Code journal: Bellman's Optimality Equation For Perlin Noise State Space

13.06.2022, *Mariana Ávalos Arce*

Reinforcement Learning

Reinforcement Learning is a technique of Machine Learning that requires both **states and reward** for the representation of a problem. It associates **actions** with states, with the objective of finding the sequence of actions (trajectory) that rewards the biggest benefit possible of a given problem. An **agent** is the program in charge of solving such problem, for which there are different techniques.

In Reinforcement Learning, there are a few fundamental concepts for the modelling of a problem:

- Set of states: represented by set $S = \{s_0, s_1, ..., s_n\}$, it is the set of all possible states or environment configurations the agent can be in.
- Set of actions: represented by set $A = \{a_0, a_1, ..., a_m\}$, it is the set of all possible actions or methods for transitioning that the agent has available in order to move from one sate to another.
- Transition Model Function: represented as $s_f = f_{TM}(s, a)$ if it is a deterministic state space, or $s_f = P_{TM}(s_f|s, a)$ in the case of a non-deterministic state space. If an agent moves, by applying action a, from state s to state s_f with a **probability of arriving to s_f of 1.0**, the problem is deterministic. Otherwise, If an agent moves, by applying action a, from state s to many possible different states, the problem is non-deterministic.
- Reward Function: represented as $r = f_R(s, a, s_f)$, it is the function that returns a numeric reward for the agent when it transitions from state s to state s_f by applying action a. Usually a reward depends only on the arrival state, that is, $r = f_R(s_f)$.
- Policy: represented as $a = f_\pi(s)$ for a deterministic agent decision policy or as $a = P_\pi(a|s)$ for a non-deterministic agent decision policy. This refers to how the agent **takes an action** a while it is in state s, whereas the Transition Model Function refers to the **result** of an action a (already taken following $f_\pi(s)$) while the agent is in state s.

Once a problem is modelled by using the above concepts, there exists the question: how to train the agent to take the trajectory that gives the maximum reward possible?, how to train the agent to solve the problem? One of the answers to that question involves **Bellman's Opimality Equations**.

Bellman's Optimality Equations

Bellman's Optimality Equations are a set of n non-linear equations, where n is the number of states in the problem's environment, whose solution allows an **optimal policy** to be deduced that gives the agent the maximum reward possible, solving the problem. Bellman's Optimality Equations are defined as two versions: one for V and one for Q, depending on the amount of memory the program

is allowed to use, where V version uses less memory for the solution of the non-linear system of n equations but more processing time for the policy deduction, and Q version uses more memory for the solution of the non-linear system of n equations but the deduction of the policy is basically an immediate computation. Due to the small scope of the current experiments, Q Learning is used.

Bellman's Optimality Equations for Q Learning are are expressed for each state s and action a as:

$$Q(s,a) = \sum_{s_f \in S} P_{MT}(s_f|s,a) \cdot (f_R(s,a,s_f) + \gamma \max_{a_f} Q(s_f,a_f)), \tag{1}$$

where the sum of the product of the probabilities given by $P_{TM}(s_f|s,a)$ with the reward term refers to the non-deterministic state space problems, which allows the special case of deterministic state space equations to basically be:

$$Q(s,a) = (f_R(s,a,s_f) + \gamma \max_{a_f} Q(s_f,a_f)), \tag{2}$$

given that for a deterministic state space, the probability of arriving to state s_f from state s by applying a is 1, or $P_{TM}(s_f|s,a) = 1$, and zero for all remaining states in S.

Perlin Noise State Space