Impoving GAN inference with EBM model

Vasiliy Viskov, Bogdan Alexandrov, Nikita Sukhorukov,
Nikolay Ivanov, Mark Nuzhnov
Skolkovo Institute of Science and Technology, Skolkovo, Russia

Skoltech
Skolkovo Institute of Science and Technology

Machine Learning, 2023

23.03.2023

Skoltech
Skolkovo Institute of Science and Technology

Outline

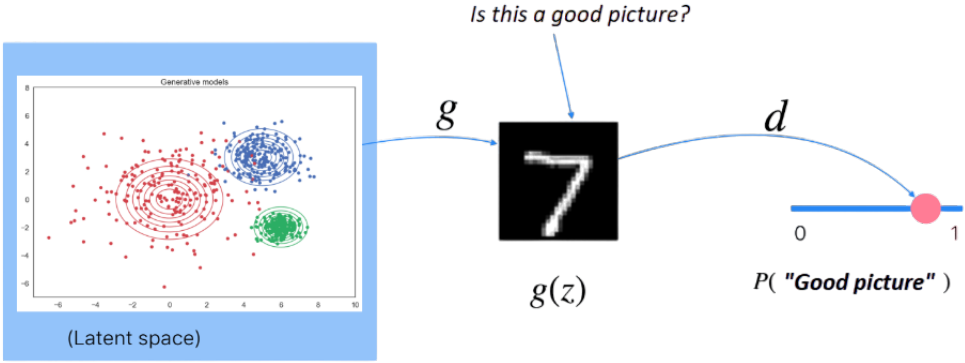
Skoltech
Skolkovo Institute of Science and Technology

Introduction

- Generative Adversarial Networks (GANs) are a type of deep learning model used for generating synthetic data.
- They consist of two parts: a generator that produces fake data and a discriminator that tries to distinguish real from fake data.
- The generator tries to fool the discriminator by generating data that is similar to real data, while the discriminator tries to correctly identify the fake data.
- GANs have shown impressive results in a variety of applications, including image and speech synthesis, data augmentation, and anomaly detection.

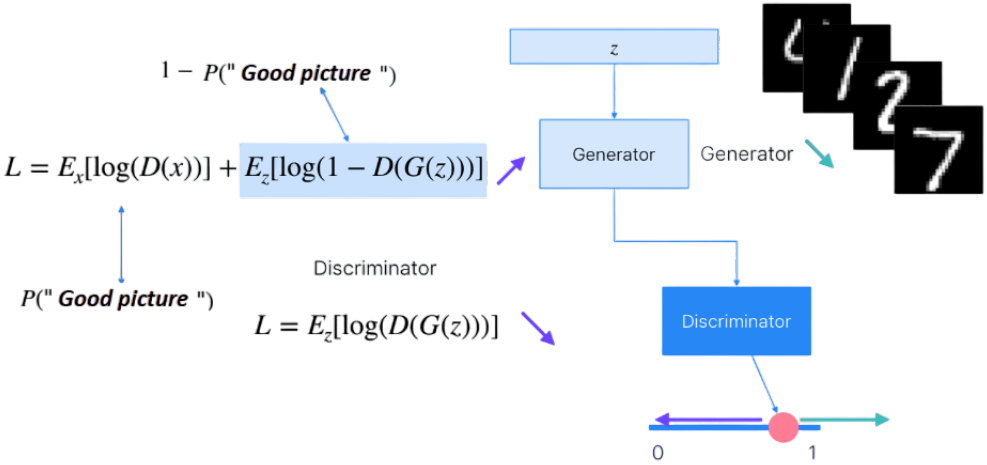
Skoltech
Skolkovo Institute of Science and Technology

GAN pipeline



Skoltech
Skolkovo Institute of Science and Technology

Loss function



Training process

Train the discriminator

1. Generate an fake image
2. Concatenate fake image and an image from the batch. Assign labels: 0's and 1's
3. Calculate BCE and update weights discriminator (do not touch the generator)

Train the generator

1. Generate a new fake image and invert it's label from 0 to 1
2. Calculate BCE and update generator weights (do not touch the discriminator)

Skoltech
Skolkovo Institute of Science and Technology

Total optimization problem

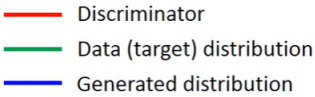
$$\min_G \max_D V(D, G) = \min_G \max_D (\mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]) \quad (1)$$

Here, G represents the generator, D represents the discriminator, p_{data} is the distribution of real data, p_z is the distribution of noise input, and x and z are the input samples. The objective function tries to maximize the log probability of the discriminator being correct on real data and fake data, respectively.

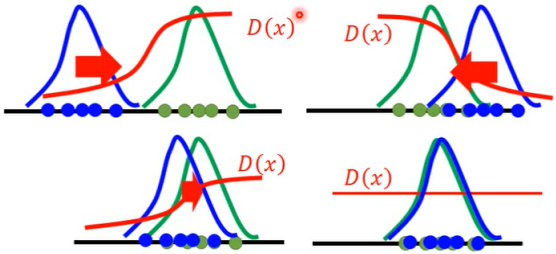
Skoltech
Skolkovo Institute of Science and Technology

Distribution restoration

Original Idea



- Discriminator leads the generator



Skoltech
Skolkovo Institute of Science and Technology

Introduction to EBM

- Energy-based models are a class of probabilistic models that use energy functions to measure the compatibility between input data and model parameters.
- In contrast to GANs, which rely on adversarial training, EBMs use a different training paradigm called contrastive divergence.
- This involves repeatedly sampling from the model's energy function and updating the parameters to reduce the difference between the model's distribution and the true distribution of the data.

Skoltech
Skolkovo Institute of Science and Technology

The fundamental idea of energy-based models is to turn any function that predicts values larger than zero into a probability distribution by dividing by its volume. Imagine we have a neural network, which has as output a single neuron, like in regression. We can call this network $E_\theta(x)$, where θ are our parameters of the network, and x the input data (e.g. an image). The output of $E_\theta(x)$ is a scalar value between $-\infty$ and ∞. Now, we can use basic probability theory to normalize the scores of all possible inputs:

$$q_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta} \quad \text{where} \quad Z_\theta = \begin{cases} \int_{\mathbf{x}} \exp(-E_\theta(\mathbf{x})), d\mathbf{x} & \text{if } \mathbf{x} \text{ is continuous} \\ \sum_{\mathbf{x}} \exp(-E_\theta(\mathbf{x})) & \text{if } \mathbf{x} \text{ is discrete} \end{cases} \tag{2}$$

Skoltech
Skolkovo Institute of Science and Technology

The *exp*-function ensures that we assign a probability greater than zero to any possible input. We use a negative sign in front of E because we call E_θ to be the energy function: data points with high likelihood have a low energy, while data points with low likelihood have a high energy. Z_θ is our normalization terms that ensures that the density integrates/sums to 1. We can show this by integrating over $q_\theta(x)$:

$$\int_{\mathbf{x}} q_\theta(\mathbf{x}) d\mathbf{x} = \int_{\mathbf{x}} \frac{\exp\left(-E_\theta(\mathbf{x})\right)}{\int_{\tilde{\mathbf{x}}} \exp\left(-E_\theta(\tilde{\mathbf{x}})\right) d\tilde{\mathbf{x}}} d\mathbf{x} = \frac{\int_{\mathbf{x}} \exp\left(-E_\theta(\mathbf{x})\right) d\mathbf{x}}{\int_{\tilde{\mathbf{x}}} \exp\left(-E_\theta(\tilde{\mathbf{x}})\right) d\tilde{\mathbf{x}}} = 1 \tag{3}$$

Note that we call the probability distribution $q_\theta(x)$ because this is the learned distribution by the model, and is trained to be as close as possible to the true, unknown distribution $p(x)$.

Skoltech
Skolkovo Institute of Science and Technology

How to calculate?

There is no chance that we can calculate Z_θ analytically for high-dimensional input and/or larger neural networks, but the task requires us to know Z_θ. Although we can't determine the exact likelihood of a point, there exist methods with which we can train energy-based models. Thus, we will look next at "Contrastive Divergence" for training the model.

Skoltech
Skolkovo Institute of Science and Technology

Contrastive Divergence

When we train a generative model, it is usually done by maximum likelihood estimation. We cannot just maximize the un-normalized probability $exp(-E_\theta(x_{train})$ because there is no guarantee that Z_θ stays constant, or that x_{train} is becoming more likely than the others. However, if we base our training on comparing the likelihood of points, we can create a stable objective. Namely, we can re-write our maximum likelihood objective where we maximize the probability of x_{train} compared to a randomly sampled data point of our model:

$$\nabla_\theta \mathcal{L}\text{MLE}(\theta; p) = -\mathbb{E}_{p(\mathbf{x})}\left[\nabla_\theta \log q_\theta(\mathbf{x})\right] = \mathbb{E}_{p(\mathbf{x})}\left[\nabla_\theta E_\theta(\mathbf{x})\right] - \mathbb{E}_{q\theta(\mathbf{x})}\left[\nabla_\theta E_\theta(\mathbf{x})\right] (4)$$

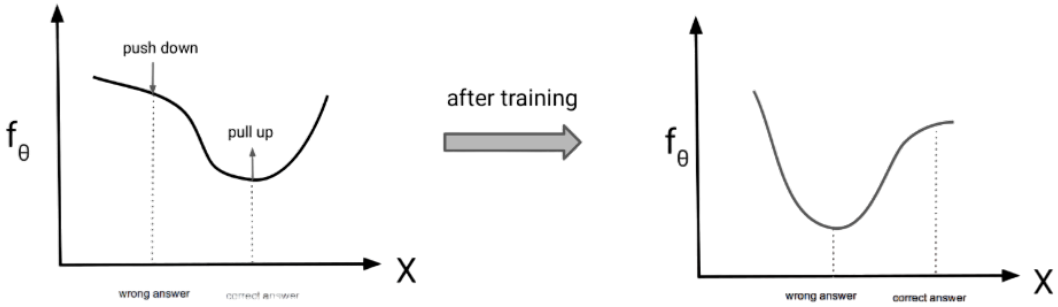
Skoltech
Skolkovo Institute of Science and Technology

Contrastive Divergence(2)

Note that the loss is still an objective we want to minimize. Thus, we try to minimize the energy for data points from the dataset, while maximizing the energy for randomly sampled data points from our model (how we sample will be explained below). Although this objective sounds intuitive, how is it actually derived from our original distribution $q_\theta(x)$? The trick is that we approximate Z_θ by a single Monte-Carlo sample. This gives us the exact same objective as written above. Visually, we can look at the objective as follows

Skoltech
Skolkovo Institute of Science and Technology

Visuall explanation



f_θ represents $exp(-E_\theta(x))$ in our case. The point on the right, called "correct answer", represents a data point from the dataset (i.e. x_{train}), and the left point, "wrong answer", a sample from our model (i.e. x_{sample}). Thus, we try to "pull up" the probability of the data points in the dataset, while "pushing down" randomly sampled points. The two forces for pulling and pushing are in balance iff $q_\theta(x) = p(x)$

Skoltech
Skolkovo Institute of Science and Technology

How to sample?

For sampling from an energy-based model, we can apply a Markov Chain Monte Carlo using Langevin Dynamics. The idea of the algorithm is to start from a random point, and slowly move towards the direction of higher probability using the gradients of E_θ. Nevertheless, this is not enough to fully capture the probability distribution. We need to add noise ω at each gradient step to the current sample. Under certain conditions such as that we perform the gradient steps an infinite amount of times, we would be able to create an exact sample from our modeled distribution. However, as this is not practically possible, we usually limit the chain to K steps (K is a hyperparameter that needs to be finetuned).

Skoltech
Skolkovo Institute of Science and Technology

Sampling algorithm

Algorithm 1 Sampling from an energy-based model

1: Sample $\tilde{\boldsymbol{x}}^0$ from a Gaussian or uniform distribution;
2: **for** sample step $k = 1$ to K **do** ▷ Generate sample via Langevin dynamics
3: $\quad \tilde{\boldsymbol{x}}^k \leftarrow \tilde{\boldsymbol{x}}^{k-1} - \eta \nabla_{\boldsymbol{x}} E_\theta(\tilde{\boldsymbol{x}}^{k-1}) + \omega$, where $\omega \sim \mathcal{N}(0, \sigma)$
4: **end for**
5: $\boldsymbol{x}_{\text{sample}} \leftarrow \tilde{\boldsymbol{x}}^K$

Skoltech
Skolkovo Institute of Science and Technology

Training algorithm

Algorithm 2 Training an energy-based model for generative image modeling

Initialize empty buffer $B \leftarrow \emptyset$
while not converged **do**
  Sample data from dataset: $\boldsymbol{x}_i^+ \sim p_{\mathcal{D}}$
  Sample initial fake data: $\boldsymbol{x}_i^0 \sim B$ with 95% probability, else $\mathcal{U}(-1, 1)$
  for sample step $k = 1$ to K **do** ▷ Generate sample via Langevin dynamics
    $\tilde{\boldsymbol{x}}^k \leftarrow \tilde{\boldsymbol{x}}^{k-1} - \eta \nabla_{\boldsymbol{x}} E_\theta(\tilde{\boldsymbol{x}}^{k-1}) + \omega$, where $\omega \sim \mathcal{N}(0, \sigma)$
  end for
  $\boldsymbol{x}^- \leftarrow \Omega(\tilde{\boldsymbol{x}}^K)$ ▷ Ω: Stop gradients operator
  Contrastive divergence: $\mathcal{L}_{CD} = 1/N \sum_i E_\theta(\boldsymbol{x}_i^+) - E_\theta(\boldsymbol{x}_i^-)$
  Regularization loss: $\mathcal{L}_{RG} = 1/N \sum_i E_\theta(\boldsymbol{x}_i^+)^2 + E_\theta(\boldsymbol{x}_i^-)^2$
  Perform SGD/Adam on $\nabla_\theta(\mathcal{L}_{CD} + \alpha \mathcal{L}_{RG})$
  Add samples to buffer: $B \leftarrow B \cup \boldsymbol{x}^-$
end while

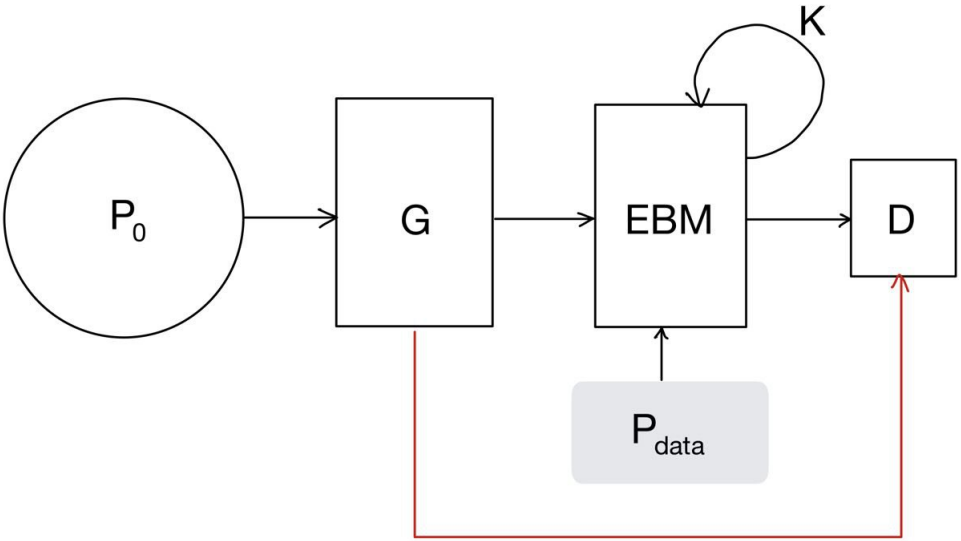
Skoltech
Skolkovo Institute of Science and Technology

GAN with EBM: training and generative inference

- Training:
  1. Fit D replacing p_{data} with K steps of MCMC sampling from $G(z)$, $z \sim \mathcal{N}(0, 1)$
  2. Fit G as in GAN training pipeline
  3. Fit EBM model replacing fake data with $G(z)$, $z \sim \mathcal{N}(0, 1)$
- Inference
  1. Sample $z \sim \mathcal{N}(0, 1)$, apply K steps of MCMC sampling to $G(z)$

Skoltech
Skolkovo Institute of Science and Technology

GAN with EBM: pipeline

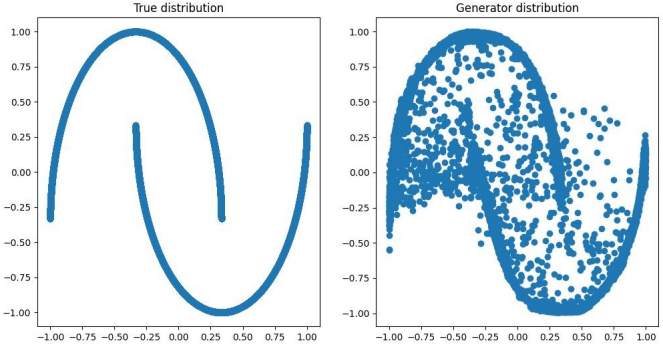


Motivation

- Stabilization of GAN learning
- After learning our model we get not only sampler (generator) from learned distribution, but it PDF(unnormalized)

Skoltech
Skolkovo Institute of Science and Technology

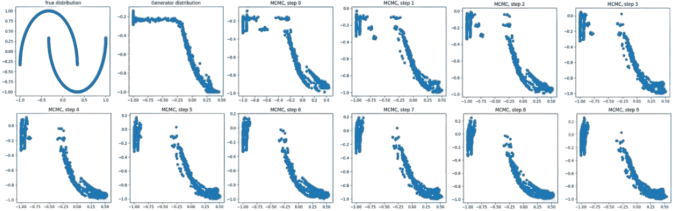
Experiment (0)



- Make Moons from Scikit-learn
- Vanilla GAN

Skoltech
Skolkovo Institute of Science and Technology

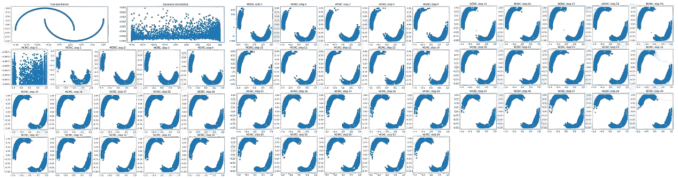
Experiment (1)



- Make Moons from Scikit-learn
- Vanilla GAN
- EBM
- Big step size

Skoltech
Skolkovo Institute of Science and Technology

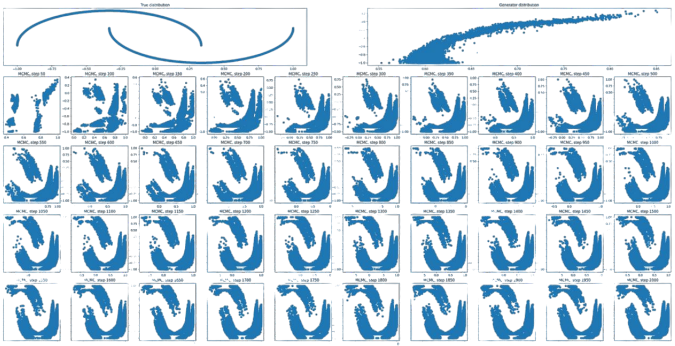
Experiment (2)



- Make Moons from Scikit-learn
- Vanilla GAN
- EBM
- Small step size

Skoltech
Skolkovo Institute of Science and Technology

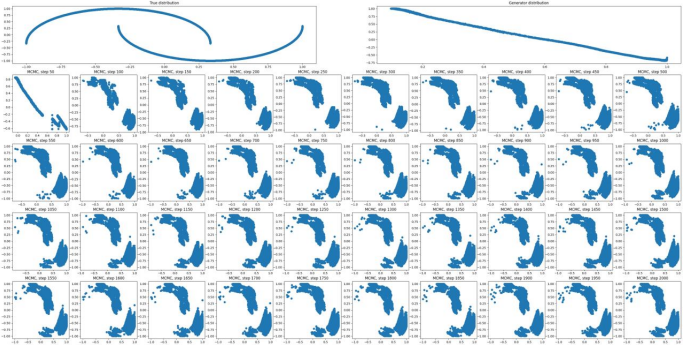
Experiment (3)



- Make Moons from Scikit-learn
- Vanilla GAN
- EBM
- Small step size

Skoltech
Skolkovo Institute of Science and Technology

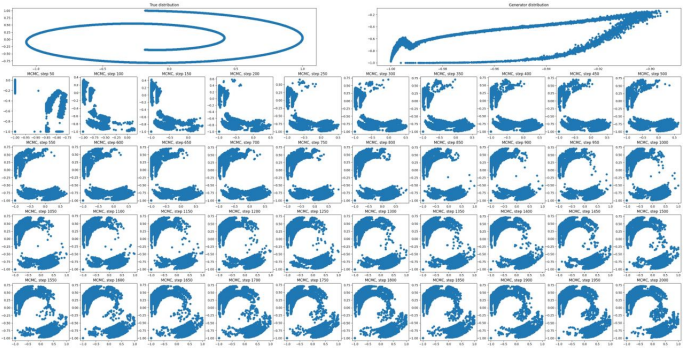
Experiment (4)



- Make Moons from Scikit-learn
- Vanilla GAN
- EBM
- Large step size

Skoltech
Skolkovo Institute of Science and Technology

Experiment (5)



- Swiss roll
- Vanilla GAN
- EBM

Skoltech
Skolkovo Institute of Science and Technology

Experiment (6)

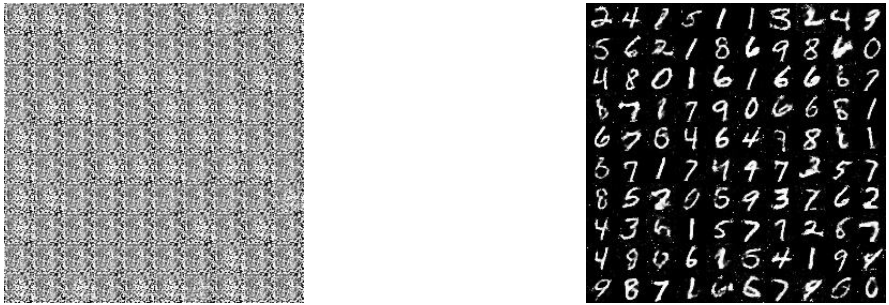


Figure: Comparison of Vanilla GAN and Vanilla GAN with EBM

Skoltech
Skolkovo Institute of Science and Technology

Conclusions

- We learned how GANs work
- We learned how EBMs work
- We implemented simultaneously GAN and EBM for 2D datasets (Make Moons, Swiss Roll) and MNIST
- We compared different step sizes, MCMC step count and model architechtures with their parameters

Skoltech
Skolkovo Institute of Science and Technology

Future work

- Use another types of GAN (Conditional, infoGAN, etc.)
- Think of using various heuristics as a playout algorithm

Skoltech
Skolkovo Institute of Science and Technology

Team contributions

- Vasiliy Viskov - project mastermind and code guru
- Bogdan Alexandrov - theory, code, presentation
- Nikita Sukhorukov - theory, code, presentation
- Nikolay Ivanov - theory, code, presentation
- Mark Nuzhnov - theory, code, presentation

Skoltech
Skolkovo Institute of Science and Technology

Bibliography

- ECCV2022. Learning Energy-Based Models With Adversarial Training. Xuwang Yin, Shiying Li, Gustavo K. Rohde
- GAN-Tutorial
- [Goodfellow et. al., 2014] "Generative Adversarial Nets"
- [Nijkamp et. al., 2019] "On the Anatomy of MCMC-Based Maximum Likelihood Learning of Energy-Based Models"

Skoltech
Skolkovo Institute of Science and Technology

Thank you for attention!
Questions? Link to Github

Skoltech
Skolkovo Institute of Science and Technology