IFT 6085 - Lecture 14
The Numerics of GANs

Scribes
Winter 2020: Oleksiy Ostapenko
Winter 2019: Adam Ibrahim, William St-Arnaud, Bhairav Mehta
Winter 2018: Joshua Romoff

Instructor: Ioannis Mitliagkas

1 Summary

In the previous lecture, we discussed the Wasserstein Generative Adversarial Network (WGAN), which uses the Wasserstein distance instead of the traditional Jenson-Shannon divergence. WGANs help alleviate the zero-gradient problem that arises when we have a mismatch in support between the generative and true distribution.

In this lecture, we will focus on the training dynamics for GANs. Specifically, we will frame the training objective for GANs as a zero-sum game, and focus on one particular method, taken from [3], that helps GANs convergence to Nash Equilibria.

2 Contraction Mappings

Definition 1 (Contraction Mappings.)**.** *Let (X,d) be a metric space. A mapping $g : X \to X$ is a contruction mapping, or contraction, if there exists a constant c , with $0 \leq c < 1$, such that:*

$$d(g(x), g(y)) \leq cd(x, y) \tag{1}$$

for all $x, y \in X$. The scalar c is called contraction modulus.

Meaning that applying the operator G brings those trajectories closer together. We note that for some choise of distance metric d, g might or might not be a contraction mapping.

Definition 2 (Cauchy sequence.)**.** *A sequence $\{x_k\}$ is called Cauchy sequence if for every $\epsilon > 0$, there exists some K such that $|x_k - x_m| < \epsilon$ for all $k \geq K$ and $m \geq K$.*

Proposition 3 (Prop. A.26 Bertsekas [1])**.** *[Contruction Mapping Theorem] Suppose that $g : X \to X$ is a contraction mapping with $c \in [0, 1)$ and X is a closed subset of $\mathcal{R}^n$. Then:*

(a) (Existence and Uniqueness of a Fixed Point) The mapping g has a unique fixed point $x* \in X$.

(b) (Convergence Rate) For every initial vector $x^0 \in X$, the sequence $\{x^k\}$ generated by $x^{x+1} = g(x^k)$ converges to $x*$. In particular $||x^k - x^*|| \leq c^k||x^0 - x^*||, \forall k \geq 0$.

(Note, intuitively a fixed point is the element of a function's domain that is mapped to itself by this function.)
Proof Prop. 3: We first prove (a):

For a sequence $\{x_k\}$ generated by $x^k - 1 = g(x^k)$, we have that g is a mapping from subset X of R^n to itself:

$$||g(x) - g(y)|| \leq c||x - y||, \tag{2}$$

and therefore:

$$||x^k + 1 - x^k|| \leq c||x^k - x^k - 1||, \forall k \geq 1. \tag{3}$$

This implies

$$||x^k + 1 - x^k|| \leq c||x^1 - x^0||, \forall k \geq 1. \tag{4}$$

We can unroll the sequence and show that its a Cauchy sequence:

$$\begin{aligned} ||x^k + m - x^k|| &\leq \sum_{i=1}^{\infty} ||x^k + i - x^k + i - 1|| && \text{unroll the sequence} \\ &\leq c^k(1 + c + ... + c^{m-1})||x^1 - x^0|| && \text{plug in (4) and use telescoping sum} \\ &\leq \frac{c^k}{1-c}||x^1 - x^0|| && \text{use the property of geometric series} \end{aligned}$$

It can be concluded that $\{x_k\}$ is a Cauchy sequence and must converge to a limit $x^* \in X$ (since we assumed that X is a closed set) (see Prop. A.5 in [1]).
We now show that x^* is a fixed point. We first use the triangular inequality:

$$||g(x^*) - x^*|| \leq ||g(x^*) - x^k|| + ||x^k - x^*|| \leq c||x^* - x^k - 1|| + ||x^k - x^*||.$$

Since x^k converges to x^*, $g(x^*) = x^*$. Therefor x^* is the fixed point og g.
By contradiction we can prove that x^* is unique. Suppose y^* is another fixed point, then we could have:

$$||x^* - y^*|| = ||g(x^*) - g(y^*)|| \leq c||x^* - y^*||,$$

but this implies that $x^* = y^*$, since this can only hold if $||x^* - y^*|$ or $c == 1$, but $c \in [0, 1)$ by definition of contraction mapping.
Now we prove (b). Here we have:

$$||x^{k'} - x^*|| = ||g(x^{k'}) - g(x*)|| \leq c||x^{k'} - x^*|| \forall k' \geq 1.$$

We can apply this relation successively fr $k' = k, k - 1, ...1$ to get the desired result.

3 Vector fields in optimization

Traditionally, in optimization, we look for $^* \in \arg\min \mathcal{L}^{(\ )}(\ )$ where the loss function $\mathcal{L}$ is smooth and differentiable. A typical strategy consists in looking for stationary points, i.e. points where the gradient of the loss function vanishes (although these may very well be maxima or saddle points too). In practice, gradient descent is often used to find local minima of $\mathcal{L}^{(\ )}$. This leads us to consider the vector field $\mathsf{v}(\ ) = \nabla\mathcal{L}^{(\ )}(\ )$.

Definition 4 (Vector field)**.** *A vector field is a function mapping a subset of* R^n *to* R^n*. In particular, a vector field* v *is said to be conservative if* $\exists f : \mathsf{R}^n \to \mathsf{R}$ *such that* $\mathsf{v} = \nabla f$*.*

Since v is conservative, the dynamics are straightforward. Indeed, the update rule $F_\eta(\ _t)\ ,\ \ _{t+1} = \ _t - \eta\mathsf{v}(\ _t)$ means that the vector field at each point will point downhill and, if the function is convex, will trace a path to a minimum of the loss function (where $\mathsf{v}(\ ) = 0$). We can see from the definition of F_η for $\eta \neq 0$ that points where the gradient of $\mathcal{L}$ vanishes (equivalently, stationary points of v) correspond to fixed points of F_η, i.e. such that $F_\eta(\ ) = \ $. One may retrieve convergence guarantees for gradient descent by looking at the Jacobian of F_η.

Definition 5 (Jacobian)**.** *A generalization of the derivative for vector-valued functions. For $f : \mathsf{R}^n \to \mathsf{R}^m$, the Jacobian J is defined as:*

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

The Jacobian of F_η is given by $\nabla F_\eta(\ ) = \mathsf{I} - \eta \nabla \mathsf{v}(\ )$. Note that $\nabla \mathsf{v}$ is the Hessian of $\mathcal{L}$, which implies that the Jacobian is symmetric and has real eigenvalues. In this case, the eigenvalues of the Jacobian are directly related to those of the Hessian of $\mathcal{L}$:

$$\lambda(\nabla F_\eta(\ )) = 1 - \eta \lambda(\nabla \mathsf{v}(\ )), \tag{5}$$

where $\lambda(\nabla \mathsf{v}(\ ))$ represents an eigenvalue of the Hessian of $\mathcal{L}$, and $\lambda(\nabla F_\eta(\ ))$ is the eigenvalue of the corresponding Jacobian. Prop. 6 provides a local convergence result, based on the spectral radius of the Jacobian of F_η.

Proposition 6 (Prop. 4.4.1 Bertsekas [1])**.** *If the spectral radius ρ_{max} , $\rho(\nabla F_\eta(\ ^*)) < 1$, then, for $_0$ in a neighborhood of *, the distance of $_t$ to the stationary point * converges at a linear rate of $\mathcal{O}\ (\rho_{max} + \epsilon)^t$, $\forall \epsilon > 0$.*

(**Note**: generally this is a local convergence result - we require to start in a small neighbourhood around the solution.)
In particular, as the spectral radius and the operator 2-norm coincide for symmetric or hermitian matrices, we have that if $\rho(\nabla F_\eta(\ ^*)) = \max |\lambda(\nabla F_\eta(\ ^*))| < 1$, fast local convergence is guaranteed by Prop. 6.
To illustrate this, consider a quadratic loss $\mathcal{L}^{(\theta)}(\theta) = \frac{h}{2}\theta^2$. Then the vector field is given by $v(\theta) = h\theta$ and its gradient by $\nabla v(\theta) = h$. Thus the eigenvalues of the Jacobian can be easily computed:

$$\begin{aligned} \lambda(\nabla F_\eta(\theta)) &= 1 - \eta \lambda(\nabla v(\theta)) \\ &= 1 - \eta h \end{aligned} \tag{6}$$

Notably, in the case of quadratics, Prop. 6 is a global convergence result.

Proof outline of Prop. 6. Note, that ρ_{max} , $\rho(\nabla F_\eta(\ ^*)) < 1$ is a spectral radius at the solution θ^*. To prove the proposition we need to establish the spectral radius as a contraction mapping in the neighbourhood around the solution, which would allow us to make statements about the convergence using the convergence property of the contraction mapping (see definition 1). To do this, we want to show that for for any point in that neighbourhood, the spectral radius of the Jacobian at that point is within the unit circle, but we want to know the radius of that neighbourhood. For this [1] constructs an argument using a careful perturbation of the Jacobian, which will implicitly give us the radius of the neighbourhood. And if we start withing this neighbourhood, we converge to the solution with the rate given in the proposition.
Here is the constructed argument. Intuitively: the eigenvalues of a matrix are a continuous mapping of the values of the matrix in a sense. That is, if I perturb the values of the matrix a bit, also the eigenvalues change a bit.
More formally, let R^* denote the Jacobian of $F_\eta(\theta^*)$. We can approximate it with $R = R^* + \delta E$, where δ is a scalar and E is some arbitrary perturbation matrix. By changing δ we can control how much perturbation we add to R^* - that is how accurate is our approximation of R^*. Then the $\lambda(R^*)$ is a continuous contraction mapping w.r.t. δ. This means:

$$\exists \delta > 0, s.t. |\lambda_i(R) - \lambda_i(R^*)| < \epsilon.$$

. Here, δ gives the neighborhood radius and ϵ influences the convergence rate.

4 GANs as games

Recall the min max formulation of GANs,

$$\min_G \max_D V(D, G) = \mathsf{E}_{x \sim \mathbb{P}_x}[\log D(x)] + \mathsf{E}_{z \sim \mathbb{P}_z}[\log(1 - D(G(z)))] \tag{7}$$

GAN optimization can be interpreted as a game, where each player is trying to minimize their own loss function. In that formalism, the goal of training is to find a Nash equilibrium.

Definition 7 (Nash Equilibrium)**.** *Given two players parametrized by and , and their respective loss functions* $\mathcal{L}^{( )}(,)$ *and* $\mathcal{L}^{( )}(,)$*, we say that* $\mathbf{z} = (^*, ^*)$ *is a Nash Equilibrium if we have simultaneously:*

$$\begin{aligned} ^* &\in \arg\min_{\in} \mathcal{L}^{( )}(, ^*) \\ ^* &\in \arg\min_{\in} \mathcal{L}^{( )}(^*,) \end{aligned} \tag{8}$$

As with extremas in optimization, a Nash equilibrium can be local or global. A local Nash equilibrium is a point $\mathbf{z}^*$ *where the above holds in a neighbourhood of* $\mathbf{z}^*$*.*

Intuitively, a Nash equilibrium is a point where neither player can improve their situation unilaterally.

As the losses involved often are smooth and differentiable, as well as the decision variables involved live in a continuous space, we look for local Nash equilibrium using Gradient Descent, which can either update simultaneously the parameters (i.e. $_{t+1}$, $_{t+1}$ only depend on $_t$, $_t$), or alternate updating w.l.o.g. then (in which case we may compute $_{t+1}$ first and then use it instead of $_t$ to compute $_{t+1}$). If we consider the vector field

$$\mathbf{v}(,) = \begin{matrix} \nabla \mathcal{L}^{( )}(,) \\ \nabla \mathcal{L}^{( )}(,) \end{matrix} \tag{9}$$

one may express the gradient descent update operator as

$$F_\eta(,) , \qquad - \eta \mathbf{v}(,) \tag{10}$$

Note that in this case the vector field $\mathbf{v}$ is no longer conservative in general, as in contrast to optimisation, it no longer can be expressed as the gradient of a real-valued function (single objective function). In particular, it is a concatenation of the gradients of two objective functions and we may observe rotational dynamics. This is due to the fact that the Jacobian is no longer symmetric in general, and may have complex eigenvalues.
For example, consider a bilinear game

$$\begin{aligned} \mathcal{L}^{( )}(\phi, \theta) &= \theta\phi, \\ \mathcal{L}^{( )}(\phi, \theta) &= -\theta\phi \end{aligned} \tag{11}$$

Then,

$$\mathbf{v}(\phi, \theta) = \begin{matrix} \theta \\ -\phi \end{matrix} , \nabla \mathbf{v}(\phi, \theta) = \begin{matrix} 0 & 1 \\ -1 & 0 \end{matrix} , \nabla F_\eta(\phi, \theta) = \begin{matrix} 1 & -\eta \\ \eta & 1 \end{matrix} \tag{12}$$

and since the eigenvalues of $\nabla \mathbf{v}$ are $-i$, i, (there is no real component) note that applying $\nabla \mathbf{v}$ 4 times to its eigenvectors yields back the eigenvectors. Note, the solution of the above game is

$$^* = \begin{matrix} 0 \\ 0 \end{matrix} , \tag{13}$$

which is a Nash Equilibrium.

4.1 Zero-sum games

We assume that we are in the smooth two-player game setting where player 1 is trying to maximize $f(,)$, player 2 is trying to maximize $g(,)$, and f, g are both smooth with respect to $(,)$. We also consider the GAN setting which can be cast as a Zero-sum game.

Below describes a notational difference between [3] and what we saw in class:

1. Within the paper, the functions are denoted as f, g, whereas in class, we used $\mathcal{L}^{(\Phi)}, \mathcal{L}^{(\Theta)}$.

2. [3] looks at maximizing the functions f, g, whereas we are interested in minimization of $\mathcal{L}^{(\Phi)}, \mathcal{L}^{(\Theta)}$. To make these equivalent, we simply note that
$$\mathcal{L}^{(\Phi)}(\ ,\ ) = -f(\ ,\ )$$
$$\mathcal{L}^{(\Theta)}(\ ,\ ) = -g(\ ,\ )$$

3. $\mathcal{L} \neq L$; the former denotes the functions we are optimizing, whereas the latter denotes a function of the vector field in [3], as we will see below.

4. The step size in [3] is denoted as h.

Definition 8 (Zero-sum games)**.**
$$\mathcal{L}^{(\Theta)}(\ ,\ ) = -\mathcal{L}^{(\Phi)}(\ ,\ )$$

In zero-sum games, any gain by one player results in a loss to the other player and vice versa; we want to converge to a Nash Equilibrium. Here, the Jacobian of the vector field is given by

$$\nabla \boldsymbol{v}(\ ,\ ) = \begin{matrix} \nabla^2 \mathcal{L}^{(\Phi)}(\ ,\ ) & \nabla\ \nabla\ \mathcal{L}^{(\Phi)}(\ ,\ ) \\ -\nabla\ \nabla\ \mathcal{L}^{(\Phi)}(\ ,\ )^T & -\nabla^2 \mathcal{L}^{(\Phi)}(\ ,\ ) \end{matrix} \tag{14}$$

Lemma 9. *For* 0*-sum games, the following are equivalent:*

- $\nabla \boldsymbol{v}(\ ,\ )$ *is positive (semi)-definite*
- $\nabla^2 \mathcal{L}^{(\Phi)}(\ ,\ )$ *is positive (semi)-definite and* $\nabla^2 \mathcal{L}^{(\Phi)}(\ ,\ )$ *is negative (semi)-definite*

Using this lemma for a stationary point of $\boldsymbol{v}$ gives a sufficient condition for the existence of a local N.E. Indeed, this is what the following corollary says:

Corollary 10. *The following holds for all* 0*-sum games:*

- $\nabla \boldsymbol{v}(\ ^*,\ ^*)$ *is positive semi-definite* $\forall$ *local N.E* $(\ ,\ )$
- *If* $(\ ,\ )$ *is a stationary point of* $\boldsymbol{v}$ *and* $\nabla \boldsymbol{v}(\ ,\ )$ *is positive definite, then* $(\ ,\ )$ *is a local N.E.*

Summarizing the previous results, if we have a stationary point $(\ ,\ )$ of $\boldsymbol{v}$, $\nabla^2 \mathcal{L}^{(\Phi)}(\ ,\ )$ is positive definite and $\nabla^2 \mathcal{L}^{(\Theta)}(\ ,\ )$ is negative definite, we have that $(\ ,\ )$ is a local N.E. Now, in the gradient descent algorithm, we compute iterates $(\ _{t+1},\ _{t+1}) = (\ _t,\ _t) - \eta \boldsymbol{v}((\ _t,\ _t))$. Proposition 6 from the optimization section gives us a convergence guarantee for the gradient descent algorithm. We understand that the convergence rate is dictated by the largest (absolute) eigenvalue. A fixed point signifies that we are in a local optimum. In order to ensure convergence to the local N.E. $(\ ,\ )$, we must start in a particular neighbourhood U of $(\ ,\ )$. In other words, we might not get convergence to a local N.E. unless we have a good initial point $(\ _0,\ _0)$.

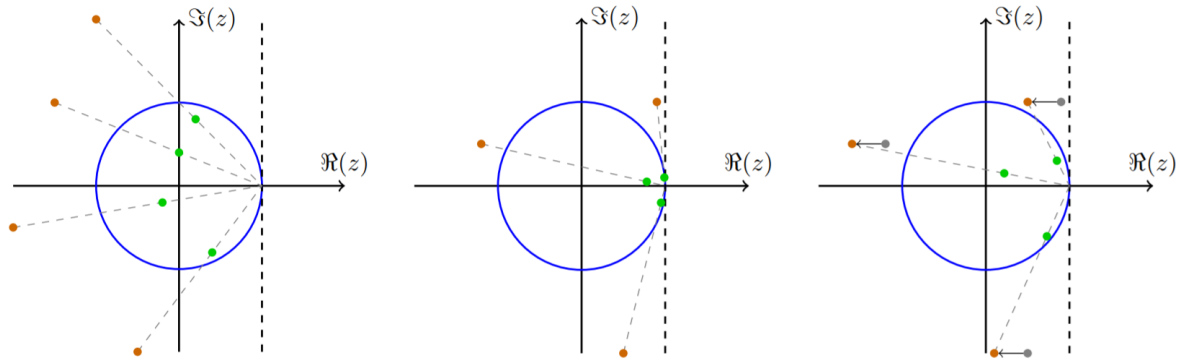
Recall the issue where $\nabla \boldsymbol{v}$ is not symmetric and therefore the eigenvalues of F_η and $\nabla \boldsymbol{v}$ are not necessarily real, they can have an imaginary component. The authors of [3] uncover a sufficient and necessary condition for the conditions of proposition 6 to hold.

Lemma 11 (Lemma 4 of Mescheder et al. 2017 [3])**.** *Assume that* $A \in \mathbb{R}^{d \times d}$*. If* $\forall \lambda \in \sigma(A)$ *(where* $\sigma(A)$ *denotes the spectrum of* A*) we have* $Re(\lambda) < 0$ *and* $h > 0$ *then:*

$$|\lambda| < 1 \quad \forall \lambda \in \sigma(I + hA)$$

if and only if

$$h < \frac{1}{|Re(\lambda)|} \frac{2}{1 + (\frac{Im(\lambda)}{Re(\lambda)})^2}$$



(a) Illustration how the eigenvalues are projected into unit ball.

(b) Example where h has to be chosen extremely small.

(c) Illustration how our method alleviates the problem.

Figure 1: Illustration of how eigenvalues are shifted into the unit ball for smoother optimization.

So, in order for the eigenvalues to be within the unit-ball, the learning rate must be made sufficiently small. Figure 1 illustrates this effect. If either the real or imaginary component have very large magnitude then a very small learning rate is needed to project the eigenvalues back into the unit ball (remark: in optimization since the Hessian is symmetric, the we only had real eigenvalues; here, because the vector field is not conservative, we lost the symmetry property of ∇v, whose eigenvalues can be complex numbers).

Since we need to ensure that the eigenvalues stay within the unit ball, we can at each step tune the learning rate or momentum parameter. We can also try to optimize a completely different objective, as shown below. All approaches try to tune the optimization with the hope of making the vector field *more conservative*. Intuitively, we want to control the rotations in the optimization, which will help us converge to local optima faster and more reliably.
So we have the problem that the eigenvalues have strong imaginary component, and so this causes strong rotations. In order to keep those eigenvalues in the unit circle (that is to have convergence guarantees) we are forced to use a very low, tiny learning rate (so long convergence). One of the ocntributions is consensus approach, where we modigy the vector field and translate the eigenvalues in that directions. So we make the reak component of the eigenvalues of the Jacobian of the vector field more positive, which pushes the eigenvalues of the Jacobian of the operator more towards the left. And so we can use a bigger step size. It improves the ration of the real to imaginary component - gives a stronger real component - makes the vector field more concervative?? -¿ the rotations are less strong.

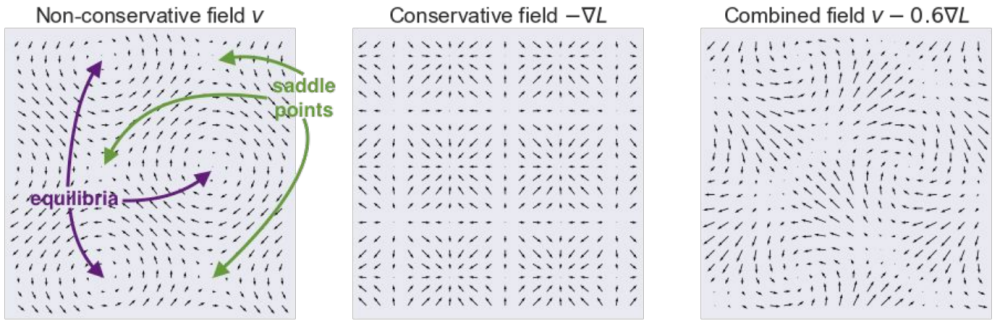


Figure 2: An illustration of an annotated vector field, conservative field, and the combined field.

The first approach that we can take to tame the rotational dynamics in the vector field is to optimize a completely different objective altogether. As seen in [3], we can try to minimize the norm of the gradient. If we have a vector field $\boldsymbol{v}$, shown in Panel 1 of Figure 2, we can optimize the following instead:

$$L(\ ,\ ) = \frac{1}{2}||\boldsymbol{v}(\ ,\ )||^2 \tag{15}$$

Unfortunately, in practice, this optimization is unstable as the minimization causes saddle points to become attractive, as shown in Panel 2 of Figure 2. This is also a second order optimization, which is expensive, yet can be accomplished using for example Jacobian vector product. This kind of approach also makes bad stationary points attractive (also saddle points).

To solve this issue while still maintaining the benefits of the conservative vector field, the authors propose a gradient penalty, using the above norm of gradient as regularizer controlled by a hyperparameter γ. In practice, with the right γ, one can get a combined field that may avoid the false equilibria associated with the conservative field $-\nabla L$ (Panel 3 of Figure 2). This combination leads to smoother optimization, as it intuitively scales back the eigenvalues back within the unit ball, while still allowing for larger learning rates.

5 Negative Momentum

The previous section showed that for smoother optimization, we need to control the eigenvalues, which can be done by adding a regularizer that shifts the eigenvalues. Effectively, we want to control the rotations in the vector field, which are troublesome as they slow (or prevent) convergence. In this section, we describe the approach taken by [2], which introduces *negative momentum* and describes how it can be used to tame the GAN optimization dynamics.

5.1 Momentum

We recall the *heavy ball* method of momentum, originally introduced by [4]:

$$_{t+1} = \ _t - \eta \boldsymbol{v}(\ _t) + \beta(\ _t - \ _{t-1}) \tag{16}$$

where the momentum term is controlled with hyperparameter β.

In the following, we describe the approach of [2], which provides analysis of *why* large, positive values of momentum are harmful in game optimization, while also providing evidence on how negative momentum (i.e negative values of β) can be helpful in certain types of zero-sum games.

We begin by rewriting the momentum equation for use with simultaneous, 2-player gradient descent as follows:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta \boldsymbol{v}(\boldsymbol{x}_t) + \beta(\boldsymbol{x}_t - \boldsymbol{x}_{t-1}), \quad \boldsymbol{x}_t = (\ _t, \ _t)$$

Using the above definition, we can also define the fixed point operator, which requires a state-space augmentation.

$$F_{\eta,\beta}(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t) := \begin{matrix} \boldsymbol{I}_n & \boldsymbol{0}_n \\ \boldsymbol{I}_n & \boldsymbol{0}_n \end{matrix} \begin{matrix} \boldsymbol{x}_t \\ \boldsymbol{x}_{t-1} \end{matrix} - \eta \begin{matrix} \boldsymbol{v}(\boldsymbol{x}_t) \\ \boldsymbol{0}_n \end{matrix} + \beta \begin{matrix} \boldsymbol{I}_n & -\boldsymbol{I}_n \\ \boldsymbol{0}_n & \boldsymbol{0}_n \end{matrix} \begin{matrix} \boldsymbol{x}_t \\ \boldsymbol{x}_{t-1} \end{matrix} \tag{17}$$

5.2 Negative Momentum in Games

The paper goes on to show that in a special class of games, called *bilinear games*, a negative momentum parameter can be optimal (and necessary for convergence) for alternated gradient descent (in a bilinear game, simultaneous gradient descent with any choice of momentum parameters may diverge). Figure 5.2 shows an illustration of how negative momentum can lead to convergence with alternated gradient descent, as well as a depiction of how various other methods lead to divergence.
For general games, the paper continues on to show that negative momentum can be helpful as well, and show strong empirical evidence on training "saturating" GANs (i.e. where we get very small or zero gradients) with negative momentum.

References

[1] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.