# Chapter 4
# Performance Evaluations

**Abstract** In this chapter, we discuss performance evaluation metrics and methods in details. Performance evaluation of reinforcement learning systems or algorithms are important to system monitoring and development improvements. It can be carried out in both offline and online. Offline evaluation evaluates the agent's performance on one or multiple fixed datasets of experiences. Online performance evaluation assesses the agent's performance in real-time interaction with the environment. Benchmarking is a common process to compare the performance of a newly developed reinforcement learning algorithm or reinforcement learning system to that of the established ones. It's generally carried out offline but some literatures also refer A/B testing as benchmarking in the sense that the method to be compared with is the reference one. Performance evaluation through simulator and emulator is another way to assess the reinforcement learning performance, and the effectiveness of this method itself heavily depends on the quality of the simulator or emulator used.

## 4.1 Evaluation Metrics

The key metrics for reinforcement learning performance evaluation include reward, sucess rate, learning curve, sample efficiency, exploration vs exploitation, and convergence. By carefully considering these metrics and evaluation methods, you can effectively assess the performance of your RL agents and make informed decisions about their development and deployment. We list the general definition of these metrics as below:

- Reward
  - Average Reward: The cumulative reward divided by the number of episodes or steps. Two popular average reward is used: average reward per episode and average reward per step. Average reward per episode is the total reward divided by the number of episodes. Average reward per step is the total reward divided by the total number of steps.
  - Discounted Reward: The sum of rewards, discounted by factors that give more weight to immediate rewards. In the majority of literatures, weights used grow

exponentially along time with the same discount factor $\gamma$. This helps to balance short-term and long-term goals.
  - Average Discounted Reward: The average discounted reward per episode or step.
  - Cumulative Reward: The total reward accumulated over a fixed number of episodes or steps.
- Success Rate: The percentage of episodes in which the agent achieves a predefined goal or reaches a specific state.
- Episode Length: The average number of steps taken per episode.
- Learning Curve: A plot that shows how the agent's performance (e.g., average reward) improves over time.
- Sample Efficiency: The amount of experience (number of interactions with the environment) required for the agent to reach a desired level of performance.
- Exploration vs. Exploitation: The balance between exploring new actions to discover better rewards and exploiting known good actions to maximize immediate rewards.
- Convergence: Whether the agent's performance stabilizes over time or continues to improve indefinitely.

Cumulative reward, Success Rate, learning curve, sample efficiency, and convergence are used frequently as performance metrics in various reinforcement learning systems.

There are additional Considerations that can be used as performance metrics, depending on the applications.

- Task Complexity: The difficulty of the RL task can influence the choice of metrics and the expected performance.
- Environment Dynamics: The nature of the environment (deterministic, stochastic, continuous, discrete) can impact the evaluation process.
- Agent Architecture: The complexity and design of the RL agent can affect its performance.
- Hyperparameters: The choice of hyperparameters (e.g., learning rate, discount factor) can significantly influence the agent's behavior.

There are also specific metrics for different tasks. We present several examples for specific tasks as below:

- Navigation Tasks
  - Distance Traveled: The total distance covered by the agent.
  - Goal Completion Time: The time taken to reach the goal. It tells how quickly the agent can achieve a goal
  - Energy Consumption: For tasks involving physical agents, energy consumption can be a critical metric.
- Control Tasks
  - Control Error: The deviation from the desired control signal.

  - Stability: The ability to maintain a stable state.
  - Safety Metrics: In safety-critical applications, metrics like collision rate or violation of constraints are essential.
- Game-Playing tasks
  - Win Rate: The percentage of games won.
  - Score: The total score achieved in the game.

Most of these performance metrics and considerations can be either used in offline performance evaluation, online performance evaluation, or both.

## 4.2 Offline Performance Evaluation

The offline evaluation is usually more straight-forward than online evaluation. The evaluation is carried out on offline datasets that are usually predefined before the training. For RL problem with episodic behaviors, many episodes in the testing dataset are conducted with different initial conditions which are usually randomized and preset parameters like the exploration rate $\epsilon$, for upto a time limit or number limit to avoid the experiment/evaluation runs for too long.

Evaluating the performance of an RL agent offline also presents unique challenges due to the lack of real-time interaction. Here are some common approaches.

- On-Policy Evaluation
  - Direct Policy Evaluation (DPE): This involves directly evaluating the policy on the offline dataset. However, it can be biased if the offline dataset is not representative of the true environment.
  - Importance Sampling: This technique weights the samples in the dataset based on how likely they are to be encountered under the current policy. It can help to reduce bias but can suffer from high variance if the policies are very different.
  - Doubly Robust Estimators: Combine DPE and importance sampling to reduce bias and variance.
- Off-Policy Evaluation
  - Q-value-based Methods: Estimate the Q-values of the current policy on the offline dataset and use them to evaluate its performance.
  - Behavior Cloning: Train a supervised learning model to mimic the behavior of the original policy on the offline dataset. This can be used to evaluate the performance of the original policy indirectly.
  - Counterfactual Estimators: Estimate the counterfactual outcomes of different actions under the current policy, allowing for a more direct evaluation.
- Simulation-Based Evaluation
  - Synthetic Environments: Create simulated environments that mimic the real-world environment and evaluate the agent's performance in these simulations.

  - Transfer Learning: Use the knowledge gained from training on the offline dataset to improve performance in a new, similar environment.
- Benchmarking
  - Standard Datasets: Use publicly available datasets to compare the performance of different offline RL algorithms.
  - Real-World Applications: Evaluate the performance of offline RL agents on real-world tasks to assess their practical applicability.

Offline evaluation of reinforcement learning systems faces several challenges and considerations with respect to data quality, distribution shift, evaluation bias and computational cost:

- Data Quality: The quality of the offline dataset is crucial for accurate evaluation. A biased or incomplete dataset can lead to misleading results.
- Distribution Shift: The distribution of states and actions in the offline dataset may not match the distribution in the real world, leading to performance degradation.
- Evaluation Bias: The choice of evaluation method can introduce bias, affecting the accuracy of the results.
- Computational Cost: Some evaluation methods can be computationally expensive, especially for large datasets or complex models.

By carefully considering these factors and selecting appropriate evaluation techniques, you can effectively assess the performance of offline RL agents and make informed decisions about their development and deployment.

## 4.3 Online Performance Evaluation

Online performance evaluation of reinforcement learning systems involves evaluating an agent's performance as it interacts with the environment in real-time. This approach provides valuable insights into the agent's behavior and learning progress. The key metrics include reward, sucess rate, episode length, and learning curve.

The frequently used online evaluation techniques including real-time monitoring, A/B testing, and human evaluation. Real-time monitoring tracks the agent's performance during training and adjust the learning process accordingly. A/B testing compare the performance of different reinforcement learning agents or algorithms in real-time. Human evaluation involves human experts to assess the agent's performance and provide feedback.

Online performance evaluation of reinforcement learning systems faces unique challenges. Firstly, reinforcement learning agents often face a trade-off between exploration (trying new actions) and exploitation (choosing actions that have worked well in the past). Online evaluation can help to assess how well the agent balances these two factors, however the alignment of exploration-exploitation tradeoffs between training and testing is critical to obtain an accurate assessement. Furthermore,

online evaluation can provide immediate feedback on the agent's performance, allowing for adjustments to the learning process as needed. However, this usually comes with the cost of delayed agent behaviors in the main procedure. Finally, Online evaluation can be computationally expensive, especially for complex environments or large-scale RL problems.

### 4.3.1 Real-time Monitoring

Real-time monitoring is a critical aspect of reinforcement learning (RL) that involves continuously tracking and analyzing an agent's performance as it interacts with the environment. This enables timely adjustments to the learning process and helps to identify potential issues early on.

Generally, there are five key components of real-time monitoring: performance metrics, visualization, logging, alerts, and debugging tools. Performance metrics track key metrics such as reward, success rate, episode length, and learning curve to assess the agent's progress. Visualization uses visualizations like plots, graphs, and animations to provide a clear and intuitive understanding of the agent's behavior. Logging records relevant data points for later analysis and debugging. Alerts sets up alerts to notify you of significant events or deviations from expected behavior. Debugging tools utilize debugging tools to inspect the agent's state, actions, and rewards at different points in time.

Common techniques for real-time monitoring include tensorboard, custom dashboards, logging frameworks, and remote access. TensorBoard is a popular visualization tool for tracking various metrics and visualizing neural network models. Custom dashboards create custom dashboards using libraries like Plotly or Matplotlib to visualize specific metrics and visualizations. Logging frameworks use logging frameworks like Python's built-in logging module or specialized RL logging libraries to record data for later analysis. Remote access sets up remote access to the training environment to monitor the agent's performance from anywhere.

There are multiple benefits of using real-time monitoring as the online performance evaluation method. Early Detection of Issues: Identify problems such as divergence, instability, or suboptimal performance before they become significant. Improved Learning Efficiency: Make adjustments to the learning process based on real-time feedback, leading to faster convergence and better performance. Enhanced Understanding: Gain a deeper understanding of the agent's behavior and decision-making process. Facilitated Debugging: Use monitoring tools to pinpoint and resolve issues more efficiently.

By effectively implementing real-time monitoring, you can gain valuable insights into your RL agent's performance, identify and address issues promptly, and ultimately improve the overall learning process. To implement an effective real-time monitoring module, we must choose relevant metrics. That is to select metrics that are most informative for your specific reinforcement learning task but not all general metrics. Furthermore, visionalization should be designed properly so that the metric

plots and drawings are easy to understand and interpret. Alerts must be configured for critical events or deviations from expected behavior, so that abnormal performance can be notified and diagnized in a timely manner. Important data must be logged informatively, so that they can be used for debugging and analysis. More importantly, the logged data can be informative feedback and be used for performance improvements. Finally, debugging tools, which usually enable human interruption and investigation to some extent, can be useful to inspect the real-time performance of the system. Thus, it's useful helpful to leverage debugging tools to inspect the agent's state and behavior.

### 4.3.2 A/B Testing

A/B testing is a powerful technique used to compare the performance of different reinforcement learning agents or algorithms. By simultaneously running multiple agents on the same environment and evaluating their performance, A/B testing can help you identify the most effective approach for a given task.

Generally, there are five key steps in A/B testing for reinforcement learning: define the experiment, set up the environment, run the agents, collect data, and analyze results. A/B testing starts with defining the experiment. It clearly specifies the goal of the experiment, the RL algorithms or agents to be compared, and the metrics that will be used to evaluate performance. Next, we need to set Up the Environment. That is to create a controlled environment where both agents can interact and learn. This environment should be consistent to ensure a fair comparison. Then, we can run the agents to allow both agents to train and interact with the environment for a specified duration. During or after the testing run, we gather data on the performance of each agent, including metrics like reward, success rate, episode length, and learning curve. Analysis of results is carried out to compare the performance of the agents based on the collected data. It uses statistical analysis to determine if the differences are significant. Finally, based on the results, we iterate on the experiment by modifying the agents or algorithms and repeating the process.

With A/B testing, we can accurately identify the most effective RL algorithm or agent for a given task. This improved performance may not available for offline performance evaluation because of its limited access to online data that serve customers in a live way. Besides performance evaluation, A/B testing reduces the risk of deploying an underperforming agent by evaluating multiple options. It enables informed decision making about RL agent design and implementation based on empirical evidence. Iterative A/B testing can also accelerate the development process by providing insights into what works and what doesn't. By effectively implementing A/B testing, you can gain valuable insights into the performance of your RL agents and make data-driven decisions to optimize their development and deployment.

Obtaining effective performance evaluation from A/B testing faces several challenges and considerations. Firstly, Ensure that the differences in performance observed between the agents are statistically significant. We can use hypothesis testing

or confidence intervals to assess significance. It's also important to determine the appropriate duration for the experiment based on the complexity of the task and the desired level of confidence in the results. Furthermore, A/B testing must be run long enough to collect sufficient data to ensure reliable and accurate comparisons. Randomization usually help ensure the effectiveness of performance evaluation. The experiments randomly assign agents to different environments or starting conditions to minimize bias. Control on influential variables can make a difference in metric quality. A/B testing should set up proper control for variables that could affect performance, such as random seeds or hardware differences. Finally, if the agents are being evaluated in a real-world setting, the design and implementation usually need take into account related ethical implications and potential risks .

### 4.3.3 Testing Interleaving

Testing Interleaving in reinforcement learning involves evaluating the effectiveness of this technique in improving the agent's performance and generalization. Some of testing interleaving techniques overlap with those of A/B testing, but we can see that the main goals of the two online performance evaluation methods are very different. We list four common approaches to testing interleaving.

- Direct Comparison
  - Control Group: Train a baseline agent without interleaving on the same tasks or environments.
  - Experimental Group: Train an agent with interleaving on the same tasks or environments.
  - Performance Metrics: Compare the performance of both agents using relevant metrics such as reward, success rate, episode length, and learning curve.
- Generalization Testing
  - Novel Tasks: Evaluate the agent's ability to perform well on new tasks that were not seen during training.
  - Transfer Learning: Assess how well the agent can transfer knowledge from previously learned tasks to new tasks.
  - Domain Randomization: Test the agent's robustness to variations in the environment by randomly changing task parameters or conditions.
- Ablation Studies
  - Varying Interleaving Frequency: Experiment with different frequencies of task switching to determine the optimal interleaving rate.
  - Task Similarity: Evaluate the impact of task similarity on interleaving effectiveness.
  - Curriculum Learning: Compare the performance of interleaving with curriculum learning, where tasks are gradually introduced in increasing difficulty.

- Visualization
  - Learning Curves: Plot learning curves for both the interleaved and non-interleaved agents to compare their convergence rates.
  - Policy Visualization: Visualize the agent's learned policies to understand how interleaving affects their structure and generalization.
  - Task Representation: Analyze how interleaving affects the agent's representation of different tasks.

To design and implement Testing Interleaving effectively and efficiently, several considerations should be taken into account: statistical significant, experimental design, data collection, and ethical considerations. Firstly, we need to ensure that the observed differences in performance between the interleaved and non-interleaved agents are statistically significant. Hypothesis testing and confidence intervals are frequently used to assess significance. The experiments should be carefully designed to control for variables that could affect performance, such as random seeds or hardware differences. Furthermore, data sufficient is also important to ensure reliable and accurate comparisons. Finally, ethical considerations may become critical if the agents are being evaluated in a real-world setting. In such cases, it's usually beneficial to consider ethical implications and potential risks especially for large-scale or long-term testings.

By conducting thorough testing and analysis, we can gain valuable insights into the effectiveness of interleaving in reinforcement learning applications and make informed decisions about its implementation.

## 4.4 Performance Emulator

Performance simulator or emulator is frequently used for performance evaluation. These simulator and emulator mimic the real environments and simulate the behavior of a reinforcement learning agent in various environments, allowing for the evaluation of its performance without the need for real-world interaction. This is particularly useful when the real-world environment is complex, expensive, or dangerous to interact with. Different from offline performance evaluation methods which may also generate trajectories offline for testing or evaluation purposes, the design and implementation of performance emulators often need more effort and they are used for performance evealuation in an online manner.

The key components of a performance emulator include environment model, agent simulator, and performance metrics. An environment model is a representation of the real-world environment, including its dynamics, states, actions, and rewards. The agent simulator is a system that can execute the RL agent's policy within the simulated environment. The performance metrics are a set of metrics to evaluate the agent's performance, such as reward, success rate, episode length, and learning curve.

Generally, there are three types of performance emulators: physical-based emulators, rule-based emulators, and data-driven emulators. Physics-based emulators use physics engines to simulate the dynamics of real-world objects and environments. Rule-based emulators use predefined rules or models to simulate the behavior of the environment. Data-driven emulators use historical data or machine learning techniques to learn the dynamics of the environment.

The benefits of using a performance emulator over traditional offline and online performance evaluation methods include cost-effective, safety, controllablity, and scalability. Firstly, emulators can significantly reduce the cost of experimentation by eliminating the need for physical hardware or real-world resources. And emulators can be used to test agents in dangerous or risky scenarios without putting humans or equipment at risk. Furthermore, emulators allow for precise control over the environment, making it easier to isolate variables and conduct controlled experiments. Finally, emulators can be scaled to simulate large-scale environments or multiple agents simultaneously. This is usually infeasible if offline the performance evaluation method is used, and it's cost inefficient to use the online performance evaluation method in such cases due to the large resource consumption.

While performance emulators are applied in different application areas of reinforcement learning, they are used more frequently in areas where the accurate simulation of the reinforcement learnign system is possible and much more cost-efficient thatn running the real-world systems. Example of such areas include Robot, Autonomous Vehicle, Game AI, and Healthcare. In robot industry, robot simulation is used to simulate the behavior of robots in various environments to test their control algorithms and safety measures. Autonomous vehicle testing evaluates the performance of self-driving cars in different driving scenarios without putting humans at risk. In game industry, game AI emulators develop and test AI agents for video games in simulated environments. In healthcare industry, healthcare emulator simulates medical procedures or patient interactions to train AI agents for healthcare tasks. By using a performance emulator, RL researchers and engineers can efficiently evaluate the performance of their agents in a controlled and scalable environment, accelerating the development of intelligent systems.

Fig. **??** shows example performance simulation results using game emulators. On each game, there is a wide range of learnign rates with random initializations. Such performance evaluation is simply not feasible and at best cost-inefficient if feasible when offline and online performance evaluation methods are used. Compared to offline and online performance evaluation methods, evaluation through simulators or emulators can usually obtain more smoothed and stable performance metrics due to its flexible parameter adjustiment and cost-efficiency.

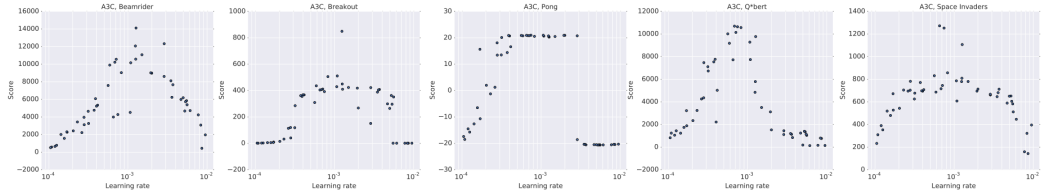


Fig. 4.1: Sample plots of scores obtained by A3C through game emulators of five games (Beamrider, Breakout, Pong, Q*bert, Space Invaders) for 50 different learning rates and random initialization. The results demonstrate the robustness of A3C to different learning rates and initial random weights [33].