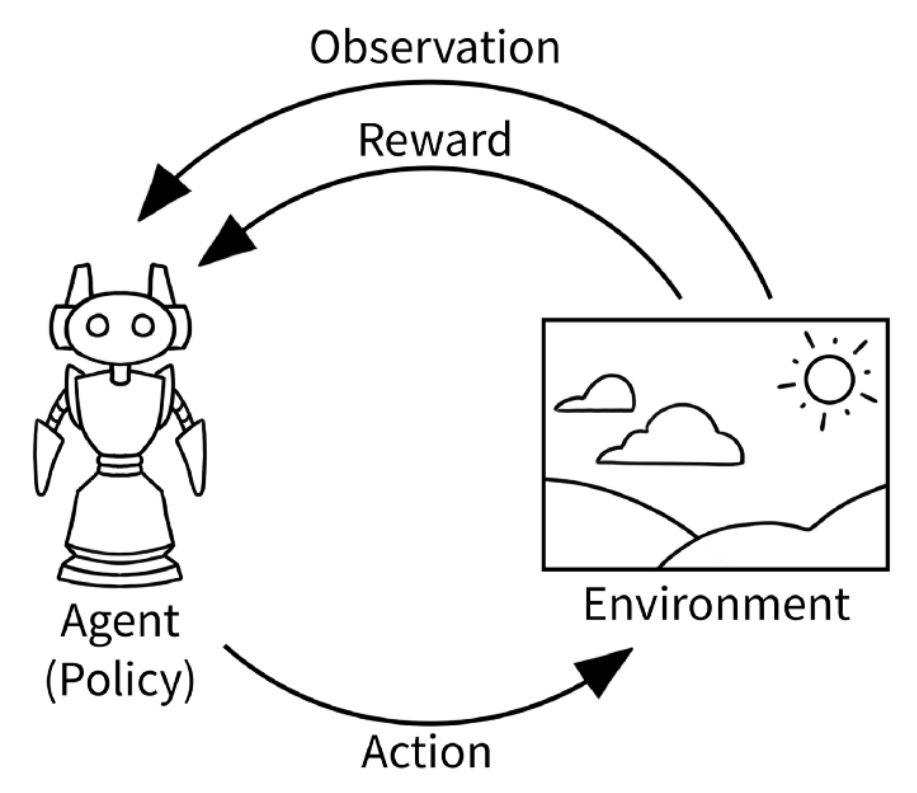
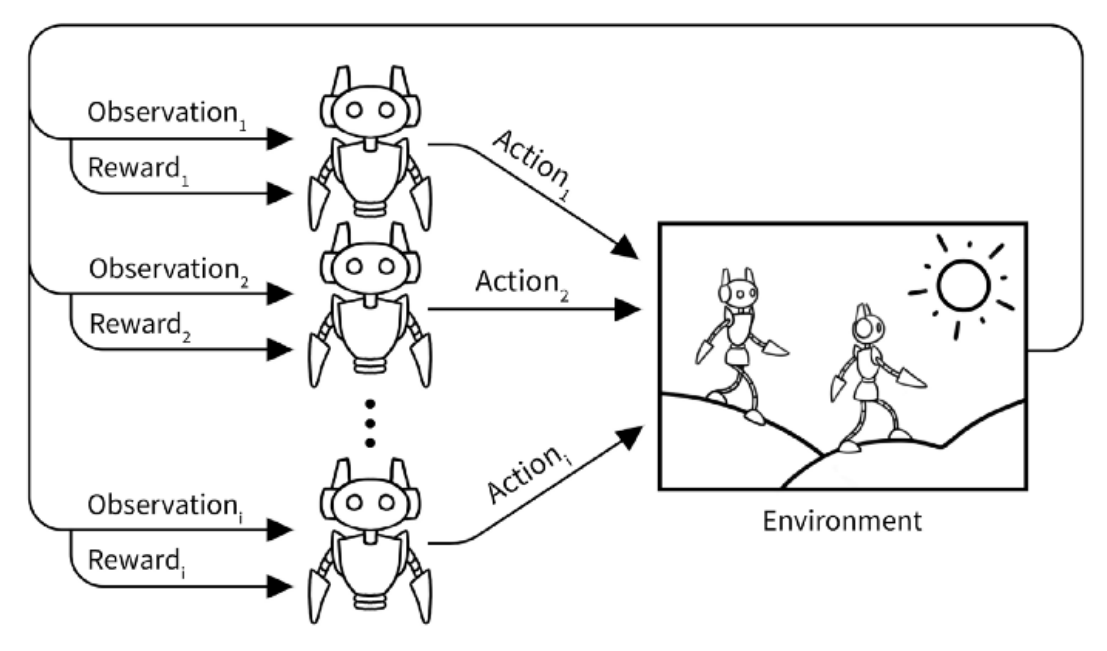
QMIX

김동영

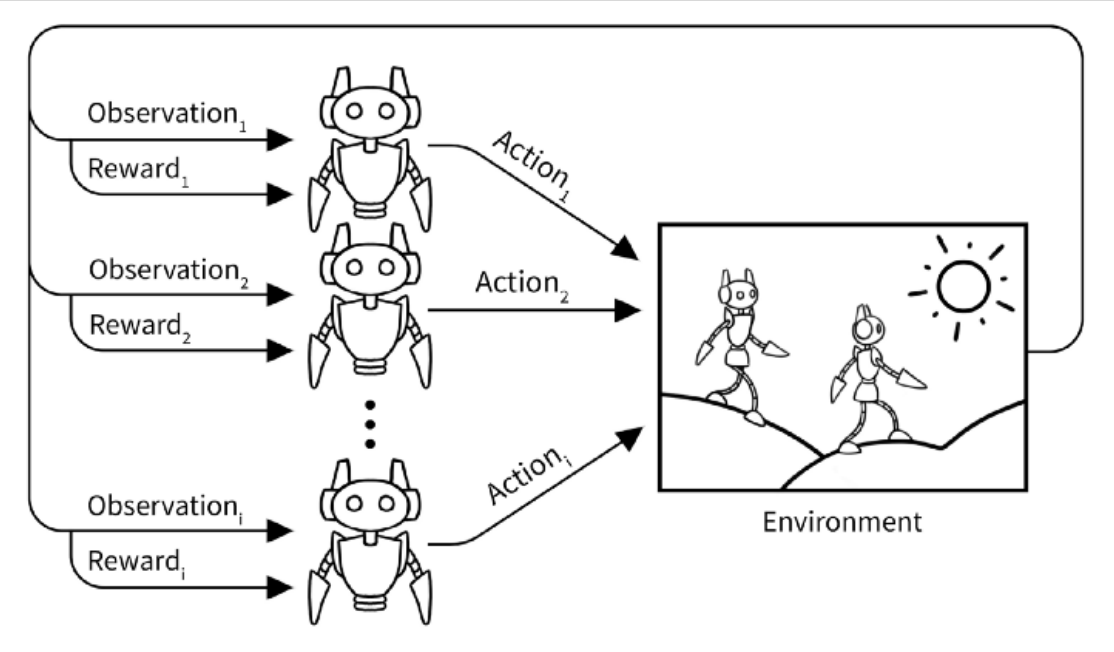
MARL의 이해

멀티에이전트 환경은 기존의 싱글에이전트 환경과 달리 여러 개의 에이전트가 환경과 상호작용한다.



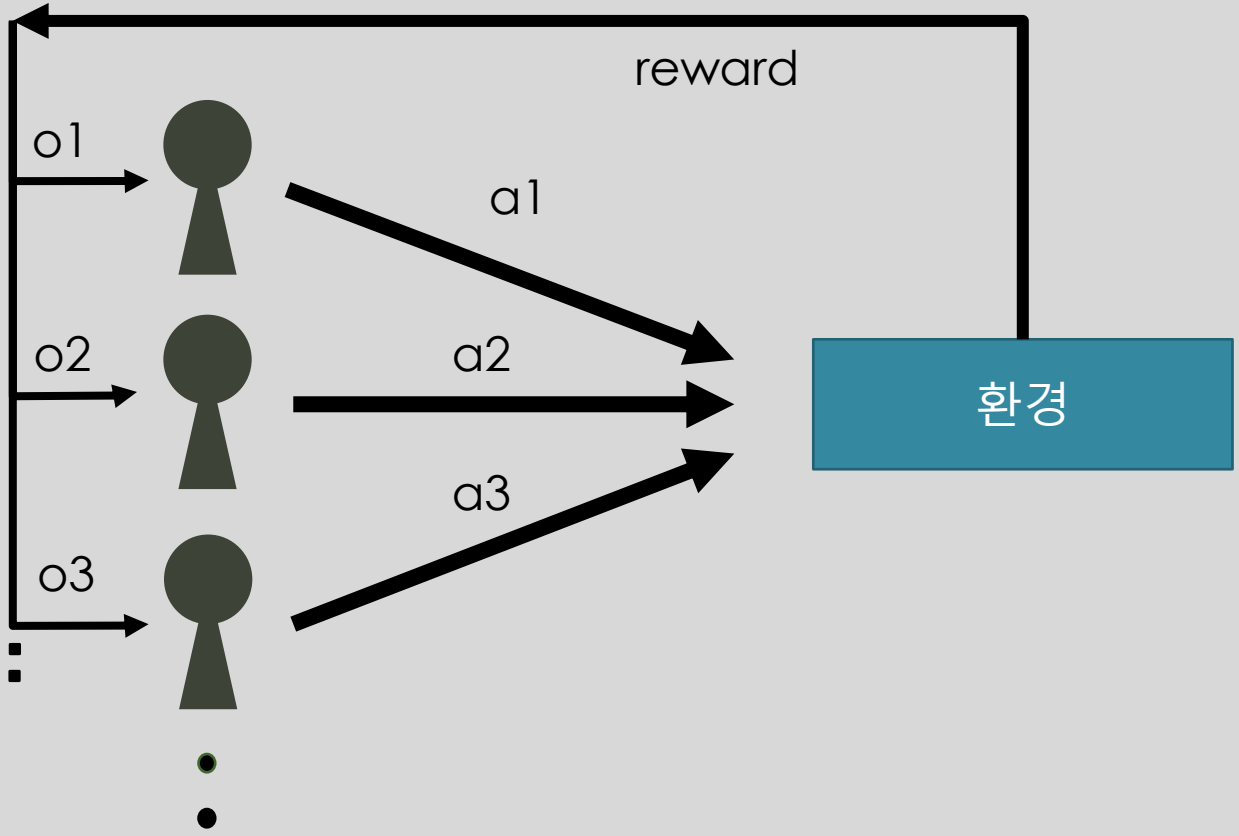


MARL의 이해



단순하게 각 에이전트는 환경에만 영향을 주는 것이 아니라, 각 에이전트의 행동들이 다른 에이전트에도 영향을 미치기에 학습이 어렵다.

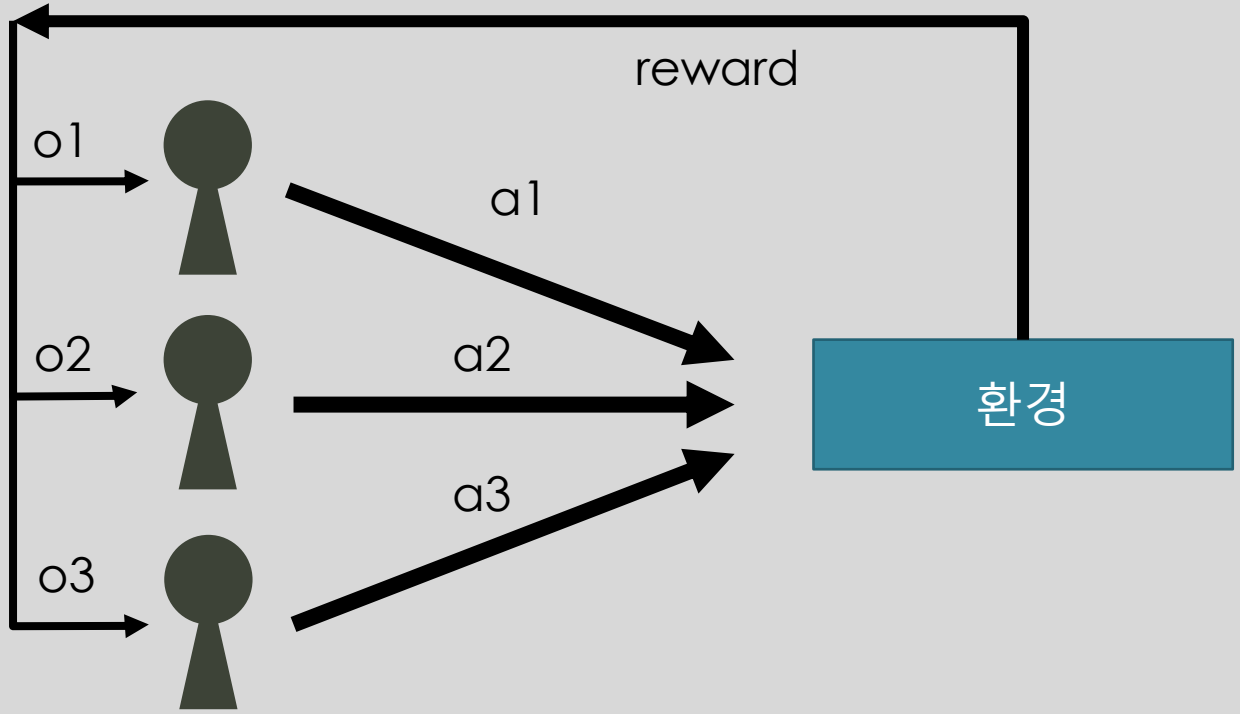
왜 MA 인가



1. 부분 관찰 환경과 이에 더해지는 물리적인 통신 제약으로 인해 분산 실행이 강요된다.

2. 단일 에이전트 방식으로 여러 에이전트를 한번에 움직이게 하는 경우, 에이전트의 수가 늘어나면서 기하급수적으로 늘어나는 joint action space로 인하 학습 성능이 떨어지게 된다.
=> action space는 에이전트 개수에 따라 곱연산으로 증가

QMIX에서 해결하고자 하는 MA 환경



1. 각 에이전트는 자신이 관찰하는 부분관찰 정보만 가지고 행동한다.

2. 환경은 각 에이전트별로 보상을 주는 것이 아니라 상황에 따라 전체 보상을 준다.

CTDE-Centralized Training and Decentralized Execution

그렇다면 그냥 각 에이전트별로 독립적으로 학습 시키면 되는거 아닌가?

아니다 이런 방식은 크게 2가지 문제에 직면한다.
1. 협동 같은 상호작용이 필요한 행동을 학습하지 못한다.
   => 환경이 조금만 복잡 해져도 매우 성능이 떨어짐

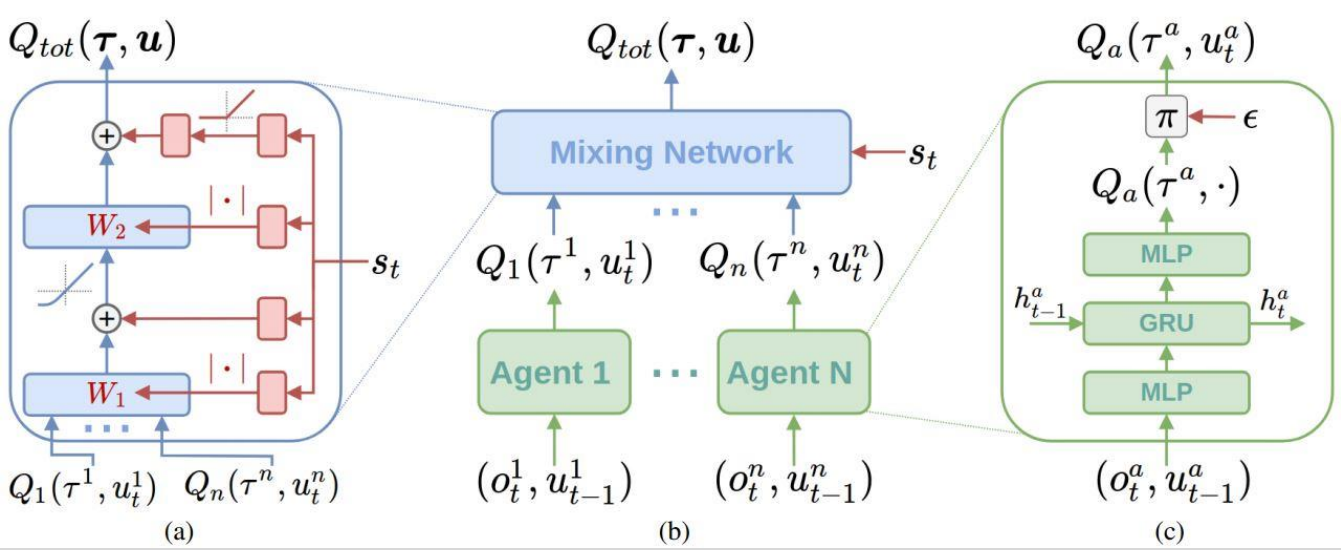
2. 에이전트의 행동에 의해서보다는 다른 에이전트들의 행동에 의해 종합적으로 보상이 주어지기에 학습에 필요한 보상 분배의 문제가 생긴다.
   => 학습 성능에 큰 문제를 준다.

그냥 중앙적으로 학습시키면 싱글 에이전트와 다를게 없다.

CTDE-Centralized Training and Decentralized Execution

학습은 실험실 환경을 통해 각 에이전트의 정보, 그리고 관찰 정보 등을 모아, 보상을 분배하고 학습을 조율하는 중앙화 학습이 이루어지고

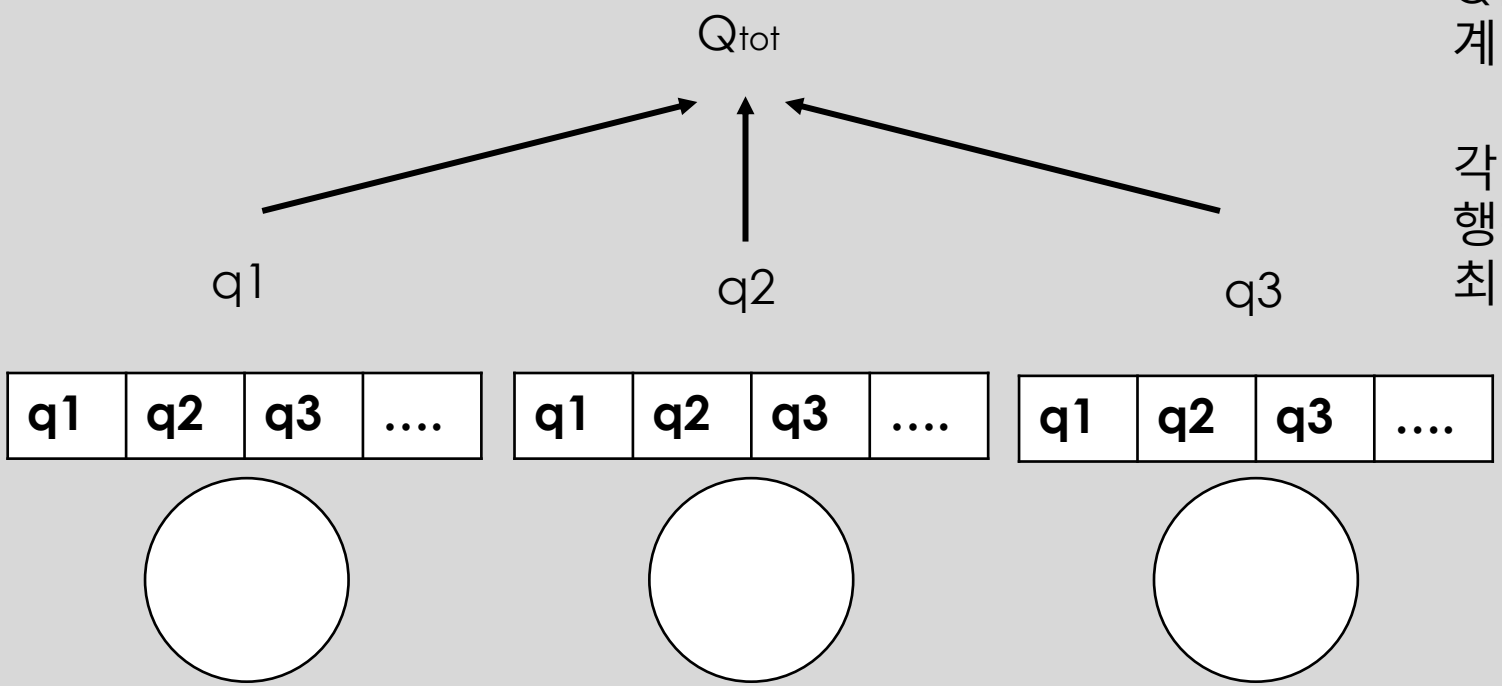
실행은 오직 에이전트에서만 일어나 분산 실행으로 진행한다.



QMIX

- QMIX는 value-base 방식의 정책을 통해 멀티에이전트 환경을 학습한다.
- 각 에이전트는 자신의 부분 관찰 정보를 바탕으로 자신의 Q 값을 계산하는 유틸함수를 가지고 이를 통해 DQN 형태로 행동한다.
- 중앙의 QMIX는 각 에이전트의 부분관찰 정보와, Q값을 입력받고 공동 Q를 계산한다.

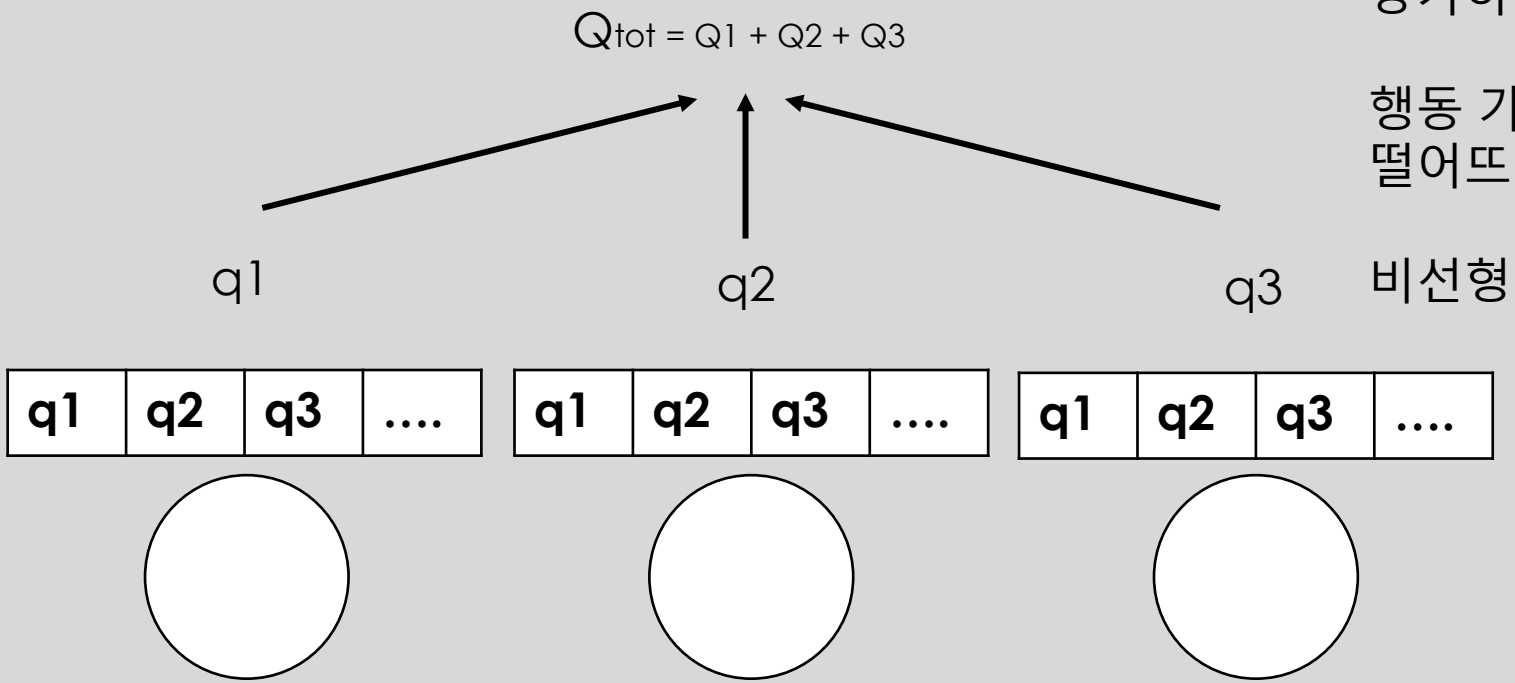
Q 방식의 문제점



여러 에이전트의 state 그리고 Q값을 받아서 공동 Q값을 계산하는것도 어렵지만

각 에이전트는 Q가 최대가 되도록 행동할텐데, 이것이 공동 Q가 최대가 된다는 보장이 없다.

이전 해결법 - VDN



공동 Q를 단순하게 각 유틸 함수의 합으로 계산, 각 Q가 증가하면 공동 Q가 증가하는 것은 보장이 되지만

행동 가치 함수의 복잡성을 매우 떨어뜨린다.

비선형 함수를 통한 근사는 안될까?

QMIX의 해결법

$$\underset{\mathbf{u}}{\operatorname{argmax}}\, Q_{tot}(\boldsymbol{\tau}, \mathbf{u}) = \begin{pmatrix} \operatorname{argmax}_{u^1} Q_1(\tau^1, u^1) \\ \vdots \\ \operatorname{argmax}_{u^n} Q_n(\tau^n, u^n) \end{pmatrix}.$$

$$\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \ \forall a \in A.$$

각 에이전트의 Q 값에 대해 공동 Q가 단조증가 관계를 가지도록만 하면 해결이 된다.

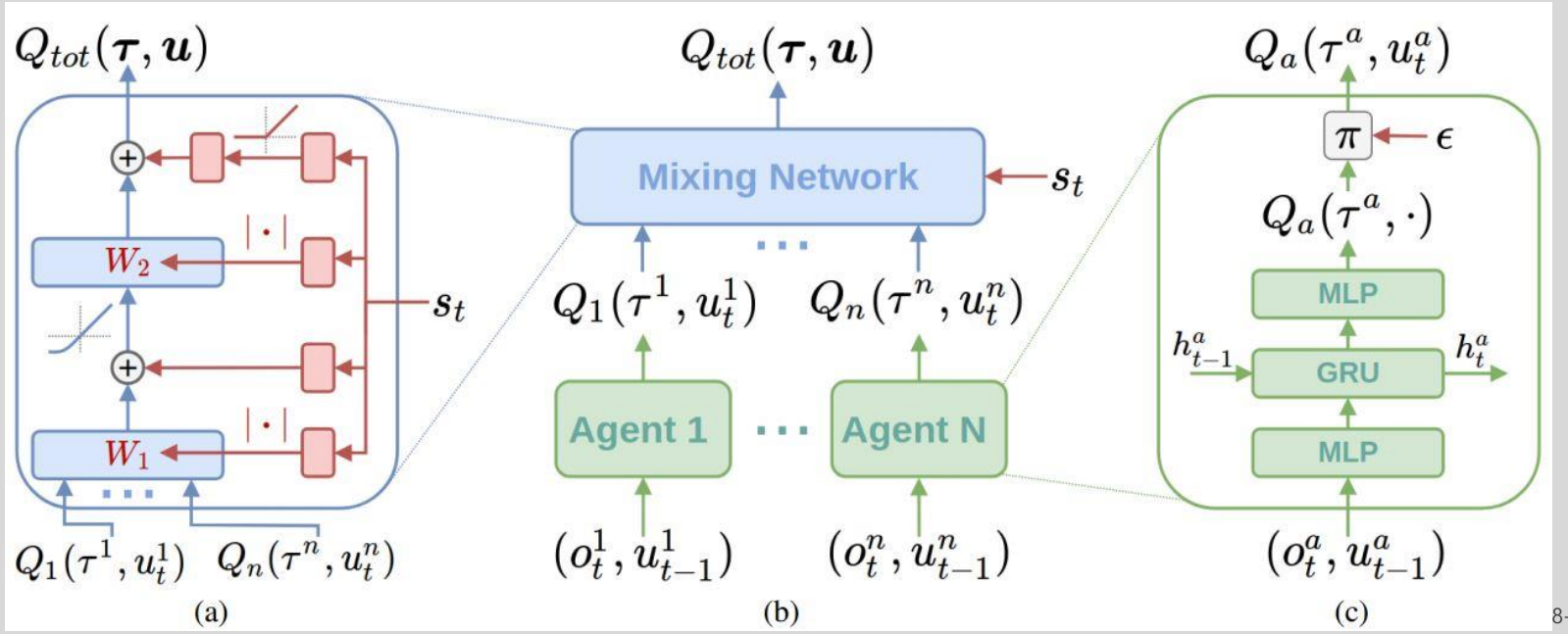
어떻게?

QMIX 신경망의 w를 모두 양수로 하자
(b는 상관없음)

QMIX

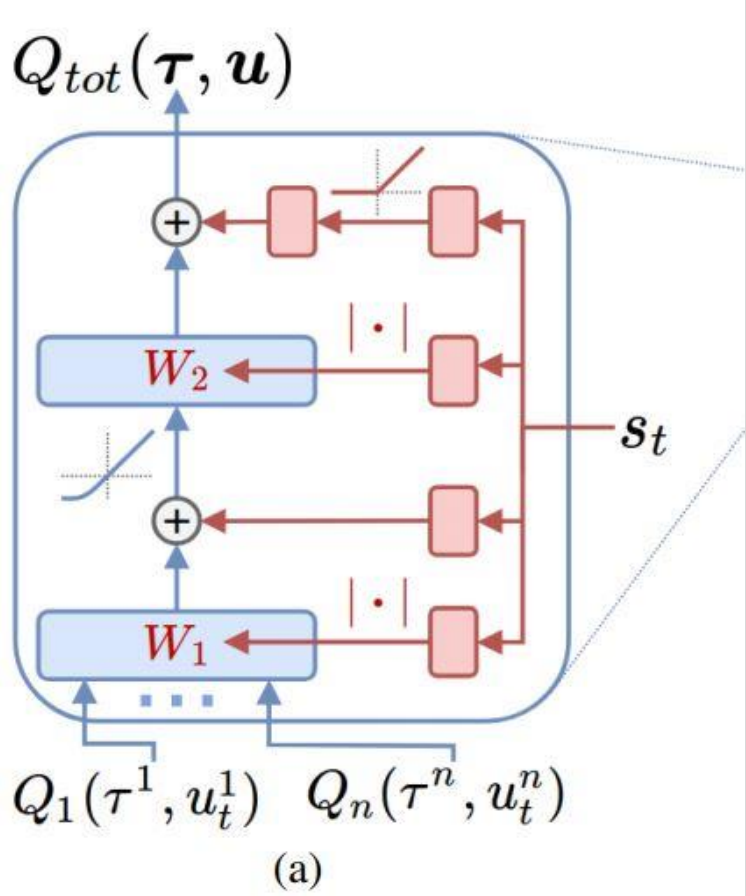
그런데, state는 굳이 그럴 필요가 없다. 단순하게 네트워크의 입력을 통해 state를 넣어주는 것은 오히려 단조 증가로 제한되어 state와 Q의 관계의 복잡성이 감소한다.

State를 입력으로 받아 mixing network의 파라미터를 출력하는 가중치 네트워크를 추가로 생성
이를 통해 가중치를 학습시키는 형태로 구성한다.



8-09

QMIX

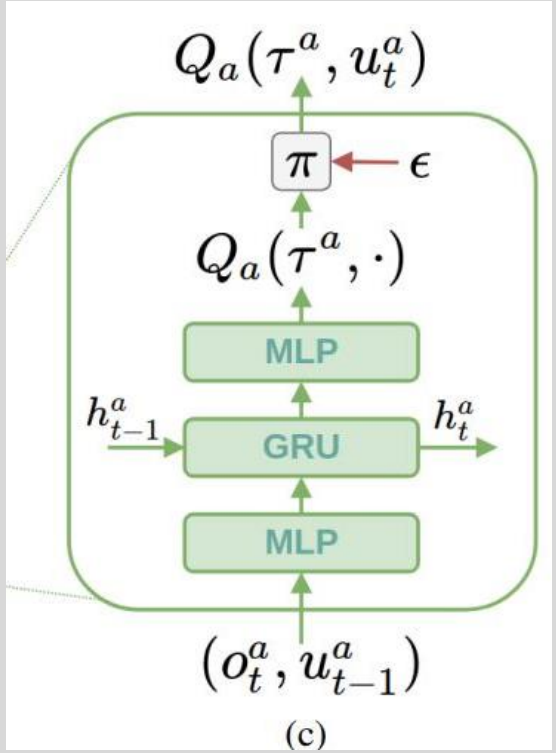


W는 가중치 신경망을 통가하고 나서 절대 함수를 통해 강제로 양수로 변환한다.

B는 첫번째 b는 단순하게 1차 레이어를 통해서 계산했지만, 최종 출력 전의 bias는 다층으로 복잡성을 높임

Hidden layer의 활성화 함수는 elu 함수를 사용했는데, 안그래도 단조 증가 함수인데 relu 를 쓰면 함수의 복잡성이 너무 떨어지기 때문이 아닐까로 본인은 판단.

QMIX



(c)

에이전트는 다음과 같이 GRU를 포함한 형태로 구성,

e-greedy 형태로 행동을 결정

DRQN과 같은 방식으로 학습 데이터를 모으고 학습시킨다.

QMIX

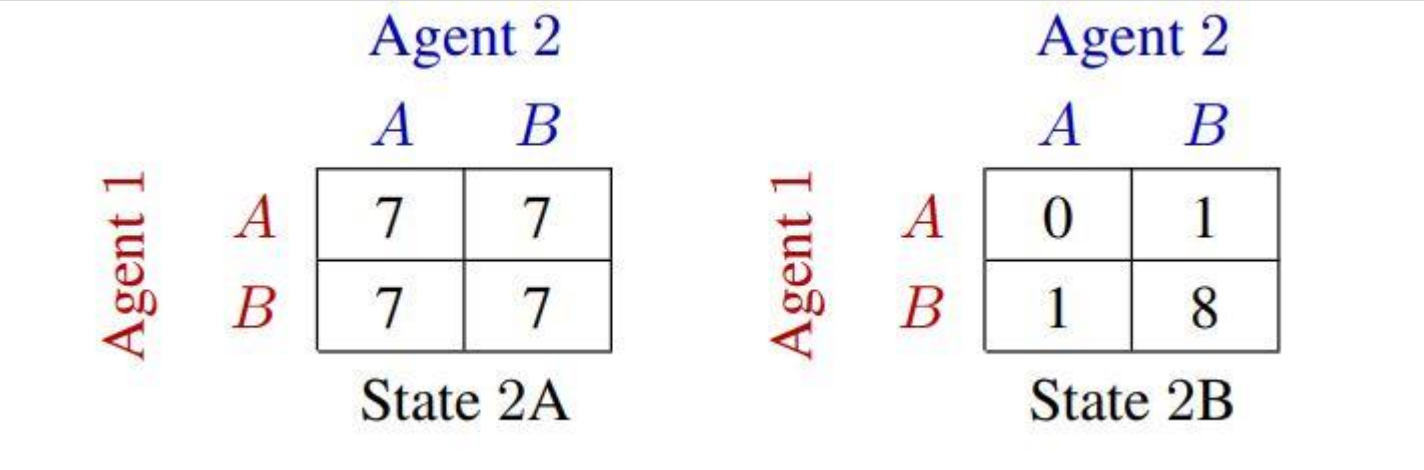
$$\mathcal{L}(\theta) = \sum_{i=1}^{b} \left[\left(y_i^{tot} - Q_{tot}(\boldsymbol{\tau}, \mathbf{u}, s; \theta) \right)^2 \right],$$

Loss 함수는 다음과 같다.

Gradient decent 방식으로 각 에이전트까지 학습이 전파된다.

$$y^{tot} = r + \gamma \max_{\mathbf{u}'} Q_{tot}(\boldsymbol{\tau}', \mathbf{u}', s'; \theta^-)$$

성능 – 2 step game



각 에이전트가 독립적으로 매트릭스를 선택하고, 같은 에이전트를 선택하면 보상을 획득

성능 – 2 step game

(a)

	State 1		State 2A		State 2B	
	A	*B*	*A*	*B*	*A*	*B*
A	6.94	6.94	6.99	7.02	-1.87	2.31
B	6.35	6.36	6.99	7.02	2.33	6.51

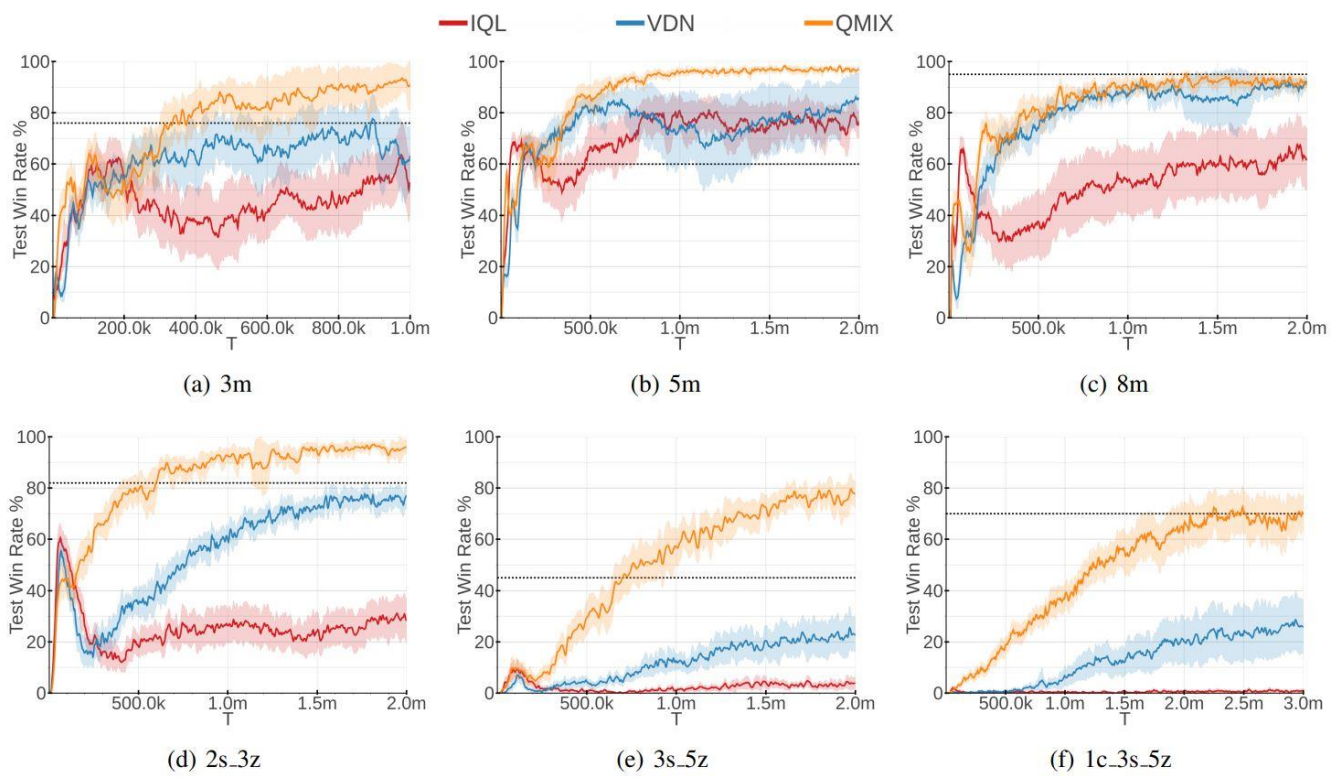
(b)

	A	*B*	*A*	*B*	*A*	*B*
A	6.93	6.93	7.00	7.00	0.00	1.00
B	7.92	7.92	7.00	7.00	1.00	8.00

Table 2. Q_{tot} on the two-step game for (a) VDN and (b) QMIX.

VDN 같은 방식은 Q 근사기의 성능이 크게 떨어지는 반면 인공신경망을 통한 Q 근사 답게 QMIX는 거의 완벽한 근사를 하는데 성공하였다.

성능 - SMAC



Starcraft multi agent challenge 환경에서도 QMIX는 다른 방식대비 확실한 성능 차이를 보임

특히 복잡한 문제로 갈수록 성능이 큰 차이를 보임

무엇이 중한가?

비선형성을 제거한, hidden layer 제거가 이루어진 QMIX –Lin
가중치 네트워크를 제거한 QMIX-NS
단조 증가를 제거한 VDN-S => 단조증가를 없애면 VDN에 가깝다 해서 VDN-S

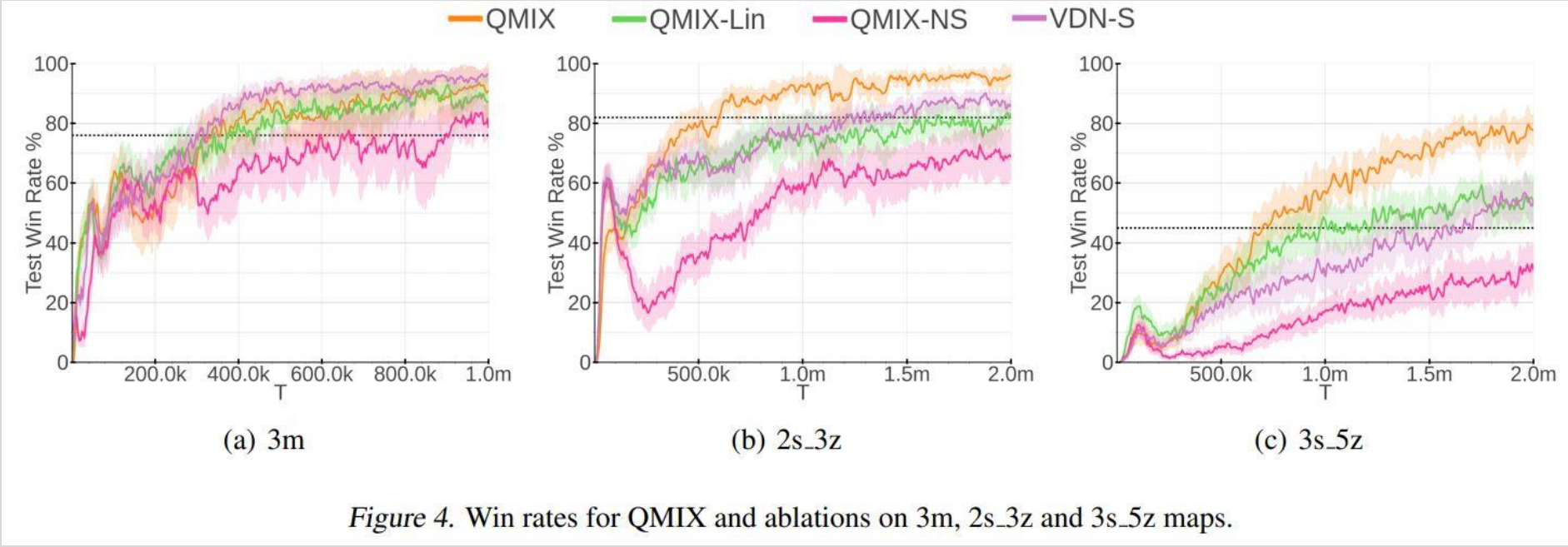


Figure 4. Win rates for QMIX and ablations on 3m, 2s_3z and 3s_5z maps.

무엇이 중한가?

간단한 환경에서는 단조 증가가 무조건 적인 성능증가를 보이는 것은
아님 그래도 복잡한 환경일수록 그 중요성은 매우 커짐
비선형성도 마찬가지
State를 통한 가중치 함수는 매우 중요한 요소이다.

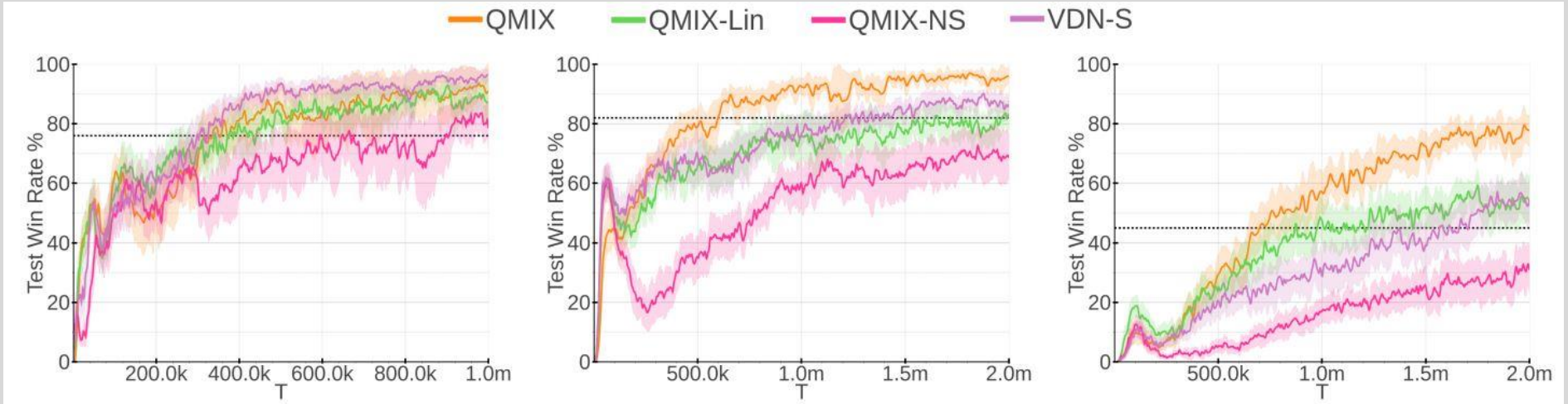


Figure 4. Win rates for QMIX and ablations on 3m, 2s_3z and 3s_5z maps.

실제 구현

- SMAC 환경에서 QMIX를 돌려보았다.

감사합니다.

https://github.com/kingdy2002/SMAC_MARL

https://kingdy2002.github.io/