(Asynchronous) Actor Advantage Critics Reinforcement Learning: A2C and A3C

Reinforcement Learning Group Meeting

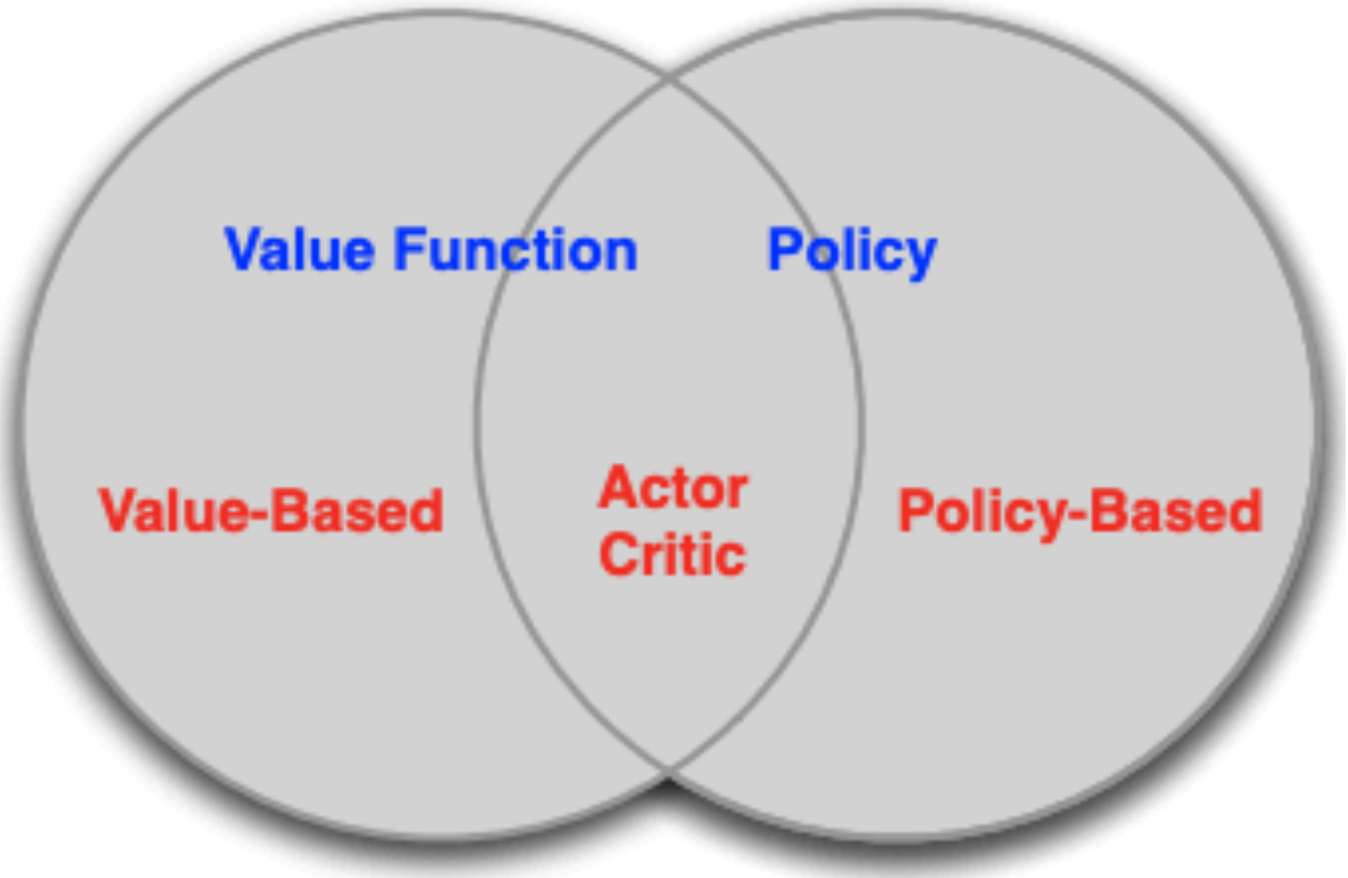
Yanjun Gao

March 14, 2019

penn state university

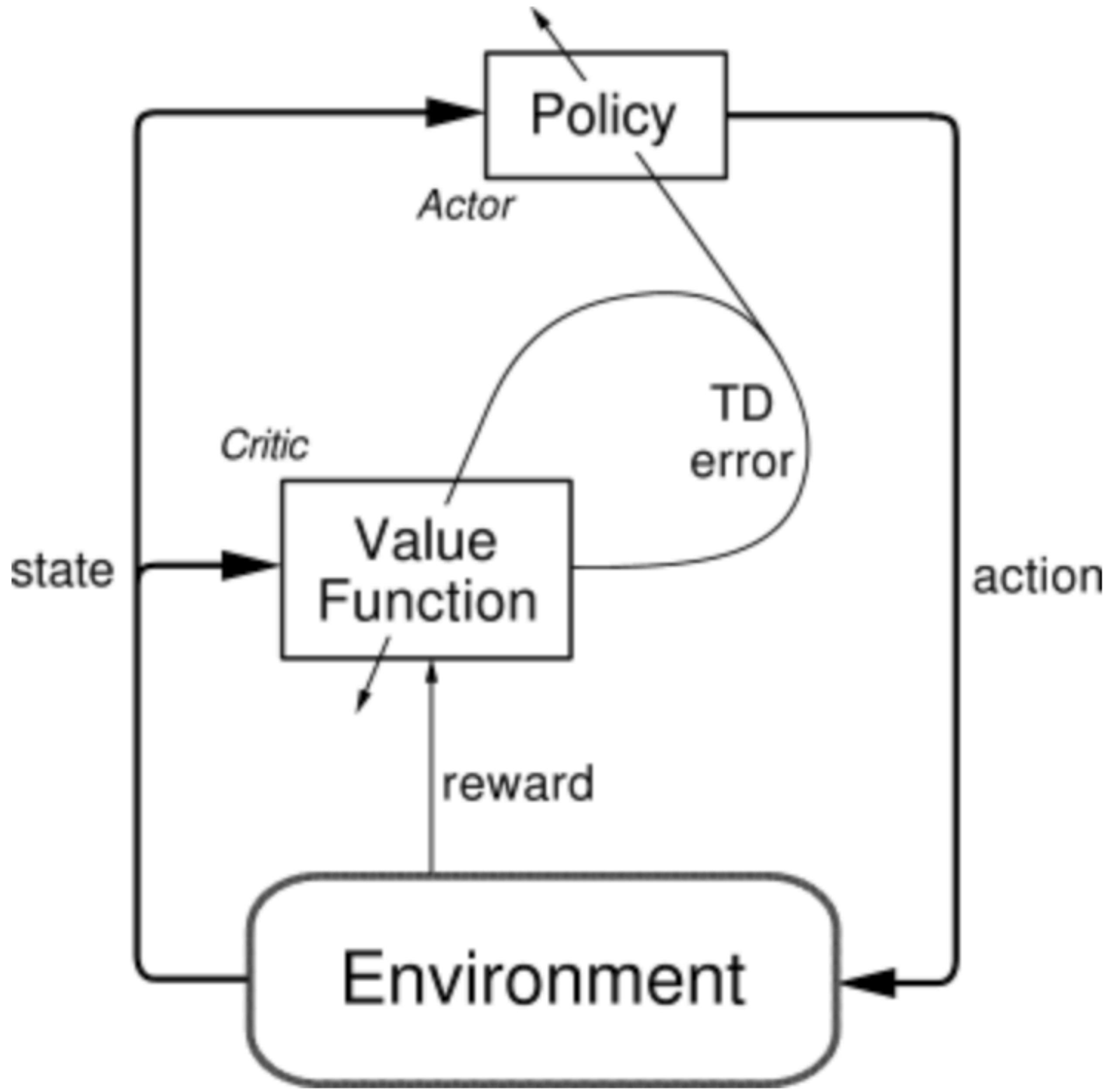
Recall … Value-Based and Policy-Based RL

- Value Based
  - Learnt Value Function
  - Implicit policy (e.g. ϵ-greedy)
- Policy Based
  - No Value Function
  - Learnt Policy
- Actor-Critic
  - Learnt Value Function
  - Learnt Policy



Slides borrowed from: David Silver - Lecture 7: Policy Gradient Methods

Recall … Actor-Critic RL



Source: https://cs.wmich.edu/~trenary/files/cs5300/RLBook/node66.html

What is A2C and A3C? Why asynchronous?

A2C: Actor Advantage Critics RL
A3C: Asynchronous Actor Advantage Critics RL

Classic actor-critics method with parallel architecture

Asynchronous Methods for Deep Reinforcement Learning

Volodymyr Mnih[1] VMNIH@GOOGLE.COM
Adrià Puigdomènech Badia[1] ADRIAP@GOOGLE.COM
Mehdi Mirza[1,2] MIRZAMOM@IRO.UMONTREAL.CA
Alex Graves[1] GRAVESA@GOOGLE.COM
Tim Harley[1] THARLEY@GOOGLE.COM
Timothy P. Lillicrap[1] COUNTZERO@GOOGLE.COM
David Silver[1] DAVIDSILVER@GOOGLE.COM
Koray Kavukcuoglu[1] KORAYK@GOOGLE.COM
[1] Google DeepMind
[2] Montreal Institute for Learning Algorithms (MILA), University of Montreal

A2C = Synchronous, Deterministic A3C

Other papers that help understand this work

Policy Gradient Methods for Reinforcement Learning with Function Approximation

Richard S. Sutton, David McAllester, Satinder Singh, Yishay Mansour
AT&T Labs – Research, 180 Park Avenue, Florham Park, NJ 07932

Fundamental math proofs

Massively Parallel Methods for Deep Reinforcement Learning

Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, Shane Legg, Volodymyr Mnih, Koray Kavukcuoglu, David Silver
{ARUNSNAIR, PRAV, BLACKWELLS, CAGDASALCICEK, RORYF, ADEMARIA, DARTHVEDA, MUSTAFASUL, CBEATTIE, SVP, LEGG, VMNIH, KORAYK, DAVIDSILVER @GOOGLE.COM }
Google DeepMind, London

Parallel DQN

Dueling Network Architectures for Deep Reinforcement Learning

Ziyu Wang ZIYU@GOOGLE.COM
Tom Schaul SCHAUL@GOOGLE.COM
Matteo Hessel MTTHSS@GOOGLE.COM
Hado van Hasselt HADO@GOOGLE.COM
Marc Lanctot LANCTOT@GOOGLE.COM
Nando de Freitas NANDODEFREITAS@GMAIL.COM
Google DeepMind, London, UK

Advanced architecture

What is A2C and A3C? Why asynchronous?

Motivation

- To utilize more intuitions (tune *inner critics, e.g. self-driving cars: if the car turns left (policy), does it still in the correct route? is it closer to the destination? (critics, state value)*)
- Training data is too large to be trained efficiently (model is too large)
- Intensive uses of hardwares (GPUs)
- Make sure reducing correlations in training data by parallel actor-learners (replace experience replay buffer)

Recall… Basics Value-based function approximation

Accumulative Return:

$$R_t = \sum_{k=0}^{\inf} \gamma^k r_{t+k}$$

$$Q^\pi(s, a) = [R_t | s_t = s, a]$$

$$V^\pi(s) = [R_t | s_t = s]$$

One-step Q learning

$$L_i(\theta_i) = \mathbb{E}\left(r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i)\right)^2$$

where s' is the state encountered after state s.

Recall… Basics Policy-based RL

Williams, Ronald J. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." *Machine learning* 8.3-4 (1992): 229-256.

learning rate factor

reinforcement baseline

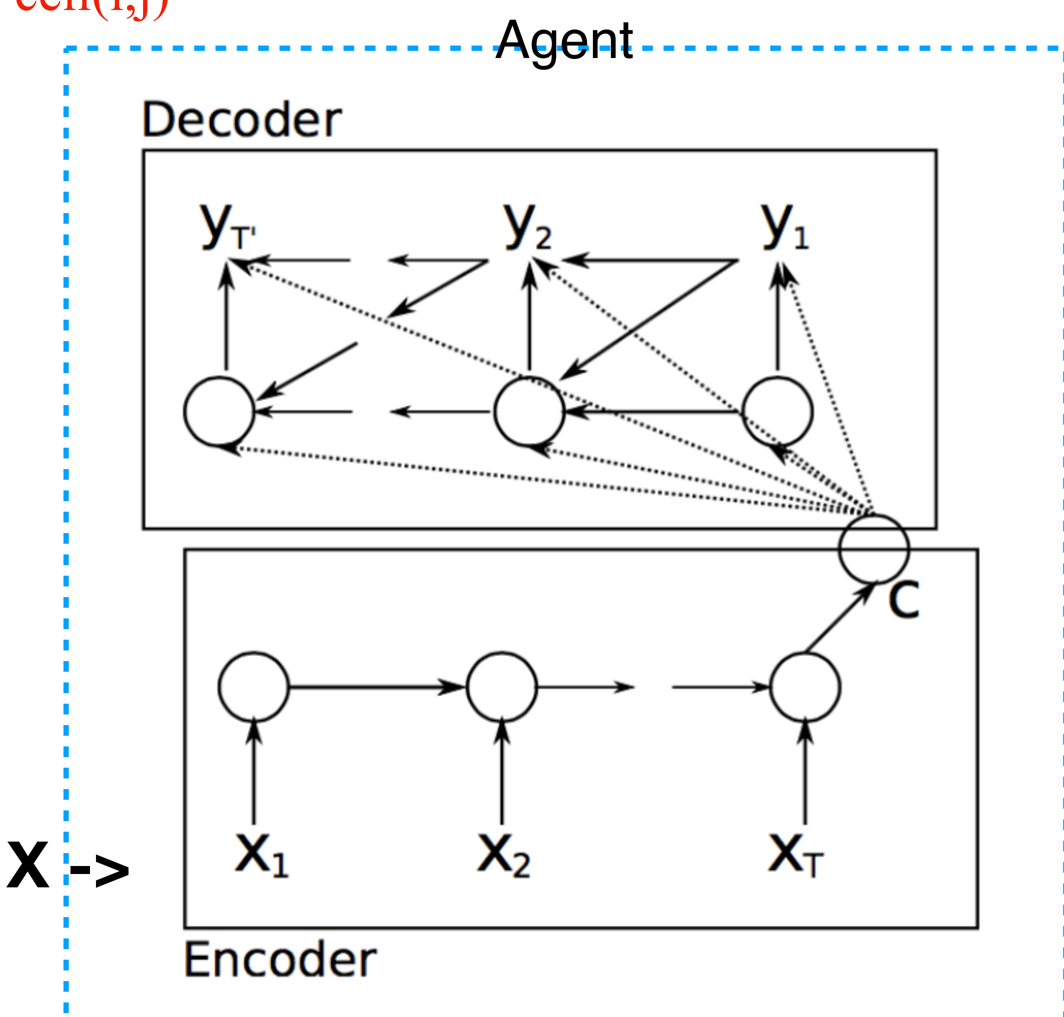
$$\Delta w_{ij} = \alpha_{ij}(r - b_{ij})e_{ij},$$

changes of weight in each cell(i,j)

loss function

$$e_{ij} = \partial \ln g_i / \partial w_{ij}$$

characteristic eligibility of w(i,j)



Actions: output sequences

RL Algorithm, compute rewards

Update Agent

Goal: learn a set of parameters that generate the best output (maximum rewards)

Could be plugged into any reward function, any networks

Definitions for A3C (A2C)

(Baseline:) $b_t(s_t) \approx V^\pi(s_t)$

Asynchronous: Parallel architectures (multi-thread execution) to improve training efficiency

Advantage: $A(a_t, s_t) = Q(a_t, s_t) - V(s_t)$,

estimation of rewards

estimation of baseline

$$A(s,a) = \underbrace{Q(s,a)}_{\text{q value for action a in state s}} - \underbrace{V(s)}_{\text{average value of that state}}$$

Change gradients update direction by A(s,a):

if A(s,a) > 0: extra rewards; update following the gradients direction

if A(s,a) < 0: current selection is not wise; follow the opposite direction

Actor-Critics:

π(s,a,Θ): Actor, policy function for telling agents how to act;

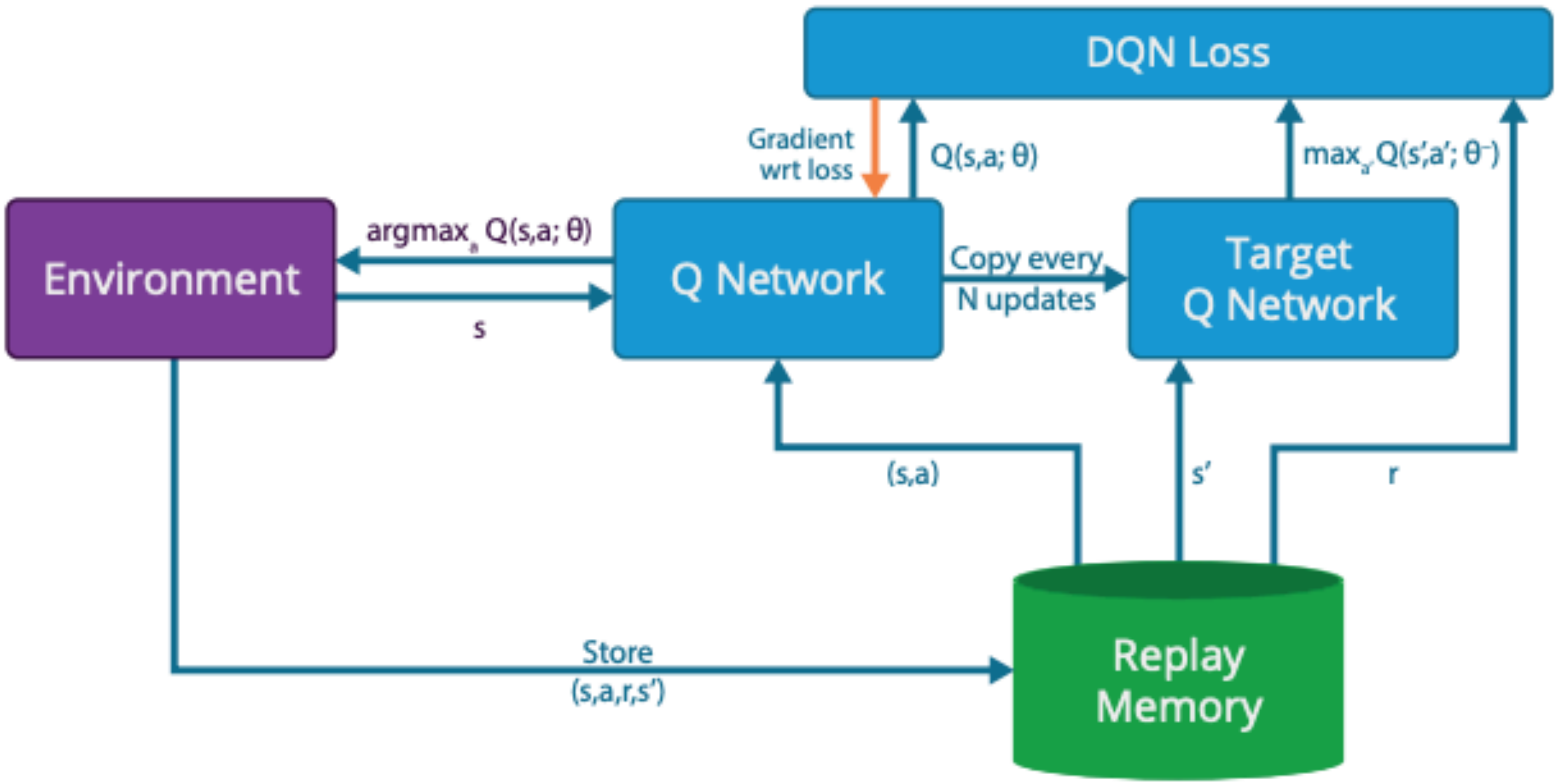
Q(s,a,w): Critics, value function for measuring how good the action is

Source: https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html#a2c

Definitions for A3C (A2C)

Asynchronous

Take DQN as example, tradition DQN architecture:

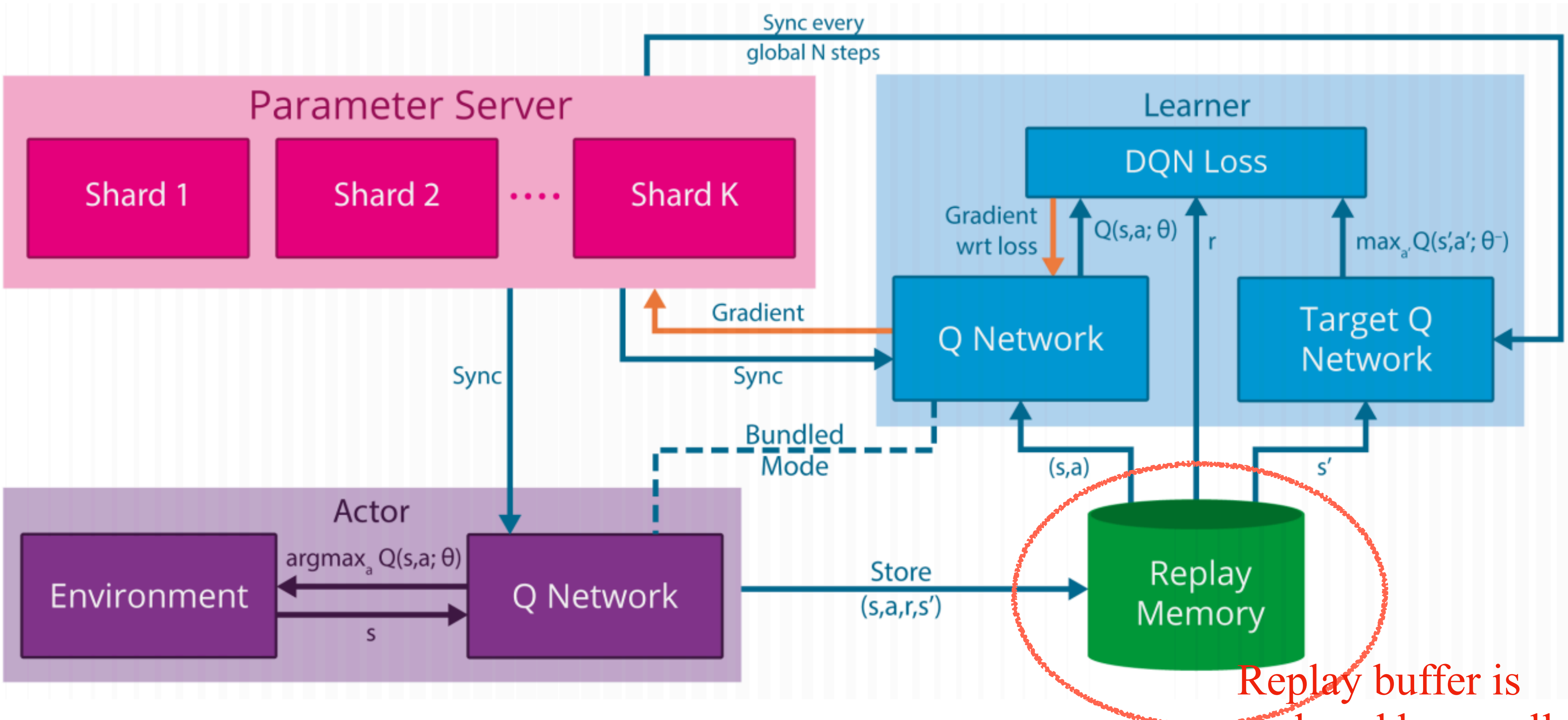


Nair, Arun, et al. "Massively parallel methods for deep reinforcement learning." *arXiv preprint arXiv:1507.04296* (2015).

Definitions for A3C (A2C)

Asynchronous

Take DQN as example, **parallel** DQN architecture proposed in previous work:



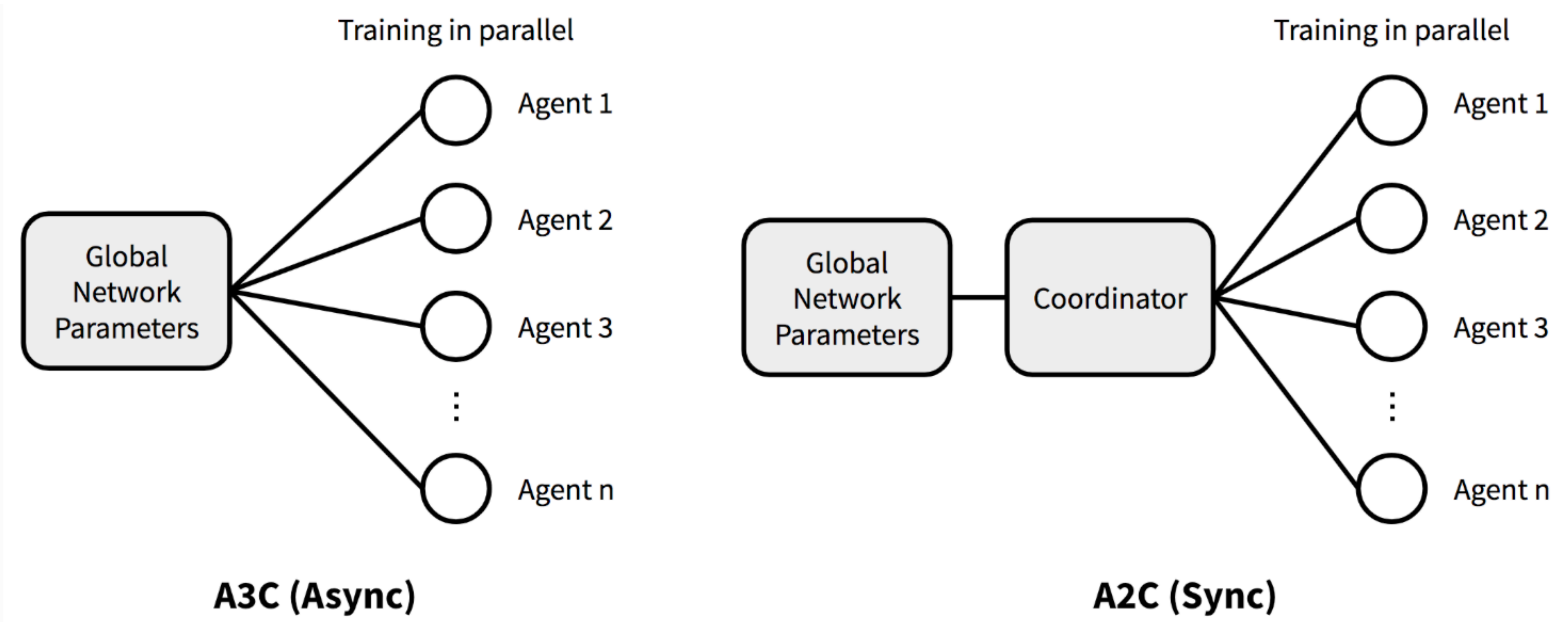
Replay buffer is replaced by parallel actor-learners in A3C

Nair, Arun, et al. "Massively parallel methods for deep reinforcement learning." *arXiv preprint arXiv:1507.04296* (2015).

Definitions for A3C (A2C)

Asynchronous

A3C VS A2C:



Source: https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html#a2c

Definitions for A3C (A2C)

Asynchronous One-step Q-learning

Algorithm 1 Asynchronous one-step Q-learning - pseudocode for each actor-learner thread.

// Assume global shared θ, θ^{-}, and counter $T = 0$.
Initialize thread step counter $t \leftarrow 0$
Initialize target network weights $\theta^{-} \leftarrow \theta$
Initialize network gradients $d\theta \leftarrow 0$
Get initial state s
repeat
  Take action a with ϵ-greedy policy based on $Q(s, a; \theta)$
  Receive new state s' and reward r

$$y = \begin{cases} r & \text{for terminal } s' \\ r + \gamma \max_{a'} Q(s', a'; \theta^{-}) & \text{for non-terminal } s' \end{cases}$$

  Accumulate gradients wrt θ: $d\theta \leftarrow d\theta + \frac{\partial (y - Q(s,a;\theta))^2}{\partial \theta}$
  $s = s'$
  $T \leftarrow T + 1$ and $t \leftarrow t + 1$
  if $T \mod I_{target} == 0$ **then**
    Update the target network $\theta^{-} \leftarrow \theta$
  end if
  if $t \mod I_{AsyncUpdate} == 0$ or s is terminal **then**
    Perform asynchronous update of θ using $d\theta$.
    Clear gradients $d\theta \leftarrow 0$.
  end if
until $T > T_{max}$

Definitions for A3C (A2C)

Advantage: $\nabla_{\theta'} \log \pi(a_t|s_t;\theta')A(s_t,a_t;\theta,\theta_v)$

$A(s_t,a_t;\theta,\theta_v)$ is the estimate of $\sum_{i=0}^{k-1}\gamma^i r_{t+i} + \gamma^k V(s_{t+k};\theta_v) - V(s_t;\theta_v)$

Algorithm S3 Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

// Assume global shared parameter vectors θ and θ_v and global shared counter $T = 0$
// Assume thread-specific parameter vectors θ' and θ'_v
Initialize thread step counter $t \leftarrow 1$
repeat
  Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
  Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$
  $t_{start} = t$
  Get state s_t
  repeat
    Perform a_t according to policy $\pi(a_t|s_t;\theta')$
    Receive reward r_t and new state s_{t+1}
    $t \leftarrow t + 1$
    $T \leftarrow T + 1$
  until terminal s_t **or** $t - t_{start} == t_{max}$
  $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t,\theta'_v) & \text{for non-terminal } s_t \text{// Bootstrap from last state} \end{cases}$
  for $i \in \{t-1,\dots,t_{start}\}$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients wrt θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\theta'_v))$
    Accumulate gradients wrt θ'_v: $d\theta_v \leftarrow d\theta_v + \partial\left(R - V(s_i;\theta'_v)\right)^2/\partial\theta'_v$
  end for
  Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.
until $T > T_{max}$

Definitions for A3C (A2C)

Advantage: $\nabla_{\theta'} \log \pi(a_t|s_t;\theta')A(s_t,a_t;\theta,\theta_v)$

$A(s_t,a_t;\theta,\theta_v)$ is the estimate of $\sum_{i=0}^{k-1}\gamma^i r_{t+i} + \gamma^k V(s_{t+k};\theta_v) - V(s_t;\theta_v)$

why not Q?

Recall: $A(a_t,s_t) = Q(a_t,s_t) - V(s_t)$

estimation of rewards

estimation of baseline

Of course you could compute Q (as in Dueling DQN), but …

Recall: $Q(s_t,a_t) = r_t + \sum_{k=0}^{\inf} \gamma^k maxQ_{t+k}(s_{t+k},a_{t+k})$

Looking ahead of which action to take that leads to the best reward estimates (n-step Q)

Instead of looking ahead, A3C use its own learned critics (td-error) and prediction against the critic

$$\underbrace{\sum_{i=0}^{k-1}\gamma^i r_{t+i}} + \underbrace{\gamma^k V(s_{t+k};\theta_v)} - \underbrace{V(s_t;\theta_v)}$$

How many rewards actually are earned?

If continue, how many rewards one could earn? (Prediction)

Current estimate (looking back each row)

- How am I performing so far (MONTE CARLO)?
- How will I perform if I continue this way?
- How did I and will I perform compared to my expectation (estimation)?

Source: https://hackernoon.com/intuitive-rl-intro-to-advantage-actor-critic-a2c-4ff545978752

Definitions for A3C (A2C)

Advantage: TD - Error (Example of Richard Sutton driving home)

State	*Elapsed Time (minutes)*	*Predicted Time to Go*	*Predicted Total Time*
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

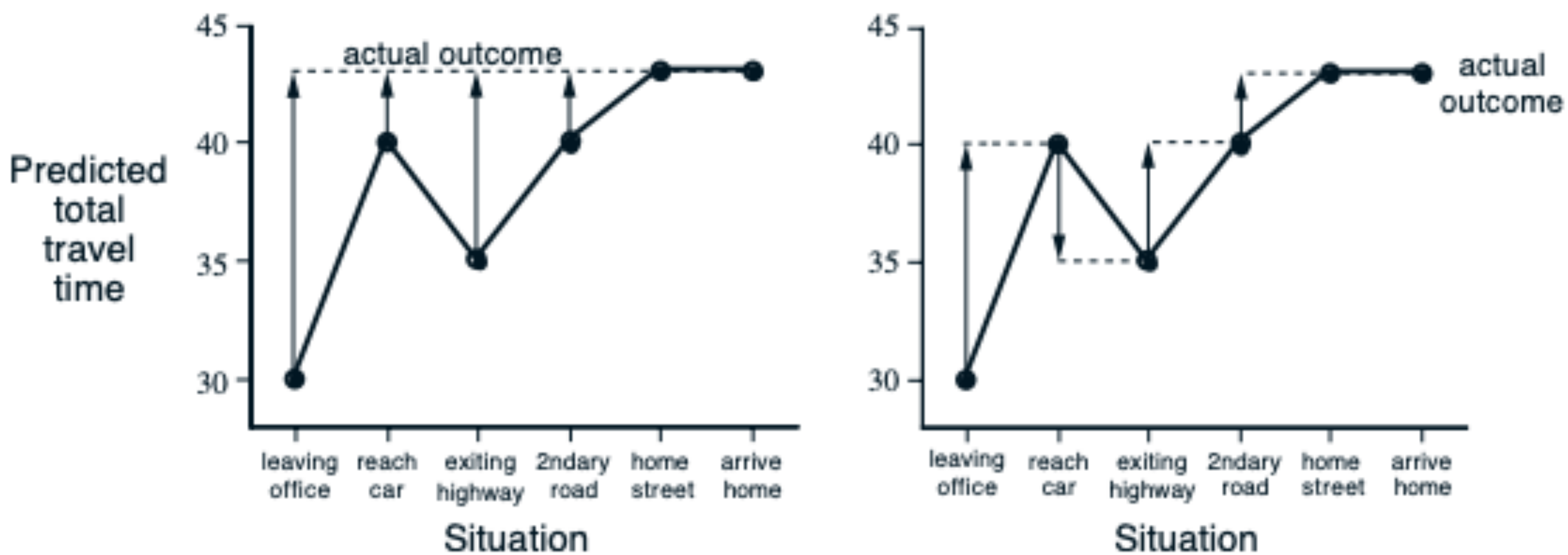


Figure 6.1: Changes recommended in the driving home example by Monte Carlo methods (left) and TD methods (right).

Source:Sutton, Richard S., and Andrew G. Barto. *Introduction to reinforcement learning*. Vol. 135. Cambridge: MIT press, 1998.

Definitions for A3C (A2C)

Actor-Critics:

for $i \in \{t-1, \ldots, t_{start}\}$ **do**

$R \leftarrow r_i + \gamma R$

Accumulate gradients wrt θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\theta'_v))$

Accumulate gradients wrt θ'_v: $d\theta_v \leftarrow d\theta_v + \partial\left(R - V(s_i;\theta'_v)\right)^2 / \partial\theta'_v$

end for

$$\nabla_{\theta'} \log \pi(a_t|s_t;\theta')(R_t - V(s_t;\theta_v)) + \beta \nabla_{\theta'} H(\pi(s_t;\theta'))$$

Entropy term that helps exploration

Experiments and Results

Method	Training Time	Mean	Median
DQN	8 days on GPU	121.9%	47.5%
Gorila	4 days, 100 machines	215.2%	71.3%
D-DQN	8 days on GPU	332.9%	110.9%
Dueling D-DQN	8 days on GPU	343.8%	117.1%
Prioritized DQN	8 days on GPU	463.6%	127.6%
A3C, FF	1 day on CPU	344.1%	68.2%
A3C, FF	4 days on CPU	496.8%	116.6%
A3C, LSTM	4 days on CPU	623.0%	112.6%

Table 1. Mean and median human-normalized scores on 57 Atari games using the human starts evaluation metric. Supplementary Table SS3 shows the raw scores for all games.

Asynchronous Methods for Deep Reinforcement Learning: TORCS

Sonic (not from this work)

Take-away

- It doesn't mean experience replay is not useful even though A3C didn't use it; just more expensive

- A3C is able to handle complicated tasks, e.g. car racing, it combines both forward view (prediction) and backward view (eligible traceability)

- Actor-critic will be the mainstream of future reinforcement learning!

References

- Mnih, Volodymyr, et al. "Asynchronous methods for deep reinforcement learning." *International conference on machine learning*. 2016.

- Sutton, Richard S., et al. "Policy gradient methods for reinforcement learning with function approximation." *Advances in neural information processing systems*. 2000.

- Nair, Arun, et al. "Massively parallel methods for deep reinforcement learning." *arXiv preprint arXiv:1507.04296* (2015).Make sure reducing correlations in training data by parallel actor-learners (replace experience replay buffer)

- Wang, Ziyu, et al. "Dueling network architectures for deep reinforcement learning." *arXiv preprint arXiv:1511.06581* (2015).

- https://hackernoon.com/intuitive-rl-intro-to-advantage-actor-critic-a2c-4ff545978752

- https://medium.freecodecamp.org/an-intro-to-advantage-actor-critic-methods-lets-play-sonic-the-hedgehog-86d6240171d

- Source: https://cs.wmich.edu/~trenary/files/cs5300/RLBook/node66.html

- Sutton, Richard S., and Andrew G. Barto. *Introduction to reinforcement learning*. Vol. 135. Cambridge: MIT press, 1998.