Monte Carlo Methods

Model-free

- This lecture introduces model-free methods as an alternative to dynamic programming to compute value and Q functions without the model dynamics of the environment.
- The Monte Carlo (MC) method is a popular model-free method used in reinforcement learning for prediction and control tasks.
- This lecture provides an example of training an agent to play blackjack using the MC prediction and control methods.

An Overview of the Monte Carlo Method

1. The Monte Carlo method is a statistical technique used to find an approximate solution through sampling.
2. It is commonly used in design and engineering to simulate a wide range of scenarios and help identify the most optimal solution.
3. The Monte Carlo method involves generating random inputs within specified ranges and running simulations to calculate the probable output based on those inputs.
4. It is particularly useful when dealing with complex systems or problems with many variables.
5. The method is named after the famous casino in Monte Carlo, as it relies on probability and chance in a similar way to gambling.
6. By using the Monte Carlo method, designers can make informed decisions based on data-driven insights and reduce the risk of costly errors in their designs.

The Monte Carlo method approximates the expectation of a random variable by sampling, with better approximations resulting from larger sample sizes. To compute the expected value of a random variable X, the sum of the values of X multiplied by their respective probabilities is taken.

Prediction and control tasks

The prediction task

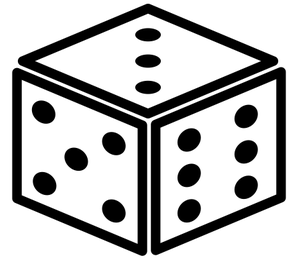
- The prediction task is a popular machine learning application that uses a model to forecast new data based on patterns learned from past data.
- Its accuracy depends on the data quality and model efficiency.
- Techniques such as cross-validation and hyperparameter tuning can enhance the model's performance. T
- he prediction task is valuable for informed decision-making based on data in various fields like finance, healthcare, and marketing.

The control task

- Control tasks involve monitoring and regulating systems to maintain operation within specified parameters using specialized hardware and software
- systems such as PLCs and DCSs. Effective control task implementation is vital for ensuring safety, reliability, and productivity in industrial processes and systems.

Monte Carlo prediction

Monte Carlo prediction



Monte Carlo prediction is a computational algorithm that uses random sampling to estimate the probability of certain outcomes. It is widely used in various fields, including finance, engineering, and physics.

The method involves generating a large number of random samples from a probability distribution and using them to calculate the expected value of a quantity of interest.

The accuracy of the prediction increases with the number of samples taken, and it can be used to simulate complex systems and analyze their behavior.

Monte Carlo prediction has proved to be a powerful tool for decision-making in many applications, and its versatility and flexibility make it a valuable technique in modern data analysis.

In contrast to other methods, the Monte Carlo method is model-free. This means that it doesn't require any specific model dynamics to calculate the value function.

$$V(s) = \frac{1}{N(s)} \sum_{i=1}^{N(s)} G_i$$

Monte Carlo method for estimating the value function

where:

- V (s) is the estimated value of state s,
- N(s) is the number of times state s has been visited.
- Gi is the return (cumulative reward) from the i-th visit to state s.
- The sum is taken over all N(s) visits to state s.

This equation represents the average of all returns observed from visits to state s, providing an estimate of its value based on empirical data.

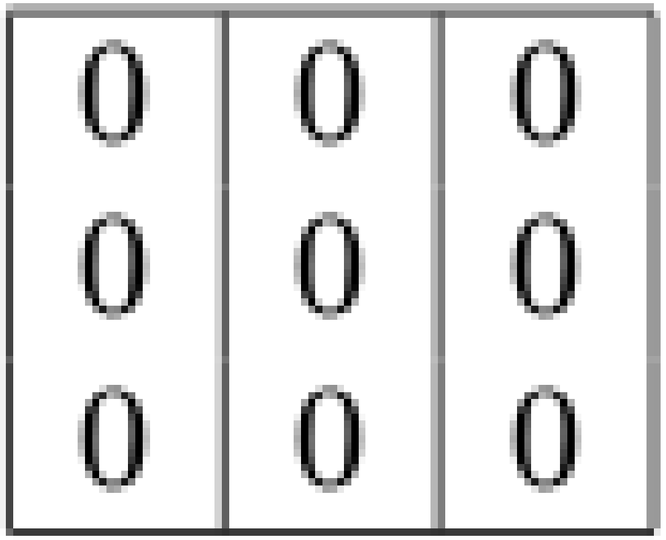
Example: Monte Carlo Method for Grid World

Problem Setup

- 3x3 grid world
- Agent can move left, right, up, or down
- Reward of -1 for each step
- Reward of +10 for reaching the goal state
- Discount factor $\gamma=0.9$

Step 1: Initialisation

- Start with an empty grid world
- Initialise the value function for each state with zero



Step 2: Episode 1

Episode 1: The agent starts at an initial state (e.g., S1), takes a sequence of actions, and reaches the goal state (G). The sequence of states visited and the rewards received during this episode are recorded:

01 ---------- **States visited: S1 → S2 → G**

02 ---------- **Rewards received: – 1 → –1 → +10**

Step 3: Update Value Function

Using the first-visit Monte Carlo method, update the value function for each state visited in the episode based on the observed returns. Since this is the first visit to each state in this episode, we can simply average the returns for each state:

$$V(S1) = \frac{-1}{1} = -1$$

$$V(S2) = \frac{-1 + 10}{1} = 9$$

$$V(G) = \frac{10}{1} = 10$$

Step 4: Episode 2

Episode 2: Repeat the process for another episode, starting from a dif- ferent initial state if desired. Record the states visited and the rewards received during this episode.

Step 5: Update Value Function

Update the Value Function: Update the value function based on the returns observed in the second episode.

Step 6: Repeat

Repeat: Continue this process for a predefined number of episodes or until convergence.

In this example, we illustrate how the Monte Carlo method can be utilized to approximate the value function for a basic grid world setup. As more episodes are completed, the estimated state values progress, leading to a more accurate estimation of the genuine values.

MC prediction algorithm

Monte Carlo Prediction Algorithm

Input:

Policy π

Number of episodes N

Initialisation:

- Initialise empty arrays *Returns*(*s*) and *Visits*(*s*) for each state *s*
- Initialise value function *V*(*s*) for each state *s* with arbitrary values

Monte Carlo Prediction Algorithm

Algorithm 1 Monte Carlo Prediction

Initialize: $\forall s \in \mathcal{S}, \hat{V}(s) \leftarrow 0,\ N(s) \leftarrow 0$
for $k = 1, 2, \ldots, K$ **do**
  Generate an episode following policy π: $S_0, A_0, R_1, \ldots, S_T$
  $G \leftarrow 0$
  for $t = T-1, T-2, \ldots, 0$ **do**
    $G \leftarrow \gamma G + R_{t+1}$
    if S_t not in $S_0, S_1, \ldots, S_{t-1}$ **then**
      $N(S_t) \leftarrow N(S_t) + 1$
      $\hat{V}(S_t) \leftarrow \hat{V}(S_t) + \frac{1}{N(S_t)}(G - \hat{V}(S_t))$
    end if
  end for
end for

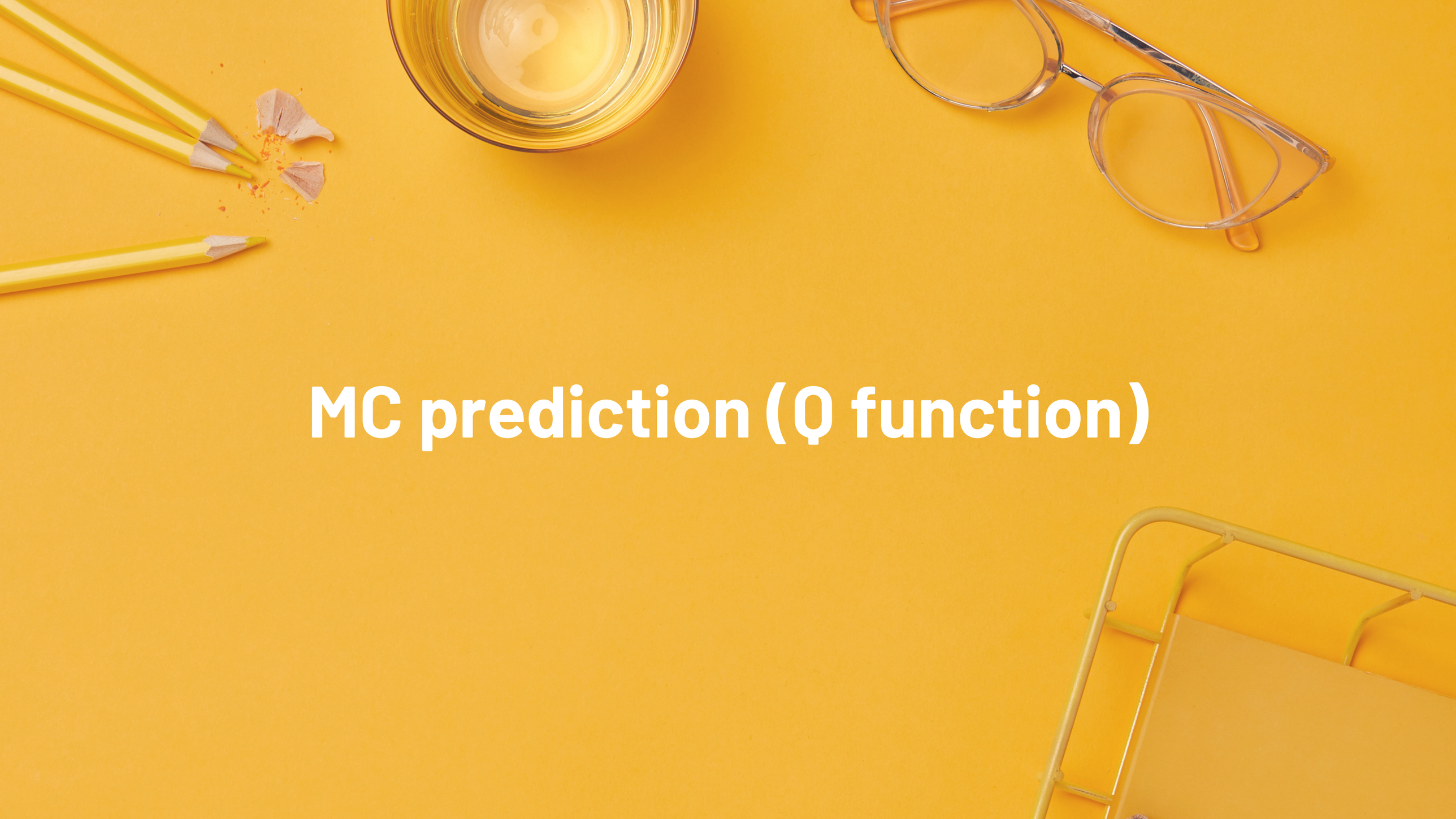
Types of MC prediction

First-visit Monte Carlo

The First-Visit Monte Carlo Method avoids recalculating the return for a state if it has already been visited within the same episode.

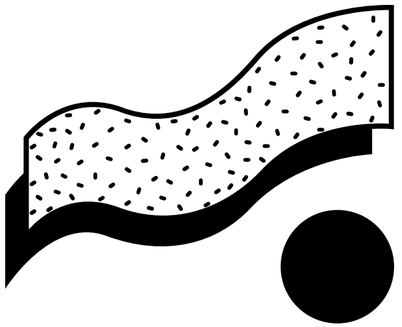
Every-visit Monte Carlo

Calculating the Return Every Time a State is Visited During the Episode.



MC prediction (Q function)

MC prediction (Q function)



MC prediction is a method used in reinforcement learning to estimate the value function of a policy. This is done by running simulations of the environment and the agent's actions, and calculating the total rewards obtained.

The Q function, which maps state-action pairs to expected rewards, can then be estimated by averaging the observed rewards for each state-action pair across many simulations.

MC prediction is a powerful technique because it does not require any knowledge of the environment's dynamics or transition probabilities, making it suitable for a wide range of problems.

Predicting the Q function using the MC method involves the same process as predicting the value function discussed previously. The only difference is that the return of the state-action pair is used instead of the return of the state to make the prediction.

Monte Carlo control

Monte Carlo control

In the context of reinforcement learning, the argmax operation is often used to select the action that maximizes the estimated value function Q(s,a) for a given state s. For example, in the Q-learning algorithm, the agent selects the action with the highest estimated Q-value for a given state:

$$\pi = \arg\max_{a} Q(s, a)$$

In the control task, finding the optimal policy through trial and error is difficult.

The Monte Carlo method can be used to iteratively search for the optimal policy that provides the highest return.

To compute a policy, the Q function is used to choose the best action in each state based on the maximum Q value.

The Q function is computed by generating several episodes using the policy and computing the average return of the state-action pair across those episodes, similar to the MC prediction method.

Monte Carlo Control Algorithm

Monte Carlo Control Algorithm

Input:

- Policy π
- Number of episodes N

Initialization:

- Initialize empty arrays $Returns(s, a)$ and $Visits(s, a)$ for each state s and action a
- Initialize action-value function $Q(s, a)$ for each state s and action a with arbitrary values

Algorithm:

1. **For** each episode $i = 1, 2, \ldots, N$ **do**:
   - Generate an episode following policy π: $(s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_T)$
   - $G \leftarrow 0$
   - **For** each step $t = T - 1, T - 2, \ldots, 0$ **do**:
     - $G \leftarrow \gamma G + r_{t+1}$ (where γ is the discount factor)
     - **If** (s_t, a_t) is not in the episode history from time step 0 to $t - 1$ **then**:
       * Append G to $Returns(s_t, a_t)$
       * Increment $Visits(s_t, a_t)$ by 1
       * Update action-value function $Q(s_t, a_t)$ based on $Returns(s_t, a_t)$ and $Visits(s_t, a_t)$:

$$Q(s_t, a_t) = \frac{1}{Visits(s_t, a_t)} \sum_{i=1}^{Visits(s_t, a_t)} Returns(s_t, a_t)[i]$$

Output: Estimated action-value function $Q(s, a)$ and optimal policy π^*

On-policy Monte Carlo control

On-policy Monte Carlo control is a model-free reinforcement learning method that estimates the value function of the policy by using Monte Carlo simulations. It is useful for real-world problems where the underlying model is unknown or difficult to estimate and involves following the current policy, collecting experience, and then using that experience to update the policy.

There are two types of on-policy Monte Carlo control methods:

- Monte Carlo exploring starts
- Monte Carlo with the epsilon-greedy policy

Monte Carlo Exploring Starts

Monte Carlo exploring starts

The Monte Carlo control method requires exploration to determine whether an action in a state is optimal or not. Lack of exploration can lead to the agent not knowing whether an action is good or not. Monte Carlo exploring helps to ensure sufficient exploration and solve this problem.

Monte Carlo Control with Exploring Starts

Input:

- Environment with states S, actions A, transition probabilities P, rewards R, and discount factor γ.
- Number of episodes N.

Initialization:

- Initialize action-value function $Q(s, a)$ arbitrarily for all s and a.
- Initialize state-action visit count $N(s, a) = 0$ for all s and a.
- Initialize policy π with random actions for each state-action pair.

Algorithm:

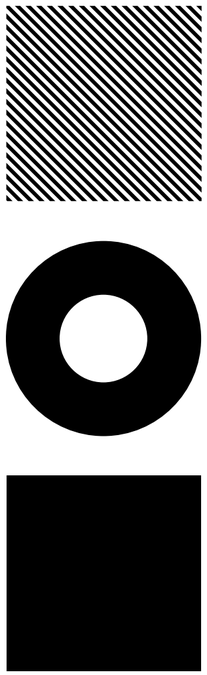
1. **For** each episode $i = 1, 2, \ldots, N$ **do**:
   - Choose a state s_0 and action a_0 arbitrarily, using exploring starts.
   - Generate an episode following policy π: $(s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_T)$.
   - $G \leftarrow 0$
   - **For** each step $t = T - 1, T - 2, \ldots, 0$ **do**:
     - $G \leftarrow \gamma G + r_{t+1}$
     - **If** (s_t, a_t) is not in the episode history from time step 0 to $t - 1$ **then**:
       - Increment $N(s_t, a_t)$ by 1
       - Update action-value function $Q(s_t, a_t)$ based on G and $N(s_t, a_t)$:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \frac{1}{N(s_t, a_t)}(G - Q(s_t, a_t))$$

       - Update policy π to be greedy with respect to Q:

$$\pi(s_t) \leftarrow \text{argmax}_a Q(s_t, a)$$

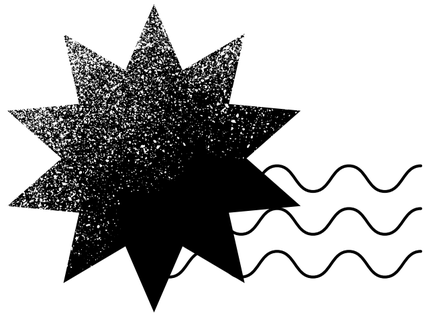
Output: Optimized policy π based on the estimated action-value function Q.



Monte Carlo with the epsilon-greedy policy

The Monte Carlo method is a way to estimate the value of a function by randomly sampling its inputs. When combined with the epsilon-greedy policy, it can be used to estimate the value of different actions in a reinforcement learning setting. The epsilon-greedy policy is a way of balancing exploration and exploitation in a learning agent. It works by choosing a random action with probability epsilon (exploration) and choosing the action with the highest estimated value with probability 1-epsilon (exploitation). By using Monte Carlo with the epsilon-greedy policy, we can estimate the value of each action by averaging the rewards received when that action was taken, and use these estimates to guide the agent's future actions.

EPSILON-GREEDY POLICY

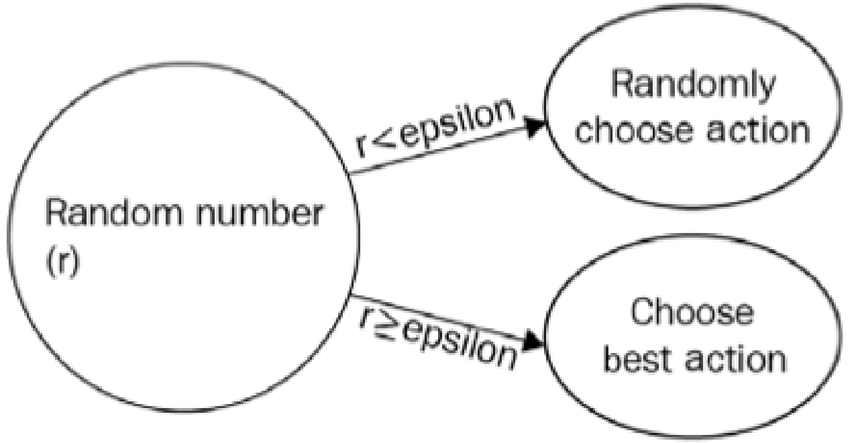


- A greedy policy selects the best available action in a given state. The example of four possible actions, up, down, left, and right, in state A is given, with the Q values for actions up and right shown in a table.
- The greedy policy selects the action with the highest Q value in a given state based on the Q table. The example provided shows that the action "up" has the highest Q value in state A, so it would be selected by the greedy policy.
- But one problem with the greedy policy is that it never explores the other possible actions; instead, it always picks the best action available at the moment.
- The exploration-exploitation dilemma arises when deciding whether to explore all possible actions and choose the best one or exploit the best action from already-explored options. This is relevant in determining the maximum Q value.

State	Action	Value
A	up	3
A	right	1

Epsilon-greedy policy

The epsilon-greedy policy is a new approach to decision-making that balances exploration and exploitation. It involves trying all actions with a non-zero probability and randomly exploring different actions with a certain probability epsilon, while choosing the action with the maximum Q value with a probability of 1-epsilon.



In the epsilon-greedy policy, if we set the value of epsilon to 0, then it becomes a greedy policy (only exploitation), and when we set the value of epsilon to 1, then we will always end up doing only the exploration. So, the value of epsilon has to be chosen optimally between 0 and 1.

The MC control algorithm with the epsilon-greedy policy

Input:

- Environment with states S and actions A
- Number of episodes N
- Discount factor γ
- Exploration parameter ϵ

Initialization:

- Initialize action-value function $Q(s, a)$ arbitrarily for all s and a
- Initialize $N(s, a) = 0$ for all s and a

Algorithm:

1. **For** each episode $i = 1, 2, \ldots, N$ **do**:
   - Generate an episode using policy derived from Q (e.g., epsilon-greedy)
   - $G \leftarrow 0$
   - **For** each step $t = T - 1, T - 2, \ldots, 0$ **do**:
     - $G \leftarrow \gamma G + R_{t+1}$ *// Incrementally calculate return*
     - **If** S_t, A_t not in episode history from time step 0 to $t - 1$ **then**:
       * $N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$
       * $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G - Q(S_t, A_t))$ *// Update action-value function*

Output: Optimal policy π derived from Q

Off-Policy Monte Carlo Control Method

The off-policy Monte Carlo method is a fascinating approach to Monte Carlo control. It utilizes two policies, the behavior policy and the target policy. The behavior policy is used to generate episodes, while the target policy is improved upon iteratively.

In contrast, the on-policy method generates an episode using the policy π and iteratively improves the same policy π to determine the optimal policy.

With off-policy Monte Carlo, however, we generate episodes using the behavior policy b and improve the target policy π.

Is the MC method applicable to all tasks?

Monte Carlo and Temporal Difference Learning are model-free methods used in Reinforcement Learning to calculate the value and Q function of a state or state-action pair. Monte Carlo averages the return of the state and state-action pair, respectively, but is limited to episodic tasks.
Temporal Difference Learning is used for continuous tasks, estimating the state value without knowing the model dynamics of the environment.

Summary

- The Monte Carlo method approximates the expectation of a random variable through sampling.
- The prediction task evaluates a given policy by predicting the value function or Q function.
- The control task finds the optimal policy. T
- he Monte Carlo method can be used for both prediction and control tasks.
- The value of a state and state-action pair can be computed by taking the average return across several episodes.
- First-visit MC and every-visit MC methods are used to compute returns for a state.
- On-policy and off-policy control methods are used to perform a control task.
- The on-policy method involves iterative improvement of one policy to find the optimal policy through Monte Carlo control exploring starts or with an epsilon-greedy policy.
- The off-policy Monte Carlo control method uses two policies: behavior policy to generate the episode and the target policy to find the optimal policy.


"The future belongs to those who believe in the beauty of their dreams."
— Eleanor Roosevelt