RL Chef: Tabular Reinforcement Learning vs Linear Function Approximation and the Impact of State Representation

Luigi Daddario (mat. 908294)
Artificial Intelligence For Science And Technology
University of Milano-Bicocca
Email: l.daddario1@campus.unimib.it

***Abstract*—We study a custom Gymnasium environment, *RL Chef*, where an agent navigates a 5×5 grid, collects ingredients under supply constraints, and decides when to cook a dish to maximize reward. We compare tabular control (Q-learning) with linear function approximation for the action-value function, and we analyze how state representation affects learning and convergence. In particular, we contrast an intentionally lossy, non-Markov representation (`simple`) with a Markov representation that augments the state with a visited-cells mask (`mask`). Across multiple random seeds we report learning curves, final evaluation return, and interpretable domain metrics (waste and incompatible ingredient usage).**

1. Introduction

Reinforcement learning (RL) studies how an agent can learn to act optimally through interaction with an environment modeled as a Markov Decision Process (MDP) [2]. In small discrete domains, tabular methods such as Q-learning can converge to the optimal action-value function under standard assumptions [2]. In larger state spaces, value function approximation is typically required to enable generalization [2].

This report focuses on *RL Chef*, a compact gridworld designed to highlight two key themes: (i) the trade-off between tabular control and linear function approximation, and (ii) the critical role of state representation and Markovity. The environment includes delayed decision-making (choosing when to cook), penalties for waste, and constraints induced by one-time ingredient availability, creating a non-trivial exploration–exploitation problem even on a small grid.

2. Objectives

The objectives of this project are:

- Model the cooking task as an MDP and implement it with the Gymnasium API [3].
- Implement and compare:
  - tabular Q-learning (and optionally SARSA);
  - linear action-value approximation trained with temporal-difference updates.
- Analyze the impact of state representation by comparing an intentionally lossy, non-Markov representation (`simple`) with a Markov representation that augments the state with a visited-cells mask (`mask`).
- Produce reproducible results across multiple seeds, including learning curves, evaluation metrics, and domain-specific diagnostics.

3. Methods

3.1. Dataset

No external dataset is used. All experience is generated online by interacting with the environment and collecting trajectories of (s_t, a_t, r_t, s_{t+1}).

3.2. Optimization and Regularization

We use ϵ-greedy exploration with an optional linear decay schedule ϵ_t from ϵ_0 to ϵ_{final} over a fixed number of episodes. The discount factor is fixed to $\gamma \in (0, 1)$. For tabular control we update a Q-table (Q-learning update); for linear approximation we parameterize:

$$Q_\theta(s, a) = w_a^\top \phi(s) + b_a$$

and perform semi-gradient temporal-difference updates on the parameters. L2 regularization is available in the implementation but is set to 0 in the main experiments.

3.3. Environment and State Representations

The environment is a 5×5 grid. Each cell contains one ingredient type; ingredients can be collected at most once per episode (supply constraint). The agent chooses among 5 actions: 4-direction movement plus `cook`. The `cook` action ends the episode and yields a reward based on the best feasible recipe given the inventory, minus penalties for waste and incompatible ingredient combinations.

We compare two discrete state keys for tabular control:

- **`simple`**: position and inventory counts (lossy / non-Markov because it omits which cells have been depleted).

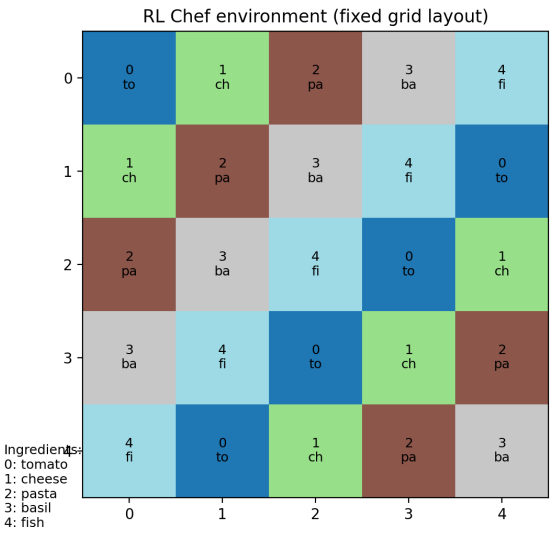

Figure 1. Static overview of the fixed grid layout used in experiments (ingredient id and short name per cell).

TABLE 1. ENVIRONMENT PARAMETERS AND RECIPE DEFINITIONS (AUTO-GENERATED FROM CODE).

Parameter	Value
Grid size	5×5
Actions	5 (up, down, left, right, cook)
Max steps	60
Pickup reward	0.05
Move penalty	0.01
Waste penalty	0.20 per wasted item
Incompatibility penalty	0.30 per incompatible pair
Fail penalty (no feasible recipe)	0.50
Extra penalty (cook with empty inv.)	0.70
Recipes (value; required ingredients)	
margherita	4.0 ; (1, 1, 0, 1, 0)
pasta$_{al}$*pomodoro*	3.5 ; (1, 0, 1, 1, 0)
cheesy$_{p}$*asta*	3.0 ; (0, 1, 1, 0, 0)
fish$_{s}$*pecial*	3.2 ; (0, 0, 0, 1, 1)

- **`mask`**: augments `simple` with a bitmask of visited/depleted cells, restoring Markovity under a fixed grid layout.

Both agents also receive a continuous observation vector with normalized features (position, current-cell ingredient one-hot, availability, inventory, remaining supply; and optional budget/round features in the challenging variant).

4. Experiments

We run a benchmark that compares four configurations across multiple random seeds:

- Q-learning with `simple` state;
- Q-learning with `mask` state;
- linear function approximation with `simple` observation;
- linear function approximation with `mask` observation.

The main results in this paper use the **base** environment variant (no budget constraint; single-round episodes). We additionally report results for a **budget** variant, where the agent must pay ingredient costs from an initial budget and receives non-negative revenue when cooking (default settings: 1 round, start budget = 10, no debt).

Each run trains for 20,000 episodes with ϵ-greedy exploration ($\epsilon_0 = 0.3 \rightarrow \epsilon_{\text{final}} = 0.05$ linearly over 3,000 episodes) and uses 10 random seeds. Final evaluation sets $\epsilon = 0$ and estimates metrics over 1,000 evaluation episodes per run. We report the mean and standard deviation (across seeds) of evaluation return, average steps, and domain-specific diagnostics (waste and incompatibility). Additionally, we compute sample-efficiency proxies from the training curve: the average training return over all episodes (AUC proxy) and the mean return over the last episodes (stability / late performance).

4.1. Implementation details and reproducibility

The base benchmark can be reproduced by running `python -m rlchef.experiments --variant base` followed by `python -m rlchef.analyze` (outputs in `results/rlchef/`). The budget benchmark can be reproduced similarly with `--variant budget` (outputs in `results/rlchef_budget/`).

Compute cost is modest for this environment: on a standard macOS laptop, a single seed run with 20,000 training episodes (covering all four configurations) takes on the order of tens of seconds, and the 10-seed benchmark completes within minutes; plotting/aggregation adds negligible overhead relative to training.[1]

4.2. Results

Table 2 reports aggregate evaluation metrics across seeds. Q-learning achieves high return with low variability: 3.307 ± 0.104 for `simple` and 3.177 ± 0.079 for `mask`. In contrast, the linear baseline is considerably less consistent, achieving 1.652 ± 0.718 in evaluation (Table 2) and showing a wide inter-seed dispersion (Figure 4).

Figure 2 summarizes training dynamics (moving average; mean $\pm$ std across seeds). Tabular learning reaches a stable plateau within a few thousand episodes. The `mask` representation does not improve over `simple` within our training budget; a plausible explanation is that augmenting the state with a visited-cells mask increases the effective tabular state space, which can slow down learning despite restoring Markovity in principle. Under a fixed grid layout, the lossy `simple` key may still be sufficient to learn a near-optimal policy for this task.

Beyond scalar return, domain diagnostics reveal qualitative policy differences. Q-learning yields short episodes

1. Wall-clock time depends on hardware and Python environment; the goal here is to provide an order-of-magnitude estimate.

($\approx$ 4 steps on average), suggesting it learns to collect a small, high-value set of ingredients and cook quickly (Figure 5). The linear baseline produces significantly longer episodes (25.02 ± 13.26 steps), higher waste (2.18 ± 1.41), and a broader trade-off curve (Figure 7), consistent with slower/less decisive policies and suboptimal ingredient collection. Figure 8 shows that the tabular policy produces a more diverse recipe mix, while the linear baseline concentrates on a smaller set of outcomes and exhibits a larger fraction of failures (no feasible recipe cooked).

Finally, Figure 9 compares sample-efficiency proxies derived from training curves. Q-learning achieves higher average train return (AUC proxy) and higher late-stage performance, while the linear baseline is lower on both metrics, matching the observed evaluation gap.

4.3. Budget-variant results (additional experiment)

Table 3 and Figure 3 report the same benchmark under the budget constraint. Under these default budget settings, the task becomes significantly harder: tabular policies achieve low but stable positive returns (0.427 ± 0.008 for `simple`, 0.360 ± 0.030 for `mask`), while the linear baseline collapses to a constant failure outcome (-1.200 ± 0.000).

4.4. Discussion and Limitations

These results highlight a key practical point: in small, structured discrete environments, tabular control can be both data-efficient and stable [1], [2]. The linear baseline here uses a simple feature vector and an off-policy TD update, which can be sensitive to feature design, exploration schedule, and reward shaping. High variance across seeds (Figure 4) suggests that learning outcomes depend strongly on early trajectories and exploration decisions.

The `simple`/`mask` comparison should be interpreted carefully for function approximation. In our implementation, `state_mode` affects only the discrete tabular key; the observation features used by the linear model are unchanged, hence `simple` and `mask` are expected to be identical for the linear baseline (as observed in Table 2 and Figures 5–9).

Limitations include the absence of extensive hyperparameter tuning and the simplicity of the linear approximator. Future work could (i) add the visited-cells mask (or a learned memory) to the feature representation for function approximation, (ii) evaluate on randomized grid layouts for robustness, and (iii) compare additional baselines (e.g., SARSA, eligibility traces, or more stable variants/analyses for temporal-difference learning with function approximation [4]).

5. Conclusion

We presented *RL Chef*, a compact Gymnasium environment designed to compare tabular control with linear value function approximation and to study the impact of state representation. Across 10 seeds, tabular Q-learning reliably achieves high return with low variability, while the linear baseline is less stable and incurs higher waste and longer episodes. The analysis illustrates how representational choices and approximation can dominate performance even in small environments, motivating careful state design and more expressive function approximation for scalable RL.

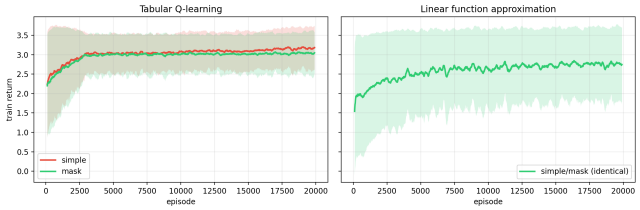

Figure 2. Aggregated learning curves (moving average; train return) across seeds: mean $\pm$ std.

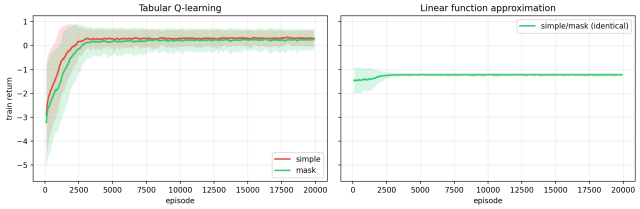

Figure 3. Budget variant: aggregated learning curves (moving average; train return) across seeds: mean $\pm$ std.

6. Visualizations

Figure 2 shows aggregated learning curves (moving average window $= 200$ episodes; mean $\pm$ std across seeds). Table 2 summarizes the aggregate evaluation metrics.

TABLE 2. AGGREGATE EVALUATION METRICS ACROSS SEEDS (MEAN ± STD).

Algo	State	n	Return (eval)	Steps (eval)	Waste	Incompat
linear	mask	10	1.652 ± 0.718	25.02 ± 13.26	2.18 ± 1.41	0.05 ± 0.03
linear	simple	10	1.652 ± 0.718	25.02 ± 13.26	2.18 ± 1.41	0.05 ± 0.03
qlearning	mask	10	3.177 ± 0.079	3.92 ± 0.24	0.68 ± 0.13	0.05 ± 0.02
qlearning	simple	10	3.307 ± 0.104	3.96 ± 0.25	0.61 ± 0.13	0.04 ± 0.03

TABLE 3. BUDGET VARIANT: AGGREGATE EVALUATION METRICS ACROSS SEEDS (MEAN ± STD).

Algo	State	n	Return (eval)	Steps (eval)	Waste	Incompat
linear	mask	10	-1.200 ± 0.000	1.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
linear	simple	10	-1.200 ± 0.000	1.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
qlearning	mask	10	0.360 ± 0.030	2.94 ± 0.04	0.00 ± 0.00	0.00 ± 0.00
qlearning	simple	10	0.427 ± 0.008	3.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00

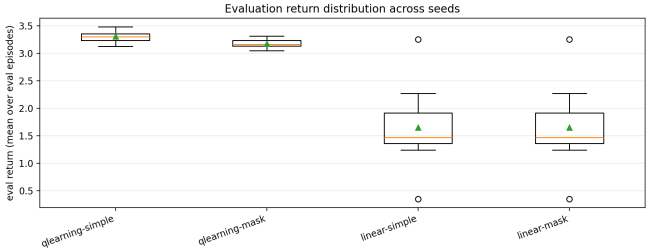

Figure 4. Evaluation return distribution across seeds (boxplot).

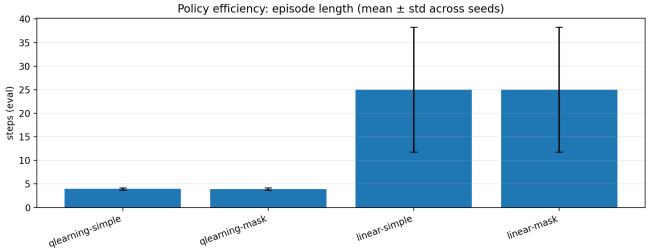

Figure 5. Episode length in evaluation (mean ± std across seeds).

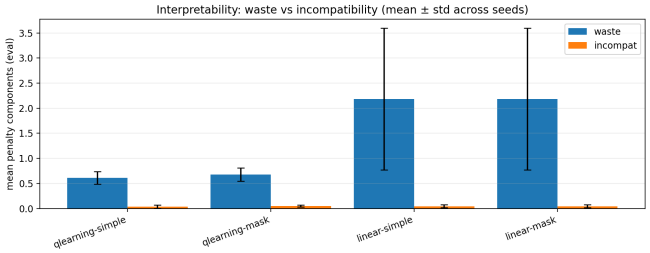

Figure 6. Domain diagnostics: waste and incompatibility (mean ± std across seeds).

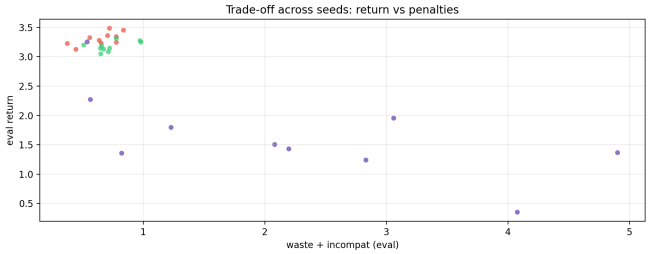

Figure 7. Run-level trade-off: evaluation return vs penalties (waste + incompatibility).

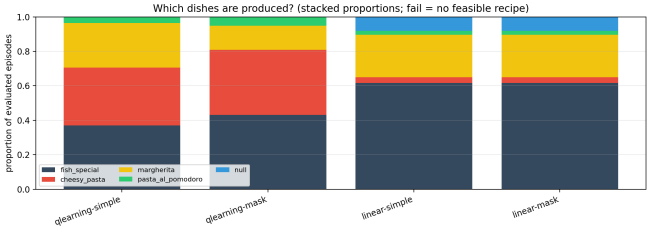

Figure 8. Which dishes are produced? Stacked proportions over evaluation episodes; `fail` indicates no feasible recipe cooked.

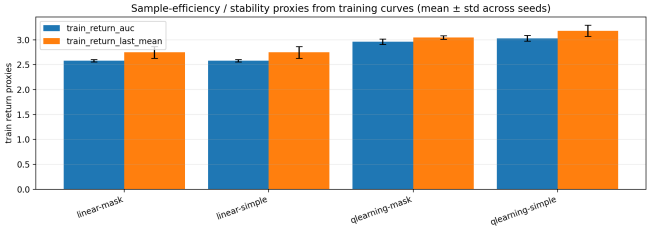

Figure 9. Training-curve proxies: mean train return (AUC proxy) and last-episodes mean (stability), mean ± std across seeds.

Disclosure Statement

The author declares that this report is entirely original and does not contain any plagiarism.

References

[1] S. Mannor, Y. Mansour, and A. Tamar, *Reinforcement Learning: Foundations*, 2023 (last update Nov. 2025). [Online]. Available: https://sites.google.com/view/rlfoundations/home

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[3] Farama Foundation, "Gymnasium: A Standard API for Reinforcement Learning Environments," Accessed: Jan. 20, 2026. [Online]. Available: https://gymnasium.farama.org/

[4] G. Patil et al., "Finite time analysis of temporal difference learning with linear function approximation," in *Proc. Intl. Conf. Mach. Learn. (ICML)*, Jul. 2023. [Online]. Available: https://proceedings.mlr.press/v206/patil23a.html