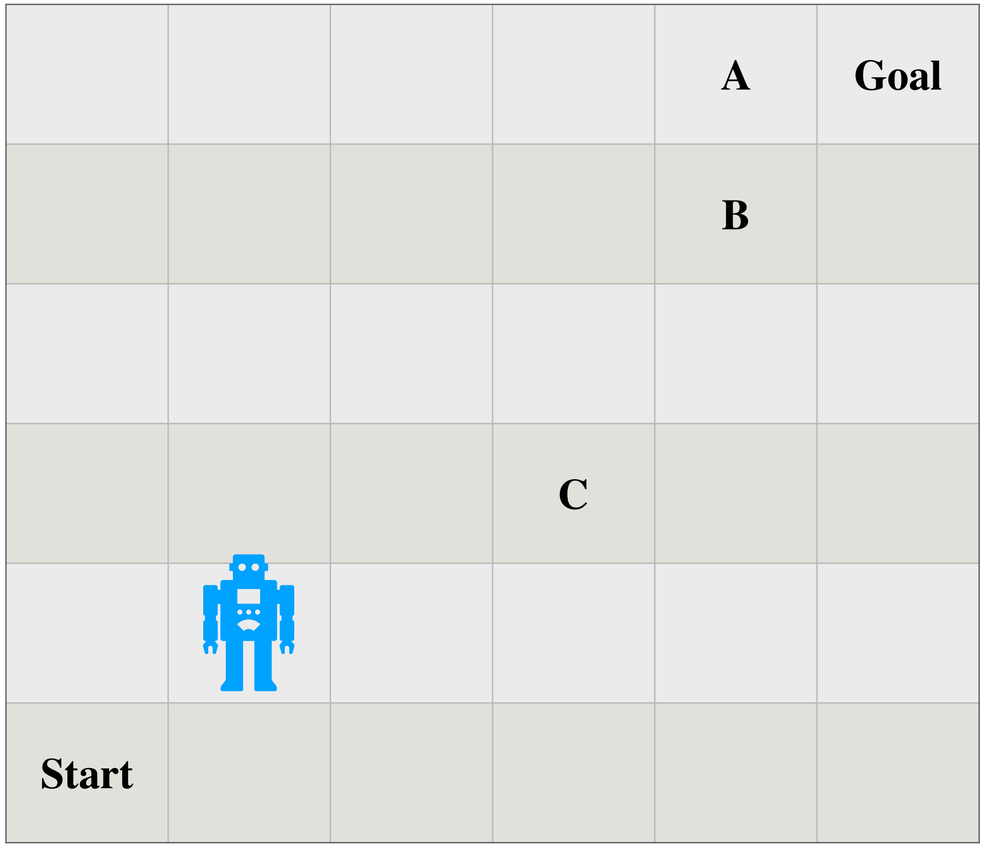
Value Function Methods

Brainstorming

- Which state is “better” than the others?



Dynamic programming

- Can store the entire value function $V(\mathbf{s})$ in a table.
- Iteratively update the table using Bellman principle:

$$V(\mathbf{s}) \leftarrow \max_{\mathbf{a}} \left(r(\mathbf{s}, \mathbf{a}) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \left[V(\mathbf{s}') \right] \right)$$

- Limited to small state and action spaces—curse of dimensionality.

Assumptions:

state and action spaces are discrete

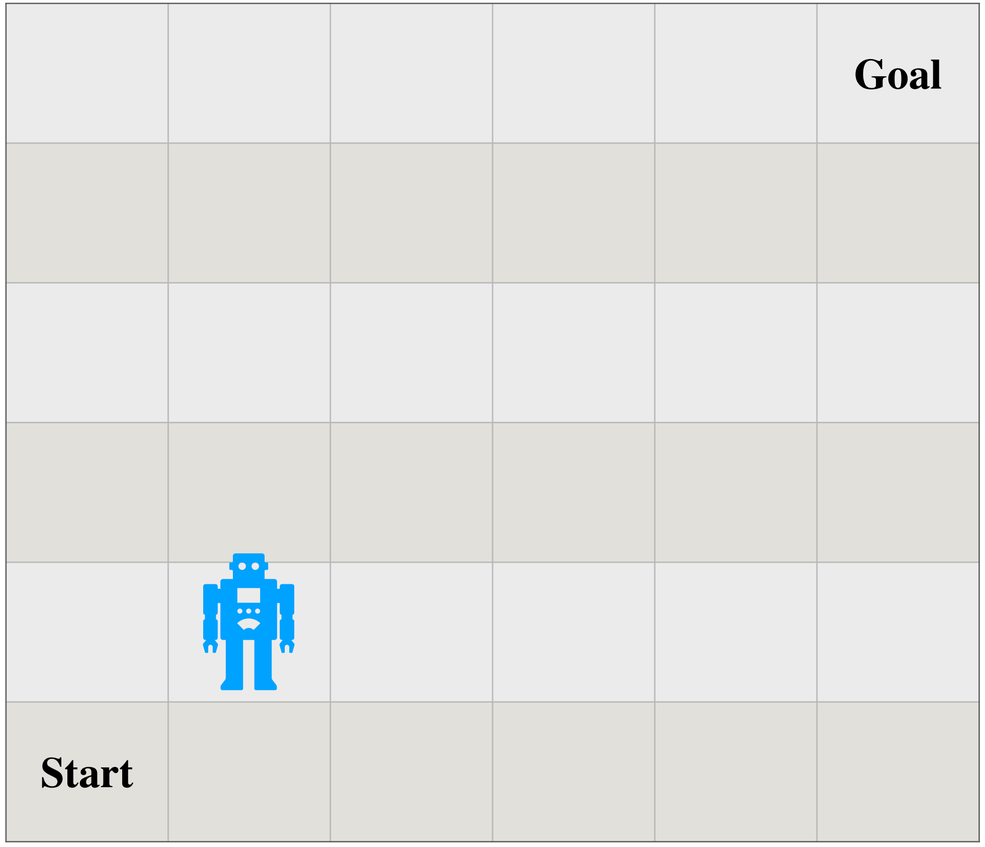
transition function is known

$V(\mathbf{s})$

-5	-4	-3	-2	-1	0
-6	-5	-4	-3	-2	-1
-7	-6	-5	-4	-3	-2
-8	-7	-6	-5	-4	-3
-9	-8	-7	-6	-5	-4
-10	-9	-8	-7	-6	-5

Quiz

- What is the size of the value function table in the 6 by 6 maze navigation?



Quiz

- Can we store the value function of autonomous driving in a table?

Euro Truck 2

Value iteration

- If we keep track of state-action values in a table, we can define the policy entirely

Assumptions:

state and action spaces are discrete

transition function is known

s \ a	$\mathbf{a}_1$	$\mathbf{a}_2$	…	$\mathbf{a}_m$
$\mathbf{s}_1$	$Q(\mathbf{s}_1,\mathbf{a}_1)$	$Q(\mathbf{s}_1,\mathbf{a}_2)$	…	$Q(\mathbf{s}_1,\mathbf{a}_m)$
$\mathbf{s}_2$	$Q(\mathbf{s}_2,\mathbf{a}_1)$	$Q(\mathbf{s}_2,\mathbf{a}_2)$	…	$Q(\mathbf{s}_2,\mathbf{a}_m)$
…	…	…	…	…
$\mathbf{s}_n$	$Q(\mathbf{s}_n,\mathbf{a}_1)$	$Q(\mathbf{s}_n,\mathbf{a}_2)$	…	$Q(\mathbf{s}_n,\mathbf{a}_m)$

Best actions are marked by yellow squares

while not converge:

$$Q^\pi(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}\left[V^\pi(\mathbf{s}')\right]$$

$$V^\pi(\mathbf{s}) \leftarrow \max_{\mathbf{a}} Q(\mathbf{s},\mathbf{a})$$

Policy iteration

- The idea of dynamic programming can be directly applied to policy iteration

Assumptions:

- state and action spaces are discrete
- transition function is known
- deterministic policy

while not converge:

policy evaluation: $V^{\pi}(\mathbf{s}) \leftarrow r(\mathbf{s}, \pi(\mathbf{s})) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \pi(\mathbf{s}))}\left[V^{\pi}(\mathbf{s}')\right]$

policy improvement: $\pi(\mathbf{s}) \leftarrow \text{argmax}_{\mathbf{a}} r(\mathbf{s}, \mathbf{a}) + \gamma E_{\mathbf{s}'}\left[V^{\pi}(\mathbf{s}')\right]$

often need to run multiple passes to update value function

Fitted value iteration

- Most motor control problems have continuous state space and action space—tabular representation will not work

Assumptions:

~~state and action spaces are discrete~~

transition function is known

The problem with fitted value iteration is that we need to maximize on the transition function, because **s**′ depends on the optimization variable **a**. This is very difficult to do.

- Update the optimal value for a random set of states and fit the value function to them.

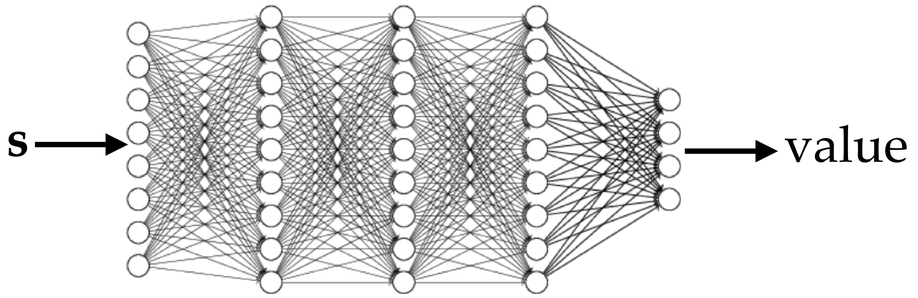
while not converge:

$$y_i \leftarrow \max_{\mathbf{a}} \left(r(\mathbf{s}_i, \mathbf{a}) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}_i, \mathbf{a})} \left[V_\phi(\mathbf{s}') \right] \right)$$

$$\phi \leftarrow \text{argmin}_\phi \frac{1}{2} \sum_i \| V_\phi(\mathbf{s}_i) - y_i \|^2$$

This is a regression problem

Fitted value function parameterized by ϕ



Fitted Q iteration

- Idea: instead of fitting the state value function V, we fit Q, the value of the state and action.

- Represent the Q function as a neural network function approximator.

- Update the optimal value for a random set of states and fit the value function to them.

while not converge:

$$y_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}_i, \mathbf{a}_i)} \left[\max_{a'} Q_\phi(\mathbf{s}', \mathbf{a}') \right]$$

$$\phi \leftarrow \text{argmin}_\phi \frac{1}{2} \sum_i \| Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - y_i \|^2$$

Assumptions:

~~state and action spaces are discrete~~

~~transition function is known~~

In practice, we use a sample estimator to approximate the expectation.

The maximization is now operating on Q, independent of transition function. A much easier problem.

Fitted Q iteration

while not converge:

for i =1 to N

Iterate N times to collect tuples based on some policy (could be the current policy)

collect tuples ($\mathbf{s}_i$, $\mathbf{a}_i$, $\mathbf{s}_i'$, r_i,) from some policy

for k =1 to K

Iterate K times to improve Q before collecting new data.

$$y_i \leftarrow r_i + \gamma \max_{a'} Q_\phi(\mathbf{s}_i', \mathbf{a}')$$

for j =1 to M

Run M steps of gradient descent to optimize Q.

$$\phi \leftarrow \text{argmin}_\phi \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - y_i\|^2$$

Quiz

- Is fitted Q iteration an on-policy or off-policy algorithm?

Only use the samples from the current policy

Use the samples from any policy

$$y_i \leftarrow r_i + \gamma \max_{a'} Q_\phi(\mathbf{s}'_i, \mathbf{a}')$$

$$\phi \leftarrow \text{argmin}_\phi \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - y_i\|^2$$

Online Q-learning

while not converge:

collect tuples ($\mathbf{s}_i$, $\mathbf{a}_i$, $\mathbf{s}_i'$, r_i) from some policy

~~**for** k =1 to K:~~

$$y_i \leftarrow r_i + \gamma \max_{a'} Q_\phi(\mathbf{s}_i', \mathbf{a}')$$

~~**for** j =1 to M:~~

$$\phi \leftarrow \text{argmin}_\phi \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - y_i\|^2$$

Flatten the for loops—making $K = M = 1$.

Quiz

- What's the gradient of the objective function in Q-learning?

$$\phi \leftarrow \text{argmin}_\phi \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - y_i\|^2$$

Online Q-learning

while not converge:

take action **a** according to some policy and observe (**s**, **a**, **s**', r)

$$y \leftarrow r + \gamma \max_{a'} Q_\phi(\mathbf{s}', \mathbf{a}')$$

$$\phi \leftarrow \phi - \alpha \frac{\partial Q_\phi}{\partial \phi}(\mathbf{s}, \mathbf{a})(Q_\phi(\mathbf{s}, \mathbf{a}) - y)$$

Stochastic policy allows for action exploration.

epsilon-greedy:

$$\pi(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 - \epsilon \text{ if } \mathbf{a}_t = \text{argmax}_{\mathbf{a}} Q_\phi(\mathbf{s}_t, \mathbf{a}) \\ \frac{\epsilon}{|\mathcal{A}|-1} \text{ otherwise} \end{cases}$$

Boltzmann exploration:

$$\pi(\mathbf{a}_t|\mathbf{s}_t) \propto \exp Q_\phi(\mathbf{s}_t, \mathbf{a}_t)$$

Deep Q-learning

- Address two problems in online Q-learning:
  - Samples are correlated—training result is highly biased.
  - Regression target changes every iteration—learning progression is unstable.
- Demonstrate that neural nets can learn human-level skill in Atari games.

Correlated samples

- Sequential data are strongly correlated

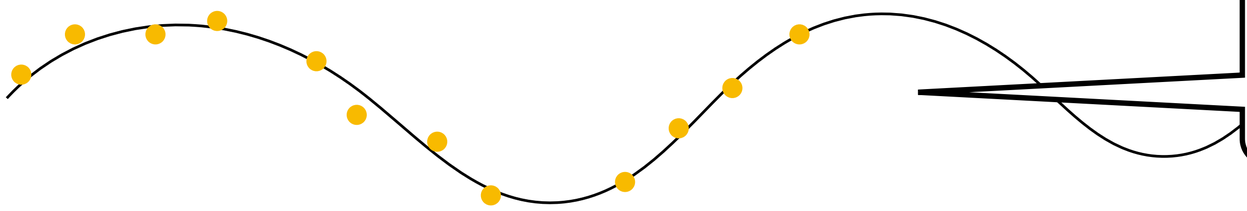
a is correlated to **a** in previous iteration

take action **a** according to some policy and observe (**s**, **a**, **s**′, r)

$$y \leftarrow r + \gamma \max_{a'} Q_\phi(\mathbf{s}', \mathbf{a}')$$

$$\phi \leftarrow \phi - \alpha \frac{\partial Q_\phi}{\partial \phi}(\mathbf{s}, \mathbf{a})(Q_\phi(\mathbf{s}, \mathbf{a}) - y)$$

- Stochastic gradient descent assumes training samples are uncorrelated



a model trained on correlated data tends to overfit to current batch of data and forget previously batches.

Replay buffers

- Since Q-learning is off-policy, we can sample data from previously generated dataset to break correlation.

  while not converge:

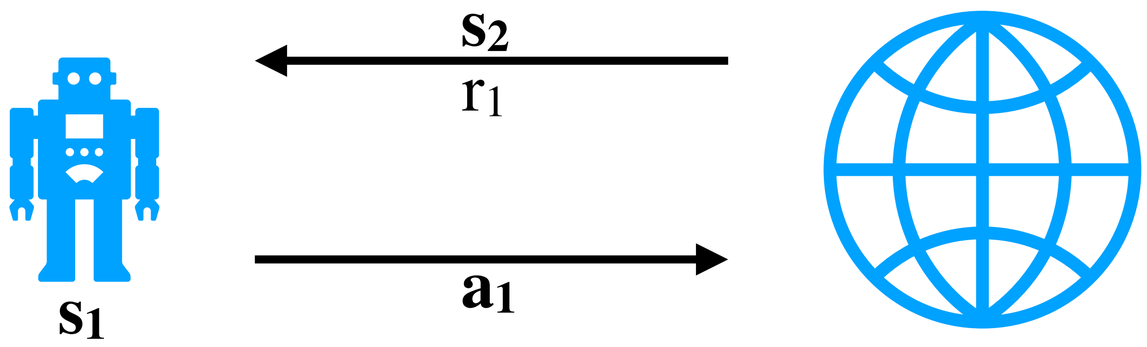
    collect tuples ($\mathbf{s}_i$, $\mathbf{a}_i$, $\mathbf{s}_i'$, r_i) according to some policy and store them in buffer

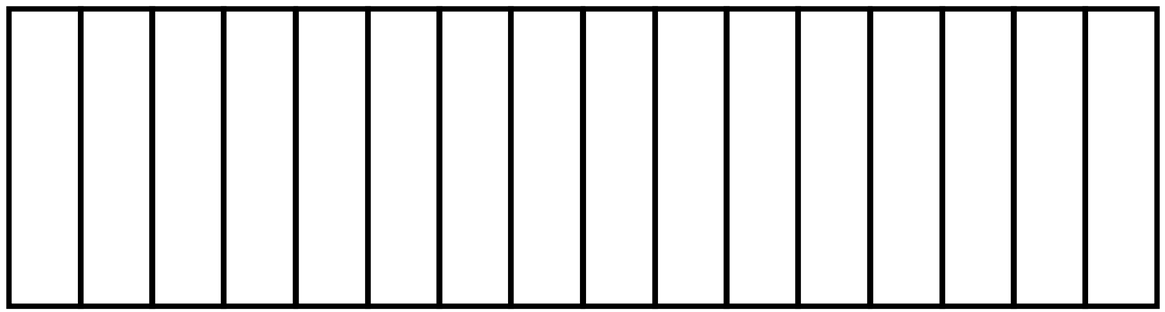
    sample a batch of tuples ($\mathbf{s}_i$, $\mathbf{a}_i$, $\mathbf{s}_i'$, r_i) from buffer

$$\phi \leftarrow \phi - \alpha \sum_i \frac{\partial Q_\phi}{\partial \phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - (r_i + \gamma \max_{\mathbf{a}} Q_\phi(\mathbf{s}_i', \mathbf{a})))$$

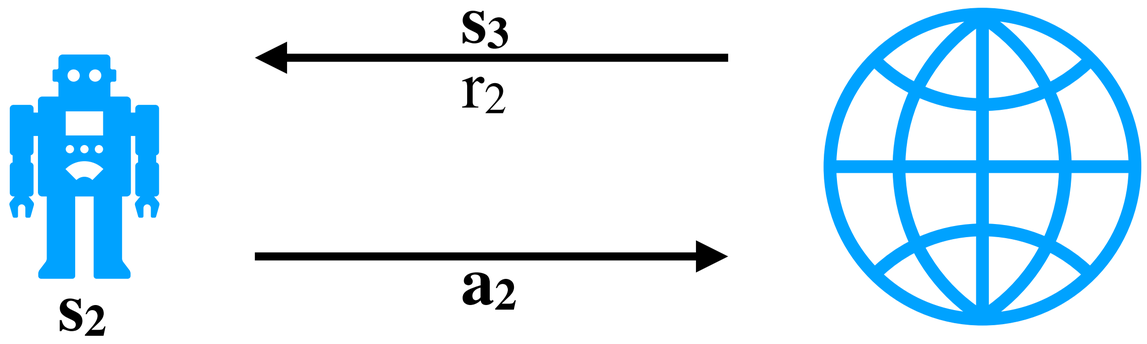
Benefits:
1. Samples are not correlated
2. Gradient is estimated by multiple samples, reducing variance

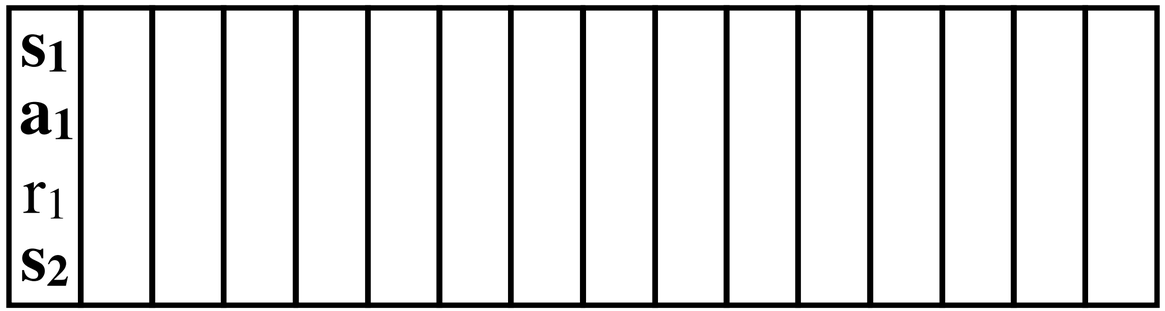
Replay buffers





Replay buffers





Brainstorming

- How would you deal with the case when the replay buffer is full?

Brainstorming

- Are all the samples equally important?

Unstable learning

while not converge:

collect tuples ($\mathbf{s}_i$, $\mathbf{a}_i$, $\mathbf{s}_i'$, r_i) according to some policy and store them in buffer

sample a batch of tuples ($\mathbf{s}_i$, $\mathbf{a}_i$, $\mathbf{s}_i'$, r_i) from buffer

$$\phi \leftarrow \phi - \alpha \sum_i \frac{\partial Q_\phi}{\partial \phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - (r_i + \gamma \max_{\mathbf{a}} Q_\phi(\mathbf{s}_i', \mathbf{a})))$$

This gradient is incorrect. We should either include maxQ in gradient calculation, or make maxQ independent of ϕ.

Q-learning treats maxQ as constant to ϕ, but this constant keeps changing every iteration.

Target networks

while not converge:

update target network: $\phi' \leftarrow \phi$

1. Create a target network and update it periodically using current Q network

for *n =1* to *N:*

collect tuples ($\mathbf{s}_i$, $\mathbf{a}_i$, $\mathbf{s}_i'$, r_i) according to some policy and store them in buffer

sample a batch of tuples ($\mathbf{s}_i$, $\mathbf{a}_i$, $\mathbf{s}_i'$, r_i) from buffer

$$\phi \leftarrow \phi - \alpha \sum_i \frac{\partial Q_\phi}{\partial \phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - (r_i + \gamma \max_{\mathbf{a}} Q'_\phi(\mathbf{s}'_i, \mathbf{a})))$$

The inner loop is just supervised regression solved by stochastic gradient descent

2. Use target network to estimate value

DQN algorithm

while not converge:

update target network:

for n =1 to N:

collect tuples ($\mathbf{s}_i$, $\mathbf{a}_i$, $\mathbf{s}_i'$, r_i) according to some policy and store them in buffer

for k =1 to K:

sample a batch of tuples ($\mathbf{s}_i$, $\mathbf{a}_i$, $\mathbf{s}_i'$, r_i) from buffer

$$\phi \leftarrow \phi - \alpha \sum_i \frac{\partial Q_\phi}{\partial \phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - (r_i + \gamma \max_{\mathbf{a}} Q'_\phi(\mathbf{s}'_i, \mathbf{a})))$$

Toward Continuous Actions

$$\phi \leftarrow \phi - \alpha \sum_i \frac{\partial Q_\phi}{\partial \phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - (r_i + \gamma \max_{\mathbf{a}} Q'_\phi(\mathbf{s}'_i, \mathbf{a})))$$

The maximization can be costly for high dimensional and/or continuous actions.

- We can optimize action a using an optimization method, such as Cross Entropy Method or CMA-ES.
- Alternatively, we can optimize a policy along with a Q function.

Continuous deep Q-learning

- Deep deterministic policy gradient (DDPG) is a off-policy actor-critic method that works for continuous problems
- Exploration: $\mu(\mathbf{s})$+ noise and replay buffer
- Value function update:

$$\mathbf{w} = \mathbf{w} + \alpha(y - Q(\mathbf{s}, \mathbf{a}))\nabla_{\mathbf{w}} Q$$

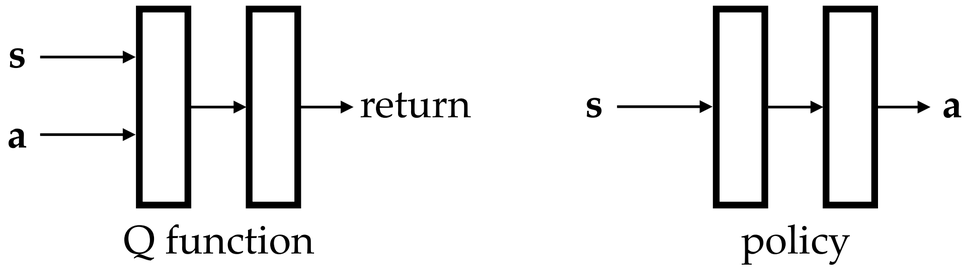
$$y = r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \hat{Q}(\mathbf{s}', \hat{\mu}(\mathbf{s}'))$$

soft target networks

- Policy update:

$$\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} \mu(\mathbf{s}) \nabla_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})|_{\mathbf{a}=\mu(\mathbf{s})}$$

- Architecture:



Reading List

Iconic DQN paper

- Paper #1: Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S., 2015. Human-level control through deep reinforcement learning. nature, 518(7540), pp.529-533.

A summary of all the DQN variants

- Paper #2: Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M. and Silver, D., 2018, April. Rainbow: Combining improvements in deep reinforcement learning. In Thirty-second AAAI conference on artificial intelligence.

Continuous Actions

- Paper #3: Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D., 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

Robust, Sample Efficient Off-policy Algorithm

- Paper #4: Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P. and Levine, S., 2018. Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905.

Reading List (cont')

TD3: Discusses Over-estimation

- Runner-up: Fujimoto, S., Hoof, H. and Meger, D., 2018, July. Addressing function approximation error in actor-critic methods. In International Conference on Machine Learning (pp. 1587-1596). PMLR.

Asynchronous Learning from DM

- Runner-up: Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D. and Kavukcuoglu, K., 2016, June. Asynchronous methods for deep reinforcement learning. In International conference on machine learning (pp. 1928-1937). PMLR.

Another Type of Off-policy Learning

- Runner-up: Peng, X.B., Kumar, A., Zhang, G. and Levine, S., 2019. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. arXiv preprint arXiv:1910.00177.

Expensive but Super Sample-efficient

- Runner-up: Chen, X., Wang, C., Zhou, Z. and Ross, K., 2021. Randomized ensembled double q-learning: Learning fast without a model. arXiv preprint arXiv:2101.05982.

Summary

- We can solve the given MDP problem by learning V or Q functions.
- We can fit neural networks instead of tables for complex problems.
- A few tricks (replay buffers, target networks) are invented for better learning.
- Value function methods can be further extended for high-dimensional continuous actions.