A Concise Review of Reinforcement Learning Methods: Foundations, Deep RL, & LLM-centric Approaches

Anjan Goswami

February 17, 2025

Contents

1 Introduction and Historical Context

Reinforcement Learning (RL) focuses on training agents to maximize a reward signal by interacting with an environment via sequential actions. Its modern foundations derive from Dynamic Programming [Bellman, 1957], Stochastic Approximation [Robbins and Monro, 1951], and Markov Decision Processes [Puterman, 1994]. Early successes such as Samuel's checkers program [Samuel, 1959] and TD-Gammon [Tesauro, 1995] confirmed RL's promise in game domains. Subsequently, *Deep RL* integrated neural networks [e.g., Mnih et al., 2015, Silver et al., 2017], unlocking rich, high-dimensional applications. Recently, RL has also found important roles in *Large Language Models* (LLMs), where single-step or short-horizon tasks still benefit from careful reward engineering and policy optimization.

2 Classical RL Foundations

2.1 Blocks World, Maze Problems, and Game-Theoretic Views

Consider a discrete Markov Decision Process (MDP) with states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, transition probabilities $P(s'|s,a)$, and reward $r(s,a)$. Examples:

- **Blocks World**: Re-stack blocks to a goal arrangement.
- **Maze Navigation**: Move on a grid to reach a terminal cell.

These tasks illustrate how iterative updates (e.g., Q-learning) converge to optimal policies. In multi-agent or adversarial contexts, RL intersects with *game theory* [Shoham and Leyton-Brown, 2008], replacing a single reward with utility or payoffs in multi-player settings.

2.2 Value-Based: Q-Learning

Q-learning [Watkins and Dayan, 1992] is a key value-based method:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\Big].$$

In a small maze or blocks puzzle, Q can be stored in a table. In large state spaces, function approximation becomes necessary.

2.3 Actor-Critic Methods

Alternatively, *actor-critic* separates the policy (actor) from the value function (critic). It updates parameters θ of $\pi_\theta(a \mid s)$ and ψ of $V_\psi(s)$ in tandem, reducing variance in policy gradient estimates. The temporal-difference error $\delta_t = r_t + \gamma V_\psi(s_{t+1}) - V_\psi(s_t)$ guides both the critic update and the actor update.

3 Deep RL Approaches

3.1 Deep Q-Network (DQN)

Mnih et al. [2015] introduced DQN for Atari games, where a CNN approximates $Q_\psi(\text{pixels}, a)$. Replay buffers and target networks stabilized training. This showed the feasibility of *neural function approximators* in RL.

3.2 Actor-Critic Extensions

DDPG [Lillicrap et al., 2015], *TD3* [Fujimoto et al., 2018], and *SAC* [Haarnoja et al., 2018] extend actor-critic to continuous action spaces, widely used in robotic control. *Model-based RL* attempts to learn a transition model $\hat{P}(s'|s,a)$, allowing planning or forward search, sometimes via matrix computations of approximate Bellman operators [Puterman, 1994].

4 Monte Carlo Tree Search (MCTS) and AlphaZero

MCTS is a *model-based* search technique for discrete, perfect-information settings (e.g., board games). Silver et al. [2017] combined MCTS with deep policy/value networks in self-play, yielding *AlphaZero* that achieved superhuman play in Chess, Go, Shogi. MCTS:

1. Selects a path using an Upper Confidence Bound-like criterion,
2. Expands a leaf,
3. Simulates or uses a value network for a terminal estimate,
4. Backpropagates the result, updating action statistics.

5 RL in Large Language Models (LLMs)

While many RL applications revolve around multi-step environments, LLMs typically see *single* or short-turn interactions (prompt → text output). Nevertheless, RL can dramatically enhance language model alignment, correctness, and user satisfaction.

5.1 Importance of RL for Improving LLM Efficacy

LLMs often generate text purely by next-token prediction from pretraining corpora. However, certain desired properties (*factuality*, *conciseness*, *non-toxicity*) are not guaranteed by raw likelihood objectives. **Reinforcement Learning from Human Feedback (RLHF)** [Ouyang et al., 2022] addresses this gap by defining a *reward function* that captures user or human-annotated preferences, then tuning the model's policy to *optimize that reward*.

- **Quality / Efficacy**: RL optimization for correctness or relevance can drastically reduce hallucinations and incoherent text.
- **Safety / Alignment**: RL-based constraints can penalize harmful or policy-violating outputs.
- **User-specific Goals**: Personalized reward signals can tailor the model for domain-specific tasks or user preferences.

5.2 PPO & FRPO: Why They Brought "Magical" Improvements

PPO (Proximal Policy Optimization) Schulman et al. [2017] introduced a clipping objective to keep new policies from diverging too far from the old policy. For LLMs, Ouyang et al. [2022] adapted PPO with a **KL penalty**, ensuring the updated policy remains close to a reference (SFT) model. This stabilizes training while still letting the model move towards answers that yield higher reward (quality or preference scores).

FRPO (Fine-tuning with Rejection or Relative Policy Optimization) While not always labeled "FRPO" in the literature, certain variants (like *Rejection Sampling Fine-tuning* or *Group Relative Policy Optimization*, a.k.a. GRPO) filter out or relatively score multiple outputs at once. They reduce the memory footprint by skipping a large critic network, use group baselines, or filter out suboptimal responses. The upshot is *ease of training* with consistent improvements in final model quality.

5.3 Reward Functions for RLHF and NDCG Maximization

In typical RLHF setups, a learned reward model $r_\phi(q, o)$ is trained from human preference data (ranking pairs of outputs). One might prefer to optimize metrics like *NDCG (Normalized Discounted Cumulative Gain)*, often used in information retrieval:

$$\text{NDCG}(o) \;=\; \frac{1}{Z}\sum_{k=1}^{|o|}\frac{2^{\text{rel}(o_k)}-1}{\log_2(k+1)},$$

where $\text{rel}(o_k)$ is a relevance or correctness label for token o_k, and Z is a normalization. However, mapping such IR-style metrics to a single scalar reward can be tricky if the text is long. In practice, RLHF often collapses to a simpler single scalar measure (e.g., overall preference) due to annotation or modeling constraints. But if we do partial scoring (token-level or segment-level), we could approximate an NDCG-like objective.

5.4 Inference-Time Computation as an RL Problem

A thought-provoking view is to treat the entire LLM inference process—generating tokens, possibly with user follow-up prompts—as *multi-step RL*:

- **State**: The partial conversation or partial sequence of tokens, i.e., the hidden Transformer state.
- **Action**: Generating the next token from the model's distribution over the vocabulary.
- **Reward**: May be derived from an external correctness function, user satisfaction, or a known ground truth if it exists.
- **Goal**: Provide a final textual solution that matches or outperforms a known standard (like an answer key).

In principle, one could do on-the-fly RL-like updates if partial answers could be tested, but the typical cost of large-scale Transformers and the difficulty of repeated partial evaluations hamper direct MCTS or Q-learning. Instead, we rely on **policy gradients** (PPO-like updates) that adjust the generation distributions from a reference baseline.

During inference, the model's distribution $p_\theta(o_t \mid o_{<t})$ can be seen as a *stochastic policy*, but full environment loops are rarely done due to high computational expense. Nevertheless, in *interactive chat or multi-turn QA*, one can conceptualize each user query $\rightarrow$ response cycle as a single environment step, so multi-step RL could in principle refine the model over repeated user interactions.

6 Challenges in RL Training Systems

Even though RL in LLMs has advanced rapidly, fundamental challenges persist:

- **Reward Design**: Mapping complex objectives (factual correctness, style, policy constraints) to a single scalar remains hard. This is why RLHF invests heavily in preference data collection and reward modeling.
- **Sample Efficiency**: Large-scale RL can require extensive sampling from the current policy. For LLMs, each sample is expensive to generate (many tokens, large batch).
- **Stability & Hyperparameters**: PPO/GRPO rely on careful tuning of learning rates, batch sizes, KL coefficients, or group sizes.
- **Scaling / Distributed Compute**: Some frameworks (e.g., *Anyscale*) claim more efficient distributed RL. The complexities lie in handling large models, large replay or sample buffers, and stable synchronization of updates.

7 Conclusions and Future Directions

Reinforcement Learning has evolved from simple tabular MDPs to complex deep RL solutions that master games (AlphaZero) and real-world tasks (continuous robotics). **LLM-based RL** typically focuses on aligning or improving final text outputs with human preferences, using single-step or short-horizon policy gradient (PPO-like) methods. Despite the difficulties (high dimensionality, expensive sampling, reward design), RL can significantly boost LLM *efficacy* (accuracy, relevance, safety). Future work may explore multi-step approaches, advanced reward shaping (NDCG or partial token-level feedback), and real-time interaction loops to refine language model responses online.

References

R. E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 2018.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, et al. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, et al. Continuous control with deep reinforcement learning. *arXiv:1509.02971*, 2015.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

X. Ouyang et al. Training language models to follow instructions with human feedback. *arXiv:2203.02155*, 2022.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

Arthur L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

John Schulman, Filip Wolski, Prafulla Dhariwal, et al. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2008.

David Silver, Julian Schrittwieser, Karen Simonyan, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.