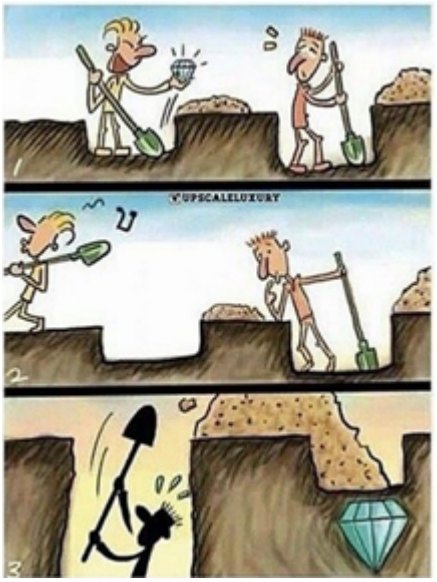
Exploitation and Exploration in Machine Learning

Exploitation and exploration are the key concepts in Reinforcement Learning, which help the agent to build online decision making in a better way.reinforcement learning is a feedback-based approach in which agent learns by performing some actions as well as their outcomes. Based on action status (good or bad), the agent gets positive or negative feedback. Further, for each positive feedback, they get rewarded, whereas, for each negative feedback, they also get penalized.

- Reinforcement learning does not require any labeled data for the learning process. It learns through the feedback of action performed by the agent. Moreover, in reinforcement learning, agents also learn from past experiences.
- Reinforcement learning methods are used to solve tasks where decision-making is sequential and the goal is long-term, e.g., robotics, online chess, etc.
- Reinforcement learning aims to get maximum positive feedback so that they can improve their performance.
- Reinforcement learning involves various actions, which include taking action, changing/unchanged state, and getting feedback. And based on these actions, agents learn and explore the environment.



In reinforcement learning, whenever agents get a situation in which they have to make a difficult choice between whether to continue the same work or explore something new at a specific time, then, this situation results in Exploration-Exploitation Dilemma because the knowledge of an agent about the state, actions, rewards and resulting

states is always partial.Exploitation is defined as a greedy approach in which agents try to get more rewards by using estimated value but not the actual value. So, in this technique, agents make the best decision based on current information.exploration techniques, agents primarily focus on improving their knowledge about each action instead of getting more rewards so that they can get long-term benefits. So, in this technique, agents work on gathering more information to make the best overall decision.

Epsilon Greedy Policy

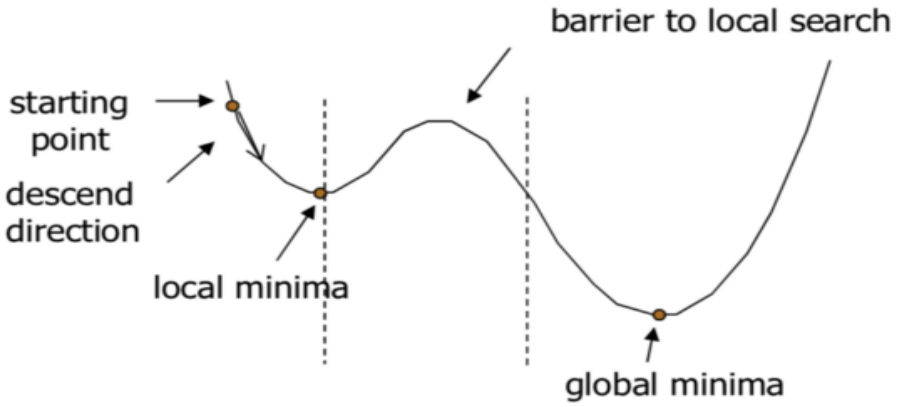
In the greedy epsilon strategy, an exploration rate or epsilon (denoted as ε) is initially set to 1. This exploration rate defines the probability of exploring the environment by the agent rather than exploiting it. It also ensures that the agent will start by exploring the environment with ε=1.

As the agent start and learns more about the environment, the epsilon decreases by some rate in the defined rate, so the likelihood of exploration becomes less and less probable as the agent learns more and more about the environment. In such a case, the agent becomes greedy for exploiting the environment.

To find if the agent will select exploration or exploitation at each step, we generate a random number between 0 and 1 and compare it to the epsilon. If this random number is greater than ε, then the next action would be decided by the exploitation method. Else it must be exploration. In the case of exploitation, the agent will take action with the highest Q-value for the current state.

Simulated Annealing

Simulated Annealing (SA) mimics the Physical Annealing process but is used for optimizing parameters in a model. This process is very useful for situations where there are a lot of local minima such that algorithms like Gradient Descent would be stuck at.



if Gradient Descent started at the starting point indicated, it would be stuck at the local minima and not be able to reach the global minima.

Algorithm

Step 1: We first start with an initial solution **s** = $\mathbf{S}_0$. This can be any solution that fits the criteria for an acceptable solution. We also start with an initial temperature **t** = $\mathbf{t}_0$.

Step 2: Setup a temperature reduction function *alpha*. There are usually 3 main types of temperature reduction rules:

1. Linear Reduction Rule: $t = t - \alpha$
2. Geometric Reduction Rule: $t = t * \alpha$
3. Slow-Decrease Rule: $t = \frac{t}{1+\beta t}$

Step 3: Starting at the initial temperature, loop through *n* iterations of Step 4 and then decrease the temperature according to *alpha*. Stop this loop until the *termination conditions* are reached. The termination conditions could be reaching some end temperature, reaching some acceptable threshold of performance for a given set of parameters, etc. The mapping of time to temperature and how fast the temperature decreases is called the **Annealing Schedule**.

Step 4: Given the neighbourhood of solutions *N(s)*, pick one of the solutions and calculate the difference in cost between the old solution and the new neighbour solution. The neighbourhood of a solution are all solutions that are close to the solution. For example, the neighbourhood of a set of 5 parameters might be if we were to change one of the five parameters but kept the remaining four the same.

Step 5: If the difference in cost between the old and new solution is greater than 0 (the new solution is better), then accept the new solution. If the difference in cost is less than 0 (the old solution is better), then generate a random number between 0 and 1 and accept it if it's under the value calculated from the Energy Magnitude equation from before.

In the Simulated Annealing case, the equation has been altered to the following:

$$P = \begin{cases} 1 & if \Delta c \leq 0 \\ e^{-\Delta c / t} & if \Delta c > 0 \end{cases}$$

Where the *delta c* is the change in cost and the *t* is the current temperature.

The *P* calculated in this case is the probability that we should accept the new solution.

High vs. Low Temperature

Due to the way the probability is calculated, when the temperature is higher, is it more likely that the algorithm accepts a worse solution. This promotes **Exploration** of the search space and allows the algorithm to more likely travel down a sub-optimal path to potentially find a global maximum.

When the temperature is lower, the algorithm is less likely or will not to accept a worse solution. This promotes **Exploitation** which means that once the algorithm is in the right search space, there is no need to search other sections of the search space and should instead try to converge and find the global maximum.

- Travelling Salesman Problem (TSP)
- Scheduling Problems
- Task Allocations
- Graph colouring and partitioning
- Non-linear function optimizations

Advantages vs. Disadvantages of SA

Advantages

- Easy to implement and use
- Provides optimal solutions to a wide range of problems

Disadvantages

- Can take a long time to run if the annealing schedule is very long
- There are a lot of tunable parameters in this algorithm