1주차 - PyTorch로 시작하는 강화학습 입문 Camp

기계 학습

- 기계학습의 정의

  "컴퓨터를 사용해 T라는 Task를 E라는 경험을 이용해 P라는 퍼포먼스를 취득하면서 풀려고하는 것이다." - *TOM M. Mitchell*

- 기계학습 분류

  Supervised learning - 데이터에 **답**에 대한 정보가 직접적으로 **있을 때** (회귀 , 분류 ...etc)

  Unsupervised learning - **답**에 대한 정보가 데이터에 직접적으로 **없을 때**, (밀도 추정, 군집 분석 ...etc)

  Reinforcement Learning - 알고리즘이 미리 정의되지 않은 **답을 스스로 만들면서** 퍼포먼스(비용함수)를 개선하도록 하는 것

- 기계학습의 비용함수
  - **최소제곱법**

    : 실제값과 예측값의 오차 제곱을 최소화하도록 한다.

    $= \arg\min_\theta \sum_{i \in \mathcal{D}} (y_i - \hat{y}_i)^2$

  - **교차엔트로피**

    : 분포를 비교하여 같으면 0으로 수렴한다. 두 비교대상이 고정된 분산을 가진 가우시안 분포라고 가정하면 교차엔트로피 수식이 최소제곱법의 수식을 포함한다고 한다. 이러한 이유로 더욱 보편적인 비용함수로 사용된다.

    $= \arg\min_\theta \sum_{i \in \mathcal{D}} \sum_{k \in \mathcal{K}} -p_i(k) \log \hat{p}_i(k)$

  - **Kullback-Leibler divergence**

    : 분포를 비교하여 같으면 0으로 수렴한다. 다만 로그 앞에 두는 분포에 따라 값이 달라지기때문에 엄밀한 의미에서 "거리"라고 할 수 없다.(대칭적이지 않기 때문), 개인적인 해석으로는 e두 분포와의 거리를 고려하면서 동시에 자기 자신과의 거리를 고려하는 것

    - 교사학습에서는 로그앞이 타겟의 분포라면 크로스엔트로피를 포함 (크로스엔트로피에서 자기 자신의 분포의 irregulation한 정보 또한 고려하는 것)

      $\min_\theta \sum_{i \in \mathcal{D}} D_{KL}(p_i, \hat{p}_i)$

      $= \min_\theta \sum_{i \in \mathcal{D}} \sum_{k \in \mathcal{K}} -p_i(k) log \hat{p}_i(k) + p_i(k) \log p_i(k)$

    - 생성 모델에서는 로그 앞에 우리가 만드는 데이터의 분포를 많이 사용한다. (실제 데이터와의 차이를 줄이되, 평평한 분포로 (해당 클래스 내에서 노이즈가 다양한, 다시말해 만들어지는 샘플이 다양한) 만들어주기 때문.

$$= \min_\theta \sum_{i \in \mathcal{D}} \sum_{k \in \mathcal{K}} -\hat{p}_i(k) \log p_i(k) + \hat{p}_i(k) \log \hat{p}_i(k)$$

뒤 쪽의 term이 우리가 모델링하는 모델의 엔트로피 정규화 작용하기 때문에 KL을 목적(비용)함수로 두고 학습할 경우에 좀더 다양한 출력을 얻을 수 있다고 한다.

- 딥러닝
  - 기저함수를 이용한 비선형 특징 추출
    - 선형모형 - 출력값과 가중치 간의 관계가 선형인 경우

      $\hat{Y} = W^T X$
    - 비선형 모형 - 출력값과 가중치 간의 관계가 비선형인 경우

      $\hat{Y} = \phi(W^T X)$ ::: $\phi =$ basis function
- CNN/RNN 기초
  - CNN
    - 공간적 구조를 반영하기 위한 Convolution 계산
    - Convolution == 기본적으로는 선형 연산
    - Convolution연산 결과를 비선형 함수에 통과시켜 비선형 커널을 여러가지 만든다.
  - RNN
    - 시계열을 처리하기 위한 뉴럴넷 아키텍쳐
    - 이전 스텝에서 도출된 어떠한 값을 다음의 테스크를 해결할 때 그 값을 고려하도록 하는 것
    - 메인은 아니기 때문에 컨셉만 다루었다.
    - 보통은 LSTM/GRU 를 Stack 하여 사용한다.
- 정규화 기법
  - Weight Decay
    - 네트워크의 파라미터가 커지는 것을 비용함수에 반영
    - 가중치의 정규화 term을 비용함수에 포함 시킨다.
    - 보통 가중치의 정규화를 L2 방식으로 한다. (이를 추가하면 가중치의 제곱합을 최소화로 하면서 목적함수를 최소화하도록한다.)
  - Dropout
    - 일정한 확률로 뉴럴넷의 노드를 지우는 것이다.
    - 서로 다른 네트워크를 학습시키는 효과가 발생하여 앙상블 모델의 이점을 얻을 수 있다고 한다.
    - 이러한 확률적인 network는 하나의 베이지안 신경망(Bayesian neural network)로도 해석이 가능
      - 관련 논문
        - https://arxiv.org/abs/1506.02142 (베이지안 최적화 관점에서 Dropout효과에 대한 논문이었다.)
        - https://arxiv.org/pdf/1506.02557.pdf (Variational Dropout에 대한 설명으로 노드를 지우는 것이 아니라 미리 정의한 확률분포에서 샘플링한 노이즈를 입력 데이터와 곱한 뒤(요소별로) 가중치와의 weighted sum을 진행하는 것이다.)
  - Batch normalization
    - 미니배치에서 얻은 평균과 분산으로 출력을 정규화(whitening)한 뒤 학습하는 파라미터 (γ, β)를 각 곱하고 더해주어서 계산값을 표준화하는 방법
    - 내부적으로 바뀌는 분포를 보정하기 위해 사용

        - 최적화 시에 더 계산을 안정적으로 하기 위해
- 생성 모델
    - 강화학습에서의 생성 모델
        - Q-learning
        - Policy gradient ;정책경사

          정책경사의 또다른 이름 - **Score fucntion trick** (확률에 로그를 취함)

            - 기존에 생성모델을 학습하기 위해 사용하던 도구 중 하나이다. 현재는 다음에 배울 **Reparameterization Trick**을 사용한다.
        - 다룰 내용은 GAN, VAE, Soft actor-critic (Policy-based model 중 SOTA로 4주차에 다룰 것이다.)
    - **Reparameterization Trick**
        - 먼저, 다음에 나오는 수식처럼 기댓값의 그레디언트에서 기댓값 안의 변수에 대한 그레디언트를 고려할 때는 기댓값의 그레디언트는 그레디언트의 기댓값과 같다. 이는 미니배치를 이용한 반복 학습을 할 때 사용된다.

$\rightarrow \ \nabla_\theta \mathbb{E}_{p(x)}[C(x,\theta)] = \mathbb{E}_{p(x)}[\nabla_\theta C(x,\theta)]$

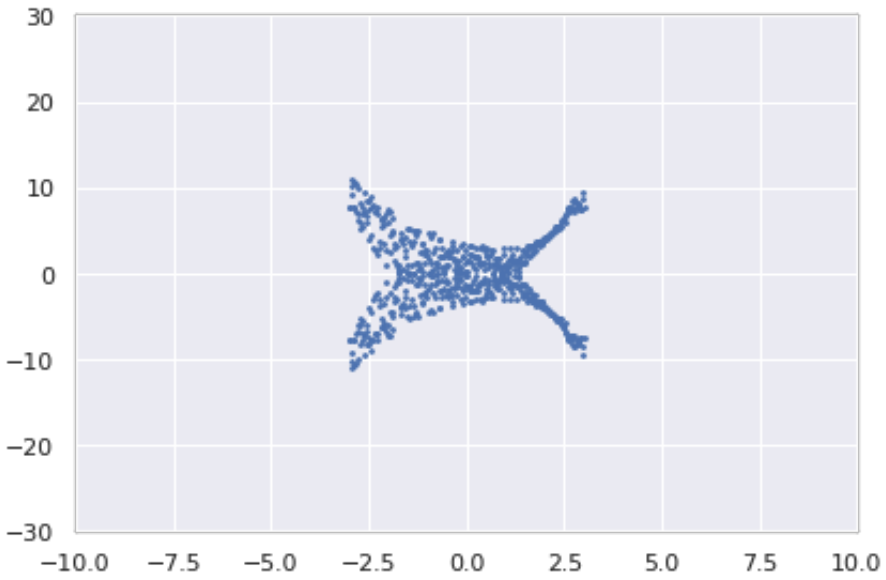
        - 여기서 x 가 모델에 의해서 만들어지게되면(모델의 출력은 원래 결정론적 값이기에 확률변수처럼 취급할 수 없고 모델의 출력에 확률적 요소가 있으면 미분이 불가능하다.)

          $\rightarrow \ \nabla_\theta \mathbb{E}_{p_\theta(x)}[C(x,\theta)] \neq \mathbb{E}_{p_\theta(x)}[\nabla_\theta C(x,\theta)]$
        - 그렇기 때문에 Reparameterization Trick은 모델과 무관하게 확률적으로 샘플링되는 noise를 모델이 변환하여 x를 출력하게끔 하도록하여 미분 가능하면서 해당 확률분포(만들어내는 출력값)의 확률적 특성(또는 확률 분포의 특성)을 유지하도록 한다.
- **GAN 실습**

  [실습 과정]

  1. 학습용 가상데이터 생성 - 선형분할된 x 를 +, - 제곱함수를 이용해 데이터를 생성한다.

     <생성된 데이터>



  2. 생성기 구성

생성기는 Input으로 하나의 스칼라 값과 5개 원소의 벡터 Noise를 입력받아서 하나의 스칼라 값을 출력한다.

이 때, 추가적인 정보를 다룬다는 의미로 Conditional GAN이라고 표현하였다.

학습을 진행하면서 하나의 x 에 10개의 noise vector를 사용해(**n_particle** in the code) 10개의 출력값을 출력하여 타겟값과 비교한다.

3. 판별기 구성

   판별기 구성에 있어 학습 시에 다른 비용함수 선언을 사용하지 않고 판별기의 출력값이 곧 비용(loss)가 되도록 설계하였다.

4. 학습과정 설계

   판별기와 생성기를 차례대로 SGD방식으로 학습 시킨다.

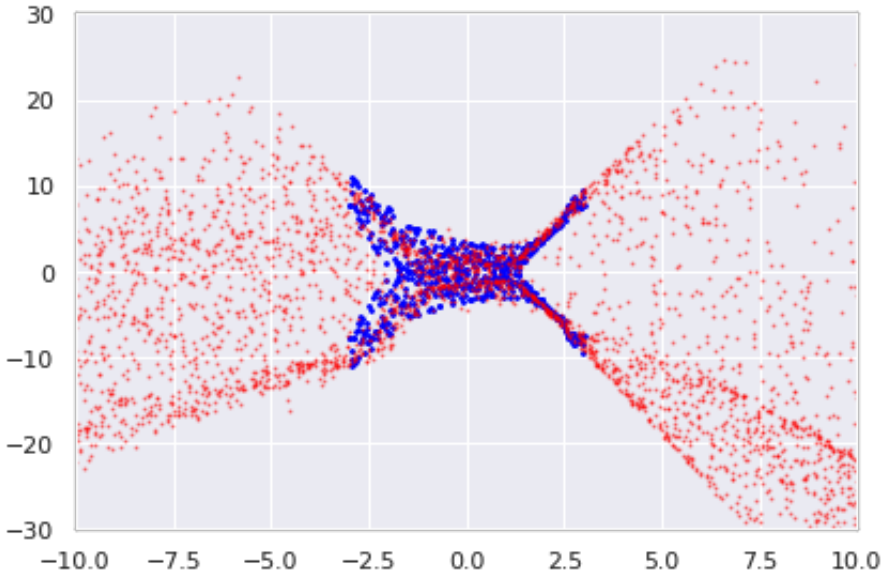
   학습시에는 인위적으로 Gradient 값을 0.1로 clipping 하는 트릭을 사용했다.

5. 학습

   optimizer는 Adam을 amsgrad=True 하여 설정했다. 이는 일반 Adam보다 더 안정적인 학습한다는 정도만 설명해주셨다.

6. 학습 결과 확인(시각화)

   <학습결과>



강화학습 기초

MDP

- **MDP** (마르코프 의사결정 과정)의 정의

  MDP는 문제, 상황 그 자체 (이를 풀기위한 알고리즘을 대변하는게 정책 함수)

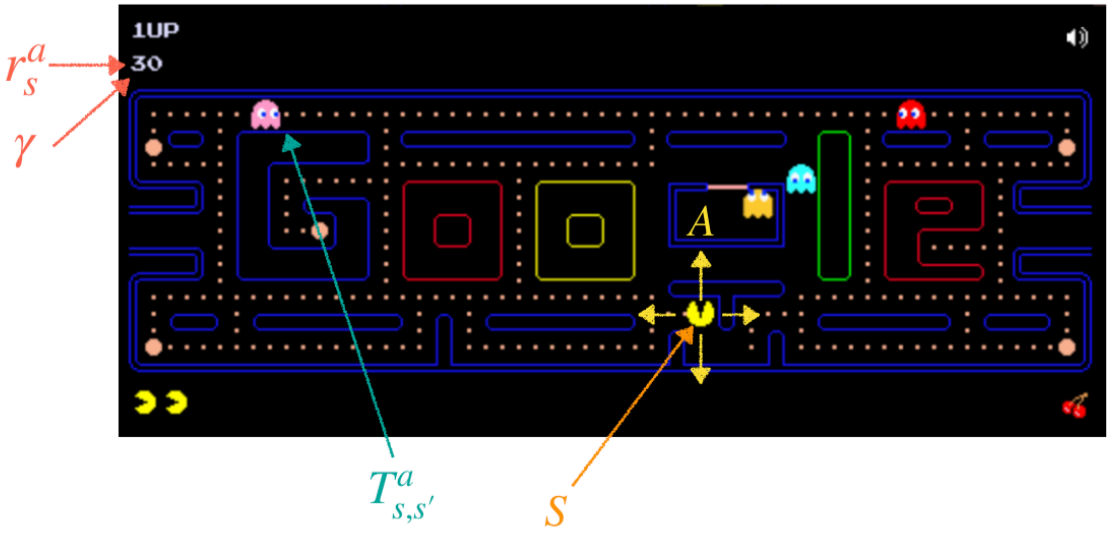
  1. 의사결정자; Agent 와 환경 Environment 가 있다.
  2. 의사 결정자는 상태;State 에 따라, 행동;Action을 고르고
  3. 환경;Environment 는 행동;에 따라 상태;State를 제공한다.
  4. 이 둘은 방금 전 (한 시점 이전의) 행동만을 고려(기억)한다.

- Denoting MDP

- 상태 집합 (S) : 환경이 제공할 수 있는 모든 상태의 집합
- 행동 집합 (A) : agent 가 고를 수 있는 모든 행동의 집합
- 전이확률 ($T^a_{s,s'}$) : 상태와 행동이 정해졌을 때, 환경이 다음 상태를 고르는 규칙(확률분포)
- 보상 (r^a_s) : 상태와 행동이 정해졌을 때 환경이 Agent에게 주는 효용 ; reward
- 할인률 (γ) : 현재 시점과 다음 시점의 가치 차이에 대한 보상의 비율, 추후 학습 수렴에 큰 역할 을 할 수 도 있다.

- 팩맨을 대상으로한 MDP Denoting

  <팩맨을 대상으로한 MDP 정의>



- **정책 함수와 분포**
  - **정책함수 (Policy function, Deterministic policy)**

    상태(s)가 주어졌을 때, Agent 가 행동을 고르는 규칙

    Agent 마다 정책함수가 다르다. 따라서, 다른 정책함수는 다른 Agent를 의미한다.

    $\pi(s) = a$ (policy 의 p 를 그릭문자 π로 표기하는 것)

    함수는 하나의 입력이 있을 때 output은 하나 밖에 못 고른다.
  - **정책분포 (Policy distribution, Stochastic policy)**

    상태(s)가 주어졌을 때, 에이전트가 행동을 고르는 확률적인 규칙

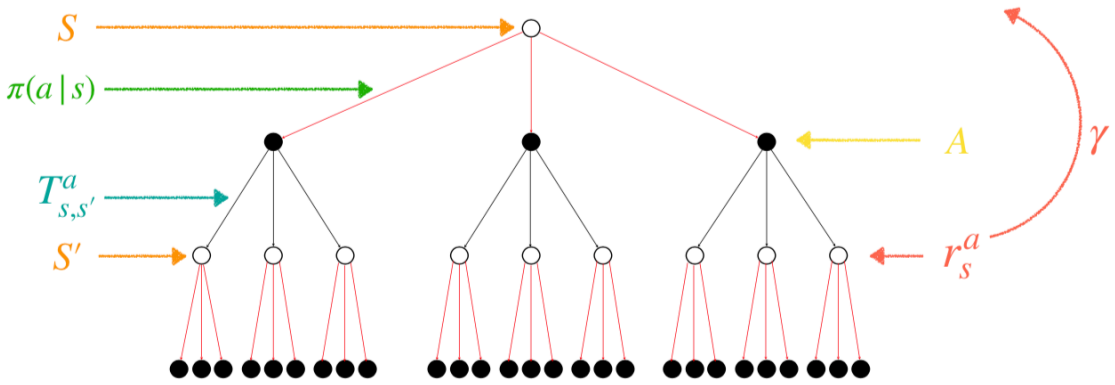
    Agent가 시간에 따라 달라지는 것

    $\pi(a|s) = 0.3$ - 조건부 확률

    정책 분포에서 조건부확률이 1.0되는 경우에는 이것이 정책함수라고 할 수 있다.

- MDP 의사결정 과정의 표현

  < MDP의사결정 과정 >



- **Agent의 목표(Goal)**

  위 "MDP의사결정 과정"그림에서 위에서 아래로 시점이 이동(시간이 흐름)한다. 이 때 보상; t^a_s 는 매시점 action 과 state 이 주어졌을 때 환경이 Agent에게 주는 효용이므로 이를 하나로 합칠 필요가 있다.(Optimal agent(또는 정책)를 찾기 위해서)

  수확 (Retrun; R)

  매 시점에 발생하는 보상(r^a_s) 을 할인률(γ)을 고려해 하나로 합친것

  - t 시점부터 끝까지의 보상을 할인률을 곱하여 더한 것.

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

  - 이 때 k 시점 뒤의 보상은 단순하게 γ^k 를 곱하여(k 번 할인하여) 더해준다.

- **MDP의 목적함수**

  각 시점에서의 보상은 선택한 행동과 상태에 따라 확률적으로 정해지는 값이므로 모든 경우의 수에 대한 보상을 고려하기 위해서 이를 확률적으로 합쳐줘야 한다. 따라서 정책 분포의 기댓값을 사용함으로써 수확(R)을 확률적으로 합친 것이 목적 함수이다.

  $$\max_{\theta} J(\pi_\theta) \doteq \max_{\theta} \mathbb{E}_{\pi_\theta}[R_0] = \mathbb{E}_{\pi^*_\theta}[R_0]$$

  MDP의 목적함수는 정책분포를 조정하는 θ 를 조정해서 초기 수확(Return at the first time, R_0)을 최대로 만드는 것(최적의 정책분포)과 거의 같다.

  강의에서는 최적경로 문제를 예로하여 A 지점 부터 C지점까지 가는 A에서 B지점까지 가는길이 최적이 아니라면 A에서 B를 가는한 이는 최적경로가 아니다라고 설명하였다.

  이 가정 때문에 한계가 존재한다.

- 따라서, MDP Agent의 목표
  - 적절한 상태에서 적절한 행동을 선택하여 목적 함수를 최대화

    $\rightarrow$ 상태와 행동이 얼마나 좋은지는 **가치 함수**를 통해 알 수 있다.

- **가치 함수(Value functions)**
  - 행동 가치 함수 (Action value function) ; $Q(s,a), s \in S, a \in A$

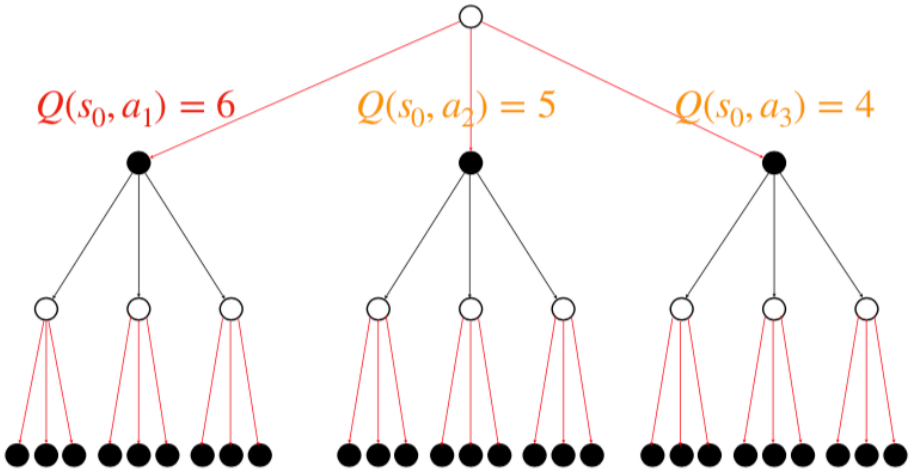
상태(s)마다 각 행동이 얼마나 좋은지를 판단

- 각 상태마다 어떤 행동이 제일 좋은 지를 알 수 있다.

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[R_t | s_t = s, a_t = a]$$

: t 시점에서의 상태 (s_t), 행동(a_t) 그리고 나머지 행동은 기존의 정책을 따를 때 기대되는 수확

<행동가치함수 표현의 예>



- 상태 가치 함수 (state value function)

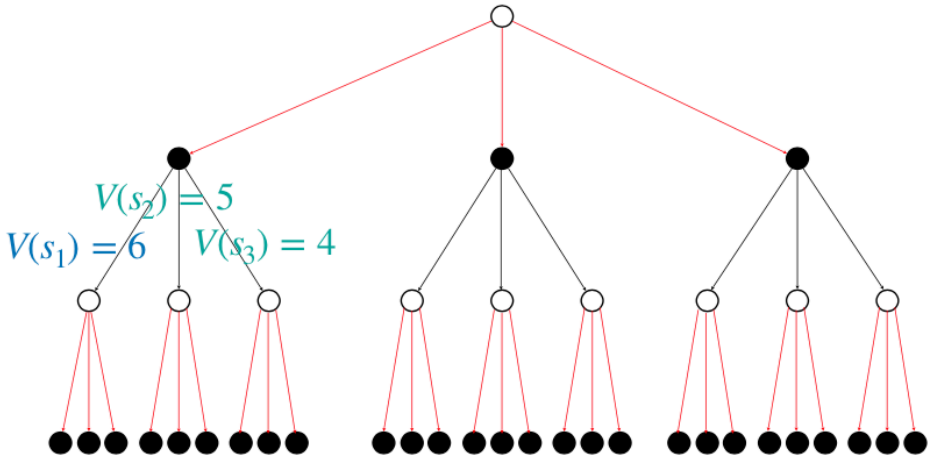
  각 상태가 얼마나 좋은지를 판단

  - 그 상태에 도달한 것이 얼마나 좋은지를 알려줌 (전이 확률을 모두 알고 있다는 가정)

  $$V^{\pi}(s) = \mathbb{E}_{\pi}[R_t | s_t = s]$$

  : t시점의 상태 (s_t)이고 행동은 기존의 정책을 따를 때 기대되는 수확

  <상태가치함수 표현의 예>



- 가치 함수 간의 관계

  행동 가치 함수에서 t 시점의 행동(a_t)또한 기존의 정책을 따르면 다음처럼 상태 가치 함수와 같아 진다.

  $$V^{\pi}(s) = \mathbb{E}_{a \sim \pi(\cdot|s)}[Q^{\pi}(s,a)]$$

  그리고 t시점의 행동을 a 로 고정하면 $a_t = a$

$$Q^{\pi}(s,a) = r_s^a + \mathbb{E}_{\pi, T_{s,s'}^a}[V^{\pi}(s')]$$

위 식에서 t 시점에서의 행동가치함수의 기댓값이 곧 t시점에서의 상태가치함수 값이므로 현재 시점의 상태와 행동의 행동가치함수의 값은 현재 시점의 행동과 상태에 대한 보상을 이전 시점의 상태 가치 함수값의 기댓값과 더한 것이다.

- **일관성**

  위의 관계를 좀더 응용하여 t+1 시점의 행동 또한 원래의 정책을 따른다고 하면 방금 행동가치함수에서 상태가치함수는 다음 처럼 표현 할 수 있다.

  $$Q^{\pi}(s,a) = r_s^a + \mathbb{E}_{T_{s,s'}^a}[\mathbb{E}_{a' \sim \pi(\cdot|s')}[Q^{\pi}(s',a')]]$$

  그리고 t시점에서의 행동(a_t)를 a 로 고정하면 상태가치함수는 다음 처럼 표현 할 수 있다.

  $$V^{\pi}(s) = \mathbb{E}_{a \sim \pi(\cdot|s)}[r_s^a + \mathbb{E}_{\pi, T_{s,s'}^a}[V^{\pi}(s')]]$$

  이 두 수식은 각각 Optimal 한 행동가치함수, Optimal 한 상태가치함수가 만족해야하는 조건이다.

  $\rightarrow$ 이를 정책에 대한 일관성(Consistency)라고 표현한다고 한다.

- **Optimal 조건을 만족하지 못한 경우**
  - *틀린가치함수를 기반으로 새로운 가치함수를 찾는다.* (최종적으로는 일관성을 유지하는 가치함수를 찾는다.)

  이 점에 기인해 "알고리즘이 스스로 답을 만든다"고 표현한 것이다.

  이에 대해 동적계획법 혹은 강화학습이라고 표현한다. (수식을 보면 결국 전체의 가치함수값을 구하는 것은 재귀함수를 푸는 것과 비슷해보인다.)

- **정책 반복 알고리즘 (Policy iteration)**
  - **Police Evaluation** : 가치 함수를 추정하는 것

    Optimal 조건식의 좌 우변이 같지 않는 경우!

    $\rightarrow$ **좌변을 우변으로 대체**

    $$Q^{\pi}(s,a) \leftarrow r_s^a + \mathbb{E}_{T_{s,s'}^a}[\mathbb{E}_{a' \sim \pi(\cdot|s')}[Q^{\pi}(s',a')]]$$
  - **Police Improvement** : 정책을 더 좋게 만드는 것
    - 현재 가치 함수가 알려주는 가장 좋은 행동을 선택

    $$\pi(a'|s) = \begin{cases} 1 & Q^{\pi}(s,a') = \max\limits_{a \in A} Q^{\pi}(s,a) \\ 0 & Q^{\pi}(s,a') \neq \max\limits_{a \in A} Q^{\pi}(s,a) \end{cases}$$

    - 현재의 가치함수가 나타내는 가장 좋은 행동을 선택하기 때문에 **Greedy update**라 표현한다.
    - 이렇게 계산하면 최적 정책을 찾을 수 있다는 것은 증명되어있다.(by Richard Bellman)
- 결과적으로 최적 정책은 항상 가장 가치가 높은 행동만을 선택 한다.

  그래서 다음처럼 행동가치함수, 상태가치함수의 일관성 식, Optimal 조건식을 다음처럼 바꿀 수 있다.

  $$Q^{\pi}(s,a) = r_s^a + \mathbb{E}_{T_{s,s'}^a}[\max_{a \in A}[Q^{\pi}(s',a')]]$$

  $$V^{\pi}(s) = \max_{a \in A}[r_s^a + \mathbb{E}_{\pi, T_{s,s'}^a}[V^{\pi}(s')]]$$

- **가치 반복 알고리즘 (Value iteration)**
  - 가치 함수를 추정 : 아래 식의 등호가 성립하지 않으면 좌변을 우변으로 대체

$$Q^{\pi}(s,a) \leftarrow r_s^a + \mathbb{E}_{T_{s,s'}^a}[\max_{a \in A}[Q^{\pi}(s',a')]]$$

  - 정책 : 행동가치 함수값이 제일 높은 행동을 선택
  - 이렇게 계산해도 최적 정책을 찾을 수 있다.
- Atari를 이용한 Policy iteration, Value iteration 실습

  State space와 action space가 모두 셀 수 있는 MDP를 'Grid world'라고 하는데, Policy iteration과 Value iteration은 Grid world 에서만 적용가능하다.

질문

- 문제에 따라서 어떤 것은 state가 action보다 먼저 주워지고 그 반대인 경우도 있을텐데, 두가지 경우에 대해서 차이가 있는가?
  - 답 - State 와 action 중 어느 것이 먼저 인가는 중요하지 않다. state가 먼저 일 때는 state를 전제조건으로 취급하고 Action을
- MDP문제를 풀 때, 전이 확률을 아는 경우와 모르는 경우에 같은 방식으로 계산하는가 ?
  - 답 - 현재는 전이 확률을 아는 경우에 대해서 전개하는 것이고 추후 전이확률을 아는지 여부가 중요하게 다루워 질 것이다.
- 초기 수확에 집중하다보면, Local mInima에 빠진다거나 하는 문제는 발생하지 않는가 ?
  - 답 - Transition 확률을 알 때, 초기 수확을 집중한다는 말은 수학적으로 보상과 할인률, 해당 transition 확률을 고려해서 구하기 때문에 local optima문제는 생기지 않는다.