Requirements and Preliminary Design for Reinforcement Learning and Safety test bed.

Jonathan Azpur
York University
Toronto, Ontario, Canada

1. Introduction

We are looking to test and implement a strategy for safety in machine learning agents. In this project we will build a test bed in which an agent has to achieve some goal **G**. The implementation of the project will support a machine learning agent in the form of reinforcement learning that will evaluate its environment, and produce optimal strategies to achieve it's goal state from any possible initial state given in it's environment. In order to test safety strategies for this agent we will introduce danger situations into the environment and the algorithms/agent will have to work around these circumstances to obtain not only an optimal strategy but also a safe strategy. The project will be developed in Python.

2. Domain

The domain of this test bed will involve a planetary robot in a starting point (initial state $\mathbf{s}_{\mathbf{start}}$) that moves around in a grid world and who's task is to reach a target destination (a goal state $\mathbf{s}_{\mathbf{goal}}$). The grid world will have craters in which the robot can fall (safety concern) and borders that limit their movement to a finite number of locations. The robot, or agent, will have the ability to observe the state of its environment ($s \in \mathbf{S}$), and a set of actions (**A**) it can perform to alter this state. Here is a visual example of a grid world with a goal state (green square), crates (red squares) and and initial state (blue circle which is the robot):

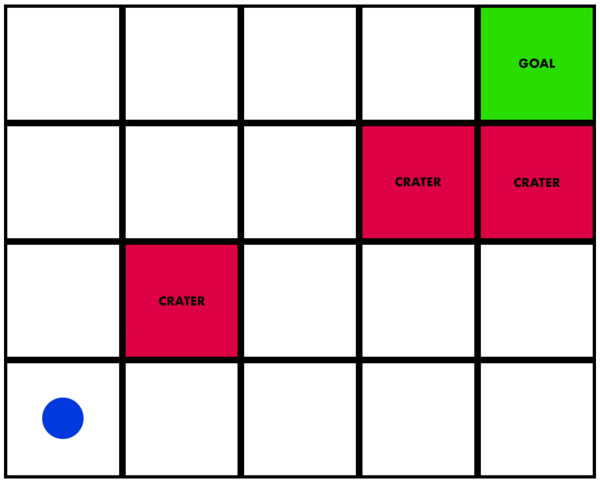

Figure 1.1 Grid map, with an initial state (blue circle), a goal state (green square) and a set of crates (red squares).

The algorithm will not be limited to this domain. But for test purposes we implement reinforcement learning in the context of a grid world. Our algorithm will be able to work on any circumstances as long as a set of states (including initial and goal state), actions, rewards and punishments are defined. We will be able to adapt our algorithm to any domain.

3. Algorithm

We will use the reinforcement learning algorithm **Q-learning** to train our agent to find the most efficient and safety path from it's initial state to the goal. The goal in reinforcement learning is to make an agent learn a safety strategy, or policy that will choose a set actions that achieve the agent's goal with maximum benefit (safe). So our robot will have to perform actions, observe their consequences, and learn a policy that will take it closer to it's goal by maximizing the rewards. Reward can be defined as the numerical value obtained from performing a specific an action at a specific state r: $A \times S \rightarrow \mathbb{R}$. The task of the agent is to learn a policy, y: $S \rightarrow A$, that will select an action a based on a current state s that will produce the greatest possible cumulative reward.

In the case of Q-learning we cannot directly calculate the maximum cumulative reward because we only have access to the agent's immediate rewards. Therefore we introduce the evaluation function Q(s,a) so that its value is the maximum cumulative reward that can be achieved starting from state s and applying action a as the first action. Because we do not have access to certain rewards (only immediate reward, reward of reaching goal state and punishment of falling into crater) our agent will have to start with an empty table of values Q'(s,a), explore all possible states/actions and create and optimal (approximate) table Q*(s,a) that will provide the best action to take in any possible scenario for the agent to reach it's goal and stay safe.

Input: S: set of states
A: set of actions
$0 \leq v < 1$: reward obtained by achieving the goal,
$0 \leq n < 1$: punishment obtained by falling into crater.
Algorithm:
1. Initialize matrix $Q^{|S|x|A|}$ to zero.
2. Explore environment and update Q until we learn Q*
Output: $Q^{|S|x|A|}$ and a path from the agent's initial state to the goal state. Figure 1.2 provides a visual example of the path our algorithm would output if it where to run on Figure 1.1.

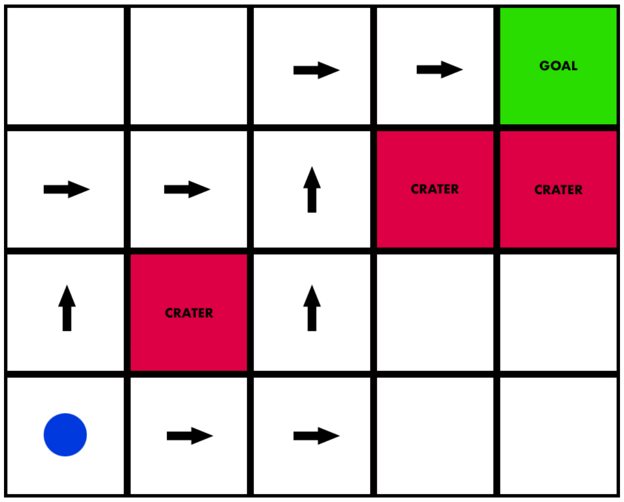

Figure 1.2 Best path from initial state to the goal using Q-learning algorithm

4. Safety

The grid map in which the planetary robot will be operating will have a certain number of crates, the safety concern will be to make sure the robot avoids falling into the crates while searching for it's goal. The crate will represent a dangerous situation by providing a negative reward (punishment) if the robot were to move in it's direction.

The weight of the punishment will be larger than the weight of reward provided by reaching to the goal state to ensure that the robot will avoid the crates. The balance between the positive reward of reaching the goal and the negative reward of falling into the crater will determine how efficient/safe the agent will be in it's solution. One wouldn't want to make the reward for the goal state much bigger because this would mean that the agent will put itself in risk in order to find the most efficient solution. On the other hand, we don't want to set the punishment for falling into a crate much larger because our agent would focus too heavily on safety and steer away from efficiency.

For other domains, where we do not have crater like in this test bed the algorithm will adapt to any environment as long as there is a negative reward value (punishment) to any state that challenges the safety of the agent.