CS3750: ADVANCED MACHINE LEARNING

GENERATIVE ADVERSARIAL NETWORKS

Adapted from Slides made by Khushboo Thaker
Presented by Tristan Maidment

GROWTH (AND DECLINE) IN GAN PAPERS
Overview

Generative Modelling

Input
Training Examples

Output
Some representation of a probability distribution, which defines this example space.

Unsupervised
Data: X
Goal: Learn hidden underlying structure of data

Supervised
Data: X, y
Goal: Learn hidden mapping from X -> y
Why Generative Modelling?

- Noisy Input
- Simulated Data
- Features Representative of Data
- Prediction of Future State
- Missing Data
- Semi-supervised Learning

MAXIMUM LIKELIHOOD BASED MODELS

\[ p(x) \mid \theta^* = \text{ARG} \text{MAX}(\theta) \mid E_{x \sim \text{P}_{\text{data}}} \log P \left( \frac{x}{\theta} \right) \]
PixelRNN
PixelCNN
WaveNet

- Generate image pixels from the corner
- Stable and Fast training
- Slow generation (sequential)
- Cannot generate samples based on latent code
- Tractable

\[ p(x) = \prod_{t=1}^{n} p(x_t|x_1, x_2, \ldots, x_{t-1}) \]

- Maximum Likelihood based Training
- Chain Rule
Variational Auto Encoder

- Able to achieve high likelihood
- Not asymptotically consistent unless $q$ is perfect
- Lower Quality (blurry) samples
- Non tractable

\[
\log p(x) \geq \log p(x) - D_{KL}(q(z) \parallel p(z|x)) = E_{z \sim q} \log p(x, z) + H(q)
\]

Boltzmann Machine

- Energy Function Based Model
- Markov Chains don’t work for long sequences
- Hard to scale on large dataset

\[
p(x, h) = \exp(-E(x, h)) | Z
\]
\[
Z = \sum_{x, h} \exp(-E(x, h))
\]
Where are some properties of GANs?

- Can use latent information
- Asymptotically consistent
- No Markov Chain assumption
- Samples produced are high quality
NEXT FRAME VIDEO GENERATION
Generative Adversarial Networks

Training Images

Generator

Discriminator

Real
Fake

$G(z)$

$D(x)$

$D(G(z))$

Real world images

Sample

$x$

$G$

Latent random variable

$z$

Differentiable module

Download from: https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-up-2016
Generative Adversarial Networks

“The generative model can be thought of as analogous to a team of counterfeitors, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles.”


Minimax Game Approach

\[
\min_G \max_D - J^D
\]

\[
J^D = -\frac{1}{2} \mathbb{E}_{x \sim P_{\text{data}}} \log D(x) - \frac{1}{2} \mathbb{E}_z \log \left(1 - D(G(z))\right)
\]

\[
J^G = -J^D
\]

- Generator minimizes the log-probability of the discriminator being correct
- Resembles Jensen-Shannon divergence
- Saddle point of Discriminator’s loss
Minimax Game Approach

- Generator minimizes the log-probability of the discriminator being correct
- Resembles Jensen-Shannon divergence
- Saddle point of Discriminator's loss

Nash Equilibrium / Saddle Point

\[
\frac{\partial J^D}{\partial D(X)} = 0 \rightarrow D^*(x) = \frac{P_{\text{data}}(x)}{P_{\text{data}}(x) + P_{\text{model}}(x)}
\]

\[P_{\text{data}}(x) = P_{\text{model}}(x) \quad \forall x\]

\[D^*(x) = \frac{1}{2} \quad \forall x\]

Vanishing Gradient Problem

- Gradient disappears if D is confident, i.e. \(D(G(z)) \rightarrow 0\)
- As can be seen that whenever the discriminator becomes very confident the loss value will be zero
- Nothing to improve for Generator
Heuristic Non-Saturating Games

- Generator maximizes the log probability of the discriminator's mistake
- Does not change when discriminator is successful

\[ J_D = -\frac{1}{2} E_{x \sim P_{data}} \log D(x) - \frac{1}{2} E_z \log \left( 1 - D(G(z)) \right) \]

\[ J_G = -\frac{1}{2} E_z \log D(G(z)) \]

\[ J_G = -J_D \]

\[ J_G = -\frac{1}{2} E_z \exp(\sigma^{-1} D(G(z))) \]

\[ J_G = -\frac{1}{2} E_z \log D(G(z)) \]

COMPARISON OF GENERATOR LOSSES
Non-Convergence

D & G nullifies each others learning in every iteration
Train for a long time — without generating good quality samples

\[ V(x, y) = xy \]
\[ x = 0, \quad y = 0 \]

\[ V(x(t), y(t)) = x(t)y(t) \]
\[ \frac{\partial x}{\partial t} = -y(t) \]
\[ \frac{\partial y}{\partial t} = x(t) \]
\[ \frac{\partial^2 y}{\partial t^2} = \frac{\partial x}{\partial t} = -y(t) \]
\[ x(t) = x(0)\cos(t) - y(0)\sin(t) \]
\[ y(t) = x(0)\cos(t) - y(0)\sin(t) \]

- Differential Equation's solution has sinusoidal terms
- Even with a small learning rate, it will not converge
- Discrete time gradient descent can spiral outward for large step size

\[ \min_G \max_D V(G, D) \neq \max_D \min_G V(G, D) \]
Why are GANs hard to train?

- Generator keeps generating similar images - so nothing to learn
- Maintain trade-off of generating more accurate vs. high coverage samples
- The two learning tasks need to have balance to achieve stability
- If discriminator is not sufficiently trained - leads to poor generator performance
- If discriminator is over-trained - vanishing gradient problem
Tricks to Train GANs

- One-Sided Label Smoothing
- Historically generated batches
- Feature Matching
- Batch Normalization
- Regularizing discriminator gradient in region around real data (DRAGAN)

One-Sided Label Smoothing
- Generator is VERY sensitive to output from Discriminator
- Regulates Discriminator gradients
- Does not reduce accuracy
- Increases confidence
- Only smooth positive samples

$$J_D = -\frac{1}{2} \sum_{x \sim P_{data}} 0.9 \log D(x) - \frac{1}{2} \sum_{z} \log \left(1 - D(G(z))\right)$$
Feature Matching

- Generated images must match statistics of real images
- Discriminator defines the statistics
- Generator is trained such that the expected value of statistics matches the expected value of real statistics
- Generator tries to minimize the L2 distance in expected values in some arbitrary space
- Discriminator defines that arbitrary space

$$\| E_{x \sim p_{data}} f(x) - E_{z \sim P_{model}} f(G(z)) \|_2^2$$

Batch Normalization

- Construct different mini-batches for real and fake
- Each mini-batch needs to contain only all real images or all generated images.
- Makes samples with-in a batch less dependent
Failed GANs typically have extreme gradients/sharp peaks around real data
Regularize GANs to reduce the gradient of the discriminator in region around real data

\[ \lambda \cdot E_{x \sim p_{data}, \delta \sim N(0, I)} \left[ \| \Delta_x D(x + \delta) \| - k \right]^2 \]

GAN Variations
- Conditional GAN
- LapGAN
- DCGAN
- CatGAN
- InfoGAN
- AAE
- DRAGAN
- IRGAN
- ProGAN
- and more!
Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution \( Z \) is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64 \( \times \) 64 pixel image. Notably, no fully connected or pooling layers are used.
Figure 4: Manipulating latent codes on 3D Chairs: In (a), we show that the continuous code captures the pose of the chair while preserving its shape, although the learned pose mapping varies across different types; in (b), we show that the continuous code can alternatively learn to capture the widths of different chair types, and smoothly interpolate between them. For each factor, we present the representation that most resembles prior supervised results [7] out of 5 random runs to provide direct comparison.

DCGAN
- Multiple Convolutional Layers
- Batch Normalization
- Strides with Convolution
- Leaky ReLUs

Conditional GANs $P(X | Y)$
- Generator Learns $P(X|Z,Y)$
- Discriminator Learns $P(L|X,Y)$

$$J_D = -\frac{1}{2} E_{x \sim p_{\text{data}}} \log D(x|y) - \frac{1}{2} E_z \log \left( 1 - D(G(z|y)) \right)$$
InfoGAN

- Rewards Disentanglement
  - (individual dimensions capturing key attributes of images)
- $Z$ - partitioned into two parts
  - $z$ - capture slight variation in the images
  - $y$ - captures the main attributes of the images
- Mutual Information
  - maximizing mutual information Between the code and generator output
InfoGAN

\[
\min_G \max_D V_I(D, G) = V(D, G) - \lambda I(c; G(z, c))
\]

\[
I(c; G(z, c)) = H(c) - H(c|G(z, c))
\]

\[
= \mathbb{E}_{x \sim G(x,c)} [D_{KL}(Q || Q) + \mathbb{E}_{c' \sim p(c|x)} \log Q(c'|x)] + H(c)
\]

\[
\geq \mathbb{E}_{x \sim G(x,c), c \sim p(c)} [\log Q(c|x)] + H(c)
\]
LapGAN

- Scale GANs for large images
- Laplacian pyramid function is used to generate different scales of image
PROGAN

ADVERSARIAL AUTOENCODER (GAN + VAE)
Conclusion

GAN is still an active area of research
GAN framework is flexible to support variety of learning problems
GAN does not guarantee to converge
GAN can capture perceptual similarity and generates better images than VAE
Needs a lot of work in theoretic foundation of Network
Evaluation of GAN is still an open research (Theis et. al)

Important Papers to dig into GAN

- **NIPS 2016 Tutorial:** [Ian Goodfellow](https://www.tensorflow.org/)


- [https://github.com/soumith/ganhacks#authors](https://github.com/soumith/ganhacks#authors)


- [https://www.araya.org/archives/1183](https://www.araya.org/archives/1183)
Software

- [https://github.com/eriklindernoren/Keras-GAN](https://github.com/eriklindernoren/Keras-GAN)
- [https://github.com/eriklindernoren/PyTorch-GAN](https://github.com/eriklindernoren/PyTorch-GAN)
- [https://github.com/znxlwm/tensorflow-MNIST-cGAN-cDCGAN](https://github.com/znxlwm/tensorflow-MNIST-cGAN-cDCGAN)

References

- Deep Learning Book
- GAN Tutorial: [https://www.youtube.com/watch?v=HGYYEUSm-0Q](https://www.youtube.com/watch?v=HGYYEUSm-0Q)
THANK YOU!