Reinforcement Learning

Miles Kovach

There are many approaches to machine learning, each with tasks to which it is best suited. Here we take a look at the concept of reinforcement learning, a mode of machine learning very similar to how animals have been observed to learn[1]. We then consider how and why we might implement a strategy like this on an assembly-line robot with cameras and proximity sensors equipped.

This similarity -and the difference between reinforcement learning and other methods of machine learning- is mainly due to the fact that an implementation of reinforcement learning assumes that the algorithm has some level of control over its environment[2]. An ordinary supervised learning algorithm relies on labels to improve its performance, and an unsupervised algorithm uses its training set to find patterns in unfamiliar sets of data. A unit trained by reinforcement learning differs in that it is given a "reward" mechanism for completing some desired task, and such a task will initially be completely unfamiliar to the unit.

The unit begins learning its task by acting in a random or only partially-optimized manner. For each step that it takes towards its goal -i.e. maximum reward-, it takes in two vectors: its *Value function* and its *state*[3]. Its state is its input from its environment. Some examples might include positioning of one's and one's opponent's pieces in a game of checkers, or input from an array of sensors. Both indeed, in the case of a checker-playing robot.

Such a checker-playing robot would need at least one *value function*: a vector which expresses how favorable a given state/next-action pairing is. Often, this takes the form of an expected value (plus any anticipated rewards) of the next state given a specific action (or some greater number of subsequent states and actions), probabilistically.

Actions are decisions that our autonomous learner can make which will result in an altered environment. Moving a checker in software, or physically moving it with a robotic limb, are examples of actions. It should be noted, though, that the latter is almost certain to be a set of actions, each with its own reward structure and set of values.

This may seem impractical at first, and for some tasks, it will be. Reinforcement learning has a distinct advantage if, for example, you want to train an agent- that is, an algorithm running independently or in physical space- to be able to perform multiple similar tasks while receiving feedback. In the case of our assembly line robot, for instance, we would see a significant decrease in the costs of reworking an assembly line to produce a new product after an older one has reached its end-of-production stage. Rather than making costly and wasteful replacements of equipment[4], a manufacturer would only need to arrange for a select technically qualified staff to perform some reprogramming before subjecting the robots to independent training.

In this implementation, we will have to identify solutions to some key problems regarding algorithm selection. Will our algorithm be:

- deterministic or stochastic?
- independent or cooperative with other AI?
- trained virtually with a model, or on "real" equipment?

Identifying the answers to these and some of the second-tier questions will help us to identify an appropriate set of constraints with which we choose an algorithm. Given the constraint -at least within the current scope- that building and testing the algorithm on actual equipment is infeasible, using an open-source testing software like "openai gym" seems to be a good way to move forward. Eventually we may need to develop our own simulations and determine how reliably these can be applied to the real world.

As we are dealing with a real-world scenario, as opposed to a game or other theoretical exercise, our learning scheme would probably benefit from being stochastic, at least initially. Our reinforcement learner would also likely be paired with, say, a neural network to perform static tasks, like image recognition from the cameras. Once an optimum had been settled on, then we could lock the algorithm in place for production.

1 Stoytchev, A., "Robot Tool Behavior: A Developmental Approach to Autonomous Tool Use", Ph.D. Dissertation, College of Computing, Georgia Institute of Technology, August 2007.
2 Reinforcement Learning: An Introduction. second edition Richard S. Sutton and Andrew G. Barto
3 MIT 6.S091: Introduction to Deep Reinforcement Learning (Deep RL), https://www.youtube.com/watch?v=zR11FLZ-O9M
4 Francis, Sam, "Farewell to the old assembly line: Automation is changing manufacturing processes in the automotive industry", Robotics and Automation News, https://roboticsandautomationnews.com/2020/03/12/farewell-to-the-old-assembly-line-automation-is-changing-manufacturing-processes-in-the-automobile-industry/31270/ retrieved 2020-09-21