남이 써놓은 자료를 계속 보자니, 왜 이소리가 여기서 나오는건지 이해가 잘 안 되는 부분이 있는 것 같다. 영어 논문을 복사해서 처음부터 끝까지 해석해보자.

제목. **Policy Gradient Methods** for Reinforcement Learning with **Function Approximation**

function approximation은 실제 함수를 여러가지 샘플을 뽑아서 근사값으로 가정한다는 말일 것이다. 그렇다면 어떤 함수를 approximate한다는 것일까? state-action value function? state-value function?

Abstract

Function approximation is essential to reinforcement learning, but the standard approach of approximating a value function and determining a policy from it has so far proven theoretically intractable.

Fuctional approximation은 강화학습에 있어서 필수적이지만, 일반적인 접근법으로 value-function을 근사하고 이를 바탕으로 policy를 결정하는 것은 이론적으로 난해한 것으로 지금까지 증명되었다.

In this paper we explore an alternative approach in which the policy is explicitly represented by its own function approximator, independent of the value function, and is updated according to the gradient of expected reward with respect to the policy parameters.

이 논문에서 우리는 새로운 어프로치를 발견했다 – policy가 명시적으로 그 자신의 function approximator로 표현되어 value function과 무관하고, 그리고 정책 파라미터에 관한 기대 reward의 그래디언트에 의해 업데이트 되는.

Williams's REINFORCE method and actor-critic methods are examples of this approach. Our main new result is to show that the gradient can be written in a form suitable for estimation from experience aided by an approximate action-value or advantage function.

윌리엄의 REINFORCE 방식과 actor-critic 방식이 이 접근법의 예가 될 수 있다. 우리의 주된 새로운 결론은 그래디언트가 approximate action-value 또는 advantage function에 의해 만들어진 experienc로부터 추측되는데 알맞은 형태로 쓰여질 수 있음을 보여주기 위함이다.

윌리엄의 Reinforce 방식?

앞서 유도했었던 이 식, monte carlo gradient estimation을 나타내는 이 식을 이용하는 방식이다. f(x)의 기대값을 하나의 유려한 함수로 나타내기는 엄청 힘든 법이다. 위를 이용하면, f(x)를 몰라도 확률분포 그래디언트와 f(x)값을 곱한 값을 다수 샘플링해 평균값을 구하면 전체 그래디언트를 얻을 수 있다는 이야기 인 것 같다. 그럼 pg는 Reinforcement 방식과 뭐가 다르다는 걸까?

Using this result, we prove for the first time that a version of policy iteration with arbitrary differentiable function approximation is convergent to a locally optimal policy.

이 결과를 이용해, 우리는 임의의 미분가능한 function approximation에 의한 policy iteration의 한 버전이 로컬 옵티멀로 수렴함을 증명했다.

본문

Large applications of reinforcement learning (RL) require the use of generalizing function approximators such neural networks, decision-trees, or instance-based methods. The dominant approach for the last decade has been the value-function approach, in which all function approximation effort goes into estimating a value function, with the action-selection policy represented implicitly as the "greedy" policy with respect to the estimated values (e.g., as the policy that selects in each state the action with highest estimated value).

대규모 RL 응용은 nn이나, decision tree나 인스턴스 기반 방법 같은 일반화된?하는? function approximators의 사용이 필요하다. 지난 십년간 가장 주된 접근법은 모든 fuction approximation 을 위한 노력이 value function을 추정하는데 투자되는 value-fuction 접근법이고, 추정된 value에 대한 greedy policy로 action-selection policy가 암묵적으로 표현되었다.(예: 각 상태에서 가장 높은 추정가치를 가진 action을 선택하는 policy)

The value-function approach has worked well in many applications, but has several limitations.

이것은 많은 응용에서 잘 동작해왔지만 몇가지 한계가 있다.

First, it is oriented toward finding deterministic policies, whereas the optimal policy is often stochastic, selecting different actions with specific probabilities (e.g., see Singh, Jaakkola, and Jordan, 1994).

첫째로, 이것은 deterministic policy를 찾는 것을 지향하는 반면에 몇몇 optimal policy는 stochastic하여 특정 확률로 action을 선택한다. (예로 Singh, Jaakkola, and Jordan논문을 보아라)

Second, an arbitrarily small change in the estimated value of an action can cause it to be, or not be, selected. Such discontinuous changes have been identified as a key obstacle to establishing convergence assurances for algorithms following the value-function approach (Bertsekas and Tsitsiklis, 1996).

둘째로, 추정된 action value값에 약간의 변화만 생겨도 선택이 되느냐 마느냐를 결정하게 된다. 이런 비연속적인 변화가 value-function 접근법을 따르는 알고리즘의 수렴을 보장하는 데에 핵심 장애물이라고 알려져있다. (Bertsekas and Tsitsiklis, 1996).

For example, Q-Iearning, Sarsa, and dynamic programming methods have all been shown unable to converge to any policy for simple MDPs and simple function approximators (Gordon, 1995, 1996; Baird, 1995; Tsitsiklis and van Roy, 1996; Bertsekas and Tsitsiklis, 1996).

Qlearning 외 등등의 방법들이 간단한 MDP나 function apporximator에 수렴하지 못하는 것으로 보여졌다. (Gordon, 1995, 1996; Baird, 1995; Tsitsiklis and van Roy, 1996; Bertsekas and Tsitsiklis, 1996). \* MDP : Markov Decision Procss

This can occur even if the best approximation is found at each step before changing the policy, and whether the notion of "best" is in the mean-squared-error sense or the slightly different senses of residual-gradient, temporal-difference, and dynamic-programming methods.

이것은 policy를 바꾸기 전 매 스텝마다 최선의 approximation이 발견되어도, 그리고 이 최선이 mse 관점인지 residual gradient, temporal-difference, dynamic-programming method의 살짝 다른 관점인지 여부와 관계없이 발생할 수 있다.

Dynamic programming : 주어진 문제를 여러 개의 하위 문제로 나누어 푼 다음 결합해 최종 문제를 해결하는 방식. 벨먼-포드, 데익스트라, 플로이드-워셜 알고리즘. 등등…

In this paper we explore an alternative approach to function approximation in RL.

새방법 발견했다.

Rather than approximating a value function and using that to compute a deterministic policy, we approximate a stochastic policy directly using an independent function approximator with its own parameters.

deterministic policy를 계산하기 위해 value function을 근사하기보다, 우리는 스스로의 패러미터를 가진 독립적인 function approximator를 이용해 stochastic policy를 근사한다.

For example, the policy might be represented by a neural network whose input is a representation of the state, whose output is action selection probabilities, and whose weights are the policy parameters.

예를들어 policy는 입력으로 state 표현, 출력으로 action 선택확률, 그리고 weight으로 policy parameter을 가진 nn으로 표현될 수도 있다.

Let \theta denote the vector of policy parameters and \rho the performance of the corresponding policy (e.g., the average reward per step). Then, in the policy gradient approach, the policy parameters are updated approximately proportional to the gradient:

그래디언트에 비례해서 policy는 업데이트된다.

where is a positive-definite step size. If the above can be achieved, then can

usually be assured to converge to a locally optimal policy in the performance measure

. Unlike the value-function approach, here small changes in can cause only small

changes in the policy and in the state-visitation distribution.

…는 보통 로컬 옵티멀 policy로 수렴이 보장된다. value function 접근법과 다르게 theta 변화는 policy와 state 방문 분포에 작은 변화만을 줄 수 있다.

In this paper we prove that an unbiased estimate of the gradient (1) can be obtained from experience using an approximate value function satisfying certain properties.

이 논문에서 우리는 unbiased estimate of the gradient가 특정 요건을 만족하는 추정된 value function을 이용한 경험을 통해 도출될 수 있음을 증명한다.

Williams's (1988, 1992) REINFORCE algorithm also finds an unbiased estimate of the gradient, but without the assistance of a learned value function. REINFORCE learns much more slowly than RL methods using value functions and has received relatively little attention.

윌리엄의 REINFORCE 알고리즘 역시 unbiased estimate of the gradient를 발견했지만, 이미 학습된 value function의 도움 없이 REINFORCE는 value function을 이용한 RL방식보다 훨씬 늦게 학습되었고 상대적으로 적은 관심을 받아왔다.

Learning a value function and using it to reduce the variance of the gradient estimate appears to be essential for rapid learning.

value function을 학습하고 이를 gradient estimate의 variance를 줄이는데 사용하는 것은 빠른 학습에 필수적인 것으로 보인다.

Jaakkola, Singh and Jordan (1995) proved a result very similar to ours for the special case of function approximation corresponding to tabular POMDPs.

tabular pomdps와 관계된 특별한 케이스의 function approximation에서 jaakkola 등이 우리와 매우 비슷한 결과를 증명했었다.

Our result strengthens theirs and generalizes it to arbitrary differentiable function approximators.

우리 결과는 그들것을 강화하였고 임의의 미분가능한 function approximator에 대해 일반화했다.

Konda and Tsitsiklis (in prep.) independently developed a very simialr result to ours. See also Baxter and Bartlett (in prep.) and Marbach and Tsitsiklis (1998).

누군가들이 개별적으로 우리와 비슷한 결과를 발견했다 참고하시길!

Our result also suggests a way of proving the convergence of a wide variety of algorithms based on "actor-critic" or policy-iteration architectures (e.g., Barto, Sutton, and Anderson, 1983; Sutton, 1984; Kimura and Kobayashi, 1998).

우리의 결과는 또한 actor-critic 또는 policy-iteration 구조에 기반한 넓은 다양성을 가진 알고리즘들의 수렴을 증명할 방법을 제시한다. (관련 논문)

In this paper we take the first step in this direction by proving for the first time that a version of policy iteration with general differentiable function approximation is convergent to a locally optimal policy.

이 논문에서 우리는 첫번째로 일반적인 미분 가능한 function approximation을 통한 policy iteration의 한 버전이 로컬 옵티멀 policy로 수렴함을 증명해 이 방향으로의 첫 걸음을 딛었다.

Baird and Moore (1999) obtained a weaker but superficially similar result for their VAPS family of methods.

barid 와 more가 좀 더 약하지만 피상적으로 비슷한 결과를 얻었다, 그들의 VAPS family of methods에서.

Like policy-gradient methods, VAPS includes separately parameterized policy and value functions updated by gradient methods.

Vaps도 pg와 마찬가지로 각각 parameterized된, gradient methods로 update되는 policy와 value function을 가지고 있다.

However, VAPS methods do not climb the gradient of performance (expected long-term reward), but of a measure combining performance and value function accuracy.

하지만 vaps는 성능의 gradient를 상승시키는 것이 아니라 성능과 value function 정확도를 결합한 값의 gradient를 상승시킨다.

As a result, VAPS does not converge to a locally optimal policy, except in the case that no weight is put upon value-function accuracy, in which case VAPS degenerates to REINFORCE.

vaps는 결국 로컬 옵티멀 policy로 수렴하지 않고, 예외적으로 value-fuction accuracy에 weight이 가해지지 않는 경우에만 수렴하는데 이는 vaps가 reinforce로 퇴보한 경우이다.

Similarly, Gordon's (1995) fitted value iteration is also convergent and value-based, but does not find a locally optimal policy.

gordon의 fitted value iteration 역시 수렴하고 value-based이지만 로컬 옵티멀 policy를 찾아주진 않는다.

1. PG theorem

We consider the standard reinforcement learning framework (see, e.g., Sutton and

Barto, 1998), in which a learning agent interacts with a Markov decision process

(MDP).

learning agent가 MDP와 상호작용하는 스탠다드한 강화학습 프레임워크를 상정하자.

The environment's dynamics are characterized by state transition probabilities, , and expected rewards , .

환경에 대한 설정

…그리고 이하 policy

where

theta는 real number 벡터이고 길이는 S 총 개수보다 극심히 적다.

We assume that \pi is differentiable with respect to its parameter, i.e., that exists. we also usually write just \pi(s,a) for \pi(s,a,\theta)

미분 가능해야 한다….

결론부터 보면 average form이건 start-state form이건 이렇게 gradient를 구할 수 있다.

policy가 변화하면 도 변한다. 하지만 gradient 계산에 를 구할 필요가 없음이 증명된다. 이점이 핵심.

would be unbiased estimate of . 이게 무슨 의미일까? unbiased estimation은 통계값을 정확히 측정할 수 있는 방법이라는 뜻이라고 한다. 표본의 통계값을 통해 모집단의 통계값을 보정 없이 정확히 측정할 수 있을 때.

남은 것은 Q에 관한 이야기인데,..

policy parameterization? 이게 뭔진 모르겠지만 아무튼 성립한다고 가정하고 풀다보면 결국

Q를 f\_w로 바꿔도 된다는 가정이 성립한다.

이걸 이리저리 지지고 볶다보면 결국

로 parameter를 local minimal로 수렴시킬 수 있다는 가정이 된다.

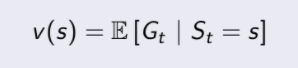
<https://www.youtube.com/watch?v=2YFBordM1fA> 팡요랩 policy gradient에 대한 강의

… 를 듣기 전에, TD 같은 생소한 개념들이 나오는 것 같아서

<https://dnddnjs.gitbooks.io/rl/content/history.html> 이 블로그 article을 사용해보려 한다.

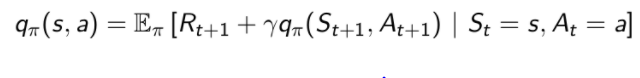
bellman equation은 두가지로 나뉜다.

expectation equation, optimality equation

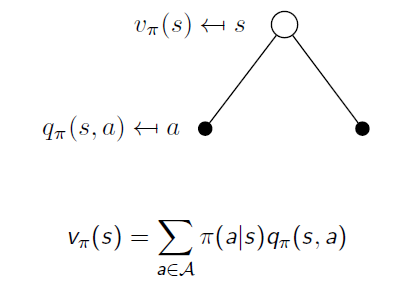


여기서 G\_t는 state에서 미래 진행을 할 동안 얻을 reward를 time discount적용해 더한 값

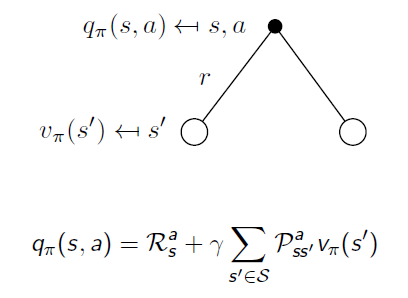




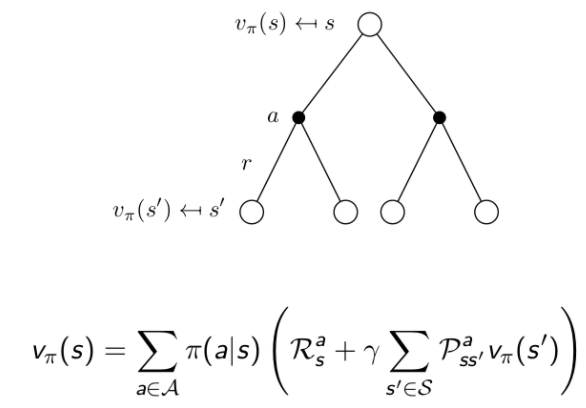
하지만 위와 같이 나타내는 것은 직관적이지 못하다.



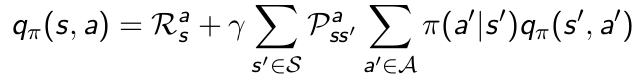
V는 Q의 기대값으로 나타내고



q는 s에서 a를 취할 때 얻을 reward를 즉각적으로 얻고, 여기에 discount된 다음 reward 평균값을 넣게 되는데 환경의 dynamics를 고려해서 평균값을 구해준다.

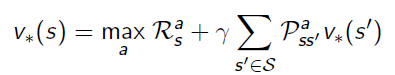


종합하면 이런 모습이다.



같은 개념을 q에 관한 식으로 나타낼 수도 있다.

optimal value function은 다릅니다. 확률적으로 움직이는 policy가 없어지며, action-value function 그 자체가 policy가 됩니다. 예전에 배웠던 Q-learning이 사용하는 그 식입니다.



expectation equation을 이용한s DP는 Policy iteration,

optimality equation을 이용한 DP는 Value iteration입니다. policy iteration의 경우 evaluation->improvement의 반복입니다. 이론상 무한step의 evaluation을 한 뒤 policy를 improve 한 후, 다시 새로운 policy에 대한 state-value function을 구하기 위한 무한 step의 evaluation을 반복하고… 하지만 value iteration은 그럴 필요가 없습니다. 무한step evaluation을 한번 하고 나면 policy도 자동으로 알아서 구해집니다. max 방향만을 찾는 policy가요.

둘 중 어느 DP가 되었건, Reward function과 state transition probability로 표현되는 environment의 모델을 알아야 하는 model based한 방법이며

Full-width Backup (“한 번 update할 때 가능한 모든 successor state의 value function을 통해 update하는 방법”)을 사용하기 때문에 backup 한번의 비용이 너무 크고

모델을 정확히 알아야 적용할 수 있다는 단점이 있습니다.

이를 해결하기 위한 방법으로 sample backup이 제시되었으며 model free하다는 특징을 가집니다.

“현재의 policy를 바탕으로 움직여보면서 sampling을 통해 value function을 update하는 것을 model-free prediction이라 하고 policy를 update까지 하게 된다면 model-free control이라고 합니다.”

여기에는 monte carlo learning과 temporal difference learning이 있습니다.

MC 부터 알아봅시다. 별거 없습니다. 반복해서 경험 시키고, reward를 평균내서 state의 Value function을 구해주는 것입니다.

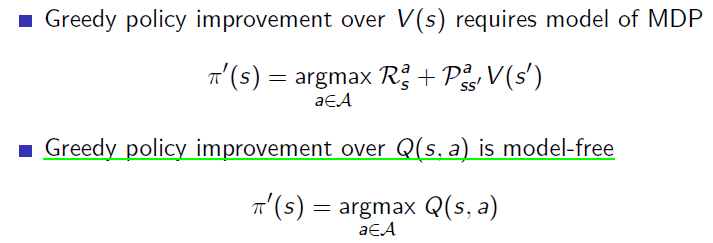


단, 특이한 점은 알파를 상수로 고정해서 초반 경험의 reward 가중치를 낮춘다는 점입니다. MC prediction입니다.

DP에서 Policy iteration 할 때 evaluation부분만 이걸로 바꿔주면 MC control이 됩니다.

문제점이 있는데, 첫째는 Value function이다.

Value function을 사용하면 policy를 improve할 때 reward 와 transition probability를 알아야 할 수 있다. 따라서 value function대신 action function을 사용한다.

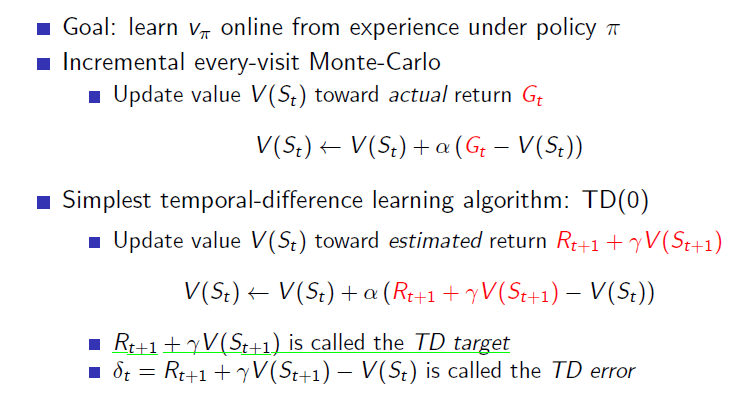


두번째는 exploration이다. 이미 간 곳만 계속 가는 현상을 방지하고 exploration을 하기 위해 **e-greedy policy improvement**를 한다.

세번째는 policy iteration이다. 수렴할 때 까지 evaluation을 반복해야 하는 문제가 있었는데, monte carlo 방식에서는 마치 value iteration처럼 evaluation과정을 줄인다.

그럼에도 여전히 MC control은 문제점이 있다. online으로 바로바로 학습할 수 없으며, episode에 끝이 있어야 한다. 끝이 있지만 그때까지 많은 시간이 걸리는 경우 – 스타크래프트 같은 게임- 학습하기 어려운 단점이 있다**. episode가 끝나지 않아도 DP처럼 time step마다 학습할 수 있지 않나? 하는 생각에서** Temporal difference 가 나왔다.

이것이 어떻게 가능한 것인가? 글에서는 대학을 처음부터 끝까지 경험하고 다시 다니는 MC와, 1학년이 2학년을 따라서 움직이는 TD의 예시를 든다. 구체적으로는 어떤 뜻일까?



G\_t는 MC에서 한 에피소드가 끝난 후 얻을 수 있었던 reward 합이었다. 여기서 TD는 이 G\_t를 R\_(t+1) + \gamma V(S\_(t+1))으로 변경해 적용한다. 가장 간단한 형태의 TD target이다.

TD는 baise가 높고 MC는 variance가 높은 경향이 있다고 한다.

TD의 value function을 action value function으로 바꿔준 것이 **Sarsa**이다. 비단 1스텝에 국한된 움직임 뿐만 아니라, 2스텝, n스텝을 기반으로 하는 TD역시 있다.

…그 다음은? MC와 TD 모두 on-policy이다. 현재 policy 위에서 control (prediction + improvement)을 하는 것을 on-policy라고 한다. 이러면 탐험의 문제가 발생한다고 한다. 움직이는 policy와 학습하는 policy를 분리하는 off-policy가 필요한 이유이다.

off policy는 …

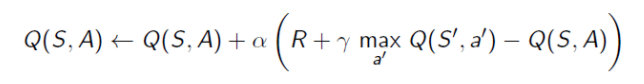
1. 다른 agent나 사람의 플레이를 보고 학습할 수 있으며,
2. 이전 policy의 경험들을 재활용할 수 있고,
3. 탐험을 계속하면서도 옵티멀한 policy를 찾을 수 있다(Q learning)
4. 하나의 policy를 따르면서 여러 개의 policy를 학습할 수 있다.

이것이 가능한 이론적 **근거는 importance sampling**이라고 한다.

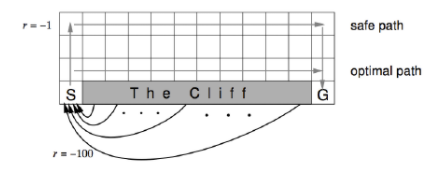
Off-policy MC는 별로 좋은 방법이 아니다. 그렇다고 Off-policy TD역시 TD에 비해 variance가 높다는 단점이 있다. off policy를 사용할 때 **importance sampling을 사용하는 것은 별로 좋은 방법이 아니라고 여겨진다**. 여기서 나오게 되는 방법이 바로…

Q learning. action-value Q(s,a)를 이용하는 off-policy 방법. behavior policy와는 다른 policy를 action 선택에 사용하면 importance sampling을 할 필요가 없다고 한다. 왜? 모르겠다. 움직이는 policy와 학습하는 policy를 따로 나눈게 off-policy방법이라며? 무슨 차이야?

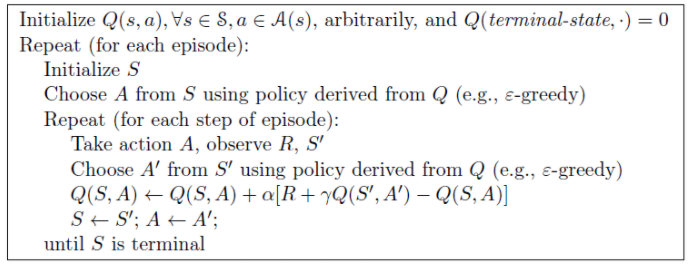
아무튼, 우리가 Q learning 할 때 선택 policy는 e-greedy를 사용하고, target policy(실제로 우리가 염두에 두는 policy?)는 greedy인 것이 위의 예시라고 한다.



완전 익숙한 식이 되었다!



Sarsa와 Q learning의 차이점을 보여줄 수 있는 예시로 Sutton은 cliff walking 과제의 예시를 들었다고 한다. 절벽에 떨어지면 -100의 reward, 그 외의 움직임마다 -1 reward를 받는 이 environment에서 Sarsa와 Q learning 모두 e-greedy를 이용하기에 가끔 절벽으로 떨어지게 된다. 하지만 Sarsa의 경우는 절벽에 붙어있는 state의 value를 낮게 보게 되는 반면, Q의 경우는 여전히 가치를 높게 치게 된다. 따라서 optimal path로 가게 되는 확률이 Q learning에서 더 높다고 한다. 둘 다 action value 기반 방법인데 왜일까?



Sarsa의 sudo code이다. 잘 보면 같은 policy derived from Q로 두 번 action을 선택해 Q를 갱신하는 모습을 볼 수 있다. 이러니 만약 e-greedy를 사용하는 policy에서 자칫 절벽으로 떨어지게 되면, 그 이전 action에 대한 value가 큰 마이너스를 먹게 된다.

하지만 Qlearning에서는 학습 시 사용하는 algorithm으로 다른 것을 사용한다. greedy algorithm, 즉 argmax 이다. e-greedy에 의해 절벽에 떨어진다고 하여도 학습에는 즉각적 영향을 주지 않는다. 왜냐? argmax에는 영향이 없기 때문이다.

Q-learning이 등장한 이후로 이것은 강화학습의 주류가 되었다. 좋은 성능을 지녔기 때문이다. 이것으로 다 끝난 것일까? 지금까지 우리가 본 방법은 Tabular method, 즉 state action value funciton을 table로 만들어 해결해온 것이었다. state, action수가 늘어나면 늘어날수록 이 방법은 사용이 불가능해진다. 해결법이 있을까?

바로 value function을 테이블로 만들지 말고 w를 이용한 함수화를 시키는 것이다.

…이후는 이미 배웠던DQN내용…

Policy gradient : 기존의 value based 한 approach에서 벗어나 policy 그 자체의 학습에 기반한 방법.

actor-critic ?

critic은 action-value function parameter w를 업데이트하고

actor는 policy function parameter theta를 업데이트 한다, critic이 제안하는 방법으로.

base line에 대해서도 나왔는데 잘 이해는 안돼서 넘어간다.