Policy Optimzation

1. 강화학습과 정책 최적화란?

강화학습은 에이전트(Agent)가 환경(Environment)과 상호작용하면서 보상(Reward)을 최대화하도록 학습하는 머신러닝 분야다.

여기서 에이전트는 **정책(Policy)**라는 걸 따라 행동을 선택하는데, 정책은 쉽게 말해 "상황(상태, State)이 주어졌을 때 어떤 행동(Action)을 할지 결정하는 규칙"이다.

정책 최적화(Policy Optimization)는 이 정책을 더 좋은 방향으로 개선하는 과정이야. 즉, 에이전트가 더 많은 보상을 받을 수 있도록 정책을 업데이트하는 거지. PPO는 이 과정을 효율적이고 안정적으로 수행하기 위해 고안된 방법 중 하나이다.

---

2. PPO의 핵심 아이디어

PPO는 이전 정책 최적화 방법들(예: TRPO 같은 것들)이 너무 복잡하거나 불안정하다는 문제를 해결하려고 나왔어. PPO의 목표는:

- **안정성**: 정책을 너무 급격하게 바꾸면 학습이 망가질 수 있으니, 업데이트를 "안전하게" 제한하자.
- **단순함**: 복잡한 수학 계산을 줄여서 구현하기 쉽게 만들자.
- **효율성**: 적은 계산으로도 좋은 성능을 내자.

PPO는 이 세 가지를 잘 조화시킨 알고리즘이야.

---

3. PPO가 동작하는 방식 (쉽게 비유로 설명)

PPO를 이해하기 쉽게, 우리가 친구와 함께 미로 탈출 게임을 한다고 생각해보자. 미로에서 출구(보상)를 찾으려면 어디로 갈지 계속 선택해야 해. 여기서 PPO는 이런 식으로 작동해:

(1) 현재 정책으로 미로 탐색하기

- 네가 지금까지 배운 "미로에서 왼쪽으로 가면 출구에 가까워진다" 같은 규칙(정책)을 따라 움직여봐.
- 이 과정에서 어떤 선택이 잘 됐는지(보상이 높았는지) 데이터를 모아.

(2) 새 정책으로 조금씩 업데이트

- 모은 데이터를 보고 "음, 왼쪽도 괜찮았지만 오른쪽으로 가는 게 더 빠를 수도 있겠다"라고 생각해.
- 근데 여기서 중요한 건, 갑자기 "이제 무조건 오른쪽만 간다!"라고 완전히 바꾸지 않는다는 거야. 왜냐? 너무 큰 변화는 오히려 길을 잃게 할 수 있거든.
- PPO는 **"기존 정책에서 너무 멀리 벗어나지 않도록"** 제한을 걸어. 마치 "기존 길에서 한두 발짝만 옆으로 가보자" 같은 느낌이야.

(3) 보상 확인하고 반복

- 새로 바꾼 정책으로 다시 미로를 탐색하면서 더 나은 보상을 받는지 확인해.
- 이 과정을 계속 반복하면서 점점 출구에 가까워지도록 정책을 개선하는 거야.

---

4. PPO의 기술적 핵심 (조금 더 깊이 들어가기)

이제 좀 더 구체적으로 PPO가 어떻게 이 "제한된 업데이트"를 하는지 설명해볼게. 수학적인 부분은 최대한 쉽게 풀어서 설명할게.

(1) Clipped Objective Function (클리핑된 목표 함수)

- PPO는 정책을 업데이트할 때 "너무 큰 변화는 안 돼!"라는 규칙을 적용해. 이를 위해 **클리핑(Clipping)**이라는 방법을 써.
- 쉽게 말해, 새 정책이 기존 정책과 얼마나 다른지를 계산하고, 그 차이가 너무 크면 "그 정도까지만 바꿔!"라고 잘라버리는 거야.
- 예를 들어, 네가 미로에서 "왼쪽으로 10번 가던 걸 갑자기 오른쪽으로 100번 가자"고 바꾸면 위험하니까, "최대 20번까지만 바꿀게" 같은 제한을 두는 식이야.

(2) Advantage Function (이득 함수)

- PPO는 어떤 행동이 얼마나 좋은지 판단하기 위해 "이득(Advantage)"라는 개념을 써. 이건 "이 행동을 했을 때 평균보다 얼마나 더 좋은 보상을 받았나?"를 계산한 거야.
- 이득이 크면 그 행동을 더 자주 하도록 정책을 조정하고, 작으면 덜 하도록 조정해.

(3) Actor-Critic 구조

- PPO는 보통 "Actor"와 "Critic"이라는 두 개의 네트워크를 같이 써.
  - **Actor**: 정책을 담당해. "어떤 행동을 할까?"를 결정해.
  - **Critic**: 보상을 평가해. "이 행동이 얼마나 좋은 결과를 낼까?"를 예측해.
- 이 둘이 협력하면서 Actor는 더 나은 행동을 배우고, Critic은 더 정확한 평가를 하게 돼.

---

5. PPO의 장점과 한계

장점:

- **안정적**: 정책 업데이트가 제한돼 있어서 학습이 망가질 확률이 적어.
- **구현 쉬움**: 복잡한 수학 계산이 덜 필요해서 코드로 쓰기 편해.
- **범용성**: 게임, 로봇 제어 등 다양한 강화학습 문제에 잘 맞아.

한계:

- **성능 제한**: 너무 안전하게 업데이트하다 보니, 가끔 최적의 정책을 찾는 데 시간이 더 걸릴 수 있어.
- **하이퍼파라미터 의존**: 클리핑 범위 같은 설정을 잘 조정해야 성능이 좋아져.

---

6. 실생활 비유로 마무리

PPO를 요리 실력 키우기로 비유해보면:

- 처음엔 "이 레시피대로만 요리하자"라는 정책을 따라.
- 요리해보고 "소금을 조금 더 넣으면 맛있겠다"라는 피드백을 받아.
- 근데 갑자기 "소금을 10배 넣자!"는 안 하고, "조금씩만 더 넣어보자"로 안전하게 실험해.
- 계속 맛보고 조정하면서 점점 완벽한 요리를 만드는 법을 배우는 거야.

▼ 개선 방안

1. 적응형 클리핑(Adaptive Clipping):

PPO는 클리핑 범위를 고정적으로 설정하는데, 이걸 상황에 따라 유연하게 바꾸는 방법이 있어. 예를 들어, 학습 초반엔 클리핑을 느슨하게 해서 더 큰 변화를 허용하고, 나중에 안정되면 조여서 세밀하게 조정하는 식이야. 이렇게 하면 느리게 가는 걸 줄이고 더 빠르게 최적의 정책에 가까워질 수 있어.

2. 자동 튜닝(Auto-Tuning):

하이퍼파라미터를 수동으로 조정하는 대신, 학습 중에 성능을 보고 자동으로 조정하는 알고리즘을 추가할 수 있어. 예를 들어, 클리핑 범위를 보상의 변화나 정책 안정성에 따라 실시간으로 바꾸는 거야.