Deep Q-Learning

Reinforcement Learning

September 29, 2020

MSU

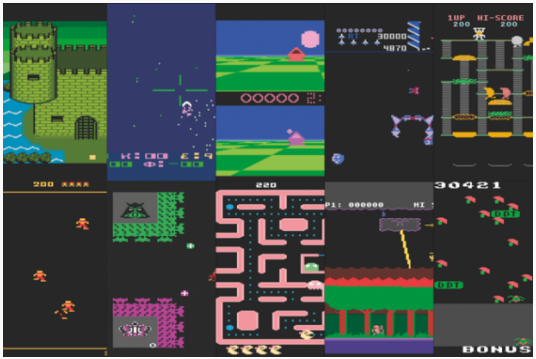
Atari Games

Setup: $|\mathcal{S}| \ll \infty, |\mathcal{A}| \ll \infty$

Atari Games

Setup: ~~$|\mathcal{S}| \ll \infty$~~, $|\mathcal{A}| \ll \infty$

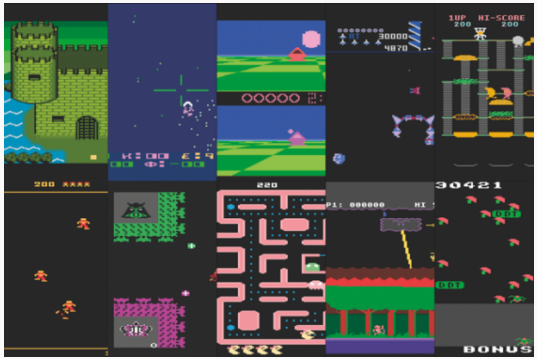
- 57 various games
- Only screen image as input.
- No game-specific features.
- Finite-state case... not quite finite.
- $|\mathcal{A}| \leqslant 18$



Atari Games

Setup: ~~$|\mathcal{S}| \ll \infty$~~, $|\mathcal{A}| \ll \infty$

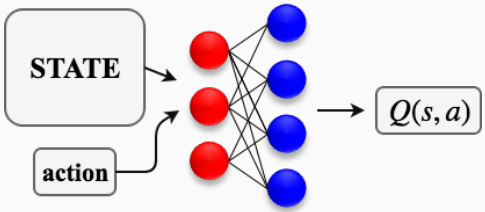
- 57 various games
- Only screen image as input.
- No game-specific features.
- Finite-state case... not quite finite.
- $|\mathcal{A}| \leqslant 18$



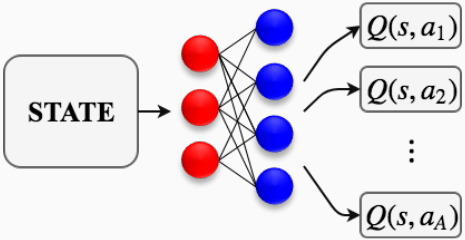
Approximate $Q^*(s, a)$ with neural network!

Deep Q-network

Option 1

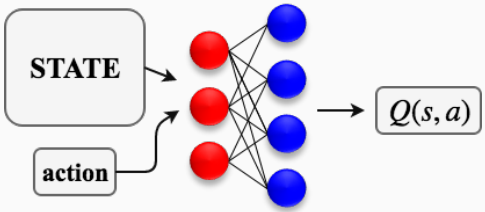


Option 2



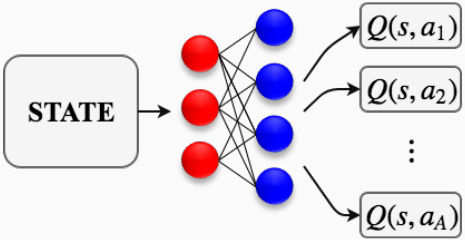
Deep Q-network

Option 1



× $\underset{a}{\operatorname{argmax}}\, Q^*(s, a, \theta)$ — expensive!

Option 2



✓ $\underset{a}{\operatorname{argmax}}\, Q^*(s, a, \theta)$ — one forward pass.

Intuition: Q-learning as gradient descent

Consider *tabular* parametric family:

$$Q^*(s, a, \theta) := \theta_{s,a}$$

Intuition: Q-learning as gradient descent

Consider *tabular* parametric family:

$$Q^*(s, a, \theta) := \theta_{s,a}$$

$$\underline{Q^*(s, a, \theta)} \leftarrow \underline{Q^*(s, a, \theta)} + \frac{\alpha}{2}\left(y - \overbrace{Q^*(s, a, \theta)}^{\hat{y}}\right)$$

Intuition: Q-learning as gradient descent

Consider *tabular* parametric family:

$$Q^*(s, a, \theta) := \theta_{s,a}$$

$$\underline{Q^*(s, a, \theta)} \leftarrow \underline{Q^*(s, a, \theta)} + \frac{\alpha}{2}\left(y - \overbrace{Q^*(s, a, \theta)}^{\hat{y}}\right)$$

$$\underline{\theta_{s,a}} \leftarrow \underline{\theta_{s,a}} - \alpha \nabla_{\hat{y}} \left(y - \hat{y}\right)^2$$

Intuition: Q-learning as gradient descent

Consider *tabular* parametric family:

$$Q^*(s, a, \theta) := \theta_{s,a}$$

$$\underline{Q^*(s, a, \theta)} \leftarrow \underline{Q^*(s, a, \theta)} + \frac{\alpha}{2}\left(y - \overbrace{Q^*(s, a, \theta)}^{\hat{y}}\right)$$

$$\underline{\theta_{s,a}} \leftarrow \underline{\theta_{s,a}} - \alpha \nabla_{\hat{y}} \left(y - \hat{y}\right)^2$$

$$\theta \leftarrow \theta - \alpha \nabla_{\hat{y}} \left(y - \hat{y}\right)^2 \mathrm{OHE}\left(s, a\right)$$

Intuition: Q-learning as gradient descent

Consider *tabular* parametric family:

$$Q^*(s, a, \theta) := \theta_{s,a}$$

$$\underline{Q^*(s, a, \theta)} \leftarrow \underline{Q^*(s, a, \theta)} + \frac{\alpha}{2}\left(y - \overbrace{Q^*(s, a, \theta)}^{\hat{y}}\right)$$

$$\underline{\theta_{s,a}} \leftarrow \underline{\theta_{s,a}} - \alpha \nabla_{\hat{y}} \left(y - \hat{y}\right)^2$$

$$\theta \leftarrow \theta - \alpha \nabla_{\hat{y}} \left(y - \hat{y}\right)^2 \mathrm{OHE}\left(s, a\right)$$

$$\theta \leftarrow \theta - \alpha \underbrace{\nabla_{\hat{y}} \left(y - \hat{y}\right)^2 \nabla_\theta \hat{y}}_{\nabla_\theta \mathrm{MSE}(s,a,y,\theta)}$$

Consider the following **regression task**:

- s, a is input;
- $y \in \mathbb{R}$ is target:

$$y(s, a) := r + \gamma \max_{a'} Q^*(s', a', \theta_k)$$

where $r = r(s, a), s' \sim p(s' \mid s, a)$;

Consider the following **regression task**:

- s, a is input;
- $y \in \mathbb{R}$ is target:

$$y(s, a) := r + \gamma \max_{a'} Q^*(s', a', \theta_k)$$

  where $r = r(s, a), s' \sim p(s' \mid s, a)$;
- MSE loss function: $\text{Loss}(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$

Consider the following **regression task**:

- s, a is input;
- $y \in \mathbb{R}$ is target:

$$y(s, a) := r + \gamma \max_{a'} Q^*(s', a', \theta_k)$$

  where $r = r(s, a), s' \sim p(s' \mid s, a)$;
- MSE loss function: $\text{Loss}(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$

$$\mathbb{E}_{(s,a,r,s')}(y - Q^*(s, a, \theta_{k+1}))^2 \to \min_{\theta_{k+1}}$$

Solving Bellman equations: deep version

$$\mathbb{E}_{(s,a,r,s')}(y - Q^*(s, a, \theta_{k+1}))^2 \to \min_{\theta_{k+1}}$$

What solution is optimal?

$$Q^*(s, a, \theta^*_{k+1}) =$$

Solving Bellman equations: deep version

$$\mathbb{E}_{(s,a,r,s')}(y - Q^*(s, a, \theta_{k+1}))^2 \to \min_{\theta_{k+1}}$$

What solution is optimal?

$$Q^*(s, a, \theta^*_{k+1}) = \mathbb{E}_{r,s'|s,a} y =$$

Solving Bellman equations: deep version

$$\mathbb{E}_{(s,a,r,s')}(y - Q^*(s, a, \theta_{k+1}))^2 \to \min_{\theta_{k+1}}$$

What solution is optimal?

$$Q^*(s, a, \theta^*_{k+1}) = \mathbb{E}_{r,s'|s,a} y = \mathbb{E}_{r,s'|s,a}\left[r + \gamma \max_{a'} Q^*(s', a', \theta_k)\right] =$$

Solving Bellman equations: deep version

$$\mathbb{E}_{(s,a,r,s')}(y - Q^*(s, a, \theta_{k+1}))^2 \to \min_{\theta_{k+1}}$$

What solution is optimal?

$$Q^*(s, a, \theta^*_{k+1}) = \mathbb{E}_{r,s'|s,a} y = \mathbb{E}_{r,s'|s,a}\left[r + \gamma \max_{a'} Q^*(s', a', \theta_k)\right] =$$

$$= \underbrace{r(s, a) + \gamma \mathbb{E}_{s' \sim p(s'|s,a)} \max_{a'} Q^*(s', a', \theta_k)}_{\text{value iteration update}}$$

Target network

For how long to solve one regression task?

Target network

For how long to solve one regression task?

- One SGD iteration

Target network

For how long to solve one regression task?

- One SGD iteration
  - × completely unstable :(

Target network

For how long to solve one regression task?

- One SGD iteration
  - × completely unstable :(
- Till convergence

Target network

For how long to solve one regression task?

- One SGD iteration
  - × completely unstable :(
- Till convergence
  - × too long

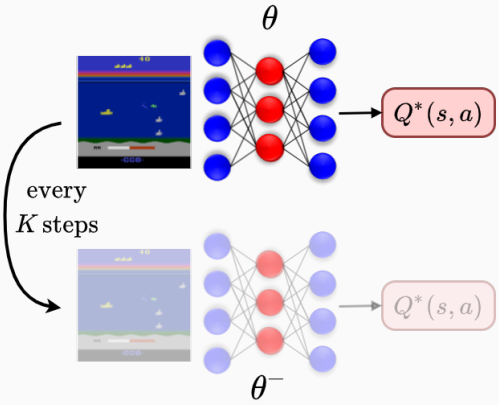
✓ ≈1000 SGD iterations

Target network

For how long to solve one regression task?

- One SGD iteration
  - × completely unstable :(
- Till convergence
  - × too long

✓ ≈1000 SGD iterations



Store a copy of your old Q-network $Q^*(s, a, \theta^-)$

Target network

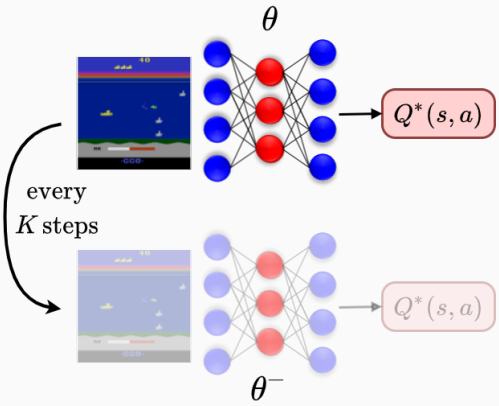
For how long to solve one regression task?

- One SGD iteration
  - × completely unstable :(
- Till convergence
  - × too long

✓ ≈1000 SGD iterations

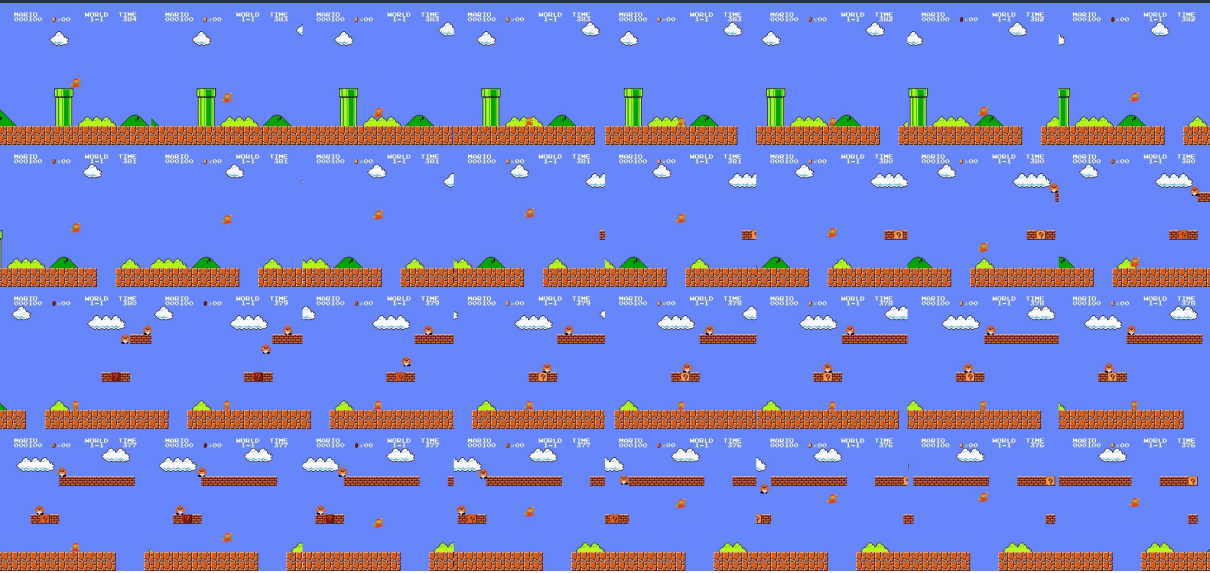
a) $\theta^- \leftarrow \theta$ every K SGD iterations

b) $\theta^- \leftarrow (1-\beta)\theta^- + \beta\theta$

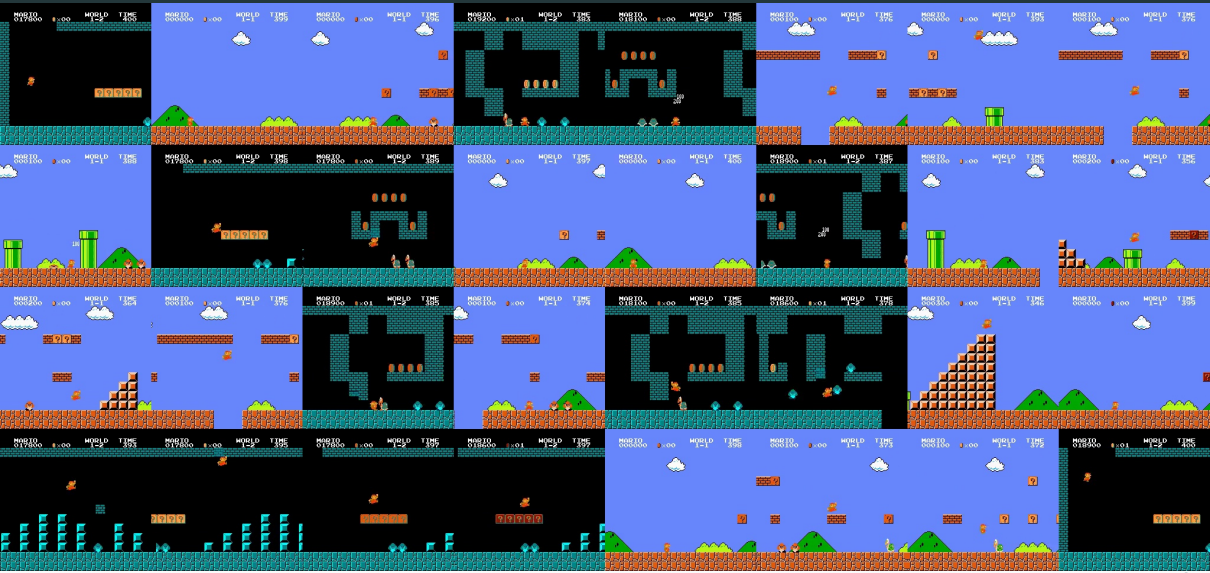


Store a copy of your old Q-network $Q^*(s, a, \theta^-)$

Sample Decorrelation



Experience Replay



Full alrogithm

Deep Q-learning

Initialize $Q^*(s, a, \theta)$ arbitrarily, $\theta^- := \theta$, $\mathcal{D} = \varnothing$;

Full alrogithm

Deep Q-learning

Initialize $Q^*(s, a, \theta)$ arbitrarily, $\theta^- := \theta$, $\mathcal{D} = \varnothing$;

observe s_0;

for $k = 0, 1, 2 \ldots$

- take action $a_k \sim \varepsilon$-greedy$(Q^*(s_k, a, \theta))$;
- observe $r_k, s_{k+1}, \mathrm{done}_{k+1}$, store $(s_k, a_k, r_k, s_{k+1}, \mathrm{done}_{k+1})$ in $\mathcal{D}$;

Full alrogithm

Deep Q-learning

Initialize $Q^*(s, a, \theta)$ arbitrarily, $\theta^- := \theta$, $\mathcal{D} = \varnothing$;

observe s_0;

for $k = 0, 1, 2 \ldots$

- take action $a_k \sim \varepsilon$-greedy$(Q^*(s_k, a, \theta))$;
- observe $r_k, s_{k+1}, \mathrm{done}_{k+1}$, store $(s_k, a_k, r_k, s_{k+1}, \mathrm{done}_{k+1})$ in $\mathcal{D}$;
- sample batch of transitions $\mathbb{T} := (s, a, r, s', \mathrm{done})$ from $\mathcal{D}$;

Full alrogithm

Deep Q-learning

Initialize $Q^*(s, a, \theta)$ arbitrarily, $\theta^- := \theta$, $\mathcal{D} = \varnothing$;

observe s_0;

for $k = 0, 1, 2 \ldots$

- take action $a_k \sim \varepsilon$-greedy$(Q^*(s_k, a, \theta))$;
- observe $r_k, s_{k+1}, \mathrm{done}_{k+1}$, store $(s_k, a_k, r_k, s_{k+1}, \mathrm{done}_{k+1})$ in $\mathcal{D}$;
- sample batch of transitions $\mathbb{T} := (s, a, r, s', \mathrm{done})$ from $\mathcal{D}$;
- $y(\mathbb{T}) := r + \gamma(1 - \mathrm{done}) \max_{a'} Q^*(s', a', \theta^-)$;

Full alrogithm

Deep Q-learning

Initialize $Q^*(s, a, \theta)$ arbitrarily, $\theta^- := \theta$, $\mathcal{D} = \varnothing$;

observe s_0;

for $k = 0, 1, 2 \ldots$

- take action $a_k \sim \varepsilon$-greedy$(Q^*(s_k, a, \theta))$;
- observe $r_k, s_{k+1}, \mathrm{done}_{k+1}$, store $(s_k, a_k, r_k, s_{k+1}, \mathrm{done}_{k+1})$ in $\mathcal{D}$;
- sample batch of transitions $\mathbb{T} := (s, a, r, s', \mathrm{done})$ from $\mathcal{D}$;
- $y(\mathbb{T}) := r + \gamma(1 - \mathrm{done}) \max_{a'} Q^*(s', a', \theta^-)$;
- perform a step of gradient descent:

$$\theta \leftarrow \theta - \frac{\alpha}{B} \sum_{\mathbb{T}} \nabla_\theta \left(Q^*(s, a, \theta) - y(\mathbb{T}) \right)^2$$

Full alrogithm

Deep Q-learning

Initialize $Q^*(s, a, \theta)$ arbitrarily, $\theta^- := \theta$, $\mathcal{D} = \varnothing$;

observe s_0;

for $k = 0, 1, 2 \ldots$

- take action $a_k \sim \varepsilon$-greedy$(Q^*(s_k, a, \theta))$;
- observe $r_k, s_{k+1}, \mathrm{done}_{k+1}$, store $(s_k, a_k, r_k, s_{k+1}, \mathrm{done}_{k+1})$ in $\mathcal{D}$;
- sample batch of transitions $\mathbb{T} := (s, a, r, s', \mathrm{done})$ from $\mathcal{D}$;
- $y(\mathbb{T}) := r + \gamma(1 - \mathrm{done}) \max_{a'} Q^*(s', a', \theta^-)$;
- perform a step of gradient descent:

$$\theta \leftarrow \theta - \frac{\alpha}{B} \sum_{\mathbb{T}} \nabla_\theta \left(Q^*(s, a, \theta) - y(\mathbb{T})\right)^2$$

- update target network: if $k \bmod K = 0$: $\theta^- \leftarrow \theta$

There is a lot to improve

Overestimation Bias and how to fight it

$$\max_{a'}(Q^*(s', a', \theta) + \underbrace{\varepsilon(s', a')}_{\text{approximation error or luck}})$$

Overestimation Bias and how to fight it

$$\mathbb{E}_{\varepsilon} \max_{a'}(Q^*(s', a', \theta) + \underbrace{\varepsilon(s', a')}_{\text{approximation error or luck}}) \geqslant \max_{a'} Q^*(s', a')$$

Overestimation Bias and how to fight it

$$\mathbb{E}_{\varepsilon} \max_{a'}(Q^*(s', a', \theta) + \underbrace{\varepsilon(s', a')}_{\text{approximation error or luck}}) \geqslant \max_{a'} Q^*(s', a')$$

Decouple **action selection** and **action evaluation**:

$$\max_{a'} Q^*(s', a') = \overbrace{Q^*(s', \underbrace{\operatorname{argmax}_{a'} Q^*(s', a')}_{\text{action selection}})}^{\text{action evaluation}}$$

Overestimation Bias and how to fight it

$$\mathbb{E}_{\varepsilon} \max_{a'}(Q^*(s', a', \theta) + \underbrace{\varepsilon(s', a')}_{\text{approximation error or luck}}) \geqslant \max_{a'} Q^*(s', a')$$

Decouple **action selection** and **action evaluation**:

$$\max_{a'} Q^*(s', a') = \overbrace{Q^*(s', \underbrace{\operatorname{argmax}_{a'} Q^*(s', a')}_{\text{action selection}})}^{\text{action evaluation}} \approx Q_1^*(s', \operatorname*{argmax}_{a'} Q_2^*(s', a'))$$

Action Evaluation: options

Name	Networks	Targets
Double Q-learning not really	Q_1^* Q_2^*	$y_1 = r + \gamma Q_?^*(s', \underset{a'}{\operatorname{argmax}}\, Q_?^*(s', a'))$

Action Evaluation: options

Name	Networks	Targets
Double Q-learning not really	Q_1^* Q_2^*	$y_1 = r + \gamma Q_2^*(s', \underset{a'}{\operatorname{argmax}}\, Q_1^*(s', a'))$

Action Evaluation: options

Name	Networks	Targets
Double Q-learning not really	Q_1^* Q_2^*	$y_1 = r + \gamma Q_2^*(s', \operatorname{argmax}_{a'} Q_1^*(s', a'))$ $y_2 = r + \gamma Q_1^*(s', \operatorname{argmax}_{a'} Q_2^*(s', a'))$

Action Evaluation: options

Name	Networks	Targets
~~Double Q-learning~~ Twin DQN still not really	Q_1^* Q_2^*	$y_1 = r + \gamma Q_2^*(s', \operatorname{argmax}_{a'} Q_1^*(s', a'))$ $y_2 = r + \gamma Q_1^*(s', \operatorname{argmax}_{a'} Q_2^*(s', a'))$
Double DQN	Q^* Q_-^* — target network	

Action Evaluation: options

Name	Networks	Targets
~~Double Q-learning~~ Twin DQN still not really	Q_1^* Q_2^*	$y_1 = r + \gamma Q_2^*(s', \operatorname{argmax}_{a'} Q_1^*(s', a'))$ $y_2 = r + \gamma Q_1^*(s', \operatorname{argmax}_{a'} Q_2^*(s', a'))$
Double DQN	Q^* Q_-^* — target network	$y = r + \gamma Q_-^*(s', \operatorname{argmax}_{a'} Q^*(s', a'))$

Action Evaluation: options

Name	Networks	Targets
~~Double Q-learning~~ ~~Twin DQN~~ ?!?	Q_1^* Q_2^*	$y_1 = r + \gamma Q_2^*(s', \operatorname{argmax}_{a'} Q_1^*(s', a'))$ $y_2 = r + \gamma Q_1^*(s', \operatorname{argmax}_{a'} Q_2^*(s', a'))$
Double DQN	Q^* Q_-^* — target network	$y = r + \gamma Q_-^*(s', \operatorname{argmax}_{a'} Q^*(s', a'))$
Twin DQN («Clipped Double DQN»)	Q_1^* Q_2^*	

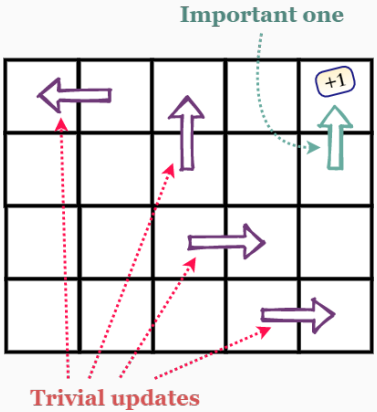
Action Evaluation: options

Name	Networks	Targets
~~Double Q-learning~~ ~~Twin DQN~~ ?!?	Q_1^* Q_2^*	$y_1 = r + \gamma Q_2^*(s', \operatorname{argmax}_{a'} Q_1^*(s', a'))$ $y_2 = r + \gamma Q_1^*(s', \operatorname{argmax}_{a'} Q_2^*(s', a'))$
Double DQN	Q^* Q_-^* — target network	$y = r + \gamma Q_-^*(s', \operatorname{argmax}_{a'} Q^*(s', a'))$
Twin DQN («Clipped Double DQN»)	Q_1^* Q_2^*	$y_1 = r + \gamma \min_{i=1,2} Q_i^*(s', \operatorname{argmax}_{a'} Q_1^*(s', a'))$

Action Evaluation: options

Name	Networks	Targets
~~Double Q-learning~~ ~~Twin DQN~~ ?!?	Q_1^* Q_2^*	$y_1 = r + \gamma Q_2^*(s', \operatorname{argmax}_{a'} Q_1^*(s', a'))$ $y_2 = r + \gamma Q_1^*(s', \operatorname{argmax}_{a'} Q_2^*(s', a'))$
Double DQN	Q^* Q_-^* — target network	$y = r + \gamma Q_-^*(s', \operatorname{argmax}_{a'} Q^*(s', a'))$
Twin DQN («Clipped Double DQN»)	Q_1^* Q_2^*	$y_1 = r + \gamma \min_{i=1,2} Q_i^*(s', \operatorname{argmax}_{a'} Q_1^*(s', a'))$ $y_2 = r + \gamma \min_{i=1,2} Q_i^*(s', \operatorname{argmax}_{a'} Q_2^*(s', a'))$

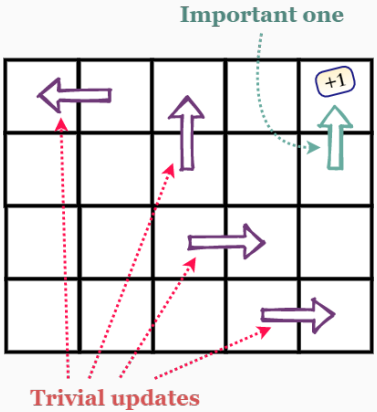
Prioritized Experience Replay



Problem: a lot of trivial updates.

Goal: propagate reinforcement from future to past

Prioritized Experience Replay



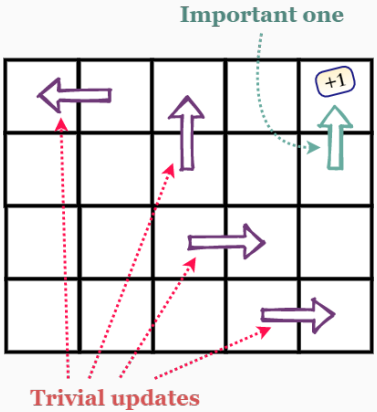
Problem: a lot of trivial updates.

Goal: propagate reinforcement from future to past

Prioritized sampling from replay buffer:

$$P(\mathbb{T}) \propto |y(\mathbb{T}) - Q^*(s, a, \theta)|$$

Prioritized Experience Replay



Problem: a lot of trivial updates.

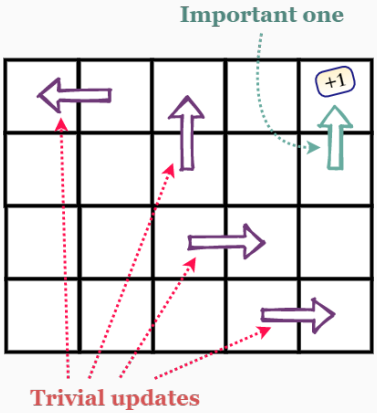
Goal: propagate reinforcement from future to past

Prioritized sampling from replay buffer:

$$P(\mathbb{T}) \propto |y(\mathbb{T}) - Q^*(s, a, \theta)|$$

Issues?

Prioritized Experience Replay



Problem: a lot of trivial updates.

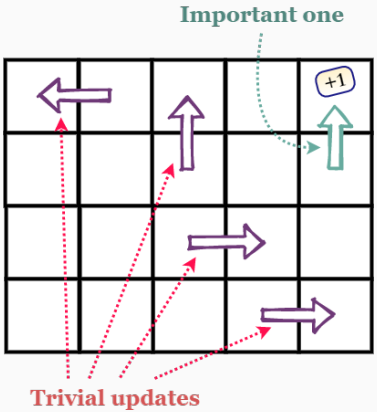
Goal: propagate reinforcement from future to past

Prioritized sampling from replay buffer:

$$\mathrm{P}(\mathbb{T}) \propto |y(\mathbb{T}) - Q^*(s, a, \theta)|$$

Issues? $s' \not\sim p(s' \mid s, a)$ now.

Prioritized Experience Replay



Problem: a lot of trivial updates.

Goal: propagate reinforcement from future to past

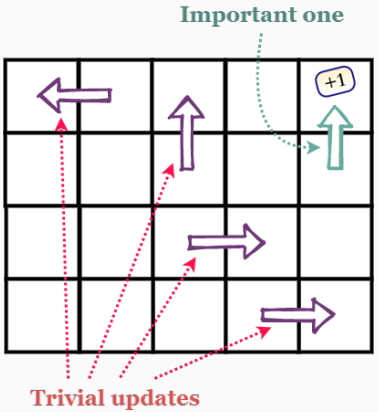
Prioritized sampling from replay buffer:

$$\mathrm{P}(\mathbb{T}) \propto |y(\mathbb{T}) - Q^*(s, a, \theta)|$$

Issues? $s' \not\sim p(s' \mid s, a)$ now.

$$\mathbb{E}_{\mathbb{T} \sim \mathrm{Uniform}} \mathrm{Loss}(\mathbb{T}) \propto$$

Prioritized Experience Replay



Problem: a lot of trivial updates.

Goal: propagate reinforcement from future to past

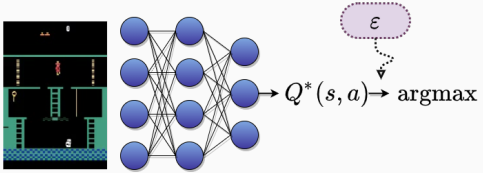
Prioritized sampling from replay buffer:

$$\mathrm{P}(\mathbb{T}) \propto |y(\mathbb{T}) - Q^*(s, a, \theta)|$$

Issues? $s' \not\sim p(s' \mid s, a)$ now.

$$\mathbb{E}_{\mathbb{T} \sim \mathrm{Uniform}} \mathrm{Loss}(\mathbb{T}) \propto \mathbb{E}_{\mathbb{T} \sim \mathrm{P}(\mathbb{T})} \underbrace{\frac{1}{\mathrm{P}(\mathbb{T})}}_{\text{weight}} \mathrm{Loss}(\mathbb{T})$$

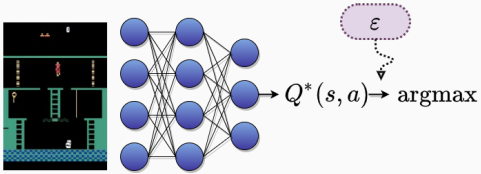
Noisy Networks



Problem: ε-greedy is naive

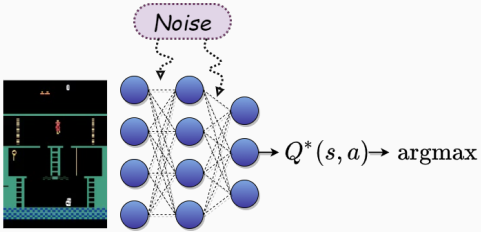
- × inconvenient hyperparameter
- × state-independent exploration

Noisy Networks



Problem: ε-greedy is naive

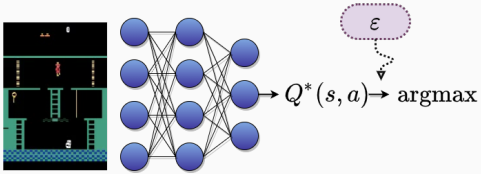
- × inconvenient hyperparameter
- × state-independent exploration

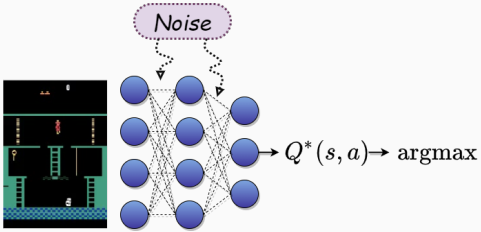


Add noise to the network weights

$$w := \mu + \sigma\varepsilon, \quad \varepsilon \sim \mathcal{N}(0,1)$$

Noisy Networks





Problem: ε-greedy is naive

- × inconvenient hyperparameter
- × state-independent exploration

Add noise to the network weights

$$w := \mu + \sigma\varepsilon, \quad \varepsilon \sim \mathcal{N}(0,1)$$

- ✓ no hyperparameters (initialization of σ?)
- ✓ state-dependent (not interpretable though)
- × expensive in wall-clock time

More modifications!

Dueling DQN

Problem: state value estimation

More modifications!

Dueling DQN

Problem: state value estimation

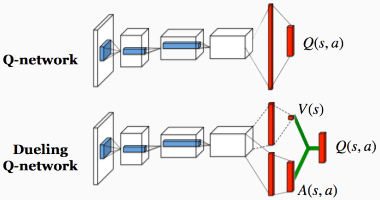
$$Q^*(s,a) := V^*(s) + \underbrace{A^*(s,a)}_{\text{not arbitrary}}$$

More modifications!

Dueling DQN

Problem: state value estimation

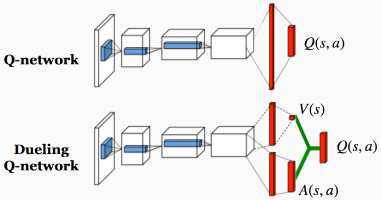
$$Q^*(s,a) := V^*(s) + \underbrace{A^*(s,a)}_{\text{not arbitrary}}$$



Dueling DQN

Problem: state value estimation

$$Q^*(s,a) := V^*(s) + \underbrace{A^*(s,a)}_{\text{not arbitrary}}$$



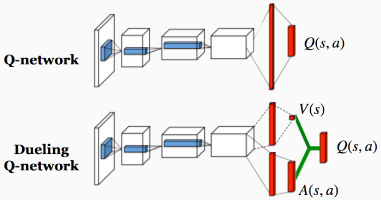
$$Q^*_\theta(s,a) := V^*_\theta(s) + A^*_\theta(s,a) - \underbrace{- \operatorname{mean}_a A^*_\theta(s,a)}_{\text{theory: take max}}$$

More modifications!

Dueling DQN

Problem: state value estimation

$$Q^*(s,a) := V^*(s) + \underbrace{A^*(s,a)}_{\text{not arbitrary}}$$



$$Q^*_\theta(s,a) := V^*_\theta(s) + A^*_\theta(s,a) - \underbrace{- \operatorname{mean}_a A^*_\theta(s,a)}_{\text{theory: take max}}$$

Partial Observability

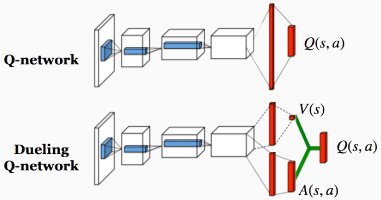
Problem: no access to full state description

More modifications!

Dueling DQN

Problem: state value estimation

$$Q^*(s,a) := V^*(s) + \underbrace{A^*(s,a)}_{\text{not arbitrary}}$$



$$Q^*_\theta(s,a) := V^*_\theta(s) + A^*_\theta(s,a) - \underbrace{- \operatorname{mean}_a A^*_\theta(s,a)}_{\text{theory: take max}}$$

Partial Observability

Problem: no access to full state description

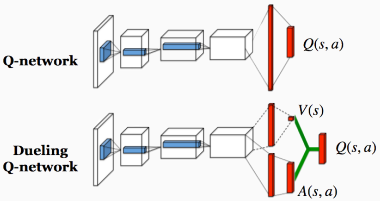
Theory: see PoMDP; **Practice:** LSTM

More modifications!

Dueling DQN

Problem: state value estimation

$$Q^*(s,a) := V^*(s) + \underbrace{A^*(s,a)}_{\text{not arbitrary}}$$

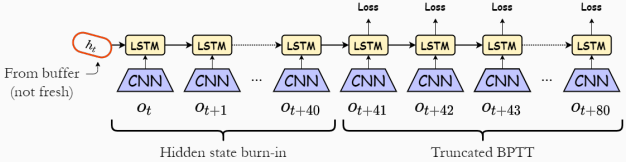


$$Q^*_\theta(s,a) := V^*_\theta(s) + A^*_\theta(s,a) - \underbrace{- \operatorname{mean}_a A^*_\theta(s,a)}_{\text{theory: take max}}$$

Partial Observability

Problem: no access to full state description

Theory: see PoMDP; **Practice:** LSTM

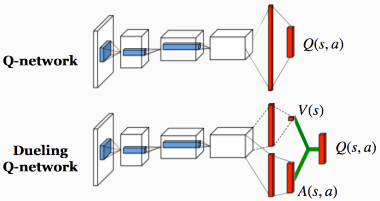


More modifications!

Dueling DQN

Problem: state value estimation

$$Q^*(s,a) := V^*(s) + \underbrace{A^*(s,a)}_{\text{not arbitrary}}$$

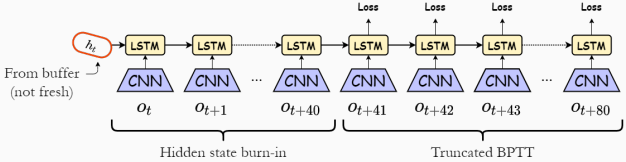


$$Q^*_\theta(s,a) := V^*_\theta(s) + A^*_\theta(s,a) - \underbrace{- \operatorname{mean}_a A^*_\theta(s,a)}_{\text{theory: take max}}$$

Partial Observability

Problem: no access to full state description

Theory: see PoMDP; **Practice:** LSTM



Multi-step DQN

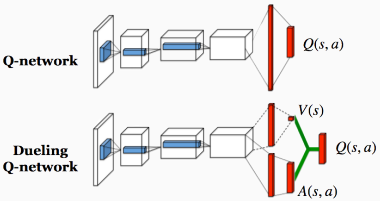
Problem: delayed rewards + compounding error

More modifications!

Dueling DQN

Problem: state value estimation

$$Q^*(s,a) := V^*(s) + \underbrace{A^*(s,a)}_{\text{not arbitrary}}$$

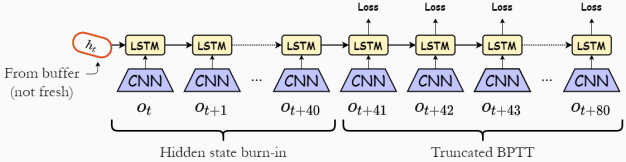


$$Q^*_\theta(s,a) := V^*_\theta(s) + A^*_\theta(s,a) - \underbrace{- \operatorname{mean}_a A^*_\theta(s,a)}_{\text{theory: take max}}$$

Partial Observability

Problem: no access to full state description

Theory: see PoMDP; **Practice:** LSTM



Multi-step DQN

Problem: delayed rewards + compounding error

$$y := \underbrace{r + \gamma r' + \gamma^2 r'' + \dots}_{\approx \text{«optimal behavior»?}} + \gamma^N \max_{a^{(N)}} Q^*(s^{(N)}, a^{(N)})$$

Frontiers

Rainbow DQN
(2017)

- Double DQN
- Prioritized buffer
- Multi-step DQN
- Noisy Nets
- Dueling DQN
- *Distributional RL*
  (next time)

Frontiers

Rainbow DQN
(2017)

- Double DQN
- Prioritized buffer
- Multi-step DQN
  - Noisy Nets
  - Dueling DQN
- *Distributional RL*
  (next time)

R2D2
(2018)

- Double DQN
- Prioritized buffer
- Multi-step DQN

\+ LSTM

\+ Massive parallelization

Frontiers

Rainbow DQN
(2017)

- Double DQN
- Prioritized buffer
- Multi-step DQN
- Noisy Nets
- Dueling DQN
- *Distributional RL*
  (next time)

R2D2
(2018)

- Double DQN
- Prioritized buffer
- Multi-step DQN

\+ LSTM

\+ Massive parallelization

Agent 57
(2020)

- Double DQN
- Prioritized buffer
- Multi-step DQN
- LSTM
- Massive parallelization

\+ Retrace

\+ Intrinsic motivation

\+ Meta-controller for hyperparameters

Literature

- Playing Atari with Deep Reinforcement Learning (2013);
- *For DQN modifications see links on previous slide*;

  Pictures from Berkeley CS 188 | Introduction to Artificial Intelligence;