Generative adversarial networks
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

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

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

• 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 = peaky 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

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

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

\[ p(x) = \sum_{h} p(x \mid h)p(h) \]

- 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>You</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

2014 NIPS Workshop on Perturbations, Optimization, and Statistics --- Ian Goodfellow
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

\[ \mathbf{p}_D(\text{data}) \]

Data distribution

Model distribution

Poorly fit model

**Learning process**
Learning process

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

Data distribution

Model distribution

Poorly fit model

After updating \( D \)

In the next section, we present a theoretical analysis of adversarial nets, essentially showing that

\[
\text{an adversarial pair near convergence:}
\]

\[
\text{(a) (b) (c) (d)}
\]

\[
\text{After updating D}
\]

\[
\text{After updating G}
\]

\[
\text{Poorly fit model}
\]

\[
\text{After updating D}
\]
We will show in section 4.1 that this minimax game has a global optimum for the space of probability density functions.

The generator $G$ is trained to discriminate samples from data, converging to a good estimator of the training data.

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.

Consider an adversarial pair near convergence: $(D^*, G^*)$. (a) $(D^*)$ is a partially accurate classifier. (b) In the inner loop of the algorithm, $D$ discriminates the two distributions, i.e. $D^*$ contracts in regions of high density and expands in regions of low density of the domain from which $z$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $D$. Therefore, we would like Algorithm 1 to converge to a good estimator of data $p_{\text{data}}$. We will show in section 4.2 that Algorithm 1 optimizes Eq 1, thus obtaining the desired result.

In other words, $p_{\text{data}}$ is similar to $p_{\text{data}}$ when $G$ and $D$ have 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 $p_{\text{data}}$ to completion in the inner loop of training is computationally prohibitive.

In practice, equation 1 may not provide sufficient gradient for learning process burning in a Markov chain as part of the inner loop of learning. The procedure is formally presented in Algorithm 1.

In the inner loop of the algorithm, $D$ is trained to discriminate samples from the data generating distribution (black, solid line) so that it discriminates between samples from the data generating distribution (black, solid line) and one step of optimizing $G$ (dotted line) (c) After an update to the discriminative distribution $(D^{t+1})$, $(D^{t+1})$ can reject samples with high confidence because they are clearly different from transformed samples. (d) After several steps of training, if $G$ contracts in regions of high density and expands in regions of low density of the model distribution, $D$ will reach a mixed strategy equilibrium. Optimizing $\log(1-D(x))$ changes slowly enough. This strategy is analogous to the way that SML/PCD [31, 29] approach.
Poorly fit model

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

Data distribution

Model distribution

\( \begin{array}{c}
\text{Poorly fit model} \\
\text{After updating D} \\
\text{After updating G} \\
\text{Mixed strategy equilibrium}
\end{array} \)
Theoretical properties

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

- 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><strong>2110 ± 50</strong></td>
</tr>
<tr>
<td>Adversarial nets</td>
<td><strong>225 ± 2</strong></td>
<td><strong>2057 ± 26</strong></td>
</tr>
</tbody>
</table>
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
Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

Emily Denton\textsuperscript{1*}, Soumith Chintala\textsuperscript{2*},
Arthur Szlam\textsuperscript{2}, Rob Fergus\textsuperscript{2}

\textsuperscript{1}New York University
\textsuperscript{2}Facebook AI Research
*Denotes equal contribution

December 16, 2015
Overview

- Parametric generative model of natural images

- Difficult to generate large natural images in one shot, but we can exploit their multi-scale structure

- We combine the power of generative adversarial networks (GAN) with a multi-scale image representation (Laplacian pyramid)
Have access to \( x \sim p_{data}(x) \) through training set

Want to learn a model \( x \sim p_{model}(x) \)

Want \( p_{model} \) to be similar to \( p_{data} \)
- Samples drawn from \( p_{model} \) reflect structure of \( p_{data} \)
- Samples from true data distribution have high likelihood under \( p_{model} \)
Why do generative modeling?

- Unsupervised representation learning
  - Can transfer learned representation so discriminative tasks, retrieval, clustering, etc.

- Train network with both discriminative and generative criterion
  - Very little labeled data
  - Regularization

- Understand data

- Density estimation

- ...
CIFAR-10 samples from other models

Goodfellow et al. (2014):

Sohl-Dickstein et al. (2015):

Gregor et al. (2015):
Generative adversarial networks (Goodfellow et al., 2014)

- Generative model $G$: captures data distribution
- Discriminative model $D$: trained to distinguish between real and fake samples, defines loss function for $G$
Generative adversarial networks

- $D$ is trained to estimate the probability that a sample came from data distribution rather than $G$

- $G$ is trained to maximize the probability of $D$ making a mistake

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}}[\log D(x)] + \mathbb{E}_{z \sim p_{noise}}[\log (1 - D(G(z)))]$$
Conditional generative adversarial networks (CGAN)

- Condition generation on additional info $y$ (e.g. class label, another image)
- $D$ has to determine if samples are realistic given $y$

$z \sim p_{\text{noise}}(z)$  
$y \sim p_{\text{data}}(y)$

Generative network

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

Discriminative network

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

Discriminative network

$D$ tries to output 0

$D$ tries to output 1

[Mirza and Osindero (2014); Gauthier (2014)]
Laplacian pyramid (Burt & Adelson, 1983)

E. Denton, S. Chintala, et al. Laplacian Pyramid of Generative Adversarial Nets
Laplacian pyramid (Burt & Adelson, 1983)
Training procedure

- Train conditional GAN for each level of Laplacian pyramid
- \( G \) learns to generate high frequency structure consistent with low frequency image

E. Denton, S. Chintala, et al. Laplacian Pyramid of Generative Adversarial Nets
Training procedure

Each level of Laplacian pyramid trained independently

E. Denton, S. Chintala, et al. Laplacian Pyramid of Generative Adversarial Nets
Sampling procedure
CIFAR-10

- Small dataset 32x32 images of objects, 50k images, 10 classes

**Generator:**

- 4 channels (3 color, 1 noise)
- Conv1 (7x7) + ReLU
- Conv2 (7x7) + ReLU
- Conv3 (5x5)

**Discriminator:**

- Conv1 (5x5) + ReLU
- Conv2 (5x5) + ReLU
- Linear + Sigmoid
- Probability image is real

E. Denton, S. Chintala, et al. Laplacian Pyramid of Generative Adversarial Nets
CIFAR-10 ship samples
CIFAR-10 horse samples

E. Denton, S. Chintala, et al.

Laplacian Pyramid of Generative Adversarial Nets
CIFAR-10 nearest neighbours (pixel space)

Airplane

Automobile

Bird

Cat
CIFAR-10 nearest neighbours (nn feature space)

E. Denton, S. Chintala, et al.

Laplacian Pyramid of Generative Adversarial Nets
CIFAR-10 human evaluations

- Humans randomly presented with real or generated image and asked to determine if real or fake.
- Humans think LAPGAN generations are real \(\sim 40\%\) of the time.

![Graph showing the percentage of images classified as real over presentation time.](image)
- Large dataset of scenes, ~10M images, 10 classes.
LSUN coarse-to-fine chain

E. Denton, S. Chintala, et al.
LSUN church samples
LSUN tower samples
LSUN variability

E. Denton, S. Chintala, et al. Laplacian Pyramid of Generative Adversarial Nets
Recent developments in GAN training

- Radford, Metz and Chintala (2015) propose several tricks to make GAN training more stable

- Future work: apply same tricks to training of LAPGAN model to potentially improve samples and produce higher resolution images
Conclusion

- Proposed a simple generative model that can produce decent quality samples of natural images

- Potential to be used as a decoder in autoencoder framework for unsupervised learning

- GAN framework is difficult to train, no clear objective function to track

- Code & demo: http://soumith.ch/eyescream
The End

Code & demo: http://soumith.ch/eyescream