**REINFORCEMENT LEARNING**

**-SAUMYA ARORA**

**ABSTRACT**

Reinforcement learning is no doubt a cutting-edge technology that has the potential to transform our world. However, it need not be used in every case. Nevertheless, reinforcement learning seems to be the most likely way to make a machine creative – as seeking new, innovative ways to perform its tasks is creativity. This is already happening: DeepMind’s now-famous AlphaGo played moves that were first considered glitches by human experts but secured victory against one of the strongest human players, Lee Sedol.  
Thus, reinforcement learning has the potential to be a ground-breaking technology and the next step in AI development.

**TABLE OF CONTENTS**

1. WHAT IS RL?
2. HOW DOES RL WORK
3. APPLICATIONS AND EXAMPLES OF RL
4. CHALLENGES OF APPLYING RL
5. COMMON RL ALGORITHMS
6. HOW IS RL DIFFERENT FROM SUPERVISED AND UNSUPERVISED LEARNING?
7. BASIC ALGORITHMS
8. **TYPES OF REINFORCEMENT LEARNING**
9. **WHEN TO USE RL?**
10. **WHEN NOT TO USE RL?**
11. **IS RL IS THE FUTURE OF ML?**

**WHAT IS REINFORCEMENT LEARNING?**

**Reinforcement learning (RL) is an area of machine learning concerned with how intelligent agents ought to take actions in an environment to maximize the notion of cumulative reward.** **It is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.**

Reinforcement learning differs from supervised learning in not needing labeled input/output pairs to be presented, and in not needing sub-optimal actions to be explicitly corrected. Instead, the focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).

**The environment is typically stated in the form of a Markov decision process (MDP) because many reinforcement learning algorithms for this context use dynamic programming techniques.** The main difference between the classical dynamic programming methods and reinforcement learning algorithms is that the latter does not assume knowledge of an exact mathematical model of the MDP and they target large MDPs where exact methods become infeasible.

**HOW DOES RL WORK?**

In reinforcement learning, developers devise a method of rewarding desired behaviors and punishing negative behaviors. **This method assigns positive values to the desired actions to encourage the agent and negative values to undesired behaviors.** This programs the agent to seek long-term and maximum overall reward to achieve an optimal solution.

These long-term goals help prevent the agent from stalling on lesser goals. With time, the agent learns to avoid the negative and seek the positive. **This learning method has been adopted in artificial intelligence (**[**AI**](https://searchenterpriseai.techtarget.com/definition/AI-Artificial-Intelligence)**) as a way of directing unsupervised machine learning through rewards and penalties.**

**APPLICATIONS AND EXAMPLES OF RL**

While reinforcement learning has been a topic of much interest in the field of AI, its widespread, real-world adoption and application remain limited. Noting this, however, research papers abound on theoretical applications, and there have been some successful use cases.

Gaming is likely the most common usage field for reinforcement learning. It is capable of achieving superhuman performance in numerous games**. A common example involves the game Pac-Man.**

Reinforcement learning can operate in a situation as long as a clear reward can be applied. In enterprise resource management ([ERM](https://searchsap.techtarget.com/definition/ERM)), reinforcement learning algorithms can allocate limited resources to different tasks as long as there is an overall goal it is trying to achieve. A goal in this circumstance would be to save time or conserve resources.

In robotics, reinforcement learning has found its way into limited tests. This type of machine learning can provide robots with the ability to learn tasks a human teacher cannot demonstrate, to adapt a learned skill to a new task, or to achieve optimization despite a lack of analytic formulation available.

It is also used in operations research, information theory, game theory, control theory, simulation-based optimization, multiagent systems, swarm intelligence, statistics, and genetic algorithms.

**CHALLENGES OF APPLYING RL**

Reinforcement learning, while high in potential, can be difficult to deploy and remains limited in its application. [**One of the barriers to the deployment**](https://searchenterpriseai.techtarget.com/feature/Key-considerations-for-operationalizing-machine-learning)**of this type of machine learning is its reliance on an exploration of the environment.**

**For example,** if you were to deploy a robot that was reliant on reinforcement learning to navigate a complex physical environment, it will seek new states and take different actions as it moves. It is difficult to consistently take the best actions in a real-world environment, however, because of how frequently the environment changes.

**The time required to ensure the learning is done properly through this method can limit its usefulness and be intensive on computing resources.** As the training environment grows more complex, so too do demands on time and compute resources.

[Supervised learning](https://searchenterpriseai.techtarget.com/definition/supervised-learning) can deliver faster, more efficient results than reinforcement learning to companies if the proper amount of data is available, as it can be employed with fewer resources.

**COMMON RL ALGORITHMS**

[Rather than referring to a specific algorithm](https://searchenterpriseai.techtarget.com/feature/5-types-of-machine-learning-algorithms-you-should-know), the field of reinforcement learning is made up of several algorithms that take somewhat different approaches. The differences are mainly due to their strategies for exploring their environments.

1. **State-action-reward-state-action (SARSA): -**

This reinforcement learning algorithm starts by giving the agent what's known as a policy. The policy is essentially a probability that tells it the odds of certain actions resulting in rewards, or beneficial states.

1. **Q-learning: -** This approach to reinforcement learning takes the opposite approach. The agent receives no policy, meaning its exploration of its environment is more self-directed.
2. **Deep Q-Networks: -** These algorithms utilize neural networks in addition to reinforcement learning techniques. They utilize the self-directed environment exploration of reinforcement learning. Future actions are based on a random sample of past beneficial actions learned by the neural network.

**HOW IS RL DIFFERENT FROM SUPERVISED AND UNSUPERVISED LEARNING?**

Reinforcement learning is considered its branch of machine learning, though it does have some similarities to other types of machine learning, which break down into the following four domains:

1. **Supervised learning: -** In supervised learning, algorithms train on a body of labeled data. Supervised learning algorithms can only learn attributes that are specified in the data set. Common applications of supervised learning are image recognition models. These models receive a set of labeled images and learn to distinguish common attributes of predefined forms.
2. **Unsupervised learning: -** In unsupervised learning, developers turn algorithms loose on fully unlabelled data. The algorithm learns by cataloging its observations about data features without being told what to look for.
3. **Semi-supervised learning: -**This method takes a middle-ground approach. Developers enter a relatively small set of labeled training data, as well as a larger corpus of unlabelled data. The algorithm is then instructed to extrapolate what it learns from the labeled data to the unlabelled data and draw conclusions from the set as a whole.
4. **Reinforcement learning: -** This takes a different approach altogether. It situates an agent in an environment with clear parameters defining beneficial activity and nonbeneficial activity and an overarching endgame to reach. **It is similar in some ways to supervised learning in that developers must give algorithms specified goals and define rewards and punishments.** This means the level of explicit programming required is greater than in unsupervised learning. But, once these parameters are set, the algorithm operates on its own, making it much more self-directed than supervised learning algorithms. For this reason, people sometimes refer to reinforcement learning as a branch of semi-supervised learning, but in truth, it is most often acknowledged as its type of machine learning.

**BASIC ALGORITHMS**

1. **PREDICTION ALGORITHMS**

 At the core of most RL algorithms lies the method of Temporal Differences (TD). We consider a sequence of states followed by rewards

***st*, *rt*+1, *st*+1, *rt*+2,…,*rT*,*sT*.**

The complete return *Rt* to be expected in the future from state **st** is, thus

***Rt*=*rt*+1+*γ*1*rt*+2+…+*γT*−*t*−1*rT*,**

where ***γ*<1** is a discount factor (distant rewards are less important). Reinforcement learning assumes that the value of a state *V*(*s*) is directly equivalent to the expected return

**V(s)=Eπ(Rt|st=s),**

where ***π*** is here an unspecified action policy. Thus, the value of state *st* can be iteratively updated with

***V*(*st*)→*V*(*st*)+*α*[*Rt*−*V*(*st*)],**

where ***α*** is a step-size (often =1). Note, if ***V*(*st*)** correctly predicts the expected complete return ***Rt*,** the update will be zero on average and we have found the final value for ***V***. This method requires waiting until a sequence has reached its terminal state before the value-update can commence. For long sequences, this may be problematic. However, given that **E(Rt)=E(rt+1)+γV(st+1**) we can also update iteratively by

***V*(*st*)→*V*(*st*)+ :*α*[*rt*+1+*γV*(*st*+1)−*V*(*st*)] ,**

which is the TD (0) procedure. The elegant trick is to assume that, if the process converges, the value of the next state ***V*(*st*+1)** should be an accurate estimate of the expected return downstream to ***st*+1.** We define the *δ*-error as

***δt*=[*rt*+1+*γV*(*st*+1)−*V*(*st*)]**

Normally we would only assign a new value to one state by performing***V*(*st*)→*V*(*st*)+*αδ*,** not considering any other previously visited states. This, however, can be desirable and can be achieved by so-called eligibility traces ***E*,** which are used to update earlier visited states "a little bit". We define ***Et*=1** at the currently visited state and let E decay gradually along states visited in the past with a decay factor ***λ*=1**, so we can define

**V(st+j)→V(st+j)+αδjEt+j**

**for *j*≥0.** This procedure is known as backward -TD(*λ*)*.* If ***λ*=1** then we are equally considering all previously visited states.

1. **CONTROL ALGORITHMS**

**SARSA** (initially known as **modified Q-learning** Rummery and Niranjan, 1994): Probably the nicest aspect of TD-formalism is that it can be used almost unaltered to address the control problem. We note first that the value of state-action pairs is given by the same formal expectation value of an expected total return *Rt* as before:

**Q(s,a)=Eπ(Rt|st=s,at=a) .**

The difference is that we have to calculate this now assuming that at the moment *t* we are visiting state*s* from where we take the specific action ***a*,** whereas above the action was left unspecified. The same **TD (0)** rule can be used to approximate *Q* with

***Q* (*st*, *at*) →*Q* (*st*, *at*) +: α [*rt*+1+*γQ* (*st*+1, *at*+1) −*Q* (*st*, *at*)]**

To calculate this, we must for t and t+1 go through the transition: **state, action, reward, state, action**; which gives this update rule its name **SARSA**. This method starts with a policy *π* which is continuously updated during learning (on-policy update).

[**Q-learning**](http://www.scholarpedia.org/w/index.php?title=Q-learning&action=edit&redlink=1)**: Uses the rule**

***Q* (*st*, *at*) →*Q* (*st*, *at*) +: α [*rt*+1+*γ*max*a* [*Q* (*st*+1, *a*)] −*Q* (*st*, *at*)]**

Taking the maximum across all actions a which are possible at state *st* seems to be only a minor modification as compared to SARSA. In effect, however, it makes learning independent of the starting policy *π* and it allows keeping this policy throughout the whole learning process.

**TYPES OF REINFORCEMENT LEARNING**

Two kinds of reinforcement learning methods are:

1. **Positive:**

It is defined as an event, that occurs because of specific behavior. It increases the strength and the frequency of the behavior and impacts positively on the action taken by the agent.

This type of Reinforcement helps you to maximize performance and sustain change for a more extended period. However, too much Reinforcement may lead to over-optimization of the state, which can affect the results.

1. **Negative:**

Negative Reinforcement is defined as the strengthening of behavior that occurs because of a negative condition that should have stopped or avoided. It helps you to define the minimum stand of performance. However, the drawback of this method is that it provides enough to meet up the minimum behavior.

**WHEN TO USE RL?**

Here are prime reasons for using Reinforcement Learning:

1. It helps you to find which situation needs an action
2. Helps you to discover which action yields the highest reward over a longer period.
3. Reinforcement Learning also provides the learning agent with a reward function.
4. It also allows it to figure out the best method for obtaining large rewards.

**WHEN NOT TO USE RL?**

1. You can't apply the reinforcement learning model in all the situation. Here are some conditions when you should not use the reinforcement learning model.
2. When you have enough data to solve the problem with a supervised learning method
3. You need to remember that Reinforcement Learning is computing-heavy and time-consuming. in particular when the action space is large.

**IS RL IS THE FUTURE OF ML?**

Although reinforcement learning, deep learning, and machine learning are interconnected no one of them, in particular, is going to replace the others. **Yann LeCun, the renowned French scientist and head of research at Facebook, jokes that reinforcement learning is the cherry on a great AI cake with machine learning the cake itself and deep learning the icing. Without the previous iterations, the cherry would top nothing.**In many use cases, using classical machine learning methods will suffice. Purely algorithmic methods not involving machine learning tend to be useful in business data processing or managing databases.  
Sometimes machine learning is only supporting a process being performed in another way, for example by seeking a way to optimize speed or efficiency. When a machine has to deal with unstructured and unsorted data, or with various types of data, neural networks can be very useful. How machine learning improved the quality of machine translation

**CONCLUSION**

Reinforcement Learning addresses the problem of learning control strategies for autonomous agents with least or no data. RL algorithms are powerful in machine learning as collecting and labeling a large set of sample patterns costs more than data itself. The key distinguishing factor of reinforcement learning is how the agent is trained. Instead of inspecting the data provided, the model interacts with the environment, seeking ways to maximize the reward. In the case of deep reinforcement learning, a neural network is in charge of storing the experiences and thus improves the way the task is performed.

**LINKS**

<https://en.wikipedia.org/wiki/Reinforcement_learning>

<https://searchenterpriseai.techtarget.com/definition/reinforcement-learning>

<http://www.scholarpedia.org/article/Reinforcement_learning>

<https://www.guru99.com/reinforcement-learning-tutorial.html#:~:text=Two%20types%20of%20reinforcement%20learning,given%20sample%20data%20or%20example>

<https://github.com/saumyaarora80/REINFORCEMENT-LEARNING> **GITHUB**