Ayan Abhiranya Singh

CS 297

Reinforcement Learning

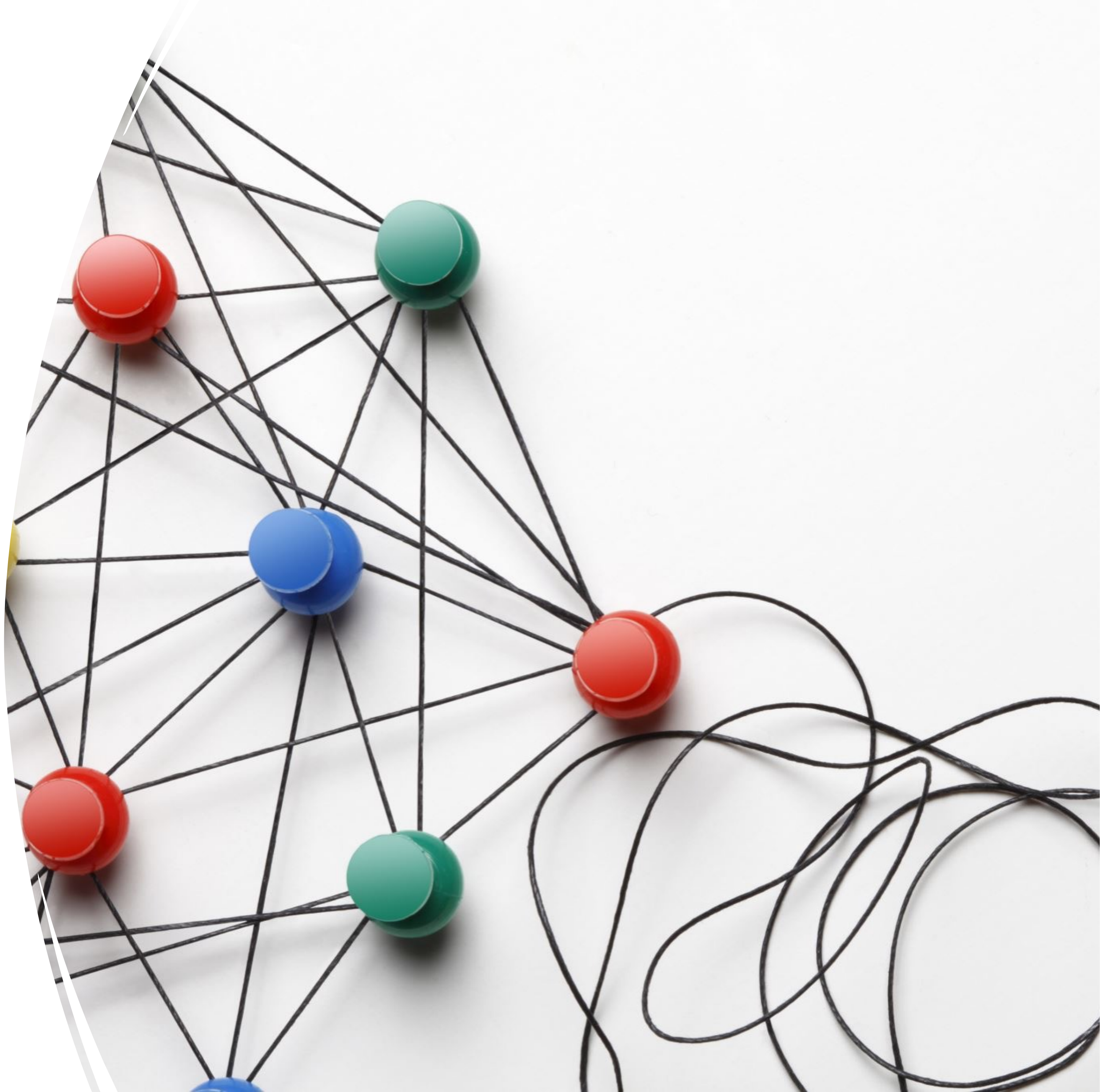
What is reinforcement learning?

Reinforcement Learning: Where an agent learns about a world through the delivery of periodic rewards.

Model-free reinforcement learning: Where an agent has no prior model/knowledge of an environment but instead acts on a quality-function, *Q(s,a),* that denotes the sum of rewards of from state *s* onwards if action *a* is taken.

Our problem

- We want to apply the reinforcement learning algorithm to solve the popular vacuum world problem.
- The Q-learning approach seems a viable method to train an agent to learn dynamic dirt generation policies.
- What is the Q-function?



Bellman Equation – the utility function

- The utility function is defined as the expected sum of the discounted rewards if a policy π is followed. The utility function is as follows:

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^{t} R(S_t, \pi(S_t), S_{t+1})\right]$$

Where $R(S_t, \pi(S_t), S_{t+1})$ is the reward received when action $\pi(S_t)$ is taken to transition between steps S_t to S_{t+1}.

Learning the model (Adaptive Dynamic Programming)

- We approach this as a supervised learning task.
- By running N (N is chosen by us) number of trials within the vacuum world, we can determine probabilities for how many times each square was visited en route to the goal.
- The model for P$(S'|\, s, a)$ is stored as a table and we maximize this value at each time step to obtain the next best approach to our goal.
- The input for each time step is a (state, action) pair, denoting a transition to a potential state s’.
- Limitations – ideal for smaller state spaces, not feasible for larger ones!

More thoughts on this approach…

This solution works for smaller state spaces.

This solution works for static policy systems.

However, what if the dirt generation policy changes at each time step?

Dynamic Policies

First, the learned model from earlier will have to be expanded for all possible outcomes at a given time step.

One step look-ahead: Look through all policies and pick out best option.

Policy iteration: the agent should simply execute the action that the optimal policy recommends.

Evolving our approach

- **Temporal difference Q-learning** combines policy iteration with the one-step look ahead, while eliminating the utility function and with it, the transition model.
- The Q-learning method, Q(s, a), denotes the expected total discounted reward if the action takes a in s and acts optimally thereafter (argmax Q(s, a)).
- A TD Q-learning agent does not need a transition model P(S' | s, a), either for learning or action selection.

$$Q(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma \max_{a'} Q(s',a')]$$