

Reinforcement Learning



































7. Deep Q Network

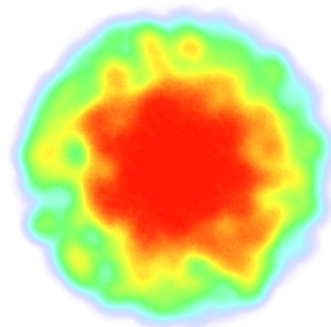
Olivier Sigaud

Sorbonne Université
<http://people.isir.upmc.fr/sigaud>



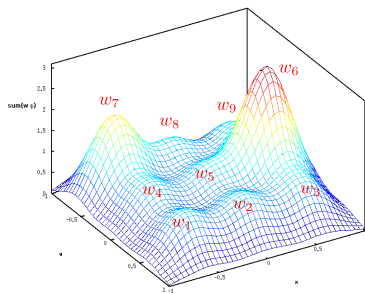
Parametrized representations

 0.0 	 0.0 	 0.0 	 0.0 	 0.0 
 0.0 	 0.0 	 0.0 		 0.0 
 0.0 		 0.0 		 0.0 
 0.0 	 0.0 	 0.0 		0.0



- ▶ If the state or action spaces become continuous, we need to represent a function over a continuous domain
- ▶ We cannot enumerate all values

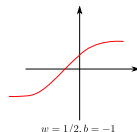
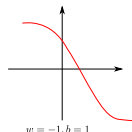
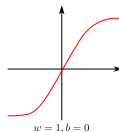
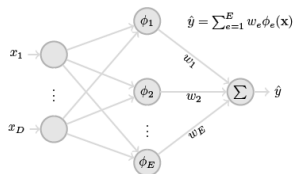
Linear representations



1 weight for each gaussian.

- ▶ To represent a continuous function, use features and a vector of parameters
- ▶ Learning tunes the weights
- ▶ Linear architecture: linear combination of features

The case of (feedforward) neural networks



- ▶ Last layer: linear combination of features (as in linear architectures)
- ▶ Sigmoids instead of Gaussians: better split of space in high dimensions
- ▶ Weight of input layer(s): tuning basis functions
- ▶ Weight of output layer: regression
- ▶ The backprop algo tunes both output and hidden weights
- ▶ Discovers adequate features by itself in a large space

General motivations for Deep RL

- ▶ Approximation with deep networks provided enough computational power can be very accurate
- ▶ All processes rely on efficient backpropagation in deep networks
- ▶ Available in CPU/GPU libraries: theano, caffe, ..., TensorFlow, pytorch



Kingma, D. P. & Ba, J. (2014) Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*

Early success



- ▶ The first world champion was using RL with neural networks
- ▶ But it was shown that RL with function approximation can diverge



Tesauro, G. (1995) Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68



Baird, L. C. (1994) Reinforcement learning in continuous time: Advantage updating. *Proceedings of the International Conference on Neural Networks*, Orlando, FL

DQN: the breakthrough

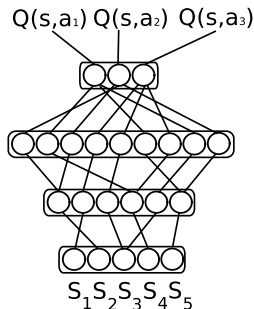


- ▶ DQN: Atari domain, Nature paper, small discrete actions set
- ▶ Learned very different representations with the same tuning



Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015) Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.

The Q-network in DQN



- ▶ Parametrized representation of the critic $Q(s_t, a_t | \theta)$
- ▶ The Q-network is the equivalent of the Q-Table (with an infinity of state rows)
- ▶ Select action by finding the max (as in Q-LEARNING)
- ▶ Limitation: requires one output neuron per action

Learning the neural Q-function

- Supervised learning: minimize a loss-function, often the squared error w.r.t. the output:

$$L(s, a) = (\overset{\text{desired output}}{y^*(s, a)} - \overset{\text{output}}{Q(s, a|\theta)})^2 \quad (1)$$

with backprop on weights θ

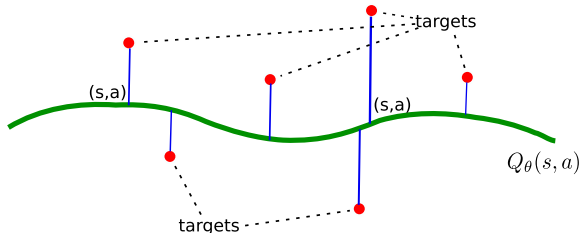
- For each sample i , the Q-network should minimize the RPE:

we want to minimize delta => delta -> 0

$$\delta_i = r_i + \gamma \max_a Q(s_{i+1}, a|\theta) - Q(s_i, a_i|\theta)$$

- Thus, given a minibatch of N samples $\{s_i, a_i, r_i, s_{i+1}\}$, compute $y_i = r_i + \gamma \max_a Q(s_{i+1}, a|\theta')$

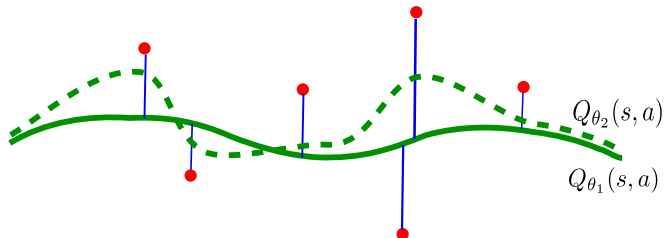
Learning the neural Q-function



- In the tabular case, each Q-value is updated separately
- In the continuous function approximation setting, interdependencies
- Thus update θ by minimizing the (squared error) loss function

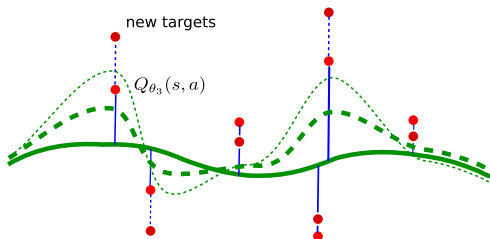
$$L = 1/N \sum_i (y_i - Q(s_i, a_i | \theta))^2$$

Learning the neural Q-function



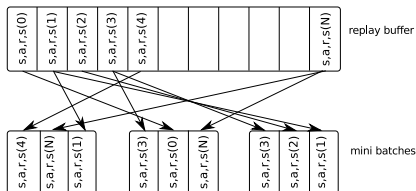
- ▶ The neural network weights are updated so as to decrease all errors on average
- ▶ Using many mini-batches, one gets global minimization over the whole Q-function

Trick 1: Stable Target Q-function



- ▶ The target $y_i = r_i + \gamma \max_a Q(s_{i+1}, a) | \theta$ is itself a function of Q
- ▶ Thus this is not truly supervised learning, and this is unstable
- ▶ Key idea: “periods of supervised learning”
- ▶ Compute the loss function from a separate *target network* $Q'(\dots | \theta')$
- ▶ So rather compute $y_i = r_i + \gamma \max_a Q'(s_{i+1}, a | \theta')$
- ▶ θ' is updated to θ only each K iterations

Trick 2: Replay buffer shuffling



- ▶ In most learning algorithms, samples are assumed independently and identically distributed (iid)
- ▶ Obviously, this is not the case of behavioral samples (s_i, a_i, r_i, s_{i+1})
- ▶ Idea: put the samples into a buffer, and extract them randomly
- ▶ Use training minibatches (make profit of GPU when the input is images)
- ▶ The replay buffer management policy is an issue



Lin, L.-J. (1992) Self-Improving Reactive Agents based on Reinforcement Learning, Planning and Teaching. *Machine Learning*, 8(3/4), 293–321



de Bruin, T., Kober, J., Tuyls, K., & Babuška, R. (2015) The importance of experience replay database composition in deep reinforcement learning. In *Deep RL workshop at NIPS 2015*



Zhang, S. & Sutton, R. S. (2017) A deeper look at experience replay. *arXiv preprint arXiv:1712.01275*

Double-DQN

- ▶ The max operator in Q-LEARNING results in the maximization bias
- ▶ Double Q-LEARNING: use Q_1 and Q_2 functions (see Class. 6)
- ▶ Double-DQN: make profit of the target network: propagate on target Q-network, select max on Q-network,
- ▶ Minor change with respect to DQN
- ▶ Converges twice faster



Van Hasselt, H., Guez, A., & Silver, D. (2015) Deep reinforcement learning with double q-learning. *CoRR*, *abs/1509.06461*

Prioritized Experience Replay

S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
$\delta = 0.8$	$\delta = 0.7$	$\delta = 0.5$	$\delta = 0.2$	$\delta = 0.1$	$\delta = 0.01$		

- ▶ Samples with a greater TD error improve the critic faster
- ▶ Give them a higher probability of being selected
- ▶ Favors the replay of new (s, a) pairs (largest TD error), as in $R - max$
- ▶ Several minor hacks, and interesting discussion
- ▶ Converges twice faster



Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015) Prioritized Experience Replay. *arXiv preprint arXiv:1511.05952*

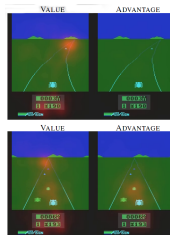
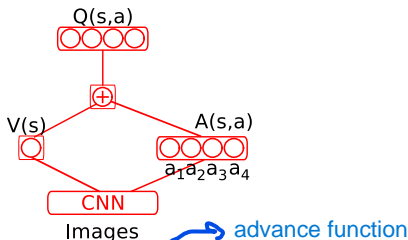
Advantage function

- ▶ $A(s_i, a_i|\theta) = Q(s_i, a_i|\theta) - \max_a Q(s_i, a|\theta)$
- ▶ Corresponds to a regret for not performing the best action
- ▶ $V(s_i) = \max_a Q(s_i, a|\theta)$
- ▶ Why use it?
 - ▶ Put forward by Baird to stabilize function approximation
 - ▶ In the likelihood ratio view, corresponds to the optimal baseline (minimizing variance)
 - ▶ In compatible actor-critic architecture, corresponds to the natural gradient
 - ▶ Link to minimizing the KL divergence between subsequent policies



Deisenroth, M. P., Neumann, G., Peters, J., et al. (2013) A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2), 1–142

Dueling networks

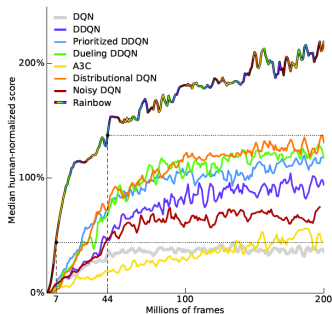


- Instead of $A(s_i, a_i | \theta) = Q(s_i, a_i | \theta) - \max_a Q(s_i, a | \theta)$
- Rather use $Q(s_i, a_i | \theta^Q) = A(s_i, a_i | \theta^A) + V(s_i | \theta^V)$
- Note that $A(s_i, a_i^* | \theta) = 0$
- Center around average A to stabilize: $Q = V + A - \text{average}(A)$
- Better captures some relevant aspects of a control task
- Similar idea in NAF with continuous actions



Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., & de Freitas, N. (2015) Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*

Rainbow



- ▶ A3C, distributional DQN and Noisy DQN presented later
- ▶ Combining all local improvements



Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., & Silver, D. (2017) Rainbow: Combining improvements in deep reinforcement learning. *arXiv preprint arXiv:1710.02298*

Any question?



Send mail to: Olivier.Sigaud@upmc.fr



Baird, L. C.

Reinforcement learning in continuous time: Advantage updating.

In Proceedings of the International Conference on Neural Networks, Orlando, FL, 1994.



Deisenroth, M. P., Neumann, G., Peters, J., et al.

A survey on policy search for robotics.

Foundations and Trends® in Robotics, 2(1–2):1–142, 2013.



Lin, L.-J.

Self-Improving Reactive Agents based on Reinforcement Learning, Planning and Teaching.

Machine Learning, 8(3/4):293–321, 1992.



Tesauro, G.

Temporal difference learning and td-gammon.

Communications of the ACM, 38(3):58–68, 1995.