RL@Coursera
week 4

Limitations of tabular methods

Yandex
Data Factory

LAMBDA

Tabular methods are limited

- In tabular SARSA

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

Any single cell update does not affect any other cell

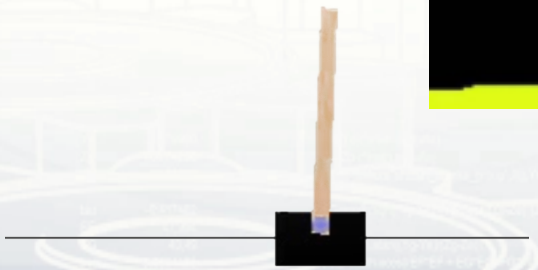
Tabular methods are limited

- In tabular SARSA

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

- Any single cell update does not affect any other cell

Screenshot Taito Corporation, Midway



Tabular methods are limited

- In tabular SARSA

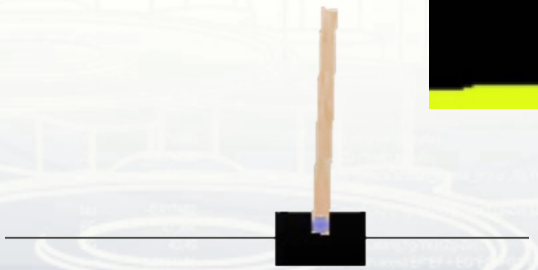
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

- Any single cell update does not affect any other cell

Limitations:

- # of parameters > # of states
- Memory Data Time

Screenshot Taito Corporation, Midway



Function approximation in RL

We aim # of parameters independent of # of states

$$\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$$

$$\hat{q}(s, a, \mathbf{w}) \approx q_\pi(s, a)$$

Any single parameter **affects** values of **all states**!

Function approximation in RL

of parameters independent of # of states

$$\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$$

$$\hat{q}(s, a, \mathbf{w}) \approx q_\pi(s, a)$$

Any single parameter **affects** values of **all states**!

It is wise to obtain the following mappings:

$$s \mapsto v_\pi(s)$$

$$s, a \mapsto q_\pi(s, a)$$

Function approximation in RL

of parameters independent of # of states

$$\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$$

$$\hat{q}(s, a, \mathbf{w}) \approx q_\pi(s, a)$$

Any single parameter **affects** values of **all states**!

It is wise to obtain the following mappings:

Supervised learning?

$$s \mapsto v_\pi(s)$$

$$s, a \mapsto q_\pi(s, a)$$

But what is the training data?

Reduction to Supervised Learning problem – MC

Monte Carlo, MC

Ideal world goals

$$s \mapsto \mathbb{E}_\pi[G_t | S_t = s]$$

$$s, a \mapsto \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$$

Reduction to Supervised Learning problem – MC

Monte Carlo, MC

Ideal world goals

$$s \mapsto \mathbb{E}_\pi[G_t | S_t = s]$$

$$s, a \mapsto \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$$

Sample based estimates of goals $g(s), g(s,a)$

$$s \mapsto R(s, \pi(s)) + \gamma G_{t+1}$$

$$s, a \mapsto R(s, a) + \gamma G_{t+1}$$

Reduction to Supervised Learning problem – MC

Monte Carlo, MC

Ideal world goals

$$s \mapsto \mathbb{E}_\pi[G_t | S_t = s]$$

$$s, a \mapsto \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$$

Sample based estimates of goals $g(s), g(s,a)$

$$s \mapsto R(s, \pi(s)) + \gamma G_{t+1}$$

$$s, a \mapsto R(s,a) + \gamma G_{t+1}$$

they don't depend on any parameters

Numbers, known at the end of an episode

Reduction to Supervised Learning problem – MC

Temporal difference, TD

Ideal world goals

$$s \mapsto \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

$$s, a \mapsto \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a]$$

Reduction to Supervised Learning problem – MC

Temporal difference, TD

Ideal world goals

$$s \mapsto \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

$$s, a \mapsto \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a]$$

Sample based estimates of goals $g(s), g(s,a)$

$$s \mapsto R(s, \pi(s)) + \gamma \hat{v}_\pi(S_{t+1}, \mathbf{w})$$

$$s, a \mapsto R(s, a) + \gamma \hat{v}_\pi(S_{t+1}, \mathbf{w})$$

* Targets here are biased!

Reduction to Supervised Learning problem – MC

Temporal difference, TD

Ideal world goals

$$s \mapsto \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

$$s, a \mapsto \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a]$$

Sample based estimates of goals $g(s), g(s,a)$

$$s \mapsto R(s, \pi(s)) + \gamma \hat{v}_\pi(S_{t+1}, \mathbf{w})$$

$$s, a \mapsto R(s,a) + \gamma \hat{v}_\pi(S_{t+1}, \mathbf{w})$$

Functions of parameters $\mathbf{w}$!

Why not to denote goals as $g(s, a; \mathbf{w})$?

Loss is the same as for a regression problem

Many losses are available – MSE, MAE, Huber loss

MSE

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2} \cdot \sum_{s,a} \rho_\pi(s,a)[g(s,a) - \hat{q}_\pi(s,a,\mathbf{w})]^2$$

Loss is the same as for a regression problem

Many losses are available – MSE, MAE, Huber loss

Weights of "importance"

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2} \cdot \sum_{s,a} \rho_\pi(s,a)\left[g(s,a) - \hat{q}_\pi(s,a,\mathbf{w})\right]^2$$

All possible states and actions

Weight of "importance" $\rho_\pi(s,a)$ can be thought of as

- fraction of time ← distribution in the limit
  - policy encounters state **s**
  - and in that state makes action **a**

finite

ρ is also sampled-based

Recap: online, offline, on-policy, off-policy

we don't control for

Off-policy – more general, very difficult

- **Behaviour** policy – collects data (makes actions)
- **Target** policy – subject to evaluation & improvement

On-policy – less general, easier

- **Behaviour** policy equals **target** policy

* An alg learning off-policy can also learn on-policy.

The opposite is not necessarily true!

Recap: online, offline, on-policy, off-policy

Off-policy – more general, very difficult

- **Behaviour** policy – collects data (makes actions)
- **Target** policy – subject to evaluation & improvement

On-policy – less general, easier

- **Behaviour** policy equals **target** policy

Usually, but not always:

- Online – update policy **during the episode** (e.g. TD)
- Offline – update policy **after an episode ends** (e.g. MC)

Minimizing loss with gradient

Loss

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2} \cdot \sum_{s,a} \rho_\pi(s,a) \overbrace{[g(s,a) - \hat{q}_\pi(s,a,\mathbf{w})]^2}^{\mathcal{L}_{s,a}(\mathbf{w})}$$

this is sample-based,

not exact

Minimizing loss with gradient

Loss

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2} \cdot \sum_{s,a} \rho_\pi(s,a) \overbrace{[g(s,a) - \hat{q}_\pi(s,a,\mathbf{w})]^2}^{\mathcal{L}_{s,a}(\mathbf{w})}$$

Gradient descent (GD)

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w})$$

fastest descent

Minimizing loss with gradient

Loss

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2} \cdot \sum_{s,a} \rho_\pi(s,a) \overbrace{[g(s,a) - \hat{q}_\pi(s,a,\mathbf{w})]^2}^{\mathcal{L}_{s,a}(\mathbf{w})}$$

Gradient descent (GD)

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w})$$

Stochastic gradient descent (SGD)

On-policy: $s, a \backsim \rho_\pi$ — target

Off-policy: $s, a \backsim \rho_b$ — behavior

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} \mathcal{L}_{s,a}(\mathbf{w})$$

From gradient to semi-gradient

Loss

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2} \cdot \sum_{s,a} \rho_\pi(s,a) \overbrace{[g(s,a) - \hat{q}_\pi(s,a,\mathbf{w})]^2}^{\mathcal{L}_{s,a}(\mathbf{w})}$$

goal

current est of goal

From gradient to semi-gradient

Loss

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2} \cdot \sum_{s,a} \rho_\pi(s,a) \overbrace{[g(s,a) - \hat{q}_\pi(s,a,\mathbf{w})]^2}^{\mathcal{L}_{s,a}(\mathbf{w})}$$

Goal is a function of **w** in case of TD

this is problematic !!

just numbers in case of MC

i.e easier to differentiate

From gradient to semi-gradient

Loss

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2} \cdot \sum_{s,a} \rho_\pi(s,a) \overbrace{[g(s,a) - \hat{q}_\pi(s,a,\mathbf{w})]^2}^{\mathcal{L}_{s,a}(\mathbf{w})}$$

Goal is a function of **w** in case of TD

Stochastic semi-gradient update treats goals as **fixed**:

$$\nabla_{\mathbf{w}} g(s,a) = 0$$

this is very similar to what is going on in supervised learning

From gradient to semi-gradient

Loss

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2} \cdot \sum_{s,a} \rho_\pi(s,a) \overbrace{[g(s,a) - \hat{q}_\pi(s,a,\mathbf{w})]^2}^{\mathcal{L}_{s,a}(\mathbf{w})}$$

Goal is a function of **w** in case of TD

Stochastic semi-gradient update treats goals as **fixed**:

$$\nabla_{\mathbf{w}} g(s,a) = 0$$

This assumption simplifies math a lot:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} \mathcal{L}_{s,a}(\mathbf{w})$$

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha[g(s,a) - \hat{q}_\pi(s,a,\mathbf{w})] \nabla_{\mathbf{w}} \hat{q}_\pi(s,a,\mathbf{w})$$

Semi-gradient update

- Treats goals $g(s, a)$ as fixed numbers
- Changes parameters to move estimates closer to targets
- **Ignores** effect update on the target
- Is **not** a proper gradient
  - No SGD's theoretical convergence properties
  - Converges reliably in most cases
  - More computationally efficient than true SGD
- Meaningful thing to do

Targets $g(s,a)$ define what and how do we learn

SARSA

$$g(s,a) = R(s,a) + \gamma \hat{q}_\pi(S_{t+1}, A_{t+1}, \mathbf{w})$$

Expected SARSA

$$g(s,a) = R(s,a) + \gamma \sum_a \pi(a|S_{t+1}) \hat{q}_\pi(S_{t+1}, a, \mathbf{w})$$

Q-learning

$$g(s,a) = R(s,a) + \gamma \max_a \hat{q}_\pi(S_{t+1}, a, \mathbf{w})$$

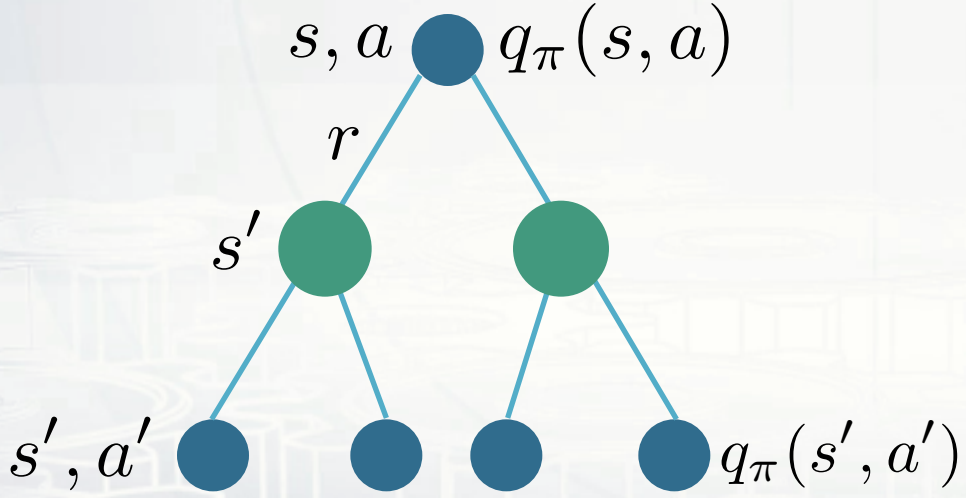
* In all $g(s,a)$ is random var, since it depends on S_{t+1}
Yet in SARSA, it additionally depends on A_{t+1}

Semi-gradient SARSA (on-policy)

Tabular SARSA

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right]$$

Bellman expectation equation for $q(s, a)$



Full-width backup

Semi-gradient SARSA (on-policy)

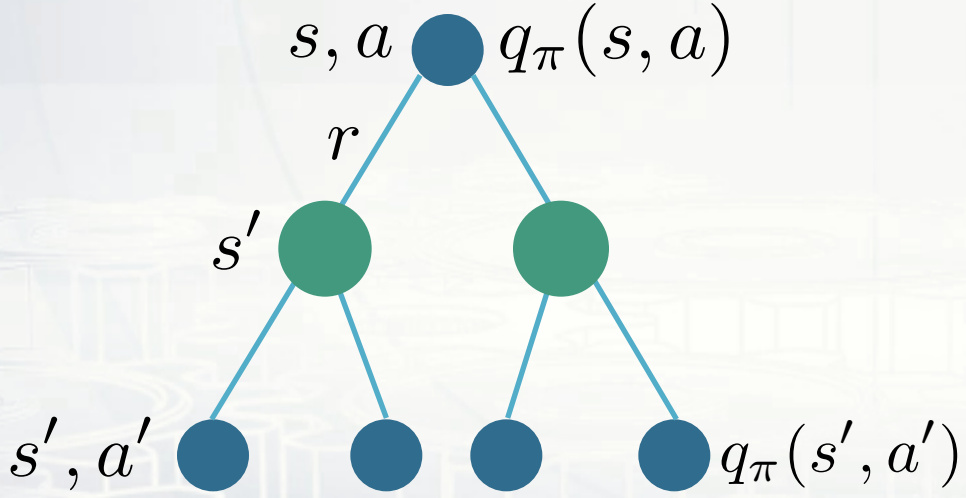
Tabular SARSA

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right]$$

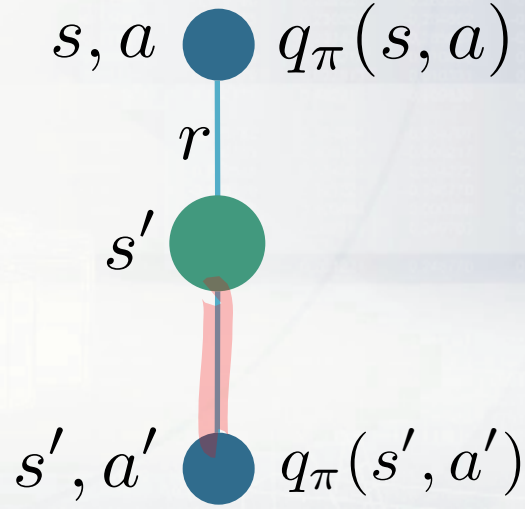
Approximate SARSA

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\left[R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w})\right] \bigtriangledown \hat{q}(S_t, A_t, \mathbf{w})$$

Bellman expectation equation for $q(s, a)$



Full-width backup



Sample based estimate

Semi-gradient expected SARSA (off-policy)

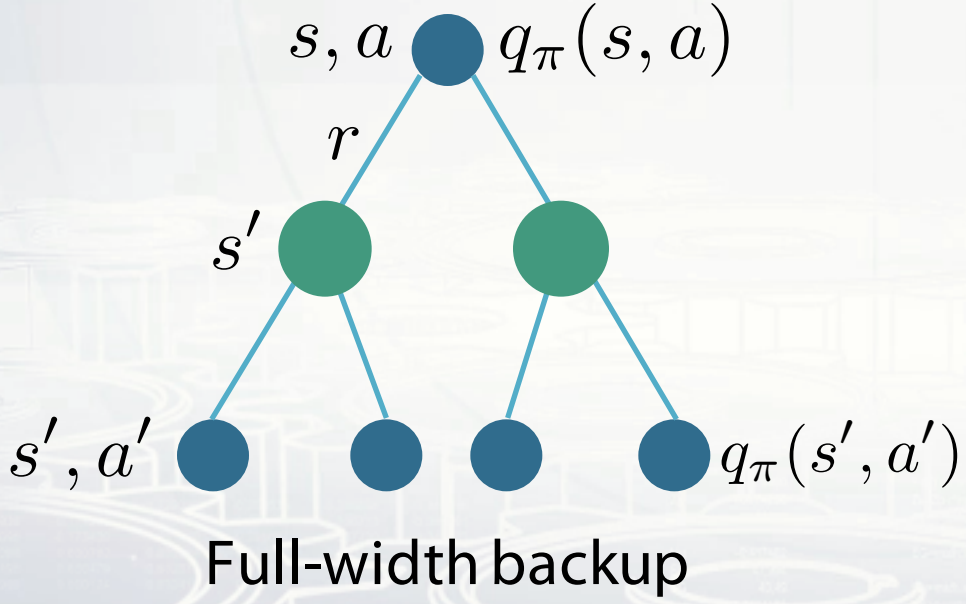
Tabular expected SARSA

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

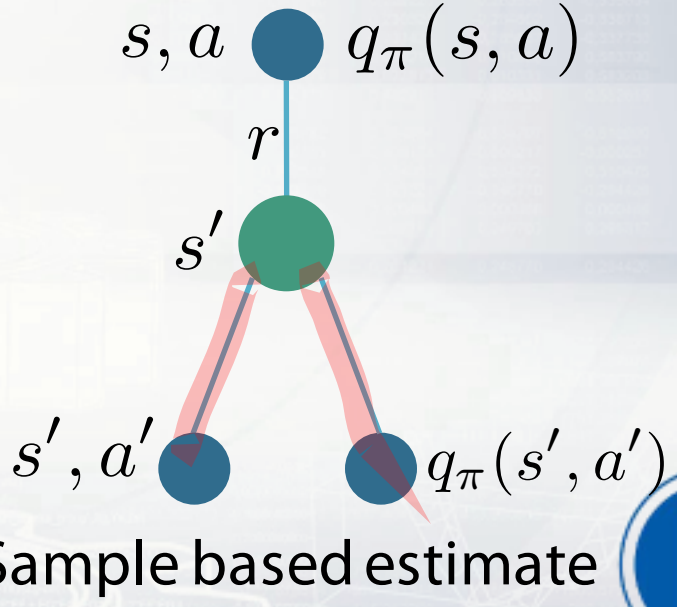
Approximate expected SARSA

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \left[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) \hat{q}(S_{t+1}, a, \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w}) \right] \nabla \hat{q}(S_t, A_t, \mathbf{w})$$

Bellman expectation equation for $q(s, a)$



Full-width backup



Sample based estimate

Semi-gradient Q - learning (off-policy)

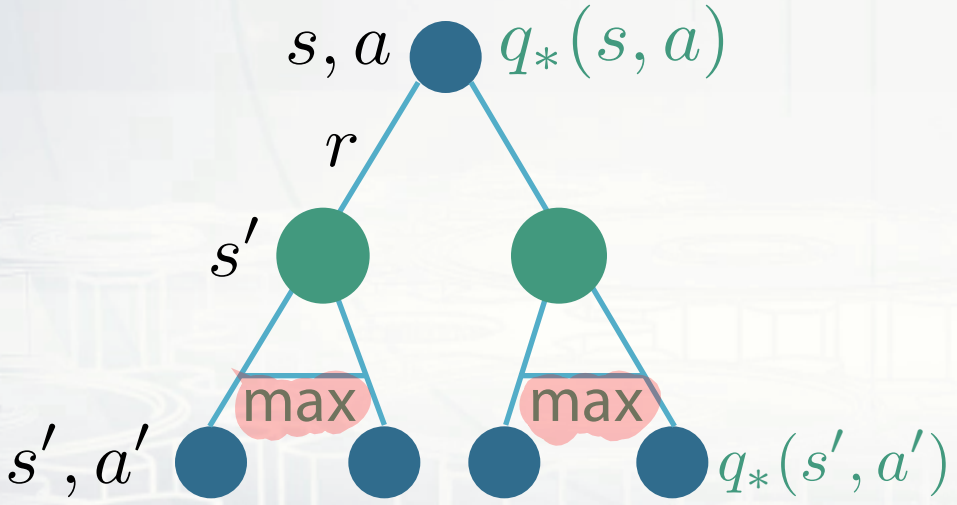
Tabular Q-learning

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

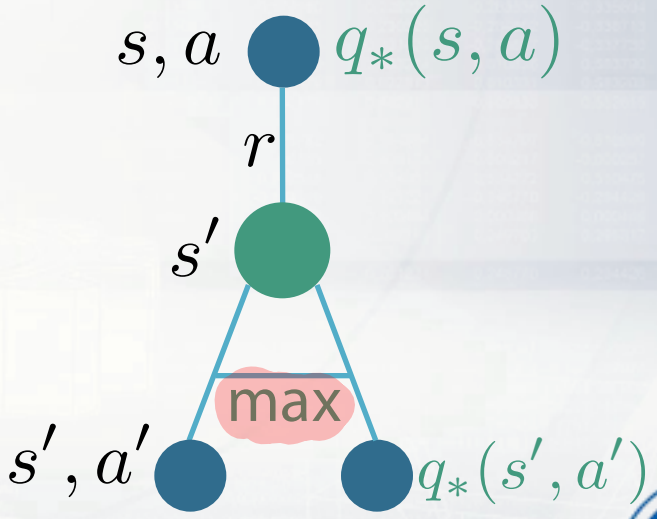
Approximate Q-learning

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\left[R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w})\right] \bigtriangledown \hat{q}(S_t, A_t, \mathbf{w})$$

Bellman optimality equation $q_*(s, a)$



Full-width backup



Sample based estimate

Recap

We already know

- What is approximate learning and why is it important
- How to reduce the RL task to supervised learning
- How does semi-gradient differ from gradient update
- What objective corresponds to what algorithm

We have not yet talked about

- Practical and theoretical caveats in RL

Application of Supervised Learning to RL

Round (0): general well-known boring problems

Round (0): general well-known boring problems

- Too many state-action pairs

$$|\mathcal{A}| \cdot |\mathcal{S}| = 18 \cdot 256^{210 \times 160 \times 3} \approx 10^{100000} \gg 10^{83}$$

  - Methods should be **sample efficient**

Curse of dimensionality

Round (0): general well-known boring problems

- Too many state-action pairs

$$|\mathcal{A}| \cdot |\mathcal{S}| = 18 \cdot 256^{210 \times 160 \times 3} \approx 10^{100000} \gg 10^{83}$$

  - Methods should be **sample efficient**
- Meaningful decisions need accurate value estimates
  - Methods should be **very flexible**

Round (0): general well-known boring problems

- Too many state-action pairs

$$|\mathcal{A}| \cdot |\mathcal{S}| = 18 \cdot 256^{210 \times 160 \times 3} \approx 10^{100000} \gg 10^{83}$$

  - Methods should be **sample efficient**
- Meaningful decisions need accurate value estimates
  - Methods should be **very flexible**
- Any model has finite # of params
  - Even universal approximators are limited
- Bias-variance tradeoff – overfitting / underfitting
  - Cross validation on scarce data is unreliable

Problems with Supervised Learning (SL) in RL

1. **Unusual data**
    1. Correlation
    2. Dependence on policy
    3. Proximity in space and time
2. **Nonstationarity**
    1. Policy improvement
    2. Nonstationary task
3. **The deadly triad** (off-policy learning)

Round (1.1): data – correlated samples

Sequences of highly correlated
non-iid data

Screenshot Taito Corporation, Midway

Round (1.1): data – correlated samples

- Sequences of highly correlated non-iid data
  - Forgetting useful facts and features
  - Much of SL rely on iid assumption
  - Learning can be inefficient
  - SGD loses convergence guarantees

Screenshot Taito Corporation, Midway

Round (1.2): data – dependence on policy

as agent learns, S & R changes subsequently

- Unseen data comes as agent learns new things
- An agent can learn the fatal behavior
  - even New data could be insufficient to unlearn it
- Agent experiments with data stream by exploration

Round (1.3): data – proximity in space and time

Non-differentiability

- $Q(s, a)$ often changes abruptly in **s** and **a**
  - Close states could be **arbitrary far in value**
  - Successive states (actions) could be **arbitrary far in value**
  - Unstable gradients, much data needed for SGD
  - Numerical problems

Round (1.3): data – proximity in space and time

- $Q(s, a)$ often changes abruptly in **s** and **a**
  - Close states could be **arbitrary far in value**
  - Successive states could be **arbitrary far in value**
  - Unstable gradients, much data needed for SGD
  - Numerical problems

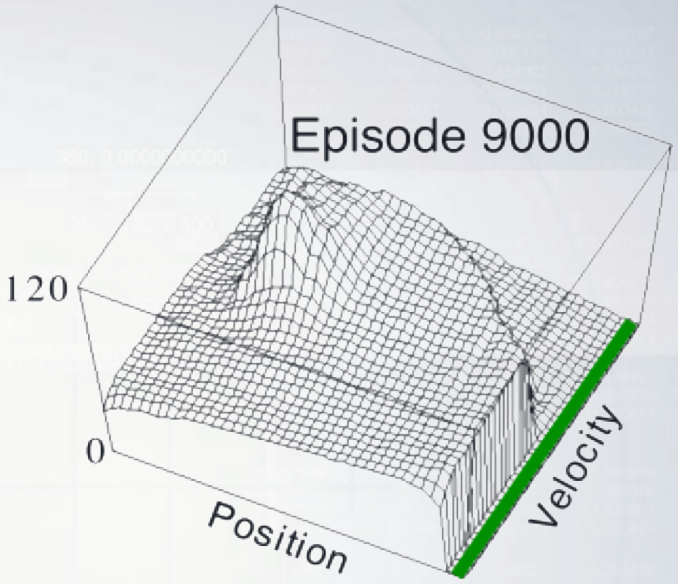
✗ (TD-specific) Errors in estimates propagate

Approximate Q-learning update

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \left[R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w}) \right] \bigtriangledown \hat{q}(S_t, A_t, \mathbf{w})$$

Makes $\hat{q}(S_t, A_t, \mathbf{w})$ closer to $R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \mathbf{w})$

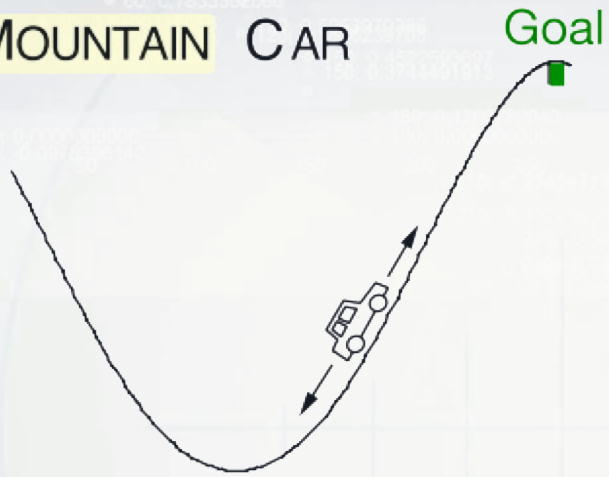
Abrupt changes in state-values

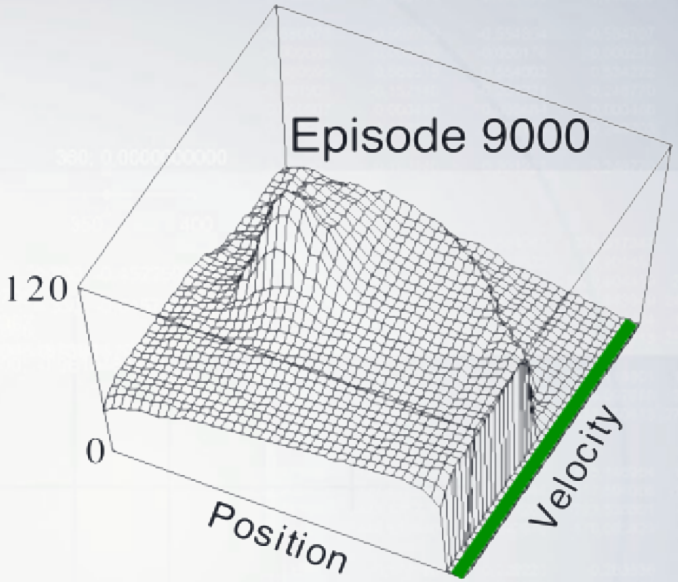


Richard S. Sutton and Andrew G. Barto, “Reinforcement Learning: An Introduction”, Draft of second edition (2017)

Abrupt changes in state-values

Ex1





EX2

Richard S. Sutton and Andrew G. Barto, “Reinforcement Learning: An Introduction”, Draft of second edition (2017)

Round (2): inherent nonstationarity – GPI

About learning task itself

- Cannot assume fixed training data distribution
  - TD targets are invalidated
  - MC targets no longer apply

* Almost all alg can be expressed as GPI

Round (2): inherent nonstationarity – GPI

- Cannot assume fixed training data distribution
  - TD targets are invalidated
  - MC targets no longer apply
- Numeric problems (oscillating behaviour)
  - Small change in Q-values
    - → drastic change in policy
    - → drastic change in training data
    - → large gradients
    - → large update in Q-values
- The environment can itself be nonstationary

Round (3): The deadly triad – model divergence

parameters $\rightarrow \infty$

1. Off-policy learning
   - E.g. learning target π while following behaviour b
2. Bootstrapping
   - Updating a guess towards another guess (TD, DP)
3. Function approximation
   - Using model with # of params smaller then # of states

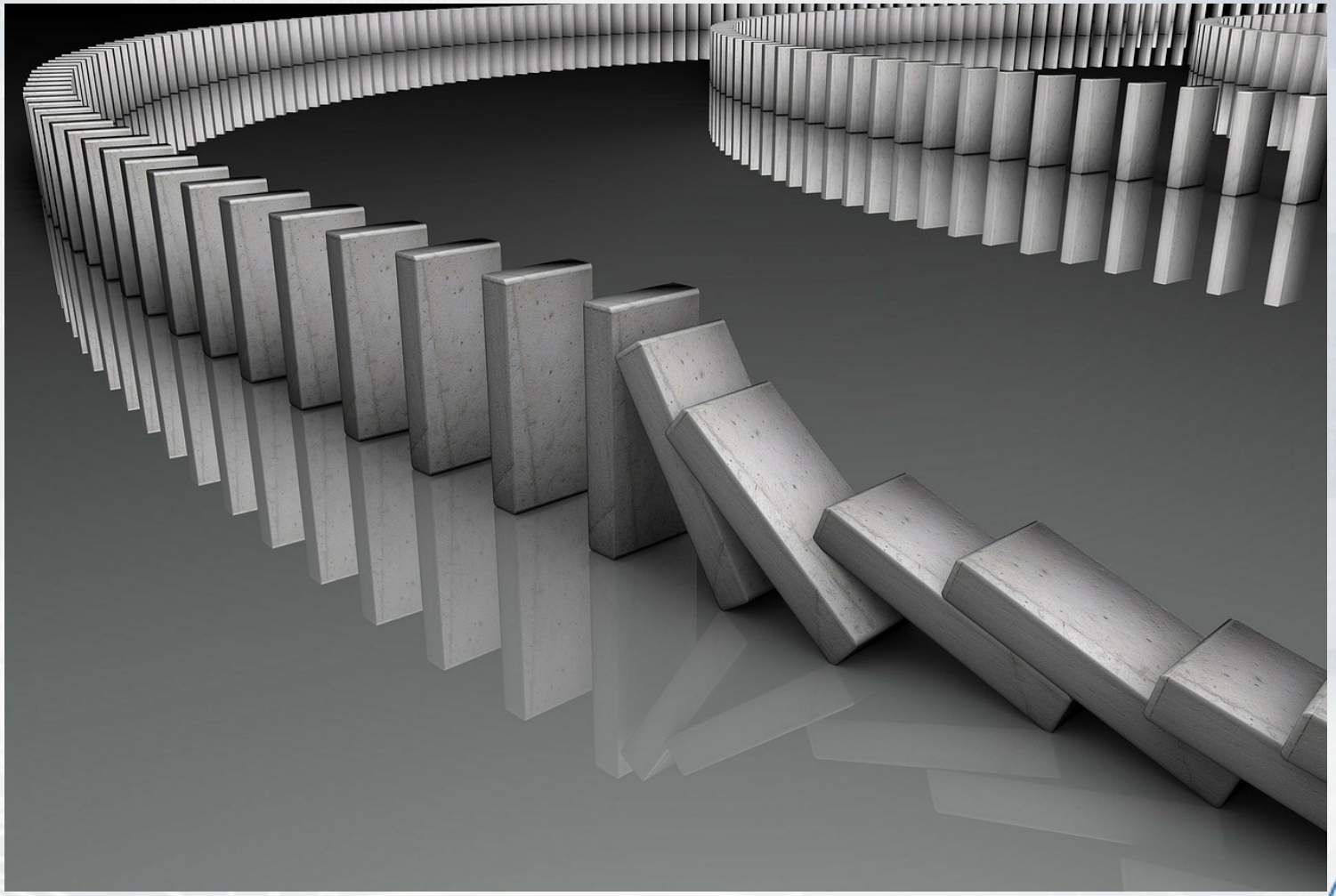
Round (3): The deadly triad – model divergence

1. Off-policy learning
   - E.g. learning target π while following behaviour b
2. Bootstrapping
   - Updating a guess towards another guess (TD, DP)
3. Function approximation
   - Using model with # of params smaller then # of states

✗ Divergence is not connected with

- Sampling, exploration, greediness, control
- Complexity of the model (even linear models diverge)

Approximate RL: how NOT to mess up



Case study: DQN

DQN: bird's eye view

- First successful application of learning
  - directly from raw visual inputs (same as humans do)
  - in a wide variety of environments (Atari games)
- Deep convolutional network
- No hand-designed features
- Q-learning with stability tricks
- Epsilon-greedy exploration

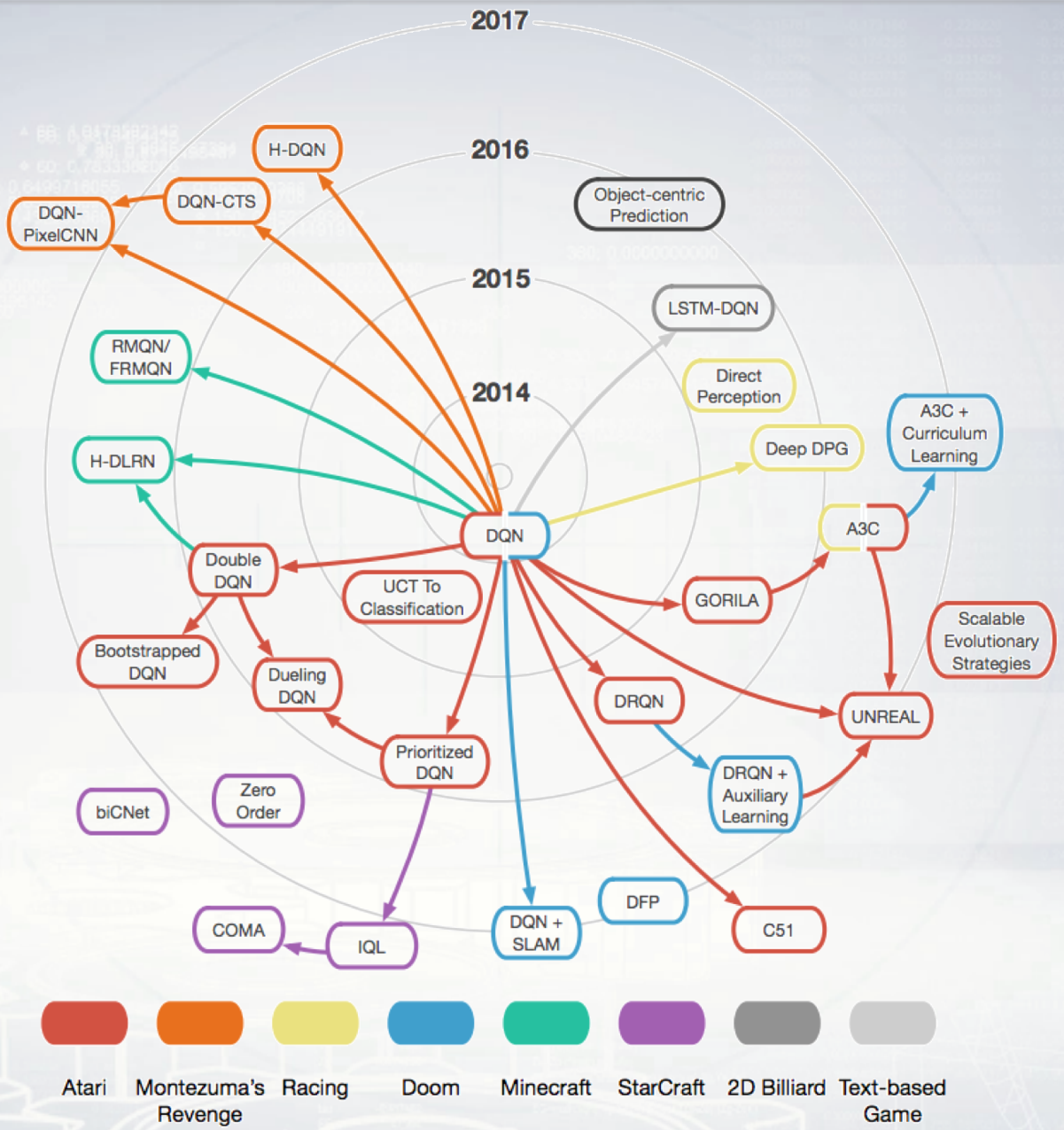
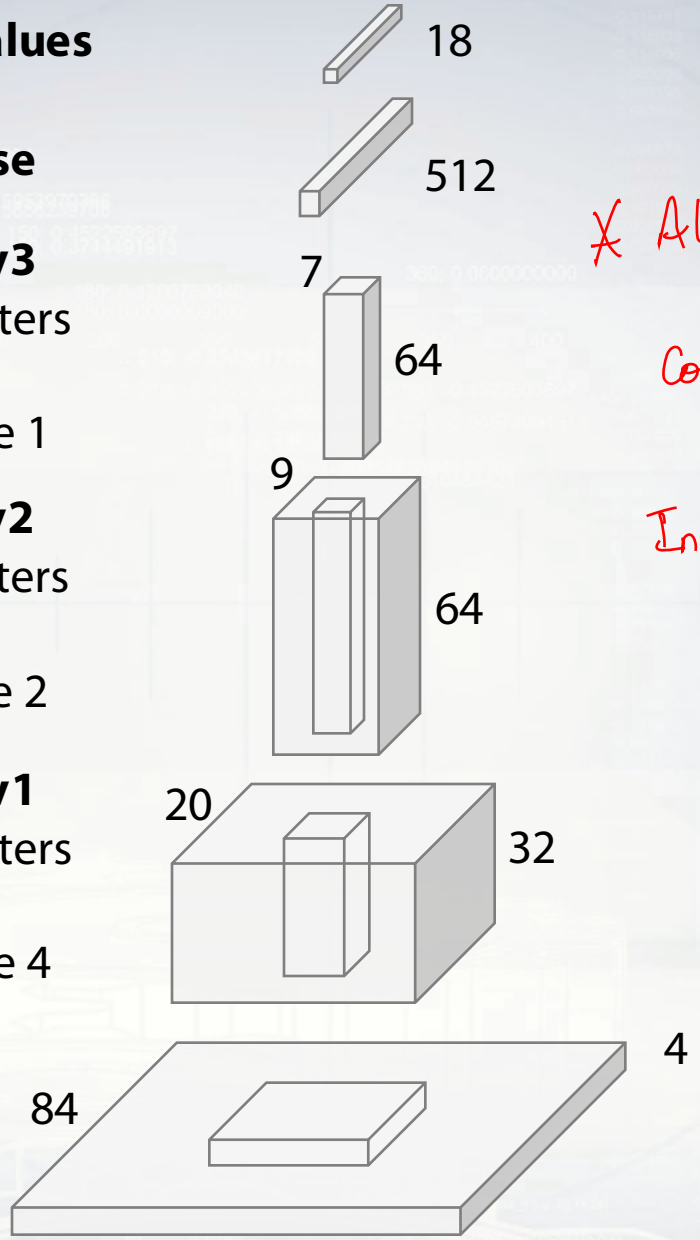


Image credit: Justesen, Niels, et al. "Deep Learning for Video Game Playing." arXiv preprint arXiv:1708.07902 (2017)

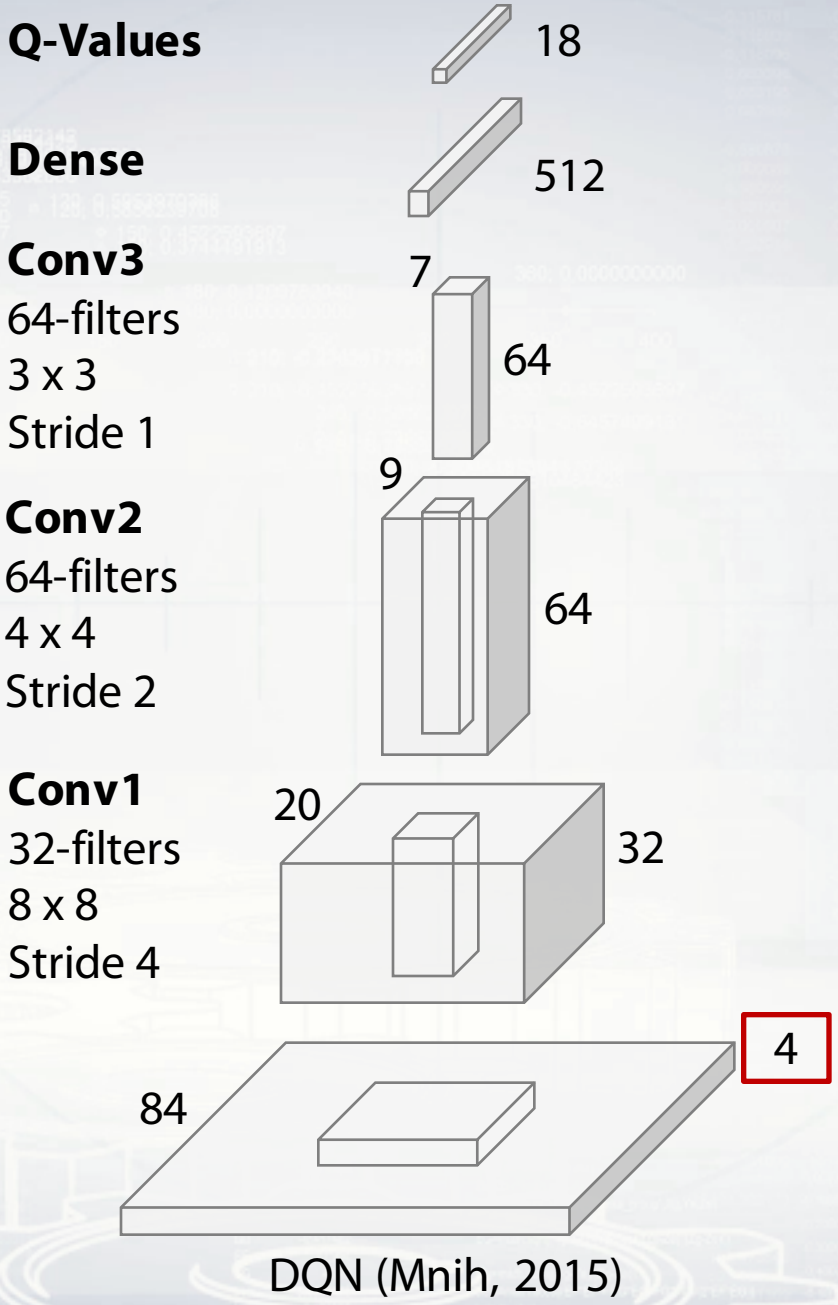
Architecture



DQN (Mnih, 2015)

* Always try to reduce the complexity of the task first

In RL, reduce state-space

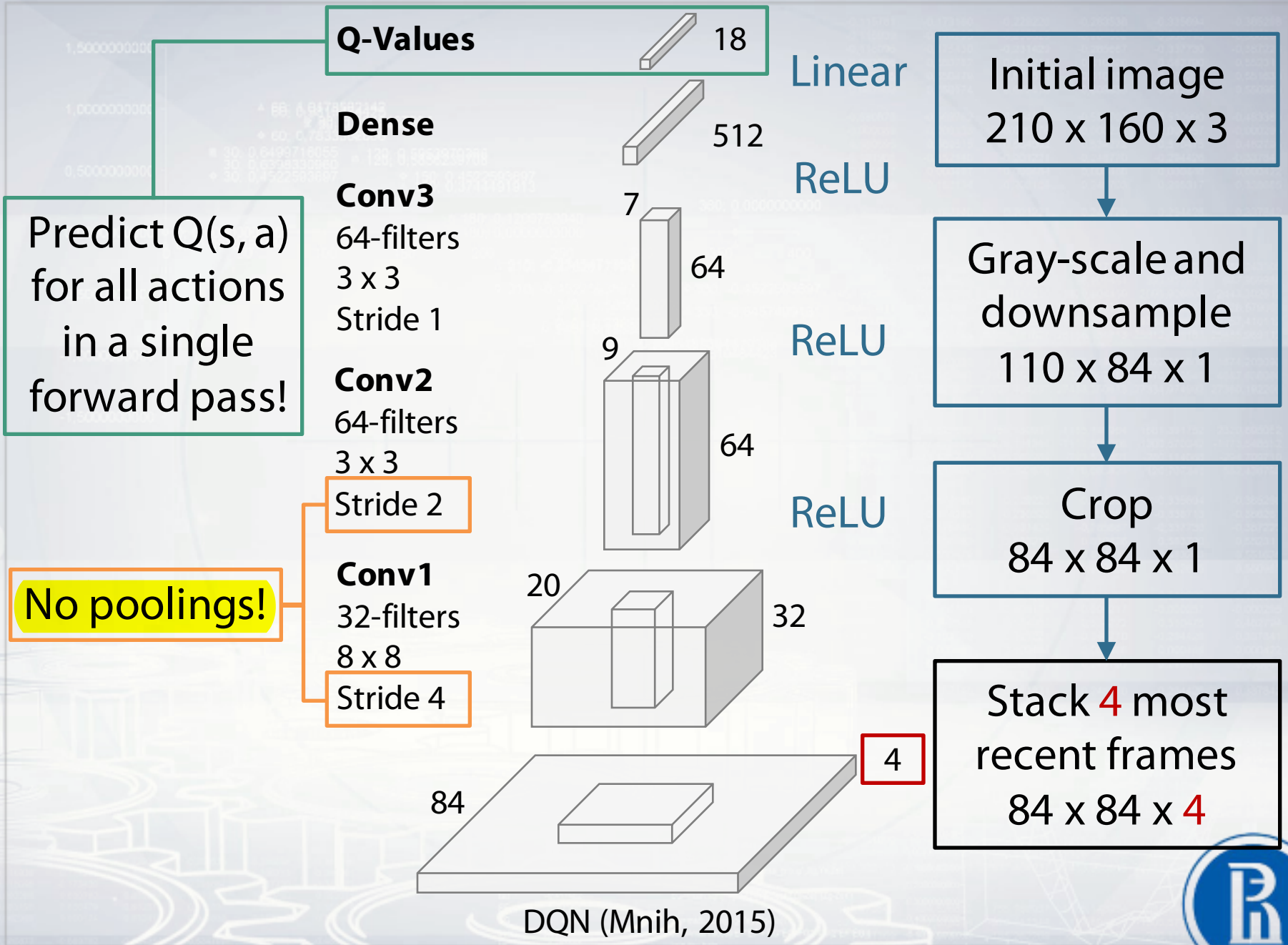


Initial image
210 x 160 x 3

Gray-scale and downsample
110 x 84 x 1

Crop
84 x 84 x 1

Stack 4 most recent frames
84 x 84 x 4

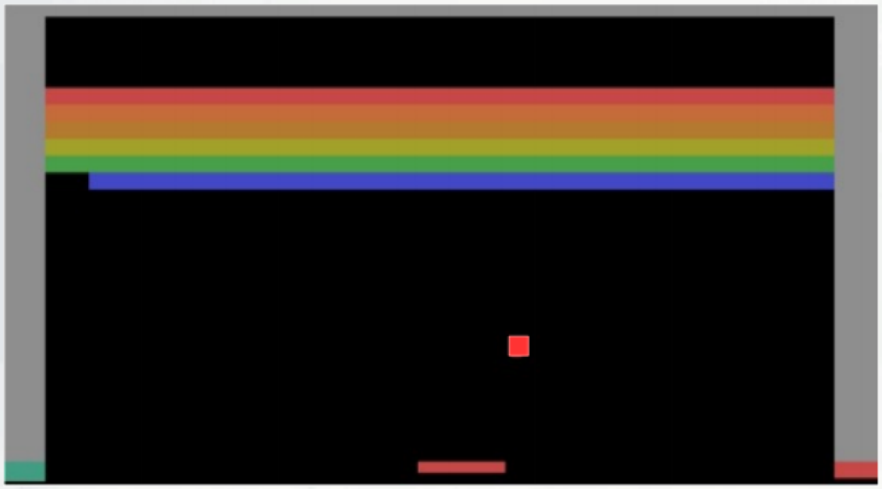


DQN (Mnih, 2015)

Stack 4 frames = fourth-order markov assumption

Markov assumption

$$p(r_{t+1}, s_{t+1} | s_0, a_0, \ldots, r_t, s_t, a_t) = p(r_{t+1}, s_{t+1} | s_t, a_t)$$



Rather impossible infering dir

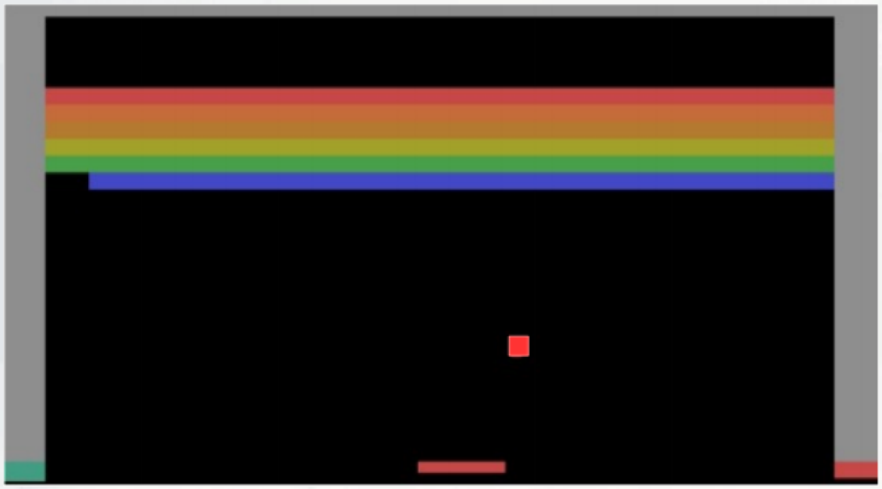
or speed of the ball!

* Partial Observability is the lack of such comprehensive info in an observation

Stack 4 frames = fourth-order markov assumption

Markov assumption

$$p(r_{t+1}, s_{t+1} | s_0, a_0, \ldots, r_t, s_t, a_t) = p(r_{t+1}, s_{t+1} | s_t, a_t)$$



Fourth-order Markov assumption

$$p(r_{t+1}, s_{t+1} | s_0, a_0, \ldots, r_t, s_t, a_t) = p(r_{t+1}, s_{t+1} | s_t, a_t, \ldots, s_{t-3}, a_{t-3})$$

* The enough order is still unclear however!

DQN fixes some of the instability problems

1. Sequential correlated data
   - May hurt convergence and performance
2. Instability of data distribution due to policy changes
   - Policy may diverge and / or oscillate
3. Unstable gradients
   - Absolute value of $Q(s, a)$ may vary much across states

DQN fixes some of the instability problems

1. Sequential correlated data

   Soln

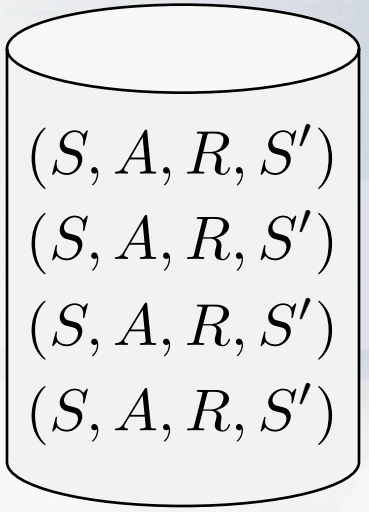
   - Experience replay
2. Instability of data distribution due to policy changes
   - Target networks
3. Unstable gradients
   - Reward clipping

(1) Sequential data → Experience replay

Semi-gradient update for approximate Q-learning

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \left[R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w}) \right] \bigtriangledown \hat{q}(S_t, A_t, \mathbf{w})$$

1. Store tuples (S, A, R, S') in a pool
2. Sample tuple(s) from pool at random
3. Update the model of Q-func
4. Act epsilon-greedy w.r.t. Q-func
5. Add new samples to pool
6. Go to 2



(1) Sequential data → Experience replay

only for off-policy learning

+ Helps against correlations
+ Increases sample efficiency
+ Reduce variance of updates
+ Computations are easy to parallel
+ Analog of sample based **model of the world**
- Memory intensive (DQN stored ~ 1 million interactions)
- Random sampling from pool could be improved
- Older interactions were obtained under weaker policy

(2) Policy oscillation → Target networks

Still

Unwanted source of instability

- Targets are functions of parameters
- Errors in estimates propagate into other estimates

Standard Q-learning update

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \Big[R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w}) \Big] \bigtriangledown \hat{q}(S_t, A_t, \mathbf{w})$$

(2) Policy oscillation → Target networks

Unwanted source of instability

- Targets are functions of parameters
- Errors in estimates propagate into other estimates

Standard Q-learning update

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\Big[R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w})\Big] \bigtriangledown \hat{q}(S_t, A_t, \mathbf{w})$$

Q-learning update with target networks

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\Big[R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \mathbf{w}^-) - \hat{q}(S_t, A_t, \mathbf{w})\Big] \bigtriangledown \hat{q}(S_t, A_t, \mathbf{w})$$

Copy parameters to target network

- (**hard**) Copy $\mathbf{w}$ to $\mathbf{w}^-$ once in awhile
- (**soft**) Maintain $\mathbf{w}^-$ as moving average of $\mathbf{w}$

of current network

Same architecture diff parameters

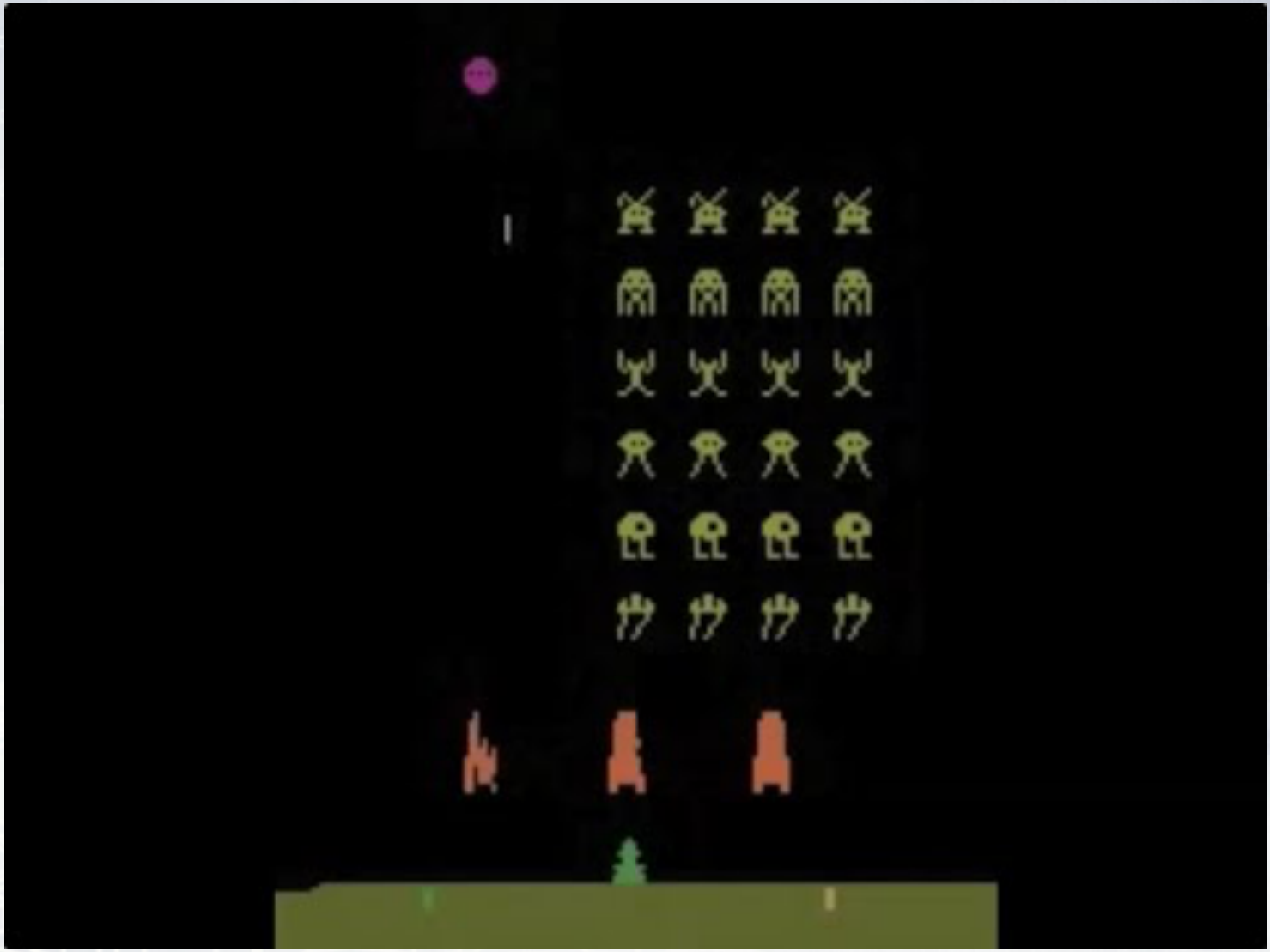
(2) Unstable gradients → Reward clipping

- Don't know the scale of reward beforehand
- Don't want numeric problems with large Q-values

- ~~Clip the reward to [-1, 1]~~

  + less peaky Q-values
  + good gradients
  – cannot distinguish between good and great

Results: Space invaders



Code is available at https://github.com/deepmind/dqn
More videos at https://www.youtube.com/playlist?list=PLgOp827qARy0qNyZq5Y6S6vRJO3tb1WcW