Personalized Education in the Era of AI

RL Recap & Latent Space Modeling & Pedagogical Control

Ruixiang Wu & Costas Courcoubetis

October 16, 2025

School of Data Science

Table of contents

Recap: Reinforcement Learning

Introduction to Markov Decision Processes (MDPs)

An MDP is formally defined by a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$:

- $\mathcal{S}$: **State Space**: A finite set of states representing the different situations an agent can be in. (e.g., current position on a board game, specific image on a screen.)
- $\mathcal{A}$: **Action Space**: A finite set of actions available to the agent in each state. (e.g., move left, jump, attack.)
- $P(s'|s, a)$: **Transition Probability Function**: The probability of transitioning to state s' from state s after taking action a. (Defines the dynamics of the environment.)
- $R(s, a, s')$: **Reward Function**: The immediate scalar reward received after transitioning from state s to s' by taking action a.
- γ: **Discount Factor**: A value between 0 and 1 that determines the present value of future rewards. A higher γ means future rewards are considered more important.

A policy π is a mapping from $\mathcal{S}$ to $\mathcal{A}$. An optimal policy π^* minimize the discounted sum over (possibly) infinite horizon: $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})\right]$

Classical Control: A Foundation for RL

Linear System Dynamics: The system's state x_k at step k evolves according to a known linear model:

$$x_{k+1} = Ax_k + Bu_k$$

- x_k: State vector
- u_k: Control input (action)
- A, B: System matrices

Optimal Control (LQR): We find the optimal control law by minimizing a cost function J:

$$J = \sum_{k=0}^{\infty} (x_k^T Q x_k + u_k^T R u_k)$$

This penalizes state deviation (Q) and control effort (R). The solution is a linear feedback policy:

$$u_k = -Kx_k$$

Model Based Reinforcement Learning i

Key point of "Model Based": Know $P(s', s, a)$, and $R(s, a)$. (i.e. Assume that the MDP model is known.)

A Model Based RL Algorithm: Policy Iteration: Policy Iteration is a classic model-based algorithm in reinforcement learning used to find an optimal policy π^*.

The Core Idea: It finds the optimal policy by iteratively repeating two simple steps:

1. **Policy Evaluation**: Determine how effective the current policy is.
2. **Policy Improvement**: Use that knowledge to improve the policy.

This process is repeated until the policy no longer improves, at which point it has converged to the optimal policy.

Model Based Reinforcement Learning ii

Step 1: Policy Evaluation

Goal: Given a fixed, deterministic policy π, calculate the state-value function $V^{\pi}(s)$ for all states s.

This step answers the question: "What is the expected long-term reward if we follow policy π from each state?"

To find V^{π}, we solve the **Bellman Equation** for the policy π. This equation expresses the value of a state as the expected immediate reward plus the discounted value of the next state.

$$V^{\pi}(s) = \sum_{s'} P(s'|s, \pi(s)) \left(R(s, \pi(s), s') + \gamma V^{\pi}(s')\right)$$

For a finite number of states, this is a system of linear equations that can be solved for V^{π}.

Model Based Reinforcement Learning iii

Step 2: Policy Improvement

Goal: Given the value function V^{π} for a policy π, find a better policy π'.

We improve the policy by acting greedily with respect to the value function V^{π}.

For each state s, we update the policy by choosing the action that maximizes the expected value, looking one step ahead:

$$\pi'(s) \leftarrow \arg\max_{a \in \mathcal{A}} \left\{ \sum_{s'} P(s'|s,a) \left(R(s,a,s') + \gamma V^{\pi}(s') \right) \right\}$$

This new greedy policy π' is guaranteed to be an improvement over (or equal to) the old policy π. That is, for all states s:

$$V^{\pi'}(s) \geq V^{\pi}(s)$$

Model Based Reinforcement Learning iv

Policy Iteration Algorithm

1. **Initialization**: Start with an arbitrary policy π_0.
2. **Iteration**: For $k = 0, 1, 2, \ldots$
   2.1 **Policy Evaluation**: Compute the value function for the current policy π_k.
   $$V_k \leftarrow V^{\pi_k}$$
   2.2 **Policy Improvement**: Create a new greedy policy π_{k+1} using V_k.
   $$\pi_{k+1}(s) \leftarrow \arg\max_a \sum_{s'} P(s'|s,a) \left(R(s,a,s') + \gamma V_k(s')\right)$$
   2.3 **Check for Convergence**: If $\pi_{k+1}(s) = \pi_k(s)$ for all states s, then stop.
3. **Termination**: Return the final policy π_{k+1} and value function V_k.

Model Based Reinforcement Learning v

Another Model Based RL Algorithm: Value Iteration.:
The Core Idea: Instead of iterating on the policy, Value Iteration iteratively improves the **value function** until it converges to the optimal value function, V^*. The optimal policy is then extracted just once at the very end.

The algorithm finds the optimal value function by repeatedly applying the **Bellman Optimality Equation** as an update rule:

The Update Rule
Starting with an initial value function V_0 (e.g., all zeros), each iteration $k+1$ updates the value of every state s using the values from the previous iteration V_k:

$$V_{k+1}(s) \leftarrow \max_{a \in \mathcal{A}} \sum_{s'} P(s'|s,a) \left(R(s,a,s') + \gamma V_k(s')\right)$$

Model Free Reinforcement Learning

Key point of "Model Free": MDP is unknown, rely on sampling strategies to learn.

A representative example of Model Free RL is Q Learning, which directly learns the **optimal action-value function**, denoted q_*, regardless of the policy being followed for exploration.

The update rule is defined as:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[\underbrace{R_{t+1} + \gamma \max_a Q(S_{t+1}, a)}_{\text{TD Target: Best possible next value}} - \underbrace{Q(S_t, A_t)}_{\text{Old value}} \right]$$

Difference between on-policy and off-policy?

Deep Reinforcement Learning i

Key point of "Deep": Using a Deep Neural Network to approximate the value function or quality function.

Policy Gradient Methods

The Core Idea

Policy Gradient methods optimize a **parameterized policy** $\pi_\theta(s, a)$ directly. The parameters θ could be the weights of a neural network.

- The goal is to adjust θ to maximize the total expected reward, $J(\theta)$.
- We do this by performing **gradient ascent** on the policy parameters.
- This approach is often more effective in high-dimensional or continuous action spaces.

The Policy Gradient Update The update is guided by the **Policy Gradient Theorem**, which provides an expression for the gradient of the expected reward.

Policy Gradient Theorem
The gradient of the objective function $J(\theta)$ is given by the expectation:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[Q^{\pi_\theta}(s, a) \nabla_\theta \log \pi_\theta(s, a) \right]$$

Want to see the proof? $J(\theta) = \sum_{s \in S} P(s) \sum_{a \in A} Q(s, a) \pi_\theta(s, a)$

The Update Rule
The policy parameters are then updated via gradient ascent:

$$\theta_{\text{new}} \leftarrow \theta_{\text{old}} + \alpha \nabla_\theta J(\theta)$$

where α is the learning rate.

Deep Reinforcement Learning iii

The Challenge for Classic RL: Representation

Classic RL methods often use tables to store values for each state (e.g., Q-tables). This is impossible for problems with huge state spaces, like video games, where a single screen has more states than atoms in the universe!

The Solution: Deep RL

Combine the power of **deep learning** for function approximation with the decision-making framework of **reinforcement learning**.

We use deep neural networks to approximate the key RL functions:

- **Policy**: $\pi(s, a) \approx \pi(s, a, \theta)$
- **Value Function**: $V(s) \approx V(s, \theta)$
- **Q-Function**: $Q(s, a) \approx Q(s, a, \theta)$

Deep Reinforcement Learning iv

Deep Q-Networks (DQN): DQN uses a **deep neural network** to approximate the Q-function: $Q(s, a) \approx Q(s, a, \theta)$.

How It Works
Instead of updating a table, DQN trains the network weights θ by minimizing a loss function based on the TD error from Q-learning:

- The network takes the state s (e.g., screen pixels) as input and outputs Q-values for all possible actions.
- The loss function to minimize is the squared difference between the TD target and the network's prediction.

DQN Loss Function

$$L(\theta) = \mathbb{E}\left[\left(\underbrace{R + \gamma \max_{a'} Q(S', a', \theta_{\text{target}})}_{\text{TD Target}} - \underbrace{Q(S, a, \theta)}_{\text{Prediction}}\right)^2\right]$$

Reinforcement Learning Overview

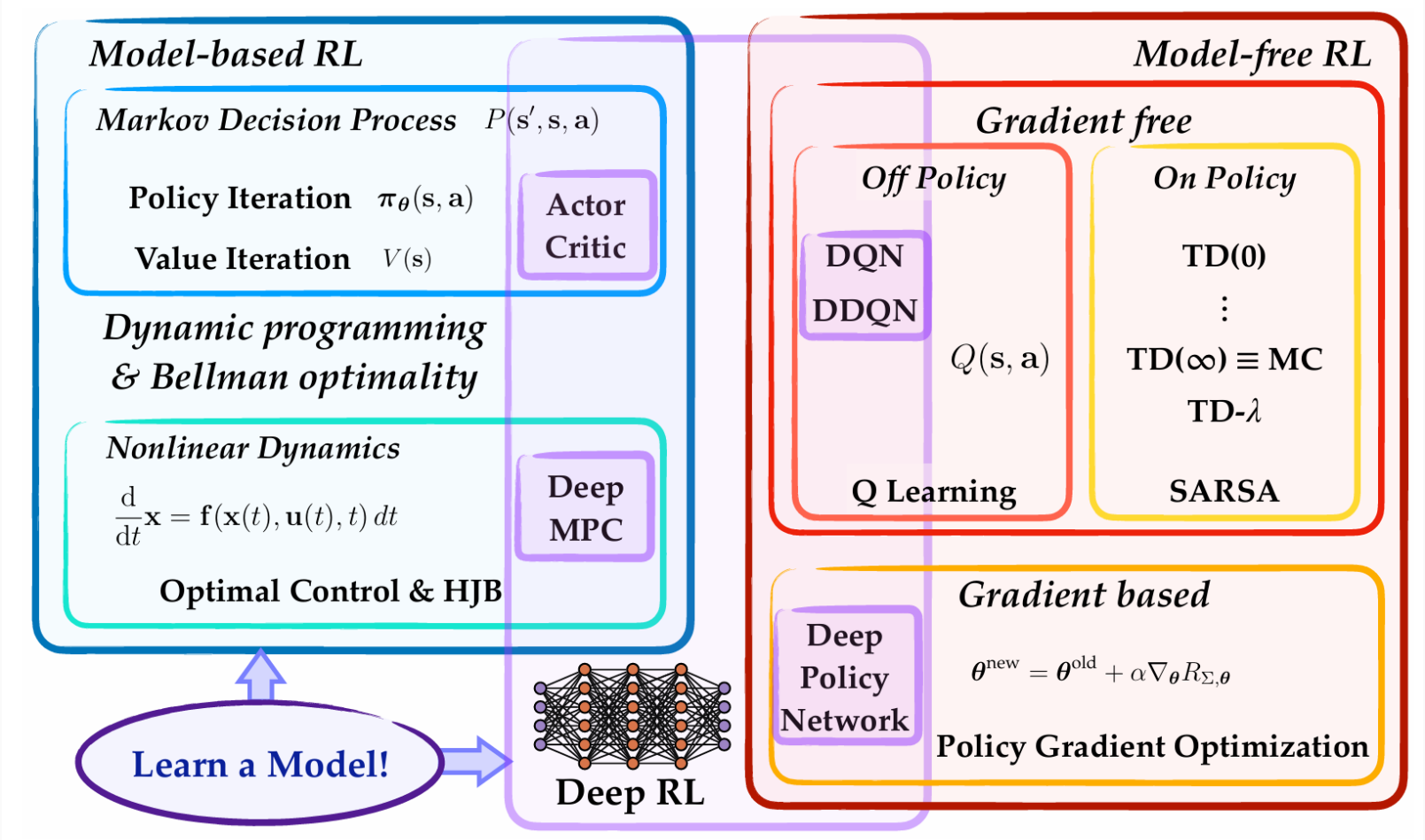

Figure 1: Overview of the World of RL.

Reinforcement Learning Resources

- **Video Series:**
  - Steve Brunton's "Control Bootcamp" series on YouTube.
  - Click: Link to the playlist
  - Steve Brunton's "Reinforcement Learning" series on YouTube.
  - Click: Link to the playlist
- **Textbook:**
  - "Reinforcement Learning: An Introduction" [6]
  - **Chap 3:** Basic MDP formulation.
  - **Chap 4:** DP formulation.
  - **Chap 6:** Classic RL algorithms (TD learning, Q learning).
  - **Chap 13:** Policy gradient method.
- **Survey Paper:**
  - *Comprehensive Survey of Reinforcement Learning: From Algorithms to Practical Challenges* [2]

Latent-State Modelling and Inference

State Estimation: An Example

Why need state estimation? Both the system's evolution and our measurements might be imperfect.

The Setup: A Linear System with Noise
1. State Dynamics (What the system does): The state evolves with some disturbances w_{k-1}.

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}$$

2. Observation Model (What we can see): We can't see the true state x_k. We only get a noisy measurement z_k.

$$z_k = Hx_k + v_k$$

Here, H is the observation matrix that maps the state to what we can measure, and v_k is the measurement noise.

3. Analogy: Imagine tracking a ship on a foggy sea. You are trying to predict its next position based on its last known speed and heading.

A Classic Approach: The Kalman Filter [3]

The Kalman Solution

Given a sequence of noisy measurements $\{z_1, z_2, ..., z_k\}$, what is the best possible estimate of the true current state $\hat{x}_k$?

The Kalman Filter provides the optimal solution by recursively performing two steps:

- **Predict:** Use the system dynamics (A, B) to predict the next state.

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$

- **Update:** Use the new measurement (z_k) to correct the prediction.

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-),$$

where K_k is the term weights the new information.

Limitations in a Modern Learning Context

The Kalman Filter's powerful framework relies on key assumptions that are often violated in education:

- **Linearity Assumption**: It assumes learning dynamics are linear. However, human learning could probably have **non-linear** dynamics.
- **Structured Data Requirement**: It requires the (x_k) and observations (z_k) to be fixed-size numerical vectors.
- **The Unstructured Data Challenge**: How do we model a learner's state from rich, unstructured data?
  - An essay explaining their reasoning
  - A dialogue with instructor
  - A piece of code written to solve a problem
- Forcing this data into a simple vector loses critical nuance. We need models that can directly interpret the complexity of natural language and other rich interactions.

This motivates the need for modern, expressive models that bridge this gap...

Related Papers i

Explicit Bayesian Inference to Uncover the Latent Themes of Large Language Models [4] (ACL 2025)

Core Idea: This paper uses a VAE to infer the latent "topics" an LLM is focusing on during text generation. This provides a probabilistic, step-by-step view of the model's thematic focus.

Application to Personalized Learning: This might be useful in analyzing a student's unstructured text, such as a piece of code, oral answer to a question. The "latent themes" can be mapped to, for example, a student's potential misconceptions. The probabilistic nature of the VAE would provide an uncertainty score, indicating how confidently the student's grasp of each concept.

Related Papers ii

Predictive, scalable and interpretable knowledge tracing on structured domains [10] (ICRL 2024)

Core Idea: Presents PSI-KT, a Bayesian inference model that tracks a learner's knowledge by combining their individual learning dynamics with the prerequisite structure of the subject matter. It aims for high interpretability.

Application to Personalized Learning: This work emphasizes the importance of combining a data-driven model of the student with a structured model of the curriculum. The latent state is not just a vector of numbers but is explicitly tied to a knowledge graph of concepts. This allows the system to make highly interpretable predictions. For example, it could infer that a student is struggling with "long division" because their mastery of the prerequisite concept "subtraction" is weak. This provides clear, actionable insights for personalization.

Related Papers iii

Teaching Language Models to Evolve with Users: Dynamic Profile Modeling for Personalized Alignment [9]

Core Idea: Introduces a reinforcement learning framework where an LLM interacts with a simulated user to iteratively build and refine a dynamic user profile. A dual-reward system encourages both accurate profile construction and generating responses consistent with that profile.

Application to Personalized Learning: This is a blueprint for creating a truly adaptive learning agent. The "user profile" is the learner's latent state. The LLM could interact with a student over time, continually updating its internal model of the student's knowledge, learning style, and motivation. The reinforcement learning agent would learn a pedagogical policy—not just what the student knows now, but what is the next best action (which problem to show, what hint to give) to maximize learning, based on this dynamic, evolving profile.

Related Papers iv

Harnessing LLM for Noise-Robust Cognitive Diagnosis in Web-Based Intelligent Education Systems [8]

Core Idea: Proposes a framework called DLLM that uses a diffusion model to denoise student interaction data before aligning it with semantic knowledge from an LLM. This makes the cognitive diagnosis robust to the noisy, inconsistent data common in real-world education scenarios.

Application to Personalized Learning: This addresses a practical problem. Real student data is messy: a student might guess correctly or make a typo on a correct answer. The diffusion-based denoising process can denoise this interaction data, leading to a more accurate and stable estimation of the student's true latent knowledge state.

Adaptive Pedagogical Control

From State Inference to Action: The Control Problem

Question: We can estimate a learner's latent state (e.g., knowledge)... **now what?**

The next step is to decide how to act on this information, where the goal is to learn an optimal teaching policy.

- **State (s_t):** The inferred learner's cognitive and affective state.
- **Action (a_t):** The pedagogical intervention to take.
  - Which problem to present next?
  - When to provide a hint vs. a worked example?
  - When to let the student struggle productively?
- **Policy ($\pi(a_t|s_t)$):** The pedagogical strategy we aim to learn.

Key Challenges:

- **Safe:** Avoid actions that harm motivation or confidence.
- **Fair:** Ensure equitable learning opportunities for all students.
- **Robust:** Must work effectively with new students who may differ from the training data.

Related Papers i

Constraint-Conditioned Policy Optimization for Versatile Safe Reinforcement Learning [7] (NeurIPS 2023)

Core Idea: This paper proposed a framework (CCPO) for learning a single, versatile policy that can adapt to different safety constraints at deployment time without retraining. It uses a novel conditioned variational inference approach to achieve this zero-shot adaptation.

Application to Personalized Learning: An educational system could use this to dynamically adjust its teaching strategy based on real-time needs. For instance, for a student showing signs of frustration, the policy can immediately switch to a more conservative, supportive mode. For a confident student, it can switch to a more challenging, exploratory mode to accelerate learning. This avoids the one-size-fits-all approach of traditional RL policies.

Related Papers ii

Constrained reinforcement learning under model mismatch [5] (ICML 2024)

Core Idea: This work addresses the following probelm: a policy trained in one environment (e.g., with historical student data) may violate safety constraints when deployed in a new, slightly different environment (e.g., with a new batch of students). The proposed RCPO algorithm is designed to be robust to this "model mismatch." (OOD)

Application to Personalized Learning: Pedagogical policies must be robust to the diversity of learners. A policy learned on last year's students might not work for this year's. This work provides a formal method to learn a policy that is guaranteed to be safe and effective even when real environment differs from the training environment, ensuring the system remains reliable and harmless.

Related Papers iii

Fairness-Aware Reinforcement Learning (FAReL): A Framework for Transparent and Balanced Sequential Decision-Making [1]

Core Idea: This paper proposes a framework that can encode individuals and groups into the RL state representation. It allows stakeholders to explore the trade-offs between maximizing performance and ensuring fairness.

Application to Personalized Learning: It can be used to train a pedagogical agent that explicitly optimizes for equity, for instance, by ensuring that the learning gains are distributed fairly across students with different prior knowledge levels or from different demographic groups.

References i


A. Cimpean, N. Orzan, C. Jonker, P. Libin, and A. Nowé.
Fairness-aware reinforcement learning (farel): A framework for transparent and balanced sequential decision-making, 2025.


M. Ghasemi, A. H. Moosavi, and D. Ebrahimi.
A comprehensive survey of reinforcement learning: From algorithms to practical challenges, 2025.


R. E. Kalman.
A new approach to linear filtering and prediction problems.
1960.

References ii

R. Li, C. Li, G. Murray, and G. Carenini.
Explicit Bayesian inference to uncover the latent themes of large language models.
In W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 21819–21833, Vienna, Austria, July 2025. Association for Computational Linguistics.

Z. Sun, S. He, F. Miao, and S. Zou.
Constrained reinforcement learning under model mismatch.
arXiv preprint arXiv:2405.01327, 2024.

R. S. Sutton and A. G. Barto.
Reinforcement Learning: An Introduction.
A Bradford Book, Cambridge, MA, USA, 2018.

References iii


Y. Yao, Z. Liu, Z. Cen, J. Zhu, W. Yu, T. Zhang, and D. Zhao.
Constraint-conditioned policy optimization for versatile safe reinforcement learning.
Advances in Neural Information Processing Systems, 36:12555–12568, 2023.


G. Zhang, G. Yuan, Z. Xu, Y. Zhang, J. Ren, Z. Deng, and D. Cheng.
Harnessing llm for noise-robust cognitive diagnosis in web-based intelligent education systems, 2025.


W. Zhao, X. Sui, Y. Hu, J. Guo, H. Liu, B. Li, Y. Zhao, B. Qin, and T. Liu.
Teaching language models to evolve with users: Dynamic profile modeling for personalized alignment, 2025.

References iv


H. Zhou, R. Bamler, C. M. Wu, and Á. Tejero-Cantero.
Predictive, scalable and interpretable knowledge tracing on structured domains.
arXiv preprint arXiv:2403.13179, 2024.