Journal of Data Science 24 (1), 86–105
January 2026
DOI: 10.6339/25-JDS1205
Data Science Reviews

Reinforcement Learning: A Statistical Perspective

Ying Zhou[1,*]

[1]*Department of Statistics, University of Connecticut, Storrs, CT 06269, U.S.A.*

Abstract

Reinforcement Learning (RL) is a powerful framework for sequential decision-making, enabling agents to optimize actions through interaction with their environment. While widely studied in computer science, statisticians have advanced RL by addressing challenges like uncertainty quantification, sample efficiency, and interpretability. These contributions are particularly impactful in healthcare, where RL complements Dynamic Treatment Regimes (DTRs), optimizing personalized medicine by tailoring treatments to individuals based on evolving characteristics. This paper serves as both a tutorial for statisticians new to RL and a review of its integration with statistical methodologies. It introduces foundational RL concepts, classical algorithms, and Q-learning variants, and highlights how statistical perspectives, especially causal inference, address challenges in DTRs. By bridging RL and statistical perspectives, the paper highlights opportunities to enhance decision-making in high-stakes domains like healthcare.

Keywords *causal inference; dynamic treatment regimes; sequential decision-making*

1 Introduction

Reinforcement Learning (RL) has emerged as a powerful paradigm for sequential decision-making, with applications that have transformed diverse domains. In artificial intelligence, RL gained worldwide attention through AlphaGo, which defeated human world champions in the ancient board game Go (Silver et al., 2016). In robotics, RL enables complex motor control and adaptive behaviors (Kober et al., 2013), while in autonomous driving it has been applied to train end-to-end driving policies (Kendall et al., 2019). In industry, RL supports large-scale recommendation systems such as YouTube (Covington et al., 2016) and has been applied to optimize real-time bidding strategies in online advertising (Cai et al., 2017). In healthcare, RL has been applied to guide treatment strategies in intensive care (Komorowski et al., 2018) and to develop principled approaches for Dynamic Treatment Regimes (DTRs) (Laber et al., 2014). These successes demonstrate RL's broad impact and underscore the importance of continued methodological development to ensure reliability in high-stakes applications.

While RL has been extensively studied within computer science and artificial intelligence, statisticians have played a pivotal role in advancing the field by addressing its theoretical and methodological challenges. Statistical perspectives enhance RL by introducing rigor in uncertainty quantification, improving sample efficiency, strengthening robustness, and fostering interpretability in decision-making processes (Murphy, 2003; Zeng et al., 2025). For example, causal inference methods can be integrated into RL to improve its interpretability and applicability. This integration is especially valuable in domains like healthcare (Komorowski et al., 2018; Luck-

* Email: yzhou@uconn.edu.

© 2026 The Author(s). Published by the School of Statistics and the Center for Applied Statistics, Renmin University of China. Open access article under the CC BY license.
Received January 1, 2025; Accepted October 27, 2025

ett et al., 2020; Yu et al., 2021), where RL methods can complement frameworks such as DTRs. DTRs aim to optimize sequential decision-making in personalized medicine by tailoring treatments to individuals based on their evolving characteristics and responses. Incorporating causal inference principles into RL allows researchers to better address the complexities of personalized and adaptive interventions in DTRs.

Several surveys and tutorials on RL already exist. Sutton and Barto (2018) provides a comprehensive textbook treatment from the computer science perspective; Yu et al. (2021) surveys RL applications across healthcare domains; and Zeng et al. (2025) reviews methods at the intersection of causality and RL. In the statistical literature, DTRs have provided a key entry point for introducing RL concepts in healthcare, with influential tutorials and comprehensive treatments such as Murphy (2003); Chakraborty and Moodie (2013); Laber et al. (2014). However, these works are primarily focused on healthcare applications and on DTR methodology, rather than on RL more broadly. What remains missing is a review that both introduces RL algorithms more broadly in a tutorial style while placing particular emphasis on their integration with statistical thinking and causal inference. This paper aims to fill that gap. Its purpose is two-fold: first, to provide statisticians unfamiliar with RL a tutorial-style introduction to fundamental concepts and methods; and second, to review how RL connects with statistical methodologies, with particular attention to causal inference and applications in DTRs. In doing so, the paper explores how statistical perspectives can address challenges in RL, such as non-Markovian dynamics, non-stationarity, interpretability, and data efficiency. It also discusses how RL can be tailored to sequential decision-making in high-stakes domains such as healthcare.

The remainder of the paper is organized as follows. Section 2 introduces foundational RL concepts including the Bellman equations. Section 3 reviews classical RL algorithms, and Section 4 expands on Q-learning, covering its variants and contrasting computer science and statistical perspectives. Section 5 discusses the relationship between RL and DTRs. Section 6 explores recent research on causal inference challenges in DTRs. Finally, Section 7 concludes with a discussion of ongoing challenges and open problems.

2 Core Concepts in RL

This section introduces the core components of RL and the mathematical structure underlying its framework. The time horizon T in RL defines the duration over which the agent evaluates rewards and makes decisions. This horizon can be either finite or infinite, with the choice depending on the specific problem being addressed. The key elements of RL include the agent, environment, actions a_t, rewards r_t, and states s_t (Sutton and Barto, 2018). The agent interacts with the environment in a cyclic process: it observes the current state, takes an action, and then the environment transitions to a new state while providing a reward to the agent. This iterative interaction forms the basis of the RL framework. A policy is the strategy an agent uses to decide on actions for each state in the environment. It can be thought of as a set of rules mapping states to actions. Policies are classified into two types: a deterministic policy always takes the same action for a given state, while a stochastic policy takes the action probabilistically according to a fixed distribution depending on the state. For example, in a given state, a deterministic policy may always choose action 1, while a stochastic policy might choose action 1 with 30% probability and action 2 with 70% probability.

Two primary tasks in RL are policy evaluation and policy optimization. Policy evaluation is the process of assessing the performance of a given policy by estimating its expected cumulative

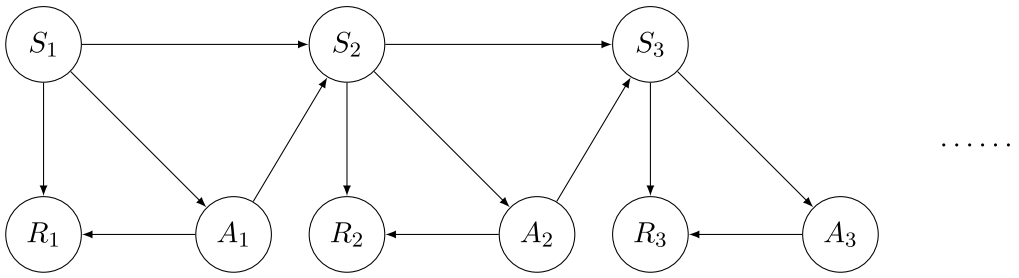

Figure 1: A simple Markov decision process (MDP).

reward, while policy optimization aims to find the policy that maximizes the expected cumulative reward. Depending on how an agent learns and updates its policy, RL can be categorized into on-policy and off-policy learning. On-policy learning involves evaluating and improving the same policy that is generating the data. This is akin to a learning-by-doing approach, where the agent refines its strategy through direct experience. While in off-policy learning, an agent can learn from data generated by a different policy. In this case, the learning policy (the target policy) is distinct from the data-generating policy (the behavior policy). This separation enables the agent to learn from past experiences or data generated by other agents. It is important not to confuse on-policy/off-policy learning with online/offline learning. Online learning refers to updating the policy in real-time as the agent interacts with the environment. Offline learning involves learning from a pre-collected dataset without further interaction with the environment. These two concepts are orthogonal, meaning they capture different distinctions. For instance, online on-policy algorithms (e.g., SARSA) update the policy while interacting with the environment; online off-policy algorithms (e.g., Q-learning) learn a policy different from the one generating the data in real-time; offline off-policy algorithms learn a policy entirely from pre-collected data; while offline on-policy algorithms are less common because learning a policy using data generated by the same policy without further interaction is counterintuitive and rarely practical.

RL often uses a framework called a Markov Decision Process (MDP) to model decision-making dynamics (Puterman, 1994). An MDP is typically defined as a tuple (S, A, P, R, γ), where S is the set of states, A is the set of actions, P denotes the transition probabilities with $P(s' \mid s, a)$ defining the likelihood of moving to the next state given the current state and action, $R(s, a)$ represents the expected immediate reward obtained when the agent takes action a in state s, and γ is the discount factor which is a value between 0 and 1 that determines the relative importance of future rewards compared to immediate ones. The Markov property, central to this model, states that the future state depends only on the current state and action, not on the sequence of past states or actions. Another important property of MDPs is stationarity, which assumes that the transition probabilities P and the reward function R are time-invariant. This means that the dynamics of the environment do not change over time, simplifying the analysis and solution of the problem. Figure 1 provides a visualization of a simple MDP. Using MDPs in RL is a widely adopted practice. While MDPs may not be suitable for every application, a point we discuss in Section 6, they remain a versatile and foundational framework for modeling and solving a wide variety of sequential decision-making problems.

To solve MDPs, the Bellman equation (Bellman, 1957) is an important tool used for both policy evaluation and policy optimization. It provides a recursive relationship for value functions by decomposing the cumulative reward into the immediate reward and the discounted expected

value of future rewards. This decomposition serves as the mathematical foundation for many RL algorithms. The Bellman equation takes different forms depending on the context. To begin, we explore the Bellman expectation equations, which are usually used for policy evaluation where the goal is to assess the value of a given policy π. For simplicity, we assume an infinite horizon in the derivation. The state-value function, denoted by $V_\pi(s)$, represents the expected cumulative reward starting from state s and following policy π. Similarly, the action-value function $Q_\pi(s,a)$ represents the expected cumulative reward starting from state s, taking action a, and then following policy π. These functions are defined as

$$V_\pi(s) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1} \mid S_t = s\right], \qquad Q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right],$$

where $\mathbb{E}_\pi$ denotes the expectation under the policy π, R_t, S_t, and A_t denote the reward, state, and action at time t. The above definitions can be rewritten recursively as

$$\begin{aligned}
\text{(state-value function) } V_\pi(s) &= \mathbb{E}_\pi\{R_{t+1} + \gamma V_\pi(S_{t+1}) \mid S_t = s\} \\
&= \sum_{a\in A}\pi(a \mid s)\{R(s,a) + \gamma\sum_{s'\in S}P(s' \mid s,a)V_\pi(s')\}, \qquad (2.1)\\
\text{(action-value function) } Q_\pi(s,a) &= \mathbb{E}_\pi\{R_{t+1} + \gamma Q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a)\} \qquad (2.2)\\
&= R(s,a) + \gamma\sum_{s'\in S}P(s' \mid s,a)\sum_{a'\in A}\pi(a' \mid s')Q_\pi(s',a').
\end{aligned}$$

These are known as the Bellman expectation equations.

For policy optimization, where the goal is to find the policy that maximizes cumulative rewards, the following Bellman optimality equations are often used to compute the optimal value functions:

$$\text{(state-value function) } V_*(s) = \max_{a\in A}\{R(s,a) + \gamma\sum_{s'\in S}P(s' \mid s,a)V_*(s')\}, \qquad (2.3)$$

$$\text{(action-value function) } Q_*(s,a) = R(s,a) + \gamma\sum_{s'\in S}P(s' \mid s,a)\max_{a'\in A}Q_*(s',a'). \qquad (2.4)$$

Here, $V_*(s)$ denotes the maximum cumulative reward achievable from state s by following an optimal policy, and $Q_*(s,a)$ denotes the maximum cumulative reward achievable from state s by taking action a and then following an optimal policy.

The Bellman equation is a cornerstone of RL algorithms, which use it to iteratively estimate value functions and guide learning. Different algorithms leverage the Bellman equation in distinct ways. For instance, value iteration directly applies the Bellman optimality equation to optimize the value function, without explicitly storing or evaluating a specific policy. In contrast, policy iteration alternates between using the Bellman expectation equation to evaluate the value function of a given policy and improving the policy based on these evaluations. Another example is Q-learning, which relies on the Bellman optimality equation for action-value functions to learn optimal Q-values (Watkins and Dayan, 1992). These Q-values implicitly define the optimal policy by selecting the action with the maximum value for each state. While the details of these algorithms differ, their shared reliance on the Bellman equation underscores its central role in RL.

3 Classical Algorithms in RL

3.1 Dynamic Programming

If the optimal action-value function $Q_*(s,a)$ or state-value function $V_*(s)$ is known, determining the optimal policy becomes straightforward. Specifically, the optimal action for a given state s is the one that maximizes $Q_*(s,a)$ or $\sum_{s'} P(s' \mid s,a)V_*(s')$. When the environment is fully known, meaning the transition probabilities $P(s' \mid s,a)$ and rewards $R(s,a)$ are available, these optimal functions can be computed using Dynamic Programming (DP) methods.

DP is a powerful tool for policy optimization in RL (Sutton and Barto, 2018). It solves the problem by leveraging the Bellman equation. Two widely used DP methods in RL are value iteration and policy iteration, both of which iteratively compute optimal value functions and policies. In value iteration, we begin by initializing the value function for all states to an arbitrary value such as 0. Then we repeatedly update the value function for each state using the Bellman optimality equation (2.3) until convergence. Policy iteration follows a slightly more complex procedure, alternating between two steps: policy evaluation and policy improvement. In the policy evaluation step, the value function $V_\pi(s)$ for the current policy π is calculated using the Bellman expectation equation (2.1). In the policy improvement step, the policy is updated by selecting actions that greedily maximize the value function: $\pi(s) = \arg\max_a\{R(s,a) + \gamma \sum_{s'} P(s' \mid s,a)V_\pi(s')\}$. These two steps alternate iteratively until the policy converges to the optimal one. The combination of policy evaluation and improvement exemplifies the Generalized Policy Iteration (GPI) paradigm, which underlies many RL algorithms. In addition to policy optimization, DP can also be used for evaluating given policies. The Bellman expectation equation (2.1), rather than the Bellman optimality equation (2.3), is used iteratively to estimate $V_\pi(s)$ until convergence. If the initial state distribution $d_0(s)$ is known, the overall value of the policy can be computed as $\sum_s d_0(s)V_\pi(s)$.

While DP algorithms have a solid mathematical foundation and are well-understood, their practicality diminishes in large or complex environments due to two main challenges: the curse of dimensionality and the model-based requirement. As the state and action spaces grow, the computational and memory requirements increase exponentially, making it infeasible to compute and store value functions for all states. Moreover, DP methods assume that the transition probabilities and rewards are fully known, which is often not the case in real-world RL problems. For these reasons, while DP is a crucial component of the RL toolkit, it is generally more applicable to problems with small, structured environments and serves as a theoretical foundation for developing more scalable algorithms (Bertsekas, 2017).

3.2 Monte Carlo Methods

Monte Carlo (MC) methods differ fundamentally from DP methods in that they do not require a model of the environment's dynamics, such as transition probabilities or reward functions. Instead, they are fully model-free and learn value functions directly from sampled experience. MC methods estimate values by averaging returns observed in complete episodes of interaction. An episode consists of a sequence of states, actions, and rewards, starting from an initial state and ending in a terminal state. MC methods require complete episodes before making updates, as the total return is only known at the end of the episode.

The core idea is to approximate the value of a state or state–action pair by the empirical mean of the returns following that state or action. Two common variants of MC methods are First-Visit MC and Every-Visit MC. First-Visit MC considers only the first time a state is

visited in each episode when computing the average return, while Every-Visit MC averages the returns observed after every occurrence of a state within an episode. Both approaches converge to the true expected return as the number of episodes approaches infinity, due to the law of large numbers.

The update rules for MC methods are straightforward and apply to both state-value and action-value functions. For a state s_t or a state-action pair (s_t, a_t), the update rules are

$$\begin{aligned} V(s_t) &\leftarrow V(s_t) + \alpha(G - V(s_t)), \\ Q(s_t, a_t) &\leftarrow Q(s_t, a_t) + \alpha(G - Q(s_t, a_t)), \end{aligned} \tag{3.1}$$

where G is the total return from time t onward, and α is the step-size parameter. These formulas can be interpreted as incremental updates to the value estimates, where G is weighted against the current estimate. The right-hand side is effectively a weighted average of the current value and the observed return. Sometimes, the value function $V(s_t)$ is explicitly written as the average of n returns. This is equivalent to the above update rule when $\alpha = 1/n$ at the n-th visit. However, the advantage of the incremental update rule is that it does not require storing all returns, allowing the agent to update values along the way.

While MC methods are particularly effective at policy evaluation based on either state-value or action-value functions, they can also be extended for policy optimization. However, state-value-based MC policy optimization is less common because improving a policy with state values requires selecting the action that maximizes $\sum_{s'} P(s' \mid s, a)V_*(s')$, which in turn requires knowledge of the transition probabilities $P(s' \mid s, a)$. Such model information is typically unavailable in model-free settings. Consequently, action-value functions are generally preferred for MC-based policy optimization. A well-known example of an on-policy MC control algorithm is On-Policy First-Visit MC Control with ϵ-Greedy Exploration. In this approach, the agent follows an ϵ-greedy policy, which balances exploration and exploitation by selecting suboptimal actions with a small probability ϵ while predominantly choosing actions that maximize the estimated value. Over time, as the action-value estimates improve, the ϵ-greedy policy converges to the optimal policy. Another important method is Off-Policy MC Control with Weighted Importance Sampling, which allows the agent to learn an optimal target policy while following a different behavior policy to generate episodes. This is particularly advantageous when reusing data collected under a policy different from the one being optimized. Importance sampling weights are used to adjust for the discrepancy between the behavior policy and the target policy, ensuring unbiased estimates of the target policy's value. Off-policy MC Control can be implemented in both online and offline settings, making it versatile for various applications.

3.3 Temporal Difference Learning

Temporal Difference (TD) learning is a powerful approach to RL that combines the strengths of MC methods and DP. Unlike MC methods, which require complete episodes to update values, or DP, which requires a model of the environment, TD updates value estimates incrementally based on partial observations and does not require a model. Some popular TD algorithms include TD(0), SARSA, and Q-learning, each with distinct characteristics and applications.

TD(0), also known as one-step TD, is the simplest form of TD learning and is used for evaluating the value function of a given policy. It updates the value of the current state $V(s_t)$ incrementally, using the immediate reward r_{t+1} and the current estimate of the value of the next

state $V(s_{t+1})$. The update rule is given by

$$V(s_t) \leftarrow V(s_t) + \alpha\delta_t, \tag{3.2}$$

where $\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$ is the TD error. The TD error quantifies the difference between the current estimate of $V(s_t)$ and an improved estimate based on the observed reward and the next state. This error serves as a learning signal, guiding updates to reduce the mismatch between expectations and observed outcomes.

One of TD(0)'s key advantages is its efficiency. By comparing (3.1) and (3.2) we observe that, unlike MC methods, which rely on the complete return G_t over an episode, TD(0) only requires r_{t+1} and the estimate $V(s_{t+1})$. This use of a one-step return reduces variance compared to MC methods, as it avoids estimating long-term returns from limited samples. However, the trade-off is the introduction of bias, as $V(s_{t+1})$ is itself an estimate rather than the actual return. Despite this, TD(0) is guaranteed to converge to the true value function under standard assumptions, such as a fixed policy and appropriately chosen learning rates (Tsitsiklis and Van Roy, 1997). Once convergence is achieved, the bias is eliminated, making TD(0) a stable and practical method for many applications.

SARSA is an on-policy TD learning algorithm that estimates the action-value function $Q(s, a)$ for the policy currently being followed. When combined with an ϵ-greedy improvement rule, it can be used for policy optimization. Its name reflects the sequence of elements involved in its update rule: State, Action, Reward, next State, next Action. Unlike TD(0), which focuses on state-value functions $V(s)$, SARSA learns action-value functions $Q(s, a)$, which satisfies the Bellman expectation equation (2.2). The update rule for SARSA is

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)].$$

As an on-policy method, SARSA updates its action-value estimates using data generated by the same (potentially exploratory) policy that it evaluates, such as an ϵ-greedy policy.

Q-learning is one of the most widely used RL algorithms and is a fundamental off-policy TD learning algorithm. It learns the optimal action-value function $Q_*(s, a)$ satisfying the Bellman optimality equation (2.4). Unlike SARSA, which updates based on the action taken under the current policy, Q-learning evaluates the action-value function using the maximum Q-value of the next state, independent of the behavior policy. Its update rule is

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)].$$

The use of the max operator enables Q-learning to learn an optimal policy regardless of the behavior policy used to generate data, making it an off-policy method. However, this also introduces a potential issue known as maximization bias. Since the algorithm always selects the action with the highest estimated Q-value, even random noise in the Q-value estimates can result in inflated values, leading the agent to overestimate the value of certain actions. This bias can hinder exploration and cause suboptimal behavior. To address this, extensions such as Double Q-learning (van Hasselt, 2010), which will be discussed in the next section, have been developed to mitigate the effects of maximization bias. An example Q-learning algorithm is provided in Algorithm 1.

For intuition, it is helpful to contrast these approaches with familiar perspectives in statistics. DP resembles a purely mathematical exercise: with full knowledge of the transition probabilities and rewards, value functions can be solved exactly. MC methods are closer to statistical

Algorithm 1 Q-learning algorithm with ϵ-greedy exploration policy.

Initialize Q-table $Q(s, a)$ arbitrarily for all state-action pairs
Set learning rate $\alpha \in (0, 1]$, discount factor $\gamma \in [0, 1]$, and exploration probability $\epsilon \in [0, 1]$
for each episode **do**
  Initialize the starting state s
  repeat
    With probability ϵ, select a random action a (exploration), otherwise select $a = \arg\max_a Q(s, a)$ (exploitation)
    Take action a, observe reward r and the next state s'
    Update Q-value: $Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a)\right]$
    Set $s \leftarrow s'$
  until s is terminal
end for

practice, where one draws conclusions from data points, such as episodes of experience, rather than from known probabilities. TD methods fall in between and are naturally suited to online learning: they update estimates sequentially as data arrive, combining observed outcomes with current predictions. In this view, DP relies entirely on a known model of the environment, MC estimates values directly from observed episodes, and TD provides an incremental, online way of updating estimates as new data arrive.

3.4 Policy-Based Methods

The algorithms discussed so far—DP, MC, and TD learning—are all value-based methods. They rely on estimating value functions such as $V(s)$ or $Q(s, a)$. A complementary family of methods, known as policy-based algorithms, instead optimize the policy parameters directly. These approaches are particularly useful when the action space is continuous or when stochastic policies are required to ensure sufficient exploration.

Let $\pi_\theta(a|s)$ denote a differentiable policy parameterized by θ, and define the expected discounted return under this policy as $J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1}\right]$. The goal of policy-based methods is to find the parameter θ that maximizes $J(\theta)$. The policy-gradient theorem (Sutton et al., 1999) provides the foundation:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(A_t|S_t)\, Q_{\pi_\theta}(S_t, A_t)\right]. \tag{3.3}$$

This expression implies that we can perform stochastic gradient ascent on $J(\theta)$ using samples from trajectories generated by the current policy.

A straightforward algorithm based on the policy-gradient theorem is the REINFORCE method (Williams, 1992). The central idea is to replace the action-value function $Q_{\pi_\theta}(S_t, A_t)$ in (3.3) with the empirical return from time t onward, $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$, yielding an unbiased Monte Carlo estimate of the gradient, $\nabla_\theta J(\theta) \approx \mathbb{E}\left[\nabla_\theta \log \pi_\theta(A_t|S_t)\, G_t\right]$. Accordingly, stochastic gradient ascent on $J(\theta)$ updates the policy parameters via $\theta_{t+1} = \theta_t + \alpha\, \nabla_\theta \log \pi_{\theta_t}(A_t|S_t)\, G_t$, where $\alpha > 0$ is the learning rate. This update increases the probability of actions that lead to higher-than-average returns, thereby improving the expected performance of the policy.

Although the REINFORCE gradient estimator is unbiased, it can exhibit high variance, particularly in long-horizon problems where the returns G_t fluctuate substantially. A common strategy to reduce variance is to subtract a baseline $b(S_t)$ that depends on the current state

but not on the action. Because $\mathbb{E}_{\pi_\theta}\big[\nabla_\theta \log \pi_\theta(A_t|S_t)\, b(S_t) \mid S_t\big] = 0$, introducing such a baseline does not change the expected value of the gradient but can substantially reduce its variance (Williams, 1992; Greensmith et al., 2004). The resulting update rule is

$$\theta_{t+1} = \theta_t + \alpha\, \nabla_\theta \log \pi_{\theta_t}(A_t|S_t)\, \big(G_t - b(S_t)\big). \tag{3.4}$$

A particularly effective choice is the state-value function $b(S_t) = V_{\pi_\theta}(S_t)$, which yields the minimum-variance unbiased estimator in a least-squares sense (Greensmith et al., 2004). Substituting this choice into (3.4) gives the advantage-weighted update $\theta_{t+1} = \theta_t + \alpha\, \nabla_\theta \log \pi_{\theta_t}(A_t|S_t)$ $A_{\pi_\theta}(S_t, A_t)$, where the advantage function $A_{\pi_\theta}(S_t, A_t) = Q_{\pi_\theta}(S_t, A_t) - V_{\pi_\theta}(S_t) \approx G_t - V_{\pi_\theta}(S_t)$ measures how much better the chosen action performs relative to the expected return under the policy at state S_t. This advantage formulation naturally motivates the family of actor–critic methods, which further improve stability and sample efficiency by estimating the advantage through TD learning rather than Monte Carlo returns.

In actor–critic algorithms (Barto et al., 1983; Konda and Tsitsiklis, 2000), the critic learns a value function $V_w(s)$ or $Q_w(s, a)$ parameterized by w, providing a low-variance estimate of the advantage, while the actor updates the policy parameters θ in the direction suggested by this estimate. A typical actor–critic update takes the form

$$\begin{aligned}
\delta_t &= R_{t+1} + \gamma\, V_w(S_{t+1}) - V_w(S_t), \\
w &\leftarrow w + \beta\, \delta_t \nabla_w V_w(S_t), \\
\theta &\leftarrow \theta + \alpha\, \nabla_\theta \log \pi_\theta(A_t|S_t)\, \delta_t,
\end{aligned}$$

where δ_t is the TD error serving as an online estimate of the advantage of the chosen action. This architecture unites the strengths of both paradigms: the critic stabilizes learning by supplying low-variance feedback through value estimation, while the actor learns a flexible, possibly stochastic or continuous policy guided by these signals.

4 From Q-Learning to Broader Perspectives

Q-learning is one of the most influential algorithms in RL and has inspired major developments across both computer science and statistics. In computer science, it has served as the foundation for algorithmic advances in RL, where methods such as Double Q-learning, Deep Q-Networks, and Fitted Q-Iteration address issues of overestimation, scalability, and data efficiency. In statistics, Q-learning has been reinterpreted as a regression-based framework for estimating DTRs, emphasizing estimation accuracy, causal interpretability, and finite-sample efficiency. This section reviews these two lines of development in turn, highlighting how the same core idea has evolved differently across fields.

4.1 More on Q-Learning

Within the computer science tradition, Q-learning has been the foundation for numerous extensions designed to improve stability and performance. The classical tabular algorithm performs well in small state–action spaces but suffers from maximization bias and the curse of dimensionality. To address these issues, researchers introduced methods such as Double Q-learning, which mitigates overestimation; Deep Q-Networks (DQN), which use neural networks to approximate Q-functions in high-dimensional settings; and Fitted Q-Iteration (FQI), which adapts Q-learning

to offline settings. Further refinements, including Double DQN and Dueling Q-learning, extend these ideas to modern deep reinforcement learning.

One known limitation of standard Q-learning is maximization bias, which arises from overestimating action values due to the use of the max operator. Double Q-Learning addresses this issue by maintaining two separate Q-functions, $Q_A(s,a)$ and $Q_B(s,a)$, which are updated independently. These two functions alternate roles during action selection and evaluation. For instance, if Q_A is being updated, the update rule is

$$Q_A(S_t, A_t) \leftarrow Q_A(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q_B(S_{t+1}, \arg\max_{a'} Q_A(S_{t+1}, a')) - Q_A(S_t, A_t) \right].$$

Here, Q_A is used to select the best action in the next state, while Q_B is used to evaluate the Q-value of that action. The roles of Q_A and Q_B are alternated randomly at each step. This decoupling reduces overestimation and leads to more stable learning.

For environments with large or continuous state spaces, directly storing and updating action-value estimates for all state-action pairs becomes impractical. DQN addresses this limitation by using deep neural networks to approximate the Q-function (Mnih et al., 2015). The network takes a state as input and outputs the estimated Q-values for all possible actions. The learning process in DQN minimizes the TD error using the following loss function

$$L(\theta) = \mathbb{E}_{s,a,r,s'} \left[\left\{ r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta) \right\}^2 \right],$$

where r is the reward for taking action a in state s, θ are the weights of the network, and θ^- are the weights of a target network. The target network is a separate network which is updated less frequently to reduce oscillations and instability in training.

DQN also introduces experience replay to improve learning stability and efficiency. Experiences, represented as tuples (s, a, r, s'), are stored in a replay memory. Here s is what the agent observed, a is the action it took, r is the received reward, and s' is the next state it ended up. During training, mini-batches of experiences are randomly sampled from the memory to break correlations between consecutive experiences. This technique not only stabilizes learning but also increases data efficiency by reusing past experiences. However, because DQN updates targets using its own estimates, and learns from previously collected data, it can still exhibit instability when combined with function approximation—an interaction often referred to as the deadly triad (van Hasselt et al., 2018).

Despite its advantages, DQN inherits maximization bias from standard Q-learning. Double Deep Q-Network (DDQN) mitigates this issue by applying the principles of Double Q-Learning (van Hasselt et al., 2016). Another notable extension is Dueling Q-learning (Wang et al., 2016), which decomposes the Q-value into the state value $V(s)$ and the advantage $A(s,a)$: $Q(s,a) = V(s) + A(s,a)$. The dueling architecture is implemented using two separate streams in the neural network: one for $V(s)$ and another for $A(s,a)$. These streams are then combined to produce the final Q-values. This structure emphasizes the relative importance of the state value when the choice of actions matters less, such as in stable or slow-changing environments.

While DQN is primarily an online algorithm where the agent interacts with the environment and updates the policy during training, its use of experience replay incorporates an offline aspect. By combining online interaction with offline experience replay, DQN improves both efficiency and stability. A purely offline counterpart is FQI (Ernst et al., 2005), an offline, off-policy algorithm that treats Q-learning as a supervised regression problem. Instead of directly updating

Q-values, FQI iteratively fits a function approximator, such as a neural network or decision tree, to minimize the temporal difference error. At each iteration, the algorithm solves

$$\hat{Q}(s,a) \leftarrow \arg\min_{f} \mathbb{E}\left[\left(R_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - f(s,a)\right)^2\right],$$

where f is the chosen function approximator. This iterative fitting process allows FQI to generalize over high-dimensional or continuous state spaces, making it particularly effective for large-scale offline learning problems.

4.2 Comparison of Computer Science and Statistical RL Approaches

The extensions of Q-learning discussed in Section 4.1, such as Double Q-learning, DQN, and FQI, illustrate how the computer science community has advanced RL by addressing bias, stability, and scalability (van Hasselt et al., 2016; Mnih et al., 2015; Ernst et al., 2005). More broadly, RL has evolved along two distinct but complementary trajectories in computer science and statistics.

In computer science, the emphasis has been on achieving high performance in large-scale, high-dimensional environments such as robotics, games, and language models (Sutton and Barto, 2018; Arulkumaran et al., 2017). Methods range from classical Q-learning to modern deep RL, relying heavily on neural networks, large-scale optimization, and stabilization techniques such as experience replay or target networks. Success is measured primarily through empirical benchmarks, with less attention to uncertainty quantification or interpretability.

In contrast, the statistical tradition has developed RL tools for settings where data are limited, observational, or sequentially dependent, as in biomedical and social sciences (Murphy, 2003; Chakraborty and Moodie, 2013; Laber et al., 2014). Here, RL is framed not only as an optimization problem but also as an inferential framework. Approaches such as regression-based Q-learning, A-learning, inverse probability weighting, and doubly robust estimators are grounded in causal inference and semiparametric efficiency theory, emphasizing bias correction, uncertainty quantification, and interpretability.

These perspectives address complementary challenges. Computer science contributes scalable algorithms for complex state and action spaces, while statistics provides inferential guarantees and scientifically interpretable decision rules under observational designs (Ertefaie and Strawderman, 2018; Luckett et al., 2020). Integrating these strengths is crucial for applications such as DTRs, which demand methods that both handle complex longitudinal data and yield reliable causal conclusions. The next section develops this statistical perspective in greater detail.

5 RL and DTRs

DTRs provide a framework for sequential decision-making, where treatment decisions are made at each stage based on an individual's evolving characteristics and responses. Widely applied in healthcare and personalized medicine, DTRs aim to optimize long-term outcomes by identifying the best policy that maximizes cumulative rewards such as improved clinical outcomes (Chakraborty and Moodie, 2013; Laber et al., 2014).

Both DTRs and RL address sequential decision-making problems, making RL a natural choice for learning DTRs. However, there are notable differences between the two. While RL often assumes the agent can interact with the environment to generate data and adapt policies

online, DTR typically relies on fixed datasets collected from clinical trials or observational studies. Additionally, DTR problems often involve finite decision points, reflecting the constraints of healthcare interventions, whereas RL can handle infinite horizons. Nevertheless, emerging applications such as wearable devices and mobile health technologies generate continuous data streams with many or even ongoing decision points, motivating growing interest in both infinite-horizon formulations and online learning approaches (Ertefaie and Strawderman, 2018; Zhou et al., 2024; Luckett et al., 2020).

In offline settings, Murphy (2005) discusses a version of batch Q-learning in a non-stationary, non-Markovian environment with a finite horizon. This is a regression-based, dynamic programming approach that iterates backward through the stages of decision-making. Starting from the terminal stage, where rewards are directly observed, the algorithm estimates Q-values for earlier stages using regression models to approximate the expected cumulative reward conditioned on the state and action at each stage. This stage-wise approach aligns with the sequential nature of DTRs, making it particularly effective for applications that require modeling non-Markovian and non-stationary processes, which often arise in healthcare settings.

A parallel approach in the same setting is A-learning (Murphy, 2003; Schulte et al., 2015). Unlike Q-learning, which estimates the Q-value function $Q(s, a)$, A-learning focuses on the advantage function, defined as $A(s, a) = Q(s, a) - V(s)$. The advantage function quantifies how much better taking a specific action a in state s is compared to the average policy in that state. In DTR contexts, the advantage function is particularly useful when the goal is to identify the best treatment among several options, where knowing the relative superiority of treatments is more actionable than estimating their exact expected outcomes. Murphy (2003) conceptualized the advantage function as a regret function, measuring how suboptimal a specific action is compared to the best possible action.

In addition to action-focused methods like Q-learning and A-learning, directly estimating the values of all candidate policies is another important approach in DTRs. Once all candidate policies are evaluated, finding the optimal one is straightforward. In DTR applications, where datasets are often observational and offline, methods such as Inverse Probability Weighting, Augmented Inverse Probability Weighting, and Marginal Structural Models are commonly employed to estimate the value of policies (Zhao et al., 2015; Zhang et al., 2013; Robins et al., 2000). These approaches are closely related to off-policy evaluation (OPE) in RL; see Uehara et al. (2022) for a comprehensive review.

While RL methods can be adapted for learning DTRs, applying them effectively requires addressing certain challenges. One primary issue is the reliance of RL on the Markov Decision Process framework, which assumes both the Markov property, where future states depend only on the current state and action, and stationarity, meaning transition probabilities and rewards remain constant over time. In real-world healthcare applications, these assumptions often fail to hold. Patient outcomes, for example, may depend on their full treatment history, violating the Markov property. Similarly, changes in medical practices or patient populations can introduce non-stationarity. These challenges necessitate the development of RL methods that relax these assumptions, enabling their application to DTRs.

6 Addressing Causal Inference Challenges in DTRs

At their core, DTRs aim to uncover causal relationships, not just correlations, between treatments and outcomes. The focus is on understanding the causal impact of a sequence of decisions

to guide adaptive, personalized treatment strategies. For example, in managing diabetes, the goal is not merely to observe that patients receiving a particular medication tend to have lower blood sugar levels but to determine whether the medication itself causes the improvement, as opposed to other factors that might be correlated with both medication use and blood sugar levels. This intrinsic relationship allows researchers to look at DTRs from a causal perspective, and use frameworks in causal inference to analyze them. Murphy (2003) has a discussion on using the potential outcomes framework to connect Robins' g-formula (Robins, 1986) to DTRs.

This close relationship between DTRs and causal inference introduces a variety of challenges, ranging from confounding to time-varying treatment effects to interference. These challenges are not just theoretical; they have practical implications for designing, analyzing, and implementing DTRs. To address them, researchers have adapted tools and methods from causal inference, as well as RL algorithms, to meet the demands of real-world applications like healthcare.

6.1 Time-Varying Treatment and Non-Markovian/Non-stationary Dynamics

In healthcare applications, patient outcomes often depend on the entire history of treatments and covariates, not just the most recent state and action. This violates the Markov property, creating a setting where treatments interact over time to affect outcomes. Additionally, the relationships between treatments, covariates, and outcomes can evolve over time due to changes in disease progression, medical guidelines, or patient populations, introducing non-stationarity. These dynamics complicate the modeling of sequential decision-making, as assumptions of static or memoryless processes no longer hold.

The longitudinal g-formula from causal inference provides a principled framework for addressing these challenges (Robins, 1986; Hernán and Robins, 2024). It generalizes the standard g-formula by iteratively modeling the joint distribution of covariates, treatments, and outcomes over time. Interestingly, the backward recursion used in the g-formula for estimating the counterfactual mean outcome under a treatment regime is mathematically equivalent to the backward iteration formula in dynamic programming used in Q-learning for DTRs (Murphy, 2003). This connection highlights the shared foundation between causal inference and RL approaches to DTRs. Another causal inference approach, Marginal Structural Models, addresses time-varying confounding by appropriately weighting observations to simulate a randomized-like treatment assignment over time (Robins et al., 2000). This reweighting adjusts for the evolving relationship between covariates and treatments, ensuring unbiased estimation of causal effects in DTRs.

From an RL perspective, non-Markovianity can be handled via state augmentation, while non-stationarity can be addressed via time indexing or adaptive methods. State augmentation involves expanding the state space to include a finite history of observations, actions, and rewards so that the process becomes Markov in the augmented state. One can then restrict attention to Markov policies without loss of optimality (Puterman, 1994; Bertsekas, 2017). Whether the augmentation is sufficient can be assessed with formal tests of the Markov property developed for sequential decision making (Shi et al., 2020). Once a Markovian representation is obtained, state abstraction compresses the augmented state to a lower-dimensional representation that preserves decision-relevant information, reducing variance and computational burden while maintaining correctness for control and evaluation (Li et al., 2006; Allen et al., 2021).

To handle non-stationarity, one option is to augment the state with an explicit time index so the dynamics become conditionally time-homogeneous on the augmented state. A complementary option is to use adaptive mechanisms that track or localize changes, applying tests of time-homogeneity, detecting change points, and then fitting policies segment-wise or updating

time-varying parameters (Padakandla et al., 2020; Li et al., 2025). By either embedding time or adapting to detected shifts, these approaches enable robust policy learning under changing dynamics.

While these tools are powerful, they entail trade-offs. State augmentation increases dimensionality and data demands; state abstraction can introduce bias if decision-relevant information is removed; and time indexing or adaptive methods may overfit or mis-specify regime changes without safeguards. In practice, careful testing, regularization, and validation, together with explicit causal targets and assumptions, help balance these risks and enable robust policy learning in DTRs.

6.2 Unmeasured Confounding

Unmeasured confounding presents a significant challenge in both causal inference and DTRs. It arises when unobserved variables influence both treatments and outcomes, leading to biased estimates of causal effects and invalid conclusions. In DTRs, which often rely on observational data, the presence of unmeasured confounders can compromise the validity of estimated treatment effects, potentially resulting in suboptimal or harmful treatment strategies. Rigorous methods to mitigate unmeasured confounding are essential to ensure the validity and reliability of DTR analyses.

Instrumental variable (IV) methods offer one approach to addressing unmeasured confounding. IVs are variables associated with the treatment but influence the outcome only indirectly through the treatment (Angrist et al., 1996). In dynamic settings, extensions of IV methods have been developed to account for the sequential nature of decision-making in DTRs. For instance, Chen and Zhang (2023) estimate properly defined "optimal" DTRs using a time-varying IV to address unmeasured confounding, even when potential outcome distributions are only partially identified. Complementarily, Xu et al. (2023) develop an IV-based framework for off-policy evaluation with confounded observational trajectories, establishing identification of a target policy's value in infinite-horizon settings and proposing consistent estimators under suitable sequential IV conditions.

Another recent advancement in addressing unmeasured confounding is proximal causal inference (Miao et al., 2018). This framework introduces two types of proxies for the unmeasured confounder: a negative control outcome and a negative control treatment. These proxies help identify causal effects even when unmeasured confounding is present. Proximal causal inference has been adapted for use in DTRs under the framework of a Partially Observable Markov Decision Process (POMDP). In a POMDP, the true state is not directly observed, but proxies about the state can be used to inform decision-making. Tennenholtz et al. (2020) applied this approach to offline policy evaluation, leveraging past and future observations as proxies for unobserved confounders. Building on this work, Bennett and Kallus (2024) extended the framework to handle continuous state spaces, enabling its application to complex and evolving environments.

While IV and proximal causal inference methods have seen significant development, other approaches, such as Difference-in-Differences (Diff-in-Diff) and Synthetic Control, have been less explored in the context of DTRs (Shahn et al., 2025; Agarwal et al., 2025). Diff-in-Diff relies on comparing treated and untreated groups over time to estimate causal effects, while Synthetic Control constructs a weighted combination of untreated observations to serve as a counterfactual for the treated group. Although these methods have been primarily applied in static settings, adapting them to dynamic treatment frameworks could provide new avenues for addressing unmeasured confounding in DTRs.

6.3 Other Issues

Beyond confounding, several other causal inference challenges also exist in DTRs. One such challenge is interference, which occurs when the treatment of one individual affects the outcomes of another, thereby violating the Stable Unit Treatment Value Assumption (SUTVA). For instance, in healthcare settings, the allocation of treatments within a community can influence not only the treated individuals but also those around them. Sherman et al. (2020) explore methods for identifying DTRs under interference conditions, offering frameworks that account for these dependencies and ensuring that sequential decision-making accounts for spillover effects.

Other challenges include missing data and measurement error, which are common in longitudinal studies and observational datasets. Incomplete observations or inaccuracies in recorded covariates can bias the estimation of causal effects and undermine the reliability of policy evaluations. Techniques such as multiple imputation and sensitivity analysis have been adapted to mitigate these issues in the DTR context (Lyu et al., 2023; Spicker and Wallace, 2020). These methods enable researchers to account for uncertainty in missing or erroneous data, improving the robustness of DTR estimates.

7 Challenges and Open Problems

The integration of RL with DTRs holds great promise for advancing personalized medicine. However, realizing this potential requires addressing several significant challenges. For example, Luo et al. (2024) critically examine the application of offline RL algorithms to DTRs, highlighting issues such as inconsistent and potentially inconclusive evaluation metrics. Below, we explore two critical issues, interpretability and sample complexity, and outline ongoing efforts and open problems in these areas.

In high-stakes domains like healthcare, interpretability is essential for fostering trust in RL-based DTR systems. Medical professionals and patients need to understand the rationale behind treatment recommendations to ensure informed decision-making. Without such transparency, RL systems risk being viewed as "black boxes", limiting their adoption and raising ethical concerns. This challenge is a major reason why methods from causal inference have played an essential role in DTRs. In addition to leveraging causal inference, there is a growing need to design RL methods that are inherently interpretable to inspire confidence in their application to DTRs.

Several strategies have been proposed to enhance interpretability in RL for DTRs. These include using tree-based or rule-based methods that express regimes as simple if–then decision structures readily understandable in clinical contexts (Silva et al., 2020; Zhang et al., 2018), rather than relying on complex neural networks. Attention mechanisms (Choi et al., 2016) can also be employed to highlight the most relevant parts of the input data when making decisions. Local explanation techniques, such as Shapley values or feature attribution methods (Gupta et al., 2020), provide insights into individual predictions, while counterfactual analysis explores how changes in input variables would alter treatment recommendations (Madumal et al., 2020).

As discussed in Section 5, DTRs are naturally embedded within the causal inference framework for time-varying treatments, which provides a more interpretable foundation than many black-box RL approaches. The challenge, however, is that modern applications such as electronic health records (EHRs) and wearable devices generate data that are irregularly sampled, heterogeneous in format, and prone to noise and missingness, which complicates efforts to maintain interpretability in practice. While traditional statistical tools such as dimension reduction (Cook, 2007), sparsity-inducing methods (Tibshirani, 1996), and tree-based models (Breiman

et al., 1984) provide one route to preserving interpretability, modern representation learning methods (Bengio et al., 2013; Miotto et al., 2016; Johansson et al., 2016) offer additional advantages by flexibly integrating multimodal data, capturing nonlinear relationships, and leveraging large unlabeled datasets. Embedding these approaches within the causal inference framework may yield clinically meaningful, low-dimensional summaries of complex patient histories while maintaining valid counterfactual interpretations. Despite these advances, developing methods that balance interpretability with predictive accuracy remains an important open problem in sequential decision-making.

Another significant challenge in applying RL to DTRs is the high sample complexity of many RL algorithms, particularly in deep RL. Sample complexity refers to the amount of data required for an RL algorithm to learn an effective policy. In healthcare, large observational resources such as EHRs and registries provide rich longitudinal information but are prone to confounding, irregular sampling, and measurement error, while Sequential Multiple Assignment Randomized Trials (SMART) generate higher-quality data tailored to DTRs but often face sample size limitations due to practical and ethical constraints. To address this mismatch, one promising direction is to leverage existing knowledge across settings. Transfer learning provides a general framework for adapting models trained in one domain to another, thereby reducing the demand for new data (Zhu et al., 2023). A particularly practical form is fine-tuning, where models pretrained on large observational datasets such as EHRs are adjusted using smaller, carefully collected trial data. This strategy better aligns models with the target population while minimizing the need for extensive new samples. Meta-learning extends this idea by training agents across multiple tasks so that they can fine-tune rapidly to new environments, making them especially effective in data-limited healthcare contexts (Finn et al., 2017). Beyond transfer-based approaches, reward shaping offers a complementary strategy by modifying the reward structure itself. When designed to align with clinical priorities, it can improve sample efficiency, though in DTRs this requires caution to ensure surrogate outcomes truly reflect long-term goals (Komorowski et al., 2018; Ng et al., 1999).

Taken together, interpretability and sample complexity represent two of the most pressing challenges in applying RL to DTRs. Addressing them requires balancing methodological rigor with clinical practicality: regimes must be transparent enough to inspire trust while also data-efficient enough to be feasible in resource-limited healthcare settings. Progress will hinge on developing methods that blend statistical foundations with modern learning techniques, while remaining grounded in the realities of clinical data. Ultimately, sustained collaboration across statistics, computer science, and medicine will be essential for moving RL-based DTRs from theoretical promise to reliable, personalized, and trustworthy clinical decision support.

References

Agarwal A, Han S, Saha D, Syrgkanis V, Yoon H (2025). Synthetic blips: Generalizing synthetic controls for dynamic treatment effects. arXiv preprint: https://arxiv.org/abs/2210.11003v2.

Allen C, Parikh N, Gottesman O, Konidaris G (2021). Learning Markov state abstractions for deep reinforcement learning. In: *Advances in Neural Information Processing Systems*, volume 34, 8229–8241.

Angrist JD, Imbens GW, Rubin DB (1996). Identification of causal effects using instrumental variables. *Journal of the American Statistical Association*, 91(434): 444–455.

https://doi.org/10.1080/01621459.1996.10476902

Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6): 26–38. https://doi.org/10.1109/MSP.2017.2743240

Barto AG, Sutton RS, Anderson CW (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man and Cybernetics*, SMC–13(5): 834–846. https://doi.org/10.1109/TSMC.1983.6313077

Bellman RE (1957). *Dynamic Programming*. Princeton University Press.

Bengio Y, Courville A, Vincent P (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8): 1798–1828. https://doi.org/10.1109/TPAMI.2013.50

Bennett A, Kallus N (2024). Proximal reinforcement learning: Efficient off-policy evaluation in partially observed Markov decision processes. *Operations Research*, 72(3): 1071–1086. https://doi.org/10.1287/opre.2021.0781

Bertsekas DP (2017). *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA, 4th edition.

Breiman L, Friedman JH, Olshen RA, Stone CJ (1984). *Classification and Regression Trees*. Chapman and Hall/CRC, New York, 1st edition.

Cai H, Ren K, Zhang W, Malialis K, Wang J, Yu Y, et al. (2017). Real-time bidding by reinforcement learning in display advertising. In: *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 661–670. ACM.

Chakraborty B, Moodie EE (2013). *Statistical Methods for Dynamic Treatment Regimes*. Springer, New York, NY.

Chen S, Zhang B (2023). Estimating and improving dynamic treatment regimes with a time-varying instrumental variable. *Journal of the Royal Statistical Society, Series B, Statistical Methodology*, 85(2): 427–453. https://doi.org/10.1093/jrsssb/qkad011

Choi E, Bahadori MT, Kulas JA, Schuetz A, Stewart WF, Sun J (2016). RETAIN: An interpretable predictive model for healthcare using reverse time attention mechanism. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 3512–3520.

Cook RD (2007). Fisher lecture: Dimension reduction in regression. *Statistical Science*, 22(1): 1–26. https://doi.org/10.1214/088342306000000682

Covington P, Adams J, Sargin E (2016). Deep neural networks for YouTube recommendations. In: *Proceedings of the 10th ACM Conference on Recommender Systems*, 191–198. ACM.

Ernst D, Geurts P, Wehenkel L (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6: 503–556.

Ertefaie A, Strawderman RL (2018). Constructing dynamic treatment regimes over indefinite time horizons. *Biometrika*, 105(4): 963–977. https://doi.org/10.1093/biomet/asy043

Finn C, Abbeel P, Levine S (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In: *Proceedings of the 34th International Conference on Machine Learning*, volume 70, 1126–1135. PMLR.

Greensmith E, Bartlett PL, Baxter J (2004). Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5: 1471–1530.

Gupta P, Puri N, Verma S, Kayastha D, Deshmukh S, Krishnamurthy B, et al. (2020). Explain your move: Understanding agent actions using specific and relevant feature attribution. In: *International Conference on Learning Representations*.

Hernán MA, Robins JM (2024). *Causal Inference: What If*. Chapman & Hall/CRC. CRC Press.

Johansson FD, Shalit U, Sontag D (2016). Learning representations for counterfactual inference. In: *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, 3020–3029. PMLR.

Kendall A, Hawke J, Janz D, Mazur P, Reda D, Allen JM, et al. (2019). Learning to drive in a day. In: *2019 International Conference on Robotics and Automation*, 8248–8254.

Kober J, Bagnell JA, Peters J (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11): 1238–1274. https://doi.org/10.1177/0278364913495721

Komorowski M, Celi LA, Badawi O, Gordon AC, Faisal AA (2018). The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. *Nature Medicine*, 24(11): 1716–1720. https://doi.org/10.1038/s41591-018-0213-5

Konda VR, Tsitsiklis JN (2000). Actor–critic algorithms. In: *Advances in Neural Information Processing Systems*, volume 12, 1008–1014. MIT Press.

Laber EB, Lizotte DJ, Qian M, Pelham WE, Murphy SA (2014). Dynamic treatment regimes: Technical challenges and applications. *Electronic Journal of Statistics*, 8(1): 1225–1272. https://doi.org/10.1214/14-EJS920

Li L, Walsh TJ, Littman ML (2006). Towards a unified theory of state abstraction for MDPs. In: *Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*, 531–539.

Li M, Shi C, Wu Z, Fryzlewicz P (2025). Testing stationarity and change point detection in reinforcement learning. *The Annals of Statistics*, 53(3): 1230–1256. https://doi.org/10.1214/25-AOS2501

Luckett DJ, Laber EB, Kahkoska AR, Maahs DM, Mayer-Davis E, Kosorok MR (2020). Estimating dynamic treatment regimes in mobile health using V-learning. *Journal of the American Statistical Association*, 115(530): 692–706. https://doi.org/10.1080/01621459.2018.1537919

Luo Z, Pan Y, Watkinson P, Zhu T (2024). Reinforcement learning in dynamic treatment regimes needs critical reexamination. arXiv preprint: https://arxiv.org/abs/2405.18556.

Lyu L, Cheng Y, Wahed AS (2023). Imputation-based Q-learning for optimizing dynamic treatment regimes with right-censored survival outcome. *Biometrics*, 79(4): 3676–3689. https://doi.org/10.1111/biom.13872

Madumal P, Miller T, Sonenberg L, Vetere F (2020). Explainable reinforcement learning through a causal lens. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2493–2500.

Miao W, Geng Z, Tchetgen Tchetgen EJ (2018). Identifying causal effects with proxy variables of an unmeasured confounder. *Biometrika*, 105(4): 987–993. https://doi.org/10.1093/biomet/asy038

Miotto R, Li L, Kidd BA, Dudley JT (2016). Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. *Scientific Reports*, 6: 26094. https://doi.org/10.1038/srep26094

Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533. https://doi.org/10.1038/nature14236

Murphy SA (2003). Optimal dynamic treatment regimes. *Journal of the Royal Statistical Society, Series B, Statistical Methodology*, 65(2): 331–355. https://doi.org/10.1111/1467-9868.00389

Murphy SA (2005). A generalization error for Q-learning. *Journal of Machine Learning Research*,

6(37): 1073–1097.

Ng AY, Harada D, Russell S (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In: *Proceedings of the International Conference on Machine Learning*, volume 99, 278–287.

Padakandla S, KJ P, Bhatnagar S (2020). Reinforcement learning algorithm for non-stationary environments. *Applied Intelligence*, 50(11): 3590–3606. https://doi.org/10.1007/s10489-020-01758-5

Puterman ML (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley Series in Probability and Mathematical Statistics*. John Wiley & Sons, New York.

Robins J (1986). A new approach to causal inference in mortality studies with a sustained exposure period—application to control of the healthy worker survivor effect. *Mathematical Modelling*, 7(9–12): 1393–1512. https://doi.org/10.1016/0270-0255(86)90088-6

Robins J, Hernán M, Brumback B (2000). Marginal structural models and causal inference in epidemiology. *Epidemiology*, 11(5): 550–560. https://doi.org/10.1097/00001648-200009000-00011

Schulte PJ, Tsiatis AA, Laber EB, Davidian M (2015). Q-and A-learning methods for estimating optimal dynamic treatment regimes. *Statistical Science*, 29(4): 640.

Shahn Z, Dukes O, Shamsunder M, Richardson D, Tchetgen Tchetgen ET, Robins J (2025). Structural nested mean models under parallel trends assumptions. arXiv preprint: https://arxiv.org/abs/2204.10291v8.

Sherman E, Arbour D, Shpitser I (2020). General identification of dynamic treatment regimes under interference. In: *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, volume 108, 3917–3927. PMLR.

Shi C, Wan R, Song R, Lu W, Leng L (2020). Does the Markov decision process fit the data: Testing for the Markov property in sequential decision making. In: *International Conference on Machine Learning*, 8807–8817. PMLR.

Silva A, Gombolay M, Killian T, Jimenez I, Son SH (2020). Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In: *International Conference on Artificial Intelligence and Statistics*, 1855–1865. PMLR.

Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G, et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587): 484–489. https://doi.org/10.1038/nature16961

Spicker D, Wallace MP (2020). Measurement error and precision medicine: Error-prone tailoring covariates in dynamic treatment regimes. *Statistics in Medicine*, 39(26): 3732–3755. https://doi.org/10.1002/sim.8690

Sutton RS, Barto AG (2018). *Reinforcement Learning: An Introduction*. MIT press.

Sutton RS, McAllester DA, Singh SP, Mansour Y (1999). Policy gradient methods for reinforcement learning with function approximation. In: *Advances in Neural Information Processing Systems*, volume 12, 1057–1063.

Tennenholtz G, Shalit U, Mannor S (2020). Off-policy evaluation in partially observable environments. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 10276–10283.

Tibshirani R (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B, Statistical Methodology*, 58(1): 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

Tsitsiklis J, Van Roy B (1997). An analysis of temporal-difference learning with

function approximation. *IEEE Transactions on Automatic Control*, 42(5): 674–690. https://doi.org/10.1109/9.580874

Uehara M, Shi C, Kallus N (2022). A review of off-policy evaluation in reinforcement learning. arXiv preprint arXiv:2212.06355.

van Hasselt H (2010). Double Q-learning. In: *Advances in Neural Information Processing Systems*, volume 23, 2613–2621.

van Hasselt H, Doron Y, Strub F, Hessel M, Sonnerat N, Modayil J (2018). Deep reinforcement learning and the deadly triad. arXiv preprint arXiv:1812.02648.

van Hasselt H, Guez A, Silver D (2016). Deep reinforcement learning with double Q-learning. In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, volume 30, 2094–2100.

Wang Z, Schaul T, Hessel M, van Hasselt H, Lanctot M, Freitas N (2016). Dueling network architectures for deep reinforcement learning. In: *International Conference on Machine Learning*, volume 48, 1995–2003. PMLR.

Watkins CJ, Dayan P (1992). Q-learning. *Machine Learning*, 8: 279–292.

Williams RJ (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8: 229–256. https://doi.org/10.1023/A:1022672621406

Xu Y, Zhu J, Shi C, Luo S, Song R (2023). An instrumental variable approach to confounded off-policy evaluation. In: *International Conference on Machine Learning*, volume 202, 38848–38880. PMLR.

Yu C, Liu J, Nemati S, Yin G (2021). Reinforcement learning in healthcare: A survey. *ACM Computing Surveys*, 55(1): 1–36.

Zeng Y, Cai R, Sun F, Huang L, Hao Z (2025). A survey on causal reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 36(4): 5942–5962. https://doi.org/10.1109/TNNLS.2024.3403001

Zhang B, Tsiatis AA, Laber EB, Davidian M (2013). Robust estimation of optimal dynamic treatment regimes for sequential treatment decisions. *Biometrika*, 100(3): 681–694. https://doi.org/10.1093/biomet/ast014

Zhang Y, Laber EB, Davidian M, Tsiatis AA (2018). Interpretable dynamic treatment regimes. *Journal of the American Statistical Association*, 113(524): 1541–1549. https://doi.org/10.1080/01621459.2017.1345743

Zhao YQ, Zeng D, Laber EB, Kosorok MR (2015). New statistical learning methods for estimating optimal dynamic treatment regimes. *Journal of the American Statistical Association*, 110(510): 583–598. https://doi.org/10.1080/01621459.2014.937488

Zhou W, Zhu R, Qu A (2024). Estimating optimal infinite horizon dynamic treatment regimes via pT-learning. *Journal of the American Statistical Association*, 119(545): 625–638. https://doi.org/10.1080/01621459.2022.2138760

Zhu Z, Lin K, Jain AK, Zhou J (2023). Transfer learning in deep reinforcement learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11): 13344–13362. https://doi.org/10.1109/TPAMI.2023.3292075