Generative Adversarial Networks (GANs)

The invention of GANs was connected to neural networks' vulnerability to adversarial examples. Researchers were going to run a speech synthesis contest, to see which neural network could generate the most realistic-sounding speech.

A neural network - the "discriminator" - would judge whether the speech was real or not.

In the end, they decided not to run the contest, because they realized people would generate speech to fool this particular network, rather than actually generating realistic speech.

These are the step to train GANs

- Randomly initialize weights of generator and discriminator networks
- Randomly initialize noise vector and generate image using generator
- Predict probability generated image is real using discriminator
- Compute losses both assuming the image was fake and assuming it was real
- Train the discriminator to output whether the image is fake
- Compute the penalty for the discriminator probability, without using it to train the discriminator
- Train the generator to generate images that the discriminator thinks are real
- Use the discriminator to calculate the probability that a real image is real
- Use L to train the discriminator to output 1 when it sees real images

Reinforcement Learning

In Reinforcement Learning, Agents interact with an Environment

They choose from a set of available Actions

The actions impact the Environment, which impacts agents via Rewards

Rewards are generally unknown and must be estimated by the agent

The process repeats dynamically, so agents learn how to estimate rewards over time

Advances in deep learning have led to many recent RL developments:

- In 2013, researchers from DeepMind developed a system to play Atari games
- In 2017, the AlphaGo system defeated the world champion in Go

In general, RL algorithms have been limited due to significant data and computational requirements.

As a result, many well-known use cases involve learning to play games. More recently, progress has been made in areas with more direct business applications.

Reinforcement Learning Architecture

The main components of reinforcement learning are: Policy, Agents, Actions, State, and Reward.

Solutions represents a Policy by which Agents choose Actions in response to the State

Agents typically maximize expected rewards over time

In Python, the most common library for RL is Open AI GYM

This differs from typical Machine Learning Problems:

Unlike labels, rewards are not known and are often highly uncertain

As actions impact the environment, the state changes, which changes the problem

Agents face a tradeoff between rewards in different periods

Examples of everyday applications of Reinforcement Learning include recommendation engines, marketing, and automated bidding.