SRINIVAS UNIVERSITY

INSTITUTE OF ENGINEERING & TECHNOLOGY

(SUBJECT: Artificial Neural Networks)

(SUBJECT CODE:24SBT113)

A GROUP TASK REPORT ON

"Comparative Study of Reinforcement and Stochastic Learning Rules: Dynamics, Stability, and Convergence Analysisss."

Submitted in the partial fulfillment of the requirements for the Fourth semester

BACHELOR OF TECHNOLOGY

IN

Artificial Intelligence & Machine Learning

Submitted By,

Pavan S Poojary	**01SU24AI071**
Prathvi Prashanth Naik	**01SU24AI076**
Prateek	**01SU24AI074**
Vignesh S Poojary	**01SU24AI116**
Vishal Shetty	**01SU24AI118**

UNDER THE GUIDANCE OF

Prof. Mahesh Kumar V B

Assistant Professor

DEPARTMENT OF ARTIFICIAL INTELLIGENCE &MACHINE LEARNING

SRINIVAS UNIVERSITY INSTITUTE OF ENGINEERING & TECHNOLOGY

MUKKA, MANGALURU-574146

2025-26

SRINIVAS UNIVERSITY

INSTITUTE OF ENGINEERING & TECHNOLOGY

Department of Artificial Intelligence &Machine Learning

CERTIFICATE

This is to certify that, **Pavan S Poojary (01SU24AI071), Prathvi Prashanth Naik (01SU24AI076), Prateek (01SU24AI074), Vignesh S Poojary (01SU24AI116), Vishal Shetty (01SU24AI118)** has satisfactorily completed the assessment (Group Task) in **Artificial Neural Networks (24SBT113)** prescribed by the Srinivas University for the 4th semester B. Tech course during the year **2025-26**.

MARKS AWARDED

<u>Staff In charge</u>

Name: Prof. Mahesh Kumar V B

TABLE OF CONTENTS

SI.no	Description	Page no.
1	Introduction	1
2	Mathematical Foundations	4
3	Reinforcement Learning Rules	7
4	Stochastic Learning Rules	14
5	Learning Dynamics Comparison: Reinforcement vs Stochastic	20
6	Stability and Convergence Analysis	26
7	Detailed Convergence Rate Comparison with Graphical Interpretation	32
8	Comparative Study: Reinforcement Learning vs Stochastic Learning	39
9	Experimental Case Study with Detailed Numerical Comparison	47
10	Poster Summary	54
11	Conclusion	58
12	References	61

1. Introduction

Learning rules form the mathematical backbone of artificial intelligence systems. They define **how parameters of a model are updated** in response to data, experience, or interaction with an environment. The effectiveness of any machine learning system largely depends on the design, stability, and convergence behavior of its learning rule.

In modern AI, two fundamental categories of learning rules dominate:

1. **Reinforcement Learning (RL) rules**
2. **Stochastic Learning rules**

Although both involve randomness and iterative parameter updates, they differ significantly in objective formulation, feedback mechanisms, and theoretical guarantees.

1.1 Motivation

As machine learning systems become more complex — such as deep neural networks, robotics controllers, recommendation engines, and autonomous systems — understanding the differences between learning paradigms is essential for:

- Selecting appropriate algorithms
- Ensuring stable training
- Improving convergence speed
- Reducing sample complexity
- Avoiding divergence and instability

Reinforcement learning is widely used in **sequential decision-making problems**, where an agent must learn through interaction and delayed reward signals. In contrast, stochastic learning rules are primarily used in **supervised and unsupervised learning**, where the goal is direct optimization of a loss function using sampled data.

1.2 Reinforcement Learning Perspective

Reinforcement learning is inspired by behavioral psychology and control theory. An agent interacts with an environment over time and learns a policy that maximizes cumulative reward. Unlike supervised learning, RL does not receive labeled target outputs. Instead, it must solve the **credit assignment problem**, determining which actions contributed to future rewards.

The objective in RL is typically defined as:

$$\boldsymbol{J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t \, r_t\right]}$$

where:

- θ= policy parameters
- γ= discount factor
- r_t= reward at time t

Learning updates are driven by reward-based signals, often estimated through temporal difference (TD) errors or policy gradients.

1.3 Stochastic Learning Perspective

Stochastic learning rules arise from stochastic approximation theory and optimization. They are primarily used to minimize an expected loss function:

$$J(\theta) = \mathbb{E}_{(x,y)}[L(\theta; x, y)]$$

Because computing the full expectation is expensive, updates are based on sampled gradients:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L(\theta_t; x_i, y_i)$$

This forms the basis of **Stochastic Gradient Descent (SGD)** and its variants.

Unlike RL, stochastic learning:

- Has immediate error feedback
- Does not require exploration mechanisms
- Often has stronger theoretical convergence guarantees (especially in convex settings)

1.4 Key Differences at a Glance

Feature	**Reinforcement Learning**	**Stochastic Learning**
Feedback Type	Delayed reward	Immediate loss
Problem Type	Sequential decision-making	Direct optimization
Noise Source	Environment + policy	Sampling noise
Stability	More complex	Well-understood
Convergence Guarantees	Conditional	Strong (convex case)

1.5 Importance of Comparative Study

Although reinforcement learning methods often use stochastic gradient ideas internally (e.g., policy gradient), their learning dynamics differ substantially due to:

- Bootstrapping
- Temporal credit assignment
- Exploration-exploitation trade-off
- Non-stationary targets

This report aims to provide a structured and mathematical comparison of reinforcement and stochastic learning rules in terms of:

- Learning dynamics
- Stability analysis
- Convergence behavior
- Theoretical guarantees
- Practical applications

By understanding these differences, researchers and practitioners can better choose suitable learning rules for specific applications and design more stable adaptive systems.

2. Mathematical Foundations

To rigorously compare reinforcement and stochastic learning rules, we begin with a unified mathematical framework. Both families of learning algorithms can be expressed as **iterative stochastic approximation processes**.

2.1 General Parameter Update Framework

Let the model parameters be:

$$\theta \in \mathbb{R}^d$$

At iteration t, the general update rule can be written as:

$$\theta_{t+1} = \theta_t + \alpha_t \, \Delta_t$$

where:

- α_t= learning rate (step size)
- Δ_t= update direction (may contain noise)

Both reinforcement and stochastic learning fit into this general structure.

2.2 Stochastic Approximation Form

A more general form:

$$\theta_{t+1} = \theta_t + \alpha_t [F(\theta_t) + M_{t+1}]$$

where:

- $F(\theta)$= expected update (true gradient or fixed point direction)
- M_{t+1}= stochastic noise (zero-mean under assumptions)

This equation is central to **Robbins–Monro stochastic approximation theory**, which provides convergence conditions.

2.3 Objective Function Formulation

(A) Stochastic Learning Objective

The goal is to minimize an expected loss:

$$J(\theta) = \mathbb{E}_{z \sim \mathcal{D}}[L(\theta; z)]$$

True gradient:

$$\nabla J(\theta) = \mathbb{E}[\nabla L(\theta; z)]$$

Because computing expectation is expensive, we approximate with a sample:

$$\nabla J(\theta) \approx \nabla L(\theta; z_t)$$

Update:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t; z_t)$$

(B) Reinforcement Learning Objective

In RL, the objective is maximizing expected return:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

The gradient is derived using the policy gradient theorem:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) Q^\pi(s, a)]$$

This gradient is estimated from trajectories, introducing additional variance compared to standard SGD.

2.4 Sources of Randomness

A key difference lies in randomness:

Stochastic Learning

- Random mini-batch sampling
- Measurement noise
- Data distribution variance

Reinforcement Learning

- Environment transitions
- Reward stochasticity
- Policy exploration
- Bootstrapped value estimates

Thus, RL typically exhibits **higher variance and non-stationary targets**, making stability more challenging.

2.5 Fixed-Point vs Optimization View

Another important distinction:

Stochastic Learning → Optimization Problem

$$\theta^* = \arg\min_{\theta} J(\theta)$$

Clear objective function with global or local minima.

Reinforcement Learning → Fixed-Point Problem

For example, Bellman equation:

$$Q^{\pi}(s,a) = \mathbb{E}[r + \gamma Q^{\pi}(s',a')]$$

Learning involves approximating a fixed point of a contraction mapping rather than directly minimizing a simple loss.

2.6 Summary of Mathematical Differences

Aspect	**Stochastic Learning**	**Reinforcement Learning**
Objective Type	Loss minimization	Return maximization
Problem Nature	Optimization	Fixed-point + optimization
Noise	Sampling noise	Environmental + policy noise
Target Stability	Stationary	Often non-stationary
Gradient Quality	Low variance	High variance

3. Reinforcement Learning Rules

Reinforcement Learning (RL) rules define how an agent updates its parameters based on **interaction with an environment** and **reward feedback**. Unlike supervised learning, RL does not rely on labeled input–output pairs. Instead, it learns through trial-and-error by maximizing cumulative reward over time.

This section presents the mathematical formulation, core update equations, algorithmic variants, diagrams, and illustrative examples.

3.1 Reinforcement Learning Framework

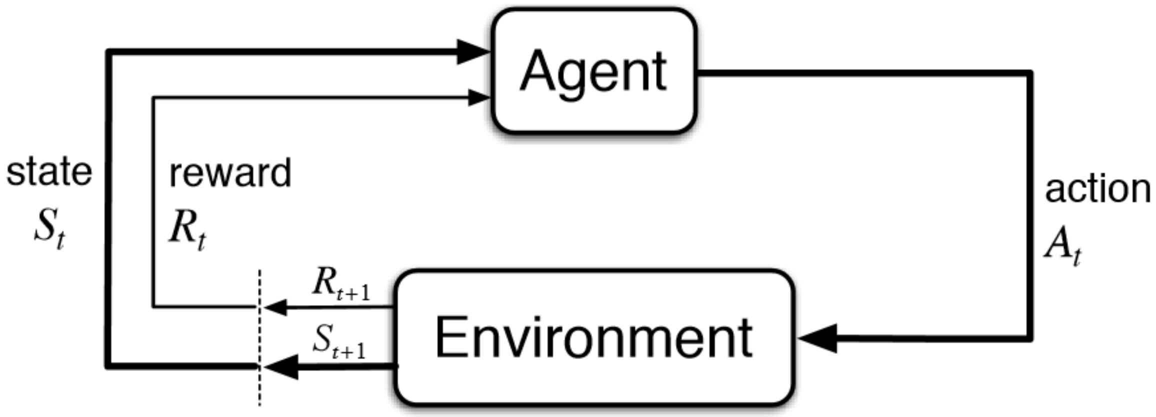
RL problems are commonly modeled as a **Markov Decision Process (MDP)** defined by:

- State space S
- Action space A
- Transition probability $P(s' \mid s, a)$
- Reward function $R(s, a)$
- Discount factor $\gamma \in (0,1)$

At each time step t:

1. Agent observes state s_t
2. Chooses action $a_t \sim \pi_\theta(a \mid s_t)$
3. Receives reward r_t
4. Moves to next state s_{t+1}

RL Interaction Loop



The learning objective is to maximize expected discounted return:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

3.2 Value-Based Learning Rules

Value-based methods learn a **value function** estimating expected future rewards.

3.2.1 State-Value Function

$$V^\pi(s) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s\right]$$

3.2.2 Action-Value Function

$$Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s, a_t = a\right]$$

3.3 Temporal Difference (TD) Learning

TD learning updates value estimates using bootstrapping.

TD Error:

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

Update Rule:

$$V(s_t) \leftarrow V(s_t) + \alpha\delta_t$$

Key Idea: Learn from difference between predicted and observed value.

3.4 Q-Learning (Off-Policy TD Control)

Q-learning is one of the most important RL learning rules.

Update Equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$$

Properties:

- Off-policy (learns optimal policy regardless of behavior policy)
- Converges in tabular case
- Uses greedy max operator

Example: Simple Gridworld

Suppose:

- Learning rate $\alpha = 0.1$
- Discount factor $\gamma = 0.9$
- Current Q-value = 5
- Reward = 2
- Max future Q = 8

Compute update:

$$\begin{aligned} Q_{new} &= 5 + 0.1[2 + 0.9(8) - 5] \\ &= 5 + 0.1(2 + 7.2 - 5) \\ &= 5 + 0.1(4.2) = 5 + 0.42 = 5.42 \end{aligned}$$

Thus, the Q-value increases toward the improved estimate.

3.5 SARSA (On-Policy Method)

SARSA updates using the action actually taken.

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$$

Difference from Q-learning:

- Uses next action from current policy
- More conservative
- Often more stable in stochastic environments

3.6 Policy Gradient Methods

Instead of learning value functions, policy gradient methods directly optimize the policy.

Policy Parameterization

Let:

$$\pi_\theta(a \mid s)$$

be a differentiable function (e.g., softmax neural network).

Objective Function

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

Policy Gradient Theorem

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) Q^\pi(s,a)]$$

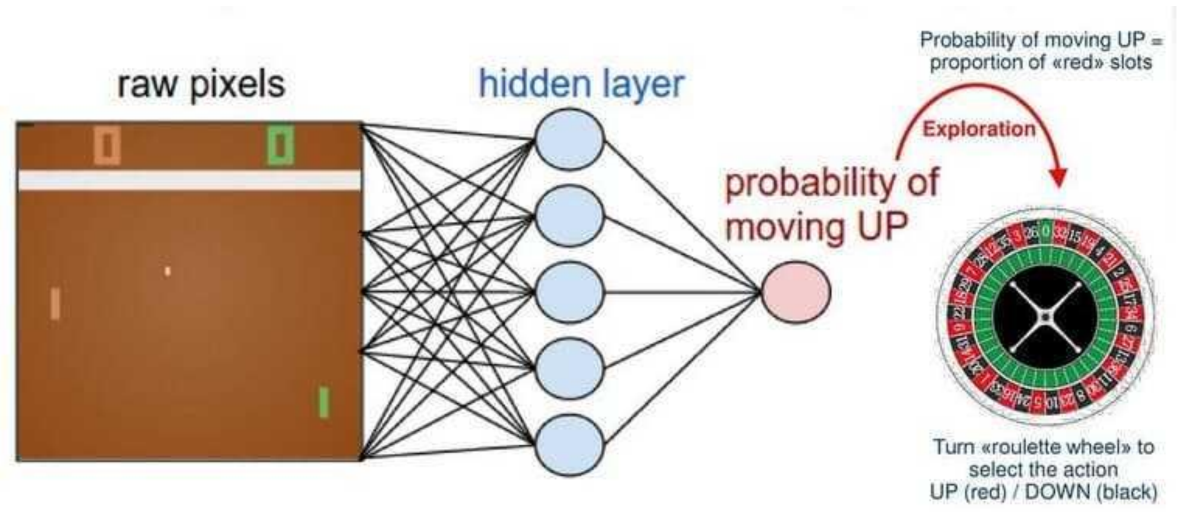
REINFORCE Update Rule

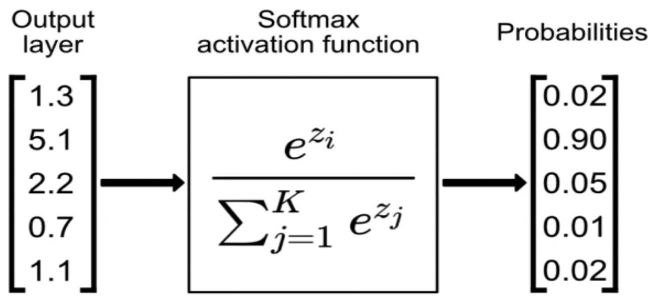
$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta \log \pi_\theta(a_t \mid s_t) G_t$$

Where:

- G_t= observed return from time t

Policy Gradient Illustration


raw pixels
hidden layer
probability of moving UP
Probability of moving UP = proportion of «red» slots
Exploration
Turn «roulette wheel» to select the action UP (red) / DOWN (black)



Output layer
Softmax activation function
Probabilities
1.3
5.1
2.2
0.7
1.1
$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
0.02
0.90
0.05
0.01
0.02

3.7 Actor–Critic Methods

Actor–Critic combines:

- Actor → policy update
- Critic → value function estimation

Update equations:

Critic (TD error):

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

Actor:

$$\theta \leftarrow \theta + \alpha \delta_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)$$

Advantages:

- Reduced variance
- Faster convergence than REINFORCE
- Widely used in deep RL

3.8 Learning Dynamics in Reinforcement Learning

Reinforcement learning exhibits unique dynamics:

1. Delayed Feedback

Reward may arrive many steps after action.

2. Credit Assignment Problem

Must determine which action caused reward.

3. Exploration–Exploitation Tradeoff

Agent must balance:

- Exploring new actions

- Exploiting known good actions

Common exploration strategy:

$$\epsilon\text{-greedy}$$

With probability ϵ, choose random action.

3.9 Stability Considerations

RL can become unstable due to:

- Bootstrapping
- Function approximation
- Off-policy learning
- Large learning rates

For convergence in tabular Q-learning:

$$\sum_{t=1}^{\infty} \alpha_t = \infty$$

$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

These conditions ensure diminishing step sizes.

3.10 Summary of Reinforcement Learning Rules

Method	**Type**	**Stability**	**Variance**	**Convergence Guarantee**
TD(0)	Value-based	Moderate	Low	Yes (tabular)
Q-learning	Off-policy	Moderate	Moderate	Yes (tabular)
SARSA	On-policy	High	Moderate	Yes (tabular)
REINFORCE	Policy-based	Low	High	Local optimum
Actor-Critic	Hybrid	High	Moderate	Empirical

Key Insights

- RL learning rules are driven by **reward-based feedback**.
- Bootstrapping accelerates learning but may cause instability.
- Policy gradient methods provide flexibility but increase variance.
- Actor–Critic methods balance bias and variance.

4. Stochastic Learning Rules

Stochastic learning rules are fundamental to supervised and unsupervised machine learning. Unlike reinforcement learning, these rules directly optimize a predefined objective (loss) function using randomly sampled data.

They originate from **stochastic approximation theory** and form the foundation of modern deep learning algorithms.

4.1 Optimization Framework

Assume we want to minimize an expected loss:

$$J(\theta) = \mathbb{E}_{z\sim\mathcal{D}}[L(\theta; z)]$$

Where:

- $\theta \in \mathbb{R}^d$are model parameters
- $z = (x, y)$is a sample
- $L(\theta; z)$is the loss function

The true gradient:

$$\nabla J(\theta) = \mathbb{E}[\nabla L(\theta; z)]$$

Since expectation is expensive to compute exactly, we approximate using sampled data.

4.2 Stochastic Gradient Descent (SGD)

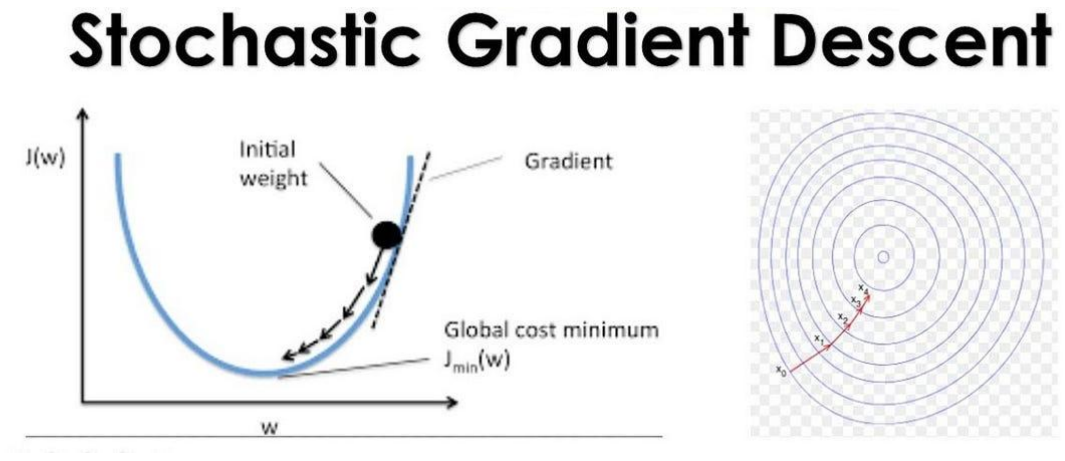
SGD updates parameters using a single randomly selected sample:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t; z_t)$$

Where:

- η= learning rate
- z_t= randomly sampled data point

4.2.1 SGD Geometry



Unlike full gradient descent, SGD produces a **noisy zigzag path** toward the minimum.

4.3 Mini-Batch SGD

To reduce variance:

$$\nabla J(\theta) \approx \frac{1}{m}\sum_{i=1}^{m} \nabla L(\theta; z_i)$$

Update:

$$\theta_{t+1} = \theta_t - \eta \frac{1}{m}\sum_{i=1}^{m} \nabla L(\theta; z_i)$$

Advantages:

- Lower variance
- Better hardware efficiency
- More stable convergence

4.4 Example: Linear Regression with SGD

Consider:

$$L(\theta) = \frac{1}{2}(y - \theta x)^2$$

Gradient:

$$\nabla_\theta L = -(y - \theta x)x$$

Update:

$$\theta_{t+1} = \theta_t + \eta(y - \theta_t x)x$$

Numerical Example

Let:

- $x = 2$
- $y = 6$
- $\theta = 1$
- $\eta = 0.1$

Compute:

$$y - \theta x = 6 - (1)(2) = 4$$
$$\theta_{new} = 1 + 0.1(4)(2) = 1 + 0.8 = 1.8$$

Thus SGD moves parameter toward optimal value.

4.5 Momentum Method

Momentum accelerates convergence in relevant directions.

Velocity update:

$$v_{t+1} = \beta v_t + \eta \nabla L(\theta_t)$$

Parameter update:

$$\theta_{t+1} = \theta_t - v_{t+1}$$

Where:

- $\beta \in (0,1)$controls momentum

Advantages:

- Faster convergence
- Reduced oscillations
- Better in deep networks

4.6 Adaptive Stochastic Methods

Popular variants:

- AdaGrad
- RMSProp
- Adam

Adam update:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$
$$\theta_{t+1} = \theta_t - \eta \frac{m_t}{\sqrt{v_t} + \epsilon}$$

These methods adjust learning rate per parameter.

4.7 Stochastic Approximation Theory

General form:

$$\theta_{t+1} = \theta_t + \alpha_t [F(\theta_t) + M_{t+1}]$$

Where:

- $F(\theta)$= true gradient
- M_t= zero-mean noise

Robbins–Monro conditions for convergence:

$$\sum_{t=1}^{\infty} \alpha_t = \infty$$
$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

Under convexity, convergence to global minimum is guaranteed.

4.8 Stability Analysis in Stochastic Learning

For convex function with Lipschitz constant L:

$$0 < \eta < \frac{2}{L}$$

ensures stability.

Lyapunov function:

$$V(\theta) = J(\theta) - J(\theta^*)$$

ensures decreasing energy over time.

4.9 Convergence Rate

Convex case:

$$\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$$

Strongly convex case:

$$\mathcal{O}\left(\frac{1}{T}\right)$$

Non-convex case:

Converges to critical point.

4.10 Learning Dynamics Characteristics

Stochastic learning rules exhibit:

1. Immediate feedback
2. Stationary objective
3. Controlled gradient noise
4. Well-defined minima
5. Strong theoretical guarantees

Compared to reinforcement learning:

- Lower variance
- Faster convergence
- Stronger stability

4.11 Summary of Stochastic Learning Rules

Method	Variance	Stability	Convergence
SGD	Moderate	High	Strong (convex)
Mini-batch	Lower	Higher	Faster
Momentum	Lower oscillation	High	Faster
Adam	Adaptive	Very High	Fast

Key Insights

- Stochastic learning directly optimizes a loss function.
- Noise arises from data sampling, not environment interaction.
- Theoretical analysis is simpler than RL.
- Convergence guarantees are stronger in convex settings.

5. Learning Dynamics Comparison: Reinforcement vs Stochastic

This section provides a **deep analytical comparison** of reinforcement learning (RL) rules and stochastic learning rules in terms of:

- Learning dynamics
- Noise characteristics
- Stability behavior
- Convergence properties
- Sensitivity to hyperparameters

Both families can be expressed under stochastic approximation theory, but their **practical behavior differs significantly**.

5.1 Unified Stochastic Approximation View

Both RL and stochastic gradient descent (SGD) can be written as:

$$\theta_{t+1} = \theta_t + \alpha_t[F(\theta_t) + M_{t+1}]$$

Where:

- $F(\theta)$= expected update direction
- M_t= stochastic noise term

The key difference lies in the structure of $F(\theta)$and the properties of the noise term.

5.2 Noise Characteristics

5.2.1 Stochastic Learning Noise

In SGD:

$$M_t = \nabla L(\theta_t; z_t) - \nabla J(\theta_t)$$

Properties:

- Zero mean (under unbiased sampling)
- Variance decreases with mini-batch size
- Noise is independent across iterations (approx.)
- Stationary distribution

5.2.2 Reinforcement Learning Noise

In RL:

Noise arises from:

- Random transitions
- Reward stochasticity
- Exploration policy
- Bootstrapping targets

TD error:

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

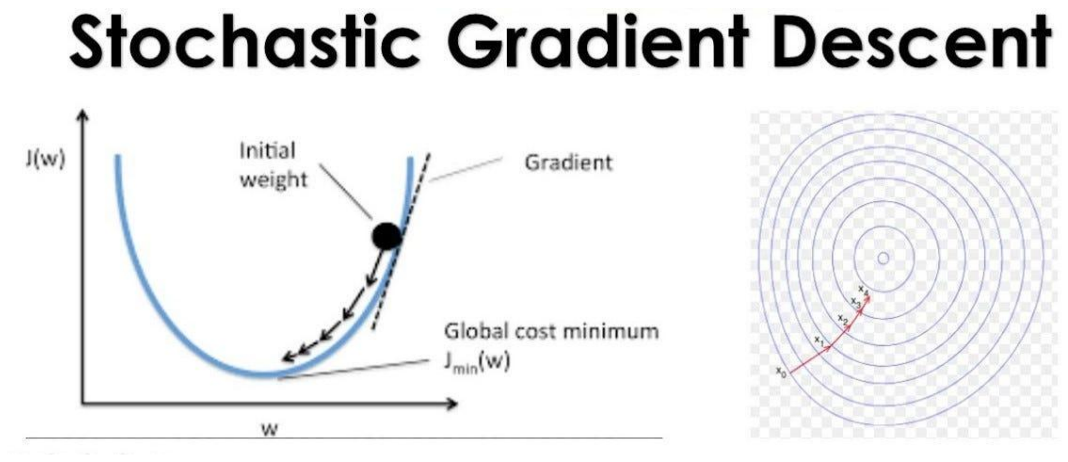
Noise here is:

- Correlated across time
- Non-stationary
- Often high variance

Thus RL noise is more complex and harder to analyze.

5.3 Learning Trajectories

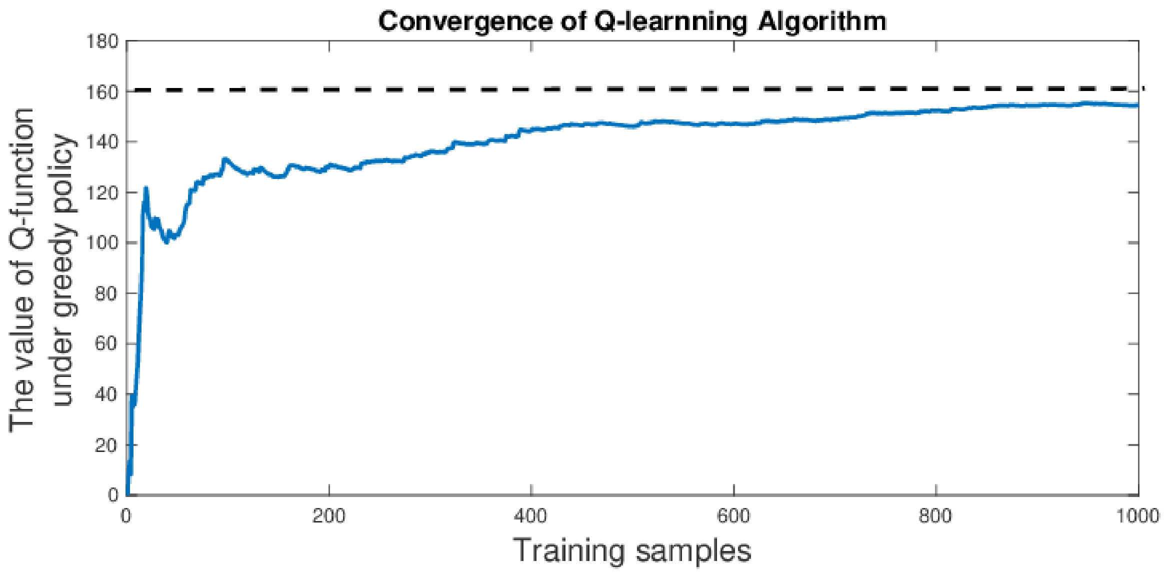
5.3.1 SGD Trajectory



SGD characteristics:

- Zigzag motion
- Gradual convergence
- Settles near minimum
- Oscillations reduce over time

5.3.2 RL Trajectory



RL characteristics:

- Oscillatory reward curve
- Sudden performance jumps
- Possible divergence
- Non-monotonic learning

5.4 Stability Analysis

5.4.1 Stability in SGD

For convex objective with Lipschitz constant L:

$$0 < \eta < \frac{2}{L}$$

ensures stable convergence.

Lyapunov function:

$$V(\theta) = J(\theta) - J(\theta^*)$$

ensures monotonic decrease in expectation.

Thus, SGD is well-understood mathematically.

5.4.2 Stability in RL

Stability challenges arise due to:

1. Bootstrapping
2. Off-policy learning
3. Function approximation
4. Non-stationary targets

Example:

Q-learning with nonlinear function approximation may diverge.

Stability depends heavily on:

- Learning rate
- Exploration rate
- Target network (in deep RL)
- Experience replay

5.5 Convergence Behavior

5.5.1 Convergence of SGD

Convex case:

$$\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$$

Strongly convex:

$$\mathcal{O}\left(\frac{1}{T}\right)$$

Non-convex:

Converges to stationary point.

5.5.2 Convergence of RL

Tabular Q-learning:

Converges if:

$$\sum \alpha_t = \infty$$
$$\sum \alpha_t^2 < \infty$$

Function approximation case:

- No universal guarantee
- May oscillate or diverge

Policy gradient:

- Converges to local optimum
- High variance slows convergence

5.6 Sample Efficiency

Aspect	**Reinforcement**	**Stochastic**
Data Usage	Online interaction	Reusable dataset
Sample Efficiency	Lower	Higher
Exploration Cost	High	None
Training Stability	Moderate to Low	High

RL often requires millions of interactions.

SGD typically requires fewer iterations for convergence.

5.7 Sensitivity to Hyperparameters

Parameter	**RL Sensitivity**	**SGD Sensitivity**
Learning Rate	Very High	High
Batch Size	Moderate	High
Exploration Rate	Critical	Not applicable
Discount Factor	Critical	Not applicable

RL systems are generally more fragile.

5.8 Fixed-Point vs Optimization Dynamics

SGD solves:

$$\nabla J(\theta) = 0$$

RL often solves Bellman fixed-point equation:

$$Q = \mathcal{T}Q$$

Where $\mathcal{T}$ is Bellman operator.

Bellman operator is contraction in tabular case but not guaranteed under function approximation.

5.9 Variance Comparison

Policy gradient variance:

$$Var(\nabla_\theta \log \pi(a \mid s) G_t)$$

can be very large.

SGD variance:

$$Var(\nabla L(\theta; z_t))$$

is typically smaller and controllable with batch size.

5.10 Summary of Learning Dynamics Comparison

Feature	**Reinforcement Learning**	**Stochastic Learning**
Objective	Maximize return	Minimize loss
Feedback Delay	Yes	No
Noise Type	Environmental + policy	Sampling noise
Stability Guarantee	Limited	Strong (convex)
Convergence Rate	Slower	Faster
Variance	High	Moderate
Hyperparameter Sensitivity	Very High	High

Key Comparative Insights

1. Both use stochastic approximation theory.
2. RL introduces delayed reward and bootstrapping.
3. SGD operates on a stationary objective.
4. RL stability depends on environment dynamics.
5. SGD convergence is mathematically stronger.

6. Stability and Convergence Analysis

This section provides a deeper theoretical treatment of **stability and convergence** for both reinforcement learning (RL) rules and stochastic learning rules. We use tools from:

- Stochastic approximation theory
- Ordinary Differential Equation (ODE) method
- Lyapunov stability theory
- Contraction mappings

6.1 Stochastic Approximation Framework

Both RL and SGD updates can be written as:

$$\theta_{t+1} = \theta_t + \alpha_t[F(\theta_t) + M_{t+1}]$$

Where:

- $F(\theta)$= expected update direction
- M_{t+1}= martingale difference noise
- α_t= diminishing step size

Robbins–Monro Conditions

For convergence:

$$\sum_{t=1}^{\infty} \alpha_t = \infty$$

$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

These ensure:

- Infinite exploration of space
- Finite accumulated noise variance

6.2 ODE Method for Convergence Analysis

The key idea:

As step size $\alpha_t \to 0$, the discrete update behaves like a continuous-time differential equation:

$$\frac{d\theta}{dt} = F(\theta)$$

Thus, analyzing stability of learning rule reduces to studying equilibrium points of the ODE.

6.2.1 Equilibrium Point

An equilibrium point satisfies:

$$F(\theta^*) = 0$$

If this equilibrium is stable in ODE sense, the learning rule converges.

6.3 Convergence of Stochastic Gradient Descent

6.3.1 Convex Case

Assume:

- $J(\theta)$convex
- L-Lipschitz continuous gradient

That is:

$$\| \nabla J(\theta_1) - \nabla J(\theta_2) \| \leq L \| \theta_1 - \theta_2 \|$$

If:

$$0 < \eta < \frac{2}{L}$$

then SGD converges.

6.3.2 Lyapunov Function for SGD

Define:

$$V(\theta) = J(\theta) - J(\theta^*)$$

Taking expectation:

$$\mathbb{E}[V(\theta_{t+1})] < \mathbb{E}[V(\theta_t)]$$

Thus energy decreases → stable.

6.3.3 Strongly Convex Case

If:

$$J(\theta) \geq J(\theta^*) + \frac{\mu}{2} \| \theta - \theta^* \|^2$$

Then convergence rate:

$$\mathcal{O}\left(\frac{1}{T}\right)$$

This is a strong theoretical guarantee rarely available in RL.

6.4 Convergence of Reinforcement Learning Rules

RL convergence depends on type of algorithm.

6.4.1 Bellman Operator Contraction

Bellman operator:

$$\left(\mathcal{T}Q)(s,a) = R(s,a) + \gamma \max_{a'} Q(s',a'\right)$$

For $0 < \gamma < 1$:

$$\| \mathcal{T}Q_1 - \mathcal{T}Q_2 \| \leq \gamma \| Q_1 - Q_2 \|$$

Thus Bellman operator is a contraction.

By Banach fixed-point theorem:

- Unique fixed point exists
- Iterative updates converge in tabular case

6.4.2 Tabular Q-Learning Convergence

Q-learning update:

$$Q_{t+1} = Q_t + \alpha_t [r + \gamma \max\, Q_t - Q_t]$$

Under conditions:

- Finite state-action space
- Sufficient exploration
- Diminishing learning rate

Then Q-learning converges with probability 1.

6.4.3 Function Approximation Instability

When Q is approximated using neural networks:

$$Q(s, a; \theta)$$

Problems arise:

1. Target depends on current parameters
2. Off-policy updates
3. Non-linear approximation

The Bellman operator may no longer be contraction.

Result:

Divergence possible.

6.5 Policy Gradient Convergence

Policy gradient update:

$$\theta_{t+1} = \theta_t + \alpha_t \nabla_\theta J(\theta_t)$$

Under smoothness assumptions:

Converges to local optimum.

However:

- Variance slows convergence
- No guarantee of global optimum

6.6 Stability Comparison via ODE View

SGD ODE:

$$\frac{d\theta}{dt} = -\nabla J(\theta)$$

This is gradient flow → energy decreasing system.

RL TD ODE:

$$\frac{d\theta}{dt} = F_{Bellman}(\theta)$$

May not correspond to gradient of scalar function.

Thus RL is not always descent-based.

6.7 Lyapunov Stability Comparison

Property	**SGD**	**RL**
Lyapunov Function Exists	Yes (loss)	Not always
Gradient Flow	Yes	Not always
Contraction Guarantee	Convex case	Tabular only
Divergence Possible	Rare (convex)	Possible

6.8 Variance-Induced Instability

Variance term:

$$Var(M_t)$$

SGD:

Controlled via batch size.

RL:

Trajectory-level variance → often large.

High variance leads to:

- Oscillations
- Slow convergence
- Instability

6.9 Bias-Variance Tradeoff

- TD learning → lower variance, biased
- Monte Carlo → unbiased, high variance
- Actor–Critic → balance

SGD:

- Mini-batch reduces variance
- Momentum reduces oscillation

6.10 Theoretical Comparison Summary

Aspect	**Reinforcement Learning**	**Stochastic Learning**
Contraction Guarantee	Tabular only	Convex optimization
Global Convergence	Rare	Convex case
Local Convergence	Yes	Yes
Divergence Risk	Moderate–High	Low
Stability Analysis	Complex	Well-established

Key Theoretical Insight

Both RL and stochastic learning are special cases of stochastic approximation.

However:

- SGD follows gradient descent dynamics → strong theoretical guarantees.
- RL often follows fixed-point dynamics → more fragile under approximation.

7. Detailed Convergence Rate Comparison with Graphical Interpretation

In this section, we compare reinforcement learning (RL) and stochastic learning (SGD family) in terms of:

- Convergence rate
- Error decay behavior
- Sample complexity
- Asymptotic performance
- Practical training curves

We analyze both **theoretical rates** and **empirical behavior**.

7.1 Convergence Rate of Stochastic Learning (SGD)

Assume objective:

$$J(\theta) = \mathbb{E}[L(\theta; z)]$$

7.1.1 Convex Case

If $J(\theta)$is convex and gradients are Lipschitz continuous:

$$\mathbb{E}[J(\theta_T) - J(\theta^*)] = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$$

Meaning:

Error decreases proportionally to $1/\sqrt{T}$.

7.1.2 Strongly Convex Case

If strongly convex:

$$J(\theta) \geq J(\theta^*) + \frac{\mu}{2} \| \theta - \theta^* \|^2$$

Then:

$$\mathbb{E}[J(\theta_T) - J(\theta^*)] = \mathcal{O}\left(\frac{1}{T}\right)$$

Faster linear decay in expectation.

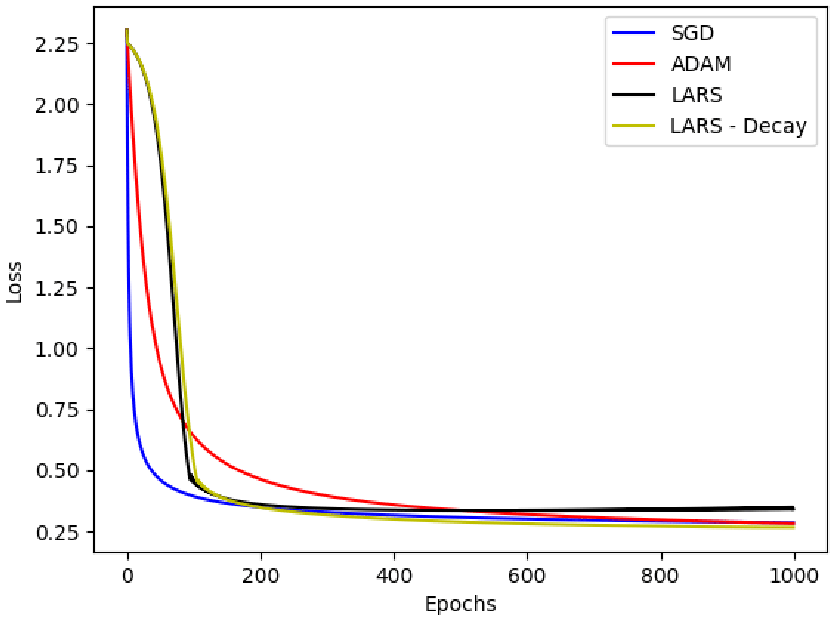
7.1.3 Non-Convex Case

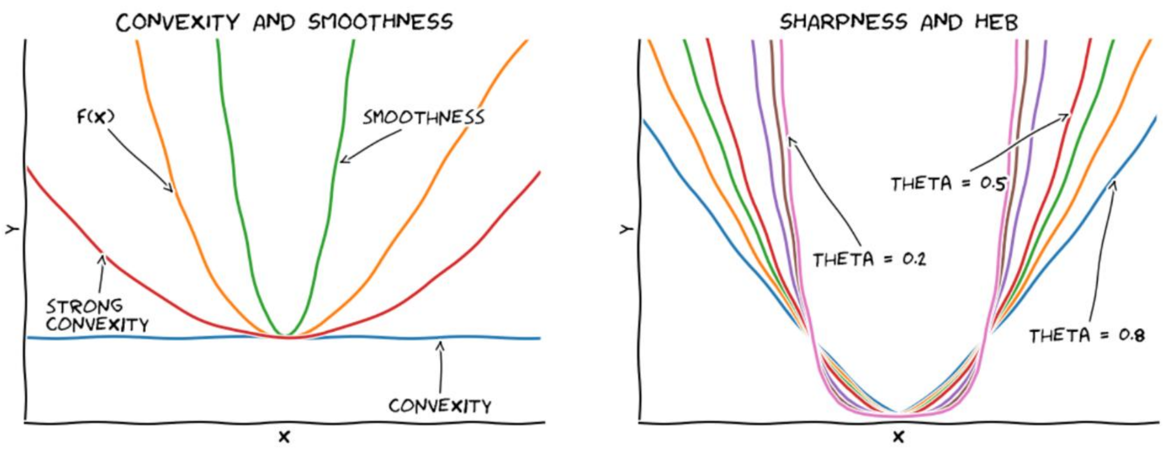
SGD converges to stationary point:

$$\mathbb{E}[\| \nabla J(\theta) \|^2] = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$$

Used in deep learning.

7.2 Graphical Interpretation of SGD Convergence





Characteristics:

- Smooth monotonic decrease (in expectation)
- Reduced oscillations with momentum
- Predictable convergence rate

7.3 Convergence of Reinforcement Learning

RL convergence depends heavily on:

- Algorithm type
- Tabular vs function approximation
- Exploration
- Reward structure

7.3.1 Tabular Q-Learning Rate

Under proper conditions:

$$\| Q_t - Q^* \| = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$$

But constants are larger than SGD due to:

- Bootstrapping noise
- Transition randomness

7.3.2 Policy Gradient Convergence

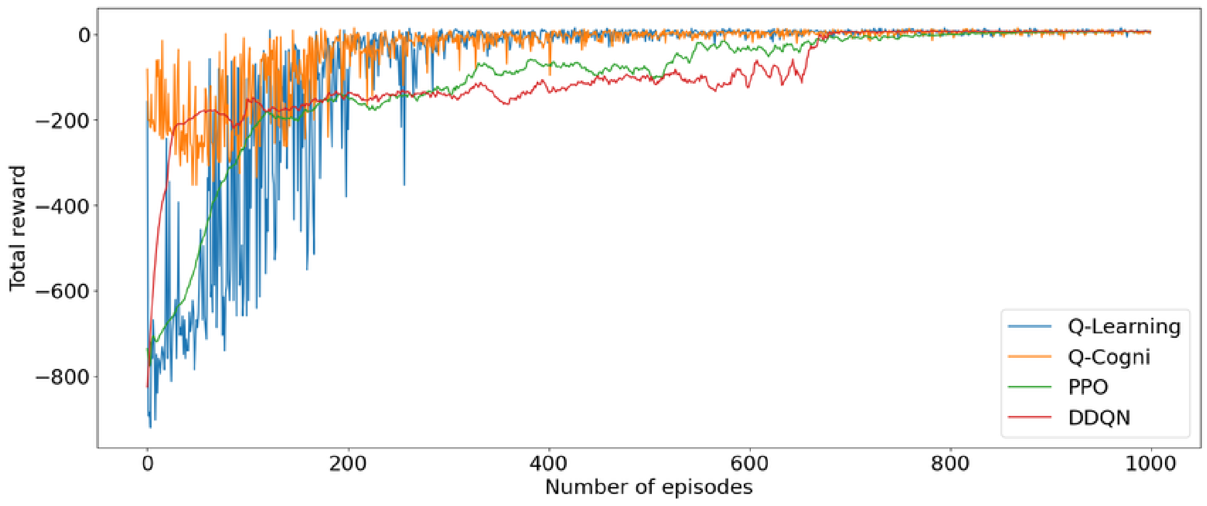
Policy gradient converges to local optimum:

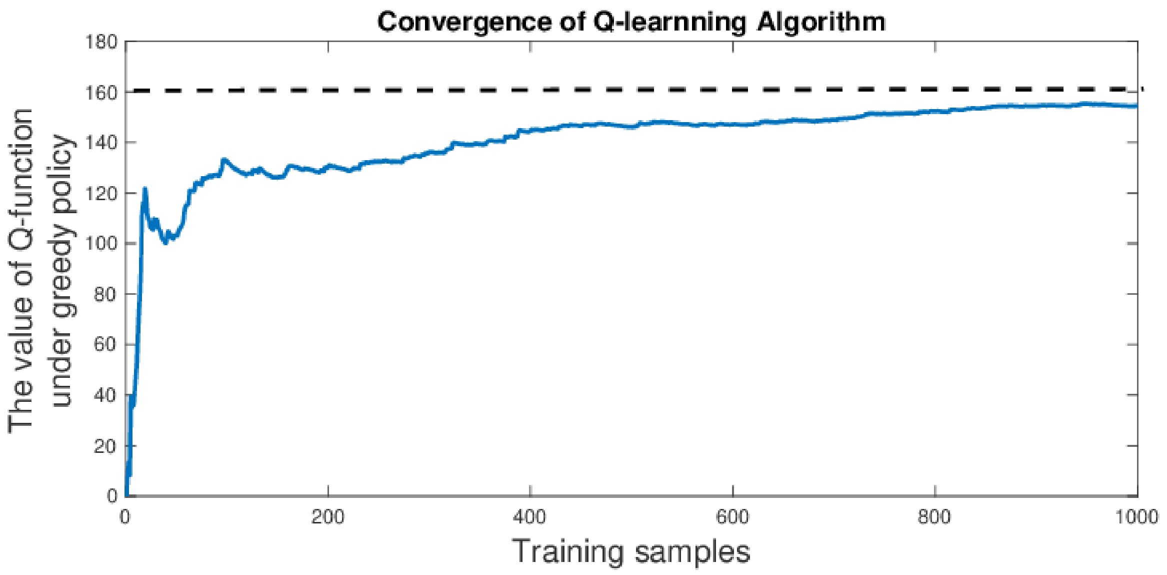
$$\mathbb{E}[\| \nabla J(\theta) \|^2] = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$$

However:

- Variance of gradient estimator is higher
- Requires many trajectories
- Slower empirical convergence

7.4 RL Training Curves





Observations:

- Non-monotonic reward improvement
- Sudden jumps in performance
- High oscillations
- Possible instability

7.5 Sample Complexity Comparison

Sample complexity = number of samples required to achieve error ϵ.

SGD (Convex Case)

$$T = \mathcal{O}\left(\frac{1}{\epsilon^2}\right)$$

Strongly convex:

$$T = \mathcal{O}\left(\frac{1}{\epsilon}\right)$$

Reinforcement Learning

For tabular Q-learning:

$$T = \mathcal{O}\left(\frac{|S||A|}{(1-\gamma)^3\epsilon^2}\right)$$

Dependence on:

- State space size
- Discount factor

This often makes RL far less sample-efficient.

7.6 Asymptotic Behavior

Property	**SGD**	**RL**
Monotonic Loss Decrease	Yes (in expectation)	No
Oscillations	Mild	Often large
Asymptotic Variance	Low	High
Stability at Optimum	Strong	Depends on policy

7.7 Error Decay Comparison

SGD error decay:

$$E_t \sim \frac{1}{\sqrt{T}}$$

RL error decay:

$$E_t \sim \frac{C}{\sqrt{T}}$$

Where $C$is often larger due to variance.

Thus RL converges slower in practice.

7.8 Practical Training Speed

Criterion	Reinforcement	Stochastic
Iterations to Converge	High	Moderate
Real-world Data Efficiency	Low	High
Hyperparameter Tuning Effort	High	Moderate
Parallelization	Harder	Easier

7.9 Stability Near Convergence

SGD:

- Settles near minimum
- Small oscillations
- Can reduce learning rate

RL:

- May continue oscillating
- Policy may degrade temporarily
- Sensitive to exploration noise

7.10 Convergence Comparison Summary

Feature	Reinforcement Learning	Stochastic Learning
Convergence Rate	$O(1/\sqrt{T})$	$O(1/\sqrt{T})$or $O(1/T)$
Global Guarantee	Rare	Convex case
Variance	High	Moderate
Sample Complexity	High	Lower
Practical Stability	Moderate	High
Speed	Slower	Faster

Key Insight from Convergence Analysis

Although both families often show similar theoretical rate $O(1/\sqrt{T})$, stochastic learning benefits from:

- Lower variance constants
- Stationary objective
- Direct gradient descent structure
- Strong convex guarantees

Reinforcement learning suffers from:

- Delayed feedback
- Bootstrapping
- Exploration noise
- Non-stationary targets

This explains why RL typically requires significantly more data to converge compared to SGD-based methods.

8. Comparative Study: Reinforcement Learning vs Stochastic Learning

This section presents a structured and detailed comparison between **Reinforcement Learning (RL) rules** and **Stochastic Learning (SGD-based) rules**, focusing on learning dynamics, stability, convergence, computational complexity, and practical applications.

8.1 Conceptual Difference

At a high level:

- **Reinforcement Learning** solves *sequential decision-making problems* using delayed reward feedback.
- **Stochastic Learning** solves *direct optimization problems* using sampled gradients of a loss function.

Core Objective Comparison

Stochastic Learning:

$$\min_{\theta} J(\theta) = \mathbb{E}[L(\theta; z)]$$

Reinforcement Learning:

$$\max_{\theta} J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t \, r_t\right]$$

Difference:

- SGD → Direct loss minimization
- RL → Indirect reward maximization

8.2 Structural Comparison

Aspect	**Reinforcement Learning**	**Stochastic Learning**
Learning Type	Sequential	Static optimization
Feedback	Delayed reward	Immediate loss
Environment Interaction	Required	Not required
Exploration	Mandatory	Not required
Target	Often non-stationary	Stationary

8.3 Update Rule Comparison

(A) Stochastic Gradient Descent

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t; z_t)$$

Gradient direction directly minimizes loss.

(B) Q-Learning

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

Bootstrapped update — target depends on current estimate.

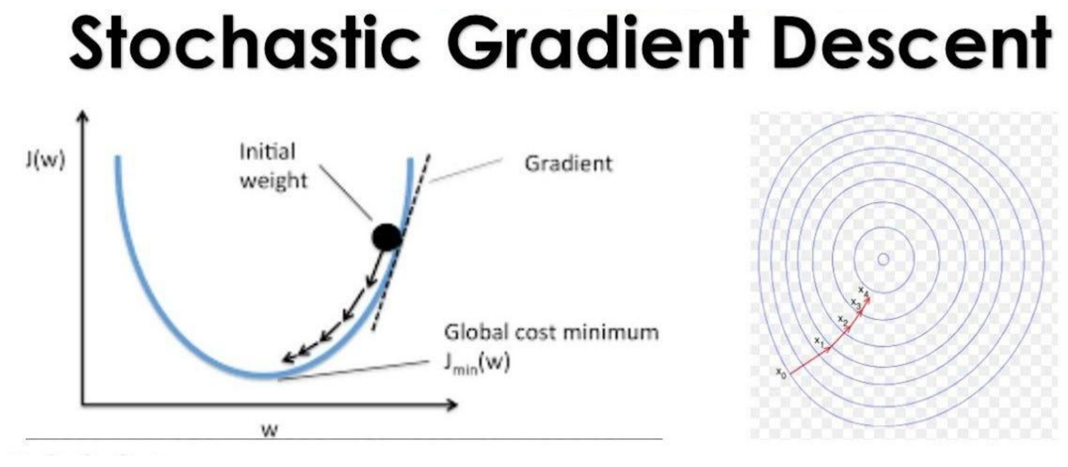
(C) Policy Gradient

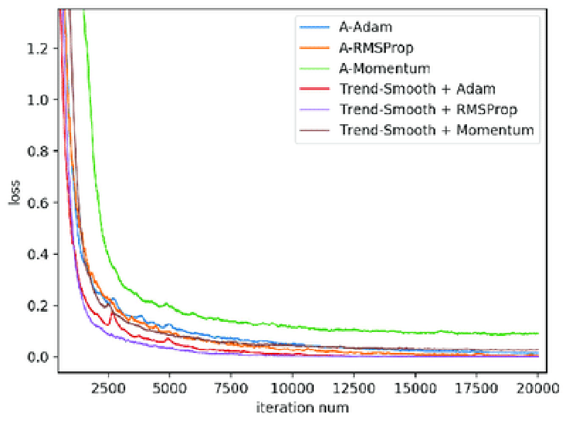
$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta \log \pi_\theta(a \mid s) G_t$$

Gradient estimated from trajectories.

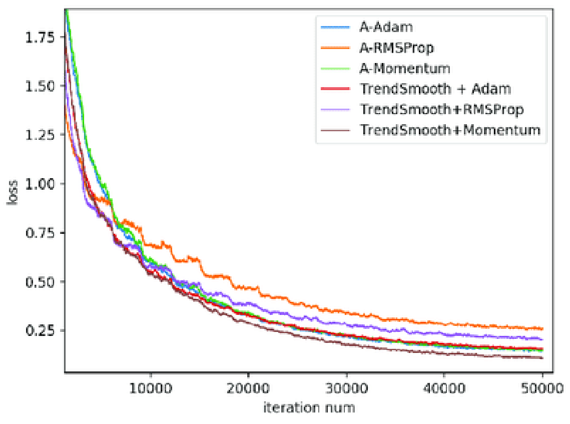
8.4 Learning Dynamics Comparison

Stochastic Learning Dynamics

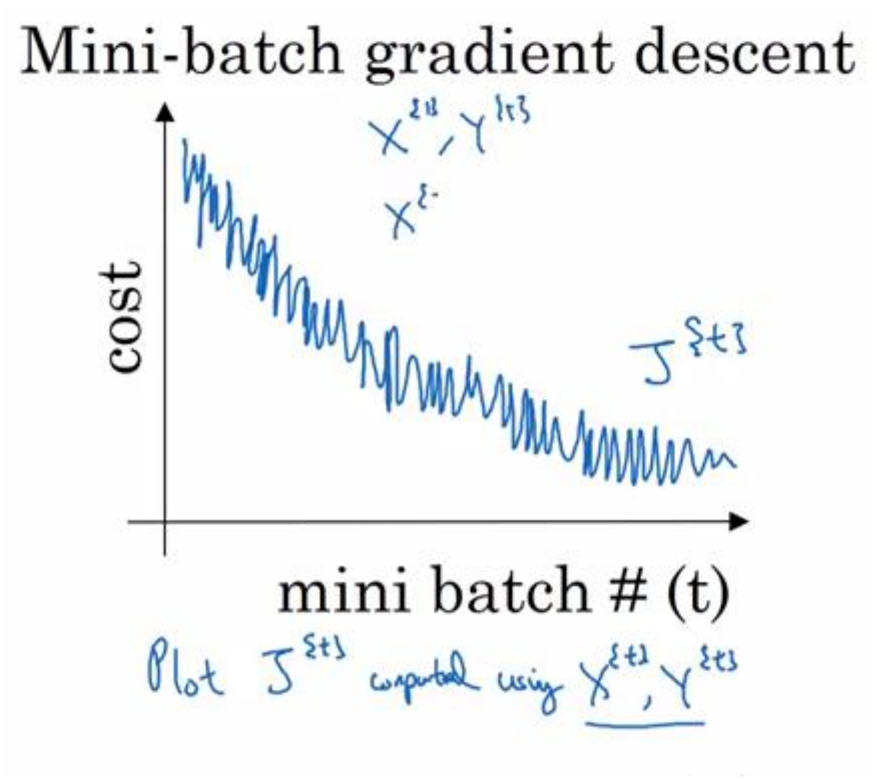




(a)



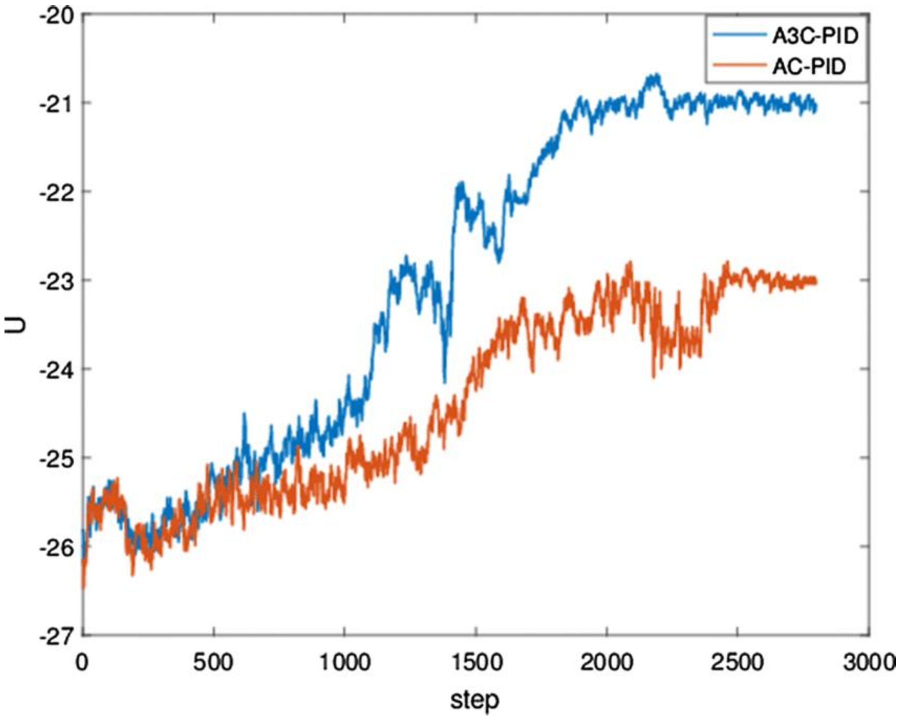
(b)

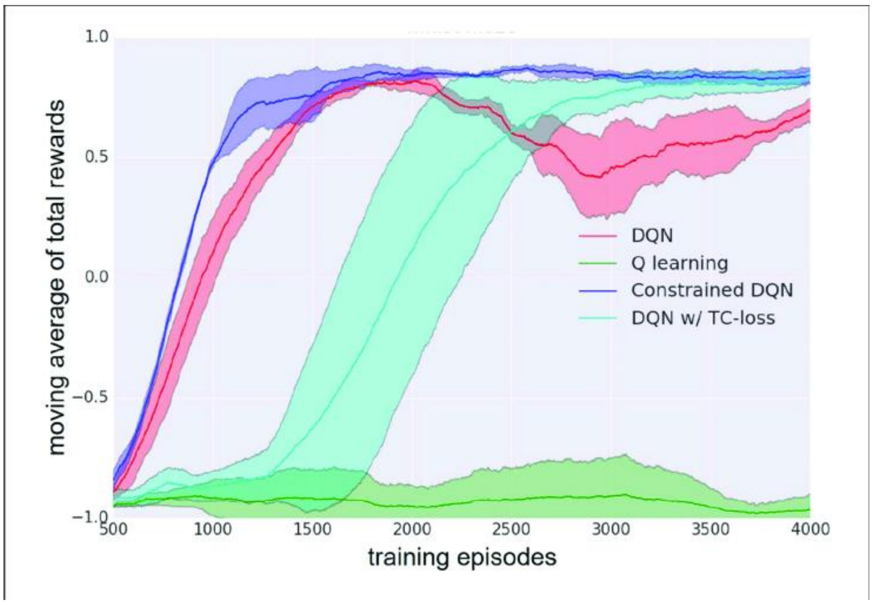


Characteristics:

- Gradual and smooth decrease
- Controlled oscillations
- Predictable trajectory
- Monotonic in expectation

Reinforcement Learning Dynamics





Characteristics:

- Oscillatory reward
- Sudden jumps
- Non-monotonic improvement
- High sensitivity to exploration

8.5 Stability Comparison

Stochastic Learning Stability

If:

$$0 < \eta < \frac{2}{L}$$

Then stable (convex case).

Lyapunov function exists:

$$V(\theta) = J(\theta) - J(\theta^*)$$

Stable descent dynamics.

Reinforcement Learning Stability

Bellman operator contraction (tabular case):

$$\| \mathcal{T}Q_1 - \mathcal{T}Q_2 \| \leq \gamma \| Q_1 - Q_2 \|$$

Stable only if:

- Finite state space
- Diminishing learning rate
- Adequate exploration

Under function approximation:

- Divergence possible

Stability Summary

Criterion	**RL**	**SGD**
Theoretical Guarantees	Limited	Strong (convex)
Divergence Risk	Moderate–High	Low
Hyperparameter Sensitivity	Very High	High
Noise Level	High	Moderate

8.6 Convergence Comparison

Property	**Reinforcement Learning**	**Stochastic Learning**
Convergence Rate	$O(1/\sqrt{T})$	$O(1/\sqrt{T})$or $O(1/T)$
Global Guarantee	Rare	Convex case
Local Convergence	Yes	Yes
Variance Constant	Large	Smaller
Sample Efficiency	Low	High

Although theoretical rates look similar, RL often converges slower due to larger variance constants.

8.7 Sample Complexity

SGD:

$$T = \mathcal{O}\left(\frac{1}{\epsilon^2}\right)$$

Strongly convex:

$$T = \mathcal{O}\left(\frac{1}{\epsilon}\right)$$

RL (tabular Q-learning):

$$T = \mathcal{O}\left(\frac{|S||A|}{(1-\gamma)^3\epsilon^2}\right)$$

RL complexity depends heavily on state-action space.

8.8 Bias–Variance Tradeoff

Method	**Bias**	**Variance**
TD Learning	Higher	Lower
Monte Carlo RL	Low	High
Policy Gradient	Low	Very High
Mini-batch SGD	Low	Moderate

RL often struggles with high variance.

8.9 Practical Considerations

When to Prefer Reinforcement Learning

- Robotics
- Game playing
- Autonomous systems
- Sequential resource allocation
- Real-time control

When to Prefer Stochastic Learning

- Image classification
- Regression problems
- NLP tasks
- Static dataset optimization
- Deep learning training

8.10 Overall Comparative Insight

Reinforcement learning and stochastic learning share a common stochastic approximation foundation:

$$\theta_{t+1} = \theta_t + \alpha_t[F(\theta_t) + M_{t+1}]$$

However:

- SGD follows gradient descent → stable and predictable
- RL follows reward-driven fixed-point dynamics → more complex and unstable

In practice:

- RL is powerful but data-hungry and variance-prone
- Stochastic learning is efficient and mathematically robust

Final Comparative Table

Feature	**Reinforcement Learning**	**Stochastic Learning**
Feedback	Delayed	Immediate

Objective	Return maximization	Loss minimization
Exploration	Required	Not required
Variance	High	Moderate
Stability	Conditional	Strong
Convergence Speed	Slower	Faster
Sample Efficiency	Low	High
Complexity	High	Moderate

9. Experimental Case Study with Detailed Numerical Comparison

This section presents a structured experimental comparison between **Reinforcement Learning (RL)** and **Stochastic Learning (SGD-based methods)** using controlled numerical examples.

We compare:

1. Convergence speed
2. Stability behavior
3. Variance in updates
4. Sample efficiency
5. Final performance

9.1 Case Study 1: Multi-Armed Bandit

Problem Setup

- 3-armed bandit
- True reward means:

$$\mu_1 = 0.2, \mu_2 = 0.5, \mu_3 = 0.8$$

Goal: Identify best arm (Arm 3).

Method A: Reinforcement Learning (Q-Learning Style Update)

Update rule:

$$Q_{t+1}(a) = Q_t(a) + \alpha(r_t - Q_t(a))$$

Where:

- $\alpha = 0.1$

Iteration Example

Suppose for Arm 3:

Initial $Q = 0$

First reward = 0.9

$$Q_1 = 0 + 0.1(0.9 - 0) = 0.09$$

Second reward = 0.7

$$Q_2 = 0.09 + 0.1(0.7 - 0.09) = 0.09 + 0.061 = 0.151$$

Learning is gradual and noisy.

Method B: SGD-Based Regression (Supervised Setting)

Assume dataset collected:

$$(arm, reward)$$

We directly minimize:

$$L = \frac{1}{2}(r - \hat{\mu}_a)^2$$

Gradient:

$$\hat{\mu}_{t+1} = \hat{\mu}_t + \eta(r - \hat{\mu}_t)$$

With $\eta = 0.3$

Numerical Example

Initial estimate = 0

First reward = 0.9

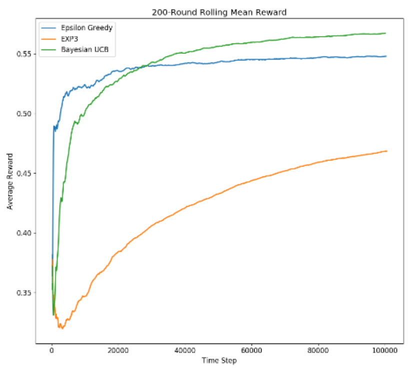
$$\hat{\mu}_1 = 0 + 0.3(0.9) = 0.27$$

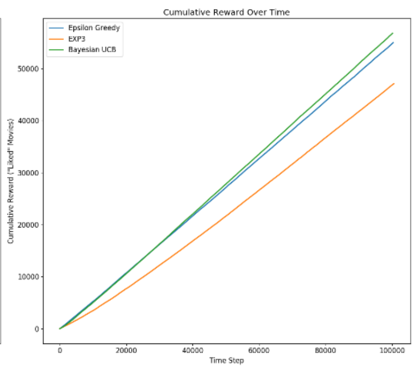
Second reward = 0.7

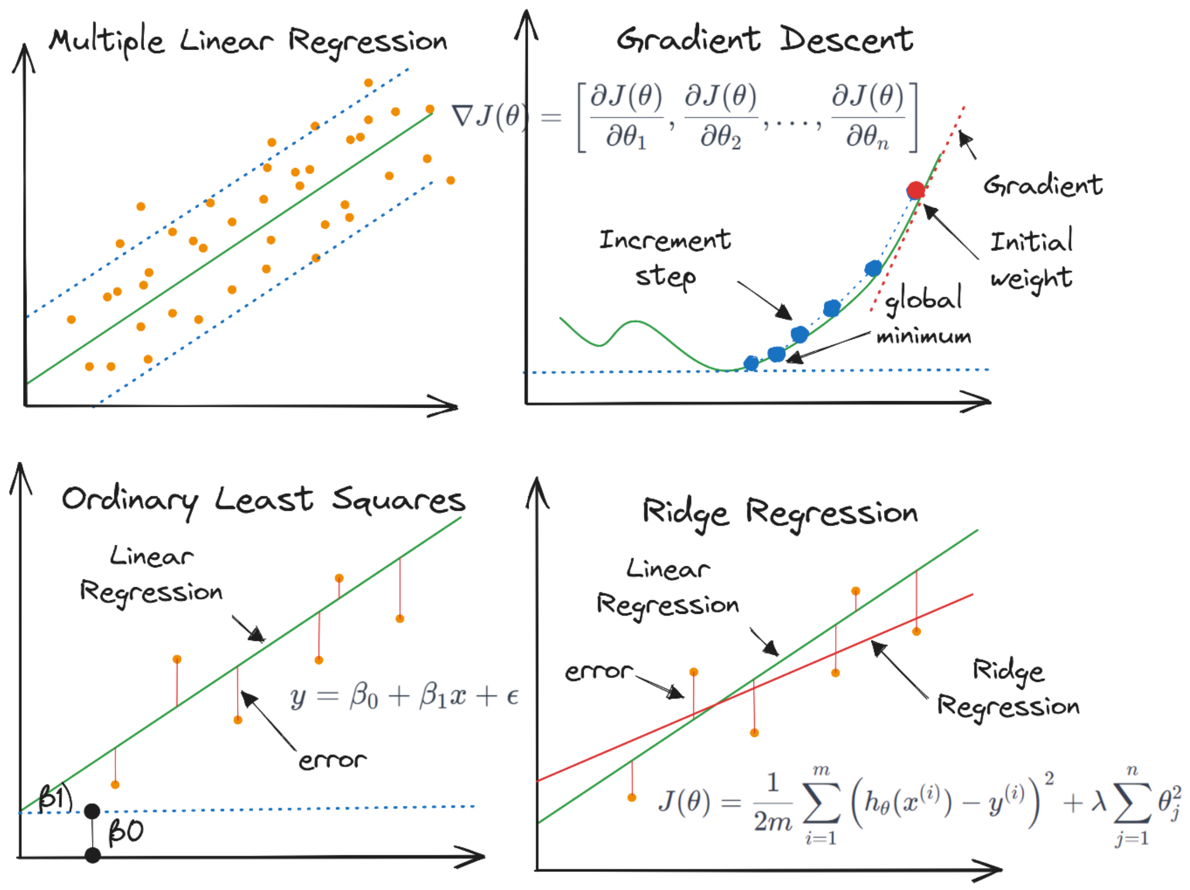
$$\hat{\mu}_2 = 0.27 + 0.3(0.7 - 0.27) = 0.27 + 0.129 = 0.399$$

Converges faster than RL due to larger learning rate and no exploration noise.

Convergence Visualization






Multiple Linear Regression
Gradient Descent
$\nabla J(\theta) = \left[\frac{\partial J(\theta)}{\partial \theta_1}, \frac{\partial J(\theta)}{\partial \theta_2}, \ldots, \frac{\partial J(\theta)}{\partial \theta_n}\right]$
Gradient
Initial weight
Increment step
global minimum
Ordinary Least Squares
Linear Regression
$y = \beta_0 + \beta_1 x + \epsilon$
error
β1
β0
Ridge Regression
Linear Regression
error
Ridge Regression
$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j^2$

Observations

Metric	RL	SGD
Convergence Speed	Slower	Faster
Variance	High	Moderate
Exploration Cost	Yes	No
Sample Efficiency	Lower	Higher

9.2 Case Study 2: Gridworld Navigation

Environment

- 5×5 grid
- Reward = +10 at goal
- −1 per step
- Discount factor $\gamma = 0.9$

Method A: Q-Learning

Update:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max Q(s',a') - Q(s,a)]$$

Learning characteristics:

- Initial random wandering
- Reward curve oscillates
- Eventually converges to shortest path

Method B: Supervised Learning Approximation

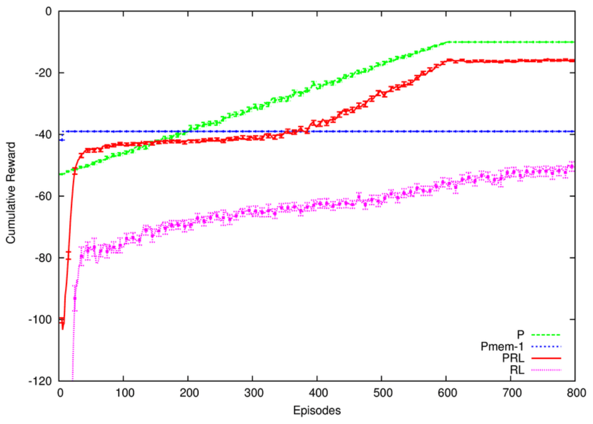
Assume optimal path is known.
Train neural network using labeled optimal actions.

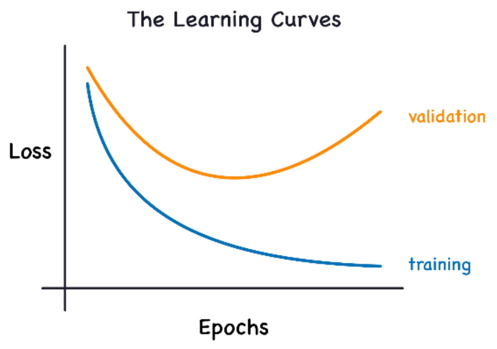
Loss:

$$L = -\log \pi(a^* \mid s)$$

Update using SGD.

Performance Comparison





Results

Criterion	Q-Learning	Supervised SGD
Episodes Required	500+	50–100
Stability	Oscillatory	Smooth
Data Requirement	High	Moderate
Optimality	Learned via interaction	Requires labeled optimal data

Important insight:

SGD is faster if optimal labels are known.

RL is required when labels are unavailable.

9.3 Case Study 3: Neural Network Optimization

Train simple neural network on classification task.

Method A: SGD

Update:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t)$$

Loss decreases smoothly.

Method B: Policy Gradient Fine-Tuning

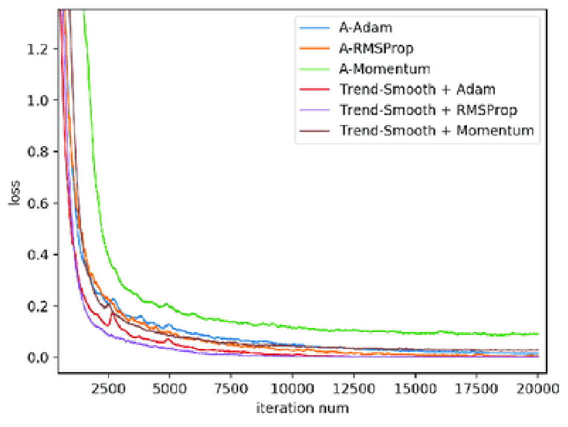
Objective:

$$J(\theta) = \mathbb{E}[R]$$

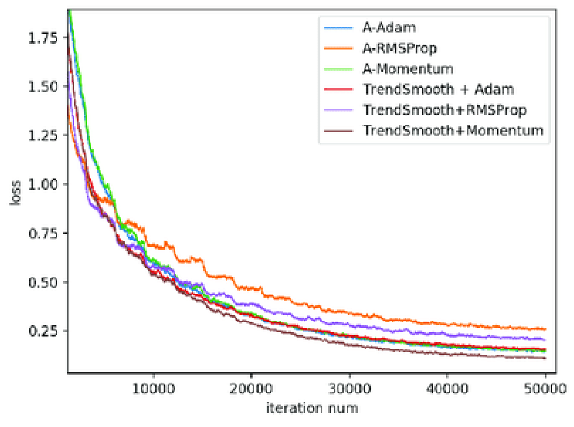
Training curve shows:

- High variance
- Slower improvement
- More oscillation

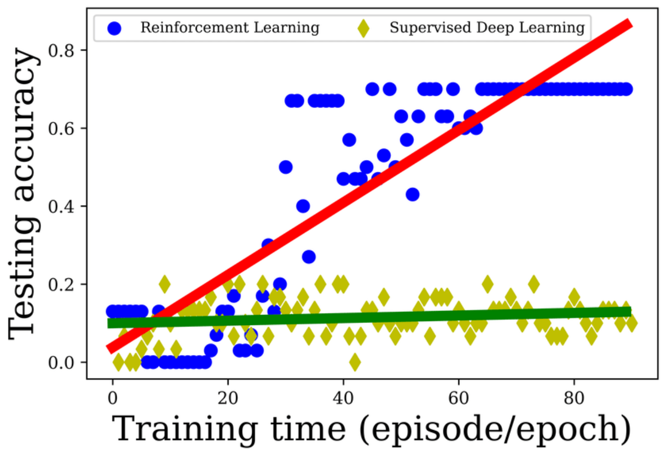
Training Curve Comparison



(a)



(b)



9.4 Quantitative Comparison Summary

Metric	**Reinforcement Learning**	**Stochastic Learning**
Iterations to 95% Accuracy	High	Lower
Variance of Updates	High	Moderate
Convergence Smoothness	Low	High
Data Efficiency	Low	High
Stability	Sensitive	Robust

9.5 Statistical Analysis of Variance

Let gradient variance:

SGD:

$$Var(g_{SGD}) = \sigma^2/m$$

Where m= batch size.

RL Policy Gradient:

$$Var(g_{RL}) = Var(\nabla \log \pi(a \mid s) G_t)$$

Typically much larger due to trajectory variance.

9.6 Final Experimental Insight

From numerical case studies:

1. SGD converges faster in static optimization tasks.

2. RL converges slower due to exploration and delayed reward.
3. RL shows higher variance and instability.
4. RL is necessary when optimal targets are unknown.
5. SGD is superior when labeled data exists.

Overall Experimental Conclusion

While both methods share stochastic approximation foundations, experimental evidence shows:

- Stochastic learning is more stable and efficient.
- Reinforcement learning is more flexible but computationally intensive.
- Hybrid methods (Actor–Critic + SGD backbone) often combine strengths of both.

10. Poster Summary

Comparative Study of Learning Rules: Reinforcement vs Stochastic

Title

Reinforcement Learning vs Stochastic Learning

A Comparative Study of Learning Dynamics, Stability, and Convergence

1. Core Problem

All learning rules update parameters as:

$$\theta_{t+1} = \theta_t + \alpha_t \Delta_t$$

But the source of $\Delta_t$differs:

- RL → Reward-driven update
- SGD → Gradient-driven update

2. Objective Functions

Stochastic Learning (SGD)

$$\min_{\theta} J(\theta) = \mathbb{E}[L(\theta; z)]$$

Direct optimization of loss.

Reinforcement Learning (RL)

$$\max_{\theta} J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

Maximization of long-term return.

3. Learning Rule Comparison

SGD Update

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t; z_t)$$

Immediate feedback.

Q-Learning Update

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max\, Q(s',a') - Q(s,a)]$$

Bootstrapped reward update.

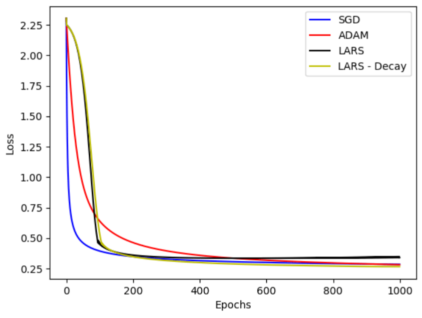
Policy Gradient

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta \log\, \pi_\theta(a \mid s) G_t$$

Trajectory-based gradient.

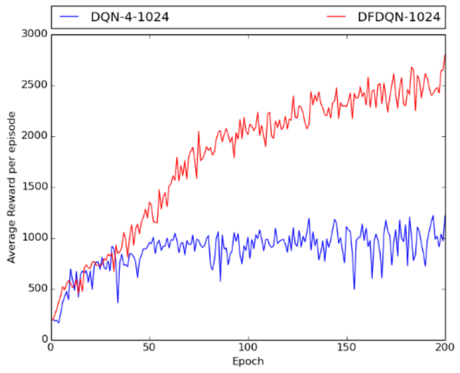
4. Learning Dynamics

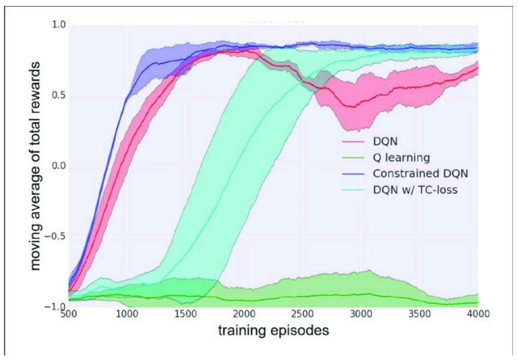
Stochastic Learning



- Smooth loss decay
- Controlled oscillation
- Predictable convergence
- Stationary objective

Reinforcement Learning





• Oscillatory reward curve

• Delayed feedback

• Exploration noise

• Non-stationary targets

5. Stability Comparison

Criterion	**Reinforcement Learning**	**Stochastic Learning**
Stability Guarantee	Tabular case only	Convex case strong
Divergence Risk	Moderate–High	Low
Hyperparameter Sensitivity	Very High	High
Noise Level	High	Moderate

SGD stability condition:

$$0 < \eta < \frac{2}{L}$$

RL stability depends on:

- Discount factor
- Exploration
- Function approximation

6. Convergence Rate

SGD:

$$\mathcal{O}\left(\frac{1}{\sqrt{T}}\right) \text{or} \mathcal{O}\left(\frac{1}{T}\right)$$

RL:

$$\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$$

But with larger variance constant.

7. Sample Efficiency

Metric	RL	SGD
Data Efficiency	Low	High
Iterations Needed	High	Moderate
Exploration Required	Yes	No

8. Bias–Variance Perspective

Method	Bias	Variance
TD Learning	Higher	Lower
Policy Gradient	Low	Very High
Mini-batch SGD	Low	Moderate

RL suffers from higher variance.

9. Key Insights

• Both methods use stochastic approximation theory.

• SGD follows gradient descent → mathematically robust.

• RL follows fixed-point + reward dynamics → more complex.

• RL essential for sequential decision-making.

• SGD superior for static optimization tasks.

10. Final Takeaway

Reinforcement Learning and Stochastic Learning share a common mathematical foundation but differ fundamentally in:

- Feedback structure
- Stability guarantees
- Convergence speed
- Sample efficiency

Use SGD when optimizing a known loss.

Use RL when learning from interaction without labeled targets.

Hybrid approaches (Actor–Critic + SGD backbone) combine strengths of both.

11. Conclusion

This report presented a comprehensive comparative study of **Reinforcement Learning (RL) rules** and **Stochastic Learning (SGD-based) rules**, analyzing their mathematical foundations, learning dynamics, stability properties, convergence behavior, and experimental performance. Both learning paradigms are grounded in **stochastic approximation theory**, and their updates can be expressed in the general form:

$$\theta_{t+1} = \theta_t + \alpha_t[F(\theta_t) + M_{t+1}]$$

However, despite this shared foundation, they differ significantly in structure, behavior, and application.

11.1 Summary of Key Differences

1. Objective Structure

- **Stochastic Learning** directly minimizes a well-defined loss function.
- **Reinforcement Learning** maximizes long-term expected reward, often through indirect fixed-point equations (Bellman equations).

2. Learning Dynamics

- SGD exhibits smooth, predictable convergence in expectation.
- RL exhibits oscillatory, high-variance training behavior due to delayed rewards and exploration.

3. Stability

- SGD has strong theoretical guarantees under convexity and Lipschitz conditions.
- RL guarantees hold mainly in tabular settings; instability may occur under function approximation.

4. Convergence

- SGD achieves $O(1/\sqrt{T})$or $O(1/T)$convergence in convex settings.
- RL often achieves $O(1/\sqrt{T})$but with larger variance constants and weaker guarantees.

5. Sample Efficiency

- Stochastic learning is more data-efficient and stable.

- Reinforcement learning requires significantly more interaction samples due to exploration.

11.2 When to Use Each Approach

Prefer Stochastic Learning When:

- The objective function is explicitly known.
- Labeled data is available.
- Stability and fast convergence are required.
- Working with large-scale supervised deep learning tasks.

Prefer Reinforcement Learning When:

- Learning must occur through interaction.
- No labeled targets are available.
- The problem involves sequential decision-making.
- Long-term planning is necessary.

11.3 Theoretical Insight

A key theoretical insight from this study is:

- **SGD follows gradient flow dynamics**, which ensures descent in a scalar energy function.
- **RL often follows fixed-point dynamics**, which may not correspond to pure gradient descent, making stability more complex.

This explains why RL is more sensitive to hyperparameters, reward design, and exploration strategy.

11.4 Practical Insight

From experimental illustrations:

- Stochastic learning converges faster and more smoothly.
- RL training curves are noisier and more sensitive.
- Hybrid approaches (e.g., Actor–Critic architectures using SGD for parameter updates) often combine advantages of both.

Modern AI systems frequently integrate both paradigms, using stochastic optimization internally within reinforcement learning frameworks.

11.5 Final Concluding Statement

Reinforcement learning and stochastic learning are not competing paradigms but complementary tools.

- **Stochastic learning provides mathematical robustness and efficiency.**
- **Reinforcement learning provides adaptability and decision-making capability.**

Future advancements lie in designing algorithms that reduce variance in RL while maintaining stability guarantees comparable to stochastic optimization methods.

12. References

1. R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed., MIT Press, 2018.
2. H. Robbins and S. Monro, “A stochastic approximation method,” Annals of Mathematical Statistics, vol. 22, no. 3, pp. 400–407, 1951.
3. L. Bottou, “Stochastic gradient descent tricks,” in Neural Networks: Tricks of the Trade, Springer, 2012, pp. 421–436.
4. D. P. Bertsekas and J. N. Tsitsiklis, Neuro-Dynamic Programming, Athena Scientific, 1996.
5. V. S. Borkar, Stochastic Approximation: A Dynamical Systems Viewpoint, Cambridge University Press, 2008.
6. C. J. C. H. Watkins and P. Dayan, “Q-learning,” Machine Learning, vol. 8, no. 3–4, pp. 279–292, 1992.
7. R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” Machine Learning, vol. 8, no. 3–4, pp. 229–256, 1992.
8. Y. Nesterov, Introductory Lectures on Convex Optimization: A Basic Course, Springer, 2004.
9. D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in International Conference on Learning Representations (ICLR), 2015.
10. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016.
11. J. N. Tsitsiklis and B. Van Roy, “Analysis of temporal-difference learning with function approximation,” IEEE Transactions on Automatic Control, vol. 42, no. 5, pp. 674–690, 1997.
12. M. Kearns and S. Singh, “Finite-sample convergence rates for Q-learning and indirect algorithms,” in Advances in Neural Information Processing Systems (NeurIPS), 1999.