

Reinforcement Learning Algorithms

Reinforcement Learning is a branch of machine learning in which an agent learns to make optimal decisions by interacting with an environment. At each time step, the agent observes a state, takes an action, and receives feedback in the form of a reward. RL is particularly effective in scenarios where system dynamics are complex or uncertain and where closed-form optimization is infeasible. This section introduces two relevant RL approaches commonly used in control applications.

1. Modified Value Iteration Algorithm

The Modified Value Iteration Algorithm is a lightweight, model-based strategy designed to select optimal actions in a discretized state space. It leverages a pre-trained surrogate model to evaluate all possible actions for each state, based on a defined reward function. The algorithm performs a one-step lookahead by predicting the outcome of each candidate action and selecting the one that maximizes the immediate reward (e.g., minimal deviation from a target value). This method is highly effective when the surrogate model accurately captures the system dynamics, enabling fast and interpretable control decisions without requiring iterative training or real-time simulation feedback.

Greedy Surrogate-Based Control Algorithm

Algorithm: Greedy Surrogate-Based Control

Input:

$S \leftarrow$ Set of all discrete states
 $A \leftarrow$ Set of all possible actions
 $M \leftarrow$ Trained surrogate model that maps $(s, a) \rightarrow$ predicted outcome
 $R \leftarrow$ Reward function (e.g., $R \leftarrow -|\text{predicted} - \text{target}|$)

Output:

$\pi(s)$: A policy that assigns the best action to each state

```
1. for each  $s \in S$  do
2.    $\text{best\_reward} \leftarrow -\infty$ 
3.    $\text{best\_action} \leftarrow \text{None}$ 
4.   for each  $a \in A$  do
5.      $\text{input} \leftarrow \text{concatenate}(s, a)$ 
6.      $\text{prediction} \leftarrow M(\text{input})$ 
7.      $\text{reward} \leftarrow R(\text{prediction})$ 
8.     if  $\text{reward} > \text{best\_reward}$  then
9.        $\text{best\_reward} \leftarrow \text{reward}$ 
10.       $\text{best\_action} \leftarrow a$ 
11.   end for
12.  $\pi(s) \leftarrow \text{best\_action}$ 
13. end for
```

Figure: Modified Value Iteration Algorithm

2. Modified DQN Algorithm

The Modified DQN Algorithm builds on the greedy control concept by enabling generalization through supervised learning. Instead of relying on explicit action enumeration during deployment, this method first constructs an offline dataset of optimal (state, action) pairs using the surrogate model. A deep neural network is then trained on this dataset to learn a policy that maps state features to optimal actions directly. This approach offers a scalable solution for control problems, particularly when fast inference is needed at runtime or when the state space is too large for exhaustive search. The learned policy model effectively distills the behavior of the surrogate-based planner into a compact and reusable neural controller.

Surrogate-Based Policy Learning Algorithm

Algorithm: Surrogate-Based Policy Learning

Input:

$S \leftarrow$ Set of all discrete states
 $A \leftarrow$ Set of all possible actions
 $M \leftarrow$ Trained surrogate model
 $R \leftarrow$ Reward function

Output:

$\hat{\pi}(\cdot)$: A learned policy model mapping state features \rightarrow optimal action

1. Initialize dataset $D \leftarrow \emptyset$
2. for each $s \in S$ do
3. $\text{best_reward} \leftarrow -\infty$
4. $\text{best_action} \leftarrow \text{None}$
5. $\text{best_prediction} \leftarrow \text{None}$
6. for each $a \in A$ do
7. $\text{input} \leftarrow \text{concatenate}(s, a)$
8. $\text{prediction} \leftarrow M(\text{input})$
9. $\text{reward} \leftarrow R(\text{prediction})$
10. if $\text{reward} > \text{best_reward}$ then
11. $\text{best_reward} \leftarrow \text{reward}$
12. $\text{best_action} \leftarrow a$
13. $\text{best_prediction} \leftarrow \text{prediction}$
14. end for
15. $x_input \leftarrow \text{concatenate}(s, \text{best_prediction})$
16. $y_output \leftarrow \text{best_action}$
17. Add (x_input, y_output) to D
18. end for
19. Train a DNN $\hat{\pi}(\cdot)$ on dataset D to learn mapping: state \rightarrow optimal action

Figure: Modified DQN Algorithm