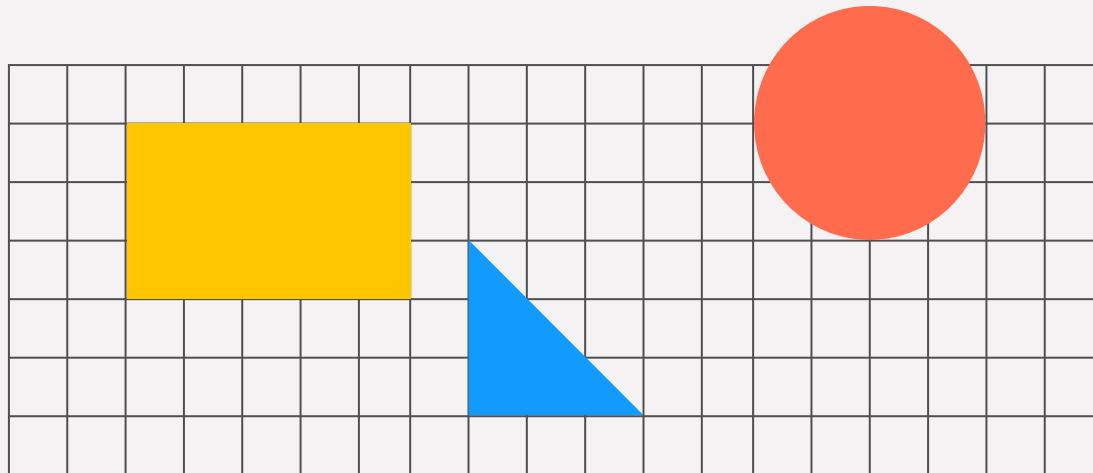
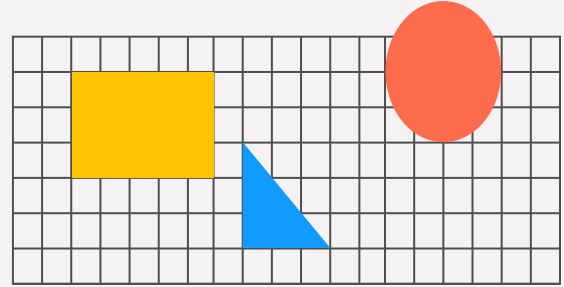


Games & Learning

Application of Game Theoretic Concepts
in Machine Learning



Game Theory



Game Theory: The study of mathematical models of strategic **interaction** among **rational** decision-makers. [Myerson, 1991]

Board Games and AI



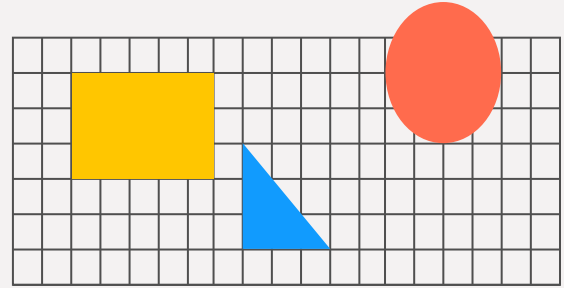
A Brief History of Chess AI

- 1951: Alan Turing published the first program on paper theoretically capable of playing chess.
- 1989: Chess world champion Gary Kasparov defeated IBM's Deep Thought in a chess match.
- 1997: IBM's Deep Blue becomes the first chess AI to defeat a grandmaster in a match.
- 2017: AlphaZero, a neural net-based digital automaton, beats Stockfish 28–0, with 72 draws in chess matches.
- 2019: Leela Chess Zero (LCZero v0.21.1-nT40.T8.610) defeats Stockfish 19050918 in a 100-game match 53.5 to 46.5 for the Top Chess Engine Championship season 15 title.
- Present: Modern chess AI engines deploy deep learning to learn from thousands of matches. They regularly have FIDE ratings, chess' rating system, above 3,400, far beyond the best human players.

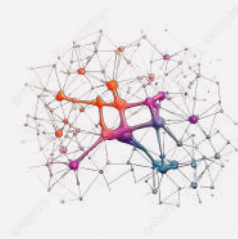
Ever Evolving GAN Networks

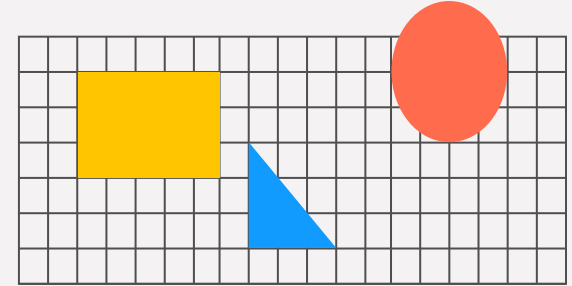
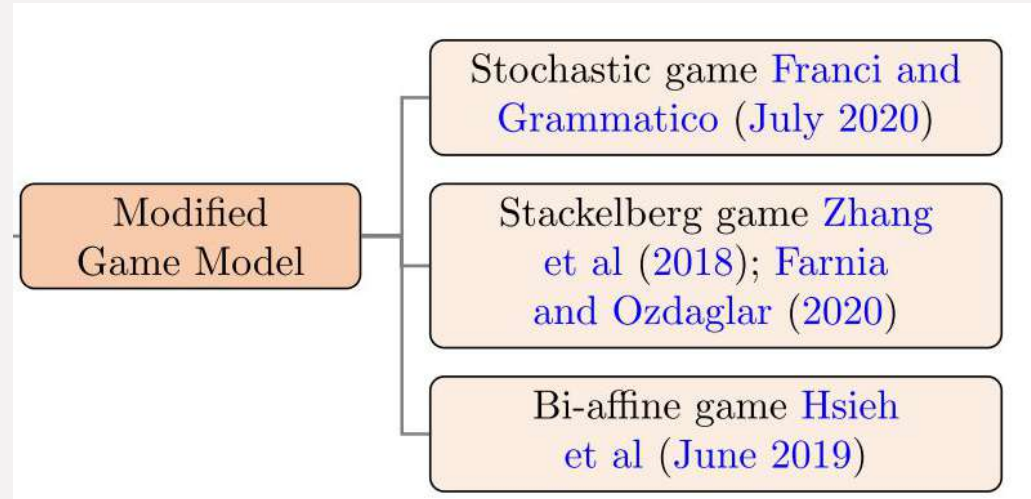


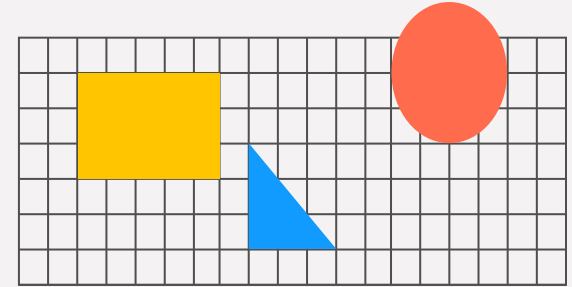
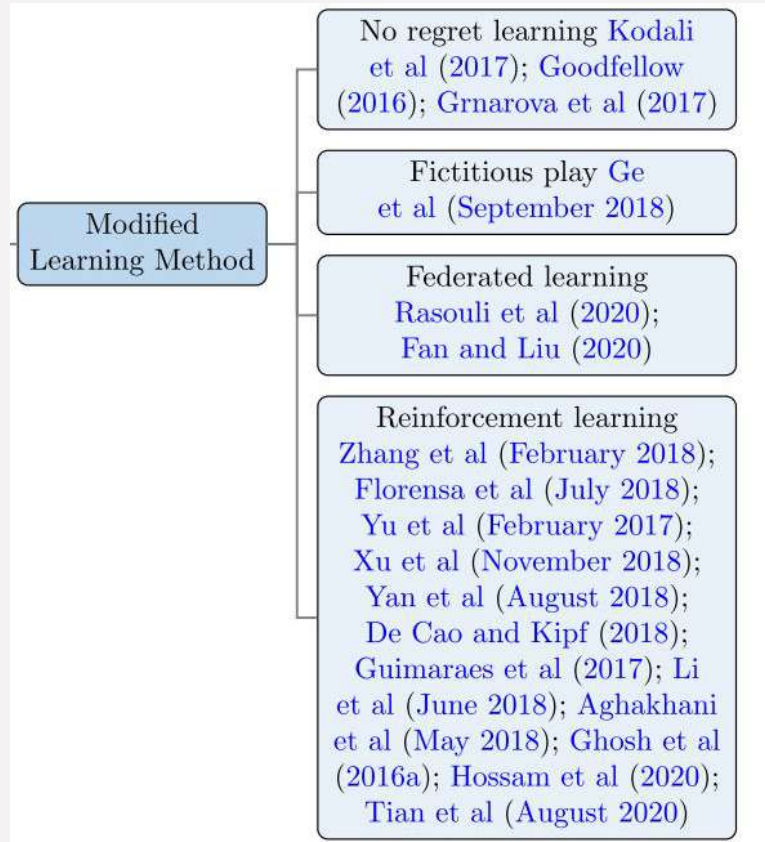
Why Game Theory



Problems with Data Bottlenecks
Huge Search Space
Adversarial Robustness
Decomposition into Players
Online Update Algorithms
Incremental Learning
Modelling Human Behaviour







Game Theoretic Improvements in GAN

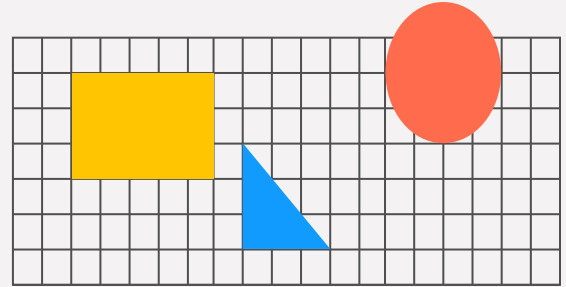
Multiple generators, One discriminator Zhang et al (2018); Ghosh et al (June 2018); Hoang et al (February 2018); Ke and Liu (October 2020); Ghosh et al (2016b)

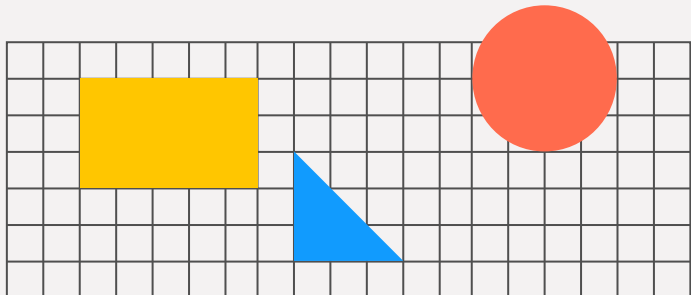
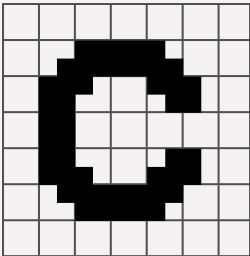
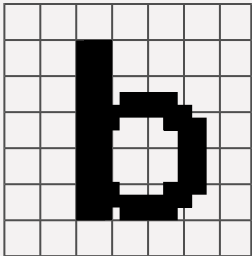
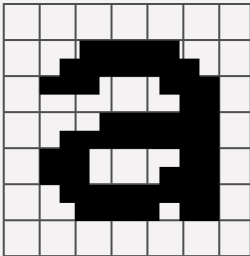
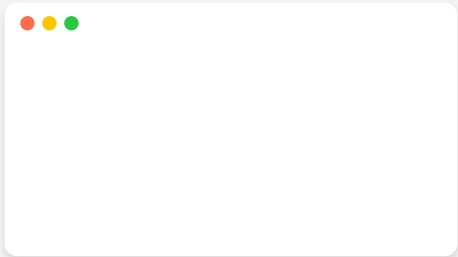
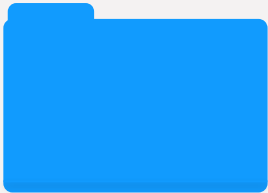
One generator, Multiple discriminators Durugkar et al (2016); Jin et al (July 2020); Hardy et al (May 2019); Aghakhani et al (May 2018); Mordido et al (March 2020); Nguyen et al (December 2017)

Multiple generators, Multiple discriminators Arora et al (2017); Rasouli et al (2020); Ke and Liu (October 2020)

One generator, One discriminator, One classifier Tran et al (2019); Li et al (June 2018)

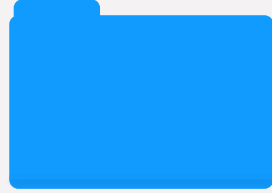
One generator, One discriminator, One RL agent De Cao and Kipf (2018); Sarmad et al (June 2019); Liang and Tang (January 2020); Guimaraes et al (2017)







Classic ML



GANs



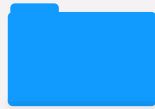
Optimization



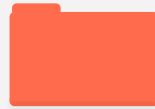
RL Agents



Classic
ML



GANs

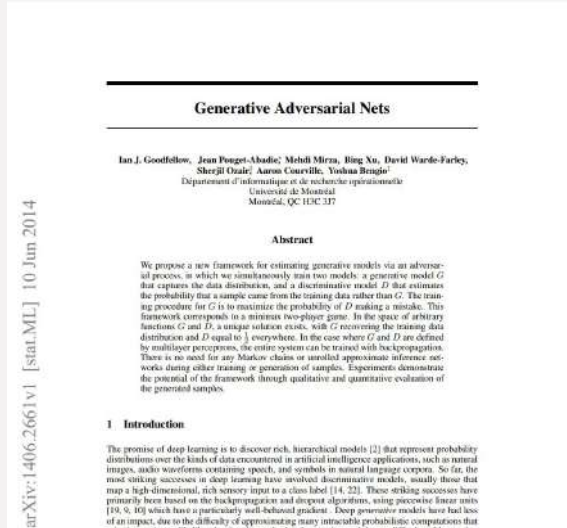


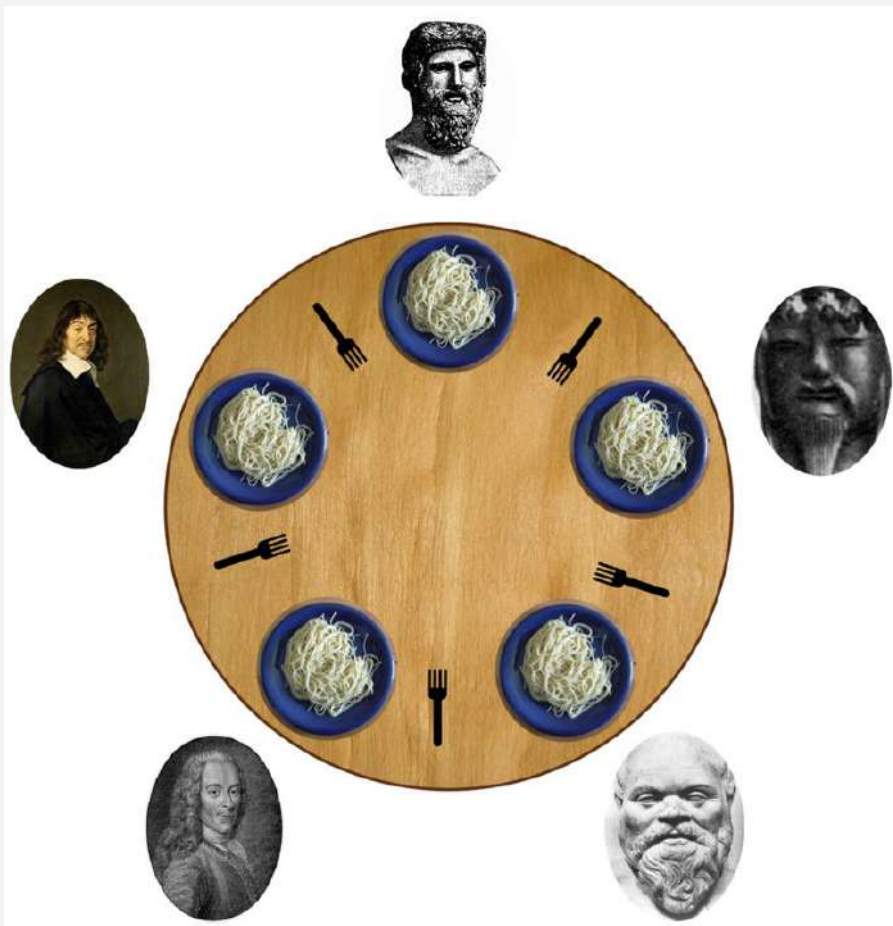
Optimization



RL
Agents

Credits : (GAN paper), (NeurIPS tutorial on learning dynamics by Marta Garnelo, Wojciech Czarnecki and David Balduzzi,), Book - Tanmay Hazra



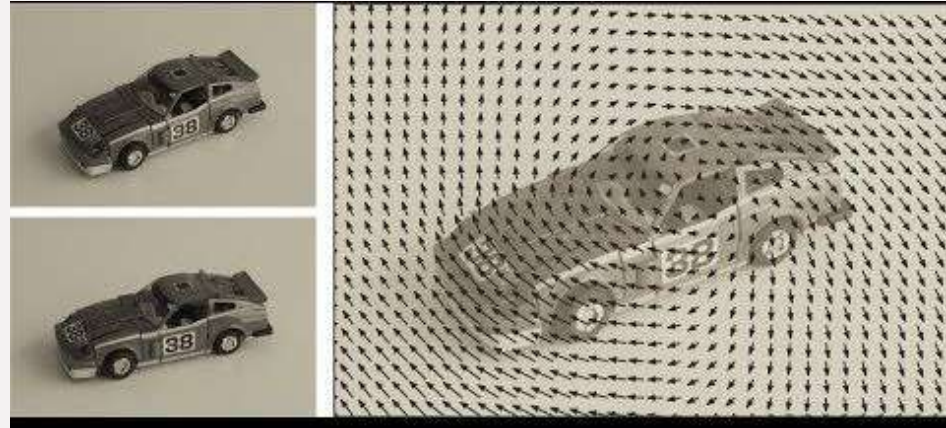
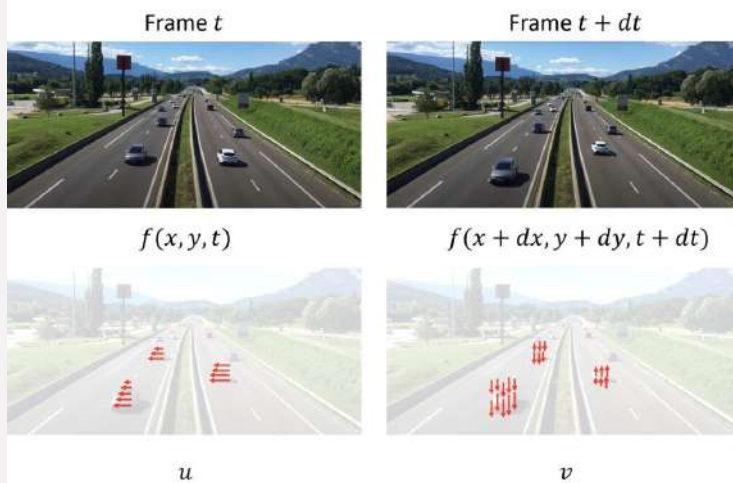
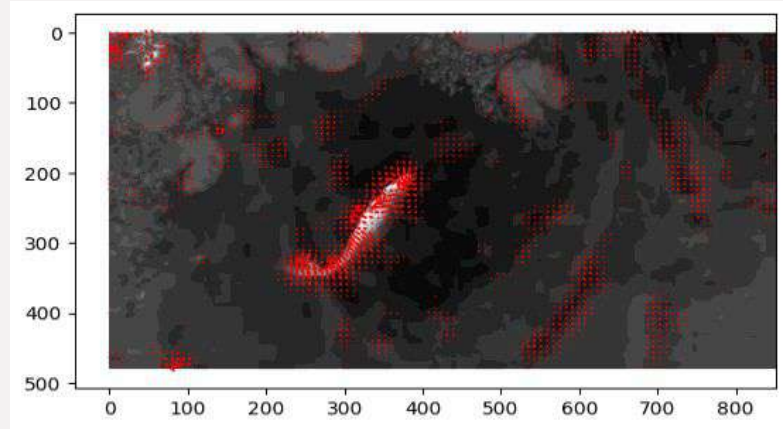
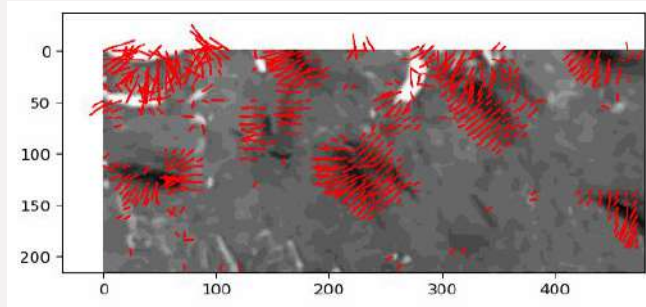


Dining Philosopher Problem

Is it a game of multiple agents ?

P_1, \dots, P_n players
Strategies - {eat, think}
Solution === Nash Equilibria

Horn Schunk Method of Optical Flow

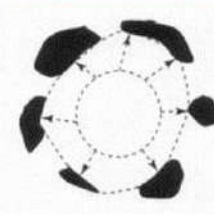
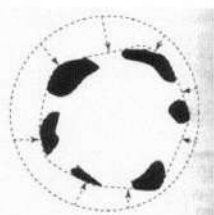
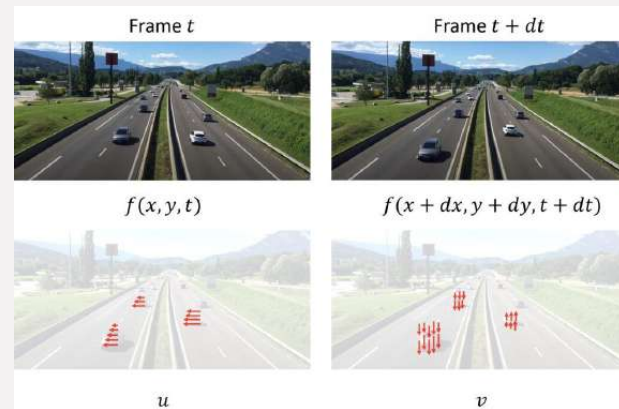


Horn Schunk Method of Optical Flow

$$E = E_s + E_b$$

$$E = \iint [(I_x u + I_y v + I_t)^2 + \alpha^2 (\|\nabla u\|^2 + \|\nabla v\|^2)] dx dy$$

Two Players



Deformable Contour

The total energy (cost) of the current snake is defined as:

$$E_{total} = E_{internal} + E_{external}$$

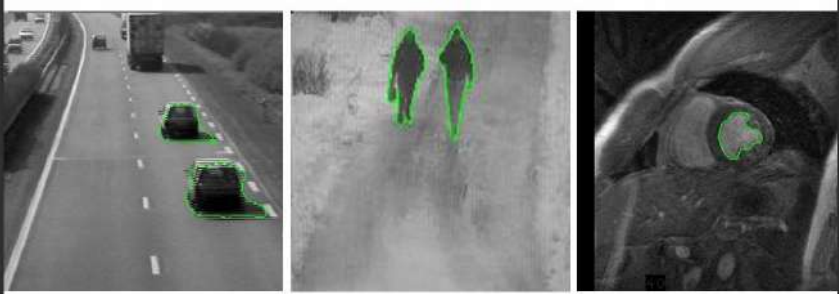
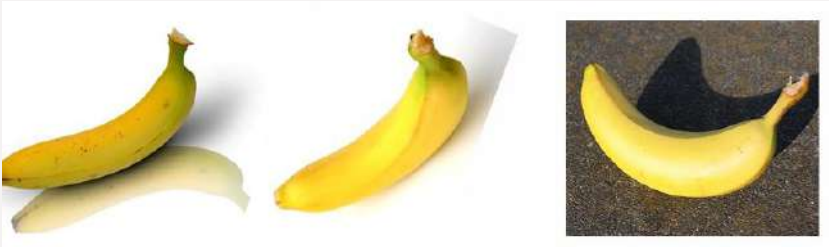


Internal energy: encourage *prior* shape preferences: e.g., smoothness, elasticity, particular known shape.

External energy ("image" energy): encourage contour to fit on places where image structures exist, e.g., edges.

A good fit between the current deformable contour

Deformable Contour




The total energy (cost) of the current snake is defined as:

$$E_{total} = E_{internal} + E_{external}$$



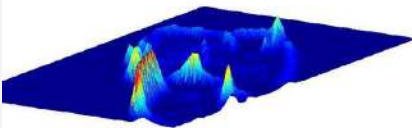
- Internal** energy: encourage *prior* shape preferences: e.g., smoothness, elasticity, particular known shape.
- External** energy (“image” energy): encourage contour to fit on places where image structures exist, e.g., edges.

A good fit between the current deformable contour



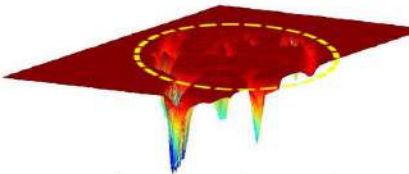
How do edges affect “snap” of rubber band?

Think of external energy from image as gravitational pull towards areas of high contrast



Magnitude of gradient

$$G_x(I)^2 + G_y(I)^2$$



-(Magnitude of gradient)

$$-(G_x(I)^2 + G_y(I)^2)$$

Mean Field Games

MFG comprises methods and techniques to study differential games with a large population of rational players. These agents have preferences not only about their state (e.g., wealth, capital) but also on the distribution of the remaining individuals in the population

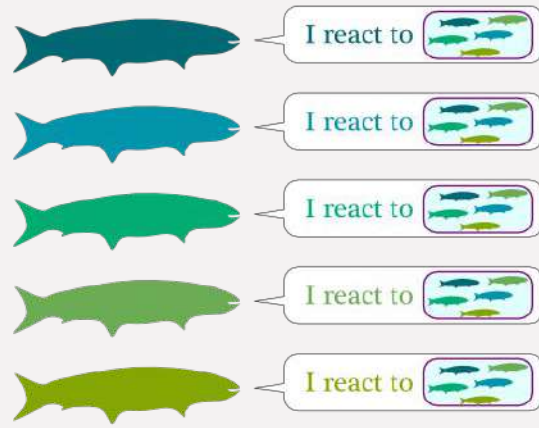
Mean Field Games

MFG comprises methods and techniques to study differential games with a large population of rational players. These agents have preferences not only about their state (e.g., wealth, capital) but also on the distribution of the remaining individuals in the population

$$\frac{\partial u(t, x)}{\partial t} + H(x, \nabla u(t, x), m(t, x)) = 0,$$

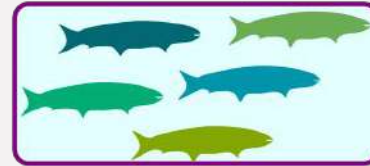
$$\frac{\partial m(t, x)}{\partial t} - \nabla \cdot (m(t, x) \nabla_p H(x, \nabla u(t, x), m(t, x))) = 0,$$

Mean Field Games



Hamilton-Jacobi-Bellman

I, the mass,
move accordingly
to what fishes do.



Fokker-Planck-Kolmogorov

K Means Clustering



A Mini-max Game

[Swagatam Das]

K Means Problem

Given a set of observations $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$, where each observation is a d -dimensional real vector, k -means clustering aims to partition the n observations into k ($\leq n$) sets $\mathbf{S} = \{S_1, S_2, \dots, S_k\}$ so as to minimize the within-cluster sum of squares (WCSS) (i.e. [variance](#)). Formally, the objective is to find:

$$\arg \min_{\mathbf{S}} \sum_{i=1}^k \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg \min_{\mathbf{S}} \sum_{i=1}^k |S_i| \text{Var } S_i$$

where $\boldsymbol{\mu}_i$ is the mean (also called centroid) of points in S_i , i.e.

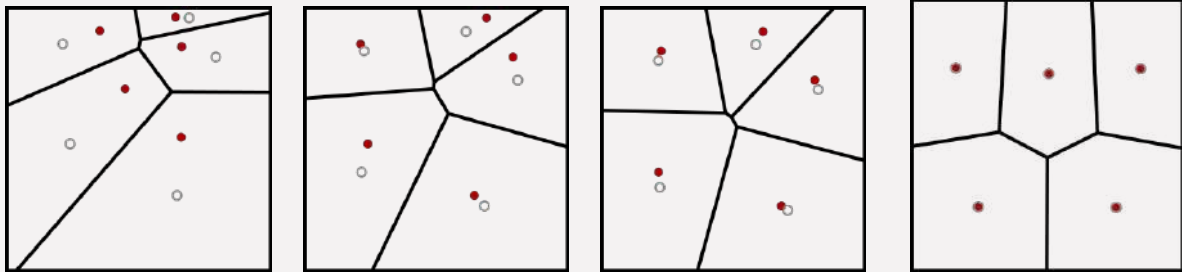
$$\boldsymbol{\mu}_i = \frac{1}{|S_i|} \sum_{\mathbf{x} \in S_i} \mathbf{x},$$

$|S_i|$ is the size of S_i , and $\|\cdot\|$ is the usual [L² norm](#). This is equivalent to minimizing the pairwise squared deviations of points in the same cluster:

$$\arg \min_{\mathbf{S}} \sum_{i=1}^k \frac{1}{|S_i|} \sum_{\mathbf{x}, \mathbf{y} \in S_i} \|\mathbf{x} - \mathbf{y}\|^2$$

The equivalence can be deduced from identity $|S_i| \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \frac{1}{2} \sum_{\mathbf{x}, \mathbf{y} \in S_i} \|\mathbf{x} - \mathbf{y}\|^2$. Since the total variance is constant, this is equivalent to maximizing the sum of squared deviations between points in *different* clusters (between-cluster sum of squares, BCSS).^[1] This deterministic relationship is also related to the [law of total variance](#) in probability theory.

Lloyd's Algorithm



The algorithm was first proposed by Stuart P. Lloyd of [Bell Labs](#) in 1957 as a technique for [pulse-code modulation](#). Lloyd's work became widely circulated but remained unpublished until 1982. Lloyd-Max algorithm.

Lloyd's algorithm starts by an initial placement of some number k of point sites in the input domain. In mesh-smoothing applications, these would be the vertices of the mesh to be smoothed; in other applications they may be placed at random or by intersecting a uniform triangular mesh of the appropriate size with the input domain.

It then repeatedly executes the following relaxation step:

- The [Voronoi diagram](#) of the k sites is computed.
- Each cell of the Voronoi diagram is integrated, and the centroid is computed.
- Each site is then moved to the centroid of its Voronoi cell.

K means Clustering

$$J = \sum_{j=1}^k \sum_{i=1}^m a_{ij} \|x_i - \mu_j\|_2^2$$

Given an initial set of k means $m_1^{(1)}, \dots, m_k^{(1)}$ (see below), the algorithm proceeds by alternating between two steps:^[7]

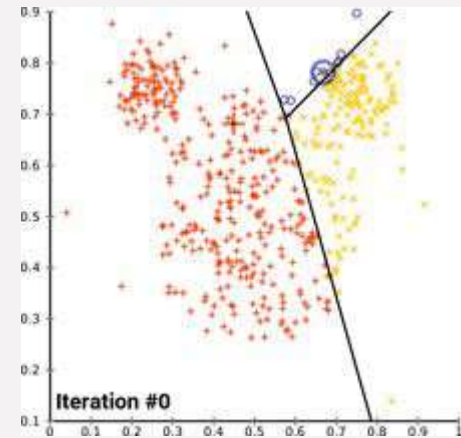
1. **Assignment step:** Assign each observation to the cluster with the nearest mean: that with the least squared [Euclidean distance](#).^[8] (Mathematically, this means partitioning the observations according to the [Voronoi diagram](#) generated by the means.)

$$S_i^{(t)} = \left\{ x_p : \|x_p - m_i^{(t)}\|^2 \leq \|x_p - m_j^{(t)}\|^2 \quad \forall j, 1 \leq j \leq k \right\},$$

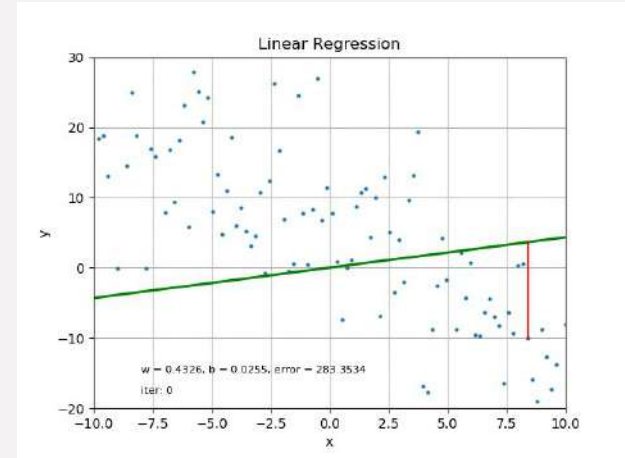
where each x_p is assigned to exactly one $S^{(t)}$, even if it could be assigned to two or more of them.

2. **Update step:** Recalculate means ([centroids](#)) for observations assigned to each cluster.

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$



Linear Regression as a Non-Cooperative Game



Linear Regression as a Non-Cooperative Game

Linear Regression as a Non-Cooperative Game

4 Nash Equilibria

Elements of the Game

- **Players** (N): Each individual $i \in N$
- **Strategies** (λ_i): Each player chooses a strategy (or, inversely, the precision) of the model (higher accuracy but higher privacy cost)
- **Payoff or Cost Function** ($J_i(\lambda_i, \lambda_{-i})$), which is composed of:
 - $c_i(\lambda_i)$: The **privacy cost** incurred by player i in λ_i .
 - $f(\lambda)$: The **estimation cost**, where $\lambda = (\lambda_1, \dots, \lambda_n)$ and is non-decreasing in λ (lower noise improves the model).

We begin our analysis by characterizing the Nash equilibria of the game Γ . Observe first that Γ is a potential game [24]. Indeed, define the function $\Phi : [0, 1/\sigma^2]^n \rightarrow \mathbb{R}$ such that

$$\Phi(\lambda) = f(\lambda) + \sum_{i \in N} c_i(\lambda_i), \quad (\lambda \in [0, 1/\sigma^2]^n). \quad (5)$$

Keywords: Linear regression, Gauss-Markov theorem, Aitken theorem, privacy, potential game, price of stability

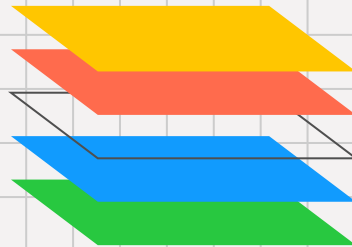
1 Introduction

The statistical analysis of personal data is a cornerstone of several experimental sciences, such as medicine and sociology. Studies in these areas typically rely on experiments, drug trials, or surveys involving human subjects. Data collection has also become recently a commonplace—yet controversial—aspect of the In-

Learning Algorithms



In frameworks like no-regret learning, agents iteratively update strategies to minimize average regret



Game-theoretic concepts, such as Nash Equilibria, enable the study of stability in these interactions:
 $J_i(s_i^*, s_{-i}^*) \leq J_i(s_i, s_{-i}^*) \quad \forall s_i$ providing insight into equilibrium dynamics.

Analogy

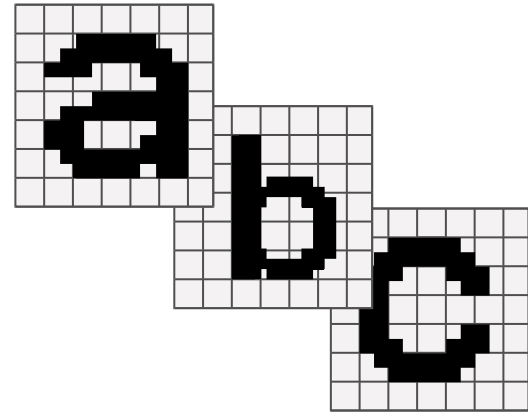


Machine learning (ML) and game theory intersect in settings where agents (or models) make decisions that influence one another, often optimizing conflicting objectives. Let $N = \{1, 2, \dots, n\}$ represent agents (or learners), each choosing a strategy $s_i \in S_i$ to minimize a loss $J_i(s_i, s_{-i})$, where s_{-i} denotes the strategies of other agents. Common frameworks include **adversarial learning**, such as GANs, where a generator and discriminator solve a zero-sum game: $\min_G \max_D J(G, D)$. Similarly, **multi-agent reinforcement learning** models agents optimizing reward functions influenced by others, while **federated learning** employs game-theoretic mechanisms to balance privacy ($c_i(s_i)$) and utility ($f(s)$). The Nash equilibrium s^* ensures stability, where no agent benefits by unilateral deviation: $J_i(s_i^*, s_{-i}^*) \leq J_i(s_i, s_{-i}^*) \quad \forall s_i$. These connections make game theory a powerful tool for understanding interaction and optimization in ML.

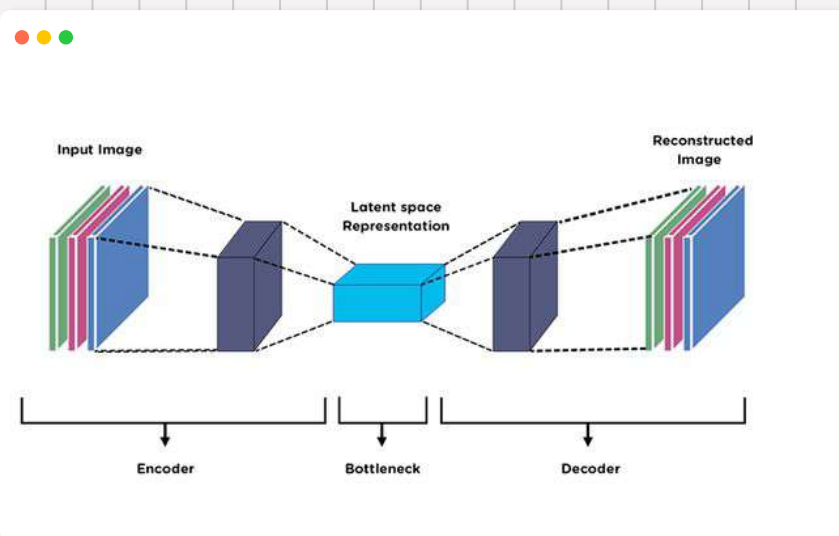
Generative Adversarial Networks

Outline

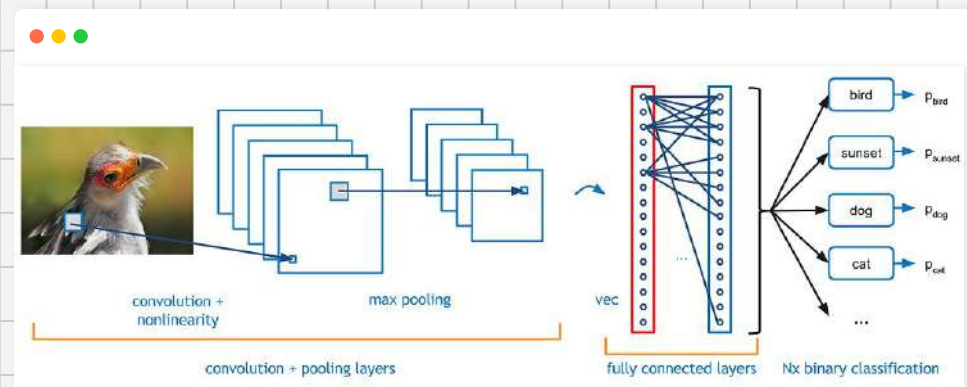
a machine learning framework where a generator and a discriminator compete in a zero-sum game to generate data indistinguishable from real data.



Two Players

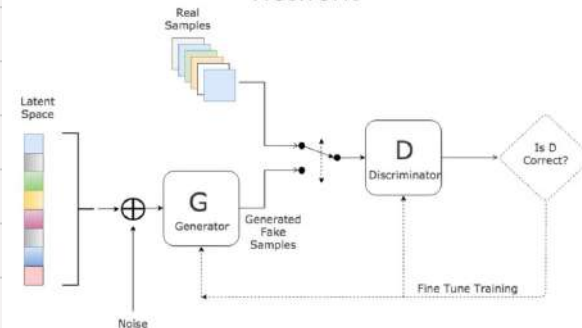


Generator



Discriminator

Generative Adversarial Network



Combined Architecture

The Players

Gen

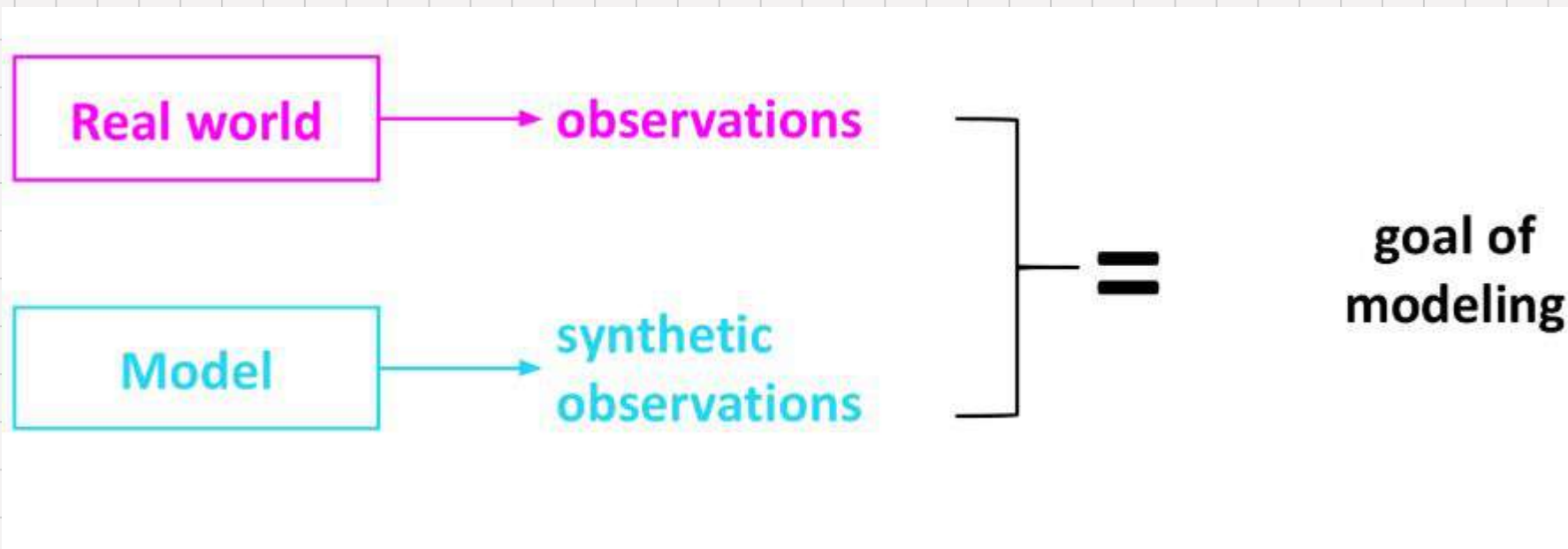
- Objective: The Generator aims to produce synthetic data $G(z)$ that closely resembles real data x . Its utility is maximized when the Discriminator fails to distinguish between real and fake data.
- Strategy: The Generator learns a mapping from latent space (z) to data space $G(z)$ by optimizing its parameters to minimize KL
- Payoff Function: The Generator minimizes the Discriminator's penalty

Adv

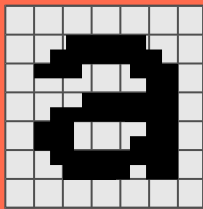
- Objective: The Discriminator's goal is to correctly classify data as real x or fake $G(z)$. It maximizes its utility when it perfectly distinguishes real data from generated data.
- Strategy: The Discriminator adjusts its decision boundary to maximize F1
- Payoff

$$\max_D V(G, D) = \mathbb{E}_{x \sim p_{data}} [\log(D(x))] + \mathbb{E}_{z \sim p_z} [\log(1 - D(G(z)))]$$

Probabilistic Generative Models

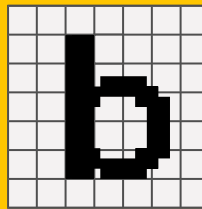


Images - Imagine them to be Sampled from some Complex Distribution



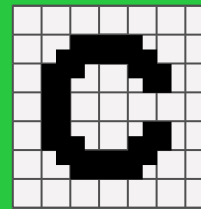
Sample

GANs assume the real images come from an unknown probability distribution p_{data}



Reconstruct

latent representations and reconstructed, capturing the underlying structure



Maximize

training between the generator and discriminator to align the generated data distribution p_G with p_{data}

GAN Obj - Two Player Game

Generator network: try to fool the discriminator by generating real-looking images

Discriminator network: try to distinguish between real and fake images

Train jointly in **minimax game**

Discriminator outputs likelihood in $(0,1)$ of real image

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on

generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

GAN Obj - Two Player Game

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Gradient signal

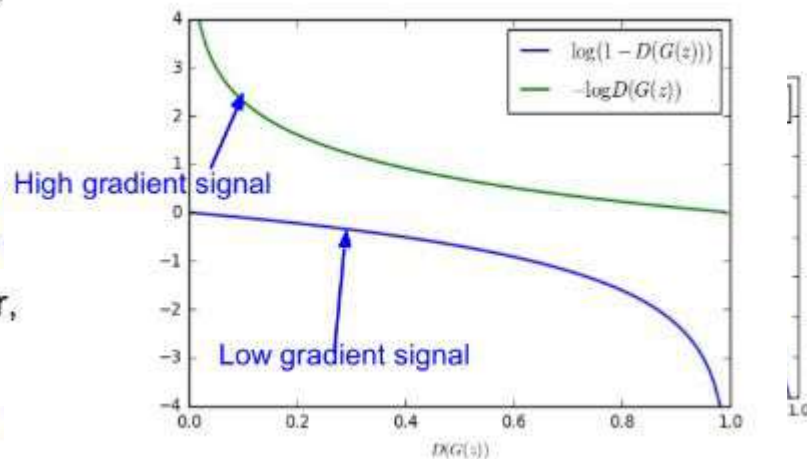
2. **Gradient descent** on

generator $\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$

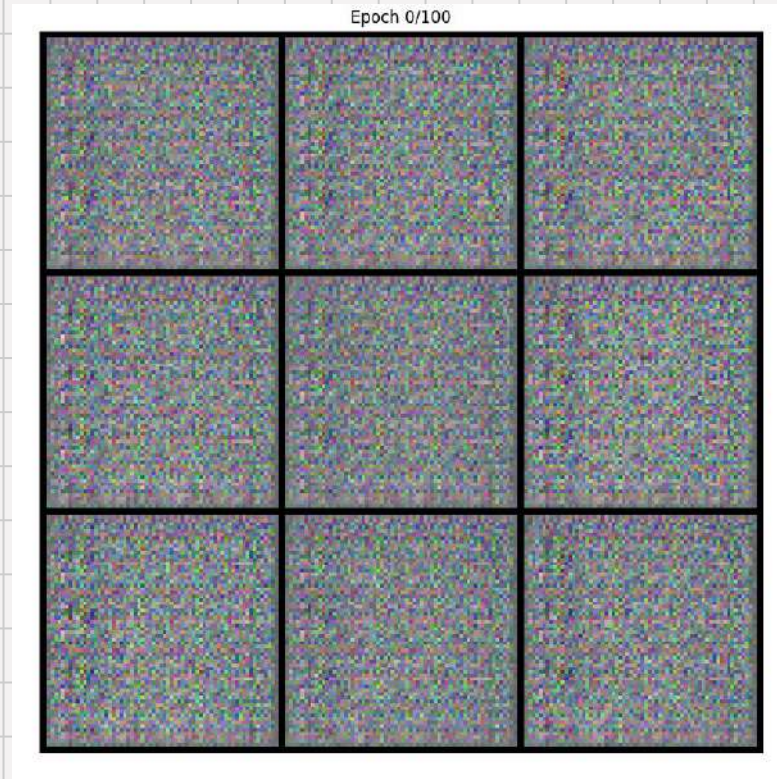
In practice, optimizing this generator objective does not work well!

2. **Instead: Gradient ascent** on generator, different objective

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$



GAN - In Action



Reinforcement Learning

Game-theoretic concepts like best response, zero-sum games, and correlated equilibria underpin many MARL algorithms,

Reinforcement Learning (RL) and Game Theory are related through their shared focus on decision-making and strategic interactions in environments involving multiple agents. In single-agent RL, an agent optimizes its policy to maximize cumulative rewards in a dynamic environment, analogous to solving a game against nature. In multi-agent RL (MARL), multiple agents interact, each optimizing their policies based on individual rewards, similar to game-theoretic scenarios where agents aim to achieve equilibrium solutions, such as Nash Equilibrium.

RL techniques, such as policy gradients and Q-learning, are applied to solve game-theoretic models like Markov games and stochastic games.

Tips

Cooperative Evolution

Date

Game Theory

AI

Monday

May
1944

John von Neumann and Oskar Morgenstern publish "*Theory of Games and Economic Behavior*", establishing the mathematical framework of game theory.

- 1956
- Dartmouth Conference

Monday

Jan
1950

John Nash introduces the concept of Nash Equilibrium,

- 1972 - Evolutionary Game Theory
- John Maynard Smith introduces evolutionary game theory

Monday

Feb
1960

Minimax search algorithms are applied to games like chess and checkers, leading to early AI systems like IBM's *Mac Hack*

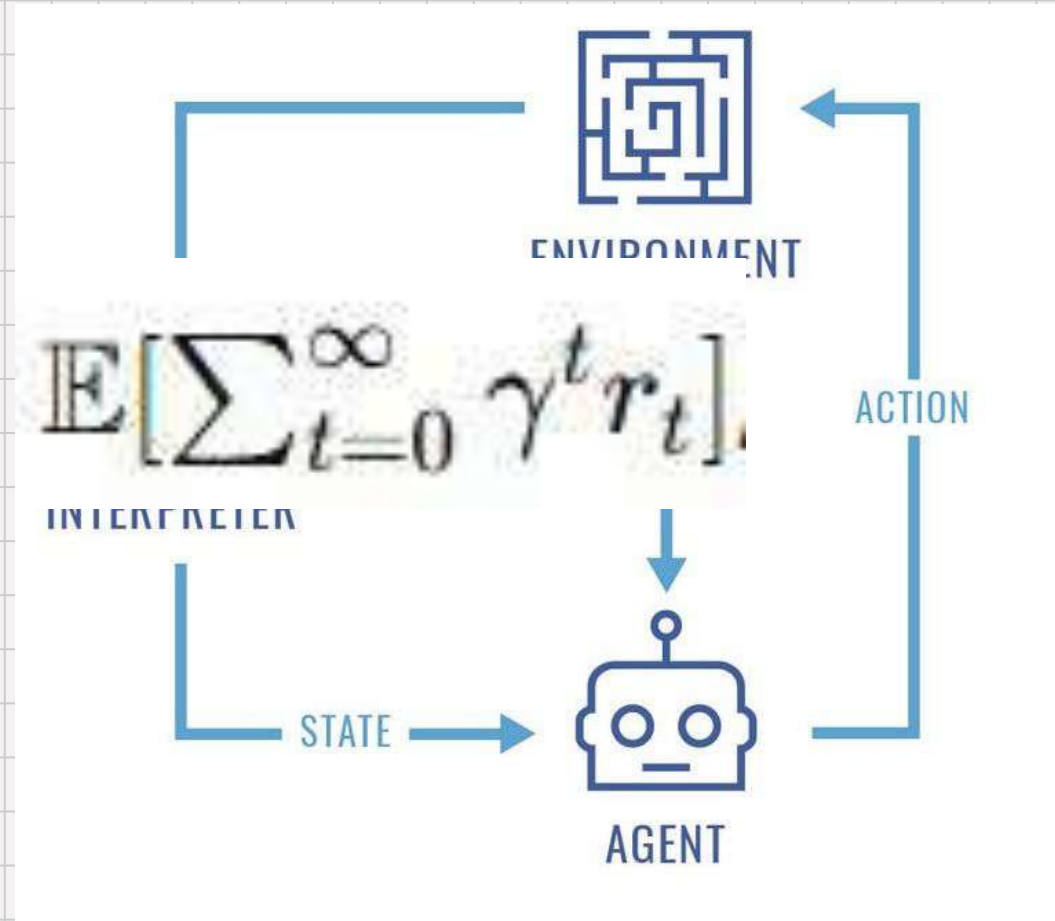
- 1997 - Game AI Breakthrough
- IBM's Deep Blue defeats world chess champion Garry Kasparov

Monday

Sep
1987

Multi-Agent Systems in AI - Research in distributed AI begins focusing on multi-agent systems, leveraging game theory for cooperative and competitive behaviors

- 2016 - AlphaGo
- Google DeepMind's AlphaGo Monte Carlo tree search (MCTS)



Reconnaissance Chess



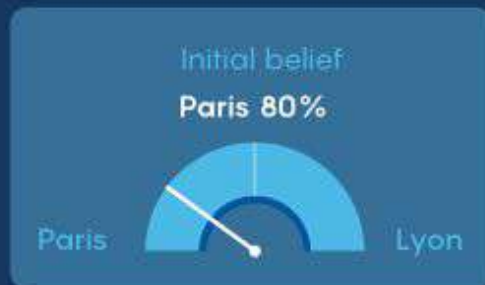
Given current history: sense e3, observe Pe4, move c7c6

Building Consensus

A large language model includes two systems: the generator and the discriminator. By using a game to align those systems, the LLM can become more accurate.

Question: What is the capital of France?

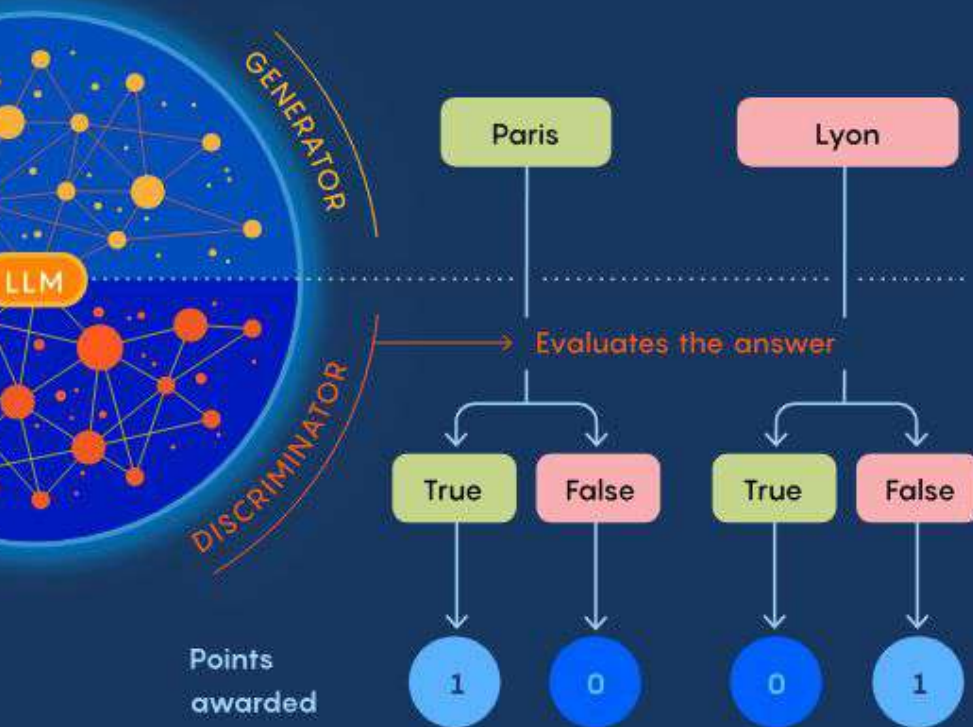
GENERATOR



DISCRIMINATOR



The generator thinks the likely answer is "Paris," but flips a coin to determine if it will answer truthfully or untruthfully. The discriminator then decides if the generator gave a true answer or not.



If the two parts agree, they receive points. To avoid converging on the wrong answer — Lyon — they lose points if they deviate too much from their initial beliefs. They play the game 1,000 times per question. Consensus emerges over time, and the model improves.

Thanks

Game theory provides the mathematical foundation to analyze interactions between rational agents, while AI focuses on building intelligent systems to act and learn in complex environments. This synergy makes them two sides of the same coin, two perspective of intelligence.

Email

SandeepChatterjee66@gmail.com

Affiliation

Student, MTech CS 2318

Github

<https://github.com/SandeepChatterjee66/GAN-Game>