Generative Adversarial Networks (GANs) conditional Generative adversarial Networks (cGANs)

Machine Learning Seminars - Spring 2023

Soheil Saghafi

February 8, 2023

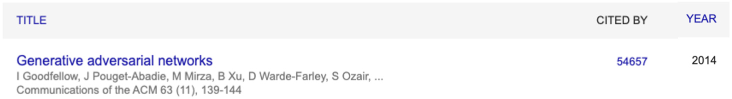
Outline

- Generative Adversarial Networks (GANs)
- conditional Generative Adversarial Networks (cGANs)
- Rosenbrock test function of two variables
- Conclusion and Future work

Generative adversarial network (Goodfellow et al, 2014), Integration of AI and mechanistic modeling in generative adversarial networks for stochastic inverse problems (Parikh et al, 2020)

Generative Adversarial Networks (GANs)

- Generative adversarial networks (GANs) are a machine learning framework designed by **Ian Goodfellow** and his colleagues in 2014.



TITLE	CITED BY	YEAR
Generative adversarial networks I Goodfellow, J Pouget-Abadie, M Mirza, B Xu, D Warde-Farley, S Ozair, ... Communications of the ACM 63 (11), 139-144	54657	2014

- The applications of GAN increased rapidly in different fields like:
  - Image processing (Editing photographs, Translating images, ...)
  - Fashion, art and advertising
  - Science (biomedical info, astrophysics, ...)
  - Improving cybersecurity
  - Improving healthcare
  - Video games (Generating animation models,...)
  - Cryptography (cipher cracking)

Generative Adversarial Networks (GANs)

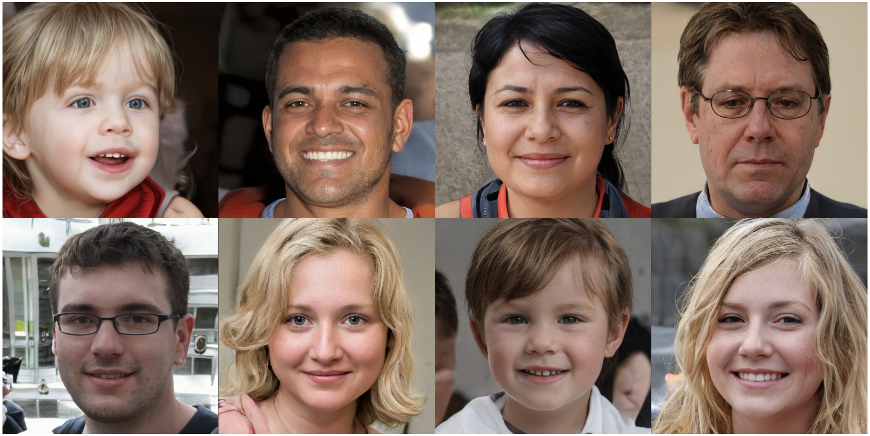


Figure 1: These images are created by a GAN (i.e. they are “fake” samples – none of them are actually real people).

https://thispersondoesnotexist.com

Generative Adversarial Network (GAN)

Generative adversarial network (GAN), is a deep neural network architecture comprised of two neural netwoks competing one against the other.

- Generative model (G): Try to find a distribution for the samples which is mimicking the real data distribution.
- Discriminative model (D): Estimates the probability that sample came from the training data rather than G.

The goal of the Generator (G) is to convert an initial random probability distribution $z \sim P_z$ into P_g where $P_g \sim P_{data}$, in order to fool the discriminator.
The role of the Discriminator (D) is to discriminate between $G(z) \sim P_g$ and the samples which are coming from real dataset $(x \in X) \sim P_{data}$.

Generative Adversarial Network (GAN)

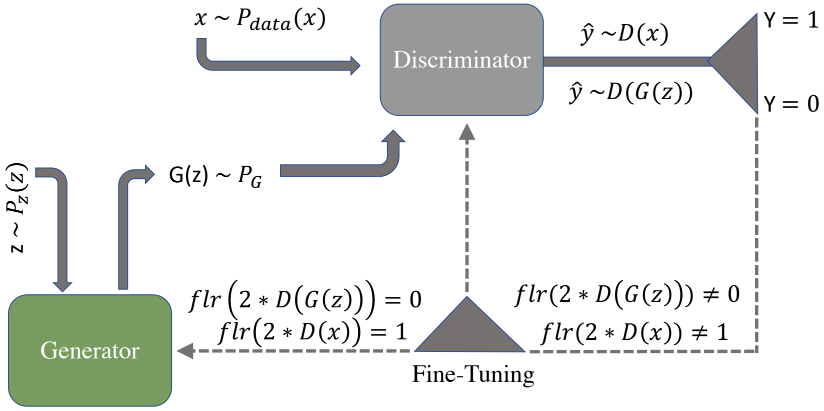


Figure 2: Generative Adversarial Network (**GAN**).

Generative Adversarial Network (GAN) Loss Function

Cross-Entropy is a measure from the field of information theory which calculate the difference between two probability distributions.

$$L(\hat{y}, y) = [ylog(\hat{y}) + (1 - y)log(1 - \hat{y})] \tag{1}$$

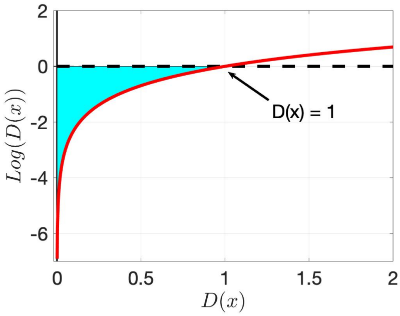
where $\hat{y}$ is the reconstructed labels like $D(x)$ or $D(G(z))$ and y is the original one. The label for the data coming from $P_{data}(x)$ is $y = 1$ and the reconstructing label after passing to the discriminator is $\hat{y} = D(x)$. Based on equation (1) we have the following:

$$L(D(x), 1) = log(D(x)) \tag{2}$$

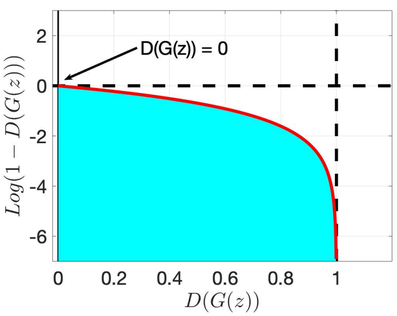
Likewise the data coming from the generator has the real label $y = 0$ and the reconstructing label is $\hat{y} = D(G(z))$ which gives us the following:

$$L(D(G(z)), 0) = log(1 - D(G(z))) \tag{3}$$

Generative Adversarial Network (GAN) Loss Function



(a)



(b)

Figure 3: Panel a and b representing the equation 1 and 2 respectively. The cyan color is the possible region, as we are working with the probability (i.e. $D(x)$ represents the probability that x came from the data rather than P_g and $D(G(z))$ the probability that $G(z)$ came from P_g rather than P_{data}).

Generative Adversarial Network (GAN) Loss Function

Discriminator role: To discriminate between the fake samples (the one which is coming from generator $(D(G(z)) = 0)$ and the real one $(D(x) = 1)$. Based on Figure 3, in order to achieve this, discriminator should maximize $log(D(x))$ and $log(1 - D(G(z)))$. Thus, the maximization term for the discriminator is like the following:

$$\underset{D}{max}\,\{logD(x) + log(1 - D(G(z)))\} \tag{4}$$

Generator role: The goal of the generator is to fool the discriminator. Thus, generator wants $D(G(z)) = 1$, in that case discriminator cannot recognize the sample coming from generator and by mistake giving $D(G(z)) = 1$, means the sample is coming from the real data and it is not correct.

$$\underset{G}{min}\,\{log(1 - D(G(z)))\} = \underset{G}{min}\,\{logD(x) + log(1 - D(G(z)))\} \tag{5}$$

Generative Adversarial Network (GAN) Loss Function

To sum up,

- We train D to maximize the probability of assigning the correct label to both training examples and samples from G.
- We simultaneously train G to minimize $log(1 - D(G(z)))$
- D and G play the following two-player minimax game with value function $V(G, D)$:

$$\underset{G}{min}\,\underset{D}{max}\,V(D,G) \tag{6}$$

Where

$$V(D,G) = \mathbb{E}_{x \sim P_{data}(x)}[logD(x)] + \mathbb{E}_{z \sim P_z(z)}[log(1 - D(G(z)))]$$

Global Optimality of $P_g = P_{data}$

We first consider the optimal discriminator D for any given generator G.
Proposition: For G fixed, the optimal discriminator D is:

$$D^*_G(x) = \frac{P_{data}(x)}{P_{data}(x) + P_g(x)} \tag{7}$$

$$D^*_G = \underset{D}{argmax} \left\{ \mathbb{E}_{x \sim P_{data}(x)}[logD(x)] + \mathbb{E}_{z \sim P_z(z)}[log(1 - D(G(z)))] \right\}$$

We know that $\mathbb{E}_{P(x)}(x) = \int_x x P_x(x)\, dx$ by substituting into above formula we have:

$$V(D, G) = \int_x P_{data}(x) log(D(x)) dx + \int_z P_z(z) log(1 - D(G(z))) dz \tag{8}$$

Global Optimality of $P_g = P_{data}$

If the probability density function of a random variable x is given as $P_x(x)$, it is possible to calculate the probability density function of some variable $Y = G(x)$. This **change of variables** is defined as:

$$P_Y(y) = P_x(G^{-1}(y))\,|\frac{d}{dy}(G^{-1}(y))|$$

Therefore,

$$\int_z P_z(z) log(1 - D(G(z)))dz = \int_x P_z(G^{-1}(x))\frac{dG^{-1}(x)}{dx} log(1 - D(x))dx$$

which is equal to the following:

$$\int_z P_z(z) log(1 - D(G(z)))dz = \int_x P_g(x) log(1 - D(x))dx$$

where:

$$P_g(x) = P_z(G^{-1}(x))\frac{dG^{-1}(x)}{dx}$$

Global Optimality of $P_g = P_{data}$

We have shown so far that:

$$\begin{aligned} V(D,G) &= \int_x P_{data}(x) log(D(x)) dx + \int_z P_z(z) log(1 - D(G(z)))\, dz \\ &= \int_x \left[P_{data}(x) log(D(x)) + P_g(x) log(1 - D(x))\right] dx \end{aligned}$$

The optimal value for discriminator (D^*) is obtained by maximizing the bracket. Thus, by taking derivative with respect to $D(x)$ and setting it equal zero we have:

$$\frac{d}{dD(x)}\left[P_{data}(x) log(D(x)) + P_g(x) log(1 - D(x))\right] = 0$$

$$\left[\frac{P_{data}(x)}{D(x)} - \frac{P_g(x)}{1 - D(x)}\right] = 0 \implies D^*_G(x) = \frac{P_{data(x)}}{P_{data(x)} + P_{g(x)}}$$

Global Optimality of $P_g = P_{data}$

Now in order to see if this value is a maximum or minimum we need to take a second derivative.

$$\frac{d}{dx}\left[\frac{P_{data}(x)}{D(x)} - \frac{P_g(x)}{1 - D(x)}\right] = -\left(\frac{P_{data}(x)}{D^2(x)} + \frac{P_g(x)}{(1 - D(x))^2}\right) < 0$$

This value helps us to know what would be the optimal value for the generator. Since the role of the generator is the reverse of the discriminator, by fixing $D = D_G^*$ the optimal G^* is like the following:

$$G^* = \underset{G}{argmin}\, V(D_G^*, G)$$

At this point we want to prove the optimization problem has a unique solution G^* which satisfies $P_g = P_{data}$.

Global Optimality of $P_g = P_{data}$

Theorem: The global minimum of the criterion $G^* = \underset{G}{argmin} V(D_G^*, G)$ is achieved if and only if $P_g = P_{data}$. At that point G^* value is $-log4$.

Proof:

By substituting the optimal value of the D_G^* in $G^* = \underset{G}{argmin} V(D_G^*, G)$ we obtain:

$$V(D_G^*, G) = argmin \left\{ \int_x [P_{data}(x) log(D_G^*(x)) + P_g(x) log(1 - D_G^*(x))] \, dx \right\} =$$

$$argmin \left\{ \int_x [P_{data}(x) log(\frac{P_{data}(x)}{P_{data}(x) + P_g(x)}) + P_g(x) log(\frac{P_g(x)}{P_{data}(x) + P_g(x)})] \, dx \right\}$$

By adding and subtracting $(log2)P_{data(x)}$ and $(log2)P_g(x)$ in the above equation we obtain:

Global Optimality of $P_g = P_{data}$

$$= argmin \left\{ \int_x [(log2 - log2)P_{data}(x) + P_{data}(x)log(\frac{P_{data}(x)}{P_{data}(x) + P_g(x)})+ \right.$$

$$\left. (log2 - log2)P_g(x) + P_g(x)log(\frac{P_g(x)}{P_{data}(x) + P_g(x)})]\, dx \right\}$$

$$= argmin \left\{ (-log2) \int_x (P_g(x) + P_{data}(x))dx + \int_x P_{data}(x)[log2+ \right.$$

$$\left. log(\frac{P_{data}(x)}{P_{data}(x) + P_g(x)})]dx + \int_x P_g(x)[log2 + log(\frac{P_g(x)}{P_{data}(x) + P_g(x)})]\, dx \right\}$$

$$= argmin \left\{ (-log4) + \int_x P_{data}(x)[log(\frac{P_{data}(x)}{\frac{P_{data}(x)+P_g(x)}{2}})]dx+ \right.$$

$$\left. \int_x P_g(x)[log(\frac{P_g(x)}{\frac{P_{data}(x)+P_g(x)}{2}})]\, dx \right\}$$

Global Optimality of $P_g = P_{data}$

Based on Kullback-Leibler Divergence and Jensen Shanon Divergence formula:

$$KL(p||q) = \int p(x) log(\frac{p(x)}{q(x)}) dx$$

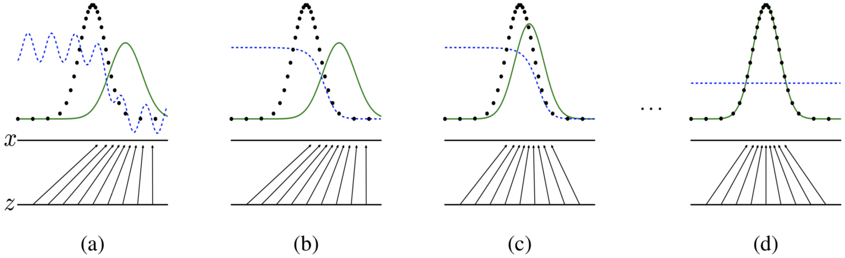
$$JSD(p||q) = \frac{1}{2}\left\{KL(p||M) + KL(q||M)\right\}, \quad M = \frac{p+q}{2}$$

We have:

$$V(D_G^*, G) = argmin\left\{-log4 + 2\,JSD(P_{data}(x)||P_g(x))\right\}$$

- $JSD = 0 \quad \leftrightarrow \mathsf{P}_g(x) = P_{data}(x)$.
- $JSD = 0 \rightarrow \mathsf{G}^* = -log4$.
- $P_g(x) = P_{data}(x) \rightarrow \mathsf{D}_G^* = \frac{1}{2}$.
- $D_G^* = \frac{1}{2} \rightarrow \mathsf{G}^* = -log4$.

Pedagogical explanation of the GAN approach



- (a) Adversarial pair near convergence: P_g is similar to P_{data} and D is a partially accurate classifier.
- (b) In the inner loop of the algorithm, D is trained to discriminate samples from data, converging to $D_G^*(x) = \frac{P_{data(x)}}{P_{data}(x)+P_g(x)}$.
- (c) After an update to G, gradient of D has guided $G(z)$ to flow to regions that are more likely to be classified as data.
- (d) $P_g(x) = P_{data}(x)$, i.e. $D_G^*(x) = \frac{1}{2}$.

conditional Generative Adversarial Networks (cGANs)

GAN has some drawbacks, one of which is having no control over the output. We can fix this issue by passing a conditional label to the objective function in a conditional GAN (cGAN).

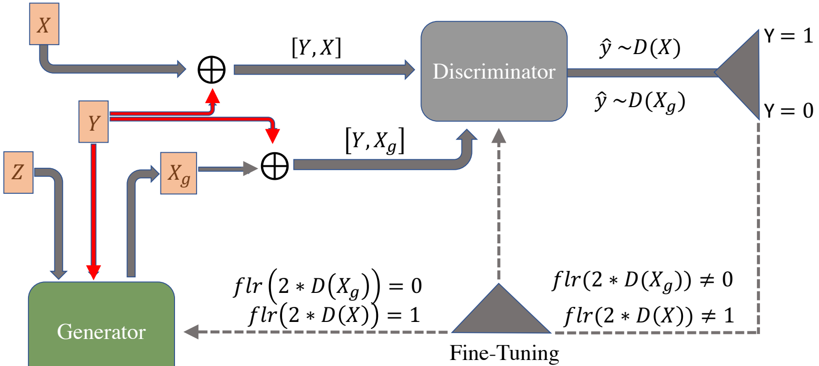


Figure 4: conditional Generative Adversarial Network (cGAN).

conditional Generative Adversarial Network (cGAN)

In general we have the following minimax problem:

$$\underset{G}{min}\,\underset{D}{max}\,V(D,G)$$

Where $V(D,G)$ for the GAN is:

$$V(D,G) = \mathbb{E}_{x\sim P_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim P_z(z)}[log(1-D(G(z)))]$$

While for the cGAN we need to insert a conditional label y to above objective function.

$$V(D,G) = \mathbb{E}_{x\sim P_{data}(x)}[logD(x|y)] + \mathbb{E}_{z\sim P_z(z)}[log(1-D(G(z|y)))]$$

Rosenbrock test function of two variables

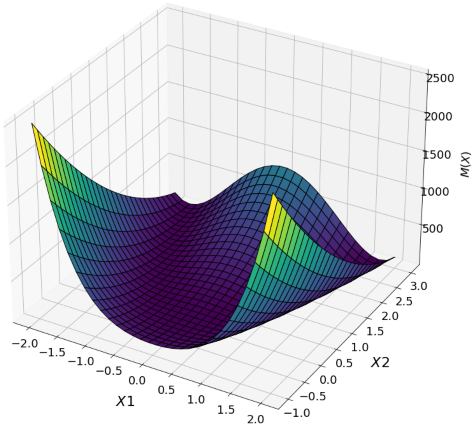


Figure 5: The mechanistic model was represented by the **Rosenbrock function** with two input parameters x_1 and x_2: $M(X) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$, where $X = (x_1, x_2)$.

- The cGAN is trained on distributions of $x \sim q_X$ and $y = M(x)$, where q_X is the prior distribution of the parameters.
- Then we train the c-GAN on this generated data.
- Here, we assume that the observation P_Y is a truncated Gaussian with the mean at 350.
- We sample parameters of the model $y \sim P_Y$ and feed them to the sampling network to obtain the final distribution P_X in X.
- We need to check if the distribution of parameters P_X is coherent to the distribution of observations P_Y .
- We pass parameters $x \sim P_X$ through the original model M and compare the resulting distribution against the observations P_Y .

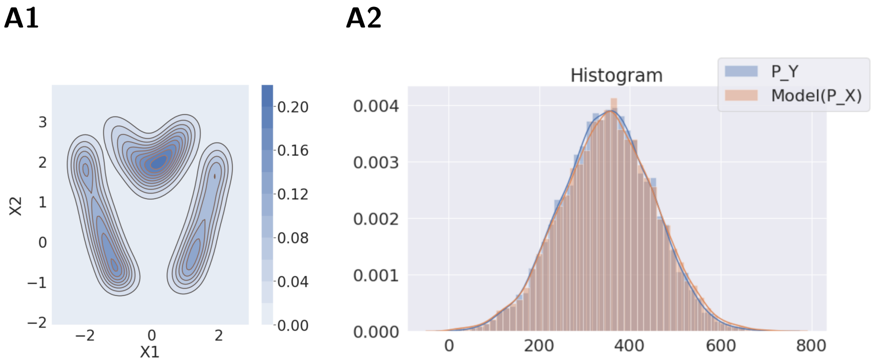


Figure 6: Panel (A1): Final distribution of the model P_X which is provided by passing the sample parameter of the model $y \sim P_Y$ to the sampling network. Panel (A2): P_X is coherent with the observation's parameters (P_Y).

Conclusion and Future work

- Generative Adversarial Network (GAN) has received attention from a wide range of fields, including image processing, science, cryptography, video games, and others.
- conditional Generative Adversarial Network (cGAN) is a novel parameter inference technique.
- If the test or target vector which is passed to the generator lies in the area that cGAN trained on, the final distribution of the generator is able to produce some samples which are very close to the original one in the parameter space.
- However, the cGAN may fail if the target features do not follow the same pattern which the cGAN learned on. Although the generator tries to get as close as possible to the features by producing some samples, there is no guarantee it will reproduce the same features.

References

Goodfellow, Ian and Pouget-Abadie, Jean and Mirza, Mehdi and Xu, Bing and Warde-Farley, David and Ozair, Sherjil and Courville, Aaron and Bengio, Yoshua,“Generative adversarial nets” *Advances in neural information processing systems*, Vol. 27, pp. 2672–2680, 2014.

Jaimit Parikh and James Kozloski and Viatcheslav Gurev, ”Integration of AI and mechanistic modeling in generative adversarial networks for stochastic inverse problems” *arXiv*, 2020.

Thanks for your attention