Deep Learning

2023-12-20

Reinforcement learning

Smartr

Agenda

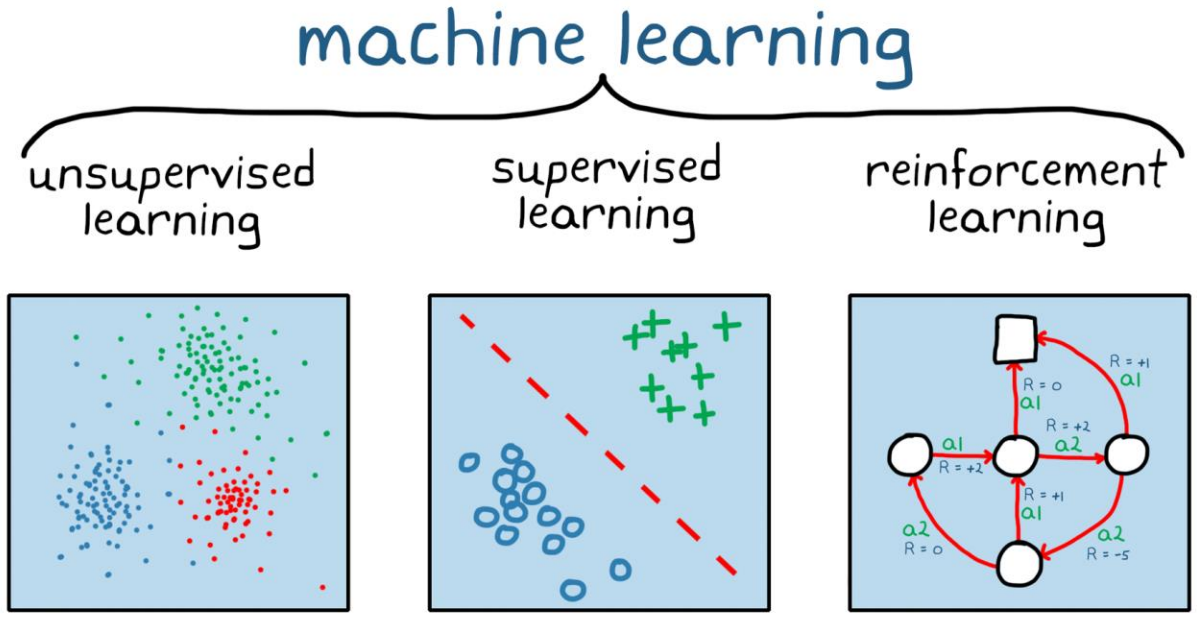
- Intro till reinforcement learning
- Agents
- Q-learning
- Deep Q networks (DQN)

Smartr

RL

Reinforcement learing skiljer sig från det vi tittat på innan (supervised learning och unsupervised learning).

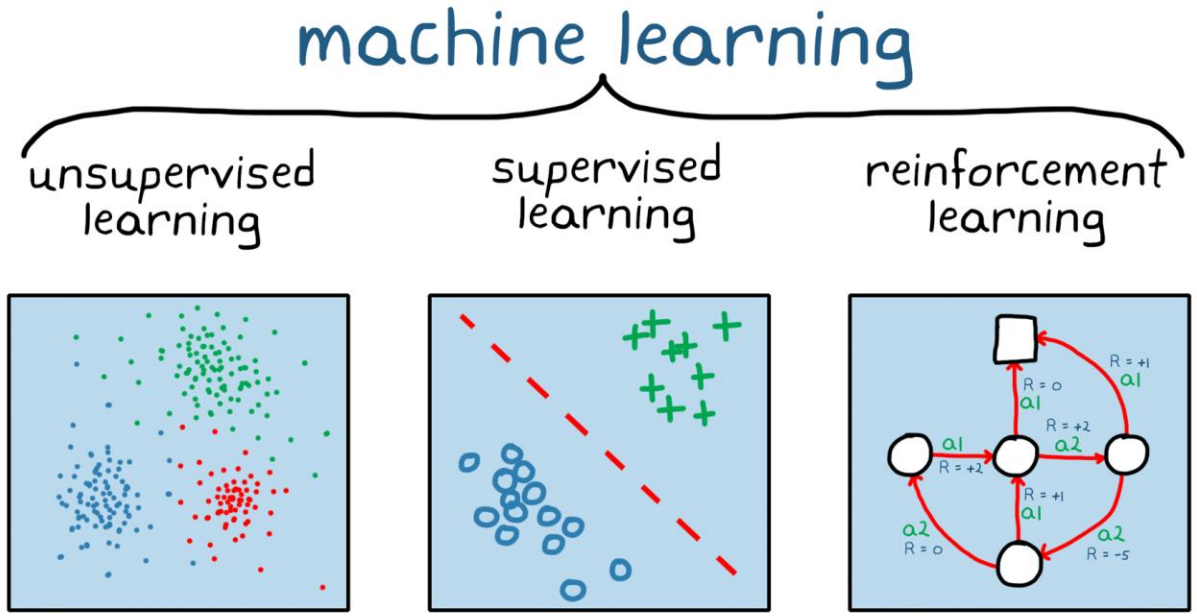
I RL har vi agenter som lär sig ta beslut i en miljö för att nå ett mål.



RL

Modellen lär sig konsekvenserna av sina misstag.

- Pluspoäng om du stannar vid rött.
- Minuspoäng om du kör över cyklisten.



RL

När vi jobbar med RL så har vi följande

- En/flera agenter
- En miljö
- State
- Möjliga val att göra
- Belöning

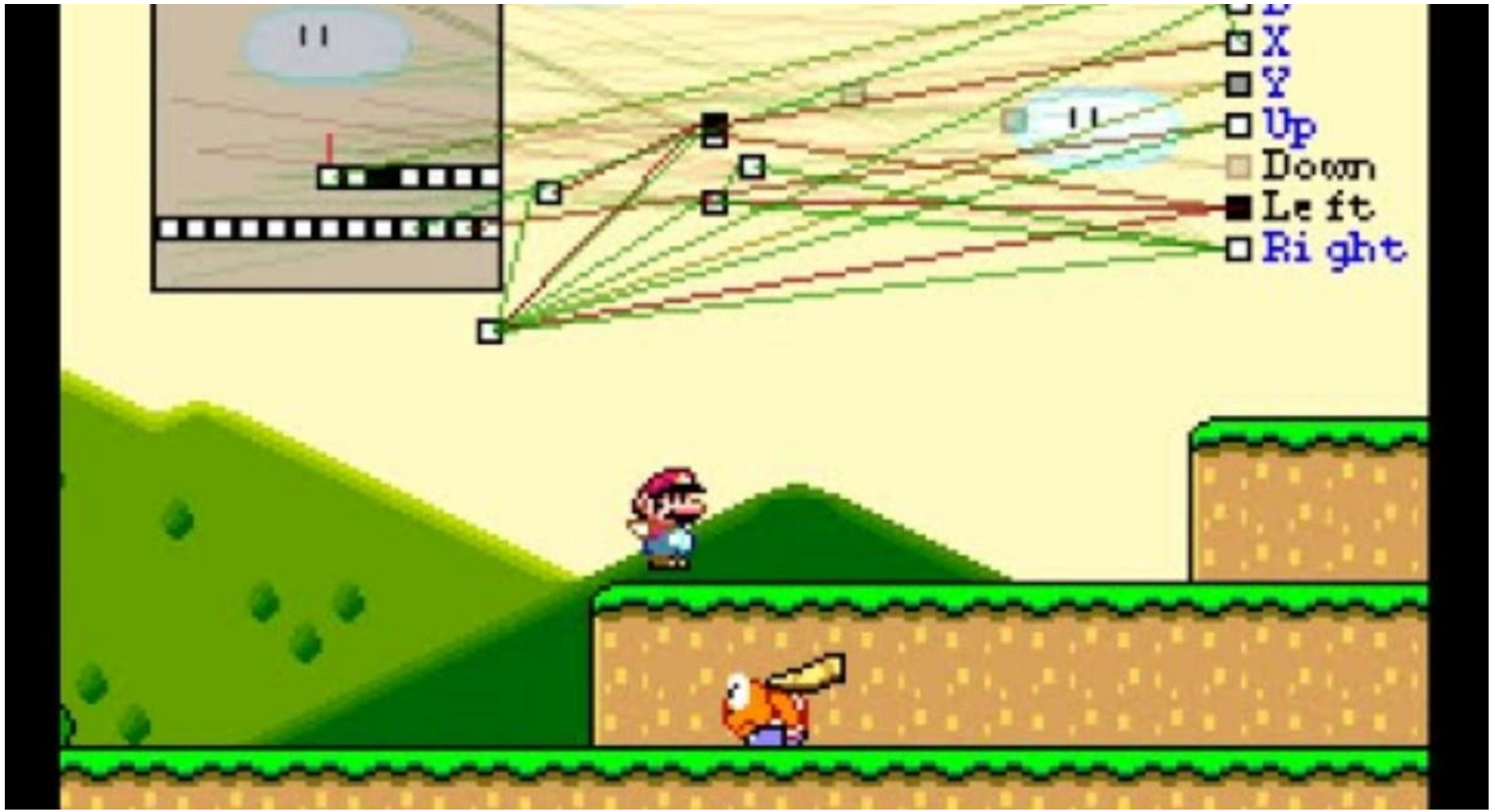
Smartr

RL

- Agent
- Environment
- Observation space – Vad vi ser
- State – All information
- Action - höger, vänster, hoppas, osv.
- Reward

Smartr

RL



Smartr

RL

RL

- Spel
- Självkörande bilar
- Handla med aktier
- Robotar
  - Monteringslinor
- Värme/Kylsystem

RL

Vi skapar en simulerad miljö som våran agent/agenter kan tränas i och skapa en policy. En policy är reglerna som agenten följer för att ta beslut.

RL

Det är väldigt svårt att skapa miljöer som helt stämmer överens med verkligheten.

I spel däremot har vi redan skapat miljön (och möjliga actions) och kan därmed snabbt komma igång med modellträning.

RL

I slutet måste agenterna testas i en kontrollerad verklig miljö. Även det är svårt att genomföra.

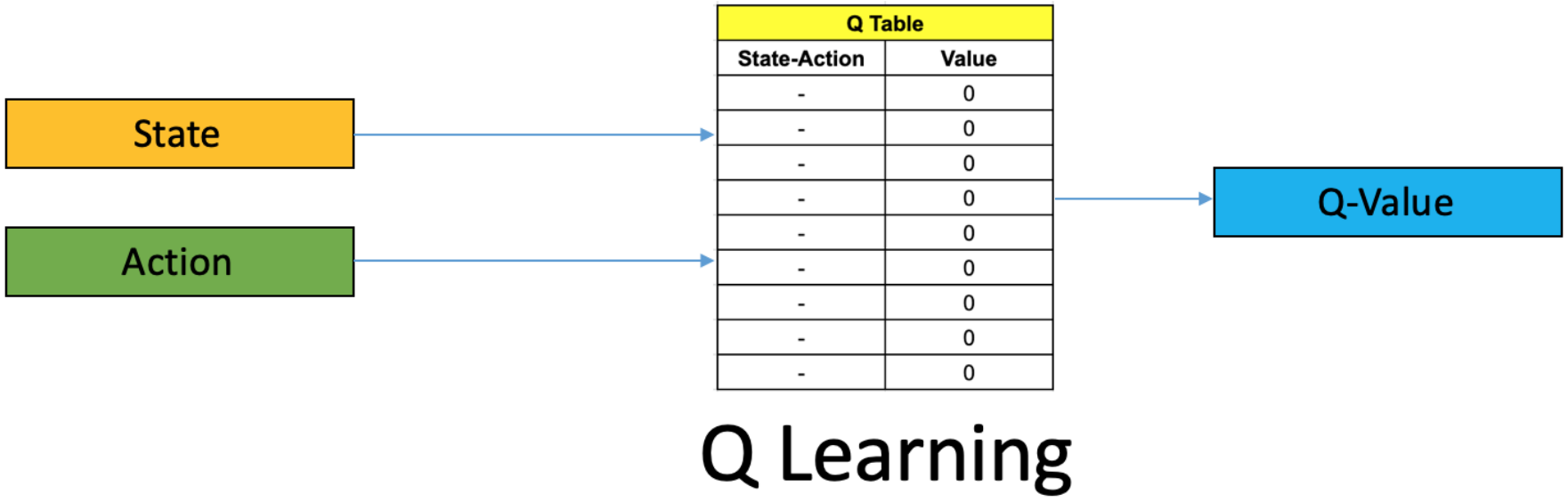
Q-Learning

Q-Learning

Q-Learning går ut på att modellen ska lära sig estimera reward för olika actions, givet ett state.

Utifrån det väljer vi att göra den action som ger högst reward.

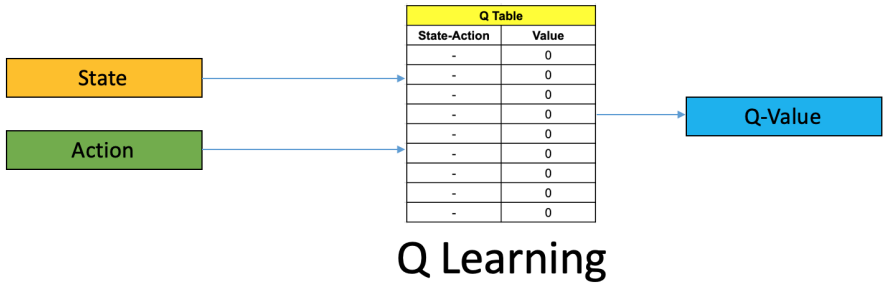
Q-Learning



Q-Learning

$$Q(state,action) \leftarrow (1-\alpha) \cdot Q(state,action) + \alpha \cdot (reward + \gamma \cdot \max(Q(next\_state, next\_action)))$$

- Q(state,action) - Q-värde för givet state och action.
- α - Learning rate
- γ - Bestämmer hur viktigt framtida rewards är, hög gamma ger högre vikt på framtida rewards
- max(Q(next_state, next_action)) - prediktion av maximal reward i nästa state givet alla möjliga actions.



Q Learning

Till slut konvegerar alla Q(state,action) till något värde och då har vi nått en optimal policy.

Q-Learning

- Initialisering
  - Initialt initialiseras Q-tabellen ofta med nollor (eller slumpmässiga värden), vilket indikerar att agenten inte har någon tidigare kunskap.
- Välja en Åtgärd - Utforskning vs. Utnyttjande
  - Agenten väljer åtgärder genom att använda en policy som härleds från Q-tabellen. En vanlig strategi är epsilon-girig-policy, där agenten väljer åtgärden med högst Q-värde (utnyttjande) med sannolikheten 1-ε och en slumpmässig åtgärd (utforskning) med sannolikheten ε. Ofta minskar vi ε över tid och modellen utforskar mindre och mindre.
- Uppdatera Q-Tabellen
  - Efter att ha tagit en åtgärd och observerat belöningen och nästa tillstånd, uppdaterar agenten Q-värdet för tillstånds-åtgärdsparet med hjälp av uppdateringsregeln.
- Upprepa Processen
  - Processen att välja åtgärder och uppdatera Q-tabellen upprepas, vilket tillåter agenten att utforska miljön och förbättra sin policy.
- Konvergens
  - Över tid och med tillräcklig utforskning, konvergerar Q-värdena till stabila värden, vilket representerar den optimala policyn.

Q-Learning

Black Jack exempel.

I black jack vill vi få att våra kort summeras till 21 (eller så nära som möjligt).

Vi kan ta så många kort vi vill, så länge vi är under 21.

Givaren måste alltid ta ett kort så länge hen har under 17.

Klädda kort är 10, ess kan vara 1 eller 11 och resterande är samma som på kortet.

Givet vad vi har för kort och vad givaren har för öppet kort vill vi hitta en policy som bestämmer när vi ska ta ett till kort och när vi ska stanna (hit/stand).

Q-Learning

Hard total	Dealer hole card									
	2	3	4	5	6	7	8	9	10	A
4	H	H	H	H	H	H	H	H	H	H
5	H	H	H	H	H	H	H	H	H	H
6	H	H	H	H	H	H	H	H	H	H
7	H	H	H	H	H	H	H	H	H	H
8	H	H	H	H	H	H	H	H	H	H
9	H	H	H	H	H	H	H	H	H	H
10	H	H	H	H	H	H	H	H	H	H
11	H	H	H	H	H	H	H	H	H	H
12	H	H	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	H	H
16	S	S	S	S	S	H	H	H	H	H
17	S	S	S	S	S	S	S	S	S	S
18	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
20	S	S	S	S	S	S	S	S	S	S
21	S	S	S	S	S	S	S	S	S	S

Q-Learning

Q-Learning är ganska begränsat. Ökar vi states eller actions får vi ett väldigt stort Q table.

Våran Q table bygger också på att (state, action) paren finns. Kommer det in et par som vi inte sett tidigare kan vi inte göra något.

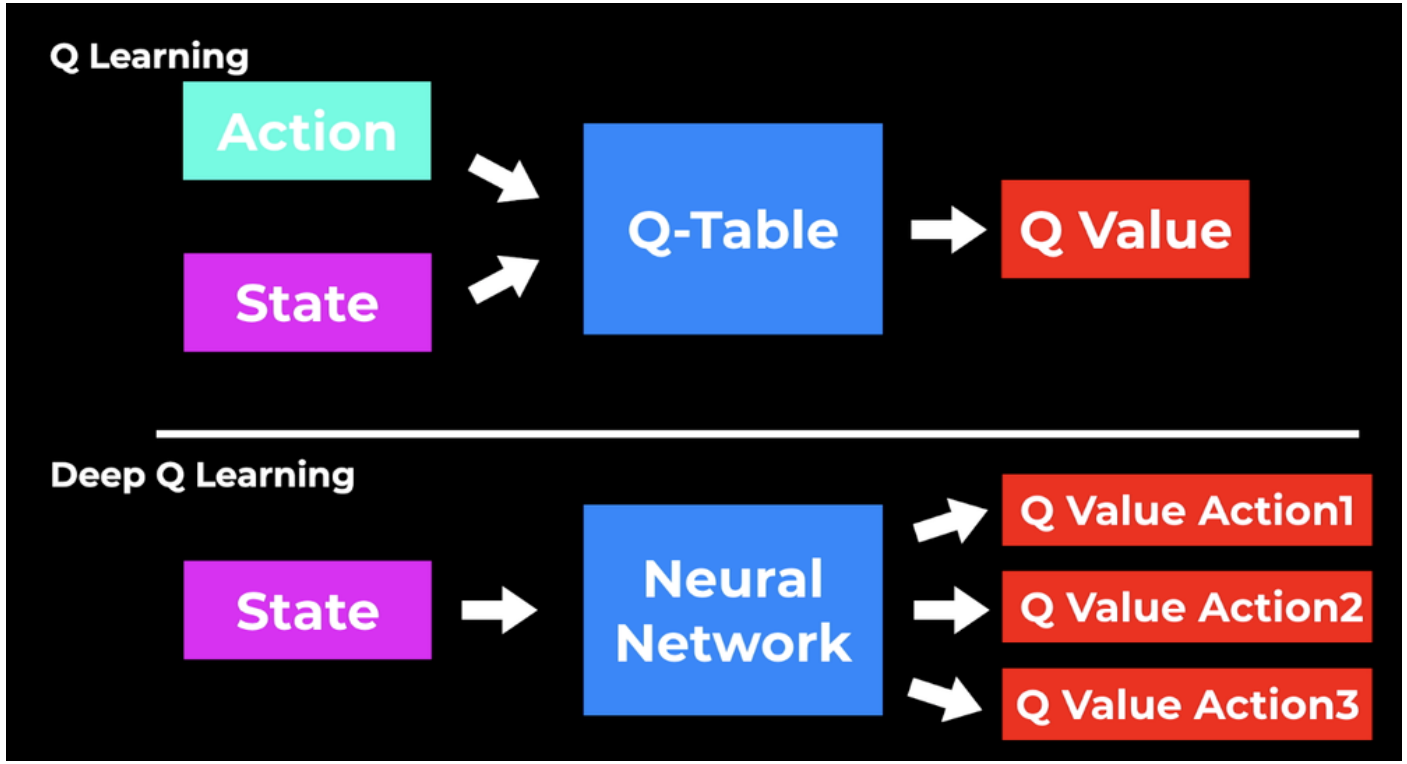
Det finns ingen koppling mellan olika (state, action) par. Även om det kan finnas stora likheter mellan dem.

Hard total	Dealer hole card									
	2	3	4	5	6	7	8	9	10	A
4	H	H	H	H	H	H	H	H	H	H
5	H	H	H	H	H	H	H	H	H	H
6	H	H	H	H	H	H	H	H	H	H
7	H	H	H	H	H	H	H	H	H	H
8	H	H	H	H	H	H	H	H	H	H
9	H	H	H	H	H	H	H	H	H	H
10	H	H	H	H	H	H	H	H	H	H
11	H	H	H	H	H	H	H	H	H	H
12	H	H	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	H	H
16	S	S	S	S	S	H	H	H	H	H
17	S	S	S	S	S	S	S	S	S	S
18	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
20	S	S	S	S	S	S	S	S	S	S
21	S	S	S	S	S	S	S	S	S	S

Deep Q-Learning

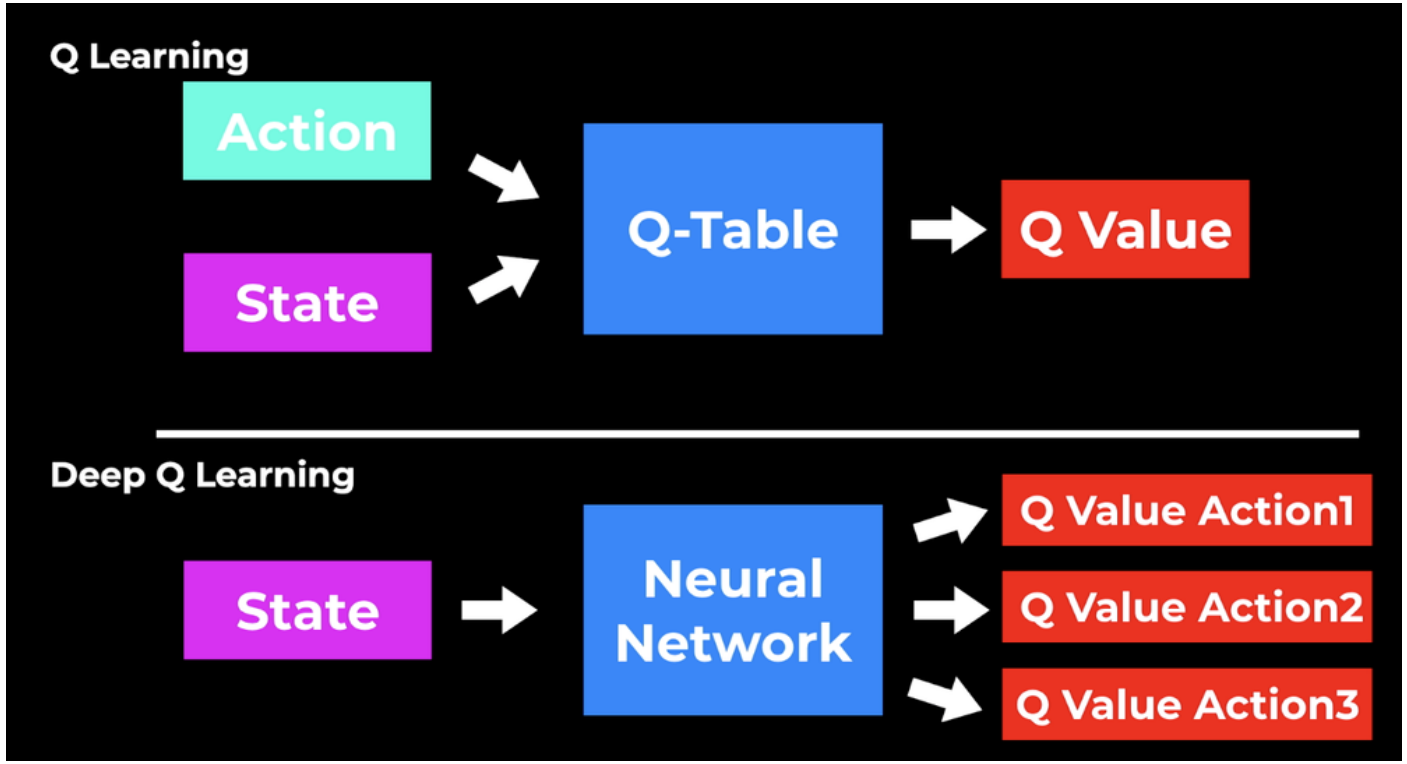
DQN

Deep Q-learnign (DQN) använder sig av Q-learning principen. Istället för ett Q table använder vi ett neuralt nätverk som lär sig att estimera Q value för alla actions, givet ett state.



DQN

- Vi kan göra beräkningar på okända states
- Vi kan hitta likheter mellan states.
- Djupa nätverk är bra på att hantera komplexa problem, vi kan alltså ha väldigt stora states.

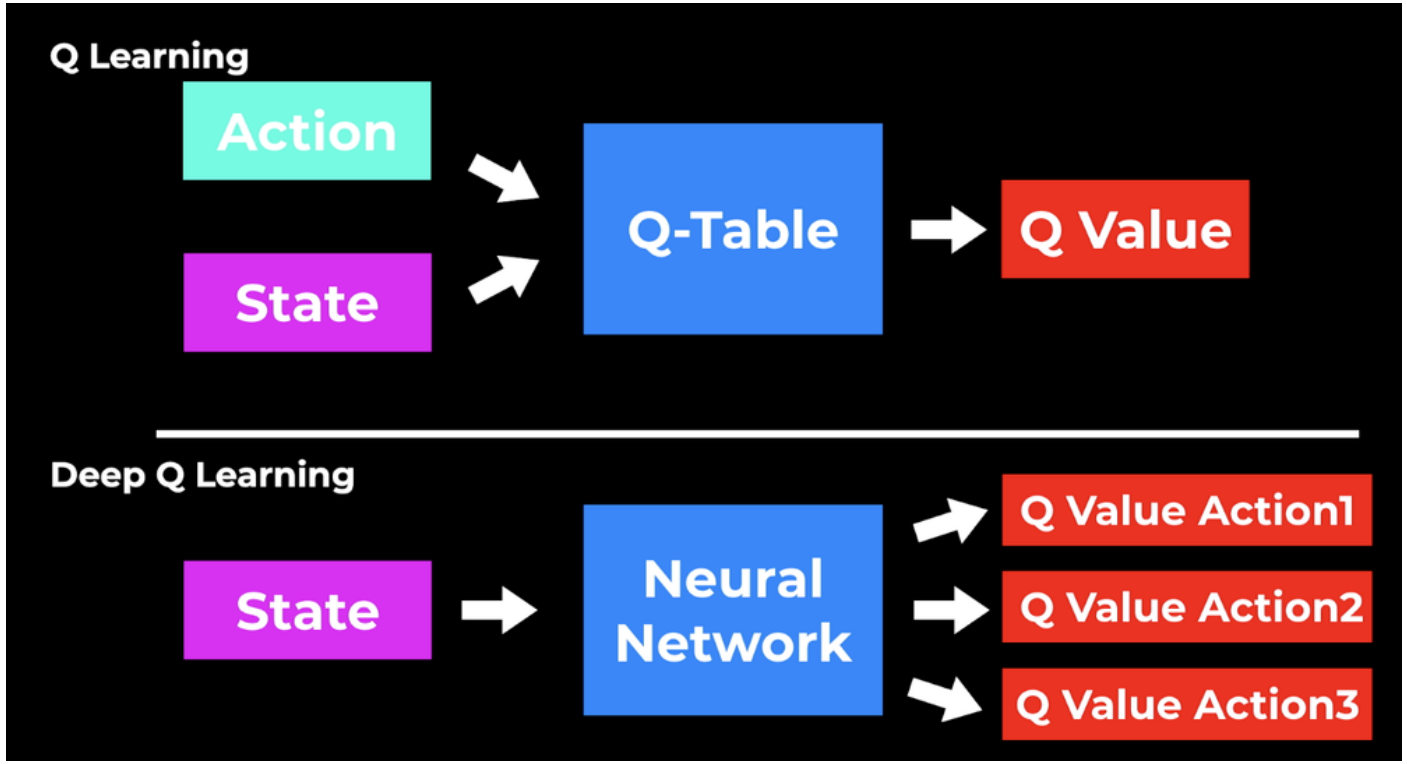


DQN

Då vi inte har någon information om vad vi kan få för reward från varje state introducerar DQN Replay

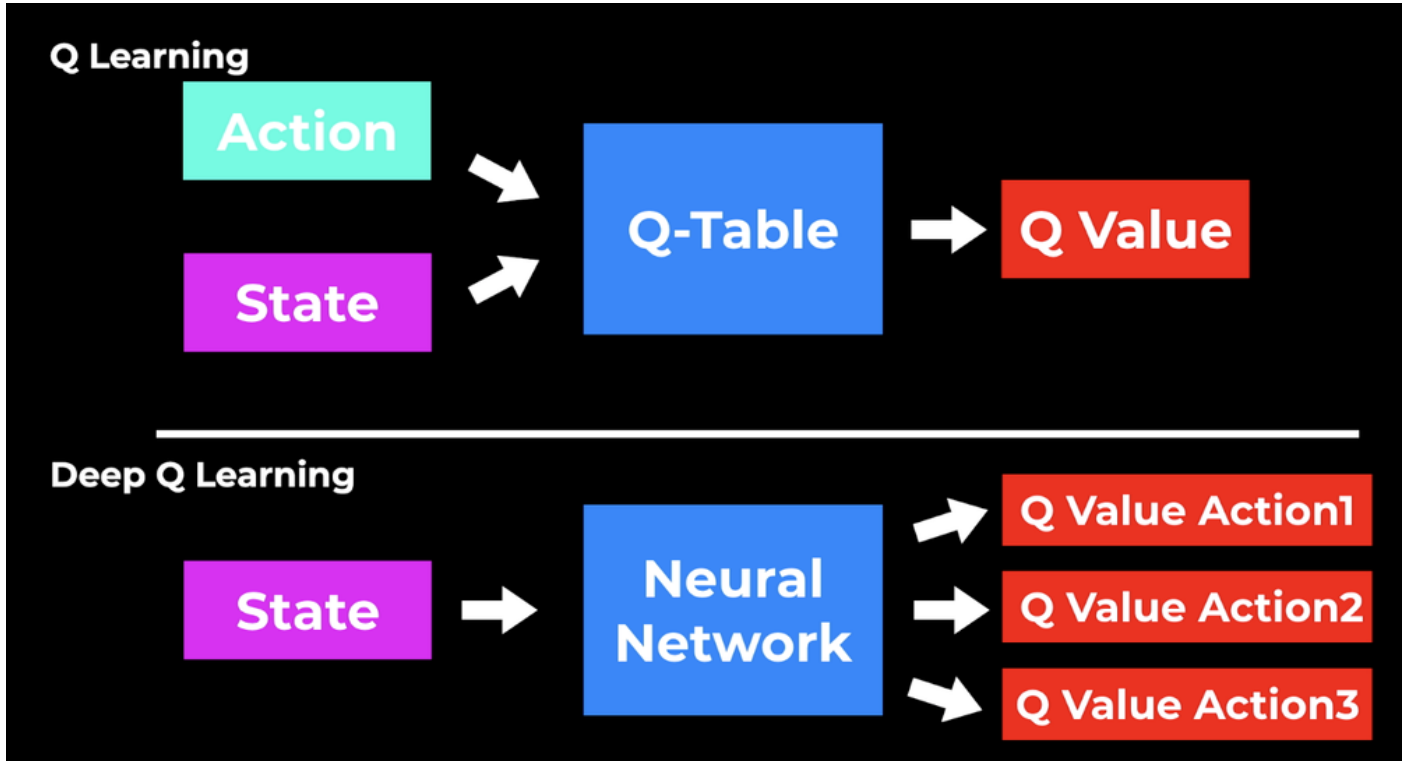
I våran träning så genomför våran agent en action (som är helt slumpmässig i början) och vi sparar Experience (state, action, reward, next state, done) i en buffert (Replay minnet).

Vi tar mini batches från denna buffert och tränar nätverket.



DQN

Vi väljer hur stort minne vi ska ha. När min blir fullt så plockar vi bort det äldsta vi har i minnet när vi läger till något nytt. På så viss får vi hela tiden nya



Q-Learning

- Välj antal episoder som agenten ska tränas (lite som epoker, men inte riktigt)
- Välj hur många tidssteg agenten kan genomföra under en episod.
- För varje episod:
  - Nollställ din miljö (börja från början)
  - För varje tidssteg i din episod:
    - **Välj en action:** Antingen gör modellen en beräkning eller så slumpas en action (här använder vi epsilon för att ibland göra slumpade actions).
    - **Genomför vald action**
    - **Observera utkomsten**: Reward och next state.
    - **Spara informationen (Experience)**: (state, action, reward, next_state, done) i Replay minnet.
    - **Lär från Replay minnet**:
      - Slumpa en minibatch med experiences från Replay minnet.
      - För varje experience, beräkna target Q-value och maximum Q-value för nästa state.
      - Uppdatera agenten.

Black Jack

Actions: Hit/Stand

Observation space:
(player value, dealer value, if player got an ace)

Reward:

- Win - +1
- Loss - -1
- Draw - 0