IFT 6085 - Lecture 14
The Numerics of GANs

This version of the notes has not yet been thoroughly checked. Please report any bugs to the scribes or instructor.

Scribes
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 Vector fields in optimization

Traditionally, in optimization, we look for $\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}} \mathcal{L}^{(\boldsymbol{\theta})}(\boldsymbol{\theta})$ 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 $\mathcal{L}^{(\boldsymbol{\theta})}$. This leads us to consider the vector field $\boldsymbol{v}(\boldsymbol{\theta}) = \nabla\mathcal{L}^{(\boldsymbol{\theta})}(\boldsymbol{\theta})$.

Definition 1 (Vector field)**.** *A vector field is a function mapping a subset of $\mathbb{R}^n$ to $\mathbb{R}^n$. In particular, a vector field $\boldsymbol{v}$ is said to be conservative if $\exists f : \mathbb{R}^n \to \mathbb{R}$ such that $\boldsymbol{v} = \nabla f$.*

Since $\boldsymbol{v}$ is conservative, the dynamics are straightforward. Indeed, the update rule $F_\eta(\boldsymbol{\theta}_t) \triangleq \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\boldsymbol{v}(\boldsymbol{\theta_t})$ 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 $\boldsymbol{v}(\boldsymbol{\theta}) = 0$). 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 $\boldsymbol{v}$) correspond to fixed points of F_η, i.e. $\boldsymbol{\theta}$ such that $F_\eta(\boldsymbol{\theta}) = \boldsymbol{\theta}$. One may retrieve convergence guarantees for gradient descent by looking at the Jacobian of F_η.

Definition 2 (Jacobian)**.** *A generalization of the derivative for vector-valued functions. For $f : \mathbb{R}^n \to \mathbb{R}^m$, 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 $\nabla F_\eta(\boldsymbol{\theta}) = \boldsymbol{I} - \eta\nabla\boldsymbol{v}(\boldsymbol{\theta})$. Note that $\nabla\boldsymbol{v}$ 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 $\mathcal{L}$:

$$\lambda(\nabla F_\eta(\boldsymbol{\theta})) = 1 - \eta\lambda(\nabla\boldsymbol{v}(\boldsymbol{\theta})) \tag{1}$$

Prop. 3 provides a local convergence result, based on the spectral radius of the Jacobian of F_η.

Property 3 (Prop. 4.4.1 Bertsekas [1]). *If the spectral radius $\rho_{max} \triangleq \rho(\nabla F_\eta(\boldsymbol{\omega}^*)) < 1$, then, for $\boldsymbol{\omega}_0$ in a neighborhood of $\boldsymbol{\omega}^*$, the distance of $\boldsymbol{\omega}_t$ to the stationary point $\boldsymbol{\omega}^*$ converges at a linear rate of $\mathcal{O}\left((\rho_{max} + \epsilon)^t\right)$, $\forall \epsilon > 0$.*

In particular, as the spectral radius and the operator 2-norm coincide for symmetric or hermitian matrices, we have that if $\rho(\nabla F_\eta(\boldsymbol{\theta}^*)) = \max|\lambda(\nabla F_\eta(\boldsymbol{\theta}^*))| < 1$, fast local convergence is guaranteed by Prop. 3.
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:

$$\begin{aligned}\lambda(\nabla F_\eta(\theta)) &= 1 - \eta\lambda(\nabla v(\theta)) \\ &= 1 - \eta h\end{aligned} \tag{2}$$

Notably, in the case of quadratics, Prop. 3 is a global convergence result.

3 GANs as games

Recall the min max formulation of GANs,

$$\min_G \max_D V(D, G) = \mathbb{E}_{x\sim\mathbb{P}_x}[\log D(x)] + \mathbb{E}_{z\sim\mathbb{P}_z}[\log(1 - D(G(z)))] \tag{3}$$

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 4 (Nash Equilibrium). *Given two players parametrized by $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, and their respective loss functions $\mathcal{L}^{(\boldsymbol{\Phi})}(\boldsymbol{\phi}, \boldsymbol{\theta})$ and $\mathcal{L}^{(\boldsymbol{\Theta})}(\boldsymbol{\phi}, \boldsymbol{\theta})$, we say that $\boldsymbol{z^*} = (\boldsymbol{\phi}^*, \boldsymbol{\theta}^*)$ is a Nash Equilibrium if we have simultaneously:*

$$\begin{aligned}\boldsymbol{\phi}^* &\in \arg\min_{\boldsymbol{\phi}\in\boldsymbol{\Phi}} \mathcal{L}^{(\boldsymbol{\Phi})}(\boldsymbol{\phi}, \boldsymbol{\theta}^*) \\ \boldsymbol{\theta}^* &\in \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \mathcal{L}^{(\boldsymbol{\Theta})}(\boldsymbol{\phi}^*, \boldsymbol{\theta})\end{aligned} \tag{4}$$

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

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

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

$$\boldsymbol{v}(\boldsymbol{\phi}, \boldsymbol{\theta}) = \begin{bmatrix}\nabla_{\boldsymbol{\phi}}\mathcal{L}^{(\boldsymbol{\Phi})}(\boldsymbol{\phi}, \boldsymbol{\theta}) \\ \nabla_{\boldsymbol{\theta}}\mathcal{L}^{(\boldsymbol{\Theta})}(\boldsymbol{\phi}, \boldsymbol{\theta})\end{bmatrix} \tag{5}$$

one may express the gradient descent update operator as

$$F_\eta(\boldsymbol{\phi}, \boldsymbol{\theta}) \triangleq \begin{bmatrix}\boldsymbol{\phi} \\ \boldsymbol{\theta}\end{bmatrix} - \eta\boldsymbol{v}(\boldsymbol{\phi}, \boldsymbol{\theta}) \tag{6}$$

Note that in this case the vector field $\boldsymbol{v}$ is no longer conservative in general, as in contrast to optimisation, it is no longer the gradient of a real-valued function. In particular, 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

$$\begin{aligned}\mathcal{L}^{(\boldsymbol{\Phi})}(\phi, \theta) &= \theta\phi, \\ \mathcal{L}^{(\boldsymbol{\Theta})}(\phi, \theta) &= -\theta\phi\end{aligned} \tag{7}$$

Then,

$$\boldsymbol{v}(\phi,\theta) = \begin{bmatrix} \theta \\ -\phi \end{bmatrix}, \nabla \boldsymbol{v}(\phi,\theta) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \nabla F_\eta(\phi,\theta) = \begin{bmatrix} 1 & -\eta \\ \eta & 1 \end{bmatrix} \tag{8}$$

and since the eigenvalues of $\nabla \boldsymbol{v}$ are $-i, i$, note that applying $\nabla \boldsymbol{v}$ 4 times to its eigenvectors yields back the eigenvectors.

3.1 Zero-sum games

We assume that we are in the smooth two-player game setting where player 1 is trying to maximize $f(\boldsymbol{\phi}, \boldsymbol{\theta})$, player 2 is trying to maximize $g(\boldsymbol{\phi}, \boldsymbol{\theta})$, and f, g are both smooth with respect to $(\boldsymbol{\phi}, \boldsymbol{\theta})$. 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
$$\mathcal{L}^{(\Phi)}(\boldsymbol{\phi}, \boldsymbol{\theta}) = -f(\boldsymbol{\phi}, \boldsymbol{\theta})$$
$$\mathcal{L}^{(\Theta)}(\boldsymbol{\phi}, \boldsymbol{\theta}) = -g(\boldsymbol{\phi}, \boldsymbol{\theta})$$
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 5 (Zero-sum games)**.**
$$\mathcal{L}^{(\Theta)}(\boldsymbol{\phi}, \boldsymbol{\theta}) = -\mathcal{L}^{(\Phi)}(\boldsymbol{\phi}, \boldsymbol{\theta})$$

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

$$\nabla \boldsymbol{v}(\boldsymbol{\phi}, \boldsymbol{\theta}) = \begin{bmatrix} \nabla^2_{\boldsymbol{\phi}} \mathcal{L}^{(\Phi)}(\boldsymbol{\phi}, \boldsymbol{\theta}) & \nabla_{\boldsymbol{\phi}} \nabla_{\boldsymbol{\theta}} \mathcal{L}^{(\Phi)}(\boldsymbol{\phi}, \boldsymbol{\theta}) \\ -\nabla_{\boldsymbol{\phi}} \nabla_{\boldsymbol{\theta}} \mathcal{L}^{(\Phi)}(\boldsymbol{\phi}, \boldsymbol{\theta})^T & -\nabla^2_{\boldsymbol{\theta}} \mathcal{L}^{(\Phi)}(\boldsymbol{\phi}, \boldsymbol{\theta}) \end{bmatrix} \tag{9}$$

Lemma 6. *For* 0*-sum games, the following are equivalent:*

- $\nabla \boldsymbol{v}(\boldsymbol{\phi}, \boldsymbol{\theta})$ *is positive (semi)-definite*
- $\nabla^2_{\boldsymbol{\phi}} \mathcal{L}^{(\Phi)}(\boldsymbol{\phi}, \boldsymbol{\theta})$ *is positive (semi)-definite and* $\nabla^2_{\boldsymbol{\theta}} \mathcal{L}^{(\Phi)}(\boldsymbol{\phi}, \boldsymbol{\theta})$ *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 7. *The following holds for all* 0*-sum games:*

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

Summarizing the previous results, if we have a stationary point $(\boldsymbol{\phi}^*, \boldsymbol{\theta}^*)$ of $\boldsymbol{v}$, $\nabla^2_{\boldsymbol{\phi}} \mathcal{L}^{(\Phi)}(\boldsymbol{\phi}, \boldsymbol{\theta})$ is positive definite and $\nabla^2_{\boldsymbol{\theta}} \mathcal{L}^{(\Theta)}(\boldsymbol{\phi}, \boldsymbol{\theta})$ is negative definite, we have that $(\boldsymbol{\phi}^*, \boldsymbol{\theta}^*)$ is a local N.E. Now, in the gradient descent algorithm, we compute iterates $(\boldsymbol{\phi_{t+1}}, \boldsymbol{\theta_{t+1}}) = (\boldsymbol{\phi_t}, \boldsymbol{\theta_t}) - \eta \boldsymbol{v}((\boldsymbol{\phi_t}, \boldsymbol{\theta_t}))$. Proposition 3 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. $(\boldsymbol{\phi}^*, \boldsymbol{\theta}^*)$, we must start in a particular neighbourhood U of $(\boldsymbol{\phi}^*, \boldsymbol{\theta}^*)$. In other words, we might not get

convergence to a local N.E. unless we have a good initial point $(\boldsymbol{\phi_0}, \boldsymbol{\theta_0})$.

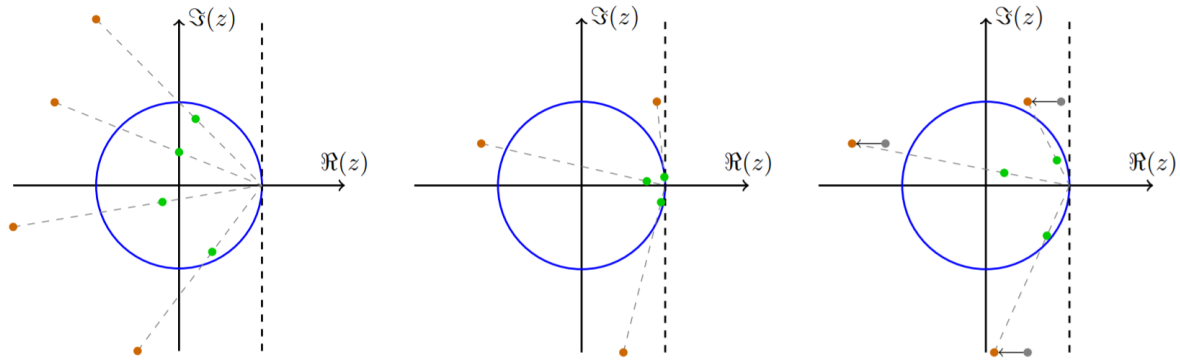
Recall the issue where $\nabla \boldsymbol{v}$ is not symmetric and therefore the eigenvalues of F_η and $\nabla \boldsymbol{v}$ are not necessarily real, they can have an imaginary component. The authors of [3] uncover a sufficient and necessary condition for the conditions of proposition 3 to hold.

Lemma 8 (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.

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.

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:

$$L(\boldsymbol{\phi}, \boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{v}(\boldsymbol{\phi}, \boldsymbol{\theta})||^2 \tag{10}$$

Unfortunately, in practice, this optimization is unstable as the minimization causes saddle points to become attractive, as shown in Panel 2 of Figure 2.

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

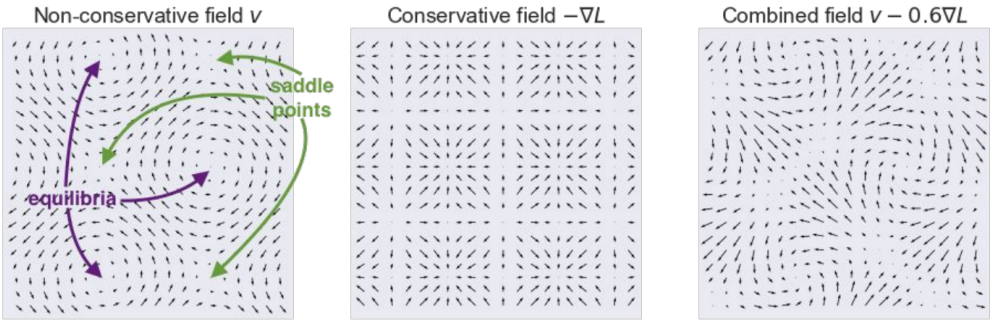


Figure 2: An illustration of an annotated vector field, conservative field, and the combined field.

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.

4 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.

4.1 Momentum

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

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{v}(\boldsymbol{\theta}_t) + \beta(\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}) \tag{11}$$

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:

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

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

$$F_{\eta,\beta}(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t) := \begin{bmatrix} \boldsymbol{I}_n & \boldsymbol{0}_n \\ \boldsymbol{I}_n & \boldsymbol{0}_n \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{x}_{t-1} \end{bmatrix} - \eta \begin{bmatrix} \boldsymbol{v}(\boldsymbol{x}_t) \\ \boldsymbol{0}_n \end{bmatrix} + \beta \begin{bmatrix} \boldsymbol{I}_n & -\boldsymbol{I}_n \\ \boldsymbol{0}_n & \boldsymbol{0}_n \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{x}_{t-1} \end{bmatrix} \tag{12}$$

4.2 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 4.2 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.

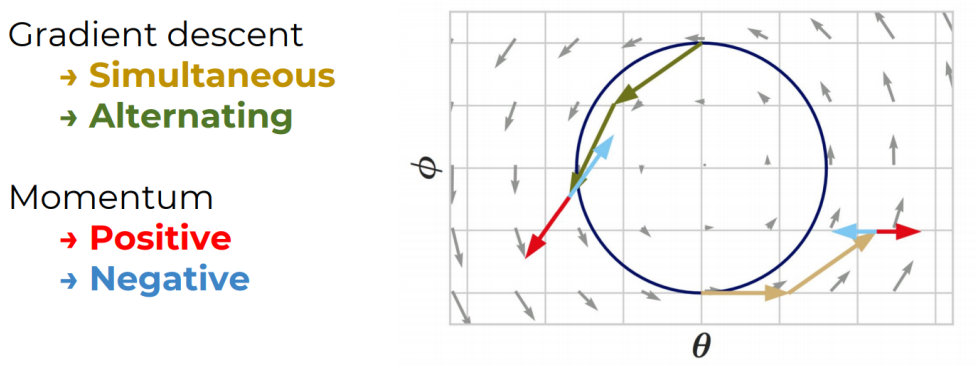


Figure 3: An illustration of how various choices for gradient descent and momentum parameters lead to different outcomes of optimization.

References

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

[2] G. Gidel, R. A. Hemmat, M. Pezeshki, G. Huang, R. L. Priol, S. Lacoste-Julien, and I. Mitliagkas. Negative momentum for improved game dynamics. *CoRR*, abs/1807.04740, 2018. URL `http://arxiv.org/abs/1807.04740`.

[3] L. M. Mescheder, S. Nowozin, and A. Geiger. The numerics of gans. *CoRR*, abs/1705.10461, 2017. URL `http://arxiv.org/abs/1705.10461`.

[4] B. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1 – 17, 1964. ISSN 0041-5553. doi: https://doi.org/10.1016/0041-5553(64)90137-5. URL `http://www.sciencedirect.com/science/article/pii/0041555364901375`.