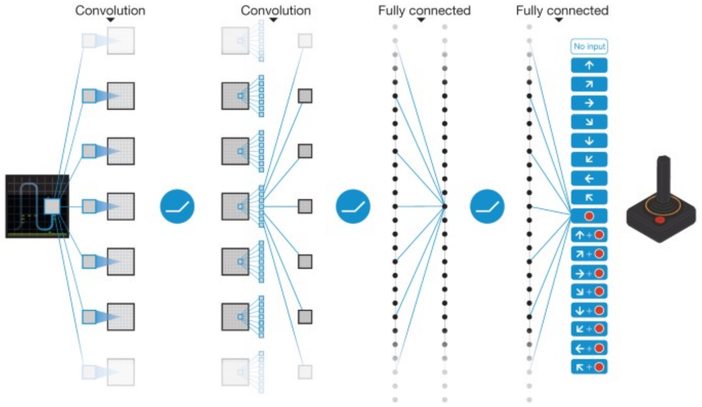
don’t get too excited just yet! RL training can sometimes be hard

many issues exist:

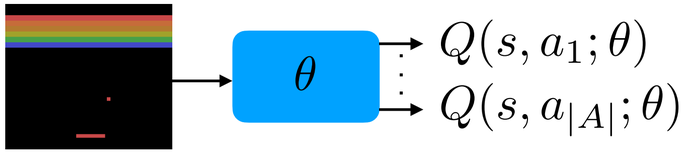
- statistical problems
- optimization problems
- sample efficiency
- …

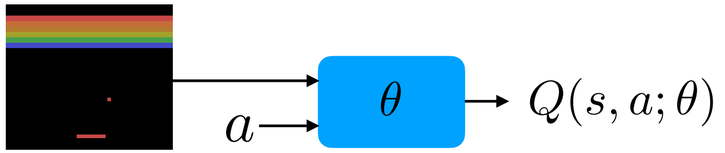
we will see two examples: the over-estimation problem, and **optimization with continuous actions**


Boston Dynamics



$Q(s, a; \theta)$ parameterized by θ





assume a perfect Q is learned

optimal action selection: $\arg\max_a Q(s, a; \theta)$

unclear how to do this operation

for learning Q itself:

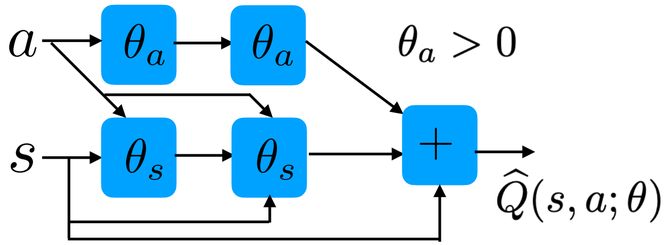
$$\theta \leftarrow \theta + \alpha\left(r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta)\right) \nabla_\theta Q(s, a; \theta)$$

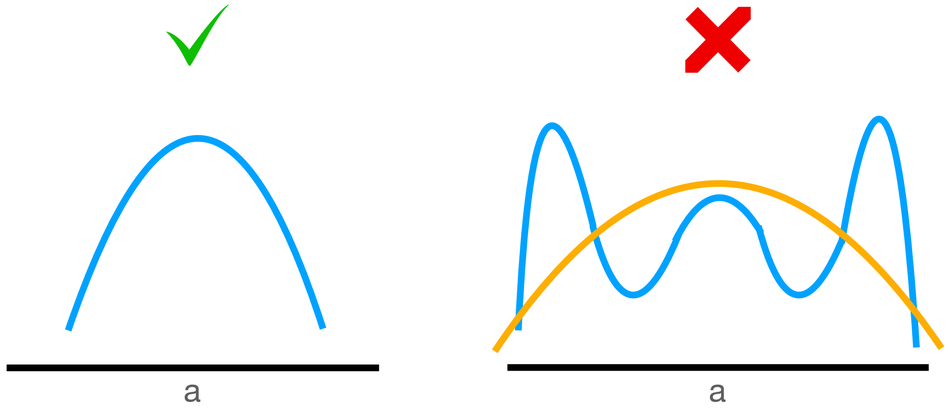
Input-Convex Neural Networks

[Amos et al., 17]

restrict the architecture (and weights) of the neural network so that it is always a convex (or concave) function of the action input

$$f(y) \leq f(x) + \nabla f(x)^\top (y - x)$$





true Q

our approximation of Q

fast action maximization &
universal function approximation

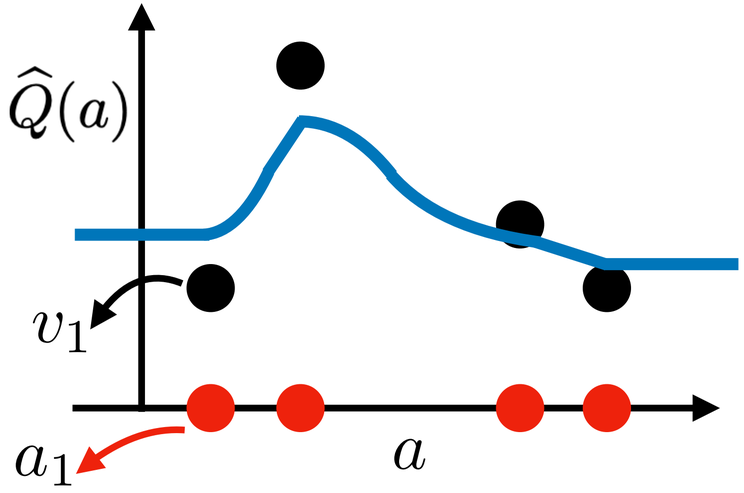
Amos et al., 2017, "Input Convex Neural Networks", In International Conference on Machine Learning (ICML)

Deep Radial-basis Value Functions (RBVFs)

[Asadi, Parikh, Parr, Konidaris, & Littman, 2021, “Deep Radial-basis Value Functions for Continuous Control”, In AAAI Conference on Artificial Intelligence (AAAI)

offers both desired properties at the same time:

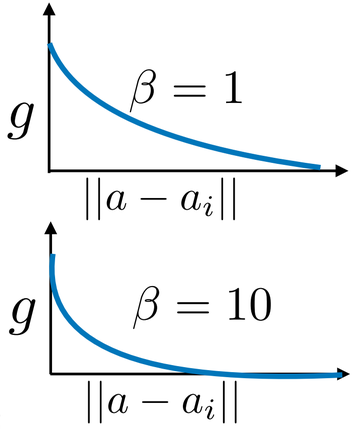
- fast maximization
- universal function approximation



$$\widehat{Q}(a) := \sum_{i=1}^{N} g(a - a_i) v_i$$

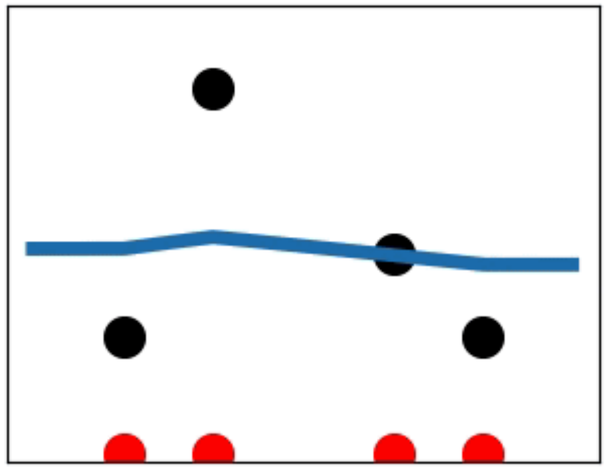
$$g(a - a_i) := e^{-\beta ||a - a_i||}$$

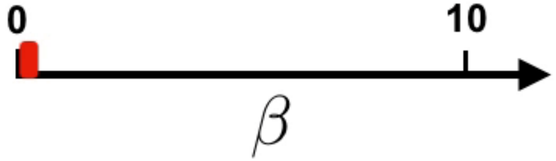
$$\widehat{Q}_\beta(a) := \sum_{i=1}^{N} e^{-\beta ||a - a_i||} v_i$$



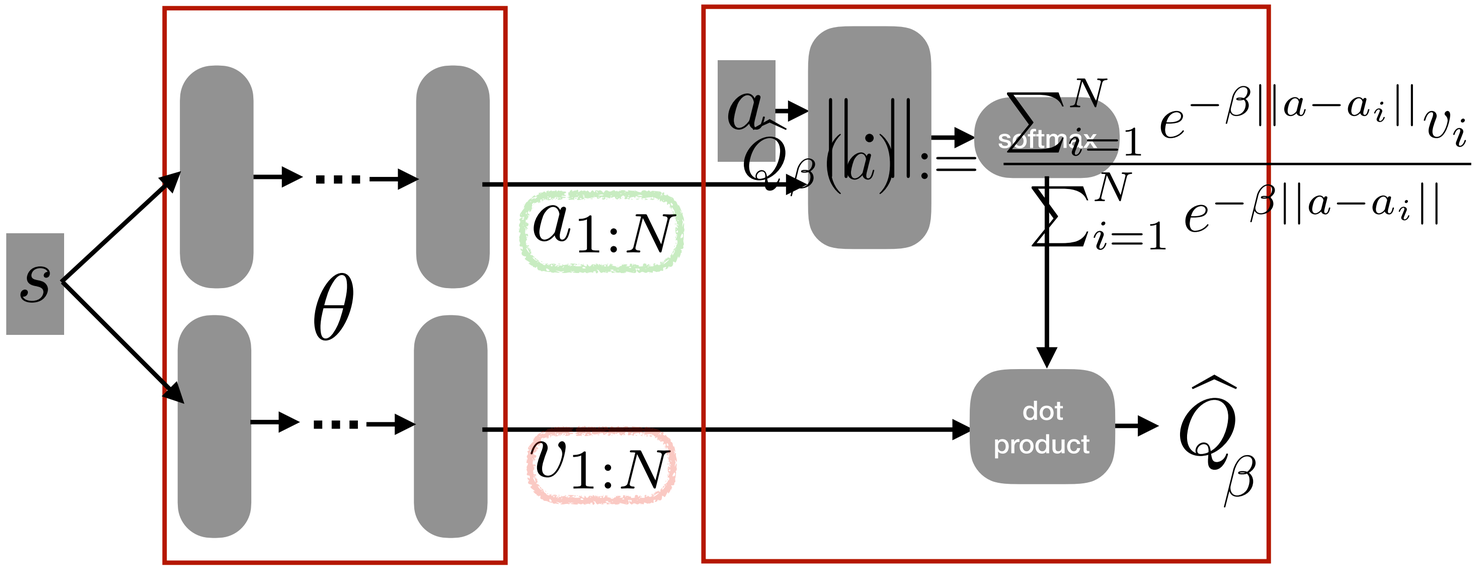
$$\widehat{Q}_\beta(a) := \frac{\sum_{i=1}^{N} e^{-\beta ||a - a_i||} v_i}{\sum_{i=1}^{N} e^{-\beta ||a - a_i||}}$$

"normalized"



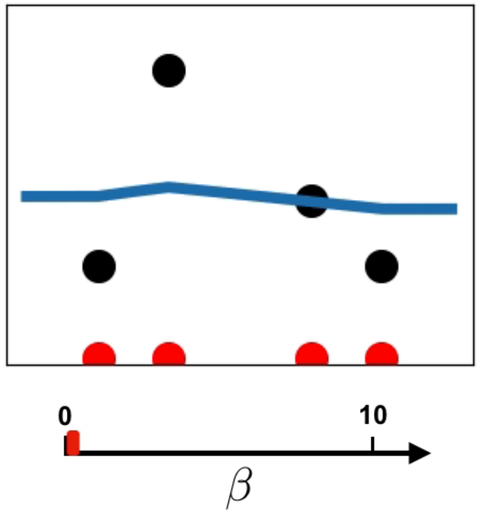


$$\widehat{Q}_\beta(s,a;\theta) := \frac{\sum_{i=1}^{N} e^{-\beta||a-a_i(s;\theta)||} v_i(s;\theta)}{\sum_{i=1}^{N} e^{-\beta||a-a_i(s;\theta)||}}$$



$$\mathcal{A} \in \mathbb{R}$$

$$\max_{a \in \mathcal{A}} \widehat{Q}_\beta(s, a; \theta) = \max_{i \in [1, N]} \widehat{Q}_\beta(s, a_i; \theta)$$



$$\mathcal{A} \in \mathbb{R}^d$$

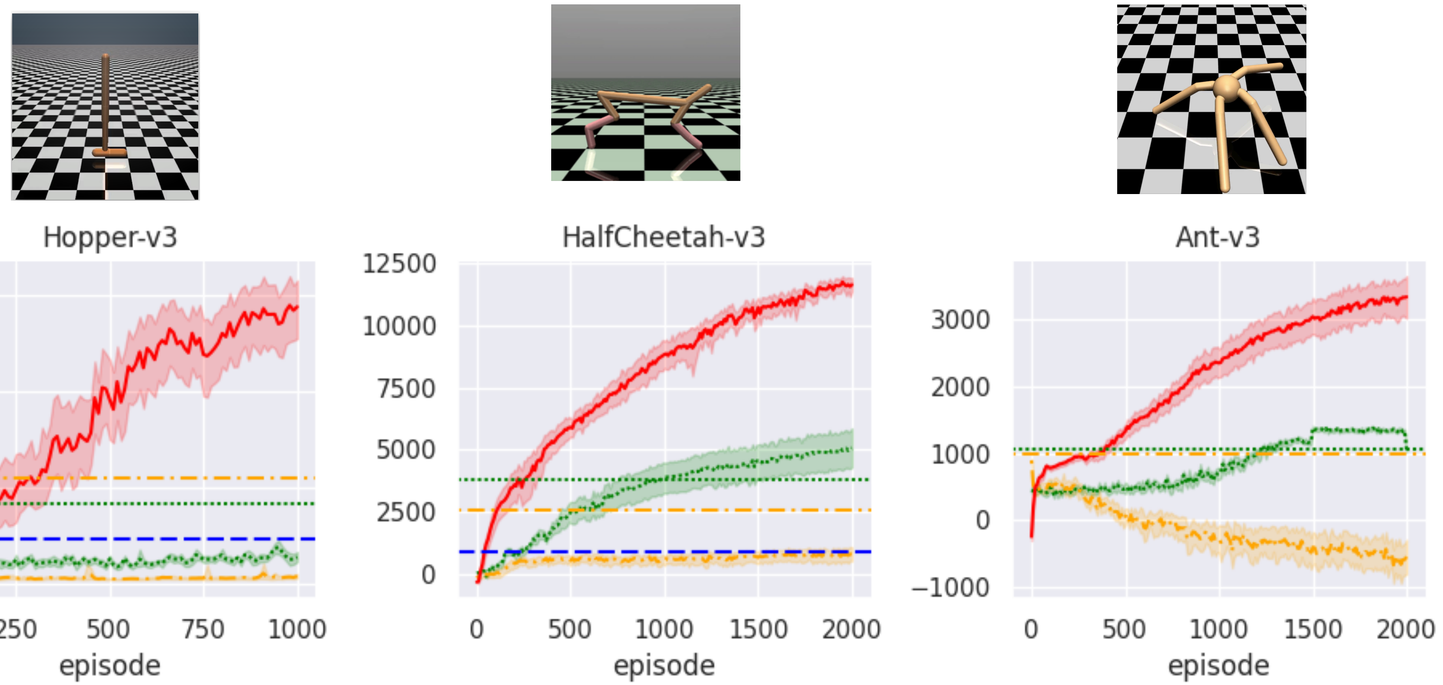
$$0 \leq \max_{a \in \mathcal{A}} \widehat{Q}_\beta(s, a; \theta) - \max_{i \in [1, N]} \widehat{Q}_\beta(s, a_i; \theta) \leq \mathcal{O}(e^{-\beta})$$

does not impede universal function approximation

$$\theta \leftarrow \theta + \alpha\big(r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta)\big)\nabla_\theta Q(s, a; \theta)$$

RBF-DQN: use a deep RBVF, and the following approximation:

$$\max_{a \in \mathcal{A}} \widehat{Q}_\beta(s, a; \theta) \approx \max_{i \in [1, N]} \widehat{Q}_\beta(s, a_i; \theta)$$



thank you!

Kavosh Asadi

email: kavasadi@amazon.com