RL Project Proposal:

Influence of Transfer Learning on Performance

Can Kocak and Paul Steinbrink (rl_ckps)
https://github.com/cako2025/rl_ckps_final_project/

1 Motivation

Transfer learning in reinforcement learning promises to reuse prior knowledge to accelerate convergence and improve learning stability. Off-policy methods may offer greater sample efficiency but can suffer from instability, while on-policy methods are more stable but slower to converge.

In this work, we explore whether transferring value functions between Q-Learning (off-policy) and SARSA (on-policy) can combine the strengths of both approaches. Furthermore, we investigate how different exploration strategies, namely ϵ-greedy and softmax, interact with the transfer process and whether they moderate the observed benefits or drawbacks.

This motivates the following general hypothesis framework:

- **H_0:** Transfer learning does not significantly improve evaluation performance (sample efficiency) or training stability compared to learning from scratch, regardless of the algorithmic combination (on-policy or off-policy) or exploration strategy used.
- **H_1:** Transfer learning improves evaluation performance and/or training stability over solo training, and these effects vary depending on the interaction between the algorithmic combination (on-policy or off-policy) and the chosen exploration strategy.

2 Related Topics

RL Algorithms: Q-Learning (off-policy), SARSA (on-policy)
Exploration Strategies: ϵ-greedy, softmax
Techniques: Solo, Transfer Learning, Hyperparameter Optimization

3 Idea

We investigate the effect of transfer learning on sample efficiency and learning stability by training policies with Q-Learning and SARSA, transferring them across algorithms, and analyzing the learning dynamics with two different exploration strategies.

- **Sample efficiency:** faster increase in evaluation performance in early training stages.
- **Stability:** lower variance in training loss (TD error) and evaluation return.

The formal goal is:

$$\pi \in \Pi, \pi : \mathcal{S} \to \mathcal{A}, \quad \text{maximize } \frac{\mathbb{E}[R_{1:T}(\pi)]}{T} \quad \text{under constraints of different pre-trained initializations} \tag{1}$$

Algorithm 1 Transfer Learning with Cross-Algorithm Evaluation

Require: environment e, RL algorithms A_1, A_2, strategy $S \in \{\epsilon\text{-greedy}, \text{softmax}\}$
  Train policy π_{pre} with A_1 and strategy S
  Initialize:
    Transfer: $\pi_{\text{transfer}} \leftarrow \pi_{\text{pre}}$
    Scratch: randomly initialize π_{scratch}
  while not converged **do**
    Train π_{transfer} with A_2, strategy S
    Train π_{scratch} with A_2, strategy S
  end while
  Evaluate and compare sample efficiency, final performance, and variance

4 Experiments

Environment We use the Blackjack-v1 environment from Gymnasium as a discrete, episodic, and inherently stochastic task. Its manageable state-action space makes it suitable for tabular reinforcement learning, allowing for explicit policy analysis. Due to its randomness in card drawing and reward signals, it is particularly sensitive to differences in exploration behavior and algorithmic structure, making it ideal for studying the effects of transfer learning between on-policy and off-policy agents.

Metrics We compare agents trained from scratch (solo) to agents initialized via transfer learning, using the following metrics:

- **Sample Efficiency:** Measured as the average episodic reward over a defined window of early episodes.
- **Asymptotic Performance:** Final average reward (win rate).
- **Training Stability:** Measured via variance in reward and TD-error.
- **Training Error:** Mean temporal difference (TD) error per episode as a proxy for convergence behavior.
- **Statistical Tests:** T-tests and Levene tests are used to assess significance in sample efficiency and stability between solo and transfer agents.

Experimental Scope

- **Algorithms:** Q-Learning, SARSA
- **Exploration strategies:** ϵ-greedy, softmax
- **Seeds:** 5 different seeds for each config
- **Hyperparameter optimization:** via SMAC
- **Transfer learning episode splits:** Different pretraining episode splits

Estimated Computational Load Training times vary by algorithm but are expected to be moderate due to the simplicity of the Blackjack environment. HPO with SMAC will be the most computationally demanding step. Experiments will be automated via scripts and run on a local machine. Overall, the project is designed to be computationally feasible within a few days to a week.

5 Timeline

- **Research (2 days):** Review on sample efficiency, algorithms, and prior TL studies.
- **Implementation (2.5 days):** Build training/evaluation framework and SMAC HPO setup.
- **Hyperparameter Optimization (2 days):** Optimize settings for each agent-strategy pair.
- **Experiments (4 days):** Transfer experiments, cross-algorithm runs.
- **Analysis (2 days):** Visualizations, statistical comparison, performance trend interpretation.
- **Reporting (2 days):** Write final report and prepare presentation slides.