AI 2025
SUMMER SCHOOL
ai.uni-jena.de

Overview of Reinforcement learning

How machines learn by trial, error, and rewards

Uday Kaipa - Technische Universität Ilmenau

Introduction and Comparison of Supervised vs RL

Reinforcement learning (RL) is a branch of machine learning where an agent learns what to do (i.e., which actions to take) by interacting with the environment to maximize a numerical reward signal. Rewards are often sparse. Hence, agents explore randomly until they find a path that produces a reward, and then continue to explore for other paths that may yield a higher reward. Over time, it balances **exploration** (trying new things) with **exploitation** (using what it knows works).

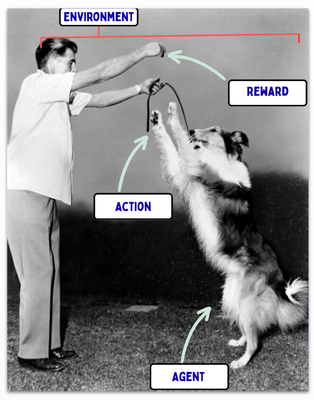

Fig 1. Reinforced behavior in dogs

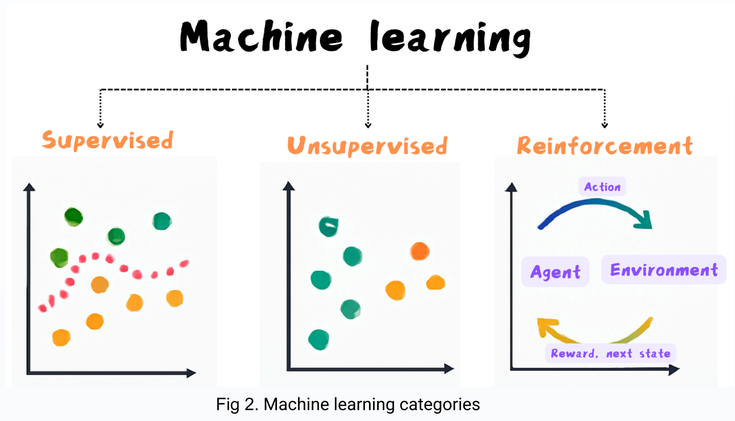

Fig 2. Machine learning categories

Supervised Learning	Reinforcement Learning
Data with labels (input-output pairs)	Interaction with environment, no labeled data
Learns mapping from data to labels	Learns policy for decision making over time
Passive learning (fixed dataset)	Active learning (agent explores environment)
Immediate feedback (loss per example)	Sparse, delayed rewards affect future states
No notion of sequential decisions	Decisions affect future states and rewards
Continuous optimization via gradient	Balances exploration vs exploitation (tunable hyperparameter)

Understanding Reinforcement Learning

1. The Loop *(core idea)*

Agent ↔ Environment

- Agent observes the **state** of the environment
- Agent chooses an **action**
- Environment gives a **reward** and a new state
- Agent updates its **policy** to do better in the future

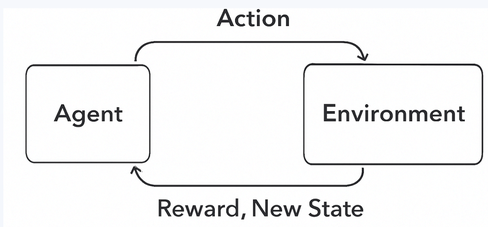

Fig 3. RL training loop [1]

2. Key Concepts

- **State (S) :** What's going on now (e.g., chessboard position, robot's location)
- **Action (A) :** What the agent can do next (e.g., move piece, step forward)
- **Reward (R) :** Feedback signal (good/bad) from the environment
- **Policy (π) :** The agent's decision-making strategy

3. The Goal

Maximize **total future reward** — not just immediate reward.

4. Why It Works

The agent **learns by trial and error** using rewards as guidance and stores the learning in form of Policy (π), improving over time without needing correct answers beforehand.

5. Algorithms

- The agent uses a Q-table (state-action) to decide which action is best in each state.
- It looks up the values for all possible actions and chooses the one with the highest Q-value.
- As the agent explores and learns, it updates the table, gradually improving its choices for each situation.

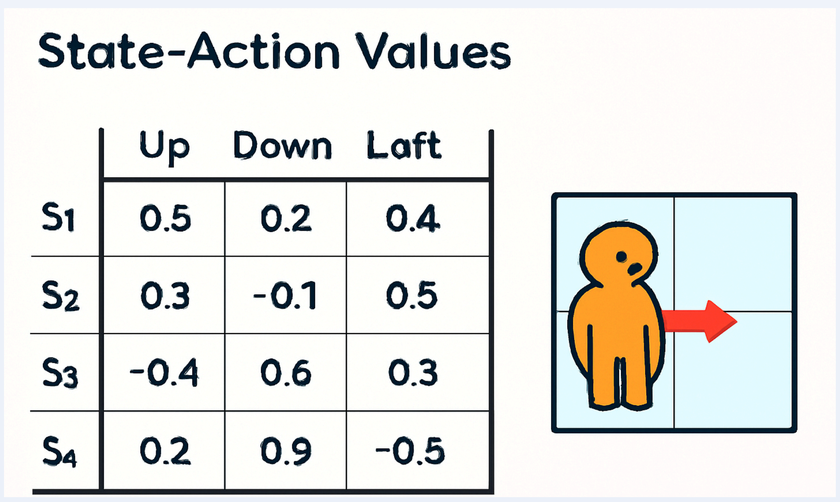

Fig 4. Q-Value updation [1]

RL Taxonomy

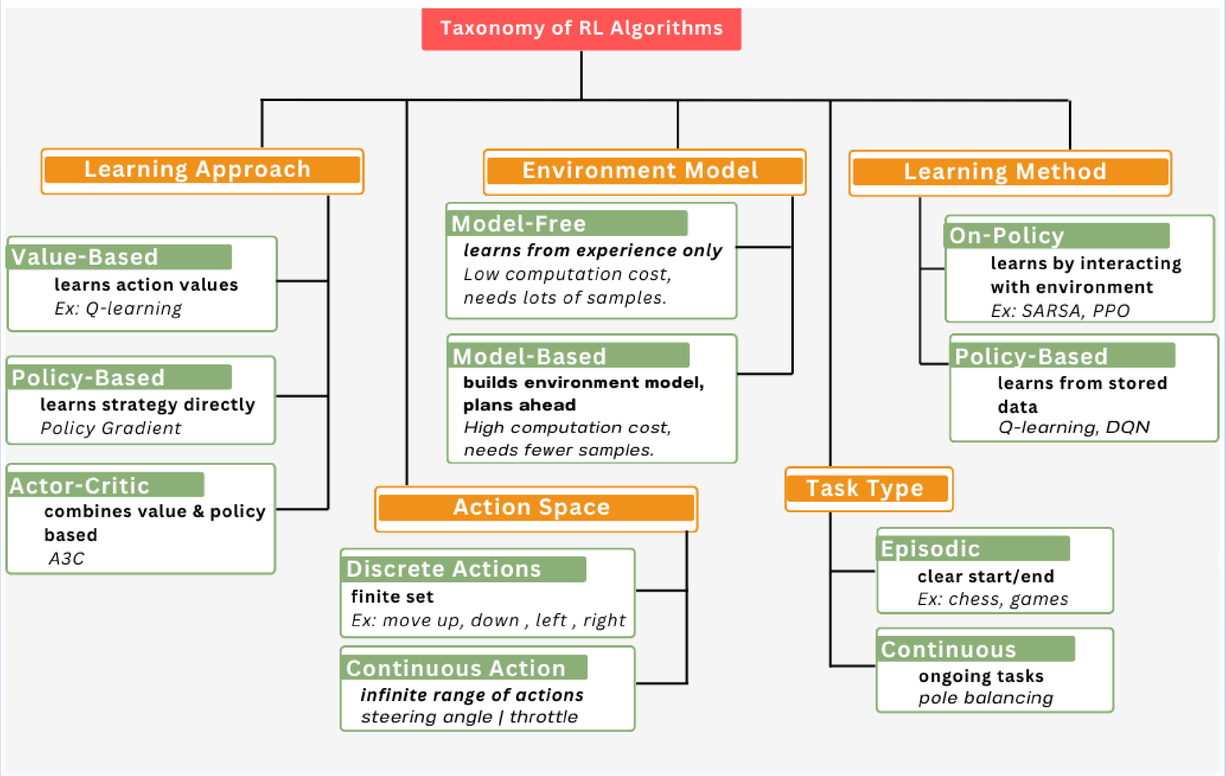

Fig 5. RL taxonomy

Applications

Success Stories of RL

1. **Game-playing**

   AlphaGo / AlphaZero → beating world champions in Go, Shogi & Chess[3]

   Atari Game Playing Agents → Deep Q-Networks (DQN) learned to play classic Atari games (like Breakout, Pong) directly from pixels, achieving superhuman performance.

2. **Robotics Control**

   Humanoid & Robo dogs

   MuJoCo humanoids

   OpenAI's Rubik's Cube hand [4]

3. **Autonomous systems**

   Self-driving cars → Tesla, Waymo.
   Drones → delivery path planning, aerial inspection.

4. **Recommendation systems**

   Netflix, **YouTube**, and **Spotify** use RL to personalize content

   ChatGPT → RLHF (*Reinforcement Learning from Human Feedback*)

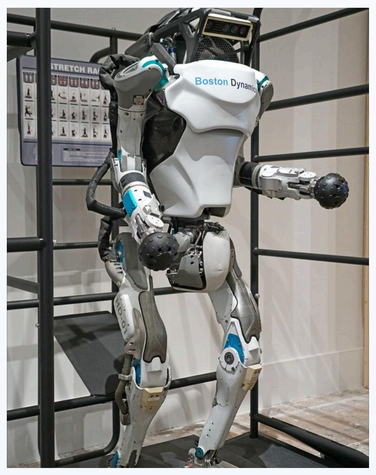

Fig 7. Humanoid [2]

Challenges and future

Why it's still hard and exciting:

- Requires lots of data and computing power
- Hard to design good reward signals
- Balancing exploration vs. exploitation is tricky

Where it's heading:

- Combining RL with other AI methods
- More sample-efficient and safer learning
- Multi-agent co-operation, feudal networks.

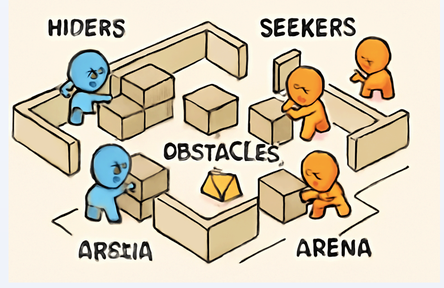

Fig 8. Multi-agent co-operation; Hide and seek - Ai-generated [1]

References

1. Image generated by Perplexity AI. (2025). "RL Training Loop, State-action value table, Hide and seek MARL." Retrieved August 15, 2025, from https://www.perplexity.ai
2. "L'exposition MATCH. Design & Sport (Musée du Luxembourg, Paris)" by dalbera is licensed under CC BY 2.0.
3. Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., … Silver, D. (2020). Mastering Atari, Go, chess and shogi by planning with a learned model. arXiv:1911.08265. Retrieved from https://arxiv.org/abs/1911.08265
4. OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., … Zhang, L. (2019). Solving Rubik's Cube with a Robot Hand. arXiv:1910.07113. Retrieved from arXiv:1910.07113

Contact and license

Contact :
Uday Kaipa, Uday.kaipa@icloud.com, Technische Universität Ilmenau