Enforcing constraints for interpolation and extrapolation in Generative Adversarial Networks

Panos Stinis
(joint work with T. Hagge, A.M. Tartakovsky and E. Yeung)

Pacific Northwest National Laboratory

Supported by “Deep Learning for Scientific Discovery Investment" and “PhILMs"

Statement of the problem

Generative Adversarial Networks (GANs) are becoming popular machine learning choices for training generators.

There is a concerted effort in the machine learning community: i) to expand the range of tasks in which learning can be applied and ii) to utilize methods from other disciplines to accelerate learning.

Task: We want to enforce given constraints in the output of a GAN generator both for interpolation and extrapolation (prediction).

Given a time series, we wish to train GAN generators to represent the flow map of a system.

Remark: The cases of interpolation and extrapolation should be treated differently.

Definition and basic properties of GANs

We are interested in training a generator to produce data from a given distribution (called the true data distribution).

GANs formulate this task as a game between the generator and an associated discriminator (Goodfellow *et al.*, 2014).

The objective of the generator is to trick the discriminator into deciding that the generator-created samples actually come from the true data distribution.

The best possible performance of the generator is to convince the discriminator *half of the time* that the samples it is generating come from the true distribution.

A two-player minimax game with value function $V(D, G)$:

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

where $p_{data}(x)$ is the true distribution and $p_z(z)$ is the input distribution of the generator.

The generator G and the discriminator D are assumed to be neural networks with parameters θ_g and θ_d respectively.

For a given generator G, the optimal discriminator $D^*_G(x)$ is given by

$$D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)},$$

where $p_g(x)$ is the generator's output distribution.

We can define $C(G) = \max_D V(D, G) = V(D^*_G, G)$.

1) The global minimum of $C(G)$ is obtained if and only if $p_g = p_{data}$.

2) Since for this minimum we have $D^*_G(x) = \frac{1}{2}$, the value of the minimum is $-\log 4$.

3) For the generator, the minimum of $\log(1 - D(G(z)))$ is $-\log 2$.

Remark: GANs are notoriously difficult to train. In particular, the discriminator learns much faster than the generator and this leads to an instability of the training process.

Remark: There are many variants of the basic GAN framework. Our construction can be applied to those variants too.

Enforcing constraints for *interpolation*

We have a prior distribution $p_z(z)$ on the input of the generator and we would like to train the generator to produce samples from a function $f(z)$ i.e., we want the data $x = f(z)$.

In the original implementation of a GAN, for each z we produce an x that satisfies the constraint $x = f(z)$. We feed the discriminator both with the data x as well as the data $x' = G(z)$ that the generator produces.

Remark: This implementation is straightforward but does not utilize the known bond between z and x.

Idea: We can enforce the constraint by augmenting the discriminator input vector as (z, ϵ) where $\epsilon = x - f(z)$ for the true sample. Similarly, for the generator-created sample, we augment the discriminator input vector as (z, ϵ') where $\epsilon' = x - G(z)$.

Pros and cons

- The modification is easy to implement with modern software e.g. Tensorflow.
- We introduce a constraint for the data created by the generator.
- This constraint influences directly the parameters of the generator neural network (weights, biases in the case of convolutional neural nets) through back propagation.
- We respect the game-theoretic setup of GANs (under reasonable assumptions about the smoothness of the constraint).
- The modification can exacerbate the known instability issue of GANs.

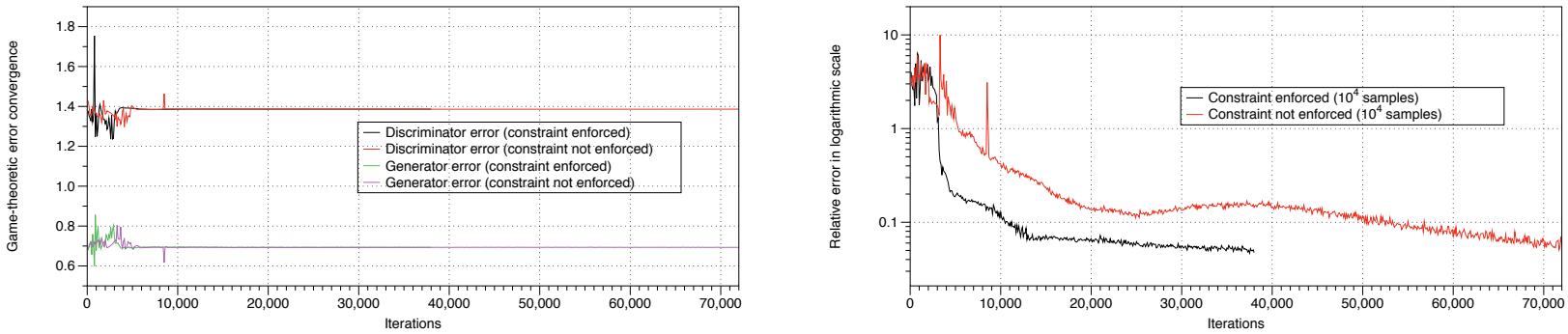
Adding noise to promote stability

To avoid the instability we can regulate the discriminator's ability to distinguish between generator-created and true data by adding noise to the constraint residual for the true data only.

- If the true data satisfy the constraint exactly, then we can decide how much noise to add based on Monte Carlo considerations.
- If the true data come from a numerical method (with given order of accuracy) or a physical experiment (with known precision), then we can use this information to decide the magnitude of the noise.

The idea behind adding the noise is that the discriminator will become more tolerant by allowing generator-created samples for which the constraint residual is not exactly 0 but essentially within a narrow interval around it.

Numerical example - interpolation

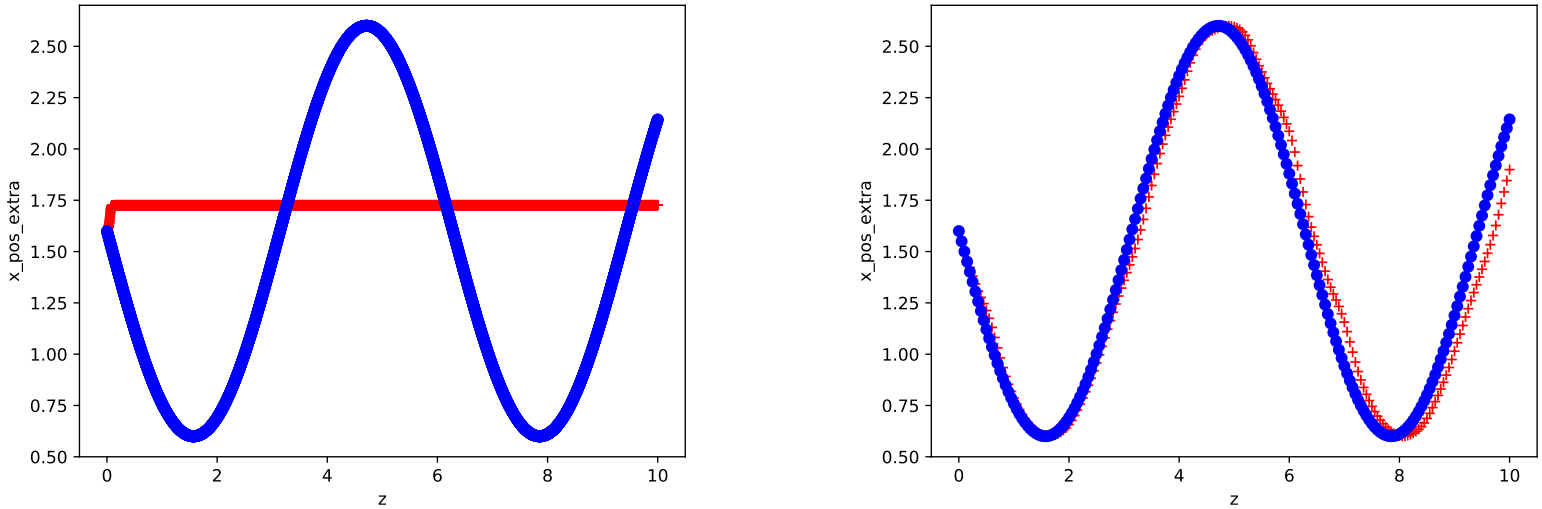


(a) (b)

Figure: Two linearly coupled oscillators. (a) Comparison of the evolution of the absolute value of the generator and discriminator game-theoretic error with and without enforcing a constraint (linear-linear plot) ; (b) Comparison of the evolution of the relative error RE_m of the function learned with and without enforcing a constraint (linear-log plot).

Remark: The game-theoretic optimum can be reached much faster than the actual relative error threshold (very different solution landscapes require adaptive learning rate)

Enforcing constraints for *extrapolation*



(a) (b)

Figure: Two linearly coupled oscillators. Comparison of target function $x_1(z) = R - \sin(z)$ (blue dots) and GAN prediction $x_{pos}^{extra}(z)$ (red crosses). (a) 10^4 samples of noiseless data with the constraints enforced during training and a projection step; (b) 10^4 samples of noisy data with the constraints enforced during training and a projection step.

The need for error-correction

Remark: The repeated application of the flow map neural network model (generative model) leads to error accumulation.

Remark: The predicted trajectory veers off in untrained parts of phase space. As a result, the generator predictions fail.

Idea: Introduce an error-correcting (restoring) force $\rightarrow$ Train an *interpolatory* generator with *noisy* data

Implementation: We center a cloud of points at each data point that is provided on the trajectory. Then, during training, at each step we force the GAN generator to map a point in this cloud to the correct (noiseless) point on the trajectory at the next step.

Remark: This is akin to temporal renormalization.

Generator dynamics = Approximate dynamics + Restoring force

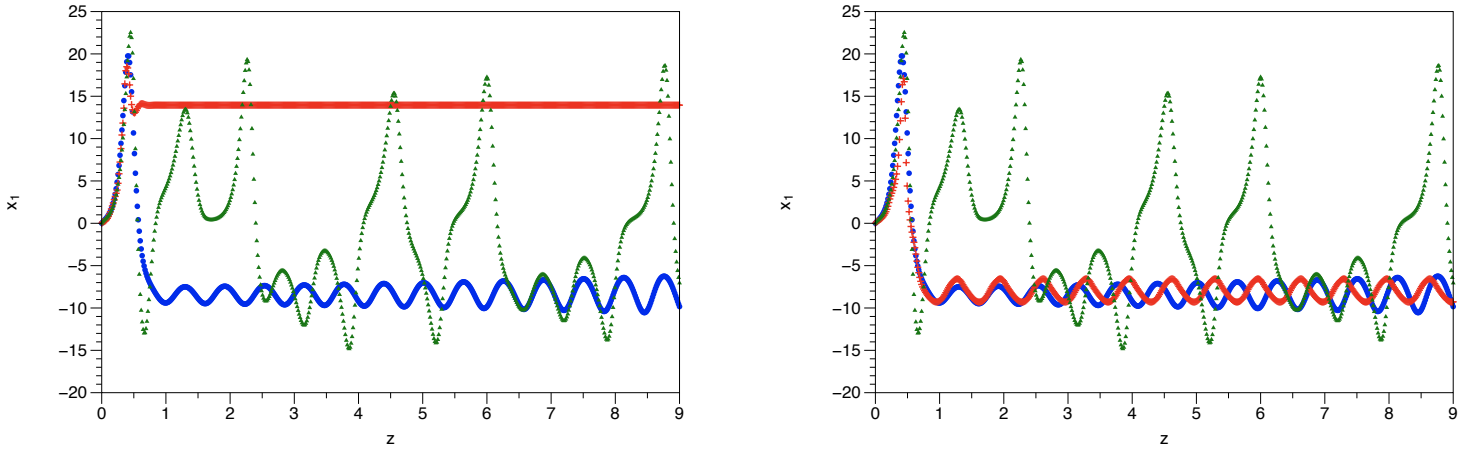
- Incorporate the restoring force implicitly by training with noisy data
- Incorporate the restoring force explicitly by training with noisy data and by learning the functional form of the restoring force

Remark: The functional form of the restoring force can come from (temporal) model reduction considerations.

Remark: It can even be represented by a *separate* neural network.

Remark: The magnitude of the noise depends on the accuracy of the ground truth (order of accuracy, measurement errors).

Remark: Expect scaling laws for parameters as a function of stepsize (incomplete similarity).



(a) (b)

Figure: Lorenz system. Comparison of ground truth $x_1(z)$ computed with the Euler scheme with stepsize $\delta z = 10^{-4}$(blue dots), the GAN prediction with stepsize $\Delta z = 1.5 \times 10^{-2}$ (red crosses) and the Euler scheme prediction with stepsize $\Delta z = 1.5 \times 10^{-2}$ (green triangles). (a) 2×10^4 samples of *noisy* data *without enforced constraints* during training; (b) 2×10^4 samples of *noisy* data *with enforced constraints* during training.

Current and future work

- We have obtained results for the construction both in the case of unsupervised learning (GANs) and supervised learning (add penalty terms to objective function).
- We will explore similar constructions for reinforcement learning. The role of noise is played there by the concept of stochastic exploration of the action policy space.
- Implementation in all 3 major models of learning allows comparison of the relative merits for each setting.
- This construction can be combined with online data acquisition to develop models that adapt to a changing environment.

P. Stinis, T. Hagge, A.M. Tartakovsky and E. Yeung, Enforcing constraints for interpolation and extrapolation in Generative Adversarial Networks (2018) arXiv:1803.08182