Extended DQN

Extended DQN

1. DQN
2. Double DQN
3. Prioritized Experience Replay
4. Dueling Network Architectures
5. Noisy Network for Exploration

문제점 : 불안정한 학습 환경

However reinforcement learning presents several challenges from a deep learning perspective. Firstly, most successful deep learning applications to date have required large amounts of hand-labelled training data. RL algorithms, on the other hand, must be able to learn from a scalar reward signal that is frequently sparse, noisy and delayed. The delay between actions and resulting rewards, which can be thousands of timesteps long, seems particularly daunting when compared to the direct association between inputs and targets found in supervised learning. Another issue is that most deep learning algorithms assume the data samples to be independent, while in reinforcement learning one typically encounters sequences of highly correlated states. Furthermore, in RL the data distribution changes as the algorithm learns new behaviours, which can be problematic for deep learning methods that assume a fixed underlying distribution.

- Delayed feedback
- Supervised Learning에서 가정하는 데이터간 i.i.d 조건 파괴 (데이터간 correlation)
- Greedy한 업데이트로 value가 급격하게 바뀜(greedy policy improve)
- 이로 인해 데이터의 분포가 계속 변한다.

+ Neural Network를 사용하게 되면 학습에 수렴이 보장되지 않다는 문제

* i.i.d : Independent and Identically Distributed Data

Mnih, V., Kavukcuoglu, K., Silver, D.,et al. (2013). Playing atari with deep reinforcement learning

어떻게 해결할 것인가?

1) Target Network와 Prediction Network의 분리

2) Experience Replay를 위한 Replay Buffer 구성

1) Target Network와 Prediction Network의 분리

$$L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[(y_i - Q(s,a;\theta_i))^2\right]$$ Loss Function : Target - Prediction

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot);s'\sim\mathcal{E}}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i)\right)\nabla_{\theta_i} Q(s,a;\theta_i)\right]$$

Target Network

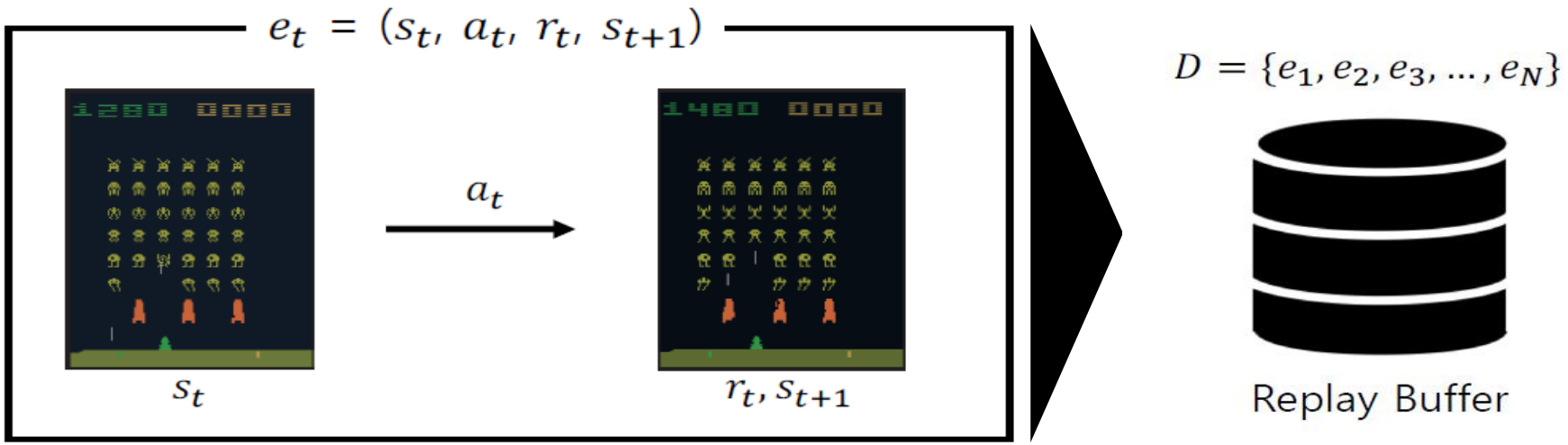
Prediction Network

Network를 두개로 나누어 급격하게 변하는 value를 조금이나마 진정시킨다.

이렇게 한다고 100% 해결되는 건 아닌 것 같다.

Mnih, V., Kavukcuoglu, K., Silver, D.,et al. (2013). Playing atari with deep reinforcement learning

2) Experience Replay를 위한 Replay Buffer 구성



randomizes over the data, thereby removing correlations in the observation sequence and smoothing over changes in the data distribution

- 시뮬레이션내에서 매 틱마다 생기는 $e_t = (s_t, a_t, r_t, s_{t+1})$ Replay Buffer에 저장하고,
- 저장된 Replay Buffer에서 batch sampling을 통해 minibatch 학습시킨다.

"DQN 논문 리뷰" PangyuLab, YouTube Channel
Mnih, V., Kavukcuoglu, K., Silver, D.,et al. (2013). Playing atari with deep reinforcement learning

세부 학습 Detail

1. 원래 Input size는 210*160 인데 84*84 로 크기를 줄였다.

2. Input에 들어오는 이미지는 가장 최근 4장을 쌓아서 제공하였다. 즉 4*84*84 가 input.

3. 모델 구조는 3개의 conv layer 이후 2개의 fc layer로 구성되었다.

4. Atari 2600의 49개의 게임에 대해서 동일한 모델 구조, 동일한 hyperparameter를 사용하여 학습하였다.

5. 게임마다 reward scale이 달라서 모든 양의 리워드는 +1, 모든 음의 리워드는 -1로 clipping 하였다.

6. Behaviour policy는 epsilon greedy를 썼으며 epsilon은 1.0에서 시작하여 100만 frame 동안 0.1까지 선형으로 줄어든다. 이후는 0.1로 고정.(exploration에 신경씀)

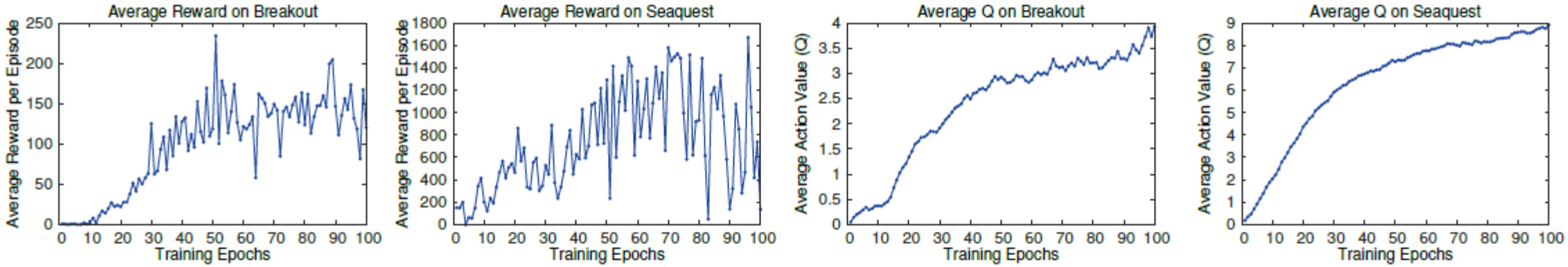
7. 총 5천만 frame 을 학습에 썼다(게임 플레이 시간으로 38일).

8. Replay memory 는 최신 100만 frame을 사용.

"DQN 논문 리뷰" PangyuLab, YouTube Channel

Mnih, V., Kavukcuoglu, K., Silver, D.,et al. (2013). Playing atari with deep reinforcement learning

Conclusion



divergence issues in any of our experiments. This suggests that, despite lacking any theoretical convergence guarantees, our method is able to train large neural networks using a reinforcement learning signal and stochastic gradient descent in a stable manner.

This paper introduced a new deep learning model for reinforcement learning, and demonstrated its ability to master difficult control policies for Atari 2600 computer games, using only raw pixels as input. We also presented a variant of online Q-learning that combines stochastic minibatch updates with experience replay memory to ease the training of deep networks for RL. Our approach gave state-of-the-art results in six of the seven games it was tested on, with no adjustment of the architecture or hyperparameters.

Mnih, V., Kavukcuoglu, K., Silver, D.,et al. (2013). Playing atari with deep reinforcement learning

2. Double DQN

기존 Q-Algorithms의 문제점

$$Y_t^{\mathrm{Q}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t) . \qquad (2)$$

$$Y_t^{\mathrm{DQN}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t^-) . \qquad (3)$$

The max operator in standard Q-learning and DQN, in (2) and (3), uses the same values both to select and to evaluate an action. This makes it more likely to select overestimated values, resulting in overoptimistic value estimates. To

Action을 Selection할 때, Action을 Evaluate할때 모두 같은 value function을 사용한다.

이것이 overestimation(overoptimistic value estimates)를 만들어낸다.

※ 왜 overestimate 될까?

하나의 Q로 action을 뽑고, 이걸 가지고 Q를 평가하게 되면 자칫 잘못된 방향으로 학습하게되고,

결국 나중에 제대로 optimal한 방향으로 돌아오지 못하게 된다.

Van Hasselt, H., Guez, A., & Silver, D. (2016, March). Deep reinforcement learning with double q-learning.

2. Double DQN

Double DQN : Reduce the Overestimates

$$Y_t^{\text{DoubleQ}} \equiv R_{t+1} + \gamma Q(S_{t+1}, \underset{a}{\text{argmax}}\, Q(S_{t+1}, a; \theta_t); \theta'_t)$$. Target을 action selection과 action evaluation으로 나눈다.

빠르게 업데이트하면서 episode를 뽑아내고, 현재까지 추정한 Q중에서 가장 좋은 action을 고르는 network **(action selection)**

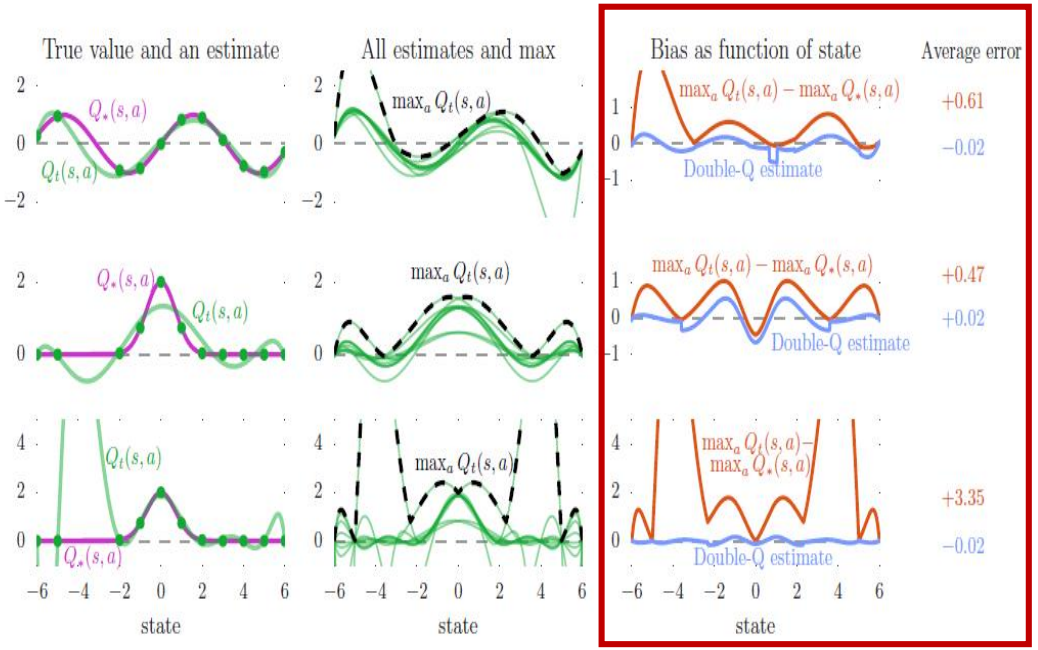
상대적으로 느리게 업데이트하면서 Q가 과도하게 커지는 것을 방지하는 역할의 network **(action evaluation)**

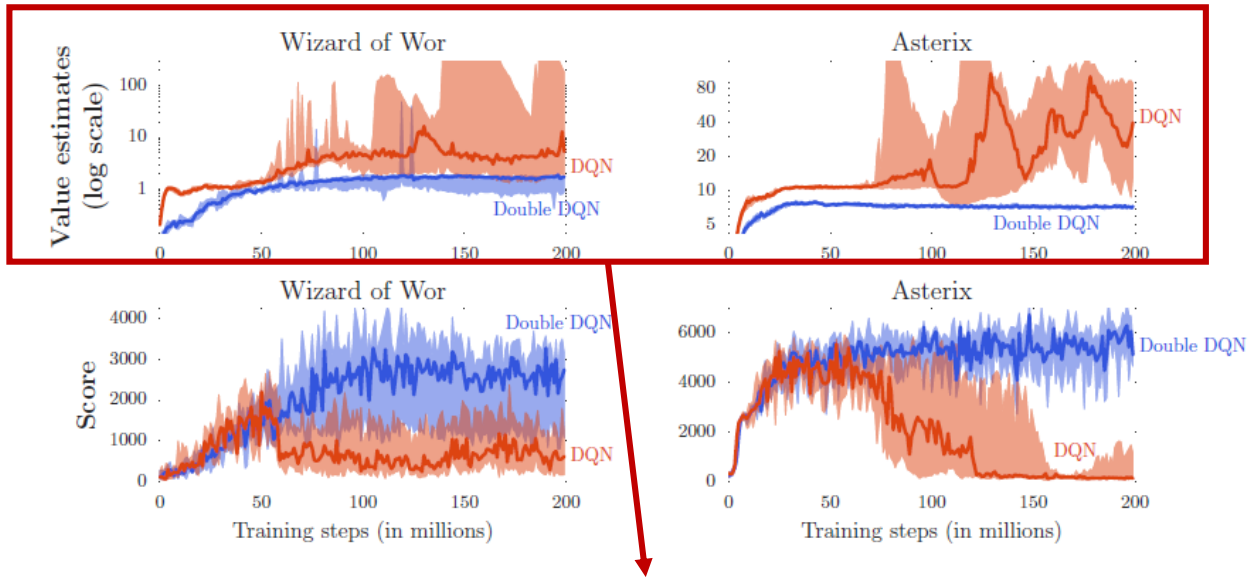
기존 Q-Learning과 DQN은 항상 Q의 최댓값을 이용하여 업데이트가 진행되었기 때문에,
한번 잘못 Q를 추정하기 시작하면 그 에러가 계속해서 커지게 되버린다.

하지만, Double DQN은 상대적으로 network의 parameter가 천천히 업데이트되기 때문에 estimation error의 전파가 빠르지 않고, 안정적인 학습이 가능해진다.

[https://brunch.co.kr/@kakao-it/102] 딥러닝 연구의 현재와 미래 part2, 카카오AI리포트(2017)
Van Hasselt, H., Guez, A., & Silver, D. (2016, March). Deep reinforcement learning with double q-learning.

Double DQN : Reduce the Overestimates





Bias (estimate error)가 0에 가깝다.

Estimation이 갑자기 확 치솟지 않는다.

※ Target의 업데이트부분을 제외하고, 전체적인 pseudo code는 DQN과 같다.

Van Hasselt, H., Guez, A., & Silver, D. (2016, March). Deep reinforcement learning with double q-learning.

3. Prioritized Experience Replay

Main Idea : Experience Replay + Prioritized

> Experience replay lets online reinforcement learning agents remember and reuse experiences from the past. In prior work, experience transitions were uniformly sampled from a replay memory. However, this approach simply replays transitions at the same frequency that they were originally experienced, regardless of their significance. In this paper we develop a framework for prioritizing experience, so as to replay important transitions more frequently, and therefore learn more efficiently. We use prioritized experience replay in Deep Q-Networks (DQN), a reinforcement learning algorithm that achieved human-level performance across many Atari games. DQN with prioritized experience replay achieves a new state-of-the-art, outperforming DQN with uniform replay on 41 out of 49 games.

- Agent는 특정 몇 개의 transitions로부터 더 많이 배운다.
- 즉, Experience마다 학습효과에 차이가 있다.
- 그러므로 중요한 Experience를 더 많이 뽑히게 해서 효과적으로 학습시키자!

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

3. Prioritized Experience Replay

어떻게?

In particular, we propose to more frequently replay transitions with high expected learning progress, as measured by the magnitude of their temporal-difference (TD) error. This prioritization can lead to a loss of diversity, which we alleviate with stochastic prioritization, and introduce bias, which we correct with importance sampling. Our resulting algorithms are robust and scalable, which we

1) TD error

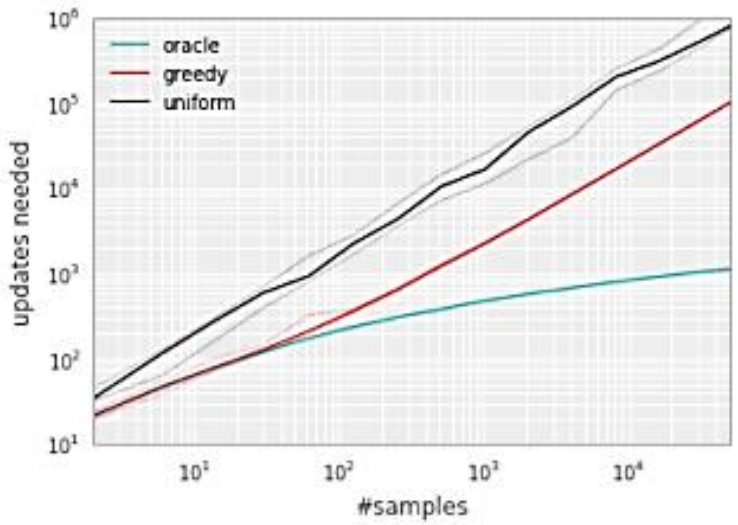
2) Stochastic Prioritization

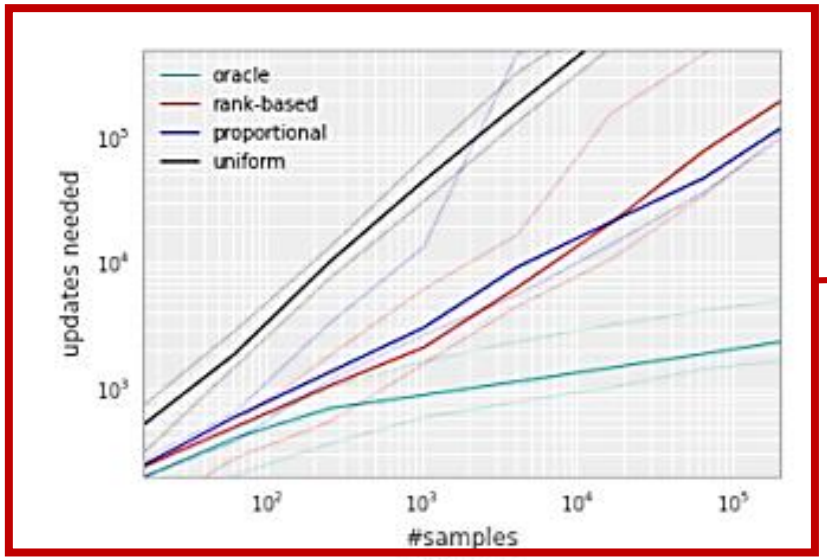
3) Importance Sampling

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

3. Prioritized Experience Replay

Stochastic Prioritization?





Stochastic Prioritization

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

- TD error가 클수록 자주 뽑히게 prioritized시킨다.
- 하지만 크기만 가지고 Greedy하게 뽑으면 experience간 편향이 심해진다.
- 기존 noise + approximation error가 추가된다.
- 결국, 일부 experience에 치우쳐 업데이트되고, overfitting이 발생한다.
- 이를 방지하기 위해 Stochastic sampling method 도입

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Weighted Importance Sampling?

The estimation of the expected value with stochastic updates relies on those updates corresponding to the same distribution as its expectation. Prioritized replay introduces bias because it changes this distribution in an uncontrolled fashion, and therefore changes the solution that the estimates will converge to (even if the policy and state distribution are fixed). We can correct this bias by using importance-sampling (IS) weights

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^{\beta}$$

- Prioritization으로 생기는 bias를 통제하고
- Non-uniform sampling을 하는 것에 대한 보상개념으로 weighted Importance Sampling 사용
- 추가적으로 w_i는 안정적인 업데이트를 위해 (1 / max w_i)로 normalize시킨다.

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

3. Prioritized Experience Replay

Pseudo Code

Algorithm 1 Double DQN with proportional prioritization

Input: minibatch k, step-size η, replay period K and size N, exponents α and β, budget T.
Initialize replay memory $\mathcal{H} = \emptyset$, $\Delta = 0$, $p_1 = 1$
Observe S_0 and choose $A_0 \sim \pi_\theta(S_0)$
for $t = 1$ **to** T **do**
  Observe S_t, R_t, γ_t
  Store transition $(S_{t-1}, A_{t-1}, R_t, \gamma_t, S_t)$ in $\mathcal{H}$ with maximal priority $p_t = \max_{i<t} p_i$
  if $t \equiv 0 \mod K$ **then**
    for $j = 1$ **to** k **do**
      Sample transition $j \sim P(j) = p_j^\alpha / \sum_i p_i^\alpha$
      Compute importance-sampling weight $w_j = (N \cdot P(j))^{-\beta} / \max_i w_i$
      Compute TD-error $\delta_j = R_j + \gamma_j Q_{\text{target}}(S_j, \arg\max_a Q(S_j, a)) - Q(S_{j-1}, A_{j-1})$
      Update transition priority $p_j \leftarrow |\delta_j|$
      Accumulate weight-change $\Delta \leftarrow \Delta + w_j \cdot \delta_j \cdot \nabla_\theta Q(S_{j-1}, A_{j-1})$
    end for
    Update weights $\theta \leftarrow \theta + \eta \cdot \Delta$, reset $\Delta = 0$
    From time to time copy weights into target network $\theta_{\text{target}} \leftarrow \theta$
  end if
  Choose action $A_t \sim \pi_\theta(S_t)$
end for

Weight normalizing

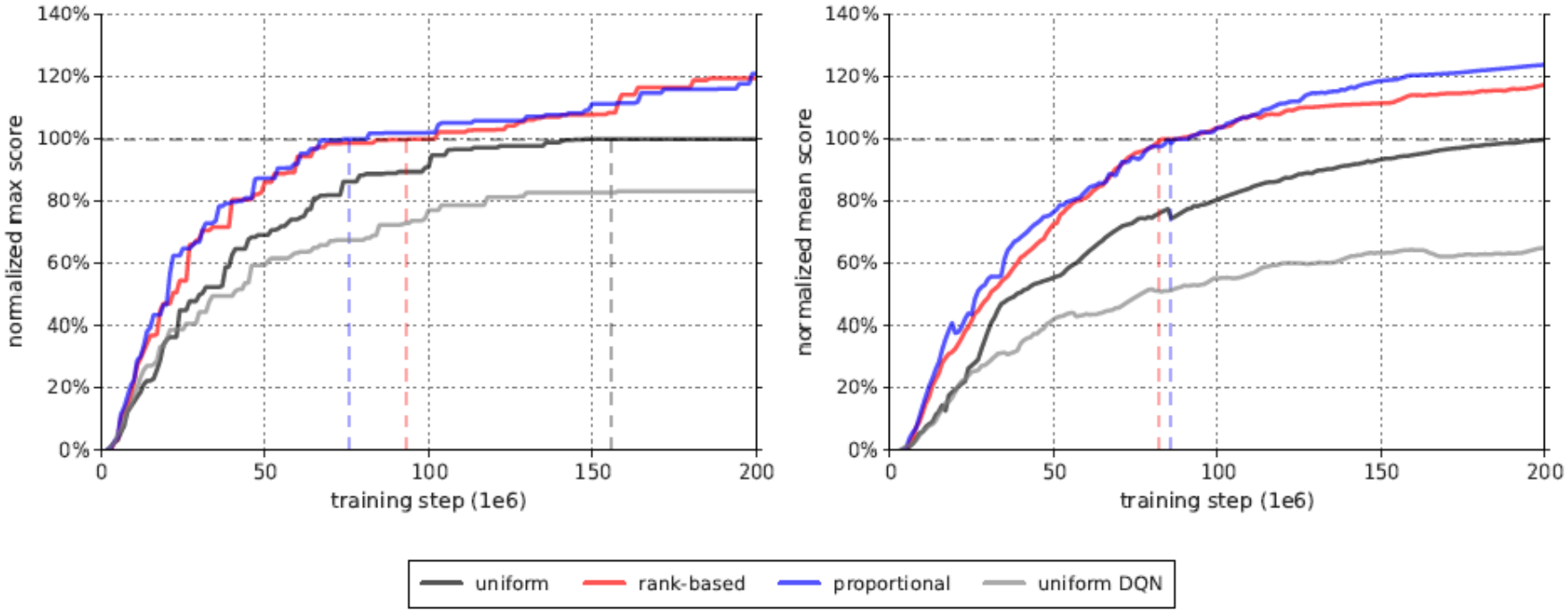
∇_θLoss sum

Update parameter θ

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

3. Prioritized Experience Replay

Performance : 그냥 DQN, Double DQN보다 성능이 좋다.



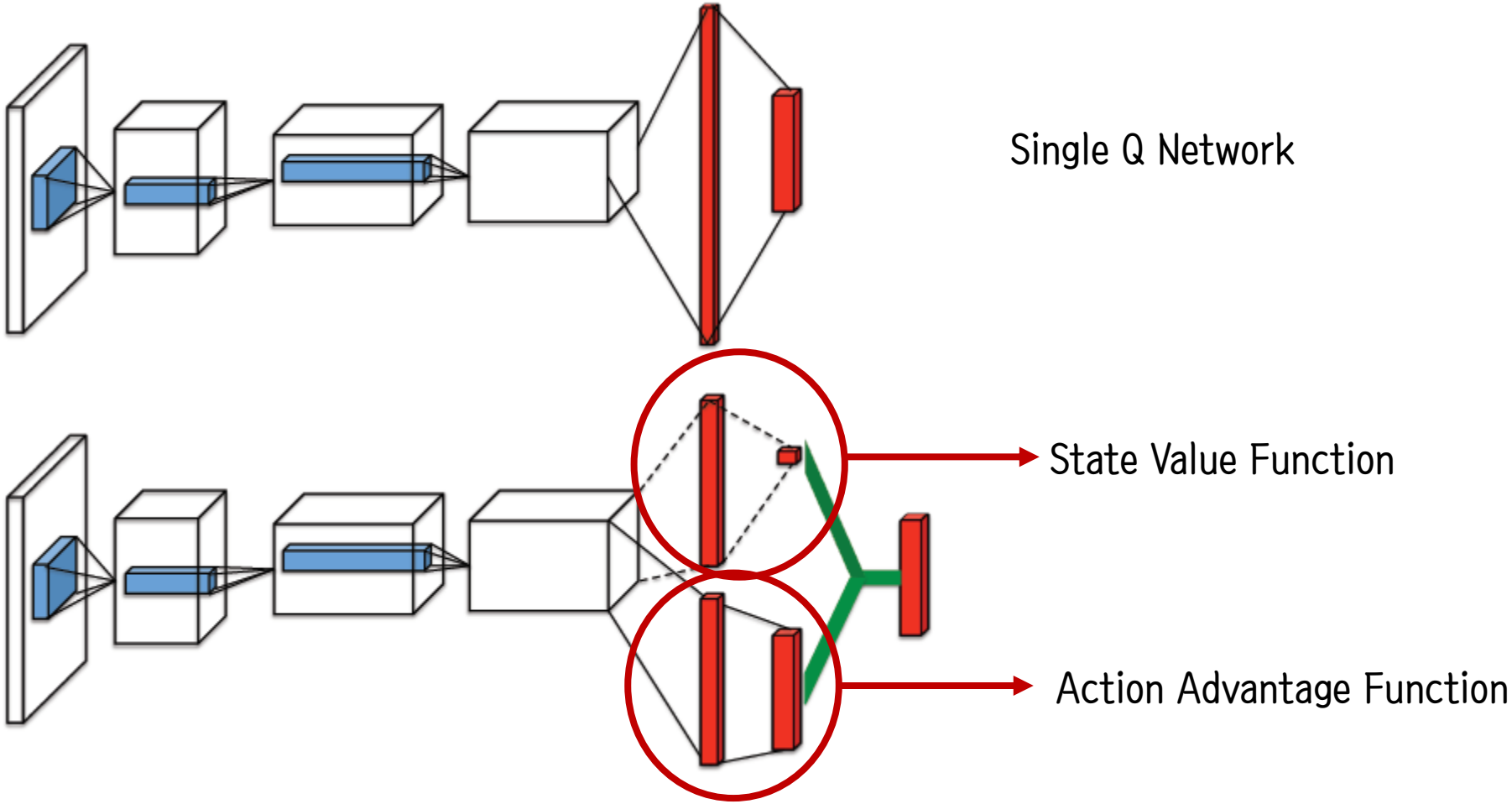
	DQN		Double DQN (tuned)		
	baseline	rank-based	baseline	rank-based	proportional
Median	48%	106%	111%	113%	128%
Mean	122%	355%	418%	454%	551%
> baseline	–	41	–	38	42
> human	15	25	30	33	33
# games	49	49	57	57	57

Table 1: Summary of normalized scores. See Table 6 in the appendix for full results.

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

4. Dueling Network Architectures

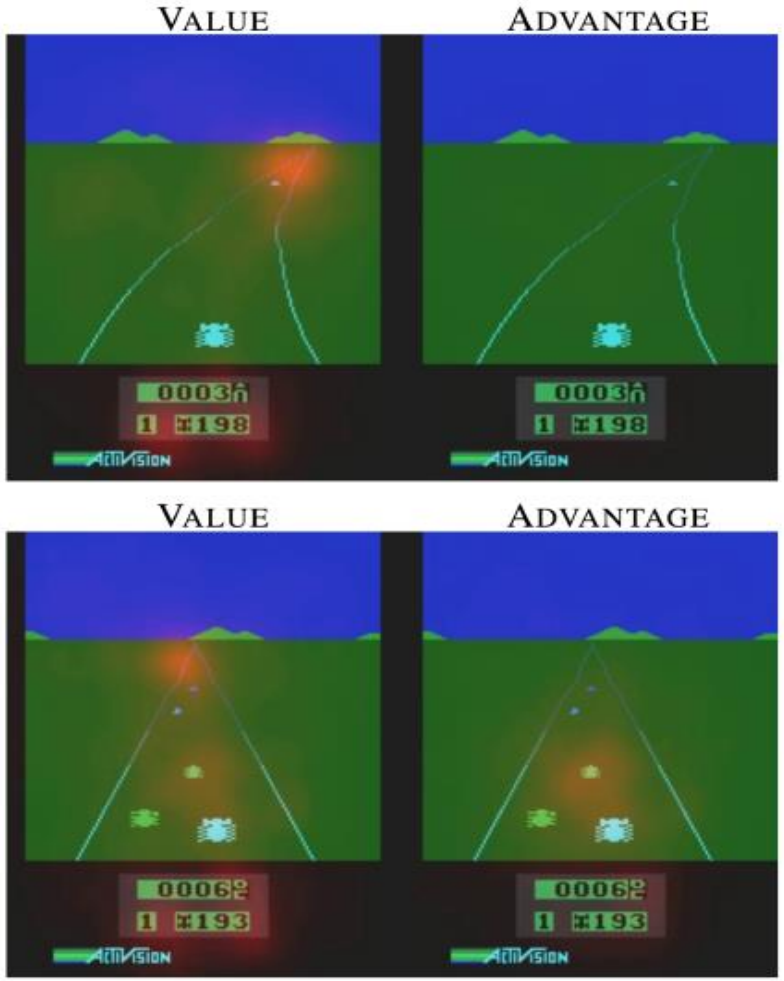
Dueling Architectures?



Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & De Freitas, N. (2015). Dueling network architectures for deep reinforcement learning.

4. Dueling Network Architectures

Dueling Architectures?



In most of states, the choice of action has no affect on what happens

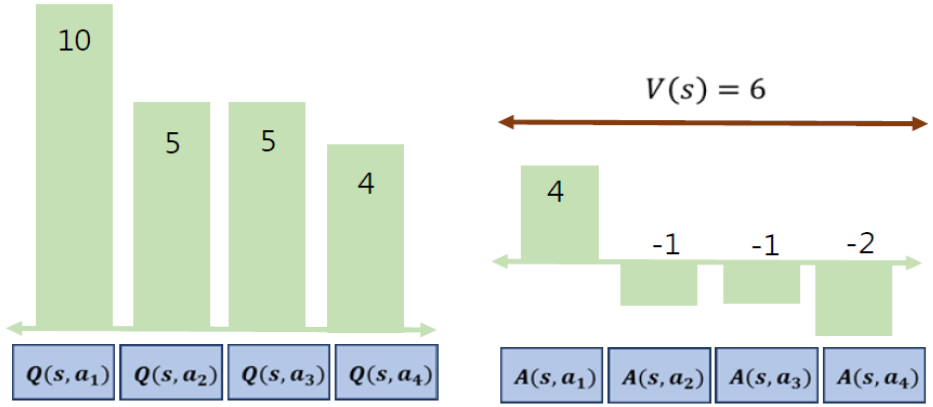
Not pay much attention when there are **no cars** in front

Attention on car immediately in front making **its choice of action very relevant**

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & De Freitas, N. (2015). Dueling network architectures for deep reinforcement learning.

4. Dueling Network Architectures

Main Idea : Decouple the Q



$$Q(s,a) = V(s) + A(s,a)$$

최선의 action과 실제 action의 보상차이

최선의 action으로 얻은 보상의 합

- 매순간 적절한 action을 선택에 대한 Value를 추정하는 건 불필요하다는 게 핵심주장.
- Action이 environment에 영향이 없는 state에서 유용하다.(오른쪽으로 움직이든, 왼쪽으로 움직이든)
- 그래서 Fully Connected Layer를 두 줄기로 나누었다.
- 하나는 State Value Function을 추정하여 현재의 state가 좋은지 나쁜지를 보겠다는 것이고,
- 다른 하나로는 Advantage를 추정하여 어떤 action을 고를지 생각해보겠다는 것이다.

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & De Freitas, N. (2015). Dueling network architectures for deep reinforcement learning.

4. Dueling Network Architectures

How to Concatenation

Cannot guarantee A(s,a) learns advantages

Three methods

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\alpha,\beta) + A(s,a;\theta,\alpha,\beta)$$

Kind of Regularization

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\alpha,\beta) + \left(A(s,a;\theta,\alpha,\beta) - \max A(s,a';\theta,\alpha,\beta)\right)$$

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\alpha,\beta) + \left(A(s,a;\theta,\alpha,\beta) - \frac{1}{m}A(s,a;\theta,\alpha,\beta)\right)$$

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & De Freitas, N. (2015). Dueling network architectures for deep reinforcement learning.

Main Idea : Distributional Value Function

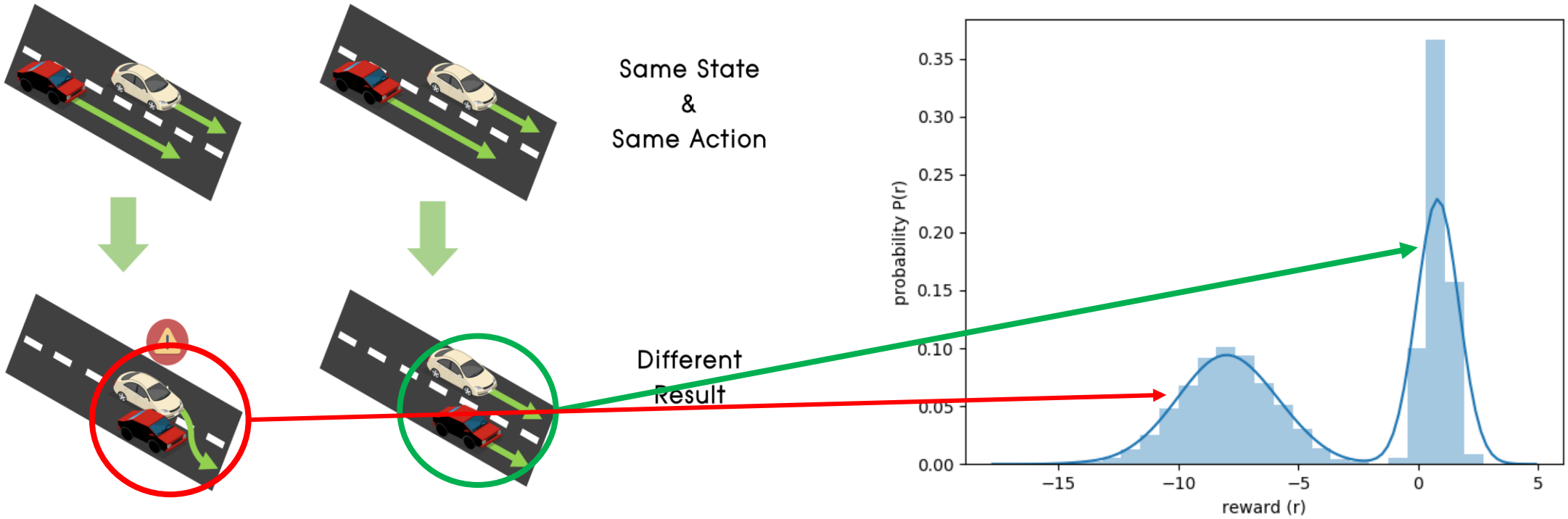
Bellman Equation: $Q(s,a) = R(s,a) + \gamma Q(s',a')$

Distributional Bellman Equation: $Z(s,a) = R(s,a) + \gamma Z(s',a')$

Bellman Equation에서 Deterministic한 value, 즉 하나의 scalar가 아닌,
Value자체의 분포를 활용하겠다.

https://reinforcement-learning-kr.github.io
Bellemare, M. G., Dabney, W., & Munos, R. (2017, August). A distributional perspective on reinforcement learning.

이러한 개념이 왜 필요하지? **Multi-Modality!**



확률적인 환경에서는 상황에 따라 받을 확률이 높은 reward가 더 다양할 수도 있다.

이를 multimodal distribution이라 하고, 이것이 더 안정적으로 학습하도록 한다.

https://reinforcement-learning-kr.github.io

Bellemare, M. G., Dabney, W., & Munos, R. (2017, August). A distributional perspective on reinforcement learning.

Algorithmic Detail(1) : Q value

$$Q(x_t, a_t) = \sum_i z_i p_i(x_t, a_t)$$

N: supports의 수, V_{min}: support의 최소값, Δz: support간 간격

z_i: support $(z_i = V_{min} + i\Delta z : \quad 0 \le i \le N)$

p_i: support z_i에 대한 확률

- Value Distribution의 기댓값으로 구한다.
- 기존 scalar보다 정확한 분포를 추정하여 얻은 값이기에 좀 더 정확한 예측이다.
- 이를 이용하여 optimal한 action을 구한다.

https://reinforcement-learning-kr.github.io

Bellemare, M. G., Dabney, W., & Munos, R. (2017, August). A distributional perspective on reinforcement learning.

Algorithmic Detail(2) : Loss Function은 Cross Entropy

$$Loss = -\sum_{i} m_i \log p_i(x_t, a_t)$$

N: supports의 수 $(0 \leq i \leq N)$

m_i: target distribution

p_i: support z_i에 대한 확률 (network의 output)

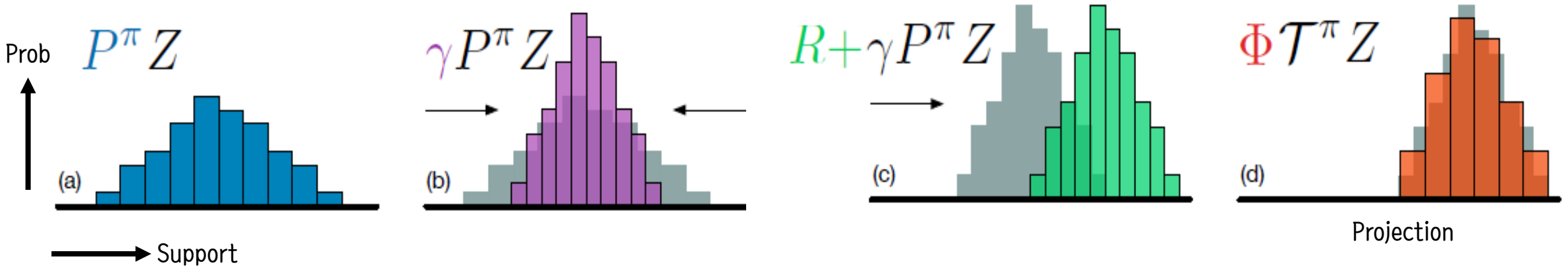
Target Distribution과 Estimated Distribution의 차이를 최소로 만드는 것이 이 알고리즘의 핵심.

그래서 Cross Entropy를 Loss Function으로 사용한다.

https://reinforcement-learning-kr.github.io

Bellemare, M. G., Dabney, W., & Munos, R. (2017, August). A distributional perspective on reinforcement learning.

Algorithmic Detail(3) : Target Distribution



a. Support의 최대 최소값을 미리 정의해주고 R + γ * z를 하여 Support를 업데이트해준다.

b. 이대로 학습시키면 분포의 범위가 맞지 않기 때문에 Support를 기존과 같게 맞춰줘야 한다.(Projection)

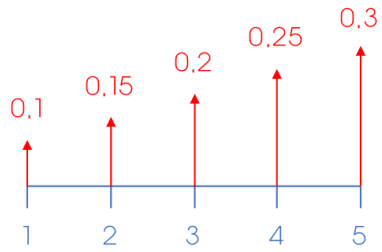
c. Projection이후 Target Distribution과 Estimated Distribution의 Cross Entropy 계산!

https://reinforcement-learning-kr.github.io

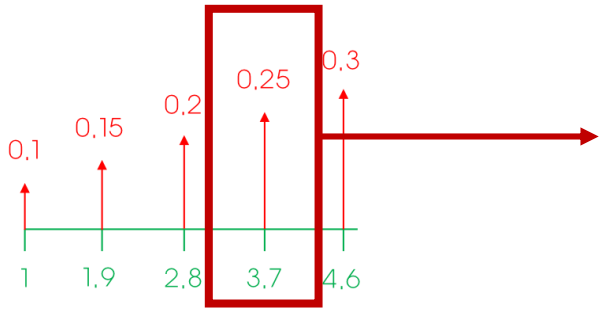
Bellemare, M. G., Dabney, W., & Munos, R. (2017, August). A distributional perspective on reinforcement learning.

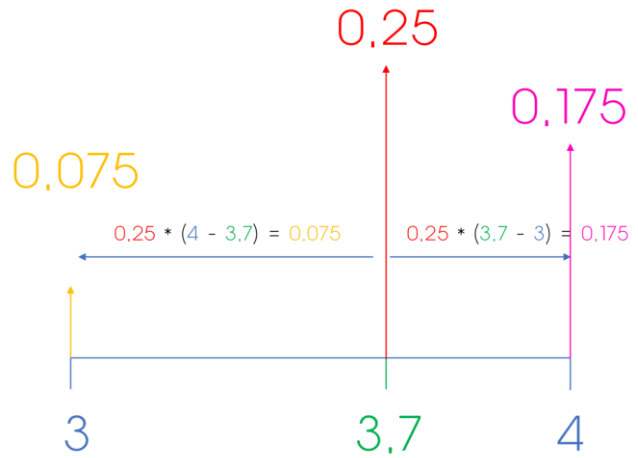
5. Distributional RL

Projection

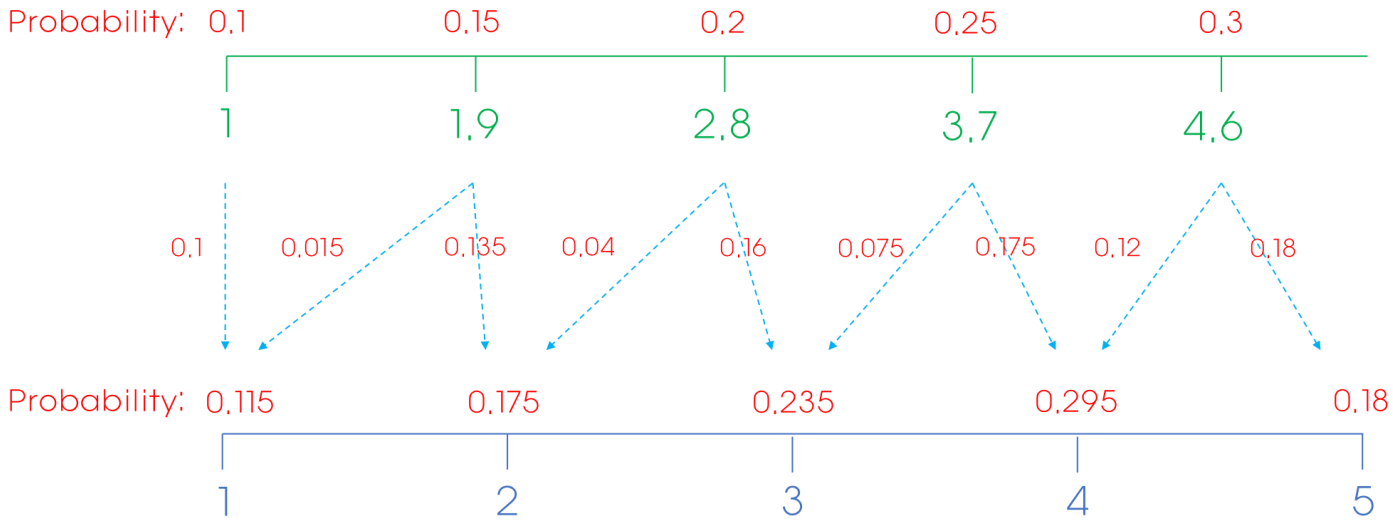


$$\hat{\mathcal{T}} z_i \leftarrow [r_t + \gamma z_i]_{V_{min}}^{V_{max}}$$





- Support: [1, 2, 3, 4, 5]
- Probability: [0.1, 0.15, 0.2, 0.25, 0.3]
- Reward: 0.1
- Discount factor: 0.9



https://reinforcement-learning-kr.github.io

Bellemare, M. G., Dabney, W., & Munos, R. (2017, August). A distributional perspective on reinforcement learning.

한계 : 후속 논문으로…

Distribution을 이용하여 value를 잘 예측하며 좋은 성능을 보이지만…..

It was shown that $\bar{d}_p$ is a metric over value distributions. Furthermore, the distributional Bellman operator $\mathcal{T}^\pi$ is a contraction in $\bar{d}_p$, a result that we now recall.

Lemma 1 (Lemma 3, Bellemare, Dabney, and Munos 2017). *$\mathcal{T}^\pi$ is a γ-contraction: for any two $Z_1, Z_2 \in \mathcal{Z}$,*

$$\bar{d}_p(\mathcal{T}^\pi Z_1, \mathcal{T}^\pi Z_2) \leq \gamma \bar{d}_p(Z_1, Z_2).$$

Distributional RL이 수렴하기 위해서는 위의 gamma-contraction 조건을 만족해야 한다.

Value distribution간 거리를 측정하는 distance metric (d_p)가 Wasserstein distance인 경우 위 조건을 만족하지만

Cross entropy 의 경우 수학적으로 위의 조건을 만족한다는 보장이 없다.

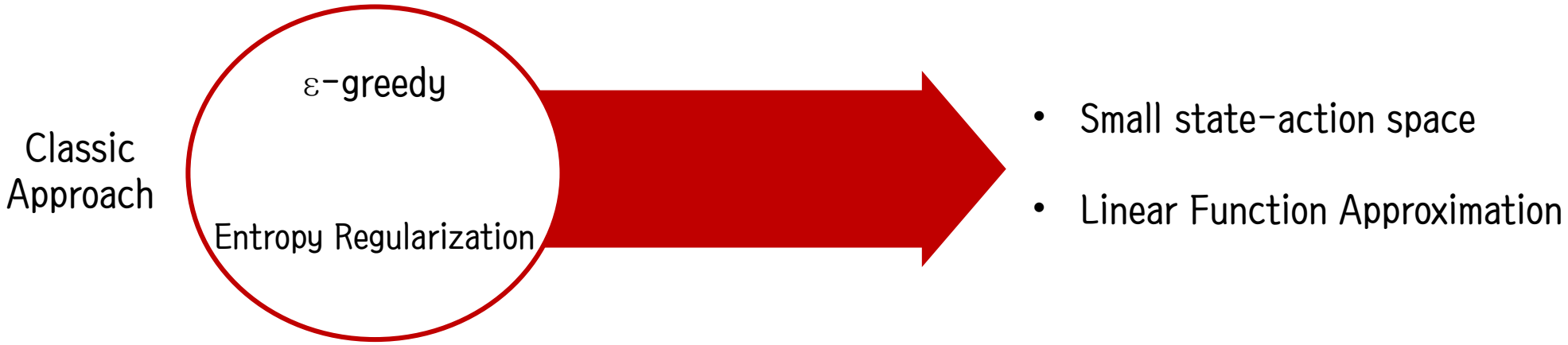
하지만 C51 논문은 Wasserstein distance를 감소시킬 방법을 찾지 못한 관계로 Cross entropy를 loss로 설정하고

이를 줄이는 방향으로 학습을 수행하는 알고리즘이기 때문에 수학적으로 distributional RL의 수렴성을 증명하지는 못하는 논문입니다

Bellemare, M. G., Dabney, W., & Munos, R. (2017, August). A distributional perspective on reinforcement learning.

6. Noisy Network for Exploration

기존 Exploration Methods의 문제점



Large scaled 환경과 Neural Network같은 복잡한 function approximation적용이 쉽지않다.

Fortunato, Meire, et al. "Noisy networks for exploration." *arXiv preprint arXiv:1706.10295* (2017).

6. Noisy Network for Exploration

Main Idea : Weight에 Noise를 섞어서 Exploration 효율을 높이자

$$y \overset{\text{def}}{=} (\mu^w + \sigma^w \odot \varepsilon^w)x + \mu^b + \sigma^b \odot \varepsilon^b$$

Weight와 bias에 noise를 섞는다.

outputs y. We represent the noisy parameters θ as $\theta \overset{\text{def}}{=} \mu + \Sigma \odot \varepsilon$, where $\zeta \overset{\text{def}}{=} (\mu, \Sigma)$ is a set of vectors of learnable parameters, ε is a vector of zero-mean noise with fixed statistics and $\odot$ represents element-wise multiplication. The usual loss of the neural network is wrapped by expectation over the

(a) Independent Gaussian noise : Unit Gaussian Distribution

(b) Factorized Gaussian noise : $\varepsilon^w_{i,j} = f(\varepsilon_i) f(\varepsilon_j), f(x) = \text{sgn}(x)\sqrt{|x|}.$

$\varepsilon^b_j = f(\varepsilon_j),$

https://medium.com/@shmuma/summary-noisy-networks-for-exploration-c8ba6e2759c7

Fortunato, Meire, et al. "Noisy networks for exploration." *arXiv preprint arXiv:1706.10295* (2017).

7. Summary

1. DQN : Network를 둘로 나눈다. + Experience Replay
2. Double DQN : action selection부분과 target updat부분으로 network를 나눈다.
3. Prioritized Experience Replay : TD error가 큰 Experience를 많이 뽑자.
4. Dueling Network Architectures : V, A를 나눠 학습하여 Q를 구하자.
5. Distributional RL : Q대신 Q의 분포 Z를 활용하겠다.
6. Noisy Network for Exploration : network의 parameter(weight, bias)에 noise를 섞자.

Discussion

- 최선이면 argmax action인거 아닌가?

  → 최선이라는 건 확률이 가장 높은 action을 의미하는 것 같다.

- 최선의 action을 했다면, Q = V가 되고 Advantage A = 0이 된다.

- Policy distribution이 uniform하다면?

  → 뭘 골라도 최선이기에 모든 action에 대해 Advantage A = 0이 아닐까?,

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & De Freitas, N. (2015). Dueling network architectures for deep reinforcement learning.