Ethan Cruz
Projects in ML, RPI

# Discussion 7& 8

October 2022

**Paper 1 | Alias-free generative adversarial networks:**

In this NVIDIA-backed paper, the authors look to solve an issue with "texture sticking" in the portraits of their previous model, StyleGAN2. When creating moving faces with StyleGAN2, the textures of the face would remain stationary while the face moved.

In the world of CGI, this would be like covering a 3D model in a 2D texture. When the model is rotated or moved, the texture would remain stationary and it's position relative to the surface of the 3D model would change.

In the real world, of course, the surface of a face would move with the face! The result is StyleGAN2 producing very uncanny moving faces. Video Example (from NVIDIA's Github):
`https://nvlabs-fi-cdn.nvidia.com/_web/stylegan3/videos/video_0_ffhq_cinemagraphs.mp4#t=0.001`

The solution proposed was to add low-pass filtering after every convolution step. Low-pass filtering is an idea to prevent aliasing given by Nyquist-Shannon sampling.

Because the paper was about anti-aliasing, I believe the low-pass filter method could be used in GANs music generation. This is because the same ideas of aliasing exist in sound as they do in images, albeit manifesting in different ways.

In the case of music/sound, a GANs which attempts to improve the quality of an mp3 could end up generating massive high-end frequencies due to thinking "high quality = more transient noise". Forcing low-pass filters prevents this idea from sticking in the GANs.

**Paper 2 | Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning:**

The paper discusses limits in Reinforcement Learning using MDPs (Markov Decision Processes).

An MDP traverses through time on a fixed value, going from time $t$ to time $t + 1$ each step. Events at $t - 1$ usually don't matter to an MDP at $t + 1$. Longstanding consequences as a result of past decisions isn't felt as much as in reality.

The MDP also has a fixed set of options which remain available through each time step, which is unrealistic. Decisions like "go to lunch" only exist between certain times (say, 11 am to 2 pm) and the consequence of not choosing it during a certain time is felt more in reality than in an MDP.

The paper suggests mixing MDPs with SMDPs (Semi-Markov Decision Processes) where options are added and removed as time progresses. This abstracts the concept of time through the persistance of previous actions. Deciding NOT to eat lunch might remove some options when entering the SMDP due to a lack of energy from not working.

The idea of further abstraction of time and consiquences of time-based actions is useful in an innumerable amount of problems! The first one I thought of was the NYC Subway Challenge.

If someone painstakingly input the graph of every subway arrival and departure in NYC over a week, you could use a series of MDPs and SMDPs to map it. Of course, this set of graphs would exponentially increase in size as time goes on since previous options in the SMPD would impact the resulting MDPs, but you'd get an incredibly accurate environment for an agent to solve the subway challenge in!

**Paper 3 | Finite-time Analysis of the Multiarmed Bandit Problem:**

This paper has to do with a really cool problem in statistics, the Multiarmed Bandit Problem (specifically, the K-armed bandit version where the bandit has k number arms).

The problem goes you've got a bandit with k arms each on k slot machines. Each slot has a different probability distribution which we know nothing about. Through playing each of the slots we can learn the distributions, but there is a catch.

As we play a machine, we gain further understanding of that slot's probability distribution. We can get comfortable playing what we know as the best slot, or we can go play other machines to see if their distribution is better.

The paper describes this relation as exploration vs exploitation. Either explore other possibilities, or exploit the best one we have.

The authors have found a couple algorithms which balance this relation to the best of our ability in logarithmic time. They do this by measuring the bandit's regret, the reward cost of exploring new machines vs using the optimal machine.

One simple algorithm was UCB1:
1. Play all the machines once
2. For each machine i, record the average reward $x_i$ and the number of plays $n_i$
3. Play the machine j which has the biggest $x_i + \sqrt{\frac{2ln(t)}{n_i}}$ where t is the total number of plays you've made so far. 4. Repeat 2-3 forever

Using this awesome algorithm, and perhaps the other ones proposed, you can solve the k-armed bandit problem with the most efficiency theoretical.

What could this be used on IRL? A way to decide explore vs exploit is useful in almost every RL problem. A specific example might be an RL agent which is tasked with learning monopoly. Is it better to exploit the land I have now, or go purchase more land? Should I auction off my land or should I go and trade with other players to profit off of owning a whole color?