강화학습특론(3)

Created	Done
Tags	SUMMARY

<강화학습의 목적>

- optimal reward 를 얻기 위해 agent 에게 optimal 한 behavior strategy 요구
- policy gradient 수식 π = policy, θ = 특정 paramete, π_θ = 특정 parameter 구성된 함수
- bellman equation 에서 나오는 reward (objective) function 값은 policy 영향 받음
- 최적의 reward 를 얻기 위해 θ를 최적화 함

<베이즈 정리>

- 데이터 조건이 주어졌을 때 조건부 확률을 구하는 공식

- 데이터가 주어지기 전 사전 확률값 데이터가 주어지면 어떻게 변하는지 계산
- 데이터가 주어지기 전 이미 어느정도 확률값을 예측하고 있을 때

  이를 새로 수집하나 데이터와 합쳐 최종 결과에 반영
- 데이터 개수가 부족한 경우 아주 유용

  (데이터를 매일 추가적으로 얻는 상황에서 매일 전체 데이터를 대상으로 새로 분석작업을 할 필요 없이

  오늘 들어온 데이터를 합쳐서 업데이트만 하면 됨)

베이즈 정리 확률 밀도 함수식

$$P_x(X) = \int_{-\infty}^{\infty} P_{XY}(x, y)dy = \int_{-\infty}^{\infty} P_{X|Y}(x|y)P_Y(y)dy$$

전체 경우의 수 Y, y - 조건이 발생할 경우의 수 X, x

$$P\{B_i|A\} = \frac{P\{A,B_i\}}{P\{A\}} = \frac{P\{A|B_i\}P\{B_i\}}{P\{A\}} \qquad P\{A\} = \sum_{j=1}^{n} P\{A|B_i\}P\{B_j\}$$

$$P_{Y|X}(y|x) = \frac{P_{X|Y}(x|y)P_Y(y)}{P_X(x)} = \frac{P_{X|Y}(x|y)P_Y(y)}{\int_{-\infty}^{\infty} P_{X|Y}(x|y)P_Y(y)dy}$$

total probability - y 교집합 x

<가우시안 분포>

- 정규 분포 (normal distribution) 또는 가우스 분포 (Gaussian distribution)
- 연속 확률 분포의 한 종류로 수집 된 자료의 분포를 근사 하는데 주로 사용
- 중심 극한 정리에 의해 독립적인 확률변수들의 평균은 정규 분포에 가까워지는 성질 이용

- 가우시안 랜덤벡터 (Gaussian Random variables)
  - 랜덤 변수 X 에 대한 평균과 표준편차를 이용해 PDF 관한 식을 세울 수 있을 때, 그 때의 식을 normal density function 이라고 하며 PDF 를 가우시안 밀도라 하며 이러한 랜덤 변수 X 가 가우시안 랜덤벡터
  - **랜덤 변수의 평균과 표준편차를 이용해 정규밀도함수 작성 가능**

  - 선형변환도 가우시안 랜덤벡터

$$X \sim N(\mu_X, P_{XX}) \rightarrow \; Z = AX \sim N(A_{\mu X}, AP_{XX}A^T)$$

  - 랜덤벡터 X, Y 가 결합 가우시안 분포 일 때 X, Y 도 각각 가우시안 랜덤벡터

$$P_{XY}(x, y) = P_z(z) = N(z|\mu_z, P_{ZZ}) \; \rightarrow \; X \sim N(\mu_X, P_{XX}), Y \sim N(\mu_Y, P_{YY})$$

  - 랜덤벡터 X, Y 가 결합 가우시안 분포 일 때 비상 단계이면 서로 독립
  - 랜덤벡터 X, Y 가 결합 가우시안 분포 일 때, X 조건부 확률 밀도 함수이면 Y 조건부 확률 밀도 함수도 가우시안

$$P_{Y|X}(y|x) = \frac{1}{\sqrt{(2\pi)^r \det P_{Y|X}}} X \exp(-\frac{1}{2}\{(y - \mu_{Y|X})^T P_{Y|X}^{-1}(y - \mu_{Y|X})\})$$

$$P_{Y|Y} - P_{YX}P_{XX}^{-1}P_{XY} \qquad\qquad \mu_Y + P_{YX}P_{XX}^{-1}(x - \mu_X)$$

가우시안 로그-정책 확률 밀도함수

→ A2C 알고리즘에서 actor 행동변수가 서로 독립인 가우시안이라 가정

$$\log \pi_\theta(u_t|x_t) = \sum_{j=1}^{m} \log \pi_\theta(u_{t,j}|x_t)$$

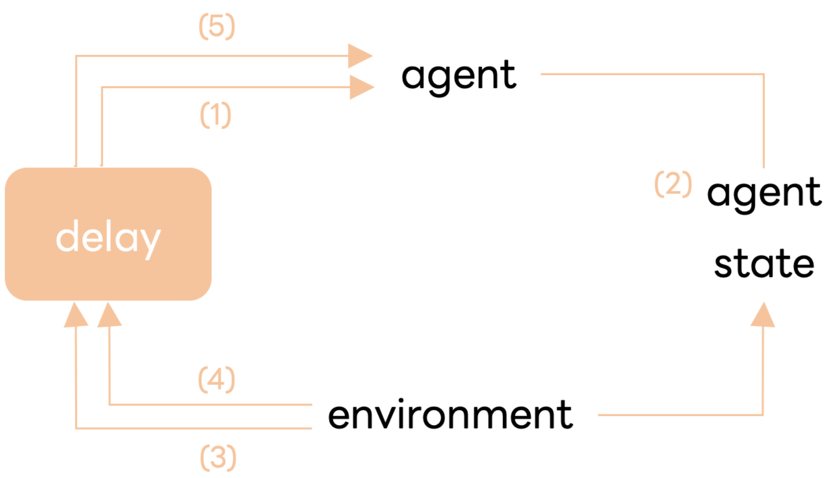
$$\frac{\{u_{t.j} - u_{\theta.j}(x_t)\}^2}{2\sigma_{\theta,j}^2(x_t)} + \frac{1}{2}\log(2\pi\sigma_{\theta.j}^2(x_t))$$

<마르코프 시퀀스>

- 현재 확률 정보가 주어진 상태에서 미래와 과거는 무관(=조건부 독립)한 랜덤 시퀀스
  → 확률 정보는 현재의 확률 정보에 녹아있다.

$$P_X(x|t)|x(s), s \le t_1) = P_X\left(x(t)\middle|x(t_1)\right), \forall t > t_1 \rightarrow \text{ Markov Process}$$

강화학습에서 에이전트와 환경 상호작용



1. agent 가 환경측정
2. agent 가 state 인허가
3. agent 행동에 의해 환경이 다음 상태로 전환
4. 전환된 환경 상태 반영하여 agent 새로운 행동 선택
5. 환경으로부터 보상 → 장기적 성과, 예측 통해 주기적 개선

강화학습 방법

환경 모델 추정(모델 기반 - 로봇 제어, 드론 제어) vs 가치 함수 추정(가치 기반 - DQN, actor-critic 구조)

모델가치 함수 추정 → 정책 개선 → 정책 실행 샘플 생성(정책파라미터 Q 계산목적) → 모델가치 함수 추정

<벨만 최적 방정식>

최적 상태 가치

- 최적 상태 가치를 무한히 반복하여 계산하면 V(Xt) 값이 수렴하게 되고 정책 π 에 대한 상태 가치임

$$V(x_t) = \max_{\pi} E_{ut\sim\pi(ut|xt)}[r(x_t,u_t) + \mathbb{E}x_{t+1\sim p(xt+1|x_t,u_t)}[rv^{\pi}(x_{t+1})]]$$

- 최적 정책을 찾아야 하기 때문에 정책 π 를 따라 갔을 때 받는 보상의 합인 가치 함수를 통해 판단해야함

$$= \max_{ut}\{r(x_t,u_t) + \mathbb{E}_{xt+1\sim p(xt+1|xt,ut)}[rV(x_{t+1})]\}$$

- 최적 상태 가치가 최대 일 때 최적 정책을 정해야 함

  → 여러가지 정책 중 상태 가치를 최대로 만드는 정책 π 에서의 최적 상태 가치 함수 V(Xt)

$$= \max_{\pi} V^{\pi}(x_t)$$

최적 행동 가치

- 최적 상태 가치가 확정적일 때 행동 가치 함수에 대한 벨만 방정식

  → 아래 식이 벨만 최적 방정식(Bellman optimality equation)

$$Q(x_t, u_t) = r(x_t, u_t) + max\{\mathbb{E}_{xt+1 \sim p(xt+1|x_t,u_t)}[rV^{\pi}(x_{t+1})]\}$$

$$= r(x_t, u_t) + \mathbb{E}_{xt+1 \sim p(xt+1|x_t,u_t)}[r \max_{ut+1} Q(x_{t+1}, u_{t+1})]$$

최적 정책

- 최적 행동 가치 식은 현재의 최적 가치와 시간 스텝의 최적 상태 관계

$$\pi(x_t) = arg\max_{ut}[r_t + \mathbb{E}_{xt+1 \sim p(xt+1|x_t,u_t)}[rV(x_{t+1})]]$$

최적 상태 가치 함수 이용

$$argmax_{ut}Q(x_t, u_t)$$

Reinforce Algorithm-Monte Carlo

- 한 에피소드가 끝나야 정책 업데이트 가능(On-policy)
- gradient 분산이 매우 큼

Workflow - Reinforce Algorithm

- episode(1) $\pi_\theta(u_i|x_i)$ → $(T_1 + 1)$ 개 샘플
  - 샘플 생성

- 반환값 계산(=object function gradient)

$$\nabla_\theta J(\theta) \approx \frac{1}{M}\sum_{m=1}^{M}[\sum_{t=0}^{T}\{\nabla_\theta \log \pi_\theta(u_t^{(m)}\,|x_t^{(m)})\,(\sum_{k=t}^{T} r^{k-t} r\,(x_k^{(m)}, u_k^{(m)})\}]$$

- Loss function 계산

$$loss = -\sum_{t=0}^{T}(\log \pi_\theta(u_t^{(m)}|x_t^{(m)})G_t^{(m)})$$

- update (episode (1) 샘플 폐기)

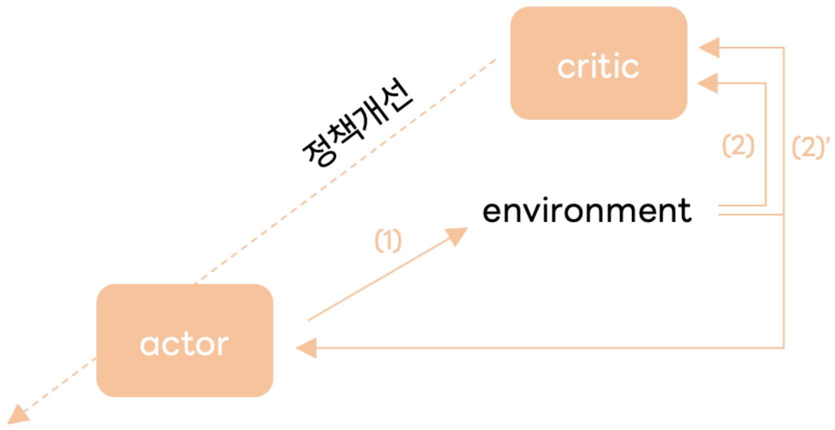
$$\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$$

- episode(2)
- update (episode (2) 샘플 폐기)
- …

A2C Algorithm

- 목적함수 최대 설정하여 항상 1인 교차 entropy 에 advantage 곱함 따라서 advantage 영향 많이 받음
- 정책과 asynchronous(On-policy)
- 가치 함수 학습 시 사용되는 샘플에 시간 상관

Workflow-A2C



정책 개선 = 배치 A2C/온라인 A2C → $u_i \sim \pi_\theta(u_i|x_i)$

1. 정책으로 행동 확률적 선택
2. 정책 개선

   2-1. 정책 실행하여 보상과 상태변수 측정하고 샘플 저장

   TD(Temporal Difference) Learning = 다음 step 미래추정가치 사용해 학습

   TD target = 보상 + value function 계산

$$y_i = r(x_i, u_i) + \gamma V_\emptyset(x_i + 1)$$

   2-2. 크리틱 신경망 손실함수 계산

$$L = \frac{1}{2}\sum_i (y_i - V_\emptyset(x_i))^2$$

$$Loss_{critic}(\emptyset) = \frac{1}{2}\sum ||r(x_i, u_i) + \gamma V_\emptyset(x_{i+1}) - V_\emptyset(x_i)||^2$$

$$Loss_{actor}(\theta) \approx -\sum_i (\log \pi_\theta(u_i|x_i) A_\emptyset(x_i, u_i))$$

2-2-1. actor 신경망 업데이트

$$\theta \leftarrow \theta - \alpha \nabla_\theta \sum_i -(\log \pi_\theta(u_i|x_i) A_\emptyset(x_i, u_i))$$

2-3. advantage 계산

$$A_\emptyset(x_i, u_i) = r(x_i, u_i) + \gamma V_\emptyset(x_{i+1}) - V_\emptyset(x_i), i = 1 \dots N$$

→ 정책 개선하고 actor update

$$\theta \leftarrow \theta + \alpha_{actor} \nabla_\theta \sum_i (\log \pi_\theta(u_i|x_i) A_\emptyset(x_i, u_i))$$

Actor class - A2C

state(3X1) → h1(64) →(ReLU) h2(32) →(ReLU) h3(16) →(ReLU) 평균값/표준편차 → Tanh/softplus

→ θ = 2898

Critic class - A2C

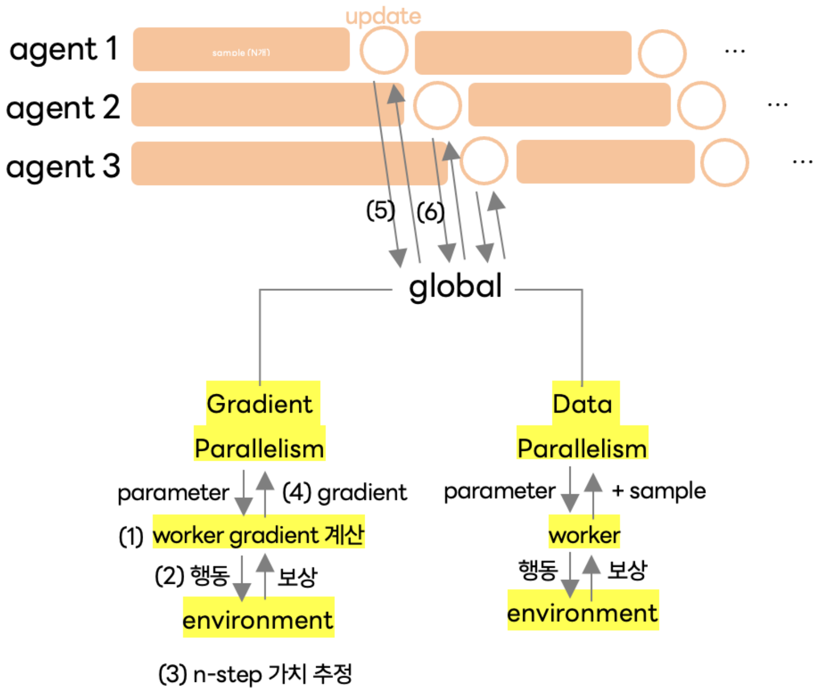
state(3X1) → h1(64) →(ReLU) h2(32) →(ReLU) h3(16) →(ReLU) 상태가치

→ θ = 2881

A3C Algorithm-Asynchronous advantage actor-critic

- Multi Thread 사용해 여러 하위 agent 비동기식 작동하여 에피소드 간 correlation 없음 (On-policy)

Workflow-A3C



N 개 샘플 → $(x_i, u_i, r_i, x_{i+1})_N$

1. 워커의 정책 확률적으로 선택하고 샘플 저장
2. 워커의 n-step 시간차 target $y_{w,i}$ 계산
3. 워커의 n-step advantage 계산

    3-1. n-step 가치 추정(목적함수 gradient 계산 시 advantage 편향 줄임)

$$V_\phi(x_t) \approx r(x_t, u_t) + \gamma r(x_{t+1}, u_{t+1}) + \cdots + \gamma^{n-1} r(x_{t+n-1}, u_{t+n-1}) + \gamma^n V_\phi(x_{t+n})$$

$$A_\phi(x_t, u_t) \approx r(x_t, u_t) + \gamma r(x_{t+1}, u_{t+1}) + \cdots + \gamma^{n-1} r(x_{t+n-1}, u_{t+n-1}) + \gamma^n V_\phi(x_{t+n}) - V_\phi(x_t)$$

$$= \sum_{k=t}^{t+n-1} \gamma^{k-t} r(x_k, u_k) + \gamma^n V_\phi(x_{t+n}) - V_\phi(x_t)$$

4. gradient 계산

    4-1. 워커 critic 신경망

$$\sum_{i=1}\left[\left(y_{w,i} - V_{\phi w}(x_i)\right) \nabla_{\phi w} V_{\phi w}(x_i)\right]$$

    4-2. 워커 actor 신경망

$$\nabla_{\theta w}\sum_{i}[\log(\pi_{\theta w}(u_i|x_i))A_{\emptyset w}(x_i,u_i)]$$

5. global 신경망으로 워커 gradient 송부

$$\emptyset \leftarrow \emptyset + \alpha_{critic}\sum_{i=1}\left[\left(y_{w,i} - V_{\emptyset w}(x_i)\right)\nabla_{\emptyset w}V_{\emptyset w}(x_i)\right]$$
$$\theta \leftarrow \theta + \alpha_{actor}\nabla_{\theta w}\sum_{i}[\log\pi_{\theta w}(u_i|x_i)\cdot A_{\emptyset w}(x_i,u_i)]$$

6. 업데이트 된 global 신경망 parameter 워커로 복사

+ sample 추가

$$\emptyset \leftarrow \emptyset + \alpha_{critic}\sum_{i}\left[\left(y_i - V_{\emptyset}(x_i)\right)\nabla_{\emptyset}V_{\emptyset}(x_i)\right]$$

$$\theta \leftarrow \theta + \alpha_{actor}\nabla_{\theta}\sum_{i}[\log(\pi_{\theta}(u_i|x_i)\cdot A_{\emptyset}(x_i,u_i)]$$

Actor class - A3C

state(3X1) → h1(64) →(ReLU) h2(32) →(ReLU) h3(16) →(ReLU) 평균값/표준편차 → Tanh/softplus

→ 로그 가우시안 정책

$$\log\pi_{\theta w}(u|x) = -\sum_{j=1}^{m}[\frac{(u_j - u_{\theta j}(x))^2}{2\sigma_{\theta,j}^2(x)} + \frac{1}{2}\log(2\pi\sigma_{\theta j}^2(x))]$$

Critic class - A3C

state(3X1) → h1(64) →(ReLU) h2(32) →(ReLU) h3(16) →(ReLU) 상태가치

PPO Algorithm-Proximal Policy Optimization

- 근접 정책 최적화-정책 점진적 업데이트

목적함수

$$J(\theta) = \mathbb{E}_{\tau\sim p\theta(\tau)}[\sum_{t=0}^{\infty} \gamma' r(x_t, u_t)]$$
$$\tau = (x_0, u_0, x_1, u_1 \cdots)$$

surrogate

-최대화 위해 TRPO(trust region policy optimization) → $L(\theta)$ 선형화 2차함수로 근사한 후 최적값 구함

$$L(\theta) = \sum_{t=0}^{\infty} \mathbb{E}_{xt\sim p\theta\text{OLD}}(x_t)[\mathbb{E}_{ut\sim\pi\theta\text{OLD}(\text{ut}|\text{xt})}[\frac{\pi_\theta(u_t|x_t)}{\pi_{\theta OLD}(u_t|x_t)} \cdot \gamma^t \cdot A^{\pi\theta\text{OLD}}(x_t, u_t)]]$$

$$= \sum_{t=0}^{\infty} \mathbb{E}_{xo',ut\sim p\theta\text{OLD}(\tau\text{ xo', u t})}[\frac{\pi_\theta(u_t|x_t)}{\pi_{\theta\text{OLD}}(u_t|x_t)} \cdot \gamma^t \cdot A^{\pi\theta\text{OLD}}(x_t, u_t)]$$

-PPO 일정범위로로 한정위해 clipping

$$\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon) = \begin{cases} 1+\epsilon, \text{if } \mathrm{r_t}(\theta) \geq 1+\epsilon \\ 1-\epsilon, \text{if } \mathrm{r_t}(\theta) \leq 1+\epsilon \\ \mathrm{r_t}(\theta), \quad \text{otherwise} \end{cases}$$

$$\mathrm{L_t^{clip}}(\theta) = \min\{\mathrm{r_t}(\theta)\mathrm{A}^{\pi\theta\text{OLD}}(x_t, u_t), \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon) \cdot A^{\pi\theta\text{OLD}}(x, u_t)\}$$

목적함수 일정수준 이상 커지는 것 제한

gradient

$$\nabla_\theta L(\theta) = \sum_{t=0}^{\infty} \mathbb{E}_{xt\sim p\theta\mathrm{OLD}(x_t), ut\sim\pi\theta\mathrm{OLD}(\mathrm{ut|xt})} [\frac{\pi_\theta(u_t|x_t)}{\pi_{\theta\mathrm{OLD}}(u_t|x_t)} \nabla_\theta \log \pi_\theta(u_t|x_t) \cdot \gamma^t \cdot A^{\pi\theta\mathrm{OLD}}(x_t, u_t)]$$

Workflow-PPO

N 개 샘플 → (x_0, u_0, r_0, x_1) → 업데이트 → (x_N, u_N, r_N, x_{N+1})

1. 이전 정책 gaussian 으로 가정하고 평균, 표준편차 계산

   확률 행동 선택 - 이전 정책 로그-확률밀도함수

$$\log \pi_\theta(u_i|x_i) = -\sum_{j=1}^{n} [\frac{(u_{i.j} - \mu_{\theta.j}(x_i))^2}{2\sigma_{\theta j}^2(x_i)} + \frac{1}{2} \log(2\pi\sigma_{\theta.j}^2(x_i))]$$

샘플 $x_1, x_2 \ldots \longrightarrow x_i$

2. 데이터 모음 (정책, reward, advantage, 확률밀도함수값) batch 저장

   (정책 = x_i, u_i / reward = y_i / advantage = A_i / 확률밀도함수값 = $log\pi_\theta OLD(u_i|x_i)$)

3. batch 크기만큼 데이터 추출

   3-1. critic 신경망 손실함수

$$L = \frac{1}{2B} \sum_i (y_i - V_\emptyset(x_i))^2$$

   3-2. critic 신경망 업데이트

$$\emptyset \leftarrow \emptyset + \beta \sum_i [(y_p - V_\emptyset(x_i))^2 \nabla_\emptyset V_\emptyset(x_i)]$$

3-3. 이전 정책, 현재 정책 비율 계산

$$r_t(\theta) = \frac{\pi_\theta(u_i|x_i)}{\pi_{\theta OLD}(u_i|x_i)}$$

4. surrogate gradient 계산하고 actor 신경망 업데이트

$$L_i^{clip} = \min\{r_t(\theta)A^{\pi\theta OLD}(x_i, u_i), clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A^{\pi\theta OLD}(x_i, u_i)\}$$

$$\nabla_\theta L^{clip}(\theta) \approx \nabla_\theta \sum_i L_i^{clip}(\theta)$$

3-3 비율 계산

$$\theta \leftarrow \theta + \alpha \nabla_\theta L^{clip}(\theta)$$

Actor class - PPO

state(3X1) → h1(64) →(ReLU) h2(32) →(ReLU) h3(16) →(ReLU) 평균값/표준편차 → Tanh/softplus

Critic class - PPO

state(3X1) → h1(64) →(ReLU) h2(32) →(ReLU) h3(16) →(ReLU) 상태가치

→ 손실함수

$$L = \frac{1}{2}\sum_i (y_i - V_\emptyset(x_i))^2$$

Agent class - GAE (역방향 시간의 궤환식)

$$\delta_{i+n-1} = r(x_{i+n-1}, u_{i+n-1}) + \gamma V(x_{i+n}) - V(x_{i+n-1})$$

$$A_{i+n-1}^{GAE} = \delta_{i+n-1}$$

$$A_{i+n-2}^{GAE} = \delta_{i+n-2} + (\gamma\lambda)\delta_{i+n-1} = \delta_{i+n-2} + (\gamma\lambda)A_{i+n-1}^{GAE}$$

$$A_{i}^{GAE} = \delta_{i} + (\gamma\lambda)\delta_{i+1} + \cdots + (\gamma\lambda)^2\delta_{i+n-1} = \delta_{i} + (\gamma\lambda)A_{i+1}^{GAE}$$

$\rightarrow y_i = A_i^{GAE} + V_\phi(x_i)$

+ 어드밴티지 추정의 일반화 (GAE)

1-step 관계식 → 어드밴티지 추정값의 분산은 낮추고, 편향은 높임

$$A_\emptyset^{(1)}(x_t, u_t) = r(x_t, u_t) + \gamma V_\emptyset(x_{t+1}) - V_\emptyset(x_t)$$

한개의 에피소드에서는 편향 X, 분산 증가-Monte Carlo

n-step 관계식 → 1, 2 … 스텝 어드밴티지 계산한 뒤 가중 합산

$$A_\emptyset^{GAE}(x_t, u_t) = \sum_{n=1}^{\infty} w_n A_\emptyset^{(n)}(x_t, u_t)$$

가중 합산

$$\lambda = 0 \rightarrow \; A_\emptyset^{GAE}(x_t, u_t) = \delta_t = r(x_t, u_t) + \gamma V_\emptyset(x_{t+1}) - V_\emptyset(x_t) = A_\emptyset^{(1)}(x_t, u_t)$$

$$\lambda = 1 \rightarrow \; A_\emptyset^{GAE}(x_t, u_t) = \sum_{k=t}^{\infty} \gamma^{k-t}\delta_k = A_\emptyset^{(\infty)}(x_t, u_t)$$

DDPG Algorithm-Deep Deterministic Policy Gradient

연속공간 행동변수 → 행동 자체 계산(replay buffer 에 저장하고 샘플 무작위로 꺼냄-experience replay)

- Off-policy 이전 정책 데이터 사용 가능

- On-policy vs Off-policy
  - On-policy
    - 학습하는 policy 와 행동하는 policy 가 반드시 같아야만 학습 가능
    - 1번이라도 학습해서 업데이트 되면 이전 experience 모두 사용 불가
    - Importance sampling 등 일련의 과정을 거쳐야 재사용 가능
  - Off-policy
    - 학습하는 policy 와 행동하는 policy 가 반드시 같지 않아도 학습
    - 이전의 학습을 통해 얻은 experience 들도 새로운 학습에 사용 가능

- DQN(Deep Q-Network) 는 시간차 target 이 신경망 업데이트 할 때마다 계속 달라져 학습 불안정

  함수 근사화 + bootstrapping + Off policy 학습 방법 = Deadly Triad → 안전성 문제

  - target actor network (θ') , target critic network (ϕ') 별도 운영

$$y_i = r(x_i, u_i) + \gamma Q_{\emptyset'}(x_{i+1}, \pi_{\theta'}(x_{i+1}))$$

  - target network 파라미터가 본 network 파라미터 느린 속도로 따라감

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta'$$
$$\emptyset' \leftarrow \tau\emptyset + (1-\tau)\emptyset'$$

$$L(\emptyset) = \frac{1}{2N}\sum_{i}^{N}(y_i - Q_\emptyset(x_i, u_i))^2$$

  - noise 도입하여 무작위성 추가

    $\pi_{noisy}(x_t) = \pi_\theta(x_t) + \varepsilon_t$ → Orientein-Uhlenbeck 노이즈

    $\varepsilon_{t+1} = \varepsilon_t + p(\mu - \varepsilon_t)\Delta t + \sqrt{\Delta t}\sigma n_t$

목적함수

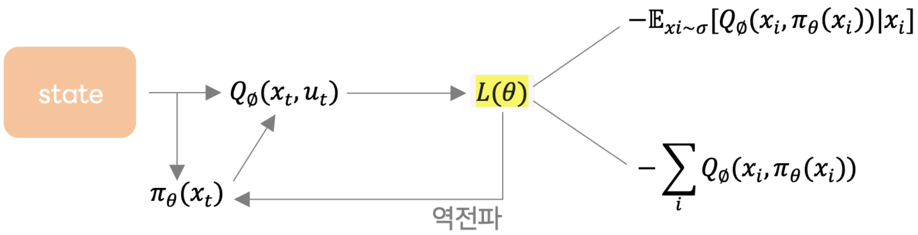
$$J(\theta) = \mathbb{E}_{\tau\sim p\theta(\tau)}[\sum_{t=0}^{\infty}\gamma^t r(x_t,u_t)]$$
$$\tau = (x_0,u_0,x_1,u_1 \cdots)$$

gradient

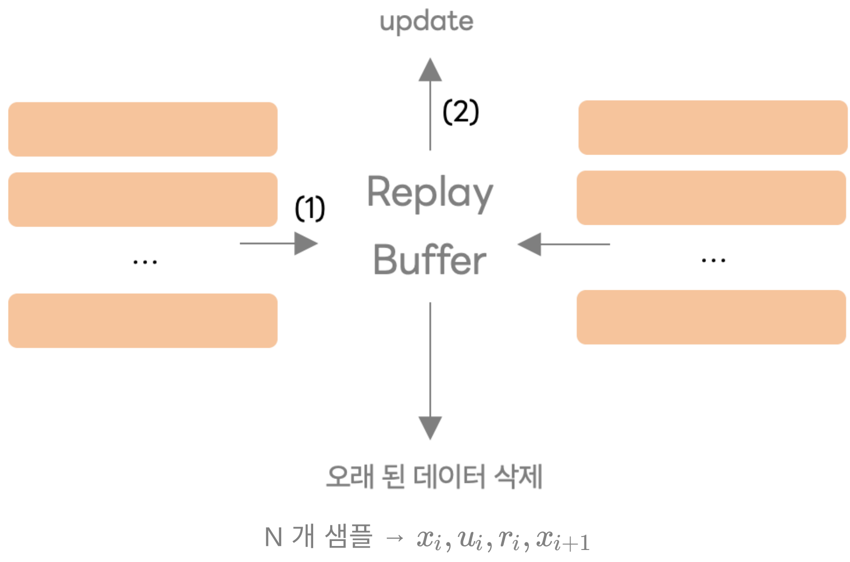
$$\nabla_\theta J(\theta) = \sum_{t=0}^{\infty}(\mathbb{E}_{\bar{\tau}x_0:x_t\sim p\theta_{OLD}(\bar{\tau}x_0:x_t)}[\nabla_\theta\pi_\theta(x_t)\nabla_{ut}Q^{\pi\theta}(x_t,u_t)])$$

$$= \sum_{t=0}^{\infty}(\mathbb{E}_{x_t\sim p\theta_{OLD}(x_t)}[\nabla_\theta\pi_\theta(x_t)\nabla_{ut}Q^{\pi\theta}(x_t,u_t)])$$

Actor 신경망 업데이트



Workflow-DDPG



1. 랜덤 프로세스 초기화
2. 손실함수 이용해 critic 신경망 업데이트

$$L(\emptyset) = \frac{1}{2N}\sum_{i}^{N}(y_i - Q_{\emptyset}(x_i, u_i))^2$$

2-1. gradient 식으로 actor 신경망 업데이트

$$\nabla_\theta L(\theta) = \nabla_\theta \sum_{i} Q_{\emptyset}\left(x_i, \pi_\theta(x_i)\right)$$

2-2. target critic, actor network 업데이트

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta'$$
$$\emptyset' \leftarrow \tau\emptyset + (1-\tau)\emptyset'$$

Actor class - DDPG

state(3X1) → h1(64) →(ReLU) h2(32) →(ReLU) h3(16) →(ReLU) 행동 → Tanh

Critic class - DDPG

state(3X1) → x1(64) / 행동(1X1) →(ReLU) {h2(32) / a1(32)}=concatenate →(ReLU) h3(16) →(ReLU) 행동가치 → 손실함수

$$L(\emptyset) = \frac{1}{2}\sum_{i}(y_i - Q_{\emptyset}(x_i, u_i))^2$$

SAC Algorithm-Soft Actor Critic

- 상황에 따라 차선 선택 가능 DDPG 접근 방식을 형성하여 비 정책 방식으로 확률적 정책 최적화
- 소프트벨만 iteration
- 소프트 행동가치 함수가 수렴할 때까지 정책 개선 번갈아 가면서 한 번씩 업데이트
- **엔트로피 정규화** - 정책의 무작위성 척도인 기대 수익과 엔트로피 간의 균형을 최대화하도록 훈련

목적함수

$$\begin{aligned} J &= \mathbb{E}_{\tau\sim p(\tau)}[\sum_{t=0}^{T}\gamma^t(r(x_t,u_t) - \alpha\log\pi\,(u_t|x_t))] \\ &= \sum_{t=0}^{T}\mathbb{E}_{(x_t,u_t)\sim p(x_t,u_t)}[\gamma^t(r(x_t,u_t) - \alpha\log\pi(u_t|x_t)] \\ &= \sum_{t=0}^{T}\mathbb{E}_{(x_t,u_t)\sim p(x_t,u_t)}[\gamma^t r(x_t,u_t)] + \sum_{t=0}^{T}\gamma^t\mathbb{E}_{x_t\sim p(x_t)}\left[\alpha\mathrm{H}\left(\pi(u_t|x_t)\right)\right] \end{aligned}$$

정책 엔트로피 $H(\pi(u_t|x_t))$

$$-\int_{ut}\log\pi\,(u_t|x_t)\pi(u_t|x_t)du_t$$

→ 최대 엔트로피 목적함수 (maximum entropy)

$$J = \sum_{t=0}^{T} \mathbb{E}_{(x_t,u_t)\sim p(x_t,u_t)}[\gamma^t r(x_t,u_t)] + \sum_{t=0}^{T} \gamma^t \mathbb{E}_{x_t\sim p(x_t)}\left[\alpha \mathrm{H}\left(\pi(u_t|x_t)\right)\right]$$

$$= \sum_{t=0}^{T} \gamma^t \mathbb{E}_{(x_t,u_t)\sim p(x_t,u_t)}[r(x_t,u_t) + \alpha \mathrm{H}\left(\pi(\cdot\,|x_t)\right)]$$

α = 온도 파라미터 (temperature parameter)

α 가 커지면 무작위성 높아짐 / **작아지면 정책확정성경향 올라감** 최적, 준최적 행동도 상황에 따라 선택 가능

Soft Bellman Equation

- Soft state-value function = 상태 가치 함수 + entropy

$$V_{soft}^{\pi}(x_t) = \mathbb{E}_{\tau(u_t:u_r)\sim p(\tau_{ut:ur}|x_t)}[\sum_{k=t}^{T} \gamma^{k-t}(r(x_k,u_k) - \alpha \log \pi(u_k|x_k))]$$

- Soft action-value function = 표준 행동 가치 함수 + entropy

$$Q_{soft}^{\pi}(x_t,u_t) = \int_{\tau_{xt+1:ur}} \sum_{k=t}^{T} \gamma^{k-t}[n_k - \gamma \log \pi(u_{k+1}|x_{k+1})] p(\tau_{x_{t+1}:ur}|x_t,u_t) d\tau_{x_{t+1}:ur}$$

$$= \mathbb{E}_{\tau_{x_{t+1}}:ur\sim p(\tau_{x_{t+1}}:ur|x_t,u_t)}[\sum_{k=t}^{T} \gamma^{k-t}(n_k - \gamma \log \pi\,(u_{k+1}|x_{k+1}))]$$

$(\tau_{x_{t+1}:u_r}|u_r, u_t)$ → 상태변수 x_t 에서 행동 u_t 선택하고 어떤 정책 π 로 생성되는 궤적

- Soft state, Soft action 관계식

$$V_{soft}^{\pi}(x_t) = \mathbb{E}_{u_t\sim\pi(u_t|x_t)}[Q_{soft}^{\pi}(x_t,u_t) - \alpha \log \pi\,(u_t|x_t)]$$

α 가 0이면 소프트가치함수는 표준가치 함수로 환원

▼ 소프트 정책 개선 - 최대 엔트로피 목적함수를 greedy 정책 계산 (현재 시간만 고려하여 최대값 구함)

- 최적 정책

$$\pi(u_t|x_t) = \exp\left(\frac{1}{\alpha} Q^{\pi}_{soft}(x_t, u_t) - 1\right) \propto \exp\left(\frac{1}{\alpha}(Q^{\pi}_{soft}(x_t, u_t)\right)$$

- Q-러닝, DDPG 최대 행동가치 노이즈 추가 인접행동가치 값으로 일정부분 확장하는 단일 모드 분포
- 정책 행동가치가 지수값에 비례하기 때문에 다중모드 분포 → 가능성이 높은 오른 상태

- 목적함수 greedy (argmax) 계산

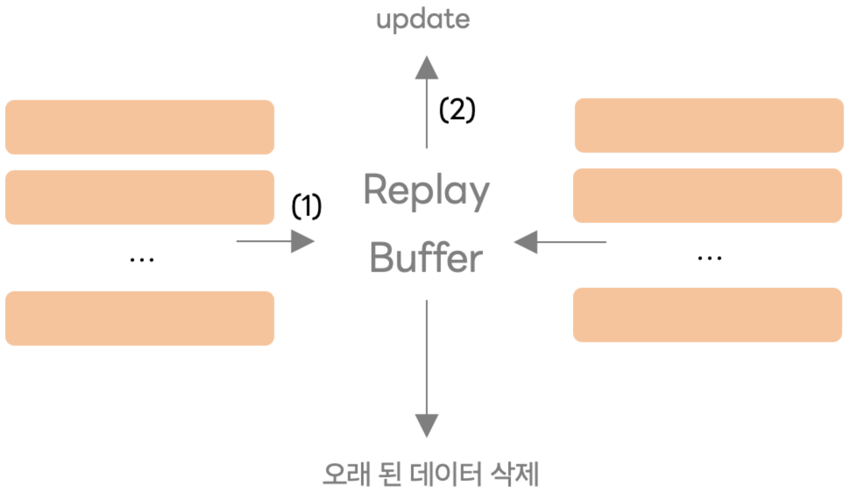
$$\begin{aligned} J_t &= \mathbb{E}_{u_t \sim \pi(u_t|x_t)}[Q^{\pi}_{soft}(x_t, u_t) - \alpha \log \pi(u_t|x_t)] \\ &= -\mathbb{E}_{u_t \sim \pi(u_t|x_t)}[\alpha \left(\log \pi\,(u_t|x_t) - \frac{1}{\alpha} Q^{\pi}_{soft}(x_t, u_t)\right)] \\ &= -\alpha \mathbb{E}_{u_t \sim \pi(u_t|x_t)}[\log \pi(u_t|x_t) - \log \exp\left(\frac{1}{\alpha} Q^{\pi}_{soft}(x_t, u_t)\right) + \log Z(x_t) - \log Z(x_t)] \\ &= -\alpha \mathbb{E}_{u_t \sim \pi(u_t|x_t)}[\log \frac{\pi(u_t|x_t)}{\frac{\exp\left(\frac{1}{\alpha} Q^{\pi}_{soft}(x_t, u_t)\right)}{Z(x_t)}} - \log Z(x_t)] \\ &= -\alpha D_{KL}(\pi(u_t|x_t) || \frac{\exp\left(\frac{1}{\alpha} Q^{\pi}_{soft}(x_t, u_t)\right)}{Z(x_t)}) + \alpha \log Z\,(x_t) \end{aligned}$$

- Soft-action, KL 발산 최소화

$$\begin{aligned} \pi(u_t|x_t) &= argmax \mathbb{E}_{u_t \sim \pi(u_t|x_t)}[Q^{\pi}_{soft}(x_t, u_t) - \alpha \log \pi(u_t|x_t)] \\ &= arg^{min}_{\pi} D_{KL}(\pi(u_t|x_t) || \frac{\exp\left(\frac{1}{\alpha} Q^{\pi}_{soft}(x_t, u_t)\right)}{Z(x_t)}) \end{aligned}$$

$exp(\frac{1}{\alpha} Q^{\pi}_{soft}(x_t, u_t))$ → 가우시안 분포 or 균등분포 or GMM (Gaussian Mixture Model)

Workflow-SAC



N 개 샘플 → x_i, u_i, r_i, x_{i+1}

1. transition sample 리플레이 버퍼 저장 → N 개 무작위 추출

$$q_i = r(x_i, u_i) + \gamma[Q_{\emptyset'}(x_{i+1}, \bar{u}_{i+1}) - \alpha \log \pi_\theta(\bar{u}_{i+1}|x_{i+1})]$$

u_{i+1} → π_θ 로 업데이트

2. Q 신경망 업데이트

2-1. q_i → on-policy → u_i 에서서 샘플링 → $\phi \leftarrow \phi - \alpha_\phi \nabla_\phi L_\phi(\phi)$

$$\begin{aligned} L_Q(\emptyset) &= \frac{1}{2}\sum_i ||Q_\emptyset(x_i, u_i) - q_i||^2 \\ &= \mathbb{E}_{(x_i,u_i)\sim D}[\frac{1}{2} Q_\emptyset(x_i, u_i) - Q_{soft}(x_i, u_i)|x_i, u_i)^2] \end{aligned}$$

$Q_{soft}(x_i, u_i)$ 에서 q_i 로 업데이트

$$\mathbb{E}_{(x_i,u_i,x_{i+1})\sim D}[\frac{1}{2}(Q_\emptyset(x_i, u_i) - q_i(x_i, u_i, x_{i+1})^2]$$

2-2. actor network 업데이트

$$\begin{aligned} L_\pi(\theta) &= \sum_i (\alpha \log \pi_\theta(\bar{u}_i|x_i) - Q_\emptyset(x_i, \bar{u}_i)) \\ &= \mathbb{E}_{x_i\sim D}[\mathbb{E}_{ui\sim\pi\theta(u_i|x_i)}[\alpha \log \pi_\theta(u_i|x_i) - Q_\emptyset(x_i, u_i)] \mid x_i] \end{aligned}$$

재파라미터화 트릭 (reparameterizaion trick)

$$u_i^j = f_\theta(x_i, \eta_j) \sim \mathcal{N}(\mu_\theta(x_i), \sigma_\theta^2(x_i))$$
$$= \mu_\theta(x_i) + \sigma_\theta(x_i)\eta_j, \eta_j \sim N(O.I)$$

η_j 노이즈 벡터

2-3. target Q 신경망 업데이트

$\phi' \leftarrow \tau\phi + (1-\phi)\phi'$

▼ Q 신경망이 2개인 경우

1. transition sample 리플레이 버퍼에 저장 → 2개 중 작은 값 추출

$$Q_{\emptyset'}(x_{i+1}, \bar{u}_{i+1}) = \min\,[Q_{\emptyset_1'}(x_{i+1}, \bar{u}_{i+1}), Q_{\emptyset_2'}(x_{i+1}, \bar{u}_{i+1})]$$

2. Q 신경망 업데이트

$$L_\emptyset(\emptyset_{1,2}) = \frac{1}{2}\sum_i ||Q_{\emptyset 1,2}(x_i, u_i) - q_i||^2$$
$$L_\pi(\theta) = \sum_i (\alpha \log \pi_\theta(\bar{u}_i|x_i) - Q_\emptyset(x_i, \bar{u}_i))$$

$$\phi_1' \leftarrow \tau\phi_1 + (1-\tau)\phi_1'$$
$$\phi_2' \leftarrow \tau\phi_2 + (1-\tau)\phi_2'$$

Actor class - SAC

state(3X1) → h1(64) →(ReLU) h2(32) →(ReLU) h3(16) →(ReLU) 평균/표준편차 → Tanh/softplus

Critic class - SAC

state(3X1) → x1(64) / 행동(1X1) →(ReLU) {h2(32) / a1(32)}=concatenate →(ReLU) h3(16) →(ReLU) 행동가치

최적 제어-Optimal Control

상태공간 차분 방정식(State-space difference equation)

$$J_0 = \sum_{t=0}^{T} c(x_t, u_t) = C_\gamma(x_\gamma) + \sum_{t=0}^{\gamma-1} C_t(x_t, u_t)$$

비용함수 $C(x_t, u_t) = -r(x_t, u_t)$ 보상함수

dynamic programming - 벨만의 최적성 원리 이용한 최적화 방법

backward-in-time 최종 상태에서서 거슬러 올라감

$$J_t^0 = {}^{min}_{u_t}(c(x_t, u_t) + J_{t+1}^0$$

LQR(Linear Quadratic Regulator) - 최소 제어 크기로 상태변수 모두 0으로 수렴

$$x_{t+1} = F_t \begin{bmatrix} x_t \\ u_t \end{bmatrix} + f_{ct}$$
$$J_0 = \sum_{t=0}^{T} (\frac{1}{2} [\begin{matrix} x_t \\ u_t \end{matrix}]^2 C_t \begin{bmatrix} x_t \\ u_t \end{bmatrix} + [\begin{matrix} x_t \\ u_t \end{matrix}]^2 C_t)$$

- 역방향 패스 (칼만게인 계산)

$$Q_t = C_1 + F_t^T \cdot V_{t+1} \cdot F_t$$
$$q_t = c_t + F_t^T \cdot V_{t+1} \cdot f_{ct} + F_t^T \cdot v_{t+1}$$

$$Q_t = \begin{bmatrix} Q_{xxt} & Q_{xut} \\ Q_{uxt} & Q_{uut} \end{bmatrix}$$
$$q_t = [\begin{matrix} Q_{xt} \\ Q_{ut} \end{matrix}]$$

$$K_t = -Q_{uut}^{-1} Q_{uxt}$$
$$k_t = -Q_{uut}^{-1} Q_{ut}$$

$$\begin{aligned} V_t &= Q_{xxt} + Q_{xut} \cdot K_t + K_t^T \cdot Q_{uxt} + K_t^T \cdot Q_{uut} \cdot K_t \\ &= Q_{xxt} - Q_{xut} \cdot Q_{uut}^{-1} \cdot Q_{uxt} \\ v_t &= Q_{xt} + K_t^T \cdot Q_{ut} + Q_{xut} \cdot k_t + k_t^T \cdot Q_{uut} \cdot k_t \\ &= Q_{xt} - Q_{xut} \cdot Q_{uut}^{-1} \cdot Q_{ut} \end{aligned}$$

- 순방향 패스 (최적 궤적 계산)

$$u_t = k_t \cdot x_t$$
$$x_{t+1} = F_t \begin{bmatrix} x_t \\ u_t \end{bmatrix} + f_{ct}$$

확률적 LQR(stochastic LQR)

$$V_{T-1} = Q_{xxt} + Q_{xuT-1} \cdot K_{T-1} + K_{T-1}^T \cdot Q_{uxT-1} + K_{T-1}^T \cdot Q_{uuT-1} \cdot K_{T-1}$$
$$v_{T-1} = Q_{xT-1} + K_{T-1}^T Q_{uT-1} + Q_{xuT-1} \cdot K_{T-1} + K_{T-1}^T \cdot Q_{uuT-1} \cdot k_{t-1}$$

$$\frac{1}{2} tr(V_t \sum\nolimits_{T-1})$$

상수항 추가 → 프로세스 노이즈 공분산은 초기 상태변수 공분산과 무관

가우시안 LQR

확정적 정책 → $\pi(u_t|x_t) = N(x_t, \mu_t, S_t)$ 이 가우시안 확률밀도 함수 갖는 확률적 정책 확장

(확정적 정책에서 S_t 는 공분산)

$\pi(u_t|x_t) = NCK_t * x_t + k_t, Q_{uut}^{-1})$ (Q_{uut}^{-1} 는 공분산)

반복적 LQR(iterative LQR, iLQR)

비선형 시스템에 LQR 적용

기존 궤적 = 명목 궤적, 실제 궤적 x_t 를 명목 궤적 $\bar{x}_t$ 와 perturbation $\triangle\ x_t$ 합으로 표현

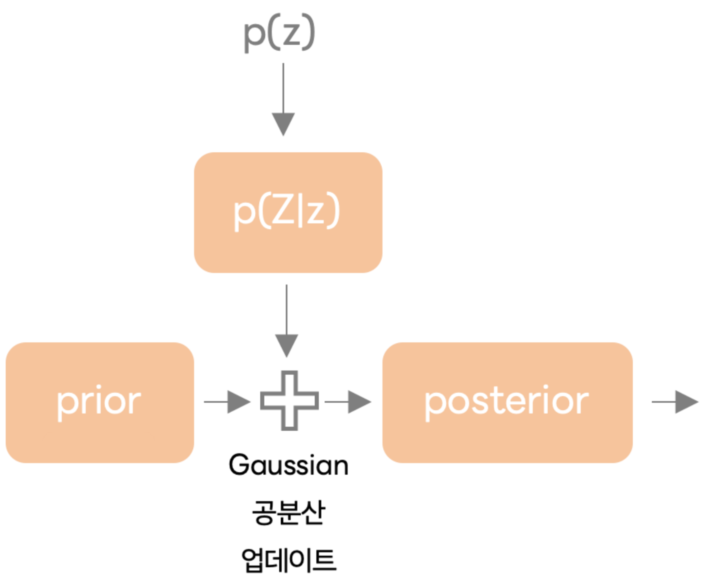
$$\begin{aligned} y_t = h(x_t) &= h(\bar{x}_t + \Delta x_t) \\ &= h(\bar{x}_t) + \frac{dh}{dx}|_{x_t=\bar{x}_t}\, \Delta x_t + H.O.T. \\ &\approx h(\bar{x}_t) + \frac{dh}{dx}|_{x_t=\bar{x}_t}\, \Delta x_t \end{aligned}$$

$$\frac{dh}{dx}|_{x_t=\bar{x}_t} = \nabla x_t, h(\bar{x}_t)$$

자코비안 행렬

$$\begin{aligned} x_{t+1} &= f(x_t, u_t) + n_t, t = 0, \cdots, T \\ x_{t+1} &\approx f_{xt}x_t + f_{ut}u_t + f_{ct} + n_t \end{aligned}$$

선형화



- 데이터 수집
- 로컬 모델 피팅(조건부 가우시안, GMM 사전분포, conjugate prior-normal inverse-Wishart)
- 로컬 제어 법칙 업데이트
    - Gaussian 공분산 업데이트 → 사전정보로 간주하고 공분산 $\mu_z^{MAP}, P_z z^{MAP}$ 업데이트

$$\mu_z^{MAP} = \frac{m\mu_0 + n_0\mu_z}{m + n_0}$$
$$P_{zz}^{MAP} = \frac{\phi + NP_{zz} + \frac{N_m}{N+m}(\mu_z - \mu_0)(\mu_z - \mu_0)^T}{N + n_0}$$

- 최적화(dual gradient descent)

$$L(p,n) = \sum_{t=0}^{T} \begin{cases} \mathbb{E}_{(x_t,u_t)\sim p(x_t,u_t)}[C(x_t,u_t) - \eta \log \bar{p}(u_t|x_t)] \\ -\eta \mathbb{E}_{x_t \sim p(x_t)} H\left(p(u_t|x_t)\right) - \eta\epsilon \end{cases}$$

- 대체 비용 함수

$$\tilde{c}(x_t, u_t) = \frac{1}{2}\begin{bmatrix} x_t \\ u_t \end{bmatrix}^T D_z \begin{bmatrix} x_t \\ u_t \end{bmatrix} + \begin{bmatrix} x_t \\ u_t \end{bmatrix}^T d_t + const$$

$$D_z = \frac{1}{h} C_t + \begin{bmatrix} \bar{K}_t^T \bar{S}_t^{-1} \bar{k}_t & -\bar{k}_t^T \bar{S}_t^{-1} \\ -\bar{S}_t^{-1} \bar{k}_t & \bar{S}_t^{-1} \end{bmatrix}$$
$$d_t = \frac{1}{n} C_t + [\bar{K}_t^T \bar{S}_t^{-1} \bar{k}_t \quad -\bar{S}_t^{-1} \bar{k}_t]$$

- 듀얼 변수

$$\eta_{max} \leftarrow \eta \qquad \rightarrow \ \eta \leftarrow \max\left(0.1\eta_{max}, \sqrt{\eta_{min}\eta_{max}}\right)$$
$$\eta_{min} \leftarrow \eta \qquad \rightarrow \ \eta \leftarrow \max\left(10\eta_{min}, \sqrt{\eta_{min}\eta_{max}}\right)$$

$$\epsilon' = \epsilon \frac{l_{k-1}^{k-1} - l_{k-1}^{k-1}}{2(l_k^k - l_{k-1}^k)}$$

- 업데이트 된 궤적 확률 밀도 사이 (KL 발산에 대한 한계값)