Table of Contents

1 Taxonomy of Generative Models

2 Generative Adversarial Networks
   - GANs Structure
   - Objective & Training
   - Conclusions
Table of Contents

1 Taxonomy of Generative Models

2 Generative Adversarial Networks
   - GANs Structure
   - Objective & Training
   - Conclusions
Last tutorial, we talked about variational inference and presented the variational autoencoder.
Generative Model with Implicit Density

- Last tutorial, we talked about variational inference and presented the variational autoencoder.
- The VAE model maximizes a lower bound on the log likelihood of the training data, where we assumed it can be modeled by some known parametric distribution conditioned on a latent random variable.
Generative Model with Implicit Density

- Last tutorial, we talked about variational inference and presented the variational autoencoder.
- The VAE model maximizes a lower bound on the log likelihood of the training data, where we assumed it can be modeled by some known parametric distribution conditioned on a latent random variable.
- This process is actually a form of maximum-likelihood estimation, where we approximate the intractable posterior distribution \( q(z|x) \) by using a neural network. We still need to define a density model explicitly.
Last tutorial, we talked about variational inference and presented the variational autoencoder.

The VAE model maximizes a lower bound on the log likelihood of the training data, where we assumed it can be modeled by some known parametric distribution conditioned on a latent random variable.

This process is actually a form of maximum-likelihood estimation, where we approximate the intractable posterior distribution \( q(z|x) \) – by using a neural network. We still need to define a density model explicitly.

In this tutorial, we will present a framework which does implicit density estimation.
Taxonomy of Generative Models

Generative Adversarial Networks

- Maximum Likelihood
  - Explicit density
    - Tractable density
      - Fully visible belief nets
      - NADE
      - MADE
      - PixelRNN
      - Change of variables models (nonlinear ICA)
  - Approximate density
    - Variational autoencoder
    - Boltzmann machine

- Implicit density
  - Markov Chain
    - GSN
Table of Contents

1. Taxonomy of Generative Models

2. Generative Adversarial Networks
   - GANs Structure
   - Objective & Training
   - Conclusions
Generative Adversarial Networks Structure

- Assume we have a large sample from some distribution $p_{\text{data}}$, and we wish to build a model that will generate samples according to this distribution.
Generative Adversarial Networks Structure

- Assume we have a large sample from some distribution $p_{data}$, and we wish to build a model that will generate samples according to this distribution.

- A Generative Adversarial Network (Goodfellow et al., 2014) consists of two components: a **Generator**, denoted $G$, and a **Discriminator**, denoted $D$. Both components are deep neural networks, parametrized by $\theta_G, \theta_D$, respectively.
Assume we have a large sample from some distribution \( p_{\text{data}} \), and we wish to build a model that will generate samples according to this distribution.

A Generative Adversarial Network (Goodfellow et al., 2014) consists of two components: a **Generator**, denoted \( G \), and a **Discriminator**, denoted \( D \). Both components are deep neural networks, parametrized by \( \theta_G, \theta_D \), respectively.

The generator tries to generate instances resembling those sampled from \( p_{\text{data}} \), and the discriminator tries to distinguish the “fake” instances from the “real” ones.
Assume we have a large sample from some distribution \( p_{\text{data}} \), and we wish to build a model that will generate samples according to this distribution.

A Generative Adversarial Network (Goodfellow et al., 2014) consists of two components: a **Generator**, denoted \( G \), and a **Discriminator**, denoted \( D \). Both components are deep neural networks, parametrized by \( \theta_G, \theta_D \), respectively.

- The generator tries to generate instances resembling those sampled from \( p_{\text{data}} \), and the discriminator tries to distinguish the “fake” instances from the “real” ones.
- The generator’s sampling process in done by first sampling from some prior distribution \( p_z \), and then mapping the sampled vector to the space of the modeled data.
**Minimax Game**

- The GAN objective can be seen as a minimax game between two players, and is formulated as:

\[
\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}} \left[ \log D(x) \right] + \mathbb{E}_{z \sim p_z} \left[ \log (1 - D(G(z))) \right]
\]
Minimax Game

- The GAN objective can be seen as a minimax game between two players, and is formulated as:

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

- The discriminator’s objective is minimizing the cross-entropy loss of a binary classifier trying to separate “real” samples drawn from the true distribution \( p_{\text{data}} \) from synthetic samples drawn by the generator.
Minimax Game

- The GAN objective can be seen as a minimax game between two players, and is formulated as:

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

- The discriminator’s objective is minimizing the cross-entropy loss of a binary classifier trying to separate “real” samples drawn from the true distribution \( p_{\text{data}} \) from synthetic samples drawn by the generator.

- The generator’s objective is the exact opposite – generate samples that are indistinguishable from the real ones by the discriminator.
Training

- In contrast to optimization objectives we have seen so far, here we are interested in an equilibrium point. Finding this point is extremely challenging.
Training

- In contrast to optimization objectives we have seen so far, here we are interested in an **equilibrium** point. Finding this point is extremely challenging.

- One simple heuristic is iterative optimization, in which we train each model separately for one gradient descent step, and switch to the other.
Training

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used k = 1, the least expensive option, in our experiments.

for number of training iterations do
    for k steps do
        • Sample minibatch of m noise samples \(\{z^{(1)}, \ldots, z^{(m)}\}\) from noise prior \(p_g(z)\).
        • Sample minibatch of m examples \(\{x^{(1)}, \ldots, x^{(m)}\}\) from data generating distribution \(p_{data}(x)\).
        • Update the discriminator by ascending its stochastic gradient:
          \[
          \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\left(x^{(i)}\right) + \log \left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \right].
          \n          \]
    end for
    • Sample minibatch of m noise samples \(\{z^{(1)}, \ldots, z^{(m)}\}\) from noise prior \(p_g(z)\).
    • Update the generator by descending its stochastic gradient:
      \[
      \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D\left(G\left(z^{(i)}\right)\right)\right).
      \]
end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.
Training

- In practice, at the beginning of the training process, the discriminator can distinguish between “fake” and “real” samples confidently, leading to gradients that are too small for the generator to train from.
Training

- In practice, at the beginning of the training process, the discriminator can distinguish between “fake” and “real” samples confidently, leading to gradients that are too small for the generator to train from.

- To mitigate this issue, the generator optimizes a slightly different objective:

\[
\min_G \mathbb{E}_{z \sim p_z} [- \log(D(G(z)))] = \max_G \mathbb{E}_{z \sim p_z} [\log(D(G(z)))]
\]
In practice, at the beginning of the training process, the discriminator can distinguish between “fake” and “real” samples confidently, leading to gradients that are too small for the generator to train from.

To mitigate this issue, the generator optimizes a slightly different objective:

$$\min_G \mathbb{E}_{z \sim p_z} [- \log(D(G(z)))] = \max_G \mathbb{E}_{z \sim p_z} [\log(D(G(z)))]$$

The new objective is heuristically motivated, and is equivalent to flipping the labels of the fake samples when minimizing the cross-entropy loss of discriminator.
Heuristic vs. Original

![Graph showing the comparison between heuristic and original objectives](image)
GANs Pros & Cons

- **Advantages:**
  - Generates very realistic looking images.
  - Can, theoretically, represent any distribution (especially useful for very sharp or even degenerate distributions).
  - Does not require selecting any distribution prior (the noise is almost always taken to be Gaussian).

- **Disadvantages:**
  - Highly unstable and challenging training (G doesn’t train, mode-collapse).
  - Hard to obtain good results for domains other than images.
  - Unable to perform posterior inference (i.e., $p(z|x)$).
  - Unclear how to evaluate the model (eye-balling, inception score).
GANs Pros & Cons

**Advantages:**

- Generates *very* realistic looking images.
GANs Pros & Cons

- **Advantages:**
  - Generates *very* realistic looking images.
  - Can, theoretically, represent any distribution (especially useful for very sharp or even degenerate distributions).

- **Disadvantages:**
  - Highly unstable and challenging training (G doesn’t train, mode-collapse).
  - Hard to obtain good results for domains other than images.
  - Unable to perform posterior inference (i.e., $p(z|x)$).
  - Unclear how to evaluate the model (eye-balling, inception score).
GANs Pros & Cons

- **Advantages:**
  - Generates *very* realistic looking images.
  - Can, theoretically, represent any distribution (especially useful for very sharp or even degenerate distributions).
  - Does not require selecting any distribution prior (the noise is almost always taken to be Gaussian).
GANs Pros & Cons

- **Advantages:**
  - Generates *very* realistic looking images.
  - Can, theoretically, represent any distribution (especially useful for very sharp or even degenerate distributions).
  - Does not require selecting any distribution prior (the noise is almost always taken to be Gaussian).

- **Disadvantages:**

- High stability and challenging training ($G$ doesn’t train, mode-collapse).
- Hard to obtain good results for domains other than images.
- Unable to perform posterior inference (i.e., $p(z|x)$).
- Unclear how to evaluate the model (eye-balling, inception score).
GANs Pros & Cons

- **Advantages:**
  - Generates *very* realistic looking images.
  - Can, theoretically, represent any distribution (especially useful for very sharp or even degenerate distributions).
  - Does not require selecting any distribution prior (the noise is almost always taken to be Gaussian).

- **Disadvantages:**
  - *Highly* unstable and challenging training (\(G\) doesn’t train, mode-collapse).
GANs Pros & Cons

**Advantages:**
- Generates *very* realistic looking images.
- Can, theoretically, represent any distribution (especially useful for very sharp or even degenerate distributions).
- Does not require selecting any distribution prior (the noise is almost always taken to be Gaussian).

**Disadvantages:**
- *Highly* unstable and challenging training ($G$ doesn’t train, mode-collapse).
- Hard to obtain good results for domains other than images.
GANs Pros & Cons

- **Advantages:**
  - Generates *very* realistic looking images.
  - Can, theoretically, represent any distribution (especially useful for very sharp or even degenerate distributions).
  - Does not require selecting any distribution prior (the noise is almost always taken to be Gaussian).

- **Disadvantages:**
  - *Highly* unstable and challenging training ($G$ doesn’t train, mode-collapse).
  - Hard to obtain good results for domains other than images.
  - Unable to perform posterior inference (i.e., $p(z|x)$).
GANs Pros & Cons

**Advantages:**
- Generates *very* realistic looking images.
- Can, theoretically, represent any distribution (especially useful for very sharp or even degenerate distributions).
- Does not require selecting any distribution prior (the noise is almost always taken to be Gaussian).

**Disadvantages:**
- *Highly* unstable and challenging training (\(G\) doesn’t train, mode-collapse).
- Hard to obtain good results for domains other than images.
- Unable to perform posterior inference (i.e., \(p(z|x)\)).
- Unclear how to evaluate the model (eye-balling, inception score).
Research

“Generative Adversarial Networks is the most interesting idea in the last ten years in Machine Learning.” – Yann Lecun, Facebook AI Research Director.
“Generative Adversarial Networks is the most interesting idea in the last ten years in Machine Learning.” – Yann Lecun, Facebook AI Research Director.

Introduction of GANs took the deep learning community by storm. Thousands of paper have been published in the past few years, focused on stabilizing, understanding, and improving the framework.
Research

- A few of the most notable ones:

  - DC-GAN: Convolutional based architecture which stabilized GAN training and created high-quality images.
  - Wasserstein-GAN: A new formulation of the GAN objective, leading to more stable training. The loss has an interpretable form which can be used as an evaluation metric.
  - Progressive Growing of GANs: A recent paper by NVIDIA that achieves incredibly realistic results.
A few of the most notable ones:

- **DC-GAN**: Convolutional based architecture which stabilized GAN training and created high-quality images.

- **Wasserstein-GAN**: A new formulation of the GAN objective, leading to more stable training. The loss has an interpretable form which can be used as an evaluation metric.

- **Progressive Growing of GANs**: A recent paper by NVIDIA that achieves incredibly realistic results.
A few of the most notable ones:

- **DC-GAN**: Convolutional based architecture which stabilized GAN training and created high-quality images.
- **Wasserstein-GAN**: A new formulation of the GAN objective, leading to more stable training. The loss has an interpretable form which can be used as an evaluation metric.
A few of the most notable ones:

- **DC-GAN**: Convolutional based architecture which stabilized GAN training and created high-quality images.
- **Wasserstein-GAN**: A new formulation of the GAN objective, leading to more stable training. The loss has an interpretable form which can be used as an evaluation metric.
- **Progressive Growing of GANs**: A recent paper by NVIDIA that achieves incredibly realistic results.