

# DEEP Q-LEARNING

DQN  
Enhancements

Diego Klabjan  
Professor, Industrial Engineering and Management Sciences


Northwestern | McCORMICK SCHOOL OF  
ENGINEERING

# Q-learning with Fitted Q-factor

- Observed data are trajectories
  - Formally,  $U = \{(s_i, a_i, s'_i, r_i) | i \in N\}$
- Loop
  - Sample  $S \subseteq U$
  - For  $i \in S$ 
    - $y_i \leftarrow r_i + \gamma \max_{a'_i} Q_\phi(s'_i, a'_i)$
  - Set  $\phi \leftarrow \operatorname{argmin}_\phi \frac{1}{2} \sum_{i \in S} (Q_\phi(s_i, a_i) - y_i)^2$
- Only remaining drawback how to compute max

# Issues

Online Q iteration algorithm:

- 
1. take some action  $a_i$  and observe  $(s_i, a_i, s'_i, r_i)$
  2.  $y_i = r(s_i, a_i) + \gamma \max_{a'} Q_\phi(s'_i, a')$
  3.  $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(s_i, a_i)(Q_\phi(s_i, a_i) - y_i)$
- These are correlated!
- Gradient descent!  
Convergence is a problem!
- Or it is not gradient descent!

Q-learning is not gradient descent!

$$\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(s_i, a_i) \left( Q_\phi(s_i, a_i) - \left[ r(s_i, a_i) + \gamma \max_{a'} Q_\phi(s'_i, a') \right] \right)$$

No gradient through target value!

# Correlated Samples in Online Q-learning

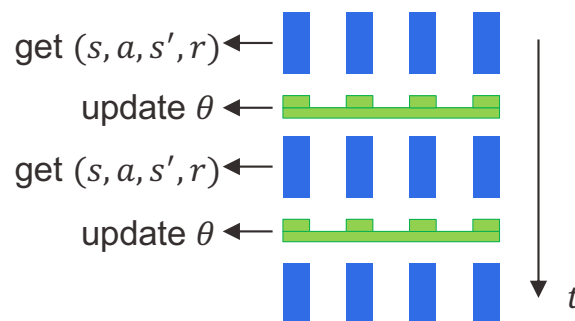
Online Q iteration algorithm:

- Sequential states are strongly correlated
- Target value is always changing

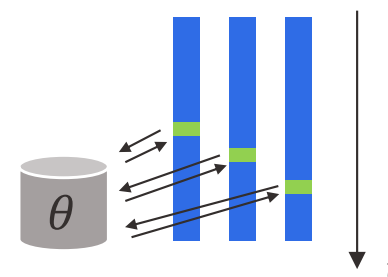
1. take some action  $a_i$  and observe  $(s_i, a_i, s'_i, r_i)$
2.  $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(s_i, a_i) \left( Q_\phi(s_i, a_i) - \left[ r(s_i, a_i) + \gamma \max_{a'} Q_\phi(s'_i, a'_i) \right] \right)$



Synchronized parallel



Asynchronous parallel



# Replay Buffers

Online Q iteration algorithm:

Special case with  $K=1$ , and one gradient step

1. take some action  $a_i$  and observe  $(s_i, a_i, s'_i, r_i)$
2.  $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(s_i, a_i) \left( Q_\phi(s_i, a_i) - \left[ r(s_i, a_i) + \gamma \max_{a'_i} Q_\phi(s'_i, a'_i) \right] \right)$

Full fitted Q-iteration algorithm:

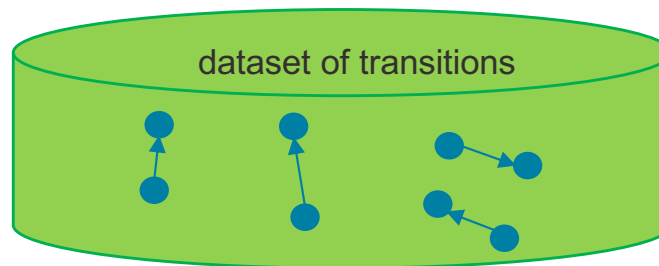
- ~~1. collect dataset  $\{(s_i, a_i, s'_i, r_i)\}$  using some policy~~
2. set  $y_i \leftarrow r(s_i, a_i) + \gamma \max_{a'_i} Q_\phi(s'_i, a'_i)$
- $K \times$  3. set  $\phi \leftarrow \operatorname{argmin}_\phi \frac{1}{2} \sum_i (Q_{\phi'}(s_i, a_i) - y_i)^2$

Any policy will work! (with broad support)

Idea: for different policies store tuple in buffer

Just load data from a buffer here

Still use one gradient step



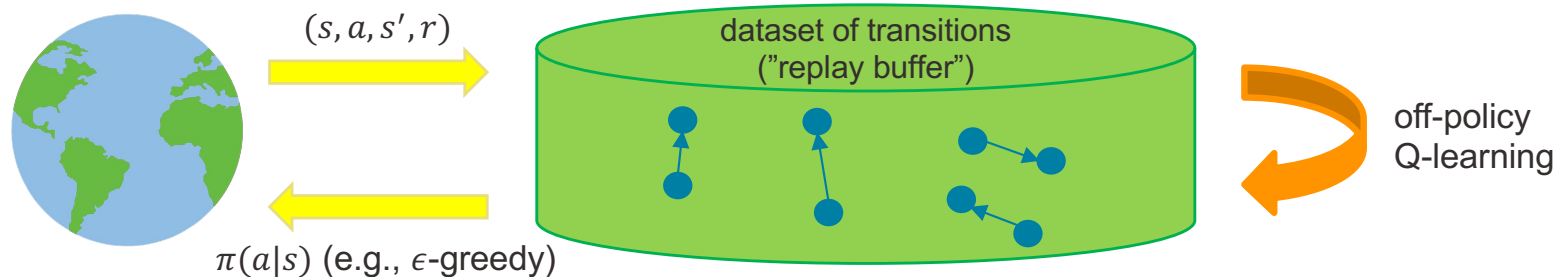
Fitted Q-iteration

# Replay Buffers

Q learning with a replay buffer:

1. sample a batch  $(s_i, a_i, s'_i, r_i)$  from  $\mathcal{B}$  + samples are no longer correlated
2.  $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(s_i, a_i) \left( Q_\phi(s_i, a_i) - \left[ r(s_i, a_i) + \gamma \max_{a'} Q_\phi(s'_i, a'_i) \right] \right)$   
+ multiple samples in the batch (low-variance gradient)

Need to periodically feed the replay buffer

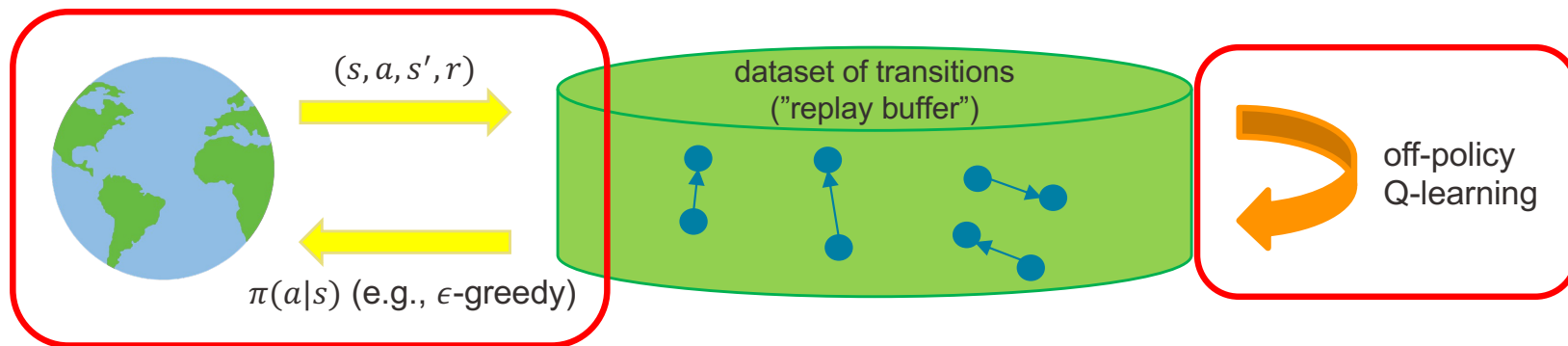


# Framework

Full Q-learning with replay buffer:

1. collect dataset  $\{(s_i, a_i, s'_i, r_i)\}$  using some policy, add it to  $\mathcal{B}$
2. sample a batch  $(s_i, a_i, s'_i, r_i)$  from  $\mathcal{B}$
3.  $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(s_i, a_i) \left( Q_\phi(s_i, a_i) - \left[ r(s_i, a_i) + \gamma \max_{a'} Q_\phi(s'_i, a'_i) \right] \right)$

K=1 is common



# Issues Revisited

Online Q iteration algorithm:

1. take some action  $a_i$  and observe  $(s_i, a_i, s'_i, r_i)$
  2.  $y_i = r(s_i, a_i) + \gamma \max_{a'} Q_\phi(s'_i, a')$
  3.  $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(s_i, a_i)(Q_\phi(s_i, a_i) - y_i)$
- ~~these are correlated!~~  
Use replay buffer

Q-learning is not gradient descent!

$$\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(s_i, a_i) \left( Q_\phi(s_i, a_i) - \left[ r(s_i, a_i) + \gamma \max_{a'} Q_\phi(s'_i, a') \right] \right)$$

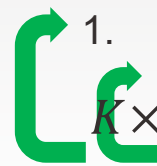
This is still a problem!

no gradient through target value



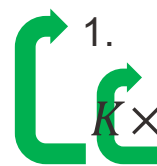
# Q-Learning and Regression

Full Q-learning with replay buffer:

- 
1. collect dataset  $\{(s_i, a_i, s'_i, r_i)\}$  using some policy, add it to  $\mathcal{B}$
  2. sample a batch  $(s_i, a_i, s'_i, r_i)$  from  $\mathcal{B}$
  3.  $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(s_i, a_i) \left( Q_\phi(s_i, a_i) - \left[ r(s_i, a_i) + \gamma \max_{a'} Q_\phi(s'_i, a'_i) \right] \right)$

One gradient step, moving target

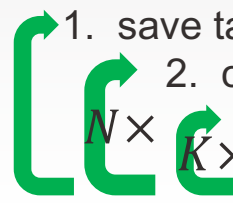
Full fitted Q-iteration algorithm:

- 
1. collect dataset  $\{(s_i, a_i, s'_i, r_i)\}$  using some policy
  2. set  $y_i \leftarrow r(s_i, a_i) + \gamma \max_{a'_i} Q_\phi(s'_i, a'_i)$
  3. set  $\phi \leftarrow \operatorname{argmin}_\phi \frac{1}{2} \sum_i (Q_\phi(s_i, a_i) - y_i)^2$

Perfectly well-defined, stable regression

# Q-Learning with Target Networks

Full Q-learning with replay buffer:

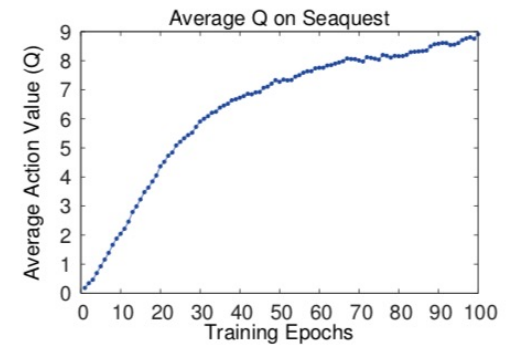
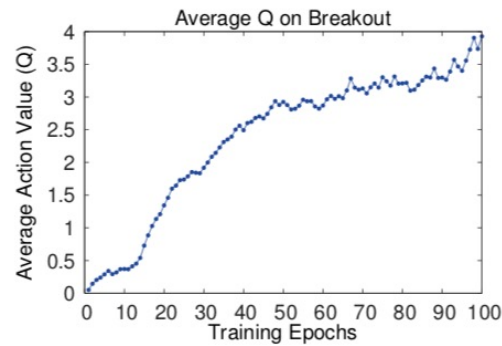
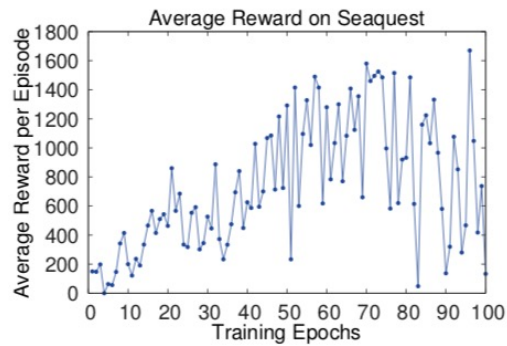
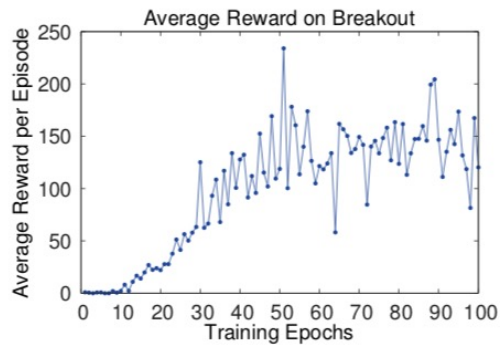
- 
1. save target network parameters:  $\phi' \leftarrow \phi$
  2. collect dataset  $\{(s_i, a_i, s'_i, r_i)\}$  using some policy, add it to  $\mathcal{B}$
  3. sample a batch  $(s_i, a_i, s'_i, r_i)$  from  $\mathcal{B}$
  4.  $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(s_i, a_i) \left( Q_\phi(s_i, a_i) - \underbrace{\left[ r(s_i, a_i) + \gamma \max_{a'} Q_{\phi'}(s'_i, a'_i) \right]}_{\text{Targets don't change in inner loop!}} \right)$
- supervised regression

# Performance

	B. Rider	Breakout	Enduro	Pong	Q*bert	Seaquest	S. Invaders
Random	354	1.2	0	−20.4	157	110	179
Sarsa [3]	996	5.2	129	−19	614	665	271
Contingency [4]	1743	6	159	−17	960	723	268
DQN	<b>4092</b>	<b>168</b>	<b>470</b>	<b>20</b>	<b>1952</b>	<b>1705</b>	<b>581</b>
Human	7456	31	368	−3	18900	28010	3690
HNeat Best [8]	3616	52	106	19	1800	920	<b>1720</b>
HNeat Pixel [8]	1332	4	91	−16	1325	800	1145
DQN Best	<b>5184</b>	<b>225</b>	<b>661</b>	<b>21</b>	<b>4500</b>	<b>1740</b>	1075

Mnih et al, Playing Atari with Deep Reinforcement Learning

# Performance



Mnih et al, Playing Atari with Deep Reinforcement Learning

# “Classic” deep Q-learning Algorithm (DQN)

Q-learning with replay buffer and target network:

1. save target network parameters:  $\phi' \leftarrow \phi$
2. collect dataset  $\{(s_i, a_i, s'_i, r_i)\}$  using some policy, add it to  $\mathcal{B}$
- $N \times$  3. sample a batch  $(s_i, a_i, s'_i, r_i)$  from  $\mathcal{B}$
- $K \times$  4.  $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(s_i, a_i) \left( Q_\phi(s_i, a_i) - \left[ r(s_i, a_i) + \gamma \max_{a'_i} (Q_{\phi'}(s'_i, a'_i)) \right] \right)$

“Classic” deep Q-learning algorithm:

1. take some action  $a_i$  and observe  $(s_i, a_i, s'_i, r_i)$ , add it to  $\mathcal{B}$
  2. sample mini-batch  $\{s_j, a_j, s'_j, r_j\}$  from  $\mathcal{B}$  uniformly
  3. compute  $y_j = r_j + \gamma \max_{a'_j} Q_{\phi'}(s'_j, a'_j)$  using *target* network  $Q_{\phi'}$
  4.  $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(s_j, a_j) (Q_\phi(s_j, a_j) - y_j)$
  5. update  $\phi'$ : copy  $\phi$  every  $N$  steps
- }  $K = 1$

# Alternative Target Network

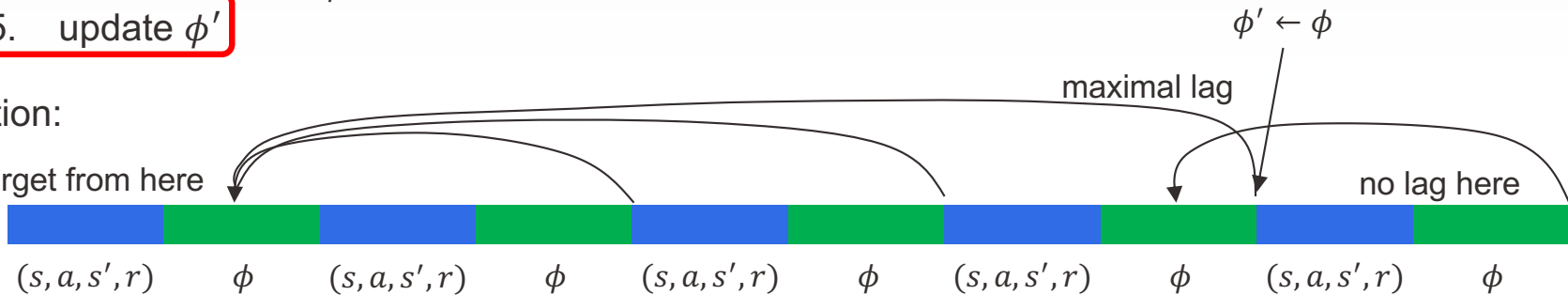
“Classic” deep Q-learning algorithm:

1. take some action  $a_i$  and observe  $(s_i, a_i, s'_i, r_i)$ , add it to  $\mathcal{B}$
2. sample mini-batch  $\{s_j, a_j, s'_j, r_j\}$  from  $\mathcal{B}$  uniformly
3. compute  $y_j = r_j + \gamma \max_{a'_j} Q_{\phi'}(s'_j, a'_j)$  using *target* network  $Q_{\phi'}$
4.  $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(s_j, a_j)(Q_\phi(s_j, a_j) - y_j)$
5. update  $\phi'$

$K = 1$

Intuition:

get target from here



Feels weirdly uneven, can we always have the same lag?

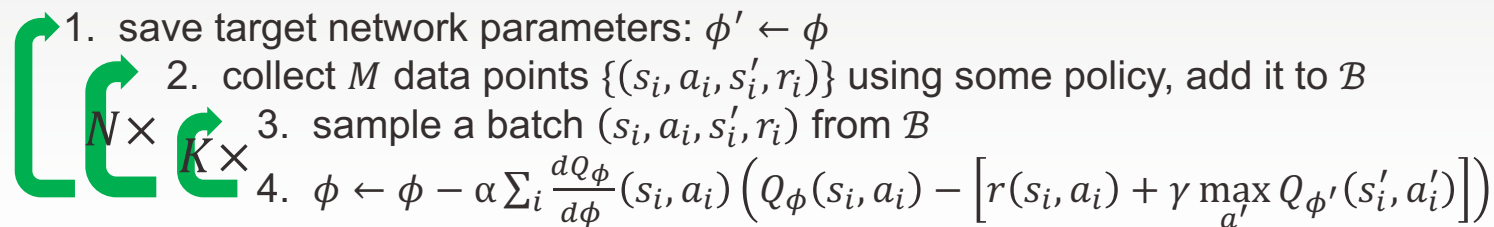
5. update  $\phi'$ :  $\phi' \leftarrow \tau \phi' + (1 - \tau)\phi$

$\tau = 0.999$  works well

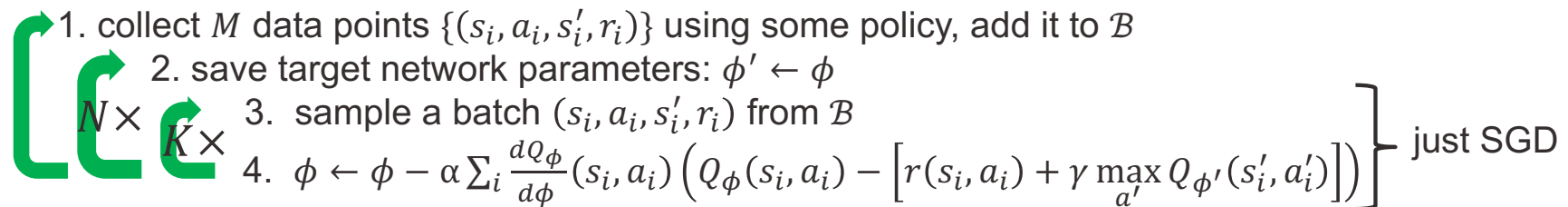
# Fitted Q-iteration and Q-learning

Q-learning with replay buffer and target network:

DQN:  $N = 1, K = 1$

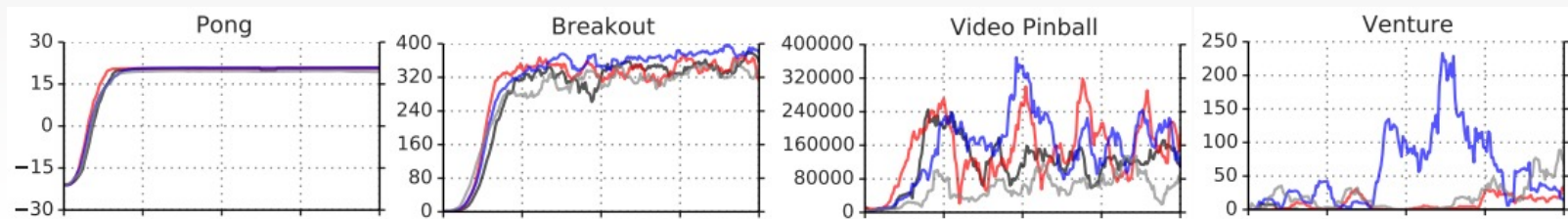
- 
1. save target network parameters:  $\phi' \leftarrow \phi$
  2. collect  $M$  data points  $\{(s_i, a_i, s'_i, r_i)\}$  using some policy, add it to  $\mathcal{B}$
  3. sample a batch  $(s_i, a_i, s'_i, r_i)$  from  $\mathcal{B}$
  4.  $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(s_i, a_i) \left( Q_\phi(s_i, a_i) - \left[ r(s_i, a_i) + \gamma \max_{a'} Q_{\phi'}(s'_i, a'_i) \right] \right)$

Fitted Q-learning (written similarly as above):

- 
1. collect  $M$  data points  $\{(s_i, a_i, s'_i, r_i)\}$  using some policy, add it to  $\mathcal{B}$
  2. save target network parameters:  $\phi' \leftarrow \phi$
  3. sample a batch  $(s_i, a_i, s'_i, r_i)$  from  $\mathcal{B}$
  4.  $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(s_i, a_i) \left( Q_\phi(s_i, a_i) - \left[ r(s_i, a_i) + \gamma \max_{a'} Q_{\phi'}(s'_i, a'_i) \right] \right)$
- } just SGD

# Simple Practical Tips for Q-learning

- Q-learning takes some care to stabilize
  - Test on easy, reliable tasks first, make sure your implementation is correct



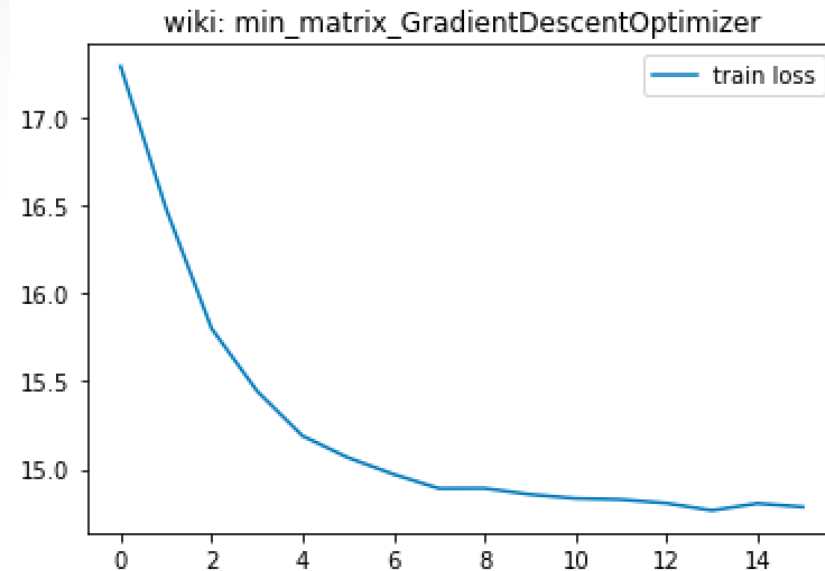
T. Schaul, J. Quan, I. Antonoglou, and D. Silver. "Prioritized experience replay". [arXiv:1511.05952](https://arxiv.org/abs/1511.05952), Figure 7

- Large replay buffers help improve stability
  - Looks more like fitted Q-iteration
- It takes time, be patient—might be no better than random for a while
- Start with high exploration (epsilon) and gradually reduce



# Simple Practical Tips for Q-learning

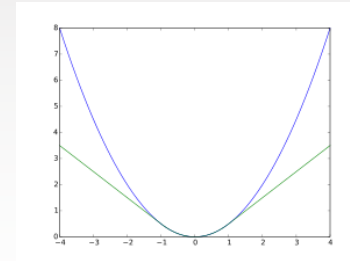
- Standard SGD in supervised learning
  - Loss drop drastic at the beginning
- DQN
  - Slow at the beginning
  - S-curve



# Advanced Tips for Q-learning

- Bellman error gradients can be big; clip gradients or use Huber loss

$$L(x) = \begin{cases} \frac{x^2}{2} & \text{if } |x| \leq \delta \\ \delta|x| - \frac{\delta^2}{2} & \text{otherwise} \end{cases}$$



- Double Q-learning helps a lot in practice
  - Did not cover
- N-step returns also help a lot
  - Have issues
  - Did not cover
- Schedule exploration (high to low) and learning rates (high to low)
  - Adam optimizer can help too
- Run multiple random seeds
  - Very inconsistent between runs