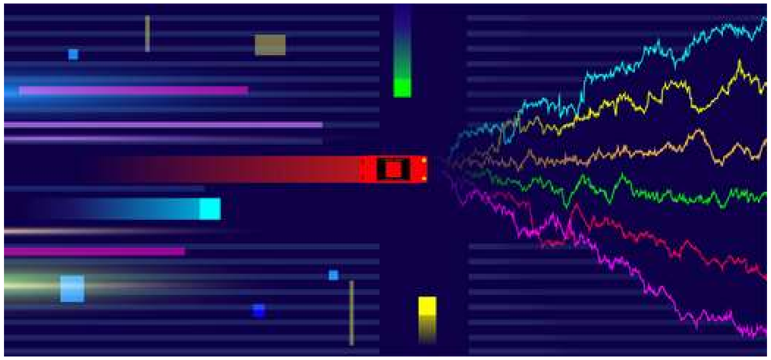
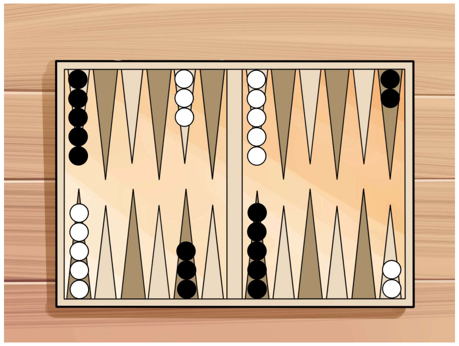
RL for Operations, 2022

Introduction to Deep RL

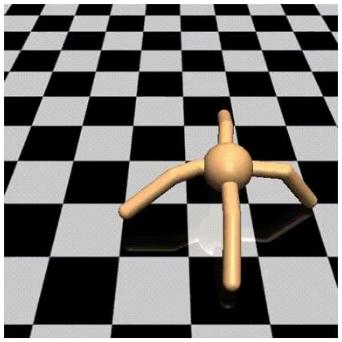
Sean Sinclair
Cornell University



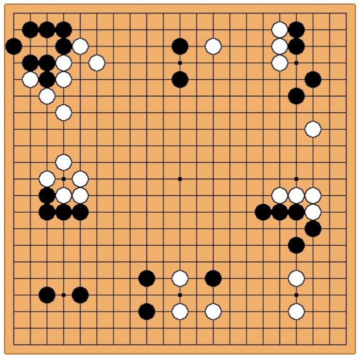
Success of RL



Backgammon



Mujoco Simulator



AlphaGo Zero

Focused on game playing + robotics

[Silver2017,Tesauro1995]

So far:

Previously saw algorithms designed with value and policy iteration for tabular (discrete) MDPs.

Model-Based

- Maintain estimates of reward and transition
- Plug estimates into Bellman equations for estimated V^*, Q^*
- Play greedy w.r.t. Q^*
- Time complexity / storage scales S^2A

Model Free

- Only maintain estimates of V^*, Q^* using fixed point
- Play greedy w.r.t. Q^*
- Better time complexity / storage

So far:

Previously saw algorithms designed with value and policy iteration for tabular (discrete) MDPs.

However, even if problem is tabular:

```
MemoryError: Unable to allocate 31.9 GiB for an array with shape (3094, 720, 1280, 3) and data type float32
```

H = 3
S = 3094*720
A = 1280

So far:

Previously saw algorithms designed with value and policy iteration for tabular (discrete) MDPs.

However, even if problem is tabular:

```
MemoryError: Unable to allocate 31.9 GiB for an array with shape (3094, 720, 1280, 3) and data type float32
```

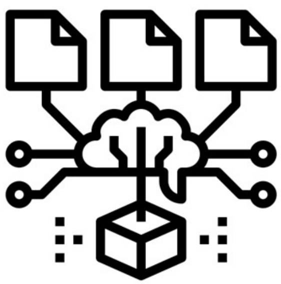
H = 3
S = 3094*720
A = 1280

function approximation + state aggregation help alleviate these issues by appealing to ML generalization

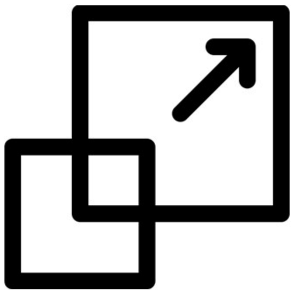
So far:

This workshop focuses on RL for Operations

We care about:



OR Models



Computation + Scale

Impact

Generalization

Want to use RL in large-scale OR problems

Generalization

Want to use RL in large-scale OR problems

- Most problems have enormous state/action spaces

Generalization

Want to use RL in large-scale OR problems

- Most problems have enormous state/action spaces
- Require representations (of models, state-action values, values, policies) that can generalize

Generalization

Want to use RL in large-scale OR problems

- Most problems have enormous state/action spaces
- Require representations (of models, state-action values, values, policies) that can generalize
- Represent quantities with a parameterized function instead of a table

Generalization

Want to use RL in large-scale OR problems

- Most problems have enormous state/action spaces
- Require representations (of models, state-action values, values, policies) that can generalize
- Represent quantities with a parameterized function instead of a table

$$s \rightarrow \boxed{\quad\theta\quad} \rightarrow \overline{V}_\theta(s) \qquad\qquad s \rightarrow \boxed{\quad\theta\quad} \rightarrow \overline{\pi}_\theta(s)$$

Generalization

- Well developed theory for linear value function estimators for known features
- Works well *if* features are right

Generalization

- Well developed theory for linear value function estimators for known features
- Works well *if* features are right

- Local representations (Kernel + discretization approaches) have nice properties (convergence results) but don't scale well

Generalization

- Well developed theory for linear value function estimators for known features
- Works well *if* features are right

- Local representations (Kernel + discretization approaches) have nice properties (convergence results) but don't scale well

- Alternative: use deep neural networks:
  - uses distributed representation instead of local
  - "universal function approximator"

Generalization

- Well developed theory for linear value function estimators for known features
- Works well *if* features are right

- Local representations (Kernel + discretization approaches) have nice properties (convergence results) but don't scale well

- Alternative: use deep neural networks:
  - uses distributed representation instead of local
  - "universal function approximator"

$$s \longrightarrow \boxed{\theta} \longrightarrow \overline{V}_\theta(s)$$

Generalization

Alternative: use deep neural networks:

- uses distributed representation instead of local
- "universal function approximator"

$$s \longrightarrow \boxed{\quad \theta \quad} \longrightarrow \overline{V}_\theta(s)$$

Generalization

Alternative: use deep neural networks:

- uses distributed representation instead of local
- "universal function approximator"

$$s \longrightarrow \boxed{\quad \theta \quad} \longrightarrow \overline{V}_\theta(s)$$

We will not focus on network representation choices, etc, until days 3+4. Consider arbitrary "parametric" representation (with gradients) for now

High level approach

Use a deep neural network to represent:

- Value function
- Policy
- Model of rewards and transitions
- “universal function approximator”

Optimize loss function via stochastic gradient descent (SGD)

High level approach

Use a deep neural network to represent:

- Value function
- Policy
- Model of rewards and transitions
- "universal function approximator"

Optimize loss function via stochastic gradient descent (SGD)

DQN

- Use neural network representation of value function
- Fits estimates using least squares

REINFORCE / VPG

- Uses neural network representation of policy
- Fits estimates using policy gradient loss

High level approach

Use a deep neural network to represent:

- Value function
- Policy
- Model of rewards and transitions
- “universal function approximator”

Optimize loss function via stochastic gradient descent (SGD)

DQN

- Use neural network representation of value function
- Fits estimates using least squares

REINFORCE / VPG

- Uses neural network representation of policy
- Fits estimates using policy gradient loss

DQN

Use a deep neural network to represent:

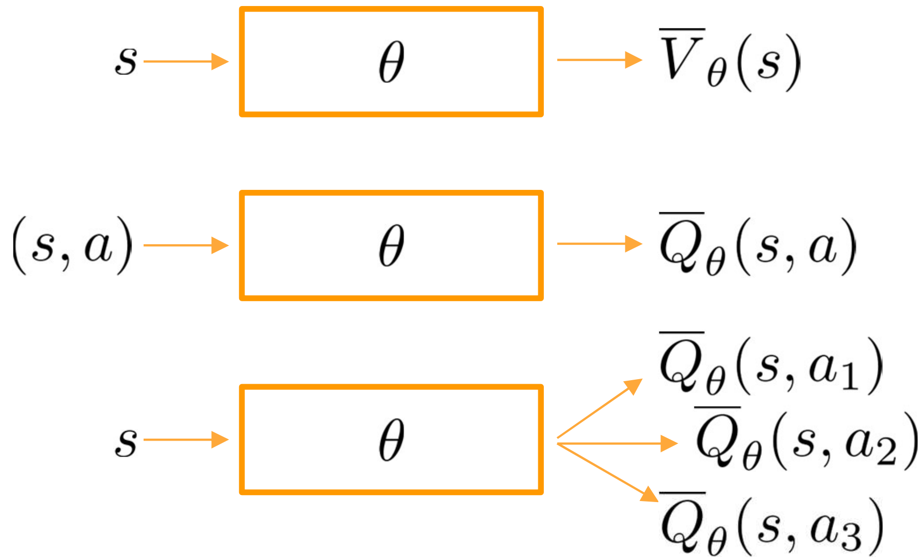
- Value function
- Policy
- Model of rewards and transitions
- "universal function approximator"

Optimize loss function via stochastic gradient descent (SGD)

[Mnih2013]

DQN

Use a deep neural network to represent value function:



Action dependent features

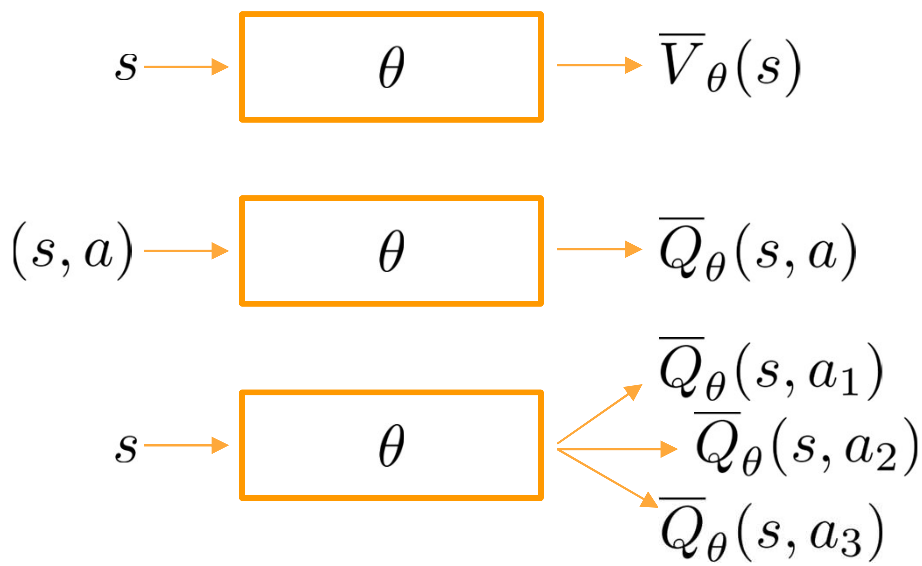
State dependent features

Question: Which representation is better?

[Mnih2013]

DQN

Use a deep neural network to represent value function:



Action dependent features

State dependent features

Question: Which representation is better?

$$\pi_\theta(s) = \max_a \overline{Q}_\theta(s, a)$$

[Mnih2013]

Recall:

The Bellman Optimality Equations note that:

$$V^*(s) = \max_a [r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[V^*(S')]]$$
$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[V^*(S')]$$

[Mnih2013]

Recall:

The Bellman Optimality Equations note that:

$$V^*(s) = \max_a [r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[V^*(S')]]$$

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[V^*(S')]$$

Or using that $V^*(s) = \max_a Q^*(s,a)$

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} Q^*(S',a')]$$

[Mnih2013]

Recall:

The Bellman Optimality Equations note that:

$$V^*(s) = \max_a [r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[V^*(S')]]$$
$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[V^*(S')]$$

Or using that $V^*(s) = \max_a Q^*(s,a)$

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} Q^*(S',a')]$$

Subtract off and plug in parametric estimates

$$0 = \overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \overline{Q}_\theta(S',a')])$$

[Mnih2013]

Recall:

Subtract off and plug in parametric estimates

$$0 = \overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} Q^*(S',a')])$$

Which objective function? Minimize MSE

$$L(\theta) = \mathbb{E}_{\pi_\theta}[(\overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \overline{Q}_\theta(S',a')])^2]$$

[Mnih2013]

Recall:

Subtract off and plug in parametric estimates

$$0 = \overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} Q^*(S',a')])$$

Which objective function? Minimize MSE

$$L(\theta) = \mathbb{E}_{\pi_\theta}[(\overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \overline{Q}_\theta(S',a')])^2]$$

"Fixed Target"

[Mnih2013]

Recall:

Subtract off and plug in parametric estimates

$$0 = \overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} Q^*(S',a')])$$

Which objective function? Minimize MSE

$$L(\theta) = \mathbb{E}_{\pi_\theta}[(\overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \overline{Q}_\theta(S',a')])^2]$$

"Fixed Target"

Note: If $\mathbf{L(\theta) = 0}$ then satisfy fixed point, so optimal

[Mnih2013]

Why MSE?

Which objective function? Minimize MSE

$$L(\theta) = \mathbb{E}_{\pi_\theta}[(\overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \overline{Q}_\theta(S',a')])^2]$$

[Mnih2013]

Why MSE?

Which objective function? Minimize MSE

$$L(\theta) = \mathbb{E}_{\pi_\theta}[(\overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \boxed{\overline{Q}_\theta(S',a')}])^2]$$

Assume tabular representation, “fixed target”, single sample from T

[Mnih2013]

Why MSE?

Which objective function? Minimize MSE

$$L(\theta) = \mathbb{E}_{\pi_\theta}[(\overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \boxed{\overline{Q}_\theta(S',a')}])^2]$$

Assume tabular representation, “fixed target”, single sample from T

$$L(\theta) = \mathbb{E}_{\pi_\theta}[(\theta(s,a) - (r(s,a) + \gamma \max_{a'} \overline{Q}_\theta(s',a')])^2]$$

[Mnih2013]

Why MSE?

Which objective function? Minimize MSE

$$L(\theta) = \mathbb{E}_{\pi_\theta}[(\overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \boxed{\overline{Q}_\theta(S',a')}])^2]$$

Assume tabular representation, “fixed target”, single sample from *T*

$$L(\theta) = \mathbb{E}_{\pi_\theta}[(\theta(s,a) - (r(s,a) + \gamma \max_{a'} \overline{Q}_\theta(s',a')])^2]$$

Using gradient descent then….

$$\begin{aligned}
\theta(s,a) &= \theta(s,a) - \alpha \nabla_{s,a} J(\theta) \\
&= \theta(s,a) - \alpha \left(r(s,a) + \gamma \max_{a'} \overline{Q}(s',a') - \theta(s,a) \right) (-1) \\
&= (1-\alpha)\theta(s,a) + \alpha (r(s,a) + \gamma \max_{a'} \overline{Q}(s',a'))
\end{aligned}$$

[Mnih2013]

Why MSE?

Which objective function? Minimize MSE

$$L(\theta) = \mathbb{E}_{\pi_\theta}[(\overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \boxed{\overline{Q}_\theta(S',a')}])^2]$$

Assume tabular representation, "fixed target", single sample from *T*

$$L(\theta) = \mathbb{E}_{\pi_\theta}[(\theta(s,a) - (r(s,a) + \gamma \max_{a'} \overline{Q}_\theta(s',a')])^2]$$

Using gradient descent then....

$$\begin{aligned}
\theta(s,a) &= \theta(s,a) - \alpha \nabla_{s,a} J(\theta) \\
&= \theta(s,a) - \alpha \left(r(s,a) + \gamma \max_{a'} \overline{Q}(s',a') - \theta(s,a) \right) (-1) \\
&= (1-\alpha)\theta(s,a) + \alpha(r(s,a) + \gamma \max_{a'} \overline{Q}(s',a'))
\end{aligned}$$

Recovers standard Q-Learning rules from earlier

[Mnih2013]

Nonstationary Target

Typically in supervised learning the “ground truth” is fixed

( ,  , 14 mins)

( ,  , 8 mins)

( ,  , 24 mins)

Nonstationary Target

Typically in supervised learning the “ground truth” is fixed

( ,  , 14 mins)

( ,  , 8 mins)

( ,  , 24 mins)

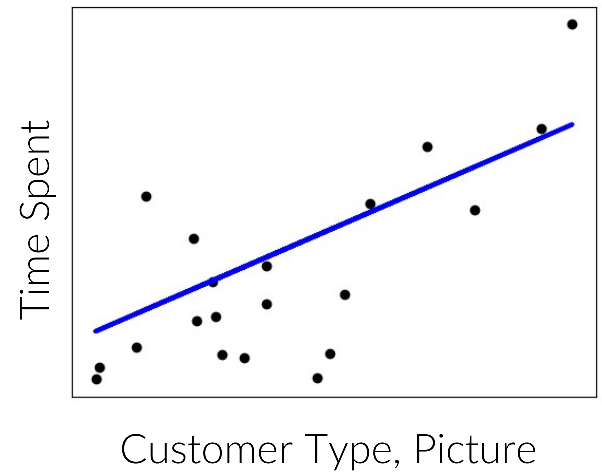
Nonstationary Target

Typically in supervised learning the “ground truth” is fixed

( ,  , 14 mins)

( ,  , 8 mins)

( ,  , 24 mins)

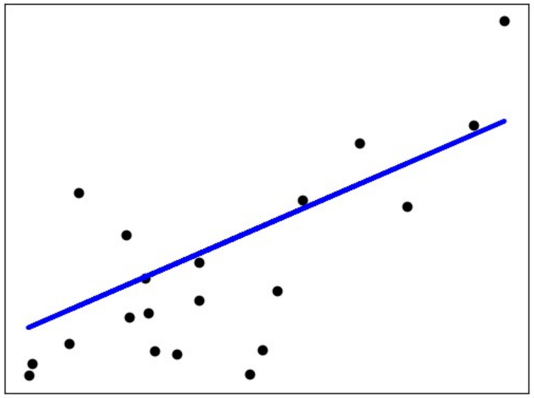


Nonstationary Target

Typically in supervised learning the "ground truth" is fixed

Prediction

Data Distribution



$$L(\theta) = \mathbb{E}_\xi[(f(\xi) - \overline{Q}_\theta(\xi))^2]$$

No impact by network parameters

Nonstationary Target

Nonstationary Target

$$L^1(\theta) = \mathbb{E}_{\pi_\theta}[(\overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \overline{Q}_{\theta^1}(S',a')])^2]$$

SGD Step for $\boldsymbol{\theta^2}$

Nonstationary Target

$$L^1(\theta) = \mathbb{E}_{\pi_\theta}[(\overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \overline{Q}_{\theta^1}(S',a')])^2]$$

SGD Step for $\boldsymbol{\theta}^2$

$$L^2(\theta) = \mathbb{E}_{\pi_\theta}[(\overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \overline{Q}_{\theta^2}(S',a')])^2]$$

SGD Step for $\boldsymbol{\theta}^3$

Nonstationary Target

$$L^1(\theta) = \mathbb{E}_{\pi_\theta}[(\overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \overline{Q}_{\theta^1}(S',a')])^2]$$

SGD Step for $\boldsymbol{\theta}^2$

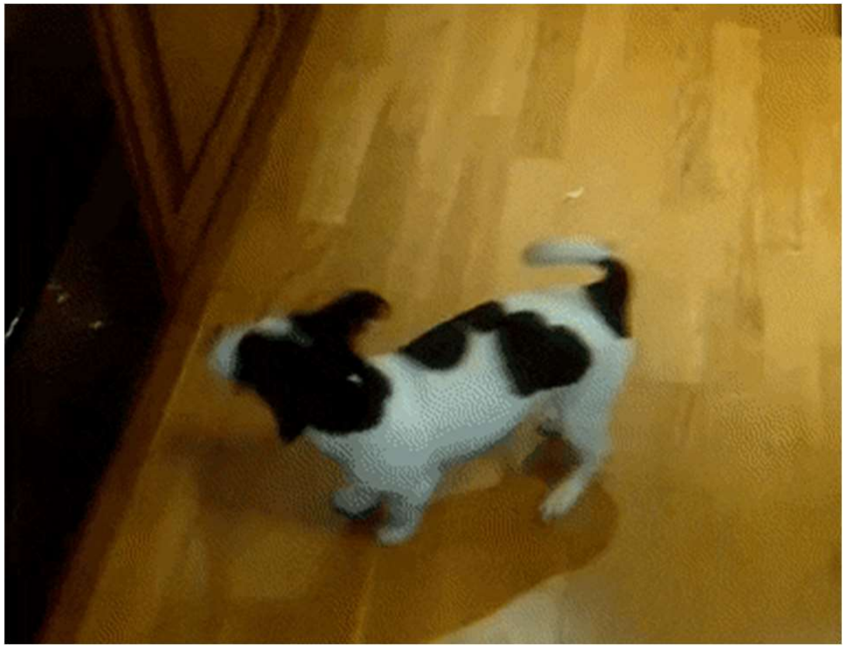
$$L^2(\theta) = \mathbb{E}_{\pi_\theta}[(\overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \overline{Q}_{\theta^2}(S',a')])^2]$$

SGD Step for $\boldsymbol{\theta}^3$

$$L^3(\theta) = \mathbb{E}_{\pi_\theta}[(\overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \overline{Q}_{\theta^3}(S',a')])^2]$$

SGD Step for $\boldsymbol{\theta}^4$

Nonstationary Target



Like a dog chasing its own tail...

DQN

Converges to true Q^* using tabular representation

Diverges in neural networks due to:

- spurious correlation in samples
- "nonstationary target"

$$L(\theta) = \mathbb{E}_{\pi_\theta}[(\overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \overline{Q}_\theta(S',a')])^2]$$

[Mnih2013]

DQN

Converges to true Q^* using tabular representation

Diverges in neural networks due to:

- spurious correlation in samples
- nonstationary target

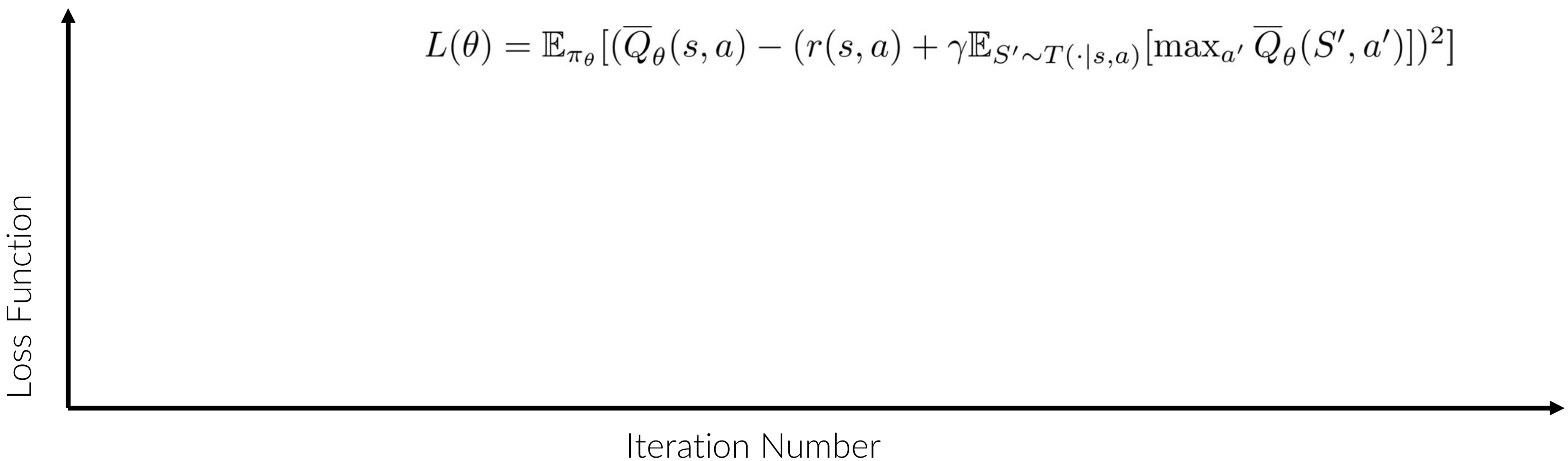
$$L(\theta) = \mathbb{E}_{\pi_\theta}[(\overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \overline{Q}_\theta(S',a')])^2]$$

DQN

Converges to true Q^* using tabular representation

Diverges in neural networks due to:
- spurious correlation in samples
- nonstationary target

$$L(\theta) = \mathbb{E}_{\pi_\theta}[(\overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \overline{Q}_\theta(S',a')])^2]$$

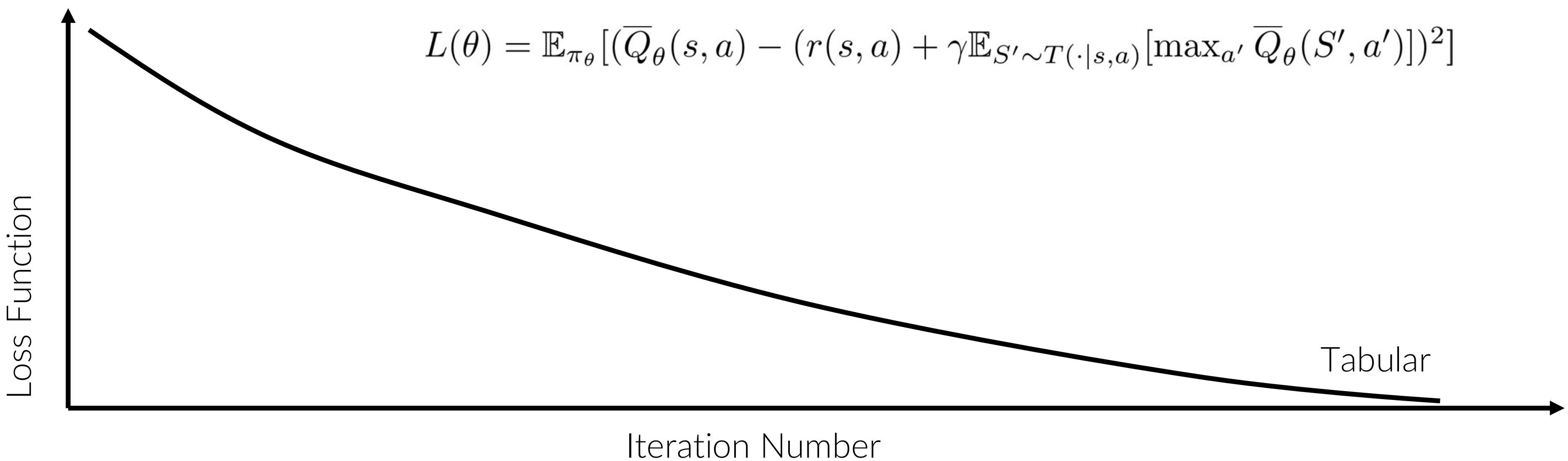


DQN

Converges to true Q^* using tabular representation

Diverges in neural networks due to:
- spurious correlation in samples
- nonstationary target

$$L(\theta) = \mathbb{E}_{\pi_\theta}[(\overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \overline{Q}_\theta(S',a')])^2]$$



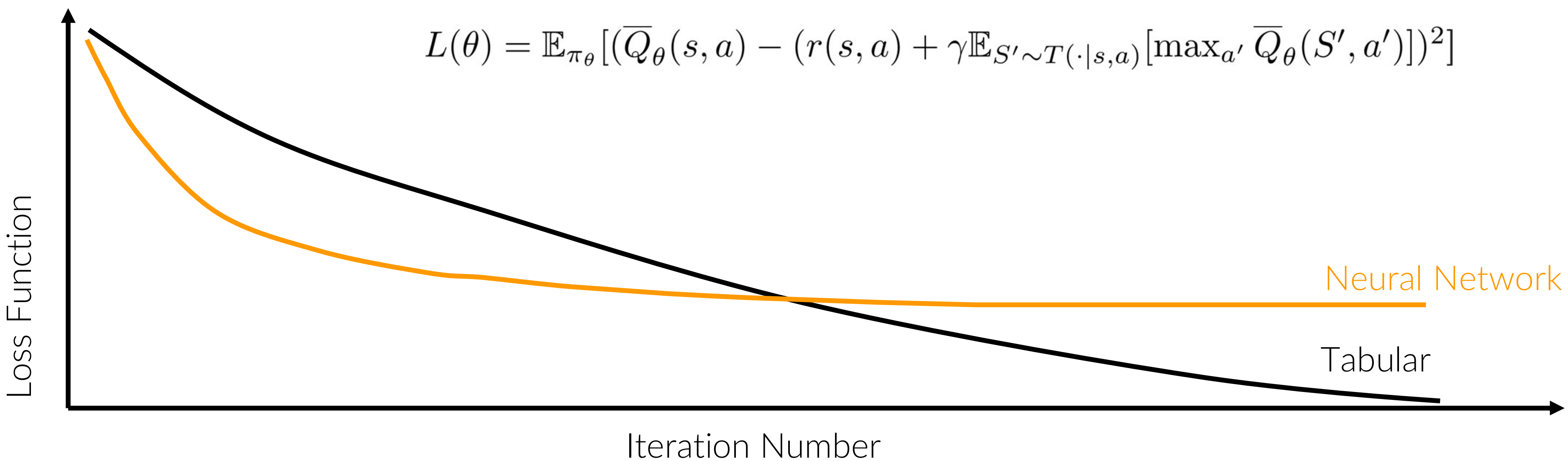
DQN

Converges to true Q^* using tabular representation

Diverges in neural networks due to:

- spurious correlation in samples
- nonstationary target

$$L(\theta) = \mathbb{E}_{\pi_\theta}[(\overline{Q}_\theta(s,a) - (r(s,a) + \gamma \mathbb{E}_{S' \sim T(\cdot|s,a)}[\max_{a'} \overline{Q}_\theta(S',a')])^2]$$



Max vs Expectation

Other issue in practice:

$$L(\theta) = \frac{1}{N} \sum_i (\overline{Q}_\theta(s_i, a_i) - (r_i + \gamma \max_{a'} \overline{Q}_\theta(s_{i+1}, a'))^2$$

[Hessel2017]

Max vs Expectation

Other issue in practice:

$$L(\theta) = \frac{1}{N}\sum_i(\overline{Q}_\theta(s_i, a_i) - (r_i + \gamma \boxed{\max_{a'} \overline{Q}_\theta(s_{i+1}, a'))^2}$$

[Hessel2017]

Max vs Expectation

Other issue in practice:

$$L(\theta) = \frac{1}{N}\sum_i(\overline{Q}_\theta(s_i, a_i) - (r_i + \gamma \boxed{\max_{a'} \overline{Q}_\theta(s_{i+1}, a'))^2}$$

$\frac{1}{N}\sum_i \max_{a'} \overline{Q}_\theta(s_{i+1}, a')$ is biased for $\max_{a'} \mathbb{E}_{S' \sim T(\cdot|s,a')}[\overline{Q}_\theta(S', a')]$

[Hessel2017]

Max vs Expectation

Other issue in practice:

$$L(\theta) = \frac{1}{N}\sum_i(\overline{Q}_\theta(s_i, a_i) - (r_i + \gamma \boxed{\max_{a'} \overline{Q}_\theta(s_{i+1}, a'))^2}$$

$\frac{1}{N}\sum_i \max_{a'} \overline{Q}_\theta(s_{i+1}, a')$ is biased for $\max_{a'} \mathbb{E}_{S' \sim T(\cdot|s,a')}[\overline{Q}_\theta(S', a')]$

Same value used both to select and evaluate an action, more likely to select overestimated values resulting in overoptimistic estimates

[Hessel2017]

DQN

Different versions of DQN fix this in two ways via:

Experience Replay

- Collects datasets which are recycled when calculating loss
- Eliminates spurious correlation in samples

Fixed Q-Targets

- Fixes the target in the MSE loss
- Many choices of potential fixed targets

[Hessel2017]

Experience Replay

Store dataset (called replay buffer) from prior experience

$$\mathcal{D} \left\{ \begin{array}{|c|} \hline (\boldsymbol{s_1}, \boldsymbol{a_1}, \boldsymbol{r_1}, \boldsymbol{s_2}) \\ \hline (\boldsymbol{s_2}, \boldsymbol{a_2}, \boldsymbol{r_2}, \boldsymbol{s_3}) \\ \hline (\boldsymbol{s_3}, \boldsymbol{a_3}, \boldsymbol{r_3}, \boldsymbol{s_4}) \\ \hline (\boldsymbol{s_4}, \boldsymbol{a_4}, \boldsymbol{r_4}, \boldsymbol{s_5}) \\ \hline \end{array} \right.$$

Sample batch on $\mathcal{D}$ and update loss

$$L(\theta) = \frac{1}{N} \sum_i (\overline{Q}_\theta(s_i, a_i) - (r_i + \gamma \max_{a'} \overline{Q}_\theta(s_{i+1}, a'))^2$$

[Hessel2017]

Experience Replay

Sample batch on $\mathcal{D}$ and update loss

$$L(\theta) = \frac{1}{N} \sum_i (\overline{Q}_\theta(s_i, a_i) - (r_i + \gamma \max_{a'} \overline{Q}_\theta(s_{i+1}, a'))^2$$

- Uniformly at random
- Weighted by loss
- Weighted by policy performance
- ....

[Hessel2017]

Fixed Q Targets

Fix weights $\boldsymbol{\theta}^{-}$ for target:

$$L(\theta) = \frac{1}{N}\sum_i (\overline{Q}_\theta(s_i, a_i) - (r_i + \gamma \max_{a'} \overline{Q}_{\theta^-}(s_{i+1}, a'))^2$$

Compute Q-learning targets with respect to “old” parameters

[Hessel2017]

DQN

1. Use current policy (potentially with ϵ exploration) to collect data

2. Store transitions (s_t, a_t, r_t, s_{t+1}) in replay data $\mathcal{D}$

3. Sample batch of transitions from $\mathcal{D}$

4. Update loss with respect to old parameters using SGD

$$L(\theta) = \frac{1}{N} \sum_i (\overline{Q}_\theta(s_i, a_i) - (r_i + \gamma \max_{a'} \overline{Q}_{\theta^-}(s_{i+1}, a'))^2$$

[Hessel2017]

Double DQN

Same value used both to select and evaluate an action, more likely to select overestimated values resulting in overoptimistic estimates

Fix: Double DQN.

Learn two value functions with different parameters by assigning each data randomly to update one of the two estimates

[Hasselt2015]

Double DQN

Two sets of weights θ_1, θ_2

1. Use current policy π_{θ_1} (potentially with ϵ exploration) to collect data

2. Store transitions (s_t, a_t, r_t, s_{t+1}) in replay data $\mathcal{D}_1, \mathcal{D}_2$

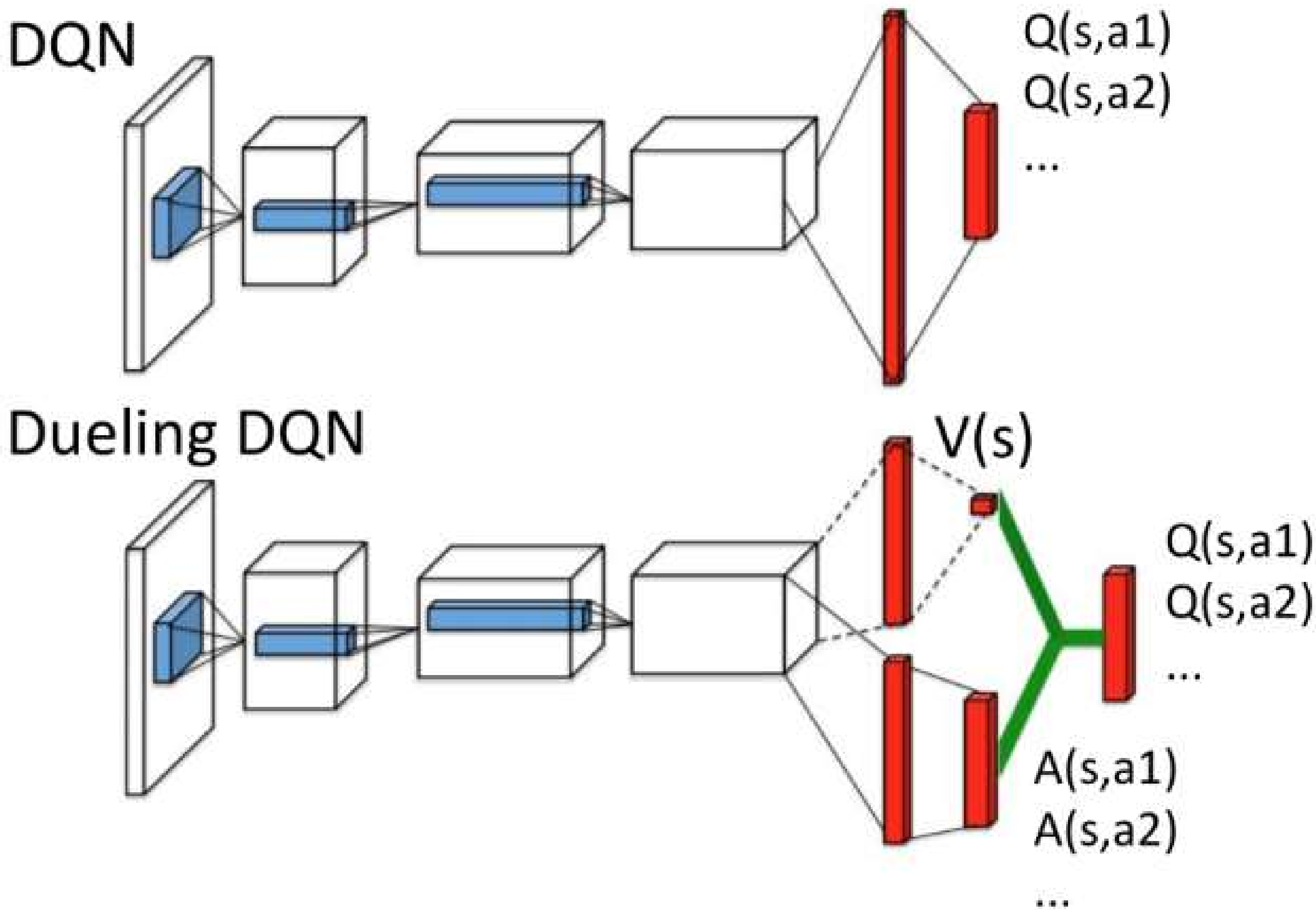
3. Sample batch of transitions from $\mathcal{D}_i$

4. Update loss with respect to opposite parameters using SGD

$$J(\theta) = \frac{1}{N}\sum_i (\overline{Q}_{\theta_1}(s_i, a_i) - (r_i + \gamma \max_{a'} \overline{Q}_{\theta_2}(s_{i+1}, a'))^2$$

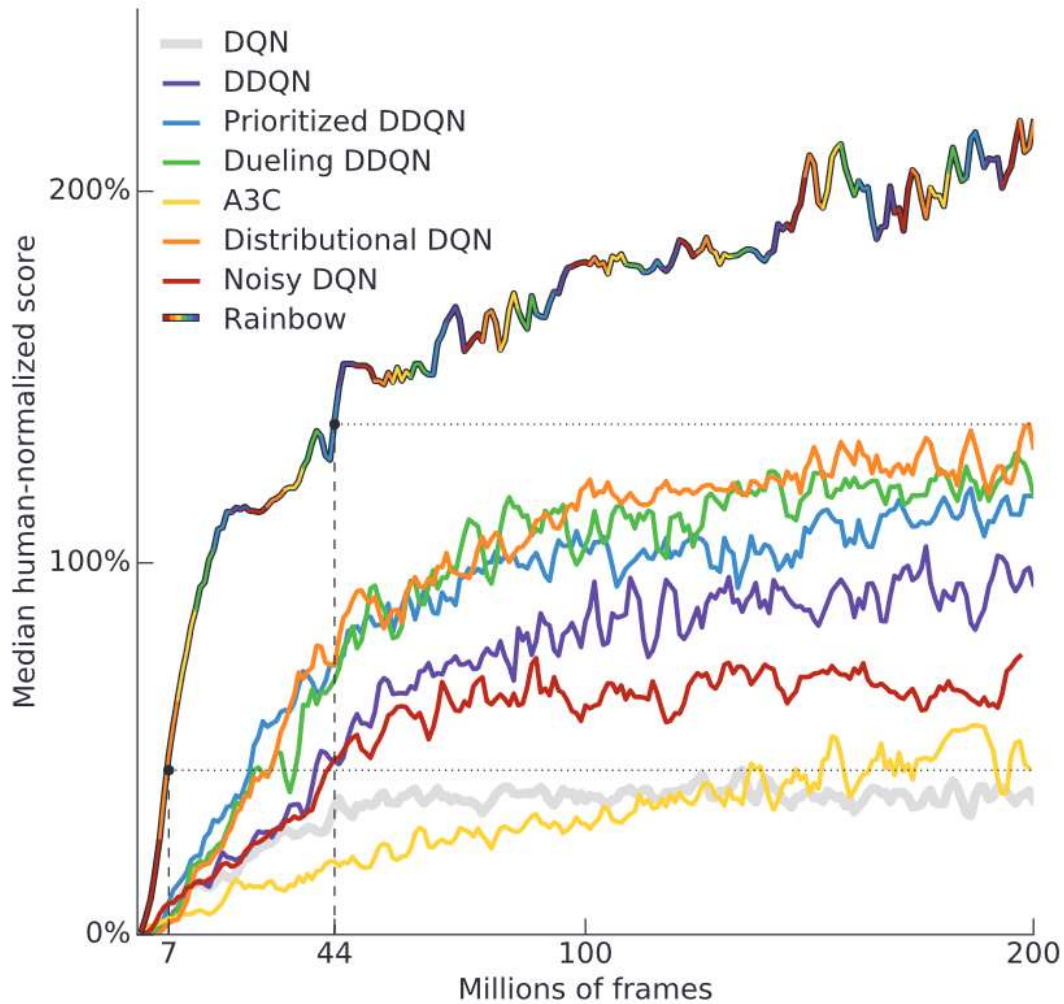
[Hasselt2015]

Dueling DQN



[Wang2015]

DQN++++



[Hessel2017]

High level approach

Use a deep neural network to represent:

- Value function
- Policy
- Model of rewards and transitions
- “universal function approximator”

Optimize loss function via stochastic gradient descent (SGD)

DQN

- Use neural network representation of value function
- Fits estimates using least squares

REINFORCE / VPG

- Uses neural network representation of policy
- Fits estimates using policy gradient loss

High level approach

Focus on REINFORCE – simplest policy based algorithm. Works for on-policy and off-policy data, and in arbitrary domains.

Many flavors and modifications – similar to DQN

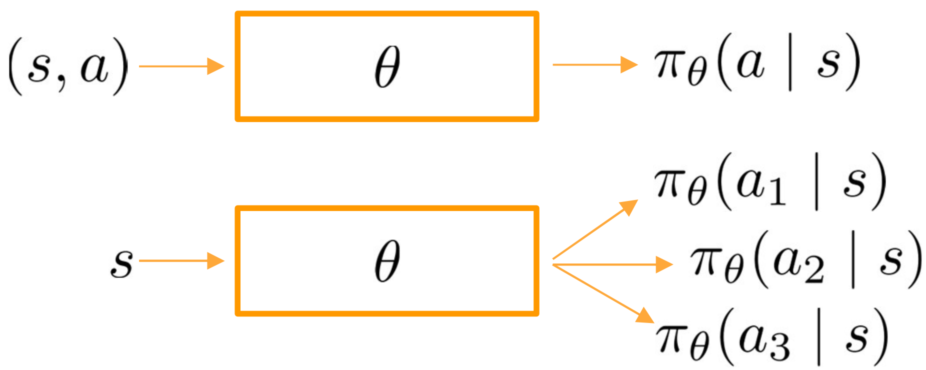
REINFORCE / VPG

- Uses neural network representation of policy
- Fits estimates using policy gradient loss

High level approach

Focus on REINFORCE – simplest policy based algorithm. Works for on-policy and off-policy data, and in arbitrary domains.

Many flavors and modifications – similar to DQN



REINFORCE / VPG

- Uses neural network representation of policy
- Fits estimates using policy gradient loss

High level approach

Focus on REINFORCE – simplest policy based algorithm. Works for on-policy and off-policy data, and in arbitrary domains.

Goal: Maximize $\sup_{\theta \in \Theta} \mathbb{E}\left[V^{\pi_\theta}(s_0)\right]$

High level approach

Focus on REINFORCE – simplest policy based algorithm. Works for on-policy and off-policy data, and in arbitrary domains.

Goal: Maximize $\sup_{\theta \in \Theta} \mathbb{E}\left[V^{\pi_\theta}(s_0)\right] = J(\theta)$

Refer to $J(\theta)$ as objective, can be modified for finite horizon + average cost

High level approach

Goal: Maximize $\sup_{\theta \in \Theta} \mathbb{E}\left[V^{\pi_\theta}(s_0)\right] = J(\theta)$

Any optimization algorithm could be applied:

- Zero-Order (Gradient-Free)
- First-Order (Gradient-Based)
- Second Order (Hessian-Based)

Gradient Free

Goal: Maximize $\sup_{\theta \in \Theta} \mathbb{E}\left[V^{\pi_\theta}(s_0)\right] = J(\theta)$

Evolutionary Algorithms or Grid Search

For t = 0,1,2,...

Sample: $\epsilon_1, \ldots, \epsilon_N \sim N(0, I)$

Compute Returns: $F_i = J(\theta_t + \sigma \epsilon_i)$

Update: $\theta_{t+1} = \theta_t + \alpha \frac{1}{N\sigma} \sum_{i=1}^{N} F_i \epsilon_i$

[Vikhar2016]

Gradient Free

Goal: Maximize $\sup_{\theta \in \Theta} \mathbb{E}\left[V^{\pi_\theta}(s_0)\right] = J(\theta)$

Evolutionary Algorithms or Grid Search

For t = 0,1,2,...

Sample: $\epsilon_1, \ldots, \epsilon_N \sim N(0, I)$

Compute Returns: $F_i = J(\theta_t + \sigma \epsilon_i)$

Update: $\theta_{t+1} = \theta_t + \alpha \frac{1}{N\sigma} \sum_{i=1}^{N} F_i \epsilon_i$

"Scalable" via strong parallelization, difficult for neural networks due to dimension

[Salimans2017]

Policy Gradient

Goal: Maximize $\sup_{\theta \in \Theta} \mathbb{E}\left[V^{\pi_\theta}(s_0)\right] = J(\theta)$

What even is $\boldsymbol{\nabla J(\theta)}$?

[Williams1992]

Policy Gradient

Goal: Maximize $\sup_{\theta \in \Theta} \mathbb{E}\left[V^{\pi_\theta}(s_0)\right] = J(\theta)$

What even is $\boldsymbol{\nabla J(\theta)}$?

Policy Gradient Theorem:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) Q^{\pi_\theta}(s, a)]$$

Can evaluate using Monte-Carlo roll outs under current policy

[Williams1992]

REINFORCE

Goal: Maximize $\sup_{\theta \in \Theta} \mathbb{E}\left[V^{\pi_\theta}(s_0)\right] = J(\theta)$

For each episode

Sample: $(s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_T, a_T, r_T) \sim \pi_\theta$

Compute Returns: $G_t = \sum_{\tau \geq t} \gamma^{\tau - t} r_\tau$

Update: $\theta_{t+1} = \theta_t + \alpha \nabla_\theta \log \pi_\theta(s_\tau \mid a_\tau) G_\tau \quad \tau = 1, \ldots, T$

[Williams1992]

Policy Gradient

Goal: Maximize $\sup_{\theta \in \Theta} \mathbb{E}\left[V^{\pi_\theta}(s_0)\right] = J(\theta)$

What even is $\boldsymbol{\nabla J(\theta)}$?

Policy Gradient Theorem:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) Q^{\pi_\theta}(s,a)]$$

Two issues:

- Variance in gradient due to log probabilities
- "Double Expectation"

[Williams1992]

Double Expectation

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) Q^{\pi_\theta}(s, a)]$$

Expectation over trajectories

Expectation over policy

[Machado2020]

Double Expectation

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) Q^{\pi_\theta}(s, a)]$$

Expectation over trajectories

Expectation over policy

Can plug in a variety of terms into Q^{π_θ}, similar to DQN

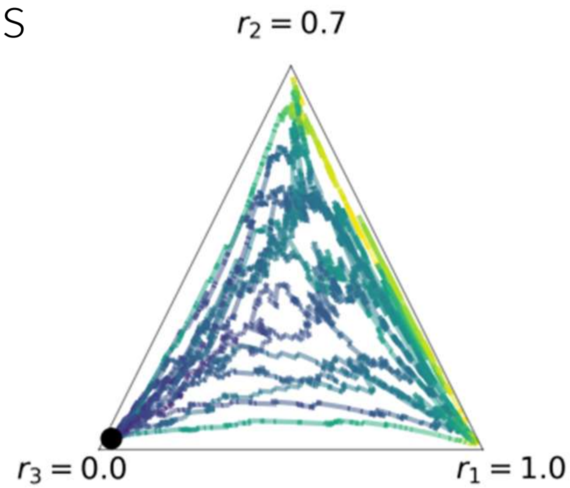
[Machado2020]

Variance Issues

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) Q^{\pi_\theta}(s,a)]$$

Biggest Issue: Variance in policy gradient objective due to:

- Credit assignment
- Log probabilities (opt policy is deterministic...)
- Different distribution of trajectory when applying updates using batches



b = b* - 1

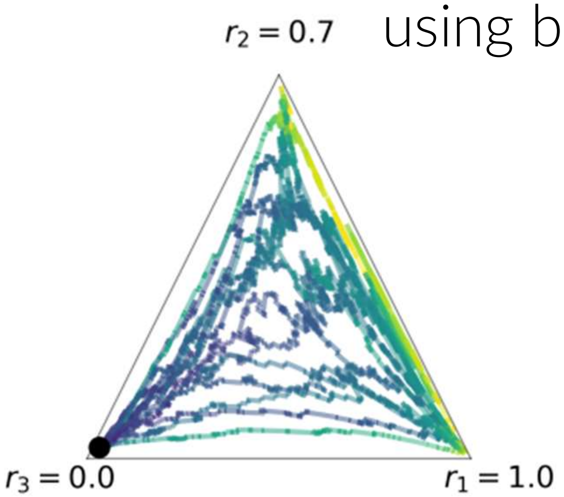
[Machado2020]

Variance Issues

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) Q^{\pi_\theta}(s,a)]$$

Biggest Issue: Variance in policy gradient objective due to:

- Credit assignment
- Log probabilities (opt policy is deterministic...)
- Different distribution of trajectory when applying updates using batches



b = b* - 1

Fix One: Clipping probabilities to $[\epsilon, 1 - \epsilon]$

Fix Two: State dependent baseline, i.e.

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s)(Q^{\pi_\theta}(s,a) - b(s))]$$

Form of control variate

[Machado2020]

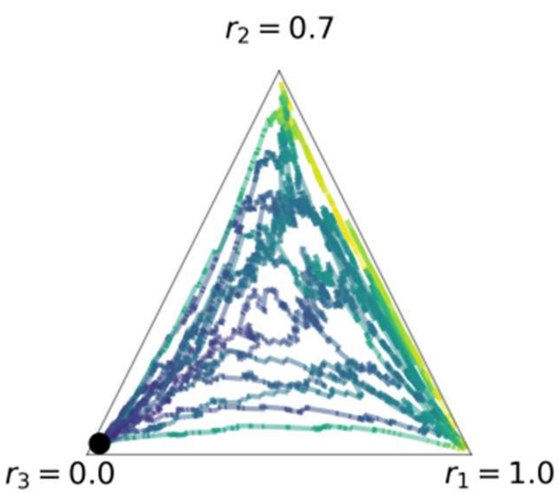
Variance Issues

Fix One: Clipping probabilities to $[\epsilon, 1 - \epsilon]$

Fix Two: State dependent baseline, i.e.

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s)(Q^{\pi_\theta}(s,a) - b(s))]$$

Form of control variate



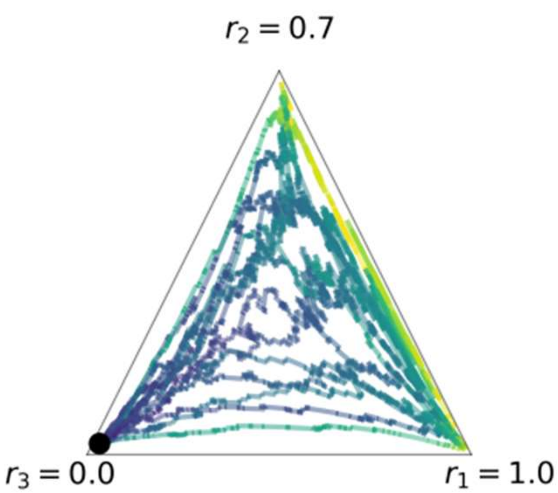
b = b* - 1

Typically use: $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s)(\underbrace{Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)}_{A^{\pi_\theta}(s,a)})]$

Advantage Function

[Machado2020]

Variance Issues



b = b* - 1

Fix One: Clipping probabilities to $[\epsilon, 1 - \epsilon]$

Fix Two: State dependent baseline, i.e.

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s)(Q^{\pi_\theta}(s,a) - b(s))]$$

Form of control variate

Typically use: $$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s)(\underbrace{Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)}_{A^{\pi_\theta}(s,a)})]$$

Using neural network for advantage function makes actor-critic algorithm

Advantage Function

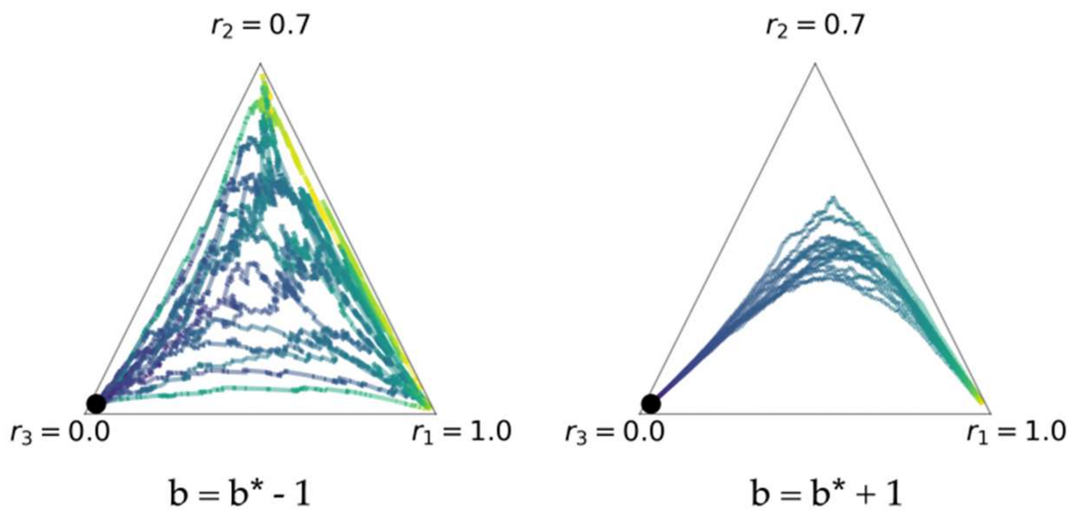
[Machado2020]

Variance Issues

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) Q^{\pi_\theta}(s,a)]$$

Biggest Issue: Variance in policy gradient objective due to:

- Credit assignment
- Log probabilities (opt policy is deterministic…)
- Different distribution of trajectory when applying updates using batches



[Machado2020]

PG+++

Per usual, many flavours and modifications with improved empirical performance.

TRPO: add on KL divergence constraint on policy updates

PPO: add on clipping of the ratio with KL regularization

[Schulman2015, Schulman2017]

High level approach

Use a deep neural network to represent:

- Value function
- Policy
- Model of rewards and transitions
- "universal function approximator"

Optimize loss function via stochastic gradient descent (SGD)

DQN

- Use neural network representation of value function
- Fits estimates using least squares

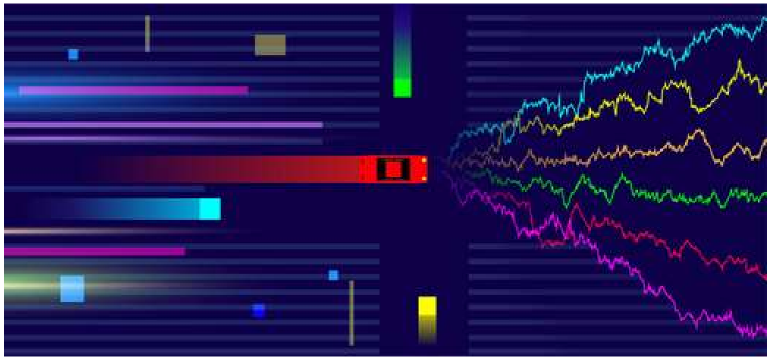
REINFORCE / VPG

- Uses neural network representation of policy
- Fits estimates using policy gradient loss

RL for Operations, 2022

Introduction to Deep RL

Sean Sinclair
Cornell University



References

[Machado2020] Marlos C. Machado. "The True Impact of Baselines in Policy Gradient Methods." *Blog Post*, 2020.

[Schulman2015] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, Pieter Abbeel "Trust Region Policy Optimization". *ICML*, 2020.

[Hessel2017] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, David Silver "Rainbow: Combining Improvements in Deep Reinforcement Learning". *AAAI*, 2018.

[Schulman2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov. "Proximal Policy Optimization Algorithms". *ArXiV*, 2017.

[Mnih2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller. "Playing Atari with Deep Reinforcement Learning". *ArXiV*, 2013.

References

[Silver2017] David Silver et a. "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm." *ArXiV*, 2017.

[Tesauro1995] Gerald Tesauro. "Temporal Difference Learning and TD-Gammon". *Communications of the ACM*, 1995.

[Hasselt2015] Hado Van Hasselt, Arthur Guez, David Silver. "Deep Reinforcement Learning with Double Q-Learning". *AAAI*, 2016.

[Wang2015] Ziju Wang et al. "Dueling Network Architectures for Deep Reinforcement Learning." *ArXiV*, 2015.

[Williams1992] Ronald J Williams. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning." *Machline Learning*, 1992.

[Vikhar2016] Pradnya A Vikhar. "Evoluationary algorithms: A Critical Review and its Future Prospects." *NeurIPS*, 2016.

References

[Salimans2017] Tim Salimans et a. "Evolution Strategies as a Scalable Alternative to Reinforcement Learning." *NeurIPS*, 2017.