Probably not. Reinforcement learning was popularized in the past decade due to the rise of AlphaGo, a computer program developed by Google DeepMind that is able to compete with World Champions in the game of Go. The performance of AlphaGo is astonishing, and soon after the success of the program, many machine learning practitioners became obsessed with the underlying algorithm (Deep Reinforcement Learning), with many attempts to recreate the success of AlphaGo in other applications in various fields. Most of these attempts are like trying to shoehorn the model into a problem. Should you really be using Reinforcement Learning?

{% include figure.liquid loading="eager" path="assets/img/reinforcement_learning/alphago.jpg" class="img-fluid rounded z-depth-1" %}

I have personally developed several reinforcement learning models deployed for production in my career and will be sharing some of my thoughts here. The purpose of this article is not to serve as a technical primer to reinforcement learning; I would like to share some practical experience with Reinforcement learning.

How I think of RL

There are few ways to think of a reinforcement learning model. One can start off as a Markov Decision Process and frame it as a dynamic programming problem; this approach can help understand the underlying problem RL tries to solve, but it is not practical enough to see the limitations of RL in practice.

{% include figure.liquid loading="eager" path="assets/img/reinforcement_learning/reinforcement-learning-fig1-700.jpg" class="img-fluid rounded z-depth-1" %}

After some experience of developing the RL model, I have developed the following mindset: A (Deep) Reinforcement learning model consists of a $$Q$$ function (often a supervised learning model like Neural Network), with the difference being in its update rule:

$$
Q\left(s,a\right) = r\left(s,a\right) + \gamma \max_{a} Q\left(s^{'},a\right)
$$

I tend to think of RL as like a supervised learning model (but not really), with incremental (online) updates by using the update rule. This sounds like a direct improvement of supervised learning, but is it really? I say it is debatable, but let me list out some further caveats.

You are going to need a good simulation

This point here is enough to cripple a RL project. In order for a RL model to learn offline, you will need a lot of data. Data usually comes from simulation, as historical data are highly biased.

For example, one of the projects I was involved with was a RL application for collections treatment in a bank. The historical data we fed in consisted the
$$\left(s,a,r,s^{'}\right)$$
state, action, reward, next state tuple, but what would happen if we take an alternative action $$\alpha$$ that is different from $$a$$? Well, that is very hard to say unless we have the historical data for that same

customer but different action taken. The simulation environment created may not be very accurate to reflect the customer's future state $$s^{'}$$ based on action $$\alpha$$ which did not take place.

This is different from what most people have encountered during their RL adventure with OpenAI Gym library, it contains lots of preloaded simulation environments.

How to define reward function?

Defining a good reward function is not easy. Depending on how reward function is defined, RL agents can behave vastly different. Simulations can help with selecting reward function, but this puts heavy reliance once again on the accuracy of simulation system.

Conclusion

You most likely do not need reinforcement learning. RL is meant for a very specific type of problem. My recommendation is never to take a technology and look for a problem, as that may create unnecessary complications. There are also many deployment complications that are outside of scope from this article. You are probably better off using other machine learning techniques.