6.883 Science of Deep Learning – Spring 2018 May 11, 2018

Lecture 8: Game-theoretic View on GAN Dynamics

Lecturer: Constantinos Daskalakis *Scribes: Zhe Feng, William Peebles, Tianxiao Shen*
(Revised by Andrew Ilyas and Manolis Zampetakis)

1 Recap: Generative Adversarial Networks

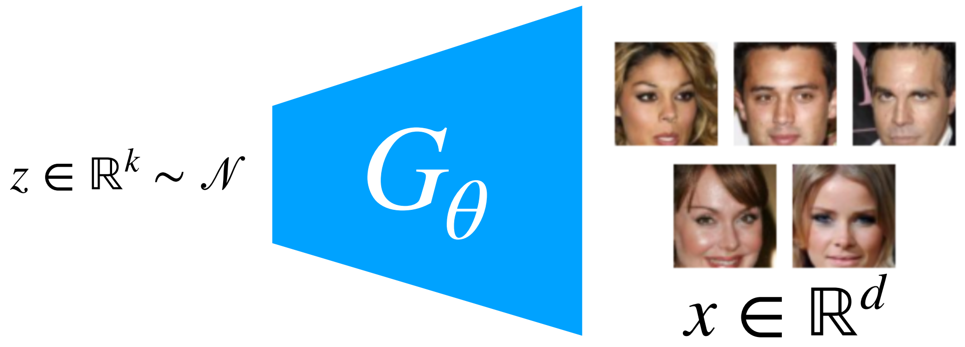

Figure 1: An example of a GAN mapping a low-dimensional random Gaussian vector to high-dimensional pictures of faces.

Generative adversarial networks (GANs) [7] map white noise to high-dimensional objects with structure (e.g. images of faces, as in Figure 1 above) by setting up a game between a generator (usually a deep neural network, with parameters θ_g) and a discriminator (also usually a deep neural network, with parameters θ_d). In the case of the Wasserstein GAN [2] (WGAN), which was the focus of our introduction to GANs last lecture, this game is inspired by minimization of the Wasserstein distance:

$$\inf_{\theta_g} \sup_{\theta_d} \mathbb{E}_{x\sim F}[D_{\theta_d}(x)] - \mathbb{E}_{z\sim N(0,I)}[D_{\theta_d}(G_{\theta_g}(z))] \tag{1}$$

The generator and discriminator are traditionally trained by running gradient descent and ascent respectively, on the loss function above. The expectations are estimated via finite-sample averages across batches of real or fake images (see Figure 2 for the training mechanism).

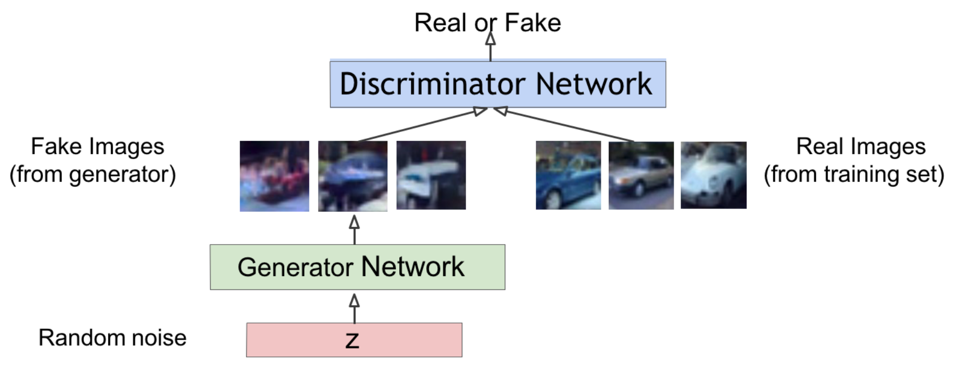

Figure 2: A schematic demonstrating the training procedure for GANs.

Even assuming we can estimate the expectations given in (1) with finite-sample means, will paired gradient descent/ascent dynamics converge? If so, to what? In this lecture we'll investigate these problems from the perspective of game theory and online convex optimization.

2 A Brief Introduction to Game Theory

Definitions. We begin with some basic game-theoretic definitions. A *two-player game* consists of two agents who, given an environment, play an action from a permissible *action space*. Given the actions from each player, the environment then distributes an appropriate *reward*. The agents' method for choosing an action from the action space is known as a *strategy*. Strategies can be deterministic (*pure* strategies), or they can be selected by sampling a probability distribution over multiple actions (*mixed* strategies). After each player takes an action according to their strategies, they each receive a *payoff*. Payoffs are specified by a *payoff matrix*, where rows correspond to actions taken by the first player and columns corresponds to actions taken by the second player. Each entry in the matrix specifies the payoff for both players.

Finally, a *Nash equilibrium* is the notion of an "optimal solution" to a game. In two-player games, Nash equilibria have a quite intuitive definition: the two players are in a Nash equilibrium if each player cannot improve their strategy given that they know the other player's strategy.

To illustrate the concepts we just introduced, we walk two classic game-theoretic examples of two-player games.

Movie night. Consider a scenario in which two friends are deciding between seeing *Avengers: Infinity War* and *Black Panther*. Friend 1 would prefer to see *Infinity War* and Friend 2 would rather see *Black Panther*. However, they want to attend the same movie together.

Let action IW correspond to seeing *Infinity War* and action BP correspond to *Black Panther*. Table 1 shows an example of a payoff matrix for this scenario.

		Friend 2	
		IW	BP
Friend 1	IW	$(7,4)$	$(0,0)$
	BP	$(0,0)$	$(4,7)$

Table 1: A payoff matrix for the movie-picking scenario described. We note the presence of three Nash equilibria: the top-left cell, the bottom-right cell, and the mixture of cells given by each Friend deploying a mixed strategy.

There are three Nash equilibria associated with this payoff matrix. First, observe that if both players use pure strategies where they end up playing the same action, then they each would be worse-off if they changed their strategy. Thus, the top-left and bottom-right entries of the payoff matrix correspond to Nash equilibria. There also exists a Nash equilibrium of this game that involves mixed strategies. Suppose Friend 1 used a mixed strategy, picking IW with probability p and BP with probability $1-p$. Friend 2's expected payoff when playing IW is then $4p$. And his expected payoff when playing BP is $7(1-p)$. The expected payouts for playing IW and BP must be identical since, if they were not, Friend 2 would have a pure strategy where he always played the action with higher expected payoff (this, in turn, would cause Friend 1 to move back to a pure strategy) [11]. Thus, we see $4p = 7(1-p) \rightarrow p = \frac{7}{11}$. We can repeat this analysis to show that the probability Friend 2 picks IW is $\frac{4}{11}$ (and thus he picks BP with probability $\frac{7}{11}$). The mixed strategy Nash equilibrium thus corresponds to the case when Friend 1 plays IW with probability $\frac{7}{11}$, plays BP with probability $\frac{4}{11}$, and Friend 2 plays IW with probability $\frac{4}{11}$ and BP with probability $\frac{7}{11}$.

2.1 The Min-Max Theorem

This brings us to one of the foundational results of game theory, namely the min-max theorem:

Theorem 1 *[10] If $X \subset \mathbb{R}^n, Y \subset \mathbb{R}^m$ are compact and convex, and $f : X \times Y \to \mathbb{R}$ is convex-concave (i.e. $f(x, y)$ is convex in x for all y and is concave in y for all x), then*

$$\min_{x\in X} \max_{y\in Y} f(x, y) = \max_{y\in Y} \min_{x\in X} f(x, y). \tag{2}$$

For illustration, Figure 3 shows a visualization of a convex-concave function $f(x, y)$.

A solution to (2) corresponds to a Nash equilibrium. To see why, consider the min player. The equality in (2) tells us that the payoff received as a result of the min player's best move does not change depending if the min player goes first (LHS) or second (RHS). This means that–even if the min player knows the strategy of the max player–the min player cannot change its strategy to obtain an improved payoff. By symmetry, the same must be true of the max player. Thus, we see that solutions to (2) meet the definition of a Nash equilibrium.

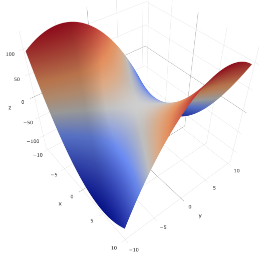

Figure 3: $f(x, y) = x^2 - y^2 + xy$, a convex-concave function

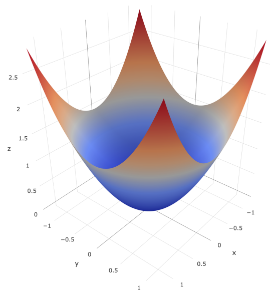

Figure 4: $f(x, y) = x^2 + y^2$, a *non*-convex concave function.

Also note that in the case of convex-concave functions, the min-max optimal point is essentially unique (if f is strictly convex-concave, otherwise there exists a convex set of solutions that all attain the same value). However, if f is not convex-concave (which is the case for GAN objective), then there can be multiple saddle points with different values, or no solution at all.

The min-max theorem is often considered as the starting point of game theory, since the min-max points of f are precisely the Nash equilibria of a zero-sum game with payoff $f(x, y)$ and $-f(x, y)$. John von Neumann, to whom the theorem is credited, said: "As far as I can see, there could be no theory of games ... without that theorem ... I thought there was nothing worth publishing until the Minimax Theorem was proved".

An interesting connection. When f is a bilinear function (linear in both x and y), for example $f(x, y) = x^T Ay + b^T x + c^T y$, min-max theorem is equivalent to strong LP duality [3, 1]. Thus, min-max solutions can be found using linear programming—this connection is arguably a crucial component in the recent success of computers in beating humans at two-player zero-sum games (chess, poker, Go, etc).

3 Repeated Games and Online Convex Optimization

In this section, we extend the basic definitions and intuitions covered in the last section to the setting of *repeated* (or *online*) games. In the simplest such setting, we consider a repeated zero-sum game between a player and "nature" (in other words, we can imagine a single player performing the minimization in $\min_x \max_y f(x, y)$). The online/repeated nature of the game being played means that rather than simply maximizing payout at any one time, the player's goal is maximize their *cumulative* payoff from playing the game repeatedly.

Concretely, we can describe the online game setup as follows:

- For every timestep $t = 1, 2, \cdots, T$
  - The learner/player chooses $x_t \in X \subset \mathbb{R}^n$

  - Nature chooses a convex function $f_t(\cdot)$
  - The learner incurs loss $f_t(x_t)$ and observes function $f_t(\cdot)$

- Learner's goal is to minimize the average regret, defined as the average loss (negative payoff) minus the loss of the best *fixed* action in hindsight:

$$R(T) = \frac{1}{T}\sum_t f_t(x_t) - \frac{1}{T}\min_{x\in X}\sum_t f_t(x)$$

.

Why only fixed actions in hindsight? We give a brief example demonstrating that it is not possible in general to attain the total reward of the best sequence of actions in hindsight with an online policy. This motivates the definition of regret as the total loss compared to the best fixed action. Consider the following repeated game: at each timestep, the action space available to the player is $[0, 1]$. Nature chooses loss function $f_t(x) = x$ with probability $1/2$ and $f_t(x) = 1-x$ with probability $1/2$. Observe that without knowledge of f_t (i.e. in the online case), and for any algorithm the learner adopts, $\mathbb{E}[f_t(x_t)] = \frac{1}{2}$, and thus $\mathbb{E}\left[\sum_{t=1}^T f_t(x_t)\right] = \frac{1}{2}T$. However, in hindsight one can just choose $x = 1$ or $x = 0$ based on f_t and obtain a loss of zero in hindsight.

In contrast, we will find that defining regret via the best fixed action in hindsight allows for online strategies that attain *vanishing* regret ($R(T) \to 0$ as $T \to \infty$).

3.1 No-Regret Learning

First, we formalize this concept of vanishing regret:

Definition 2 *We say that an online learning algorithm is a no-regret algorithm if it achieves regret $R(T) \to 0$ as $T \to \infty$.*

One of the central results in online convex optimization tells us that under some conditions on the loss functions f_t (namely, Lipschitz continuity[1], no-regret algorithms are indeed attainable:

Theorem 3 *Suppose, $\forall t = 1, \cdots, T$, f_t is convex and L-Lipschitz. There exists a learning algorithm such that $R(T) \le O(L/\sqrt{T})$*

Before proving Theorem 3, we first propose a natural algorithm that does not attain vanishing regret on its own, but is a key building block in our construction of no-regret algorithms.

3.1.1 First attempt: follow-the-leader(FTL)

A natural approach that a learner might use to minimize regret is as follows: after playing an action x_t and observing the loss function f_t, the player can compute what the best (loss-minimizing) action was in hindsight, i.e. $\arg\min_x \sum_{\tau\le t} f_\tau(x)$. On the next day, the learner simply uses this "best x" as their action for the next loss function.

This algorithm, known as *follow-the-leader* (FTL), bears an interesting resemblance to our definition of the regret $R(T)$ (Definition 2). In particular, one might hope that by playing the best action in hindsight at each timestep, the player minimizes the difference in total loss between itself and the overall best action in hindsight.

Unfortunately, this turns out not to be the case: in the following example, we show that there exist games where FTL incurs constant average regret. Specifically, by making the functions at times t and $t + 1$ dramatically different, the environment can force the player to always play poorly:

[1]Recall that a function f is L-Lipschitz continuous if for all x and y, $|f(x) - f(y)| \le L \cdot |x - y|$.

Example. The permissible action space is $\mathcal{X} = [-1, 1]$. Nature chooses function f_t as:

$$f_t(x) = \begin{cases} \frac{1}{2}x & \text{if } t = 1 \\ -x & \text{if } t \neq 1 \text{ is even} \\ x & \text{if } t \neq 1 \text{ is odd} \end{cases}$$

Then for any even t, $f_t(x_t) = -x_t$, with x_t set according to hindsight:

$$x_t = \arg\min_{x \in [-1,1]} \sum_{\tau < t} f_\tau(x) = \arg\min_{x \in [-1,1]} \frac{1}{2}x = -1,$$

and thus $f_t(x_t) = 1$. Similarly, for any odd t,

$$x_t = \arg\min_{x \in [-1,1]} \sum_{\tau < t} f_\tau(x) = \arg\min_{x \in [-1,1]} -\frac{1}{2}x = 1,$$

while the loss function at time t is $f_t(x) = x$, making the incurred loss again 1. Thus, the total loss of FTL is thus $R(T) = T$, while playing the fixed action 0 at each timestep yields a loss of $\sum_t f_t(0) = 0$ (an upper bound on the loss of the best fixed action in hindsight). Thus, FTL has constant average regret (also known as linear total regret), and is thus not a no-regret algorithm.

As we find in the following section, however, a slight modification to the FTL algorithm does in fact yield a no-regret playing strategy.

3.1.2 A Second Attempt: Follow-the-Regularized-Leader(FTRL)

The fatal flaw of the proposed FTL algorithm, we claim, was its instability: by changing the loss function dramatically from timestep to timestep, the environment was able to induce highly suboptimal play from the learner. To address this, we consider the addition of a *convex regularizer* to FTL—specifically, rather than simply choosing the best action in hindsight, the learner now chooses:

$$x_t \in \arg\min_{x \in X} \left[\sum_{\tau < t} f_\tau(x) + \frac{1}{\eta} R(x) \right],$$

for some value of η, and a strongly convex function $R(\cdot)$. The algorithm resulting from this decision model is known as follow-the-regularized-leader, or FTRL. For the sake of clarity, we recall here the definition of a strongly convex function:

Definition 4 *$R : X \to \mathbb{R}$ is α-strongly convex w.r.t norm $\| \cdot \|$ iff for all $x, x_0 \in X$:*

$$R(x) \geq R(x_0) + \nabla R(x_0)^T (x - x_0) + \frac{\alpha}{2} \|x - x_o\|^2$$

Adding this strongly convex regularizer to the decision of the learner actually turns out to be the key in designing a no-regret algorithm. In fact, the following theorem (which we leave unproved, as the proof is rather long and outside of the scope of the class) demonstrates that for any strongly convex regularizer $R(\cdot)$, there is a setting of η that turns FTRL into a no-regret algorithm:

Theorem 5 *Suppose, $\forall t = 1, \cdots, T$, f_t is convex and L-Lipschitz w.r.t some norm $\| \cdot \|$, and R is α-strongly convex w.r.t $\| \cdot \|$. Then FTRL with parameter $\eta = \frac{\sqrt{\max R(x) - \min R(x)}}{\alpha L \sqrt{T}}$, the regret $R(T)$ is upper bound by $L\sqrt{\frac{\max R(x) - \min R(x)}{T}}$.*

Illustration. Although a proof of the above theorem is beyond the scope of these notes, the performance of FTRL on the counterexample to the vanishing regret of FTL proposed in the previous section is quite illustrative. Recall the "pathological" game presented in our discussion of FTL:

$$f_t(x) = \begin{cases} \frac{1}{2}x & \text{if } t = 1 \\ -x & \text{if } t \neq 1 \text{ is even} \\ x & \text{if } t \neq 1 \text{ is odd} \end{cases}$$

In the case of FTL, this sequence of loss functions caused the learner to alternate between playing -1 and 1, which in turn were actually the worst actions at their corresponding times. Now we consider FTRL, with the simplest 1-strongly convex regularizer, $R(x) = \frac{1}{2}x^2$, and $\eta = 2T^{-1/2}$ set according to Theorem 5. Then, at even values of t the learner instead plays:

$$x_t = \arg\min_{x\in[-1,1]} \sum_{\tau<t} f_\tau(x) + \frac{1}{\eta}R(x) = \arg\min_{x\in[-1,1]} \frac{1}{2}x + \frac{1}{4}\sqrt{T}x^2 = -\frac{1}{\sqrt{T}},$$

and by symmetry the learner plays $1/\sqrt{T}$ on odd timesteps, leading to average regret $1/\sqrt{T}$ (or total regret $\sqrt{T}$). Intuitively, adding regularization to FTL prevented it from "overfitting" x to the past—the learner oscillates between $1/\sqrt{T}$ and $-1/\sqrt{T}$, rather than -1 and 1.

Special cases of FTRL. It turns out that under specific choices of regularizer $R(\cdot)$, FTRL is reduced to other well-known learning algorithms. We illustrate this through the following two claims:

Claim 6 *FTRL with an ℓ_2-regularizer is approximately gradient descent, when f_t are linear functions.*

Proof Let $R(x) = \frac{1}{2}\|x\|_2^2$ and $g_t(\cdot)$ be the gradient of $f_t(\cdot)$. Let $c_t(x) = \sum_{\tau<t} f_\tau(x) + \frac{1}{2\eta}\|x\|_2^2$ and $x_t = \arg\min_x c_t(x)$, which implies $\nabla c_t(x_t) = \sum_{\tau<t} g_\tau + \frac{1}{\eta}x_t = 0$. Thus, for any t, $x_t = -\eta \sum_{\tau<t} g_\tau$. Rewrite the above equation, we have $x_t = x_{t-1} - \eta g_{t-1}$, where $x_1 = 0$. ■

In fact, the connection between FTRL with an ℓ_2 regularizer and gradient descent runs even deeper. In general, FTRL with ℓ_2-regularizer on function sequence $h_t(x) = f_t(x_t) + \nabla f_t(x_t)(x - x_t)$, $t = 1, 2, \cdots, T$ is the same as gradient descent on fucntion sequence $f_t(x)$, $t = 1, 2, \cdots, T$.

Claim 7 *FTRL on simplex with negative entropy regularizer, which is approximate to multiplicative-weights-update method.*

Proof The proof proceeds in a similar manner to that of Claim 6. ■

3.2 Follow The Regularized Leader and the Min-Max Theorem

In this subsection, we describe a crucial connection between FTRL and the central Min-Max theorem presented earlier. Suppose $f(x, y)$ convex-concave, and both x and y players run FTRL, that is:

- The x player chooses x_t by applying FTRL to losses $f(\cdot, y_t)$
- The y player chooses y_t by applying FTRL to losses $-f(x_t, \cdot)$

The following theorem connects the actions of the players using FTRL with minimax equilibria:

Theorem 8 *If x and y players play as above, then*

$$\frac{1}{T}\sum_{t=1}^{T} f(x_t, y_t) = \min_x \max_y f(x, y) \pm O\left(\frac{1}{\sqrt{T}}\right)$$

Moreover, the average strategies $\bar{x}_T = \frac{1}{T}\sum_t x_t$ *and* $\bar{y}_T = \frac{1}{T}\sum_t y_t$ *are a* $O\left(\frac{1}{\sqrt{T}}\right)$*-approximate Nash Equilibrium, i.e.*

$$f(\bar{x}_T, \bar{y}_T) \le \min_x f(x, \bar{y}_T) + O\left(\frac{1}{\sqrt{T}}\right)$$
$$f(\bar{x}_T, \bar{y}_T) \ge \max_y f(\bar{x}_T, y) - O\left(\frac{1}{\sqrt{T}}\right)$$

Proof Since both min and max players run FTRL, following Theorem 5, for x-player we have

$$\begin{aligned} \frac{1}{T}\sum_{t=1}^{T} f_t(x_t, y_t) &\le \min_x \frac{1}{T}\sum_{t=1}^{T} f_t(x, y_t) + O\left(\frac{1}{\sqrt{T}}\right) && \text{Theorem 5} \\ &\le \min_x f(x, \frac{1}{T}\sum_t y_t) + O\left(\frac{1}{\sqrt{T}}\right) && f \text{ is convave w.r.t } y \\ &\le \min_x \max_y f(x, y) + O\left(\frac{1}{\sqrt{T}}\right) \end{aligned} \tag{3}$$

Similarly, for y-player,

$$\begin{aligned} \frac{1}{T}\sum_{t=1}^{T} f_t(x_t, y_t) &\ge \max_y \frac{1}{T}\sum_{t=1}^{T} f_t(x_t, y) - O\left(\frac{1}{\sqrt{T}}\right) && \text{Theorem 5} \\ &\ge \max_y f\left(\frac{1}{T}\sum_t x_t, y\right) - O\left(\frac{1}{\sqrt{T}}\right) && f \text{ is convex w.r.t } x \\ &\ge \max_y \min_x f(x, y) - O\left(\frac{1}{\sqrt{T}}\right) \\ &= \min_x \max_y f(x, y) - O\left(\frac{1}{\sqrt{T}}\right) \end{aligned} \tag{4}$$

Combining the two inequalities as above, we have

$$\min_x \max_y f(x, y) - O\left(\frac{1}{\sqrt{T}}\right) \le \frac{1}{T}\sum_{t=1}^{T} f_t(x_t, y_t) \le \min_x \max_y f(x, y) + O\left(\frac{1}{\sqrt{T}}\right)$$

Moreover,

$$f(\bar{x}_T, \bar{y}_T) \le \max_y f(\bar{x}_T, y) \overset{(4)}{\le} \frac{1}{T}\sum_t f_t(x_t, y_t) + O\left(\frac{1}{\sqrt{T}}\right) \overset{(3)}{\le} \min_x f(x, \bar{y}_T) + O\left(\frac{1}{\sqrt{T}}\right)$$

$$f(\bar{x}_T, \bar{y}_T) \ge \min_x f(x, \bar{y}_T) \overset{(3)}{\ge} \frac{1}{T}\sum_t f_t(x_t, y_t) - O\left(\frac{1}{\sqrt{T}}\right) \overset{(4)}{\ge} \min_x f(\bar{x}_T, y) - O\left(\frac{1}{\sqrt{T}}\right)$$

Thus, $(\bar{x}_T, \bar{y}_T)$ is an approximate Min-Max equilibrium. ■

4 Back to GANs

4.1 On the Convergence of Parameters

Recall the formulation of the Wasserstein GAN [2]:

$$\inf_{\theta_g} \sup_{\theta_d} \; f(\theta_g, \theta_d) \tag{5}$$

where

$$f(\theta_g, \theta_d) = \mathbb{E}_{x \sim F}[D_{\theta_d}(x)] - \mathbb{E}_{z \sim N(0,I)}[D_{\theta_d}(G_{\theta_g}(z))]. \tag{6}$$

In a typical GAN, f will not be convex-concave w.r.t. θ_g and θ_d. This makes the direct use of Min-Max Theorem inapplicable to the above GAN formulation. Even if f were convex-concave, then using the results from Section 3 we could at best argue that $\bar{\theta}_g$ and $\bar{\theta}_d$ when optimized via FTRL would approach a saddle point, where:

$$\bar{\theta}_g = \frac{1}{T} \sum_t \theta_{g_t} \tag{7}$$

$$\bar{\theta}_d = \frac{1}{T} \sum_t \theta_{d_t} \tag{8}$$

However, this analysis does not have any guarantees on what the final parameters θ_{g_T} and θ_{d_T} will converge to, even in the case where f is convex-concave. To illustrate this problem, consider $f(x, y) = (x - \frac{1}{2})(y - \frac{1}{2})$. Alternating between optimizing x and y with gradient ascent/ descent yields to divergence as we can see in Figure 5.

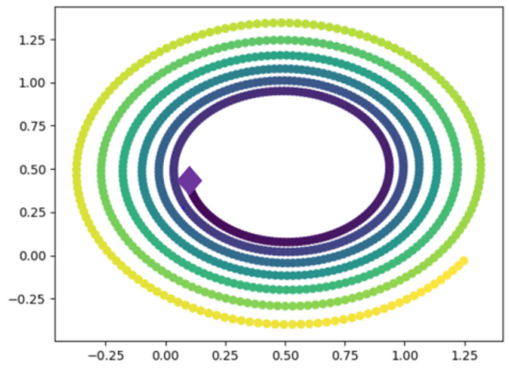

Figure 5: $f(x, y) = xy$. x and y are initialized at the purple diamond. Alternating between gradient ascent/ descent on x and y leads to divergent behavior, spiraling away from the optimum, but the average of the parameters is close to the optimal solution.

In this example, while averaging x and y would be close to the optimal solution $x = y = \frac{1}{2}$, the final values x_T and y_T diverge away from the optimum.

4.2 Optimistic Mirror Descent

Oscillatory behavior can also be observed in the GAN setting. Consider the case when the GAN is attempting to learn the isotropic normal distribution $N(V, I)$. Let $G_{\theta_g}(z) = z + \theta_g$ and $D_{\theta_d}(x) = \theta_d \cdot x$ (where z, x, θ_g and θ_d are all vectors). Figure 5 depicts oscillatory behavior when optimizing using gradient descent.

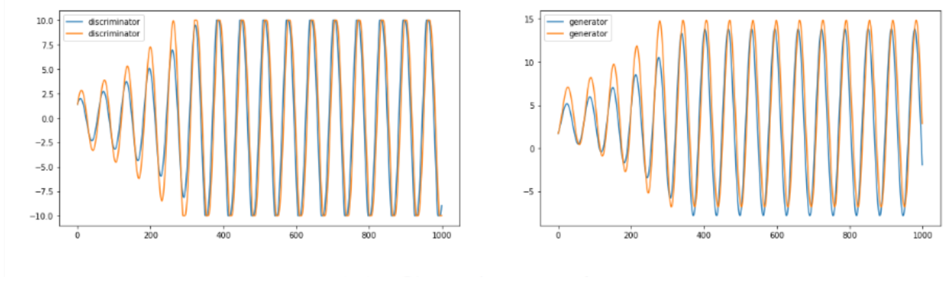

Figure 6: [4] Gradient descent fails to converge to a solution for a two-dimensional θ_g and θ_d. The optimal solution is $\theta_{g_1} = 3$, $\theta_{g_2} = 4$ and $\theta_d = \vec{0}$.

One recently proposed solution to address the above oscillatory behavior is to use **optimistic mirror descent** in place of gradient descent [4] for training Wasserstein GANs [2]. The update rule for optimistic mirror descent is given by:

$$x_t = x_{t-1} - 2\eta\nabla f(x_{t-1}) + \eta\nabla f(x_{t-2}) \tag{9}$$

4.2.1 Momentum

To provide some intuition on optimistic mirror descent, we begin by reviewing momentum. Recall the update rule for gradient descent with momentum:

$$v_t = \gamma v_{t-1} - \eta\nabla f(x_{t-1}) \tag{10}$$

$$x_t = x_{t-1} + v_t \tag{11}$$

To obtain an ϵ-accurate solution for a quadratic f, it is well-known that gradient descent takes $O\left(\kappa \log\left(\frac{1}{\epsilon}\right)\right)$ iterations, where κ denotes the condition number of f's Hessian. For poorly condition Hessians, the linear dependence on κ can cause gradient descent to converge very slowly. It can be shown that momentum reduces the number of steps to $O\left(\sqrt{\kappa} \log\left(\frac{1}{\epsilon}\right)\right)$ [6], which makes the algorithm significantly faster than gradient descent in such cases. This speed-up is often referred to as "acceleration" in the literature.

Importantly, while the above bound on the number of steps gradient descent takes to converge applies to general strongly convex functions, it has yet to be shown that momentum leads to accelerated convergence in the general strongly convex case. However, it has been shown that momentum can converge for strongly convex function asymptotically in the same number of steps as gradient descent [5].

In non-convex optimization (notably, for training deep networks), momentum commonly outperforms gradient descent in speed of convergence. A common justification for this tendency is that v_t strengthens the components of the gradient that consistently point in the same direction for multiple t [13] and weakens the components that do not. This ultimately means that the updates quickly move x_t in directions of the gradient that are consistent over prior t compared to gradient descent which does not use prior gradient history in its updates.

Interestingly, in non-convex settings momentum often converges to *better* local minima than gradient descent. [13] presented several experiments training recurrent networks and deep autoencoders where using momentum lead to significantly lower losses compared to gradient descent. Momentum remains one of the most widely used optimization algorithms in deep learning due to its empirical successes. For example, many architectures that have won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) have been trained with momentum. This includes ResNet [8], which won in 2015, as well as Squeeze-and-Excitation networks [9], which won in 2017.

4.2.2 Returning to Optimistic Mirror Descent

Both optimistic mirror descent and momentum incorporate prior observed gradients in their update rules. However, whereas momentum moves partially in the direction of past gradients, optimistic mirror descent moves partially in the *opposite* direction of the gradient observed in the prior timestep. To this extent, optimistic mirror descent can be roughly interpreted as employing a "reverse" momentum that undoes some of its past progress.

Using optimistic mirror descent, it has been shown that the final timestep parameters θ_g and θ_d will converge to the optimum for bilinear games. For example, it fixes the oscillatory behavior in the isotropic normal distribution example whereas other variants of gradient descent fail.

Outside of convex-concave problems, optimistic mirror descent (as well as other variants of gradient descent) provides no guarantees that stable solutions will be local min-max points. Optimistic mirror descent has been shown to produce samples from a WGAN trained on CIFAR-10 of quality comparable with GANs trained using other popular gradient descent-based optimization algorithms when measured using the Inception score.

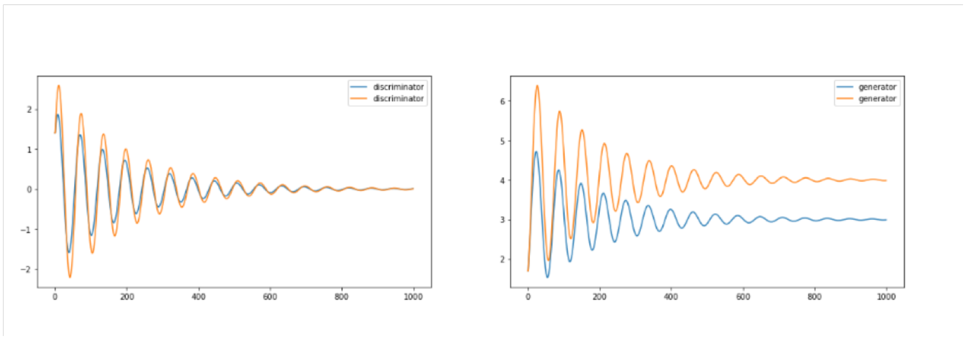

Figure 7: [4] Optimistic mirror descent converges to the optimal solution for θ_g and θ_d.

4.3 Mode Collapse

A significant challenge with evaluating the performance of GANs is that it is unclear which saddle points in f correspond to a G_{θ_g} that has learned the underlying distribution of data it is trying to model and a D_{θ_d} that is competent at distinguish data coming from synthetic distribution created by G_{θ_g} versus the true one. There may be many local saddle points in f, but we do not know how to identify which ones have these properties. It is also possible that f contains no local saddle points at all, in which case it is unclear where a good final solution would converge to.

One particularly problematic solution that f can converge to involves a discriminator that is good at distinguishing between real samples and synthetic samples from a subset of modes in the true distribution. When this happens, no matter what move θ_g makes (i.e. regardless of a small change in θ_g), the discriminator can clearly distinguish between the two distributions at such modes. Such a solution can result in the problem of **mode collapse**, when the generator becomes incapable of producing diverse samples that cover the full underlying distribution it is trying to model. Instead, the generator focuses only on producing samples from a subset of the distribution's modes where it is still capable of confusing the discriminator.

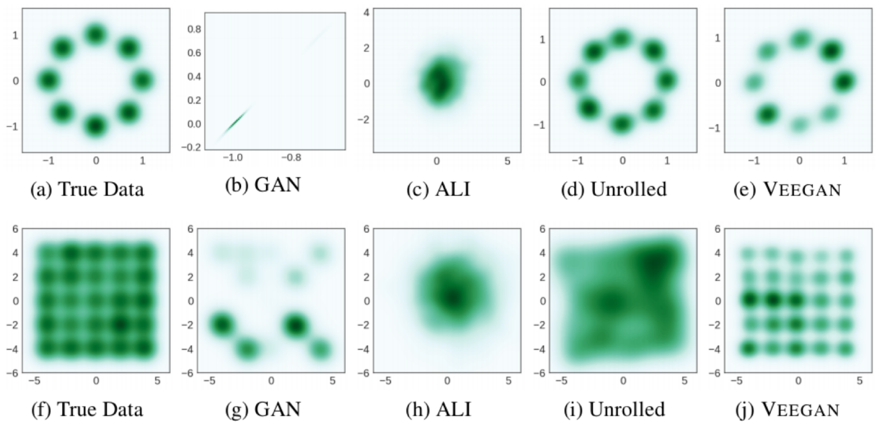

Figure 8: [12] Various GAN architectures' attempt to learn underlying data ((a) and (f)). The original GAN suffers from mode collapse since it generates samples from only a few modes in the true distribution ((b) and (g)). ALI has completely failed to generate samples from any modes in the distribution in (c), and it seemingly suffers from mode collapse in (h).

References

[1] Ilan Adler. The equivalence of linear programs and zero-sum games. *International Journal of Game Theory*, 42(1):165–177, 2013.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[3] George B Dantzig. Maximization of a linear function of variables subject to linear inequalities. *New York*, 1951.

[4] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training gans with optimism. *arXiv preprint arXiv:1711.00141*, 2017.

[5] Euhanna Ghadimi, Hamid Reza Feyzmahdavian, and Mikael Johansson. Global convergence of the heavy-ball method for convex optimization. In *Control Conference (ECC), 2015 European*, pages 310–315. IEEE, 2015.

[6] Gabriel Goh. Why momentum really works. *Distill*, 2017.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017.

[10] J v Neumann. Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1):295–320, 1928.

[11] Ben Polak. Handout on mixed strategies, 2007.

[12] Akash Srivastava, Lazar Valkoz, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pages 3310–3320, 2017.

[13] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.