Generative adversarial networks

Ian Goodfellow
Jean Pouget-Abadie
Mehdi Mirza
Bing Xu
David Warde-Farley
Sherjil Ozair
Aaron Courville
Yoshua Bengio
Discriminative deep learning

• Recipe for success
Discriminative deep learning

• Recipe for success:

Google's winning entry into the ImageNet 1K competition (with extra data).
Discriminative deep learning

- Recipe for success:
  - Gradient backpropagation.
  - Dropout
  - Activation functions:
    - rectified linear
    - maxout

Google’s winning entry into the ImageNet 1K competition (with extra data).
Generative modeling

• Have training examples $x \sim p_{data}(x)$
• Want a model that can draw samples: $x \sim p_{model}(x)$
• Where $p_{model} \approx p_{data}$
Why generative models?

• Conditional generative models
  - Speech synthesis: Text $\Rightarrow$ Speech
  - Machine Translation: French $\Rightarrow$ English
    • French: Si mon tonton tond ton tonton, ton tonton sera tondu.
    • English: If my uncle shaves your uncle, your uncle will be shaved
  - Image $\Rightarrow$ Image segmentation

• Environment simulator
  - Reinforcement learning
  - Planning

• Leverage unlabeled data
Maximum likelihood: the dominant approach

- ML objective function

\[ \theta^* = \max_{\theta} \frac{1}{m} \sum_{i=1}^{m} \log p \left( x^{(i)}; \theta \right) \]
Undirected graphical models

• State-of-the-art general purpose undirected graphical model: **Deep Boltzmann machines**

• Several “hidden layers” $h$

$$p(h, x) = \frac{1}{Z} \tilde{p}(h, x)$$

$$\tilde{p}(h, x) = \exp(-E(h, x))$$

$$Z = \sum_{h,x} \tilde{p}(h, x)$$
Undirected graphical models: disadvantage

• ML Learning requires that we draw samples:

\[
\frac{d}{d\theta_i} \log p(x) = \frac{d}{d\theta_i} \left[ \log \sum_h \tilde{p}(h, x) - \log Z(\theta) \right]
\]

• Common way to do this is via MCMC (Gibbs sampling).
Boltzmann Machines: disadvantage

- Model is badly parameterized for learning high quality samples.

- Why?
  - Learning leads to large values of the model parameters.
    - Large valued parameters = peaked distribution.
  - Large valued parameters means slow mixing of sampler.
  - Slow mixing means that the gradient updates are correlated $\Rightarrow$ leads to divergence of learning.
Boltzmann Machines: disadvantage

- Model is badly parameterized for learning high quality samples.
- Why poor mixing?

- MNIST dataset
- 1st layer features (RBM)

Coordinated flipping of low-level features
**Directed graphical models**

\[ p(x, h) = p(x | h^{(1)})p(h^{(1)} | h^{(2)}) \ldots p(h^{(L-1)} | h^{(L)})p(h^{(L)}) \]

\[
\frac{d}{d\theta_i} \log p(x) = \frac{1}{p(x)} \frac{d}{d\theta_i} p(x)
\]

\[
p(x) = \sum_h p(x | h)p(h)
\]

- Two problems:
  1. Summation over exponentially many states in \( h \)
  2. Posterior inference, i.e. calculating \( p(h \mid x) \), is intractable.
Directed graphical models: New approaches

• The Variational Autoencoder model:
  - Rezende, Mohamed and Wierstra, Stochastic back-propagation and variational inference in deep latent Gaussian models. ArXiv.
  - Use a reparametrization that allows them to train very efficiently with gradient backpropagation.
Generative stochastic networks

• General strategy: Do not write a formula for $p(x)$, just learn to sample incrementally.

• Main issue: Subject to some of the same constraints on mixing as undirected graphical models.
Generative adversarial networks

• Don’t write a formula for $p(x)$, just learn to sample directly.

• No summation over all states.

• How? By playing a game.
Two-player zero-sum game

• Your winnings + your opponent's winnings = 0
• Minimax theorem: a rational strategy exists for all such finite games
Two-player zero-sum game

- Strategy: specification of which moves you make in which circumstances.
- Equilibrium: each player’s strategy is the best possible for their opponent’s strategy.
- Example: Rock-paper-scissors:
  - Mixed strategy equilibrium
  - Choose you action at random

<table>
<thead>
<tr>
<th></th>
<th>Your opponent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rock</td>
</tr>
<tr>
<td>You</td>
<td></td>
</tr>
<tr>
<td>Rock</td>
<td>0</td>
</tr>
<tr>
<td>Paper</td>
<td>1</td>
</tr>
<tr>
<td>Scissors</td>
<td>-1</td>
</tr>
</tbody>
</table>
Generative modeling with game theory?

• Can we design a game with a mixed-strategy equilibrium that forces one player to learn to generate from the data distribution?
Adversarial nets framework

• A game between two players:
  1. Discriminator D
  2. Generator G

• D tries to discriminate between:
  - A sample from the data distribution.
  - And a sample from the generator G.

• G tries to “trick” D by generating samples that are hard for D to distinguish from data.
Adversarial nets framework
Zero-sum game

• Minimax objective function:

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

• In practice, to estimate G we use:

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

Why? Stronger gradient for G when D is very good.
Discriminator strategy

- Optimal strategy for any $p_{\text{model}}(x)$ is always

$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}$$
We will show in section 4.1 that this minimax game has a global optimum for the space of probability density functions. In other words, if the two distributions, i.e. $p_{\text{data}}$ and $p_{\text{model}}$, are both $\phi$-divergence bounded, then there is a point at which both cannot improve because $\phi(p_{\text{data}}, p_{\text{model}})$ is infinite.

In the next section, we present a theoretical analysis of adversarial nets, essentially showing that Algorithm 1 optimizes Eq 1, thus obtaining the desired result. This results in the model distribution $p_{\text{model}}$ converging to the data distribution $p_{\text{data}}$.

Consider an adversarial pair near convergence: $(D, G)$ where $D$ is trained to maximize $\log D(z)$, and $G$ minimizes $\log (1 - D(z))$.

Learning process

$$D(p_{\text{data}})$$

Data distribution

Model distribution

Poorly fit model
Poorly fit model

\[ p_D(\text{data}) \]

Data distribution

Model distribution

\[ x \]

\[ z \]

Poorly fit model

After updating D

\[ \text{Learning process} \]
Poorly fit model

We will show in section 4.1 that this minimax game has a global optimum for the space of probability density functions.

In section 4.2, we will show that Algorithm 1 optimizes Eq 1, thus obtaining the desired result.

In the parametric setting, e.g. we represent a model with infinite capacity by studying convergence in the space of probability density functions.

The generator contracts in regions of high density and expands in regions of low density of the domain from which data is sampled, in this case uniformly. The horizontal line above is part of the domain of the data distribution.

After several steps of training, if the two distributions, i.e., the model distribution and the data distribution, are close enough, then both cannot improve because they are at their respective optima.

We must be able to classify data accurately, e.g. as data. (d) After several steps of training, if the two distributions are close enough, then both cannot improve because they are at their respective optima.

Consider an adversarial pair near convergence:

\[
\rho_D(\text{data}) \rightarrow \text{Data distribution} \rightarrow \text{Model distribution}
\]

\[x \rightarrow \text{Poorly fit model} \rightarrow \text{After updating D} \rightarrow \text{After updating G}\]

In the inner loop of the algorithm, we must implement the game using an iterative, numerical approach. Optimizing equation 1 may not provide sufficient gradient for convergence in the non-parametric limit. See Figure 1 for a less formal, more pedagogical explanation of the approach. In practice, we must implement the game using an iterative, numerical approach.
Learning process

Consider an adversarial pair near convergence:

\[ \min_{D} \max_{G} V(D, G) \]

\[ V(D, G) = \mathbb{E}_{x \sim \rho_D^{data}} \log(D(x)) + \mathbb{E}_{z \sim \rho_Z} \log(1 - D(G(z))) \]

The generator \( G \) contracts in regions of high density and expands in regions of low density of \( p \).

After updating \( D \):

\[ D \] becomes unable to differentiate between \( p \) and \( q \).

After updating \( G \):

\[ G \] is trained to discriminate samples from data, converging to \( p \).

In other words, \( G \) is trained to resemble \( p \).

The discriminator is unable to differentiate between \( p \) and \( q \) if \( G \) has enough capacity, i.e., in the non-parametric limit. See Figure 1 for a less formal, more pedagogical explanation of the approach. In practice, we must implement the game using an iterative, numerical approach. Optimizing \( D \) and \( G \) alternately would result in overfitting. Instead, we alternate between steps of optimizing \( G \) and \( D \) to maintain samples from a Markov chain from one learning step to the next in order to avoid the training criterion allowing one to recover the data generating distribution as \( p \).

In the next section, we present a theoretical analysis of adversarial nets, essentially showing that Algorithm 1 optimizes Eq 1, thus obtaining the desired result.

We will show in section 4.1 that this minimax game has a global optimum for the space of probability density functions.

In the parametric setting, we represent a model with infinite capacity by studying convergence in the space of probability density functions.
Theoretical properties

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

- Theoretical properties (assuming infinite data, infinite model capacity, direct updating of generator's distribution):
  - Unique global optimum.
  - Optimum corresponds to data distribution.
  - Convergence to optimum guaranteed.
Quantitative likelihood results

- Parzen window-based log-likelihood estimates.

Table 1: Parzen window-based log-likelihood estimates. The reported numbers on MNIST are the mean log-likelihood of samples on test set, with the standard error of the mean computed across examples. On TFD, we computed the standard error across folds of the dataset, with a different model chosen using the validation set of each fold. On TFD, was cross validated on each fold and mean log-likelihood on each fold were computed.

For MNIST we compare against other models of the real-valued (rather than binary) version of dataset. Of the Gaussians was obtained by cross validation on the validation set. This procedure was introduced in Breuleux et al. [8] and used for various generative models for which the exact likelihood is not tractable [25, 3, 5]. Results are reported in Table 1. This method of estimating the likelihood has somewhat high variance and does not perform well in high dimensional spaces but it is the best method available to our knowledge. Advances in generative models that can sample but not estimate likelihood directly motivate further research into how to evaluate such models.

In Figures 2 and 3 we show samples drawn from the generator net after training. While we make no claim that these samples are better than samples generated by existing methods, we believe that these samples are at least competitive with the better generative models in the literature and highlight the potential of the adversarial framework.

![Image](image.png)

**Figure 2:** Visualization of samples from the model. Rightmost column shows the nearest training example of the neighboring sample, in order to demonstrate that the model has not memorized the training set. Samples are fair random draws, not cherry-picked. Unlike most other visualizations of deep generative models, these images show actual samples from the model distributions, not conditional means given samples of hidden units. Moreover, these samples are uncorrelated because the sampling process does not depend on Markov chain mixing.

\begin{tabular}{|l|c|c|}
\hline
Model & MNIST & TFD \\
\hline
Stacked CAE [3] & 121 ± 1.6 & 2110 ± 50 \\
Adversarial nets & 225 ± 2 & 2057 ± 26 \\
\hline
\end{tabular}
Visualization of model samples

MNIST

CIFAR-10 (fully connected)

TFD

CIFAR-10 (convolutional)
Learned 2-D manifold of MNIST
1. Draw sample (A)
2. Draw sample (B)
3. Simulate samples along the path between A and B
4. Repeat steps 1-3 as desired.
Visualization of model trajectories

MNIST digit dataset

Toronto Face Dataset (TFD)
Visualization of model trajectories

CIFAR-10 (convolutional)
Extensions

• Conditional model:
  - Learn $p(x \mid y)$
  - Discriminator is trained on $(x, y)$ pairs
  - Generator net gets $y$ and $z$ as input
  - Useful for: Translation, speech synth, image segmentation.
Extensions

- Inference net:
  - Learn a network to model $p(z \mid x)$
  - Infinite training set!
Extensions

• Take advantage of high amounts of unlabeled data using the generator.
• Train G on a large, unlabeled dataset
• Train G’ to learn $p(z|x)$ on an infinite training set
• Add a layer on top of G’, train on a small labeled training set
Extensions

• Take advantage of unlabeled data using the discriminator

• Train G and D on a large amount of unlabeled data
  - Replace the last layer of D
  - Continue training D on a small amount of labeled data
Thank You.

Questions?