

DQN vs HRL

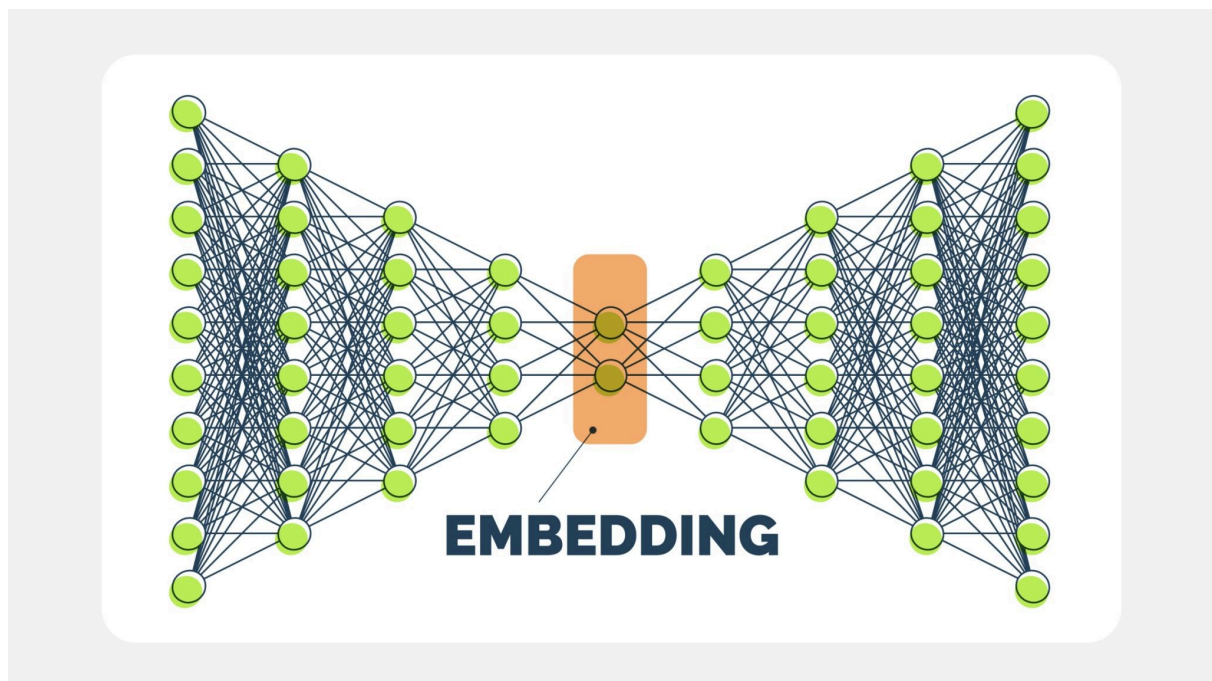
두 코드는 모두 **"똑똑한 시장 조성자(Market Maker)"**를 만들려는 목적은 같지만, 그 **"두뇌(Manager)"**를 구성하는 방식에서 결정적인 기술적 차이가 있음.

..., **DQN**은 '직관을 가진 전문가'라면, **HRL(Q-Table)**은 '꼼꼼한 매뉴얼을 가진 관리자'

1. 두뇌의 형태 (Brain Architecture)

가장 큰 차이는 **"어떻게 기억하고 판단하느냐"**.

- **DQN (Neural Network):** * 이 녀석은 **인공신경망(뉴런)**을 써. 뇌세포처럼 얽힌 수식들이 들어온 정보를 처리.
 - 데이터를 입력하면 "... , 이 상황은 예전에 본거랑 80% 정도 비슷하니까 이렇게 하면 되겠군" 하고 **직관적인 추론**.



- **HRL Q-Table (Dictionary):**
 - ... 그냥 아주 커다란 **엑셀 표(표)**.
 - "거래량이 '상'이고 변동성이 '하'일 때는 액션 1번을 해라"라는 식으로 모든 상황을 칸(Cell)에 적어두고 학습.

2. 세상을 바라보는 방식 (State Representation)

- **DQN: "세상은 부드러운 흐름이야" (Continuous)**
 - 환율 변동성이 0.0001인지 0.00015인지 소수점 끝까지 그대로 받아들임.
 - 상황이 조금만 바뀌어도 그에 맞춰 미세하게 판단을 바꿀 수 있는 **유연함**.
- **HRL Q-Table: "세상은 딱딱 나뉘어 있어" (Discrete/Binning)**
 - 표에 적어야 하니 상황을 단순화해야 함.
 - 코드에 보면 `np.digitize` 라는 게 있지? 변동성이 얼마든 간에 그냥 "낮음(0)", "중간(1)", "높음(2)" 세 칸 중 하나로 구겨 넣음.
 - 세밀한 차이를 무시하는 대신, 판단 속도가 광속만큼 **단순하고 빠름**.

3. 학습하는 방법 (Learning Logic)

- **DQN: "복습과 시뮬레이션" (Experience Replay)**
 - `deque(maxlen=5000)` 이라는 기억 저장소가 있어. 과거의 경험을 모아두었다가 나중에 무작위로 꺼내서 ****반복 학습(복습)****.
 - `target_model` 이라는 '가상의 정답지'를 하나 더 만들어서 본인의 판단과 비교하며 조금씩 수정해 나가는 아주 복잡하고 고등한 방식.
- **HRL Q-Table: "그때그때 적어넣기" (Direct Update)**
 - 방금 한 행동이 결과가 좋았으면, 표의 그 칸에 있는 점수를 즉시 수정.
 - "아, 아까 그 상황(칸)에선 이게 정답이었네!" 하고 바로 적고 끝. 복습 같은 건 안함.

4. 한눈에 비교하기

특징	DQN 코드	HRL (Q-Table) 코드
두뇌 엔진	PyTorch (인공지능)	Python Dictionary (표)
판단 근거	수천 개의 수식 연산 (직관)	상황별 점수 매칭 (매뉴얼)
상황 파악	매우 정밀함 (소수점 단위)	단순함 (상/중/하 등급제)
컴퓨터 사양	GPU나 고성능 CPU 필요	계산기 수준에서도 돌아감
장점	학습 데이터가 많으면 신의 영역에 도달	왜 그런 결정을 했는지 이해하기 쉬움
단점	'왜' 그렇게 행동했는지 알기 어려움	처음 보는 상황이 오면 바보가 됨

5. 결론

- **DQN**은 시장이 아주 복잡하고, 단순한 규칙만으로는 설명이 안 되는 **미묘한 타이밍**을 잡아야 할 때 유리. (단, 학습시키는 게 어렵고 시간이 걸림)
- **HRL Q-Table**은 시장 상황을 몇 가지로 딱딱 정의할 수 있고, **안정적이고 투명하게** 운영하고 싶을 때 적합. (금융권에서는 검증이 쉬운 이 방식을 선호)

쉽게 한 문장 요약: > "DQN은 인공지능 알파고가 바둑을 두는 느낌이고, Q-Table은 바둑 고수의 정석 책을 보고 그대로 따라 두는 느낌"

다음 단계

'보상 함수'나 '액션 맵' 이 어떻게 다른지

- DQN 코드의 보상 체계
- 이 두 전략을 합친 하이브리드 모델