AI1001 - Assignment 15

Shivram S
ai24btech11031@iith.ac.in

1 Reinforcement Learning

In **reinforcement learning**, the learner is not told what actions to take, and must discover which actions yield the most reward. The problem is formalized using concepts from dynamical systems and optimal control of Markov decision processes.

In reinforcement learning there is a tradeoff between exploration and exploitation. The agent can take actions that it has tried in the past and found to be effective, but this prevents it from trying new actions and discovering potentially better selections. The agent must try a variety of actions and progressively favour those that appear to be best.

Unlike other fields of ML which focus on isolated subproblems, reinforcement learning focuses on the interactions of a decision-making agent which seeks to achieve a goal despite uncertainty about its environment. RL operates on simple general principles, and is close to the kind of learning that humans and other animals do.

There are four major subelements of a reinforcement learning system:

- The **policy** defines the agent's behaviour at a given time. Given the state of the element, it determines the action to be taken. It may be a simple lookup table or an extensive search process.
- The **reward signal** is a number that determines if the agent's course of action was "good" or "bad". It is the primary basis for altering the agent's policy.
- The **value function** is a way to specify the long-term value of a state. It is the total amount of reward the agent can expect to accumulate over the future, starting from that state. It is much harder to determine values, and value estimation is the most important component of almost all RL algorithms.
- An RL system might include a **model** of the environment, which mimics how the environment might behave. Models are used for **planning** - deciding an action by considering possible future situations. There are also **model-free** agents that only learn by trial-and-error.

In an RL system, state is a crucial part of the policy and value functions. Most RL systems are structured around estimating the value functions, but other methods such as solution methods, genetic algorithms, simulated annealing, etc. can also be used.

Evolutionary methods apply multiple randomized policies and use the policy which obtains the most reward as the starting point for the next generation. However, these methods ignore much of the useful structure of the reinforcement learning problem: they don't use the the fact that the policy is a function from states to actions, and they don't notice which states a system passes through during its lifetime, or which actions it takes.

Preprint. Under review.

2 Tic-Tac-Toe

Consider the game of tic-tac-toe against an imperfect opponent. We want to construct a player that will find the imperfections. The player must learn from experience rather than rely solely on predefined strategies such as minimax. One approach involves learning a model of the opponent's behavior through repeated play, then using dynamic programming to compute optimal moves based on the model.

Evolutionary methods can explore various policies through simulated games and choose one with a high probability of winning. A typical evolutionary method would **hill-climb** through policy-space. Alternatively, a genetic algorithm could be used, which would maintain and evaluate a population of policies.

For a method making use of a value function, we might set up a table of numbers, one for each possible state of the game. Each number would be the latest estimate of the state's value. The initial values could be assigned as 1 for states where we have won, 0 for states where we have lost, and 0.5 for every other state.

We can play many games against the opponent, mostly moving **greedily** by selecting the move that leads to the state with greatest value. Occasionally, we can play **exploratively** by selecting randomly from the other moves.

At the end of the game, we can "back up" through our tree of moves, and update the value $V(S_t)$ of the state S_t to bring it closer to the value $V(S_{t+1})$ of the next state S_{t+1}

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

α is a very small positive fraction called the **step-size** parameter, which influences the rate of learning. This update rule is an example of a **temporal-difference** learning method.

If the step-size parameter is reduced over time, then the model converges to any fixed opponent. If the parameter is not reduced at all, the player can also play well again opponents that slowly change their way of playing. Systems can also learn to set up multi-move traps for a shortsighted opponent.

This example shows how value function methods can use the structure of the reinforcement learning: value functions can give credit to independent moves, whereas evolutionary methods rewards the entire policy, including moves that never occurred.

Reinforcement learning is applicable even in scenarios without an explicit adversary and is not confined to problems with discrete time steps or distinct episodes. It can effectively handle situations where part of the state is hidden.

Reinforcement learning methods can be applied without a need for a model, but models can be easily used if they are available. Model-free approaches do not predict how the environment will react to an action, yet they can offer advantages over more complex models, particularly due to the challenges involved in creating an accurate model.

Reinforcement learning can also be used at higher levels. Although the tic-tac-toe player only learned the basic moves of the game, reinforcement learning can also work at higher levels where each "action" may itself be the application of a problem-solving method.