Playing BlackJack with Reinforcement Learning Algorithms

Group G

CHAN Chak Yan

CHOY Ka Hei

LEE Chun Wing

YUEN Ho Man

Content

1. Background

1. Theory / Algorithm

1. Result

1. Discussion

1. Conclusion

Background

Background

1. BlackJack is one of the most popular card games around the world.

1. Game objective : obtaining the total value higher than dealer’s without exceeding 21.

1. Challenge : Finding the optimal blackjack strategy with the stochastic game property

1. It is well-known for developing RL algorithms and their performance evaluation.

Blackjack’s Rule and Setting

Card type : Poker cards without Jokers

Number of player : 1 Number of dealer : 1

Actions’ option in each round :

1. Request an additional card (called a “hit”)
2. Request a stop (called a “stick”)

Setting

1. All cards in player’s hand should be face-up
2. Dealer round starts after player’s round is finished
3. Dealer plays with fixed strategy

Setting

Wining condition :

-> total card values of player's hand > that of dealers' hand ; otherwise, player loses

-> dealer bust when player does not bust

 # Anyone bust (exceeds 21) -> lose

Card Value :

1. Face cards including Jack, Queen, King = 10
2. Numerical cards (2-9) = their face value
3. Ace (A) = "1 or 11" , called "usable ace"

Game Procedure

Starting :

Dealer - one face-up card and one face-down card.

Player - two face-up cards.

Player's Round

1. "Hit"- ask for more cards until they bust (total exceeding 21)
2. "Stick" - stand / stop, then move to dealer's round

Dealer's Round

1. draw more cards until the total value is greater than or equal to 17

Motivaton

1. Optimal Blackjack strategy to maximize the wining rate / its gain is a big challenge.
2. Supervised learning algorithm is one of possible approach to solve this problem but doesnt not take advantage of reward structure of the game in the long-run.
3. Instead, RL -> perform well in stochastic environments with its goal.
4. Blackjack - > Markov decision process (MDP) * which can be easily formulated
5. State and action are simple

Main Idea

Goal of our project : obtain the possible best policy on playing Blackjack

-> maximize the rewards of the players in the game.

We choose the following algorithms to achieve this goals :

1. Monte Carlo
2. Temporal Difference Learning
3. SARSA
4. Q-learning

2. Theory / Algorithm

2.1 Exploration-Exploitation Dilemma and ε-greedy policy

Exploration versus Exploitation's Trade-off

Exploration: a long-term benefit concept such that the agent aims to improve its knowledge about the environment and each action at any given time step.-> the agent may execute a less optimal policy or actions, which cause the loss of immediate rewards

Exploitation: a greedy approach in which agents aim to maximize the total rewards based on the current estimated value instead of the true value. -> agent makes the best decision given information while the total rewards at the end may not be optimal

Problem : "Should the algorithm spends more time exploring unknown states, actions for each state, associate reward and the transition probability, or should it choose the best action on updating the Q-value with observable information from its history?"

ε-greedy policy

The idea is to pick a greedy action with probability 1-ε , otherwise, choose a random action uniformly with probability ε.

$$\pi(a \mid s) = \begin{cases} \epsilon/A + 1 - \epsilon & \text{if } a = \arg\max_{a \in \mathcal{A}} Q(s,a) \\ \epsilon/A & \text{otherwise} \end{cases}$$

2.2 Monte Carlo Methods

Monte Carlo Reinforcement learning

Average sample return

Learns directly from episodes of experience

We don’t know the MDP transitions

Learns from complete episodes

Can only apply MC to episodic MDPs

Monte Carlo First Visit Control method

Initialization: $N(s,a) = 0$, $G(s,a) = 0$, $Q\pi(s,a) = 0$, $\forall s \in S$, $\forall a \in A$

For loop (looping over episodes i):

Get episode observations

Define return G in step t.

For every state-action pair visited in episodes i, and for the first time t that (s,a) is visited in episodes i.

$N(s,a) = N(s,a)+1$

$G(s,a) = G(s,a)+1$

$Q\pi(s,a) = G(s,a)/N(s,a)$

GLIE Monte Carlo First Visit Control method

Initialization: $Q(s,a) = 0$, $N(s,a) = 0$, $\forall s \in S$, $\forall a \in A$

For loop (looping over episodes i):

Set epsilon <- 1/k, πk = epsilon-greedy(Q)

Get episode observations

Define return G in step t.

For every state-action pair visited in episodes i, and for the first time t that (s,a) is visited in episodes i.

$N(s,a) = N(s,a)+1$

$Q(s,a) = Q(s,a)+(1/N(s,a))^*(G-Q(s,a))$

ε-greedy policy for GLIE Monte Carlo control

For epsilon <- 1/k in each time step k, we discover that it may not be that efficient to make the learning algorithms converge through training.

Therefore, instead of initializing epsilon, we apply a decay rate for the epsilon to decay.

We can control the decay speed of epsilon, to let the agent explore more or stay more safe.

```
For loop (over episodes):
    epsilon_start=1.0
    epsilon_decay=0.99999
    epsilon_min=0
    epsilon = eps_start*(eps_decay^(episodes-1))
```

2.3 Temporal-Difference Learning

Temporal-Difference Learning

Idea : Predicting a variable's future value in a sequence of states, replace complicated reasoning with simple learning procedures and generate the same results

In our case :

Predict the chance of winning if we hit and tell whether we should hit or stick in order to have a larger sum of value while not being busted

Temporal-Difference Learning

The formula is given as:

constant step-size parameter TD Error

$$V(S_t) \leftarrow V(S_t) + \alpha\left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right]$$

TD Error : Difference between the discounted value estimate of Vt+1, current estimate for Vt and the actual reward gained from transitioning between st and st+1

Constant step-size parameter : affects the speed of how the Temporal Difference algorithm learns

Temporal-Difference Learning

How it works ?

✕ Calculate the total future reward

✓ Predicts the combination of immediate reward and its own reward prediction at the next moment in time

If new prediction produced at next state different from expected prediction :

- Calculate the difference between the predictions
- Use this temporal difference to produce a new prediction by adjusting the old one

Temporal-Difference Learning

Advantage :

- Low variance, depends on one random action, transition and reward
- Able to learn online
- Able to learn from incomplete sequence → tackle continuous problems

2.3.1 State–Action–Reward–State–Action (SARSA)

SARSA-Background

- Belongs to a on-policy temporal difference method.
- The formula of SARSA is given as:

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

- The on-policy algorithm allows the SARSA model to update its policy depending on the actions taken.

where,
alpha is the step size,
gamma is the discount factor.

SARSA-Algorithm

The algorithm of SARSA is described as:

1. Initialize the Q value (Q(s,a))
2. Give a observation to the state (S0=s0)
3. Choose an action (At) based on ε-greedy policy π0
4. Take the action A0~π0(S0), and observe the reward, R1, also the new state,S1.
5. Repeat the following stop for each episode until terminate(t=0,1,2…):
    1. Take action At+1 ~πt(St+1) and observe (Rt+2,St+2)
    2. Update the Q-value for the state with the observed reward and expected reward for the next state.
    3. Update the policy πt+1 with εt+1-greedy(Q)

SARSA-Application

- Convergence of SARSA
  - Aiming for converging to the optimal action-value function, Q(s,a) →q*(s,a).
  - We set the policy is:
    - Satisfies the condition of GLIE
    - and $\sum_{t=1}^{\infty} a_t = \infty, \sum_{t=1}^{\infty} a_t^2 < \infty$
- Tradeoff of explore and exploit
  - Aiming for exploring more at the first and eventually stay on the best action.
    - Set 1-ε as the probability of greedy action
    - ε be the probability of random action.

2.3.2 Q-learning (SARSAmax)

Q-learning-Background

- A popular algorithm in reinforcement learning
- Belongs to an off-policy temporal difference method.
- Find the best action in the current state.
- Will learn from action which is not from the current policy.

- The formula of Q-learning is given as:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha_t \cdot \left(R_{t+1} + \gamma \max_{a \subset \mathcal{A}} Q(S_{t+1}, a) - Q(S_t, A_t) \right)$$

where,
alpha is the step size,
gamma is the discount factor.

Q-learning-Algorithm

The algorithm of Q-learning is described as:

1. Initialize the Q value (Q(s,a))
2. Give a observation to the state (S0=s0)
3. Initialize ε-greedy policy π'0
4. Repeat the following stop for each episode until terminate(t=0,1,2…):
    1. Take action At ~π't(St) and observe (Rt+1,St+1)
    2. Update the Q-value for the state using the observed reward and the maximum reward for the next state.
    3. Update π't+1 with ε-greedy(Q)

3.Result

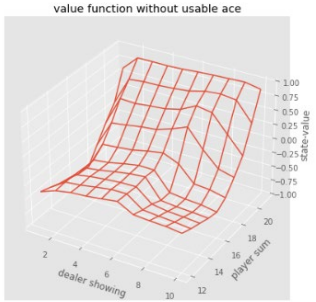
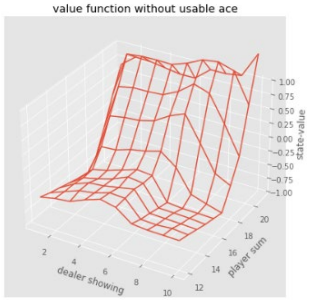
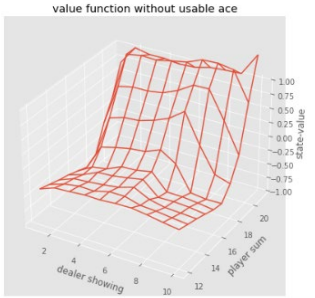
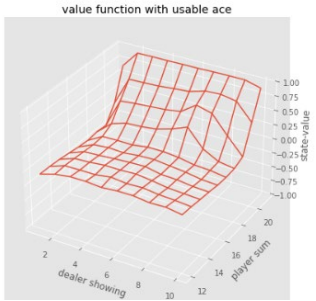
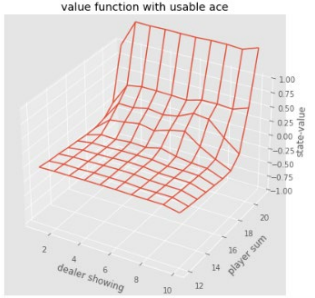
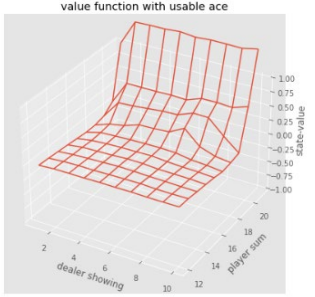
Result

- GLIE Monte Carlo control method
  - Set the number of episodes= 1000000, gamma = 0.4 and evaluating episodes = 10000.

- For GLIE SARSA and GLIE Q-learning
  - we set the number of episodes = 1000000, alpha = 0.001, gamma = 0.4 and evaluating episodes= 10000.
- The gamma = 0.4 was selected for the sake of the short horizon needed (1-3 turns) for Blackjack.

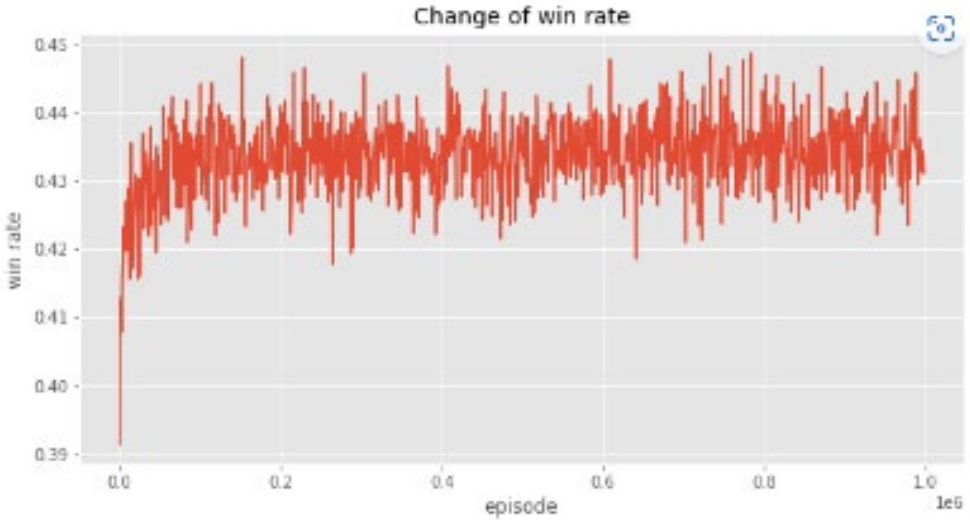
State Value Function Plot

State Value Function Plot

	GLIE Monte Carlo control	GLIE SARSA	GLIE Q-Learning (SARSAMAX)
Without usable Ace			
With Usable Ace			

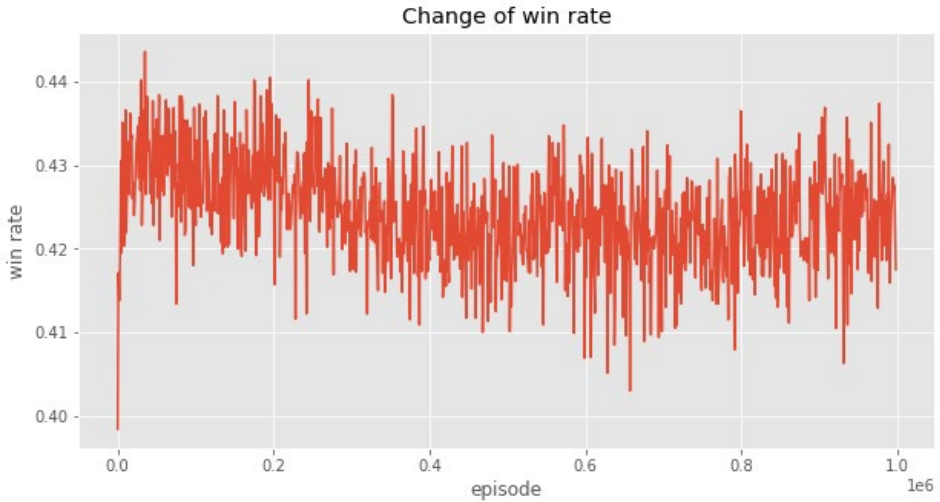
Win rate curve

Win Rate Curve (GLIE Monte Carlo control）



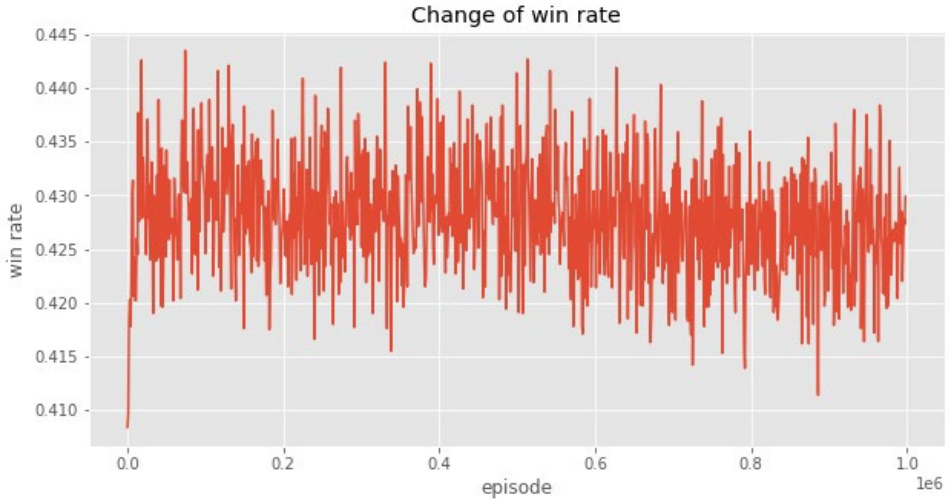
It has increased trend and becomes stable at the value between 0.43 to 0.44 after 100000 episodes.

Win Rate Curve (GLIE SARSA)



It is first increased to around 0.43. However, after around 200000 episodes, the win rate starts to decrease and become steady at the value between 0.42 to 0.44

Win Rate Curve
GLIE Q-Learning (SARSAMAX)

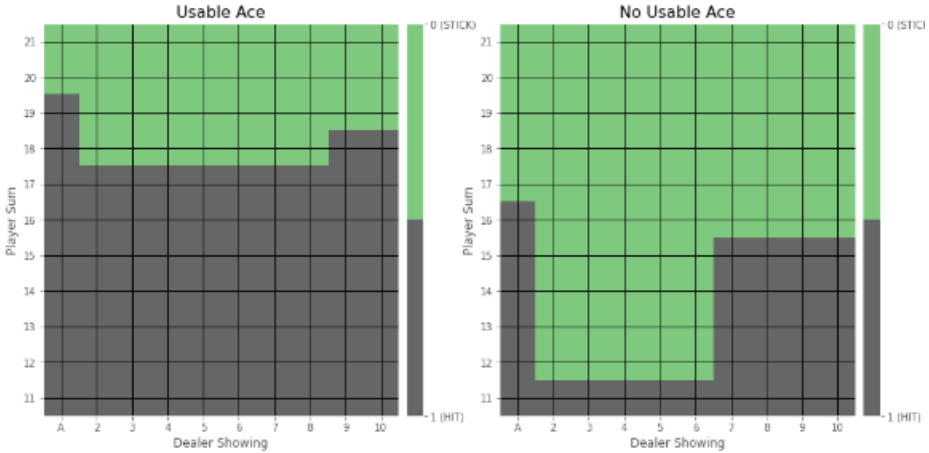


The value varies a lot between 0.415 to 0.44 and therefore no trend is observed from the graph

Optimal Policy Table

Optimal Policy Table (GLIE Monte Carlo control)

With Usable Ace **Without usable Ace**



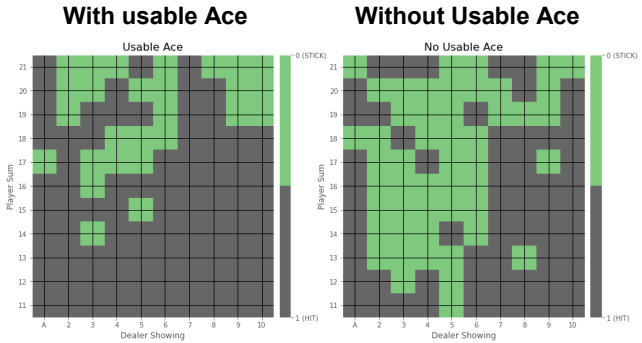
Black - Hit action

Green - Stick action

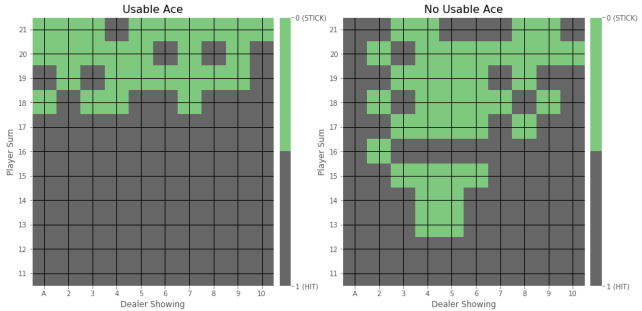
- well-rounded
- clear strategy
- perfect way in sight

Optimal Policy Table (GLIE SARSA , GLIE Q-learning)

GLIE SARSA



GLIE Q-learning



- relatively scattered
- not yet ready to be an output

4. Discussion

Perfomance of Method

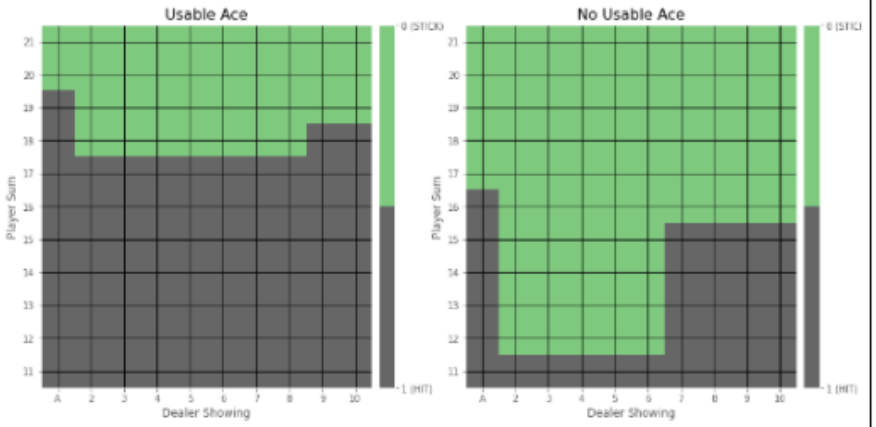
Perfomance of Method

If we look through the win rate curve and the optimal policy table, it should be easy to draw a conclusion that GLIE Monte Carlo Control is the best method in this project.

Blackjack strategy table comparative analysis with strategy table by Monte Carlo control

Hard Totals

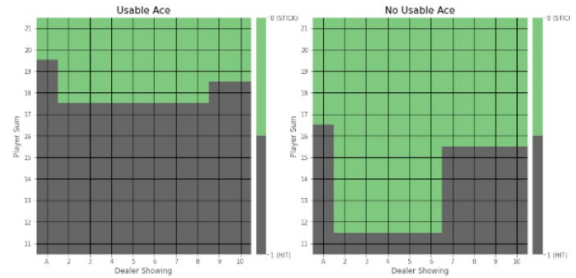
Your Hand	Dealer 2	3	4	5	6	7	8	9	T	A
Hard 5	H	H	H	H	H	H	H	H	H	H
Hard 6	H	H	H	H	H	H	H	H	H	H
Hard 7	H	H	H	H	H	H	H	H	H	H
Hard 8	H	H	H	H	H	H	H	H	H	H
Hard 9	H	D	D	D	D	H	H	H	H	H
Hard 10	D	D	D	D	D	D	D	D	H	H
Hard 11	D	D	D	D	D	D	D	D	D	D
Hard 12	H	H	S	S	S	H	H	H	H	H
Hard 13	S	S	S	S	S	H	H	H	H	H
Hard 14	S	S	S	S	S	H	H	H	H	H
Hard 15	S	S	S	S	S	H	H	H	H	H
Hard 16	S	S	S	S	S	H	H	H	H	H
Hard 17	S	S	S	S	S	S	S	S	S	S
Hard 18+	S	S	S	S	S	S	S	S	S	S



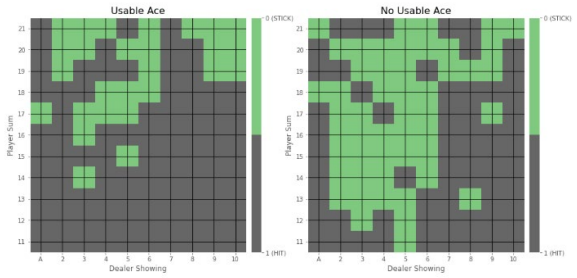
Left table: If dealer showing 7, 8, 9, T (having a value of 10) -> should hit, risk for high

For MC control: Similar performance!

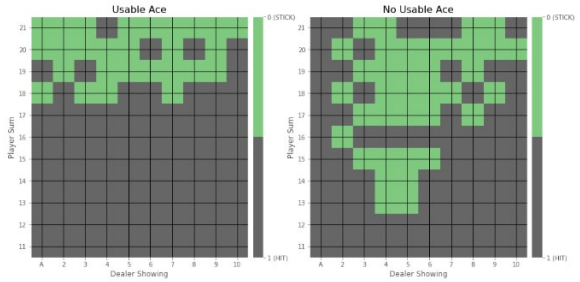
Potentials of the other model



MONTE CARLO CONTROL



SARSA



SARSAMAX

Although the performance of the SARSA and SARSAMAX algorithm may not be satisfactory, we can still see some correlation between the optimal policy table between three algorithms.

Even the optimal policy table for SARSA and SARSAMAX is very scattered, we can see that sticking patterns of SARSA and SARSAMAX are very similar to Monte Carlo Control's sticking patterns.

Advantages and disadvantages of the methods

Monte Carlo methods VS Temporal-Difference learning

	Monte Carlo methods	**Temporal-Difference learning**
Varaince	High	Low
Bias	Zero	Some
Initial Value	Not sensitive	Sentitive
Learning	Wait until the end of the episode	Learn online after every step
How to Learn	Learn from complete sequences	Learn without the final outcome, from incomplete sequences
Environment	Episodic (terminating)	Continuing (non-terminating)

Limitation and difficulties

Limitation and difficulties

- Several limitations regarding the computation, the optimization and strategy on deciding the policy.
  - Difficult to take a complete list of actions :
    - “Hit” , “Stand”, “Double down”, “Split”, “Surrender”

- “Edward Thorp’s Strategy” will be a better strategy instead of random strategy
  - allows players to gain the best possible return in long-term success

Conclusion

Conclusion

Goal : Obtaining the possible best Hit-Stand policies on the game of Blackjack.

Impremented Algorithm : GLIE Monte Carlo , GLIE Sarsa, and Q-learning

For each of these algorithms,
State value functions: description of how the algorithms converged
Winning rate varies from 0.43 to 0.44, 0.42 to 0.44 and 0.415 to 0.44, respectively.
Optimal policy table: GLIE Monte Carlo is very clear to see the rules, while GLIE Sarsa and Q-learning (SARSAMAX) are scattered.
We observe that GLIE Monte Carlo control is the best method applied in our project.

Future possible works

1. Explore the effect of different strategies that already hold together
2. Try some extensions of RL algorithms such as Deep Q-network and Bayesian Q-learning algorithm .

Thank you !