Introduction to Machine Learning
Foundations and Applications

Paul J. Atzberger
University of California Santa Barbara
Generative Adversarial Networks (GANs)

Paul J. Atzberger

Diagram showing the process of GANs:
- Training target data
- Noise
- Generator G(z)
- Discriminator D(x)
- Classification of target data and generated data
- Generated data
Motivations: Image Generation

GANs: CIFAR-10, 32x32

GANs: LSUN, 256x256

CycleGANs

GANs Celeb-HQ

Many other applications...

Karras 2018

Zhu 2018

Paul J. Atzberger, UCSB

Machine Learning

http://atzberger.org/
Motivations

Manifold-like structures in high dimensional spaces (natural images, audio, physical fields, PDE solutions).

**Challenge:** How to learn high dimensional probability distributions, generators $G(z)$ for sampling?

Generative Modeling

Approaches for learning models:
- Bayesian Methods
- Maximum Likelihood Estimation (MLE)
- and many more...

**Challenge:** How to do this in a tractable way?
Generative Modeling

$p_{model}(z; \theta)$

generated data

$p_{data}(z)$

$p_{data}(z)$  $p_{model}(z; \theta)$
Generative Models

**Goal**

Learn a model distribution $p_{\text{model}}(z; \theta)$ approximating the data distribution $p_{\text{data}}(z)$.

**Classification:** For input $x$ assign the class $y^* = \arg\max_y p_{\text{model}}(y|x; \theta)$ (approximates the Bayes classifier).

For $z = (x, y)$ this is typically broken down using $p(x, y) = p(y|x)p(x)$. The model distribution with parameter $\theta$ is then $p_{\text{model}}(x, y; \theta) = p(y|x; \theta)p_{\text{data}}(x)$, where $p_{\text{data}}(x) = \int p_{\text{data}}(x, y) d\mu_y$.

**Optimization Formulation**

For an objective function $J[p_{\text{model}, \theta}, p_{\text{data}}]$, find

$$\theta^* = \arg\min_{\theta} J[p_{\text{model}, \theta}, p_{\text{data}}].$$

**Maximum Likelihood** is a widely used approach, corresponds to the objective

$$J[\theta] = J[p_{\text{model}, \theta}, p_{\text{data}}] = -\mathbb{E}_{(x, y) \sim p_{\text{data}}} \left[ \log \left( p_{\text{model}}(x, y; \theta) \right) \right].$$

This is equivalent to minimizing the **Kullback-Leibler Divergence** $D_{KL}$ with

$$J[\theta] = D_{KL} \left( p_{\text{data}} \parallel p_{\text{model}, \theta} \right).$$
Generative Models

**In practice:** We do not have data distribution but only training samples \( \{z_i\}_{i=1}^m \).
We construct the **empirical data distribution**

\[
\tilde{p}_{\text{data}}(z) = \frac{1}{m} \sum_{i=1}^{m} \delta(z - z_i).
\]

**Goal (empirical distribution)**

Learn a model distribution \( p_{\text{model}}(z; \theta) \) approximating the data distribution \( \tilde{p}_{\text{data}}(z) \).

Find

\[
\theta^* = \arg\min_{\theta} J[p_{\text{model}, \theta}, \tilde{p}_{\text{data}}].
\]

**Maximum Likelihood (empirical data distribution):** For \( \tilde{p}_{\text{data}} \) becomes

\[
J[\theta] = -\mathbb{E}_{(x, y) \sim \tilde{p}_{\text{data}}} \left[ \log \left( p_{\text{model}}(x, y; \theta) \right) \right] = -\frac{1}{m} \sum_{i=1}^{m} \log \left( p_{\text{model}}(x_i, y_i; \theta) \right).
\]

**In practice:** \( p_{\text{data}} \) often high dimensional requiring rich class of \( p_{\text{model}, \theta} \). Above requires some way to compute the log-likelihood function. To get good gradient need overlap of distributions (absolute continuity). Often difficult to compute functional form of \( p_{\text{model}} \). Need for alternatives.
Generative Adversarial Networks (GANs)

**Goodfellow 2014:** Generative Adversarial Networks (GANs).

**GANs:** Utilizes deep learning with DNNs for generators $G(z; \theta)$.

**Key idea:** Use properties of supervised learning and generalization behaviors of classifiers $D$ to train generators $G(z; \theta)$.

**Synthetic data distribution** mixture of “real” and “fake” samples.

**Two player-game:**
(i) $D$ aims to classify $x$ as “real” or “fake.”
(ii) $G$ aims to generate “fake” samples so well $D$ can not tell difference.

**Successes:** image generation, video augmentation, and other applications. Challenges (counting, spatial alignment,...)
Generative Adversarial Networks (GANs)

Learn generative models using:

**GANs**

**Generator** $G$: samples $x \sim p_{\text{model}}(x; \theta^G)$.

**Discriminator** $D(x)$: binary classifier for if
(i) input $x$ is sampled from $p_{\text{data}}(x)$, or
(ii) generated from $p_{\text{model}}(x; \theta^G)$.

**Remark:** Two-player game with $G$ generating samples so well that the discriminator $D$ can not distinguish from samples of the data distribution.

**Remark:** The objective is similar to a counterfeiter $G$ printing money so that the police $D$ can not tell if the bills are real or fake.

**Key Idea:** Replaces the problematic calculation using $D_{KL}$-objective by instead using the discriminator $D$ to serve to drive the model distribution $p_{\text{model}}$ toward $p_{\text{data}}$. Leverages capabilities of supervised learning methods.
Generative Adversarial Networks (GANs)

Learn generative models using:

**GANs**

**Generator** $G$: samples $x \sim p_{\text{model}}(x; \theta^G)$.

**Discriminator** $D(x)$: binary classifier for if
(i) input $x$ is sampled from $p_{\text{data}}(x)$, or
(ii) generated from $p_{\text{model}}(x; \theta^G)$.

**Synthetic Labeled Data:** Create a synthetic labeled set of data as follows:
(i) with probability $1/2$ sample $x$ from the data distribution $p_{\text{data}}(x)$ and assign the label 1,
(ii) with probability $1/2$ sample $x$ from the model distribution $p_{\text{model}}(x; \theta^G)$ and assign the label 0.

**Binary Classifier:** Consider generative classifier that assigns the probability $D(x)$ that $x$ was sampled from the data distribution. Then $1 - D(x)$ is the assigned probability that $x$ was generated from the model distribution.

$$D(x) = p_D(y = 1 | x) \approx \Pr\{Y = 1 | X = x\}, \quad 1 - D(x) = p_D(y = 0 | x) \approx \Pr\{Y = 0 | X = x\}.$$ 

**Classification:** For input $x$ assign the class $y^* = \arg\max_y \Pr\{Y = y | X = x\}$ (approximates Bayes classifier).
Generative Adversarial Networks (GANs)

**Synthetic Labeled Data:** This has the data distribution \( p_{\text{synth-l}} \) given by

\[
p_{\text{synth-l}}(x, y) = 1_{y=1} \frac{1}{2} p_{\text{data}}(x, y) + 1_{y=0} \frac{1}{2} p_{\text{model}}(x, y; \theta^G).
\]

For this distribution we have

\[
\Pr\{ Y = 1 | X = x \} = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}.
\]

Thus, \( D(x) = p_{\text{data}}(x) / (p_{\text{data}}(x) + p_{\text{model}}(x)) \) would give us the best possible discriminator (Bayes classifier).

**Remark:** If we were successful in getting our model distribution to exactly match the data distribution then \( p_{\text{model}} = p_{\text{data}} \) and \( D(x) = 1/2 \).

**Remark:** When \( D(x) = 1/2 \) the discriminator can not tell if the sample was more likely to come from the data distribution or from the generator. For generative discriminator, let \( p_D(x, y; \theta^D) := p_D(y|x; \theta^D) p_{\text{synth-l}}(x) \).

We aim to achieve this outcome by learning simultaneously \( \theta^D \) for the optimal discriminator \( D \) and learning \( \theta^G \) for an optimal generator \( G \). Let \( C(\theta^G) \) term be entropy of the synthetic distribution.

We formulate the classification problem for \( D \) using cross-entropy loss with objective function

\[
\hat{J}^D(\theta^D, \theta^G) = -\mathbb{E}_{x,y \sim p_{\text{synth-l}}, \theta^G} \left[ \log p_D(x, y; \theta^D) \right] = -\frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \left[ \log (D(x)) \right] - \frac{1}{2} \mathbb{E}_{x \sim p_{\text{model}}, \theta^G} \left[ \log (1 - D(x)) \right] + C(\theta^G).
\]
Generative Adversarial Networks (GANs)

**Discriminator $D$**

Find $\theta^{D*} = \text{arg-min} \ J^D(\theta^D, \theta^G)$ with

$$J^D(\theta^D, \theta^G) = -\mathbb{E}_{x,y \sim p_{\text{synth-l}}, \theta^G} [\log p_D(y|x; \theta^D)] = -\frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} [\log (D(x))] - \frac{1}{2} \mathbb{E}_{x \sim p_{\text{model}}, \theta^G} [\log (1 - D(x))].$$

Entropy term $C(\theta^G)$ not used. Generator $G$ aims for distribution close to data distribution.

**Generator $G$: Approach I**

Find $\theta^{G*} = \text{arg-max} \ J^G(\theta^D, \theta^G)$ with $J^G = J^D$. 

---

Paul J. Atzberger, UCSB  
Machine Learning  
http://atzberger.org/
Generative Adversarial Networks (GANs)

**Discriminator $D$**

Find $\theta^{D*} = \arg\min J^D(\theta^D, \theta^G)$ with

$$J^D(\theta^D, \theta^G) = -\mathbb{E}_{x,y \sim p_{\text{synth-l}}, \theta^G} \left[ \log p_D(y|x; \theta^D) \right] = -\frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \left[ \log (D(x)) \right] - \frac{1}{2} \mathbb{E}_{x \sim p_{\text{model}}, \theta^G} \left[ \log (1 - D(x)) \right].$$

Entropy term $C(\theta^G)$ not used. Generator $G$ aims for distribution close to data distribution.

**Generator $G$: Approach I**

Find $\theta^{G*} = \arg\max J^G(\theta^D, \theta^G)$ with $J^G = J^D$.

This gives a zero-sum game, so has valuation function $V(\theta^D, \theta^G) = J^D = J^G$.

**Remark:** Deep Neural Networks will be used to learn $D(x; \theta^D)$ and $G(z; \theta^G)$.

**Remark:** Notice the objective functions now no longer require evaluating the expression of the model probability distribution. They only require expectations, which can be approximated from sampling $x \sim p_{\text{model}}$.

We use the **reparameterization technique** to generate $x \sim p_{\text{model}}$ using $x = G(z; \theta^G)$, where $z \sim \hat{p}_{\text{model}}$ with $\hat{p}_{\text{model}}$ an easy to generate distribution. The challenge is shifted to learning the function $G(z; \theta^G)$.
Generative Adversarial Networks (GANs)

**Discriminator $D$**

Find $\theta^D_* = \text{arg-min } J^D(\theta^D, \theta^G)$ with

$$J^D(\theta^D, \theta^G) = -\mathbb{E}_{x,y \sim p_{\text{synth-l}}, \theta^G} \left[ \log p_D(y|x; \theta^D) \right] = -\frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \left[ \log (D(x)) \right] - \frac{1}{2} \mathbb{E}_{x \sim p_{\text{model}}, \theta^G} \left[ \log (1 - D(x)) \right].$$

Entropy term $C(\theta^G)$ not used. Generator $G$ aims for distribution close to data distribution.

**Generator $G$: Approach I**

Find $\theta^G_* = \text{arg-max } J^G(\theta^D, \theta^G)$ with $J^G = J^D$.

**Vanishing Gradient Issue:** For bad generators the discriminator can become very good at just rejecting samples from the model distribution resulting in vanishing gradient in $\theta^G$ and no learning.

**Alternative Formulation:** We aim for generator to make the discriminator probability $D(x)$ as large as possible (hence fooling it). We use

**Generator $G$: Approach II**

Find $\theta^G_* = \text{arg-max } J^G(\theta^D, \theta^G)$ with $J^G = \mathbb{E}_{z \sim p_{\text{model}}, \theta^G} \left[ \log (D(x; \theta^D)) \right]$. 

Paul J. Atzberger, UCSB

Machine Learning

http://atzberger.org/
Generative Adversarial Networks (GANs)

**Discriminator $D$**

Find $\theta^D_* = \arg\text{-min} \ J^D(\theta^D, \theta^G)$ with

$$J^D(\theta^D, \theta^G) = -\mathbb{E}_{x \sim p_{\text{synth-l}}^I, \theta^G} \left[ \log p_D(y|x; \theta^D) \right] = -\frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \left[ \log D(x) \right] - \frac{1}{2} \mathbb{E}_{x \sim p_{\text{model}}, \theta^G} \left[ \log (1 - D(x)) \right].$$

**Generator $G$: Approach II**

Find $\theta^G_* = \arg\text{-max} \ J^G(\theta^D, \theta^G)$ with $J^G = \mathbb{E}_{z \sim p_{\text{model}}, \theta^G} \left[ \log (D(x; \theta^D)) \right].$

No longer a zero-sum game, the solution $(\theta^D_*, \theta^G_*)$ now characterized as a Nash Equilibrium.

**Training Protocol:** Alternate minimizing discriminator objective with maximizing the generator objective.

**Remark:** This can result in oscillatory learning dynamics. Current area of research on best ways to address (likely this is application dependent).
JS-GANs: Jensen-Shannon Distance

Jensen-Shannon Distance

\[
JS(p_{\text{data}}, p_{\text{model}}) = \frac{1}{2} KL \left( p_{\text{data}} \parallel \frac{p_{\text{data}} + p_{\text{model}}}{2} \right) + \frac{1}{2} KL \left( p_{\text{model}} \parallel \frac{p_{\text{data}} + p_{\text{model}}}{2} \right)
\]

\[JS(p, q) \geq 0 \text{ and } JS(p, q) = 0 \Rightarrow p = q \text{ (a.s.). } KL(p \parallel q) = \mathbb{E}_{x \sim p} \left[ \log \left( \frac{p}{q} \right) \right].\]

The optimal discriminator is \(D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}\). Substituting, we have

\[
J^D(\theta^D, \theta^G) = -\frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \left[ \log \left( \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)} \right) \right] - \frac{1}{2} \mathbb{E}_{x \sim p_{\text{model}}, \theta^G} \left[ \log \left( \frac{p_{\text{model}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)} \right) \right].
\]

This gives

\[
J^D(\theta^D, \theta^G) = -JS(p_{\text{data}}, p_{\text{model}}, \theta^G) + \log(2).
\]

As a result, when \(J^G = J^D\), we have \(\theta^G = \arg\max_{\theta^G} J^D(\theta^D, \theta^G) = \arg\min_{\theta^G} JS(p_{\text{data}}, p_{\text{model}}, \theta^G)\).

Shows that original GANs with optimal discriminator \(D^*(x)\) is equivalent to following gradients to minimize the JS-Distance between the model distribution \(p_{\text{model}}\) and \(p_{\text{data}}\).

GANs have been successfully applied in many practical applications: Image Synthesis, Super-Resolution Imaging, Generative Art, Face and Video Synthesis. Other formulations of GANs (Wasserstein WGANs, E-GANs, etc...)
**Task:** Use GANs to learn Gaussian target data distribution $\rho_{data}(x)$.

**Generator** $\rightarrow$ Approximated by Deep Neural Network (DNN) and SGD.

**Training:** Alternate between minimization for D(x) and maximization for G(z).

**Remark:** Cumulative Distribution Function (CDF) $\rightarrow$ Inverse gives a generator.

**Remark:** Gaussians this diverges to give small probability for tails. Noise sources type important consideration in practice.
Example: Gaussian Target Distribution

GANs

Results:

Atzberger 2020
**Task:** Use GANs to learn Gaussian target data distribution $\rho_{data}(x)$.

**Generator** $\rightarrow$ Approximated by Deep Neural Network (DNN) and SGD.

**Training:** Alternate between minimization for $D(x)$ and maximization for $G(z)$.

**Remark:** Cumulative Distribution Function (CDF) $\rightarrow$ Inverse gives a generator.

**Remark:** Gaussians this diverges to give small probability for tails. Noise sources type important consideration in practice.
GANs Celeb-HQ
Task: Use GANs to generate images similar $\rho_{data}(x)$.

Generator $G(z)$: maps noise from latent space $Z \rightarrow$ images $X$.

DNN Generator: Generate images using deep Transpose Convolutional Neural Networks (T-CNNs).

Discriminator $D(x)$: Image classifier based on Convolutional Neural Networks (CNNs).

GANs: Use SGD to learn both classifier and generator at the same time.

Important Considerations: architecture, regularizations (batch normalization), data quality, training protocols (balancing D and G),…
CycleGANs

Zhu 2018
CycleGANs

**Task:** Use input image to generate image of another class.

**GANs** trains two generator maps $G(X)$ and $F(Y)$.

**Two discriminators:** $D_X$ and $D_Y$ try to keep in space of natural images.

**Reconstruction condition:** $X \rightarrow Y \rightarrow \hat{X}$ for information preservation.

**Training:** SGD over a large corpus of images or videos.

**Results:**
- image-to-image conversions (style, time-of-year, object class).
- video-to-video conversions (style, time-of-year, object class).
CycleGANs

horse → zebra

zebra → horse

winter Yosemite → summer Yosemite

summer Yosemite → winter Yosemite
CycleGANs
Summary

GANs provides approach for training Generative Models.

JS-GANs uses properties of supervised learning for discriminator D to obtain loss functions related to classifier behaviors.

Many variants of GANs: Wasserstein (WGANs), Gradient Penalty (GP-GANs), Energy-based (E-GANs), ...

Provides representations and parameterizations for subsets of manifold-like structures.

Challenges remain:
- computationally expensive (involves training DNNs).
- learning full probability distribution (mode collapse).
- reliable training (oscillations, lack of convergence).

Successes in image processing / video (interpolation, super-resolution, reconstruction, augmentation).

Emerging applications in the sciences and engineering (surrogate models, subgrid models, model reductions).