팡요랩 강의 내용:

actor-critic은 MCPG에서 실제 Reward를 Q에 대한 sample로 썼었는데 그러지 말고, Q 자체도 w로 parameterize하여 학습시키자는 발상에서 출발한다. 보다 보면 기시감이 들 수 있는데, policy iteration과 매우 유사하다고 볼 수 있다.

baseline : 99만과 100만 reward를 그대로 사용하면 차이가 별로 안 느껴진다. ->학습이 비효율적이다. 99만 5000을 각각 빼서 학습을 시켜보자. 수학적으로 맞는 이야기인가? 맞다. s에만 관련한 식인 value function을 빼 주어서 평균 값은 변화 없이, variance만 줄일 수 있다. Q-V를 **advantage함수라고 한다.**

여기서 에바참치 발생. V에 대한 파라미터까지 학습시켜야 한다? 오우 쉣.

결론적으로, Q는 안 필요하다. V만으로 advantage를 계산할 수 있다. TD error가 advantage function의 unbiased sample이다. 왜냐? Q(s,a) = r + \gammaV(s’)니까. 마법처럼 TD error만을 신경쓰면 되는 식이 된다.

구현 강의 내용

AutoDiff? loss를 -r\*log\pi로 지정해 주면 된다. 우리가 구해야 하는 것은 r\*log\pi의 미분값.

-------------

DPG로 넘어가 보자!

DPG는 Expected gradient of the action-value function의 형태이다.

Policy variance가 0에 수렴할 경우 DPG는 Stochastic Policy Gradient와 동일해진다.

적절한 exploration을 위해 model-free, off policy actor critic algorithm을 제안한다.

DPG는 SPG보다 성능이 좋은데, 특히 high dimensional action spaces를 가지는 tasks에서의 성능 향상이 크다.

* SPG의 policy gradient는 state와 action spaces 모두에 대해서, DPG의 policy gradient는 state spaces에 대해서만 평균을 취합니다.
* 결과적으로, action spaces의 dimension이 커질수록 data efficiency가 높은 DPG의 학습이 더 잘 이뤄지게 됩니다.
* 무한정 학습을 시키면, SPG도 최적으로 수렴할 것으로 예상되기에 위 성능 비교는 일정 iteration 내로 한정합니다.

>> 결국 빠르게 수렴할 뿐, 최종 성능을 비교했을 땐 별 차이 없다는 뜻.?

기존 기법들에 비해 computation 양이 많지 않습니다.

* Computation 은 action dimensionality 와 policy parameters 수에 비례합니다.