Reinforcement Learning
Monte Carlo and Temporal Difference Learning

Navneet Kashyap

1 MC and TD methods on Frozen Lake

1.1 Custom Environment

We tested these methods in three environments:

- **C**ustom Frozen Lake - 5*5
- **C**ustom Frozen Lake - 50*50
- **E**xpanded Frozen Lake - 4*4

The expanded frozen lake environment had an additional key that the agent had to collect before reaching the goal. The map for custom frozen lake was generated through another function, which first created a monotonic path with a 3 cell wide corridor and then later filled everything with holes with a given probability.

1.2 Algorithm's performance

Table 1: Reinforcement Learning Algorithm Performance Comparison

Environment	**Algorithm**	**Time (seconds)**	**Average Return**
CustomFrozenLake-v1	Monte Carlo	4.9393	0.8429
	Q-Learning	13.7458	0.8429
	SARSA	8.5974	0.8429
ExpandedFrozenLake-v1	Monte Carlo	2.4597	0.3882
	Q-Learning	3.1243	0.5054
	SARSA	2.3434	0.5188
	Monte Carlo	1.4024	0.9510
	Q-Learning	3.6078	0.9510
	SARSA	1.6445	0.9510
Large-scale Environment (50x50 Grid)			
50x50 Frozen Lake	Q-Learning	1108.3	0.38
	SARSA	1233.14	0.38

Note: Monte Carlo results not available for 50x50 environment due to computational constraints

2 Cliff Walking Experiment

2.1 Setup

The Cliff Walking environment represents a standard grid world with a "cliff" region. The agent receives a large negative reward for falling off the cliff and a small step penalty otherwise. The environment is created completely from scratch without creating a subclass from 'gym.Env'. We implemented:

- **Monte Carlo Control:** Episodic every-visit MC with greedy epsilon policy
- **Q learning:** with epsilon greedy exploration
- **SARSA:** on-policy TD control

Following is the map of our environment with the index and state of each cell mentioned:

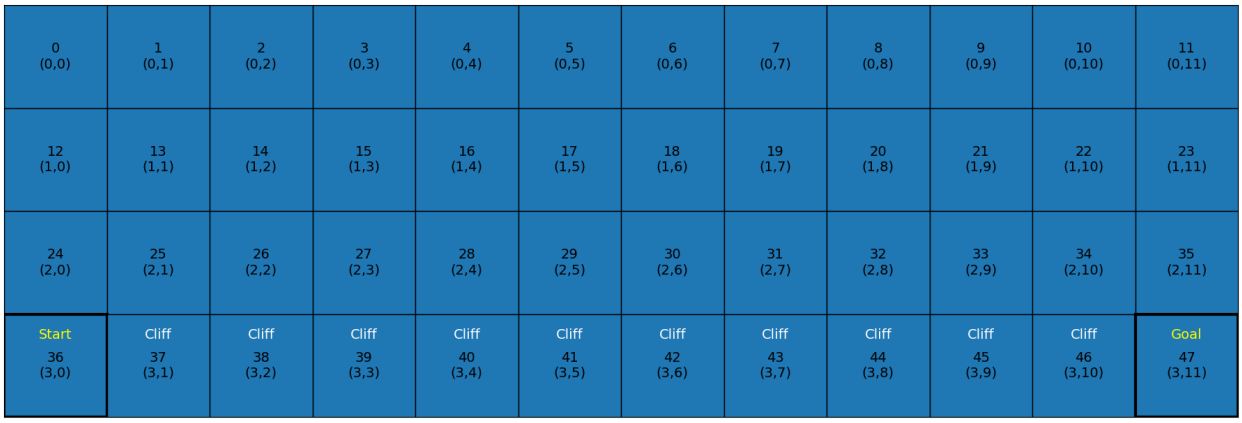

Figure 1: Cliff Walking environment map

2.2 Results

Table 2: Performance Comparison

Algorithm	Episodes	Training Time (s)	Eval. Time (s)	Avg. Return
Monte Carlo	500 000.0000	215.6600	0.1700	−13.9900
Q-Learning	50 000.0000	10.7600	0.1400	−12.2500
SARSA	500 000.0000	108.2600	0.1800	−15.7100

All three methods learn a near-optimal policy taking nearly the same time. Though the optimal path learnt by all of them are significantly different. Figure 2 shows an overview of them :

2.3 Conclusion

We tested all three algorithms on the Cliff Walking experiment and observed the average rewards and the optimal path found by them. We can see that SARSA took the safest path because it is an on-policy method and has no guarantee of choosing the best possible actions again and again. Q learning on the other hand being off policy chooses the path maximizing the return that is the shortest path hugging the cliff. Monte Carlo chooses an inbetween path between them as it is based on the expected returns following a policy so it's neither too far away reducing the reward nor too close reducing the risk of falling into cliff.

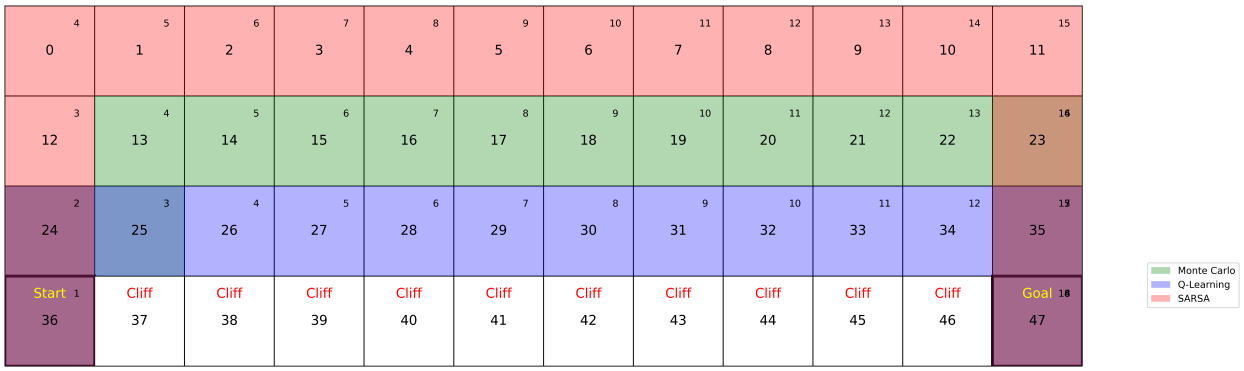

Figure 2: Optimal path chosen by all algorithms

3 BlackJack

Blackjack Rules Overview

Game Flow

A single round proceeds as follows:

1. **Place your bet**
2. **Deal cards**
   - You receive two cards face up
   - Dealer receives one card face up and one face down
3. **Player's turn**
4. **Dealer's turn**
5. **Win/Lose conditions:**
   - Beat the dealer without busting – win a 1:1 payout
   - Get Blackjack (21 in two cards) when dealer doesn't – win a 3:2 payout
   - Bust or dealer beats you – lose your bet
   - Both get the same total – push (no profit/loss)

Dealer Rules

The dealer must follow strict rules:

- Must hit until reaching 17 or higher
- Must stand on any hand of 17 or higher
- No discretion - follows fixed rules regardless of player's hand

Player Options

Option	Description
Hit	Take another card.
	Example: You have 12 (7 + 5). You "hit" and get a 9 → 21! You stand.
Stand	Keep your current hand.
	Example: You have 18 (10 + 8). You "stand" to avoid busting.
Double Down	Double your bet, take exactly one more card, then stand.
	Example: You have 11 (5 + 6). You double your bet, take one card (e.g., 10), and now have 21.
Split	If your first two cards match (e.g., two 8s), split them into two separate hands (each with its own bet).
	Example: You have two 8s (16). You split → now play two hands: Hand 1 (8 + ?) and Hand 2 (8 + ?).
	Aces: If you split Aces, you usually get only one card per Ace.
Surrender	(If allowed) Give up your hand and lose half your bet.
	Example: You have 16 (10 + 6), dealer shows a 10. You surrender to save half your bet.
Insurance	When dealer's upcard is an Ace, bet half your original wager that dealer has Blackjack.
	Payout: If dealer has Blackjack, insurance pays 2:1. Otherwise, you lose the insurance bet.

3.1 Setup

Environment Specifications

- **Action Space:** Discrete(4) with:
  - 0: Stand
  - 1: Hit
  - 2: Double down
  - 3: Split
- **Observation Space:** Tuple containing:
  - Player total (0-31)
  - Dealer's upcard (0-10)
  - Usable ace flag (0/1)
  - Split possibility flag (0/1)
  - Hand active status (0/1)
- **Reward Structure:**
  - Win: +1×bet (standard), +1.5×bet (blackjack)
  - Loss: -1×bet
  - Push: 0

Key Features

- Supports multiple active hands through splitting

- Implements standard casino rules:
  - Dealer stands on all 17s
  - Blackjack pays 3:2
  - Double down allowed on any two cards
- Tracks usable aces and split eligibility
- Includes all essential blackjack actions

Game Dynamics

- Initial deal: 2 cards to player (face up), 1 up/1 down to dealer
- Player acts first with full decision options
- Dealer plays automatically according to fixed rules
- Hand values computed with optimal ace valuation
- Automatic payout calculation when round completes

Using this Blackjack setup, we seek the policy maximizing expected return. We compare:

- **Monte Carlo Control**
- **SARSA**
- **Q-Learning**
- **Double Q-Learning**
- **Expected SARSA**

4 Training Curves

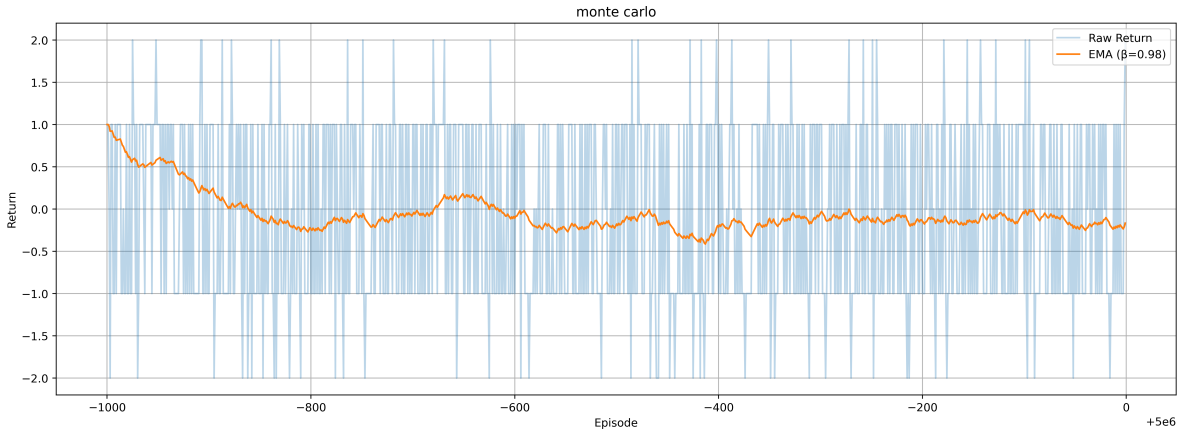

Figure 3: Monte Carlo (MC)

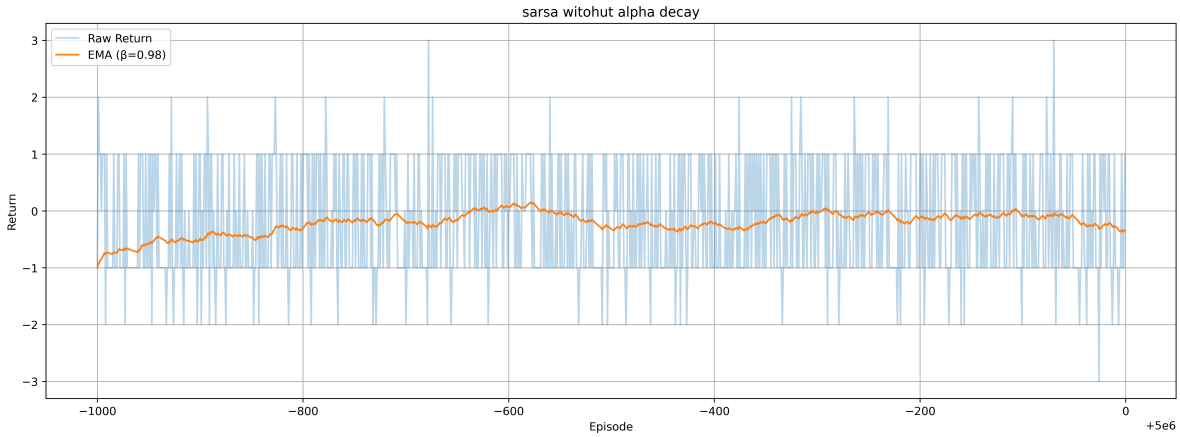

Figure 4: SARSA

Figure 5: Q-learning

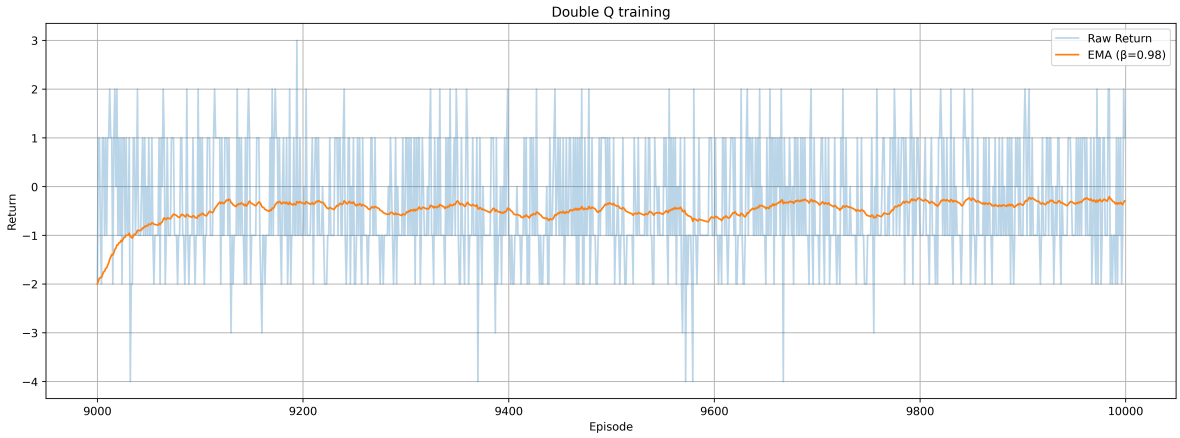

Figure 6: Double Q-learning

Figure 7: Expected SARSA

5 RL Results on Blackjack Environment

Table 3: Best Results by Algorithm

Algorithm	**Average Return**	**Episodes**	**Alpha**	**Discount**	**Epsilon**
Double Q-Learning	−0.0820	100,000	0.1	1.00	0.1
Q-Learning	−0.0890	5,000,000	0.5	0.99	0.05
Expected SARSA	−0.0980	100,000	N/A	N/A	N/A
SARSA	−0.1150	5,000,000	0.15	0.99	0.05
Monte Carlo	−0.1520	5,000,000	0.03	0.99	0.05

Key Observations

- **Double Q-Learning** achieved the best performance (-0.082) with only 100,000 episodes
- **Q-Learning** required 50× more episodes (5M) to achieve similar performance (-0.089)
- Higher α values (0.3-1.0) with decay performed better than fixed low α
- ϵ=0.5 with decay outperformed ϵ=0.1 in Double Q-Learning
- All algorithms converged to approximately 8-10% house edge

Table 4: Hyperparameter Tuning for Double Q-Learning

Return	Alpha	Epsilon	Episodes	Alpha Decay	Epsilon Decay
−0.2655	0.1000	0.1000	1,000	0.0000	0.0000
−0.1480	0.1000	0.1000	10,000	0.0000	0.0000
−0.0920	0.1000	0.1000	100,000	0.0000	0.0000
−0.1100	0.1500	0.1000	100,000	0.0000	0.0000
−0.1460	0.1300	0.1000	100,000	0.0000	0.0000
−0.2180	0.3000	0.1000	100,000	0.0000	0.0000
−0.1100	0.3000	0.1000	100,000	0.0010	0.0000
−0.1075	0.3000	0.1000	100,000	0.0001	0.0000
−0.1370	0.4000	0.1000	100,000		0.0000
−0.1760	0.4000	0.1000	100,000	0.0000	0.0000
−0.1100	0.2000	0.1000	100,000	0.0001	0.0000
−0.1100	0.2500	0.1000	100,000		0.0000
−0.1150	0.3500	0.1000	100,000	0.0001	0.0000
−0.1050	1.0000	0.1000	100,000	0.0001	0.0000
−0.1050	1.0000	0.1000	100,000	0.0010	0.0000
−0.0920	0.1000	0.5000	100,000	0.0000	0.0010
−0.1270	0.1000	1.0000	100,000	0.0000	0.0001
−0.1270	0.2000	0.5000	1,000,000	0.0001	0.0010
−0.0950	0.1000	0.5000	1,000,000	0.0000	0.0010
−0.0950	0.1000	0.1000	1,000,000	0.0000	0.0000

Conclusion

- **Double Q-Learning** emerged as the most efficient algorithm, achieving the best return (-0.082) with minimal computational requirements
- The optimal hyperparameters for Double Q-Learning were: α=0.1, ϵ=0.1, discount=1.0 with no decay
- SARSA showed sensitivity to α values, performing better with moderate α=0.15
- Monte Carlo methods underperformed temporal difference methods in this environment
- Despite extensive hyperparameter tuning, we were unable to reach the theoretical optimum (-0.005), likely due to environment stochasticity and random seed constraints