Comparison of Q-Learning, Dynamic Programming, and Monte Carlo Methods in Reinforcement Learning

Aspect	Dynamic Programming (DP)	Monte Carlo (MC)	Q-Learning
Model Requirement	Requires a complete and accurate model of the environment, including transition probabilities and rewards.	Does not require a model; uses sampled episodes to estimate values.	Does not require a model; learns action-value function ($Q(s, a)$) directly through interaction with the environment.
Applicability	Suitable when the model of the environment is known.	Suitable when the model is unknown or difficult to define but interaction with the environment is possible.	Suitable for model-free environments where the model is unavailable or infeasible to compute.
Update Mechanism	Deterministic updates using Bellman equations. Updates values for all states simultaneously based on expectations.	Stochastic updates based on sampled trajectories. Updates values episodically based on observed returns.	Stochastic updates using temporal-difference (TD) learning. Updates a single $Q(s, a)$ pair at a time based on sampled rewards and the TD error.
Algorithm Type	Iterative method: combines Policy Evaluation and Policy Improvement steps until convergence (e.g., Policy Iteration).	Episodic method: relies on the aggregation of sampled returns from complete episodes (e.g., First-Visit or Every-Visit Monte Carlo).	Incremental TD method: directly learns optimal action-value function ($Q^*(s, a)$) without needing a policy evaluation step.
Exploration	Does not inherently explore; assumes a predefined policy and relies on a full model.	Explores the state space by sampling episodes generated from a behavior policy.	Requires explicit exploration strategies (e.g., ϵ-greedy or softmax) to balance exploration and exploitation.
Efficiency	Computationally efficient for small to moderate state spaces when the model is known. Simultaneous updates reduce the number of iterations.	Computationally less efficient due to reliance on sampling. Requires many episodes for convergence, especially in large state spaces.	Efficient for large or unknown state spaces due to incremental updates but sensitive to hyperparameter tuning (e.g., learning rate).
Convergence	Converges to the true value function if the model is accurate and the convergence criteria are met.	Converges to the true value function of the behavior policy as the number of episodes approaches infinity (assuming sufficient exploration).	Converges to the optimal $Q^*(s, a)$ under sufficient exploration and learning rate decay.
Variance	Low variance due to deterministic updates.	High variance due to stochastic updates from sampled episodes.	Moderate variance; variance is reduced compared to MC due to TD updates but higher than DP.

Aspect	Dynamic Programming (DP)	Monte Carlo (MC)	Q-Learning
Dependency on Episodicity	Works for both episodic and continuous tasks.	Typically designed for episodic tasks where episodes terminate.	Can handle both episodic and continuous tasks but requires exploration strategies to visit all states.
Discount Factor (γ)	Utilizes the discount factor explicitly to compute expected returns.	Can handle $\gamma = 1$ safely for episodic tasks as episodes terminate.	Utilizes the discount factor explicitly to balance immediate and future rewards.
Complexity	Scales poorly with the size of the state and action spaces due to the need to store and compute values for all states.	Scales poorly with the number of episodes required for convergence in large or complex state spaces.	Scales well for large state-action spaces due to its incremental updates but depends on sufficient exploration.
Use Case	Best suited for problems with a known model and moderate state/action spaces. Example: solving gridworld problems with a predefined MDP model.	Best suited for problems where the model is unknown, or the environment is too complex to model explicitly. Example: estimating state values from experience in a game-like environment.	Ideal for real-world applications where the model is unavailable, and interactions with the environment are required. Example: robotics, games, and online learning tasks.