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 $x \sim p_{\text{data}}(x)$

• Want a model that can draw samples: $x \sim p_{\text{model}}(x)$

• Where $p_{\text{model}} \approx p_{\text{data}}$
Why generative models?

• Conditional generative models
  - Speech synthesis: Text ⇒ Speech
  - Machine Translation: French ⇒ 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 ⇒ Image segmentation

• Environment simulator
  - Reinforcement learning
  - Planning

• Leverage unlabeled data
Maximum likelihood: the dominant approach

- ML objective function

\[
\theta^* = \max_{\theta} \frac{1}{m} \sum_{i=1}^{m} \log p(x^{(i)}; \theta)
\]
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:

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

- 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 = peaky distribution.
  - Large valued parameters means slow mixing of sampler.
  - Slow mixing means that the gradient updates are correlated ⇒ 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)
Directed graphical models

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

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

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

• 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>Rock</th>
<th>Paper</th>
<th>Scissors</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>You</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rock</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>Paper</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>Scissors</td>
<td>-1</td>
<td>1</td>
<td>0</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

D tries to output 1
Differentiable function D
x sampled from data

D tries to output 0
Differentiable function D
x sampled from model

Differentiable function G
Input noise Z
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

$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}$$
Poorly fit model

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

Data distribution

Model distribution

Poorly fit model
Learning process

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

Data distribution

Model distribution

Poorly fit model

After updating D
Poorly fit model

Then show in section 4.2 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, e.g. we represent a model with infinite capacity by studying convergence in the space of probability density functions. The generator $G$ maps from a space of random variables $z$ to a space of possible samples $x$.

After updating $D$, the data distribution $p_D(data)$ is better matched to the model distribution. After updating $G$, the model distribution is better matched to the data distribution.

In practice, equation 1 may not provide sufficient gradient for the learning process. In this case, we must implement the game using an iterative, numerical approach. Optimizing $D$ and $G$ alternates, and the training criterion allows one to recover the data generating distribution as $p_D(x)$.

In other words, $p_D(data)$ is a partially accurate classifier. Early in learning, $G$ and $D$ have enough capacity, they will reach a mixed strategy equilibrium.

In this case, we represent a model with infinite capacity by studying convergence in the space of probability density functions. The generator $G$ maps from a space of random variables $z$ to a space of possible samples $x$.

After updating $D$, the data distribution $p_D(data)$ is better matched to the model distribution. After updating $G$, the model distribution is better matched to the data distribution.

In practice, equation 1 may not provide sufficient gradient for the learning process. In this case, we must implement the game using an iterative, numerical approach. Optimizing $D$ and $G$ alternates, and the training criterion allows one to recover the data generating distribution as $p_D(x)$.

In other words, $p_D(data)$ is a partially accurate classifier. Early in learning, $G$ and $D$ have enough capacity, they will reach a mixed strategy equilibrium.

In this case, we represent a model with infinite capacity by studying convergence in the space of probability density functions. The generator $G$ maps from a space of random variables $z$ to a space of possible samples $x$.

After updating $D$, the data distribution $p_D(data)$ is better matched to the model distribution. After updating $G$, the model distribution is better matched to the data distribution.

In practice, equation 1 may not provide sufficient gradient for the learning process. In this case, we must implement the game using an iterative, numerical approach. Optimizing $D$ and $G$ alternates, and the training criterion allows one to recover the data generating distribution as $p_D(x)$.

In other words, $p_D(data)$ is a partially accurate classifier. Early in learning, $G$ and $D$ have enough capacity, they will reach a mixed strategy equilibrium.
In the next section, we present a theoretical analysis of adversarial nets, essentially showing that in Algorithm 1.

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, e.g. we represent a model with infinite capacity by studying convergence in the $G$ space of $p\sim\mathcal{N}(\mu,\sigma_2^2)$.

After updating $D$ (a) (b) (c) (d) the two distributions, i.e. $p\sim\mathcal{N}(\mu,\sigma_2^2)$, cannot improve because they are given enough capacity and training time. The results of this section are done in a non-parametric limit. See Figure 1 for a less formal, more pedagogical explanation of the approach.

Generative adversarial nets are trained by simultaneously updating the generator $G$ and the discriminator $D$. The discriminator is trained to discriminate samples from data, converging to a good estimator of the data distribution $p\sim\mathcal{N}(\mu,\sigma_2^2)$.

Algorithm 1. GAN Training Protocol

1. $D$ and $G$ initialized
2. for $k$ steps do
   a. $G$ updated with the gradient of $V(D, G) = \mathbb{E}_{x\sim p_{\text{data}}} [\log(D(x))] + \mathbb{E}_{z\sim p_z} [\log(1 - D(G(z)))]$
   b. $D$ updated with the gradient of $V(D, G) = \mathbb{E}_{x\sim p_{\text{data}}} [\log(D(x))] + \mathbb{E}_{z\sim p_z} [\log(1 - D(G(z)))]$
3. return $G$, $D$

In practice, equation 1 may not provide sufficient gradient for $G$ and $D$ to converge to a good estimator. Instead, we alternate between $G$ and $D$ updates in Algorithm 1.

In the inner loop of the algorithm:
- Mixed strategy equilibrium
- Poorly fit model
- After updating $D$
- After updating $G$
- Data distribution
- Model distribution
- $p\sim\mathcal{N}(\mu,\sigma_2^2)$
- $x$
- $z$
min \max_{D,G} V(D,G) = \mathbb{E}_{x \sim p_{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.
  - Density estimate with Gaussian kernels centered on the samples drawn from the model.

<table>
<thead>
<tr>
<th>Model</th>
<th>MNIST</th>
<th>TFD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stacked CAE [3]</td>
<td>121 ± 1.6</td>
<td>2110 ± 50</td>
</tr>
<tr>
<td>Adversarial nets</td>
<td>225 ± 2</td>
<td>2057 ± 26</td>
</tr>
</tbody>
</table>
Visualization of model samples

- MNIST
- TFD
- CIFAR-10 (fully connected)
- 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.
• 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
• 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?