An Online Learning Approach to GAN - Review

Vaishnav K.V.
vaishnavk@iisc.ac.in

Abstract:

Even though GANs can accurately model complicated distributions, they are known to be difficult to train due to instabilities caused by a difficult minimax optimisation problem. In this paper, the author views the problem of training GANs as finding a mixed strategy in a zero-sum game. Building on ideas from online learning, we propose a novel training method named Chekov GAN. There are two main contributions of this paper, one in theory other in practice. On the theory side, the author shows that the algorithm proposed provably converges to equilibrium for semi-shallow GAN architectures. On the practical side, the paper proposes an efficient heuristic guided by the theoretical results, which we apply to commonly used deep GAN architectures.

The training procedure for Vanilla GAN:

- The optimal parameter is a PSNE, and we need to compute that point.

- Goodfellow's approach-Alternative gradient descent:
  We need to update the parameters of both blocks alternatively. Here we could use SGD to update. By training the vanilla GAN, I understood that the hyperparameters and the initial parameter values are essential in determining various aspects such as stability, sample diversity etc. I have given a simple version of the algorithm in the figure :

Algorithm 1 Single-Step Gradient Method

function SINGLESTEPGRADIENTITERATION($P, \theta^t, \omega^t, B, \eta$)
  Sample $X_P = \{x_1, \ldots, x_B\}$ and $X_Q = \{x'_1, \ldots, x'_B\}$, from P and Q_{θ^t}, respectively.
  Update: $\omega^{t+1} = \omega^t + \eta \nabla_\omega F(\theta^t, \omega^t)$.
  Update: $\theta^{t+1} = \theta^t - \eta \nabla_\theta F(\theta^t, \omega^t)$.
end function

Problems with Standard GAN:(Alternating gradient descent)

- Even for small games, failures are noted (Salimans et al. 4(#4))
- Convergence and oscillation issues (Metz et al. 5(#5))
  - Non-convergence refers to the Stability issue.
  - Oscillation across samples refers to Mode collapse.

Here is a problem, and a lot of papers have come up with a variety of solutions. This paper proposes one such.

Chekhov GAN

I assume the reader is familiar with the definition of the following terms: Two zero-sum player games, MSNE, PSNE and FTRL. Here I will give the definitions for eps-MSNE and some commonly used GAN architectures.

Theoretical background :

- On the existence of MSNE: We have theoretical guarantees for the existence of an MSNE (Mixed Strategy Nash Equilibrium) Nash et al. 6(#6)
- Regret bound for FTRL: For the FTRL algorithm, given the loss functions are convex, we have theoretical results for achieving sub-linear regret. (An important result from online learning literature)
- Computation of MSNE: If the loss and reward functions are convex and concave respectively, and these functions are the utility function for D and G respectively, then Freund and Schapire 7(#7)) (1999) proposes that no-regret algorithms can be used to find an approximate MSNE where the epsilon follows Big O(1 /sqrt(T))

Contribution-Theory side :

- For a semi-shallow architecture for GAN, the game structure it induces is a semi-concave game when the appropriate choice of activation function is made. By semi-concave, I mean that the objective function is concave with respect to the discriminator.

- The authors propose an algorithm which builds on top of Schapire's work on the Computation of MSNE. The following is the algorithm to

Algorithm 1 CHEKHOV GAN

Input: #steps T, Game objective $M(\cdot,\cdot)$
for $t = 1 \dots T$ **do**
Calculate:

$$\text{(Alg. } \mathcal{A}_1) \quad \mathbf{u}_t \leftarrow \text{FTRL}_1(f_0, \dots, f_{t-1}) \quad \& \quad \text{(Alg. } \mathcal{A}_2) \quad \mathbf{v}_t \leftarrow \text{FTRL}_2(g_0, \dots, g_{t-1})$$

Update: $f_t(\cdot) = M(\cdot, \mathbf{v}_t) \quad \& \quad g_t(\cdot) = M(\mathbf{u}_t, \cdot)$
end for
Output mixed strategies: $\mathcal{D}_1 \sim \text{Uni}\{\mathbf{u}_1, \dots, \mathbf{u}_T\}$, $\mathcal{D}_2 \sim \text{Uni}\{\mathbf{v}_1, \dots, \mathbf{v}_T\}$.

find the MSNE:

- In this particular setting(semi-shallow GAN), the authors are able to prove the convergence of the algorithm. Here I will provide a proof sketch:
The proof makes use of a theorem due to Schapire, which shows that if both A_1 and A_2 ensure no-regret then it implies convergence to approximate MNE. Since the game is concave with respect to P_2, it is well known that the FTRL version A_2 appearing in Thm is a no-regret strategy. The challenge is therefore to show that A_1 is also a no-regret strategy. This is non-trivial, especially for semi-concave games that do not necessarily have any special structure with respect to the generator.However, the loss sequence received by the generator is not arbitrary but rather it follows a special sequence based on the choices of the discriminator, $\{f_t(\cdot) = M(\cdot, v_t)\}_t$. In the case of semi-concave games, the sequence of discriminator decisions, $\{v_t\}_t$ has a special property which "stabilizes" the loss sequence $\{f_t\}_t$, which in turn enables us to establish no-regret for A_1.

- Essentially,the contribution made by this paper is to Convert a problem of finding nash equilibrium to a problem of solving an optimization problem.

- The analysis enables us to get some effcient heuristic while training the standard GAN.

Contribution - Practical side:

The following assumptions/restrictions are taken while proposing the practical version of the algorithm.

- Use FTRL for both the players.
- Take the gradient step each time instead of finding the argmin or argmax.
- Use a Queue of finite length.
- The latest generator is used for generating new samples.

Algorithm proposed:

Algorithm 2 Practical CHEKHOV GAN

Input: #steps T, Game objective $M(\cdot,\cdot)$, number of past states K, spacing m
Initialize: Set loss/reward $f_0(\cdot) = 0, g_0(\cdot) = 0$, initialize queues $\mathcal{Q}_1$.insert(f_0), $\mathcal{Q}_2$.insert(g_0)

for $t = 1 \dots T$ **do**
Update generator and discriminator based on a minibatch of noise samples and data samples:

$$\mathbf{u}_{t+1} \leftarrow \mathbf{u}_t - \nabla_{\mathbf{u}_t} \left(\frac{1}{|\mathcal{Q}_1|} \sum_{f \in \mathcal{Q}_1} f(\mathbf{u}) + \frac{C}{\sqrt{t}} \|\mathbf{u}\|^2 \right) \quad \& \quad \mathbf{v}_{t+1} \leftarrow \mathbf{v}_t - \nabla_{\mathbf{v}_t} \left(\frac{1}{|\mathcal{Q}_2|} \sum_{g \in \mathcal{Q}_2} g(\mathbf{v}) - \frac{C}{\sqrt{t}} \|\mathbf{v}\|^2 \right)$$

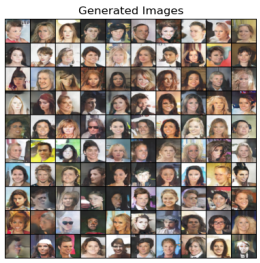
Calculate: $f_t(\cdot) = M(\cdot, \mathbf{v}_t)$ & $g_t(\cdot) = M(\mathbf{u}_t, \cdot)$
Update $\mathcal{Q}_1$ and $\mathcal{Q}_2$ (see main text or Algorithm 3 in the appendix)
end for

Implementation:

- Replace the init step with choosing the optimal parameter for the corresponding regulariser.
- Two queues are maintained. One for the generator and the other for the discriminator.
- DC-GAN 8(#8) architecture is used for implementation.

Experiments:

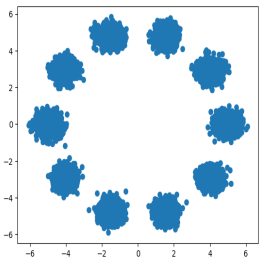
- The authors are comparing the performance of the proposed GAN algorithm with respect to the standard GAN's performance in the following datasets:
  - MNIST, CelebA,multi-modal gaussian.
- The comparison is with respect to the following aspects:
  - Stability during the training.
  - Sample diversity.
  - Mode collapse.
- The following is the configuration used for training(from the paper):
  batch_size = 128,image_size = (64,64),number of channels = 3,latent dimension = 100,epochs=20,lr (the parameter in Adam optimiser)= 0.0002,K=10,C=0.01
- For the gaussian dataset mixture, use a latent space dimension as 2.
- I have implemented GAN training using CelebA dataset and generated 100 images using the trained generator. The noise follows Normal distribution. I have given the images generated using standard GAN and the proposed approach. For GAN, the DC GAN architecture is used.



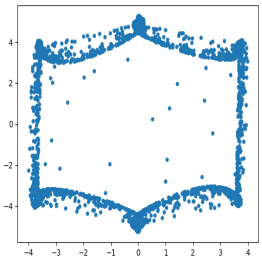
Images generated with vanilla GAN

Images generated with chekhov GAN

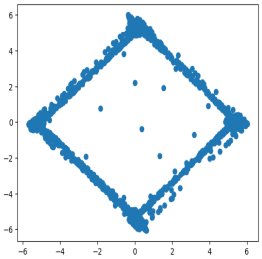
- To analyse the improvement in the modal collapse aspect, I have used multi-modal Gaussian. The heat map of the dataset is given in the figure. Symmetrical architectures are used for the Generator as well as for the Discriminator.



Heatmap for the training data



Heat map for the generated samples with vanilla GAN



Heat map for the generated samples with Chekhov GAN

Learning/Takeaways:

- There were no implementations available as the reference. So I have decided to build on top of standard GAN architectures, and the code will be available on "paper with code" as well as in the following GitHub repo: https://github.com/vaishn99/modified-GAN-training
- For the same problem, multiple solutions(Papers) were available. But I couldn't get if there are any connections across the solutions, whether one implies the other.
- There is a mistake in the queue updation algorithm given in the paper. But the organisation of the paper is really good, and it is appreciable.
- Here multiple loss terms contribute to the final loss. We need to see how PyTorch compute the gradient with respect to all these terms for a single update. There are some nuances we need to consider.9(#9)
- Building up theory-to-back concepts in deep learning is hard. Each time we update the parameters, the entire structure changes, which makes things challenging.

References

[1] Auto-Encoding Variational Bayes,https://arxiv.org/abs/1312.6114
[2] Understanding Diffusion Models: A Unified Perspective,https://arxiv.org/abs/2208.11970
[3] f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization,https://arxiv.org/abs/1606.00709
[4] Improved Techniques for Training GANs,https://arxiv.org/abs/1606.03498
[5] Unrolled Generative Adversarial Networks,https://arxiv.org/abs/1611.02163
[6] Lecture 5: Existence of a Nash Equilibrium, MIT lecture series.
[7] On Learning Algorithms for Nash Equilibria,https://people.csail.mit.edu/costis/learning.pdf
[8] Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,https://arxiv.org/abs/1511.06434
[9] How to combine multiple criteria for a loss function? ,https://discuss.pytorch.org/t/how-to-combine-multiple-criterions-to-a-loss-function/348