RL Tutorial 1

Scott Brownlie

Problem 1

In supervised learning we have a training set of examples (x, y). Each x is an input and the corresponding y is its label. The label can be thought of as the correct answer for the input. The training set is generally fixed before learning and we wish to learn a mapping from inputs to labels which generalises to new unseen data from the same distribution as the training data.

In contrast, RL is goal-directed learning which is achieved through interaction with an environment. In RL we do not have a fixed training set. Instead, an agent generates data by interacting with an environment over discrete time steps. On each time step the agent is presented with a representation of its current state, x say, and it has to decided how to act. Unlike in supervised learning, there is no label y which tells the agent how to act. Instead, the agent chooses and executes an action and receives feedback from the environment in the form of a scalar reward. However, in many tasks the immediate reward may not convey the value of the action. Often rewards are delayed, but the reward signal should be designed in such a way that maximising the long term reward results in the agent achieving its goal. In the absence of supervised labels the agent must find the best action for each state in a trial and error fashion.

The answer given by Shaikh is flawed because it suggests that only in supervised learning do we have knowledge of the environment. In fact, in many reinforcement learning tasks we have complete knowledge of the environment, however, for all but the smallest problems learning without interaction is intractable.

Problem 2

One example of an RL application is autonomous vehicle control. For example, controlling the steering wheel angle of a car. We may try to model this problem as a supervised learning task where x is data from the car's sensors and y is the desired steering wheel angle for that situation. However, the world is so vast that collecting and labelling a useful training set would be pretty much impossible. It is much easier to design a reward signal which conveys "good driving" at a high level than specify optimal low level controls for every possible situation. For example, we may give a positive reward for the car staying within its lane and a large negative reward if it hits another vehicle. An autonomous agent could then learn to drive by maximising the long term reward while interacting with a driving simulator.

Another potential application is deciding which push notifications to send to customers. This type of problem almost certainly involves delayed rewards. For example, a customer may receive a push notification which catches his/her interest but decides to take no immediate action. A subsequent push at a later time may ultimately trigger the action and generate a reward, but the earlier push was responsible for starting the customer off down the "sales funnel". RL allows an agent to assign credit to both pushes. If we were to model this problem as supervised learning then we ourselves would have to assign credit by defining some crude rule or manually labelling the push notifications in a time consuming and tedious process.

Problem 3

There are two non-terminal states $rock_0$ and $rock_1$ and a single terminal state *land*. Suppose that the rocks are arranged from left to right. When the frog is in state $rock_0$ it can only jump to the right and with probability 0.9 it transitions to $rock_1$, else it transitions to back to $rock_0$ (after climbing out of the water). When the frog is in state $rock_1$ it can either jump to the left or to the right. If it jumps to the left then it transitions to $rock_0$ with probability 0.9, else it transitions back to $rock_1$. If it jumps to the right then it transitions to $land$ with probability 0.9, else it transitions back to $rock_1$.

We assume that the frog's goal it to reach land as quickly as possible. We also assume that it takes the frog twice as long to climb out of the water as it does to jump from a rock. We therefore define the reward for each $state \to action \to nextstate$ transition as

- $rock_0 \to jump_{right} \to rock_0$: -3
- $rock_0 \to jump_{right} \to rock_1$: -1
- $rock_1 \to jump_{left} \to rock_1$: -3
- $rock_1 \to jump_{left} \to rock_0$: -1
- $rock_1 \to jump_{right} \to rock_1$: -3
- $rock_1 \to jump_{right} \to land$: 0.

In this way the cumulative reward is essentially one minus the time taken for the frog to reach land. Hence, by maximising the cumulative reward the frog reaches land in the shortest possible time, achieving its goal.