Lecture 5

Dynamic Programming and Operator Theory

Goals of this Lecture

1. Understand the limitations of planning methods that rely on full environment models.
2. Introduce the multi-armed bandit problem as a simplified framework for understanding exploration–exploitation tradeoffs.
3. Define regret and evaluate exploration strategies including ε-greedy, UCB, and Thompson Sampling.
4. Learn how to estimate value functions via Monte Carlo methods using sampled trajectories.
5. Develop model-free policy improvement techniques using Monte Carlo control and GLIE strategies.

5.1 Policy Iteration

Overview. Building on our earlier discussions of policy evaluation (Lecture 4) and policy improvement (Lecture 3), we now combine these two ingredients into a complete algorithm for computing an optimal policy. This method is known as *policy iteration.*

Algorithm. Let π_0 be any initial Markov policy. The policy iteration algorithm proceeds by alternating policy evaluation and policy improvement:

- For $k = 0, 1, 2, \ldots$, repeat:
    1. **Policy Evaluation:** Compute v^{π_k}, the unique fixed point of the Bellman operator T_{π_k}, satisfying:
    $$v^{\pi_k} = T_{\pi_k} v^{\pi_k}.$$
    2. **Policy Improvement:** Define a new policy π_{k+1} by acting greedily with respect to v^{π_k}:
    $$\pi_{k+1}(s) \in \arg\max_{a \in \mathcal{A}} \sum_{s',r} p(s', r \mid s, a) \ \left[r + \gamma v^{\pi_k}(s')\right].$$
- **Termination:** Stop when $\pi_{k+1} = \pi_k$; at this point, the policy is optimal.

Preliminary: Proof of Policy Improvement via Operator Theory. Before proving the convergence of the policy iteration algorithm, we revisit the *policy improvement theorem* and present a formal proof using the operator-theoretic perspective. This highlights how monotonicity and contraction properties underlie the guarantee that greedy updates improve or preserve performance.

Theorem 5.1 (Policy Improvement)**.** *Let π be a Markov policy, and define a new policy π' to be greedy with respect to the action-value function q^π, i.e.,*

$$\pi'(s) \in \arg\max_{a \in \mathcal{A}} q^\pi(s,a) \quad \textit{for all } s \in \mathcal{S}.$$

Then the new policy π' satisfies:

$$v^{\pi'}(s) \geq v^\pi(s), \quad \textit{for all } s \in \mathcal{S}.$$

Moreover, if $q^\pi(s, \pi'(s)) > v^\pi(s)$ for some state s, then $v^{\pi'}(s) > v^\pi(s)$, i.e., π' is strictly better than π.

Proof. Let $T_{\pi'}$ denote the Bellman operator associated with the policy π', and consider the value function v^π under the current policy π. For any $s \in \mathcal{S}$, the greediness of π' with respect to q^π implies:

$$[T_{\pi'} v^\pi](s) = \mathbb{E}\left[R_{t+1} + \gamma v^\pi(S_{t+1}) \mid S_t = s, A_t = \pi'(s)\right] = q^\pi(s, \pi'(s)) \geq v^\pi(s),$$

where the last inequality uses the identity $v^\pi(s) = \sum_a \pi(a \mid s) q^\pi(s,a)$ and the fact that the maximum over a is at least the expected value.

Thus, we have:

$$T_{\pi'} v^\pi \geq v^\pi.$$

Now apply the monotonicity of $T_{\pi'}$ repeatedly:

$$v^\pi \leq T_{\pi'} v^\pi \leq T_{\pi'}^2 v^\pi \leq \cdots \leq \lim_{k \to \infty} T_{\pi'}^k v^\pi = v^{\pi'},$$

where the last equality follows from the Banach fixed-point theorem, since $T_{\pi'}$ is a γ-contraction and thus has a unique fixed point $v^{\pi'}$.

Therefore,

$$v^{\pi'}(s) \geq v^\pi(s) \quad \text{for all } s.$$

Finally, if $q^\pi(s, \pi'(s)) > v^\pi(s)$ for some s, then:

$$[T_{\pi'} v^\pi](s) > v^\pi(s),$$

which implies that the inequality remains strict in future iterates of $T_{\pi'}$, and hence:

$$v^{\pi'}(s) > v^\pi(s).$$

This completes the proof. □

From Local Improvement to Global Optimality. The policy improvement theorem guarantees that greedy updates yield non-decreasing value functions. We now show how iteratively applying this principle—by alternating between policy evaluation and greedy improvement—leads to global convergence. This is formalized in the following theorem.

Theorem 5.2 (Convergence and Optimality of Policy Iteration)**.** *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P)$ be a finite Markov decision process with discount factor $\gamma \in [0, 1)$, and let $\pi_0 \in \Pi_M$ be any initial Markov policy. Consider the sequence $\{\pi_k\}_{k\geq 0}$ produced by the policy iteration algorithm. Then the following holds:*

1. *(**Monotonic Improvement**) The value functions improve monotonically:*

$$v^{\pi_0}(s) \leq v^{\pi_1}(s) \leq v^{\pi_2}(s) \leq \cdots \quad \text{for all } s \in \mathcal{S}.$$

2. *(**Finite Termination**) The algorithm terminates in at most $|\Pi_D| = |\mathcal{A}|^{|\mathcal{S}|}$ steps, where Π_D is the set of deterministic Markov policies. In particular, termination occurs after finitely many steps.*
3. *(**Optimality at Termination**) When the algorithm terminates with $\pi_{k+1} = \pi_k$, the final policy π_k is optimal:*

$$v^{\pi_k} = v^*, \quad \text{and} \quad \pi_k \in \arg\max_{\pi \in \Pi_M} v^{\pi}.$$

Proof. Let (π_k) be the sequence of policies generated by policy iteration. At each step, we perform:

- **Policy Evaluation:** Compute v^{π_k}, the unique fixed point of the Bellman operator T_{π_k}.
- **Policy Improvement:** Define

$$\pi_{k+1}(s) \in \arg\max_{a\in\mathcal{A}} q^{\pi_k}(s, a) = \arg\max_{a\in\mathcal{A}} \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v^{\pi_k}(s')\right].$$

(1) Monotonic Improvement. From the policy improvement theorem, we have:

$$v^{\pi_{k+1}}(s) \geq v^{\pi_k}(s) \quad \text{for all } s \in \mathcal{S}.$$

If $\pi_{k+1} \neq \pi_k$, then there exists at least one state s such that

$$q^{\pi_k}(s, \pi_{k+1}(s)) > v^{\pi_k}(s),$$

which implies strict improvement at that state:

$$v^{\pi_{k+1}}(s) > v^{\pi_k}(s).$$

Hence, the sequence v^{π_k} is monotonically non-decreasing and strictly increasing unless the policy stabilizes.

(2) Finite Termination. Because the number of deterministic stationary policies is finite ($|\Pi_D| = |\mathcal{A}|^{|\mathcal{S}|}$), and each iteration either improves the policy or stops, the algorithm does not visits the same policy twice and at most can visit $|\Pi_D|$ distinct policies. Therefore, it must terminate in finitely many steps.

(3) Optimality at Termination. Suppose the algorithm terminates at some step k, so $\pi_{k+1} = \pi_k$. Then by definition of the improvement step:

$$\pi_k(s) \in \arg\max_{a} \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v^{\pi_k}(s')\right],$$

which means π_k is greedy with respect to v^{π_k}. Applying the Bellman optimality principle (Theorem 3.4), it follows that π_k is optimal:

$$v^{\pi_k} = T_* v^{\pi_k} \quad \Longrightarrow \quad v^{\pi_k} = v^*.$$

□

Modified Policy Iteration. Modified Policy Iteration (MPI) interpolates between value iteration and policy iteration. At each iteration, instead of fully solving for the value function v^{π_k} as in standard policy iteration, MPI performs a limited number of Bellman updates to approximate it. This results in lower computational cost per iteration while preserving convergence to the optimal policy.

Modified Policy Iteration (MPI) Algorithm

- Initialize an arbitrary Markov policy π_0 and initial value v_0.
- For $k = 0, 1, 2, \dots$ until convergence:
  1. **Partial Policy Evaluation:** Set $v_0^{(k)} := v_k$ and perform m iterations of value iteration for the current policy π_k:

     $$v_{j+1}^{(k)} := T_{\pi_k} v_j^{(k)}, \quad j = 0, \dots, m-1.$$

     Let $v_{k+1} := v_m^{(k)}$ be the result of the approximate evaluation.
  2. **Policy Improvement:** Define a new policy π_{k+1} that is greedy with respect to v_{k+1}:

     $$\pi_{k+1}(s) \in \arg\max_{a \in \mathcal{A}} \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_{k+1}(s') \right].$$

Interpretation and Special Cases.

- When $m = 1$, MPI reduces to *value iteration*, where one Bellman update is used per step.
- When $m = \infty$, MPI becomes *policy iteration*, with exact policy evaluation.

Theorem 5.3 (Convergence of Modified Policy Iteration)**.** *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P)$ be a finite Markov decision process with discount factor $\gamma \in [0, 1)$. Let (v_k, π_k) be the sequence generated by modified policy iteration (MPI), where:*

- *$\pi_k \in Greedy(v_k)$, i.e., $\pi_k(s) \in \arg\max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_k(s')]$,*
- *$v_{k+1} := T_{\pi_k}^m v_k$, for some fixed $m \geq 1$, where T_{π_k} is the Bellman operator for π_k.*

Then:

1. *The sequence v_k converges to the optimal value function v^*,*
2. *The corresponding policies π_k converge in finitely many steps to an optimal policy π^*,*
3. *The convergence rate of v_k to v^* is geometric in the ℓ_∞ norm.*

Remarks. A full convergence proof for Modified Policy Iteration is significantly more involved than for standard policy or value iteration, and is omitted here. However, it can be shown that the fixed point of MPI corresponds to the optimal value function. In particular, if at some iteration we have $v_k = T_* v_k$, then $v_k = v^*$ and the corresponding greedy policy is optimal. This confirms that MPI correctly converges to the optimal solution in finite MDPs, and that any termination point implies optimality.

Use in Practice. MPI is often favored when solving large-scale MDPs where full policy evaluation is expensive, and a few evaluation steps offer a reasonable tradeoff between accuracy and speed. It forms the basis of many approximate dynamic programming and reinforcement learning algorithms.

Discussion. Policy iteration is exact and efficient for small MDPs but can be computationally intensive when:

- The state space is large (value function evaluation is costly),
- The model $p(s', r \mid s, a)$ is not known or cannot be easily queried.

In the next sections, we address these limitations via:

- *Modified Policy Iteration (MPI)*—a compromise between full evaluation and pure value iteration,
- *Sample-based methods* such as Monte Carlo and Temporal Difference learning.

5.2 From Planning to Learning

Limitations of Dynamic Programming. The algorithms we have studied so far—such as policy iteration, value iteration, and modified policy iteration—belong to the class of *planning* methods. These approaches assume full knowledge of the MDP model, including the transition dynamics $p(s', r \mid s, a)$. While powerful and foundational, this assumption is unrealistic in many practical scenarios. In real-world applications, the agent often interacts with an unknown environment, without access to transition probabilities or reward distributions. This renders dynamic programming infeasible, as computing Bellman updates requires expectations over unknown quantities.

The Need for Sampling-Based Approaches. To address the lack of a known model, we must turn to *model-free* methods. These approaches allow the agent to learn optimal behavior purely from experience—by collecting samples through interaction with the environment, rather than querying the true model. In this setting, the agent observes sequences of transitions $(S_t, A_t, R_{t+1}, S_{t+1})$ and uses these samples to estimate value functions or improve policies.

Overview of Model-Free Methods. Model-free reinforcement learning methods can be broadly classified into two categories:

- **Value-based methods:** These aim to estimate value functions (e.g., v^π or q^π) from data and use them to derive improved policies. Classical examples include Monte Carlo methods and temporal-difference learning (TD, TD(λ), Q-learning).
- **Policy-based methods:** These directly parametrize and optimize policies without explicitly computing value functions. Examples include REINFORCE, actor-critic methods, and policy gradient algorithms.

These approaches form the foundation of modern reinforcement learning, enabling learning from data in complex, high-dimensional, or stochastic environments. In the remainder of the course, we transition from planning-based techniques to these sample-based learning algorithms.

5.3 Multi-Armed Bandits

Problem Setup. We now move toward learning to make decisions under uncertainty. As a stepping stone toward full reinforcement learning, we begin with a simplified setting where the environment has no state. This setting is known as the *multi-armed bandit* problem.

At each round $t = 1, 2, \ldots$, the agent selects an action $A_t \in \mathcal{A} = \{1, \ldots, K\}$, and the environment responds with a random reward $R_t \in \mathbb{R}$, drawn from an unknown distribution $p(r \mid a)$ associated with arm $a = A_t$. The agent receives no additional feedback (e.g., no state transitions or next observations). The goal is to select actions to maximize long-term reward.

Policies and Value Functions. A stochastic policy π_t assigns probabilities over actions:

$$\pi_t(a) := \mathbb{P}(A_t = a).$$

The *action-value function* is defined as the expected reward of pulling arm a:

$$q(a) := \mathbb{E}[R_t \mid A_t = a] = \int_{\mathbb{R}} r \cdot p(r \mid a)\, dr,$$

assuming $p(r \mid a)$ admits a density, or more generally, as a discrete expectation when rewards are bounded. Importantly, $q(a)$ depends only on the environment and not on the policy.

The value of a policy π is the expected reward when actions are selected according to π:

$$v_\pi := \sum_{a \in \mathcal{A}} \pi(a)\, q(a).$$

Optimal Value and Regret. The optimal value is the reward of the best arm:

$$v^* = \max_{a \in \mathcal{A}} q(a).$$

We define the expected *regret* at time t as the difference between the optimal value and the expected reward at that round:

$$\ell_t := v^* - \mathbb{E}[q(A_t)].$$

The total expected regret over T rounds is:

$$\text{Regret}(T) := \sum_{t=1}^{T} \ell_t = Tv^* - \mathbb{E}\left[\sum_{t=1}^{T} q(A_t)\right].$$

Decomposition of Regret. Let $\Delta_a := v^* - q(a)$ denote the suboptimality gap for arm a. Define $N_T(a)$ as the number of times arm a was pulled up to time T. Then:

$$\text{Regret}(T) = \sum_{a \in \mathcal{A}} \Delta_a \cdot \mathbb{E}[N_T(a)],$$

or equivalently, using $\bar{N}_T(a) := \mathbb{E}[N_T(a)]$,

$$\text{Regret}(T) = \sum_{a \in \mathcal{A}} \Delta_a \cdot \bar{N}_T(a).$$

This decomposition shows that regret accumulates in proportion to how often suboptimal arms are selected.

Key Insight. The challenge in bandit problems is that the suboptimality gaps Δ_a are unknown. The agent must explore to estimate $q(a)$, but also exploit known information to maximize reward. A well-designed algorithm minimizes regret by concentrating pulls on the best arms, while still exploring sufficiently to identify them.

5.4 Exploration Strategies

We now describe three foundational strategies for managing the exploration–exploitation tradeoff in multi-armed bandit problems: ε-greedy, Upper Confidence Bound (UCB), and Thompson Sampling.

ε-Greedy. The ε-greedy strategy maintains empirical estimates $\hat{q}_a(t)$ of the expected reward for each arm a, based on observed samples:

$$\hat{q}_a(t) := \frac{1}{N_a(t)} \sum_{k=1}^{t-1} \mathbb{1}\{A_k = a\} \cdot R_k,$$

where $N_a(t)$ is the number of times arm a has been selected up to round t. At each time step:

- With probability $1 - \varepsilon$: select the empirically best arm:

$$A_t = \arg\max_{a \in \mathcal{A}} \hat{q}_a(t),$$

- With probability ε: select an arm uniformly at random.

The parameter $\varepsilon \in (0, 1)$ determines the exploration rate. To reduce regret, it is often annealed over time, e.g., $\varepsilon_t = 1/\sqrt{t}$, to emphasize exploration early and exploitation later.

Upper Confidence Bound (UCB). UCB is a frequentist algorithm based on the principle of optimism in the face of uncertainty. It augments the empirical estimate $\hat{q}_a(t)$ with a confidence bonus that decreases with $N_a(t)$, the number of times arm a has been selected:

$$A_t = \arg\max_{a \in \mathcal{A}} \left[\hat{q}_a(t) + c \cdot \sqrt{\frac{\log t}{N_a(t)}} \right],$$

where $c > 0$ is a tunable parameter controlling the degree of optimism. The $\log t / N_a(t)$ term encourages early exploration of all arms and gradually shifts toward exploitation as uncertainty decreases.

Thompson Sampling. Thompson Sampling is a Bayesian exploration strategy that maintains a posterior distribution over each arm's expected reward. At each round t, the agent proceeds as follows:

1. For each arm $a \in \mathcal{A}$, sample $\theta_a(t)$ from the posterior distribution over $q(a)$,
2. Select the arm with the highest sampled value:
$$A_t = \arg\max_{a \in \mathcal{A}} \theta_a(t).$$

The sampled value $\theta_a(t)$ represents a plausible estimate of the expected reward $q(a)$, drawn from the agent's current belief. This randomization induces structured exploration: arms with greater uncertainty (i.e., wider posterior distributions) are more likely to be selected, even if their current empirical mean is lower.

Example: Bernoulli Bandits. In the Bernoulli bandit setting, each arm yields binary rewards $R_t \in \{0, 1\}$, with unknown mean $q(a) = \mathbb{E}[R_t \mid A_t = a]$. The agent models its uncertainty about each arm's mean reward $q(a)$ using a Bayesian approach. Specifically, the agent treats $q(a)$ as a random variable with a prior distribution:

$$q(a) \sim \mathrm{Beta}(\alpha_a, \beta_a),$$

where $\alpha_a > 0$ and $\beta_a > 0$ are parameters that encode the agent's belief about how likely the arm is to yield rewards of 1 or 0, respectively.

- The Beta distribution is a natural prior for Bernoulli outcomes because it is the *conjugate prior*—the posterior remains a Beta distribution after observing data.
- Initially, we use the uniform prior $\mathrm{Beta}(1, 1)$, which represents no prior preference for 0 or 1 outcomes.
- Each time arm a is pulled and a reward $R_t \in \{0, 1\}$ is observed, the agent updates the posterior parameters:
$$\alpha_a \leftarrow \alpha_a + R_t, \qquad \beta_a \leftarrow \beta_a + (1 - R_t).$$
That is:
  - If $R_t = 1$, we increment α_a, reinforcing the belief that arm a tends to yield reward 1,
  - If $R_t = 0$, we increment β_a, reinforcing the belief that arm a tends to yield reward 0.
- After the update, the posterior over $q(a)$ becomes $\mathrm{Beta}(\alpha_a, \beta_a)$. To choose the next action, we:
  1. Sample a mean reward estimate:
  $$\theta_a(t) \sim \mathrm{Beta}(\alpha_a, \beta_a),$$
  2. Select the arm with the highest sample:
  $$A_t = \arg\max_{a \in \mathcal{A}} \theta_a(t).$$

Interpretation. This strategy is Bayesian because it explicitly models and updates the agent's uncertainty about each arm's mean reward via a posterior distribution. The parameters α_a and β_a track the number of observed rewards and failures, respectively. Early in training, the posteriors are wide (uncertain), promoting exploration. Over time, the posterior concentrates around the empirical mean, leading to exploitation. Sampling from the posterior thus naturally balances exploration and exploitation without requiring explicit tuning of exploration parameters.

5.5 Regret Bounds and Performance Guarantees

We now examine the performance of different exploration strategies in terms of cumulative regret. Recall:

$$\text{Regret}(T) := T \cdot \mu^* - \sum_{t=1}^{T} \mathbb{E}[\mu_{A_t}],$$

where $\mu^* = \max_a \mu_a$ is the mean reward of the best arm.

Regret of ε-Greedy. With a fixed $\varepsilon > 0$, exploration occurs at a constant rate, leading to linear regret:

$$\text{Regret}(T) = O(\varepsilon T + \log T).$$

To achieve sublinear regret, ε must decay over time. For example, $\varepsilon_t = \Theta(1/\sqrt{t})$ yields regret $O(\sqrt{T \log T})$, though this requires careful tuning.

Regret of UCB. UCB achieves strong performance guarantees. Assuming bounded rewards (e.g., $r \in [0, 1]$), UCB satisfies:

$$\text{Regret}(T) = O\left(\sum_{a:\mu_a<\mu^*} \frac{\log T}{\Delta_a}\right),$$

where $\Delta_a = \mu^* - \mu_a$ is the suboptimality gap. This gap-dependent bound shows that UCB achieves near-optimal regret rates in stochastic bandits.

Regret of Thompson Sampling. Thompson Sampling also enjoys strong theoretical guarantees. For Bernoulli rewards, it achieves:

$$\text{Regret}(T) = O\left(\sum_{a:\mu_a<\mu^*} \frac{\log T}{\Delta_a}\right),$$

matching the performance of UCB up to constant factors. Recent results also establish minimax-optimal bounds under general reward distributions.

Summary.

- ε-greedy is simple but may suffer high regret unless carefully tuned.
- UCB provides principled exploration and logarithmic regret in T.
- Thompson Sampling offers a Bayesian alternative with competitive empirical and theoretical performance.
- These methods illustrate core ideas in balancing exploration with exploitation and lay the foundation for more general reinforcement learning algorithms.

5.6 Monte Carlo Prediction

Monte Carlo (MC) methods provide a simple and powerful approach to estimating value functions using sample episodes, without requiring a model of the environment. These methods are particularly useful when interacting with an environment in an episodic fashion.

Estimating v^π via Sampling. Suppose we have a policy π and wish to estimate its state-value function $v^\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$. Monte Carlo methods estimate this expectation using empirical averages from sampled episodes:

$$v^\pi(s) \approx \frac{1}{N(s)} \sum_{i=1}^{N(s)} G_t^{(i)},$$

where $G_t^{(i)}$ is the return following the i-th occurrence of state s under policy π, and $N(s)$ is the number of such occurrences.

First-Visit vs Every-Visit Estimators. There are two common variants for generating these estimates:

- **First-Visit MC:** Estimate $v^\pi(s)$ using only the first time state s is visited in each episode.
- **Every-Visit MC:** Estimate $v^\pi(s)$ using the return from *every* occurrence of state s within the episode.

Both estimators are unbiased under mild assumptions, and converge to the true value function as the number of episodes goes to infinity. However, the variance and sample efficiency may differ in practice.

Advantages and Limitations.

- Monte Carlo methods do not require knowledge of the transition probabilities or reward model.
- They apply naturally in episodic environments and are easy to implement.
- However, they require episodes to terminate and may exhibit high variance.

5.7 Monte Carlo Control

So far we have used Monte Carlo methods to evaluate a fixed policy. We now turn to the problem of finding an optimal policy through experience—using only samples collected from interaction with the environment. This setting, where we do not assume access to a model, is referred to as *model-free control.*

GLIE and ε-Greedy Policy Improvement. To learn the optimal policy, we combine Monte Carlo prediction with ε-greedy exploration. The key requirement is that the policy eventually becomes greedy while still exploring enough in the early stages. This is formalized via the notion of *GLIE* (Greedy in the Limit with Infinite Exploration):

Definition 5.1 (GLIE). *A sequence of policies (π_t) is said to satisfy the GLIE conditions if:*

- *Every action is taken infinitely often in every state:*

$$\lim_{t\to\infty} N_t(s,a) = \infty, \quad \forall (s,a),$$

- *The policy becomes greedy in the limit:*

$$\lim_{t\to\infty} \pi_t(a \mid s) = \begin{cases} 1 & \text{if } a \in \arg\max_{a'} Q_t(s,a'), \\ 0 & \text{otherwise.} \end{cases}$$

This condition is often satisfied by using an ε-greedy policy with decaying ε_t, e.g., $\varepsilon_t = 1/t$.

Algorithm: Monte Carlo Control with ε-Greedy. At each episode:

1. Generate an episode using the current ε-greedy policy π_t.
2. For each state-action pair (s, a) in the episode, compute the return G_t and update the action-value estimate $Q(s, a)$ using first-visit or every-visit Monte Carlo.
3. Update the policy to be ε_t-greedy with respect to Q.

Convergence Guarantees. The following theorem ensures that under GLIE conditions, Monte Carlo control converges to the optimal policy:

Theorem 5.4 (Convergence of Monte Carlo Control)**.** *Suppose each episode is generated by an ε-greedy policy with $\varepsilon_t \to 0$ and the resulting sequence of policies (π_t) satisfies GLIE. Then:*

$$\lim_{t\to\infty} Q_t(s, a) = q^*(s, a), \quad \textit{and} \quad \lim_{t\to\infty} \pi_t = \pi^*,$$

with probability 1, for all (s, a).

Remarks.

- Monte Carlo control requires episodic tasks with complete returns from each episode.
- GLIE ensures sufficient exploration and eventual exploitation.
- In practice, convergence can be slow; temporal-difference methods (e.g., SARSA, Q-learning) often provide faster alternatives.