The Duality of Learning: Reinforcement Learning Navigates Stochasticity and Variable Objectives in a Simulated Betting Environment

AI Pair Programming Project

June 2, 2025

Abstract

This paper investigates the application of Reinforcement Learning (RL), specifically Proximal Policy Optimization (PPO), to model agent behavior in a highly stochastic environment—the game of Two-Up—under varying behavioral objectives, termed "personalities." We developed a flexible Numba-optimized simulation environment supporting a novel action space with variable bet sizes. PPO agents were trained to embody distinct personalities (`loss_averse`, `play_for_time`, `thrill_seeker`, and `standard`) by shaping their reward functions. Our findings indicate that agents successfully learned behaviors consistent with their assigned personalities, such as extreme capital preservation by the `loss_averse` agent. Notably, continued training of a `standard` agent from 100,000 to 200,000 timesteps resulted in an improved mean episode reward during training, yet this did not translate to enhanced performance in deterministic evaluation metrics like average final balance or evaluation-time average reward. This divergence highlights the challenges of generalization in stochastic settings and underscores the critical role of reward function design in aligning training optimization with desired real-world outcomes. The study serves as a practical illustration of RL's capabilities in modeling goal-driven behavior while also emphasizing its limitations when faced with pure stochasticity and potential misalignments between intrinsic rewards and extrinsic performance indicators.

Keywords: Reinforcement Learning, Proximal Policy Optimization, Stochastic Environments, Reward Shaping, Agent-Based Modeling, Game Theory, Two-Up.

1 Introduction

Reinforcement Learning (RL) has emerged as a powerful paradigm for training agents to make optimal decisions in complex, often dynamic, environments [1]. The fundamental goal in RL is to find a policy $\pi(a_t|s_t)$ that maps states s_t to actions a_t in order to maximize the expected discounted cumulative reward, often represented as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

where $\gamma \in [0, 1]$ is the discount factor and r_t is the reward at timestep t. Its success spans diverse fields, from game playing [2, 3] to robotics and autonomous systems. A core challenge in RL is designing agents that not only optimize a given reward signal but also exhibit desired, often nuanced, behaviors, especially when faced with inherent environmental stochasticity.

This project explores these challenges within the context of a classic game of chance, Two-Up. Our primary motivation is twofold: first, to develop RL agents capable of adopting distinct "personalities" or behavioral profiles (e.g., risk-averse vs. thrill-seeking) through tailored reward mechanisms; second, to investigate the efficacy and limitations of RL in an environment where outcomes are fundamentally stochastic and where a "winning" strategy, in the traditional sense, is mathematically elusive. We introduce a novel, variable bet-sizing mechanism into the Two-Up environment, compelling agents to learn not just *what* to bet on, but also *how much* to bet, adding a layer of strategic depth.

This study makes the following contributions:

- Development of a computationally efficient, Numba-optimized Two-Up simulation environment featuring a discrete action space that accommodates variable bet sizes.

- Training of Proximal Policy Optimization (PPO) agents to embody four distinct behavioral personalities through specific reward shaping.
- An empirical analysis of the relationship between an agent's optimization of its training reward signal and its performance on extrinsic evaluation metrics (e.g., final balance, break-even probability), particularly highlighting the impact of extended training in a highly stochastic setting.
- A practical demonstration of RL's utility in modeling complex, goal-oriented behaviors alongside an illustration of its inherent limitations when confronted with irreducible randomness.

2 Methodology

2.1 The Two-Up Environment

The environment is based on the traditional Australian game of Two-Up. Two coins are tossed; outcomes are "Heads" (two heads), "Tails" (two tails), or "Odds" (one head, one tail). Bets are typically placed on Heads or Tails.

Our simulated environment, `TwoUpEnvNumba`, was implemented in Python and optimized using Numba for computational efficiency.

- **Observation Space:** A 6-dimensional continuous vector including: normalized rounds played, the numerical representation of the last three toss results (-1 for none, 0 for Heads, 1 for Tails, 2 for Odds), the last bet amount normalized by initial balance, and current balance normalized by initial balance.
- **Action Space:** A `gymnasium.spaces.Discrete(7)` space, representing:
  - Action 0: No Bet.
  - Actions 1-3: Bet on Heads with multipliers $[0.5x, 1.0x, 2.0x]$ of a `base_bet_unit`.
  - Actions 4-6: Bet on Tails with multipliers $[0.5x, 1.0x, 2.0x]$ of a `base_bet_unit`.

  The `base_bet_unit` was set to \$10, and the initial balance to \$100 for all experiments. Episode length was capped at 200 steps.
- **Dynamics:** If Odds are tossed, the bet is a push (stake returned). Winning bets on Heads or Tails pay 1:1. A bet resulting in a balance of ≤ 0 terminates the episode.

2.2 Agent Personalities and Reward Structures

Four distinct agent "personalities" were defined, with the PPO agent's reward function implicitly shaped during training via the `gambler_personality` parameter in the environment, which modified the reward calculation r_t based on game events (wins, losses, no-bets, bankruptcy). For example, the reward function for a `loss_averse` agent might be conceptualized as:

$$r_t = w_1 \cdot \Delta B_t - w_2 \cdot \mathbb{I}(B_t < B_{t-1}) - w_3 \cdot \mathbb{I}(B_t \leq 0) + c$$

where ΔB_t is the change in balance, $\mathbb{I}(\cdot)$ is the indicator function, w_1, w_2, w_3 are positive weighting factors (with w_2 and w_3 being particularly large for `loss_averse`), and c is a small constant reward or penalty (e.g., for each step taken).

Specific personalities included:

- `standard`: A baseline personality with balanced rewards/penalties.
- `thrill_seeker`: Presumably rewarded more for large wins and potentially less penalized for risks.
- `loss_averse`: Presumably heavily penalized for losses and bankruptcy, incentivizing capital preservation.
- `play_for_time`: Presumably rewarded for episode longevity, encouraging cautious play or no-bet actions.

2.3 Reinforcement Learning Agent

- **Algorithm:** Proximal Policy Optimization (PPO) [4], using the implementation from Stable Baselines3 [5]. PPO algorithms are policy gradient methods that optimize a clipped surrogate objective function. A simplified version of the PPO-clip objective is:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$$

  where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio, $\hat{A}_t$ is an estimator of the advantage function, and ϵ is a hyperparameter, typically 0.1 or 0.2.

- **Network:** Default Multi-Layer Perceptron (MLP) policy (`MlpPolicy`).
- **Training:** Agents were typically trained for 100,000 timesteps. The `standard` agent underwent an additional 100,000 timesteps of continued training to assess the impact of extended learning. Key hyperparameters included a learning rate of 0.0003. Training was conducted on a CPU. TensorBoard was used for logging training progress, particularly `rollout/ep_rew_mean`.

2.4 Baseline Strategies

To contextualize the PPO agents' performance, several rule-based strategies were implemented and evaluated:

- Always No Bet: Never places a bet.
- Random Action (New Space): Uniformly samples from the `Discrete(7)` action space.
- Always Bet Heads (1x): Always bets 1.0x `base_bet_unit` on Heads.
- Always Bet Tails (1x): Always bets 1.0x `base_bet_unit` on Tails.
- Martingale Heads (Multi-level): Bets on Heads, doubling the bet multiplier (0.5x $\rightarrow$ 1.0x $\rightarrow$ 2.0x) on a loss, resetting to 0.5x on a win.

2.5 Evaluation Metrics

Each strategy and trained PPO agent was evaluated over 1,000 episodes with deterministic actions for PPO agents. Key metrics included:

- Average final balance.
- Standard deviation of final balance.
- Break-even probability (percentage of episodes where final balance $\geq$ initial balance).
- Average total episode reward (accumulated reward during an evaluation episode, using the agent's learned personality context).

3 Results

3.1 Baseline Strategy Performance

Rule-based strategies established performance benchmarks. "Always No Bet" consistently achieved a 100% break-even rate with a final balance equal to the initial balance. "Always Bet Heads (1x)" and "Always Bet Tails (1x)" yielded break-even probabilities of approximately 48%. The "Random Action (New Space)" strategy achieved around 45% break-even. "Martingale Heads (Multi-level)" had a lower break-even probability of $\sim$39%, reflecting its inherent risk of ruin.

3.2 PPO Agent Performance by Personality

PPO agents successfully learned behaviors aligned with their designated personalities:

- `loss_averse`: Achieved a 100% break-even probability, indicating an extremely conservative strategy focused on capital preservation. Average final balance was effectively the initial balance.
- `play_for_time`: Also demonstrated strong capital preservation with an 85.3% break-even probability.
- `thrill_seeker`: Exhibited a lower break-even probability (34.4%), consistent with a risk-taking profile.
- `standard` (initial 100k timesteps): Achieved an average final balance of $107.68 and a break-even probability of 38.7%. Its average evaluation reward was not explicitly captured in this initial phase but was implicitly positive given the final balance.

3.3 Impact of Continued Training on the "Standard" Agent

The `standard` agent was trained for an additional 100,000 timesteps (total 200k).

- **Training Reward (`rollout/ep_rew_mean` from TensorBoard):** The smoothed training reward for this agent (run `standard/.../ppo_two_up_standard_5`) showed a notable increase in the 100k-200k step phase, rising from values fluctuating near zero or negative around the 100k mark to a smoothed value of +10.8 by 200,704 steps (Figure 1 - *referencing the provided image*). Other contemporaneous runs trained for only 100k steps showed smoothed `ep_rew_mean` values of -0.5 and -5.65 at their conclusions.
- **Evaluation Metrics (after 200k total timesteps, report `...-231952.json`):**
  - Average final balance: $100.26 (a decrease from $107.68 after 100k steps).
  - Average total episode reward (evaluation): -5.31.
  - The break-even probability (derived from analyzing the raw data from the corresponding report via `analyze_breakeven_chances.py`) was also observed to not improve, aligning with the drop in average final balance.

4 Discussion

4.1 Alignment of Agent Behavior with Personalities

The distinct performance profiles of the PPO agents (e.g., 100% break-even for `loss_averse` vs. 34.4% for `thrill_seeker`) demonstrate PPO's capability to optimize for varied, personality-driven reward functions. Agents learned strategies that qualitatively matched their intended behavioral objectives, highlighting the expressive power of reward shaping in RL.

4.2 The "Standard" Agent: Training Optimization vs. Evaluation Performance

The most striking finding pertains to the `standard` agent. While its `ep_rew_mean` during training significantly improved between 100k and 200k timesteps (from $\sim$0 to +10.8), its performance on key evaluation metrics (average final balance, average evaluation reward) either stagnated or degraded. This divergence suggests that while PPO was effectively optimizing the cumulative reward signal as experienced *during training* (which includes exploration), the resultant policy did not generalize favorably to deterministic evaluation.

This phenomenon can be attributed to the highly stochastic nature of the Two-Up environment. The agent, in its extended training, may have learned to exploit specific stochastic sequences or noisy signals present during its training trajectory. These "optimizations" might be brittle, failing to hold under deterministic exploitation or when faced with slightly different (yet statistically equivalent) sequences of random events. The negative average total episode reward (-5.31) in evaluation for the 200k-step agent, despite a positive training reward trend, further underscores this disconnect. It indicates that the strategy, while perceived as increasingly rewarding by the agent during learning, was, on average, detrimental when assessed objectively under evaluation conditions.

4.3 Reinforcement Learning in Highly Stochastic Environments

This study corroborates the understanding that RL cannot conjure winning strategies from purely stochastic processes that lack exploitable information or an inherent statistical imbalance favoring the agent. In Two-Up, a fair game, no amount of learning can guarantee long-term profit. The agent's attempt to find patterns in randomness can lead to overfitting to the specific training history, as likely evidenced by the `standard` agent. The project thus serves as a practical illustration of the boundaries of RL application: it excels where learnable patterns and cause-effect relationships exist, but struggles against irreducible uncertainty.

4.4 Implications of Reward Design and Fast Simulation

The critical role of reward function design is again highlighted. The `standard` agent's experience suggests that even a seemingly neutral reward structure can lead to policies that, while optimal for that structure during training, might not align with external metrics of success if those metrics are not perfectly encapsulated by the reward. Furthermore, the ability to rapidly iterate due to the Numba-optimized environment was crucial for conducting the extensive training and evaluation runs necessary for these observations.

5 Conclusion and Future Work

This project successfully demonstrated the use of PPO to train RL agents exhibiting distinct "personalities" in a variable bet-size Two-Up game. Agents learned behaviors consistent with their shaped reward functions, showcasing RL's adaptability. However, the extended training of a `standard` agent revealed a critical insight: optimizing rewards during training in a highly stochastic environment does not guarantee improved, or even stable, performance on desired evaluation metrics. The agent improved its ability to garner training rewards, but this strategy proved less effective, or even detrimental, under deterministic evaluation.

This underscores that for environments dominated by chance, RL agents may learn policies that overfit to training data or exploration dynamics, rather than robust, generalizable strategies. The endeavor highlights the importance of careful reward engineering to align intrinsic motivation with extrinsic goals and the need for rigorous evaluation to detect divergences between training progress and real-world efficacy.

Future Work could explore several avenues:

- Refining the reward function for the `standard` agent to better promote desired evaluation outcomes like higher final balance or break-even probability.
- Conducting more extensive hyperparameter tuning for the PPO algorithm for each personality.
- Investigating alternative RL algorithms, perhaps model-based approaches, to see if they offer different learning dynamics in this stochastic context.
- Introducing a "house edge" to the game to explore learning against a non-neutral opponent.
- Exploring curriculum learning or more sophisticated exploration strategies.

References

References

[1] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[2] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529-533.

[3] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature, 529*(7587), 484-489.

[4] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[5] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research, 22*(268), 1-8.