Reinforcement learning applications: what’s under the hood?

Originally published at https://www.itransition.com/blog/reinforcement-learning-applications

Published: June 11, 2021 | By Martin Anderson Independent AI expert

Reinforcement learning (RL) is a technique in machine learning where an automated system learns to achieve a useful task by performing it over and over again until it gets it right. It's a 'brute force' approach, and the machine doesn't usually learn anything else in the process; if you teach it how to unscrew a bottle-cap, it doesn't learn anything about bottles, caps, or even necessarily how to put the cap back on the bottle again (which involves different torque variables, and would be a whole new course of study for it).

On the other hand, a machine trained in this way has a number of notable advantages over other popular machine learning techniques:

- It can teach itself to achieve a goal without human supervision or intervention, through a linear process of trial-and-error.
- It requires little or no prior knowledge of the environment.
- It can, as necessary, adapt to changes in an environment that it has already mapped, without needing to retrain itself from zero. Again, it can do this autonomously.
- Unlike supervised and unsupervised learning, this process does not usually entail the exhaustive gathering, pre-processing, curation and training of high-volume datasets; the machine develops its own streamlined dataset of 'optimal' actions for any one task.
- It can be deployed as a useful component within more complex machine learning frameworks, as necessary.

In this article, we'll take a look at how RL is rising across sectors, how it works, where it is a viable approach in machine learning consulting, and which use cases it might best be suited for.

The fall and rise of reinforcement learning

As interest around complex neural networks and other model-intensive deep learning techniques increased towards the end of the 2000s, it seemed that reinforcement learning would be left behind as an artefact of robotic process automation.

As it transpires, however, RL is a remarkably robust approach to solving well-defined challenges, and a useful adjunct in data exploration methodologies. As the neural network sector has hit increasing bottlenecks (many of which involve the high cost of data acquisition and curation), reinforcement learning appears to be on the rise again—partly as a component of the newer systems that once threatened to replace it, in the form of deep reinforcement learning (DRL).

AlphaGo revives interest in reinforcement learning

The 2016 turnaround of interest in reinforcement learning seems to issue from the headline-grabbing success of Google’s AI system AlphaGo in defeating one of the best Go players of all time. Subsequent to the AlphaGo victory, Kathryn Hume and Matthew E. Taylor of Harvard Business Review suggested that RL could be the most logical machine learning technology for a wide range of AI use cases, from stock inventory system planning to determining the correct reagent for a molecule simulation, or managing data center infrastructure.

In regard to data centers, RL received an added PR fillip in 2018 when Google publicized a 40% reduction in data center cooling costs through reinforcement learning experiments.

Reinforcement learning in Fortune 1000 companies

Reinforcement learning is a core technology across a wide range of sectors, used by a growing number of Fortune 1000 companies:

- US-based electronic design automation company Synopsys uses reinforcement learning as the engine for DSO.ai (Design Space Optimization AI), which identifies solution spaces in chip design.
- J.P. Morgan uses reinforcement learning as a core technology in its deep neural network for Algo Execution (DNA) market pricing toolset.

- Formal Verification provider JasperGold uses online and offline reinforcement learning in its JasperGold Expert System.
- Starbucks uses reinforcement learning in its provisioning routines.
- Cybersecurity provider Fortinet employs RL to 'prove' results obtained from supervised and unsupervised learning frameworks.
- Dell uses reinforcement learning to help allocate critical system resources in its storage solutions.

Google Brain makes use of reinforcement learning to select optimal placement of components in proprietary microchip and ASIC architectures.

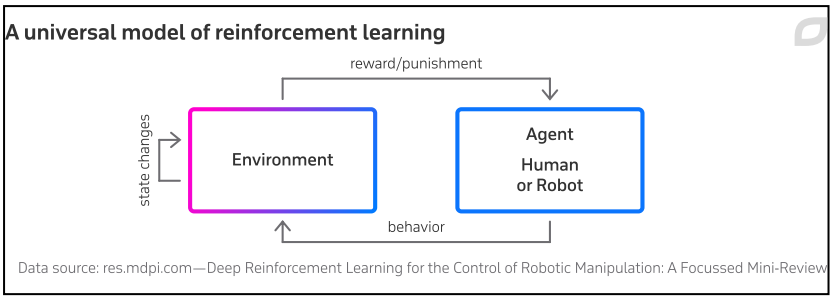
Core principles of reinforcement learning

Reinforcement learning has a fairly simple core architecture broken down into four components:

1: Agent

The *agent* is the exploratory capacity of the system. It navigates the data environment in a linear way, like a player trying out a video game without any prior instruction, and re-spawning near the point of its most recent defeat after it gets 'killed' (i.e. fails in its objectives). It remembers what got it killed before, and will try something else next time.

Sometimes the agent will make significant progress through a branch of decisions, only to find that the branch itself leads away from the main objective; in such cases, the agent (or, as necessary, the host framework) will recursively 'reset' to an earlier point than the most recent failure, in order to establish which branch-juncture marked the road to a 'dead end', and then proceed in a different direction.



2: Environment

Continuing the video game analogy, the data that the goal-oriented agent must traverse is known as the *environment*. The agent builds up and retains a map of the environment as its failures mount and its knowledge increases.

The environment could be literal or abstract, such as the 4D coordinates of a robot learning to pour a glass of water, combined with feedback video and tactile or haptic data; a series of raster images that comprise footage from a video game (of any complexity); the semi-mapped, darkly-lit roads that an autonomous vehicle must safely navigate; or any other data architecture or real-world environment that lends itself to repetitive, procedural feedback learning of this kind.

3: Actions

The actions in reinforcement learning are decisions that the agent makes, from quickly braking through to moving a video game cursor up or down, or tilting a robotic armature one way or another.

4: Policy

Policy in reinforcement learning is the set of rules and/or objectives that define the agent's mission, and decide the way that the agent will be educated as it makes mistakes on its journey.

The agent can be rewarded positively or negatively; unlike Pavlovian response in human psychology, the difference is relatively semantic, since the machine's motivational architecture is likely to have been designed to fit either methodology (or a mix of both). Nevertheless, negative reinforcement can help the agent learn and avoid unfruitful approaches, at the possible expense of inhibiting its general willingness to explore.

Combining policies

The training of an RL system may involve multiple policies, prioritized diversely—a schema illustrated in the Three Laws Of Robotics set out by science-fiction author Isaac Asimov in the 1940s:

- 'A robot may not injure a human being or, through inaction, allow a human being to come to harm.'
- 'A robot must obey orders given it by human beings except where such orders would conflict with the First Law.'
- 'A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.'

Examined closely (and even without a stated *goal* of any kind), these three rules comprise *at least* seven possible considerations that an agent must take into account before deciding on an action. Therefore the potential for internal dissonance in more complex rule-sets is obvious, making policy 'collisions' arguably the most famous staple of AI in science-fiction.

Policy gradients

It's clear, then, that defining an effective and versatile policy architecture is necessary in order to ensure that meeting the objective does not entail unacceptable 'risk' or a catastrophic logical conflict in the system.

Furthermore, due to the relatively 'blind' nature of reinforcement learning, policies are *likely* to encounter unexpected situations, but will nonetheless need to provide the agent with practical guidance to prevent over-frequent 'stop' scenarios or the entire re-tooling of the policy or the process.

To this end, Policy Gradient Theory provides a method of developing rules that can operate in various contexts and that can change the value of the agent's end-goal/s as and when necessary.

To understand the need for policy gradients, we should consider that there are two types of available policy in reinforcement learning:

- **Deterministic Policy**
  …where we can be certain that the action an agent takes will have a definite desired outcome, such as the decision to tilt a full beaker by 90 degrees, which will definitely lead to some of its liquid being poured out.
- **Stochastic Policy**
  …where the environment and/or the result of actions have not necessarily been derived from known physical laws, as with the previous example; are not 'certain'; and where the agent must evaluate its choices more carefully—a process known as Partially Observable Markov Decision Process (POMDP, see 'Markov Decision Process' below).

Deterministic approaches can operate outside of any policy framework and are often characterized as a 'direct' route to quick results in reinforcement learning; but, like many of life's shortcuts, they come with a number of caveats, as we'll see shortly.

Markov Decision Process (MDP)

A Markov Decision Process (MDP) decides the next best move based on the current circumstances, or 'state'. Under a basic MDP, an agent will evaluate the state it finds itself in, take action, and move to the next state. It will not necessarily consider the successful actions that brought it to its current progress point when evaluating the next state.

Though MDP is computationally efficient and semantically very lean, it can be myopic sometimes, favoring the attainment of short-term goals without broader consideration for the longer-term objective of the experiment.

This is why complex policy architecture is often needed in order to correctly calibrate the way the agent will navigate the data environment. To ensure that the agent does not prioritize short-term gains over long-term objectives, discounted reward can be used, among other techniques, as a balancing agent in policy.

Q-Learning

In contrast to the Markov Decision Processes that underpin stochastic reinforcement learning, Q-Learning is an 'off-policy', deterministic method that takes an even more solipsistic approach to RL.

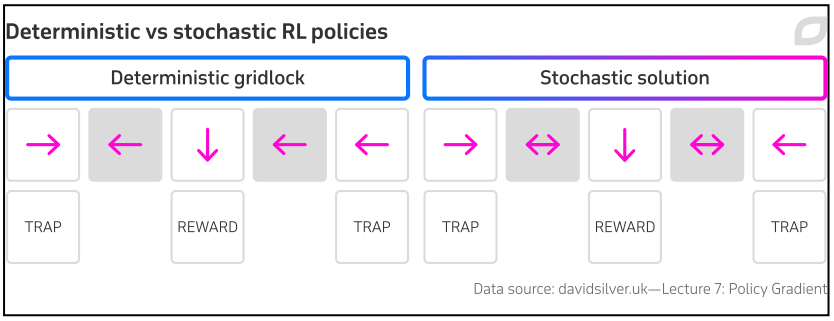
Q-Learning learns directly from actions, regardless of *whether those actions are covered by a policy or not*, and attempts to maximize reward for the agent by basing decisions on current circumstances. Q-Learning is therefore a value-based method (or *value function*) rather than a policy-based method.

Though it may seem like a more pragmatic and direct solution to navigating decisions, Q-Learning has some limitations. For one, the effect of a small change in actions can cause spikes in training, since the consequences of different actions have not been learned and pre-calculated as with a policy gradient method, which iterates more cleanly through the potential range of actions up to the optimal result.

In their haste, deterministic methods can get stuck

Additionally, a policy gradient can integrate a stochastic policy, while a value function (Q-Learning) cannot. This can make a value function a bit of a blunt instrument by comparison or cause it to fail entirely.

For instance, in the below example from Google DeepMind's researcher David Silver, we see a deterministic approach falling foul of *perceptual aliasing*, where two states seem to be identical but require different actions. In this case, a rigid, *deterministic* approach (pictured top) will either move west or east in both grey states (red arrows), because it does not comprehend the usefulness of moving backwards.



In this way, the deterministic approach will spend a long time analyzing the potential corridor of movement, and even then may get stuck and never reach the reward.

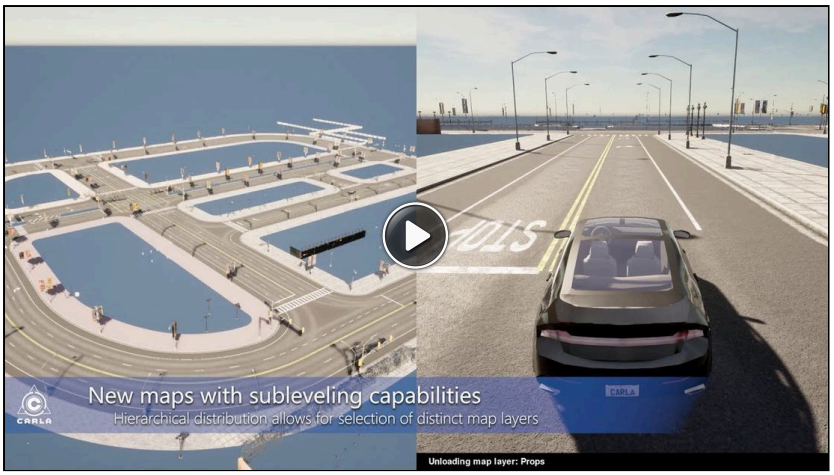
By contrast, a stochastic approach (pictured on the right in the same image) will randomly move in various directions in the grey states and will reach the goal quickly, in just a few steps.

3 popular use cases for reinforcement learning

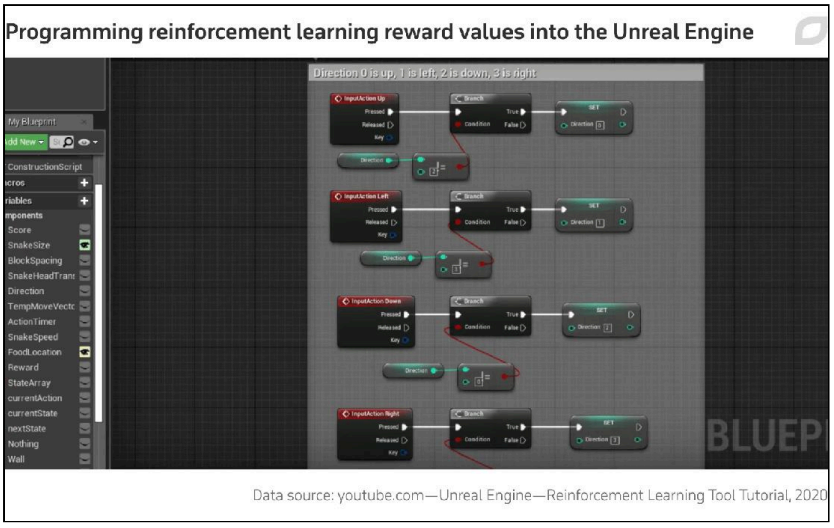
Autonomous technologies

Though autonomous ground and flight vehicles may rely on labor-intensive offline data curation and model training, they often require real-time adaptive capacity in the face of obstacles, both literal and figurative. Therefore reinforcement learning features prominently in such architectures.

A number of popular simulation environments are available for testing reinforcement learning techniques for autonomous vehicle technologies. These include the open-source Carla, which provides a complex urban world for agents to navigate. Carla features a dedicated RL agent, with the inference code available on GitHub.



A similar environment is available in the AI simulator Voyage Deep Drive, which supports deep reinforcement learning on OpenAI baselines and integration with UnrealEnginePython, a port of the popular game engine that can be incorporated directly into a Python framework.



Amazon's DeepRacer gives AWS developers the opportunity to drive an actual prototype car around a track in service of improved RL algorithms.

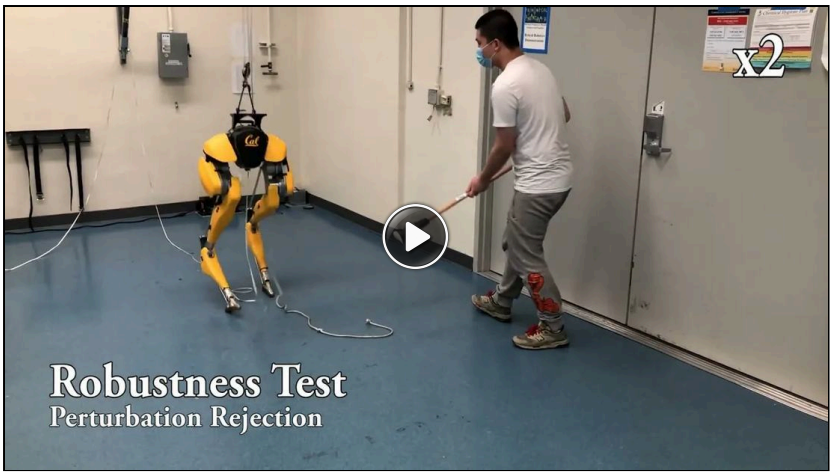
DeepRacer does have a simulation platform but concentrates primarily on interaction with the real-world sandbox created for the vehicle.

Robotics movement and manipulation

Since robots must interact with the real world, reinforcement learning is well-represented in robotics research, though not necessarily in the best-known examples.

Reinforcement learning is not a major component in the 'real-world' testing of Boston Dynamics' crowd-pleasing free-moving robots, which are far too expensive to risk on the turn of a Markov chain, and whose RL-derived algorithms are rigorously vetted in virtual environments first.

However, in March 2021, UC Berkeley revealed a truncated bipedal robot designed to be robust and affordable enough to withstand the process, and which taught itself to walk entirely by trial and error through reinforcement learning.



In controlled environments where industrial robots can operate from fixed stances under less precarious circumstances, reinforcement learning is among the leading machine learning technologies currently being researched.

In March 2021, Amazon, among the current industry leaders in robotics research, launched SageMaker Reinforcement Learning Kubeflow Components, a secondary toolkit for its RoboMaker development environment for robotics engineers and researchers. SageMaker operates on a Kubernetes cluster within an AWS environment.

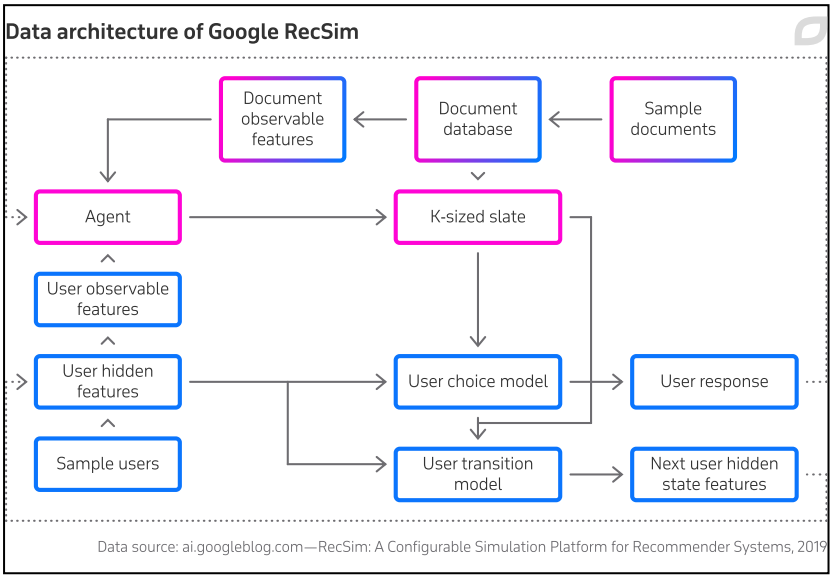
Recommender systems

Reinforcement learning-based recommender systems (RLRSs) are becoming a popular alternative (or at least an adjunct) to the matrix factorization-based deep learning approaches which have come to dominate this sector.

It's a relatively controversial movement, since RL seeks a single goal, which leads by default to a single recommendation, whereas a recommendation system is largely designed to provide a delimited list of ranked recommendations through a complicated process of domain evaluation and keyword/feature analysis. This trend towards RL is largely driven by the popularity of Q-Learning as a highly optimized and prosaic method of obtaining an item>item match.

Over the last two years Netflix has been running a growing series of tests for the use of reinforcement learning across its consumer and back-end systems, and currently uses RL to help choose artworks for the various GUIs of its app ecostructure.

Google, a world leader in AI research, is taking the potential of reinforcement learning in recommender systems very seriously: in late 2019, Google AI launched RecSim, a configurable platform for the development of recommender systems that use sequential interaction and to explore the possibilities that RL offers to one of the hottest sectors in tech.



Conclusion

Some challenges cannot be usefully or economically addressed solely by the 'linear' methods of reinforcement learning. Therefore, this approach does not automatically represent the 'least complicated' solution that a potential machine learning architecture might provide.

Furthermore, the applicability of an RL approach will depend greatly on how compatible your data architecture or 'central problem' is, and how abstract the challenge is. If you're looking for hidden relationships in high volume data (i.e., you don't know exactly what it is you're expecting to find), RL could prove a blind alley, except as a supporting tool.

On the other hand, if your goal is specific, exclusive and can be mapped and quantified, reinforcement learning may indeed be the most direct and economical approach.