Study Material - Gen AI Week 2

Document Information

- **Generated:** 2025-08-02 13:45:00
- **Source:** Handwritten lecture slides
- **Platform:** IITM Course Material
- **Word Count:** ~3,500 words
- **Estimated Reading Time:** ~18 minutes
- **Number of Chapters:** 6
- **Content:** Variational Divergence Minimization (VDM) and Generative Adversarial Networks (GANs)

Table of Contents

Lecture Overview

This week's material focuses on the practical realization of Variational Divergence Minimization (VDM) and its application in Generative Adversarial Networks (GANs). The lectures provide a mathematical foundation for understanding how GANs work, starting from the theoretical framework of f-divergences and leading to the practical implementation details of training adversarial networks.

Learning Objectives

Upon completing this material, students will be able to: - Understand the mathematical foundation of Variational Divergence Minimization (VDM) - Implement VDM using neural networks for generative modeling - Comprehend the architecture and training procedure of Generative Adversarial Networks (GANs) - Analyze the min-max optimization problem in adversarial training - Implement practical GAN training algorithms with alternating optimization - Understand the role of discriminator and generator networks in the adversarial framework

Prerequisites

To fully grasp the concepts in this material, students should be familiar with: - **Deep Learning Fundamentals:** Neural network architectures, backpropagation, gradient descent - **Probability Theory:** Probability distributions, expectation, divergences (KL divergence, JS divergence) - **Information Theory:** f-divergences, mutual information concepts - **Optimization Theory:** Min-max optimization, saddle point problems - **Mathematical Analysis:** Convex optimization, duality theory

---

Realization of Variational Divergence Minimization (VDM)

Mathematical Foundation of VDM

Variational Divergence Minimization provides a framework for training generative models by minimizing the divergence between the data distribution and the model distribution.

Problem Setup

Given data $D = \{x_1, x_2, \ldots, x_n\}$ drawn i.i.d. from p_x, we want to learn a generator $g_\theta(z)$ where $z \sim \mathcal{N}(0, I)$ such that $\hat{x} \sim p_\theta(\cdot)$ approximates the true data distribution.

The objective is to find:

$$\theta^* = \arg\min_\theta D_f(p_x \| p_\theta)$$

where D_f represents an f-divergence between the true data distribution p_x and the model distribution p_θ.

f-Divergence and Variational Representation

For any f-divergence, we can use the variational representation:

$$D_f(p_x \| p_\theta) \geq \max_{T(x) \in \mathcal{T}} \left[\mathbb{E}_{p_x}[T(x)] - \mathbb{E}_{p_\theta}[f^*(T(x))]\right]$$

where: - f^* is the convex conjugate of the f-divergence function f - $\mathcal{T}$ is a class of functions (typically neural networks) - $T(x)$ is the critic function

Neural Network Parameterization

The optimal solution becomes:

$$\begin{aligned}\theta^* &= \arg\min_\theta D_f(p_x \| p_\theta) \\ &\approx \arg\min_\theta \left[\text{lower bound on } D_f\right] \\ &= \arg\min_\theta \max_{T(x)} \left(\mathbb{E}_{p_x}[T(x)] - \mathbb{E}_{p_\theta}[f^*(T(x))]\right)\end{aligned}$$

We represent the critic function T via neural networks $T_w(x)$ where w are the parameters of the network.

With this parameterization, the objective becomes:

$$\theta^*, w^* = \arg\min_\theta \max_w \left[\mathbb{E}_{p_x}[T_w(x)] - \mathbb{E}_{p_\theta}[f^*(T_w(x))]\right]$$

Implementing VDM for Generative Modeling

Architecture Overview

The VDM framework consists of two main components:

1. Generator Network

- **Input:** Random noise $z \sim \mathcal{N}(0, I)$
- **Function:** $g_\theta(z) \to \hat{x} \sim p_\theta(\cdot)$
- **Purpose:** Generate synthetic data samples

2. Critic Network (Discriminator)

- **Input:** Data samples x (real or generated)
- **Function:** $T_w(x) \to T_w(x)$
- **Purpose:** Evaluate the quality of samples according to the f-divergence

Mathematical Formulation

The complete objective function becomes:

$$J(\theta, w) = \mathbb{E}_{p_x}[T_w(x)] - \mathbb{E}_{p_\theta}[f^*(T_w(x))]$$

The optimization problem is a saddle point:

$$\theta^*, w^* = \arg\min_\theta \max_w J(\theta, w)$$

This is an **adversarial problem** where: - The generator (parameterized by θ) tries to minimize the objective - The critic (parameterized by w) tries to maximize the objective

f-Divergence Specific Activations

For different f-divergences, we need specific activation functions:

General Form

$$T_w(x) = \sigma_f(V_w(x))$$

where: - $V_w(x) : \mathcal{X} \to \mathbb{R}$ is a neural network - $\sigma_f(v) : \mathbb{R} \to \text{dom} f^*$ is the f-divergence specific activation - $\text{dom} f^*$ is the domain of the conjugate function f^*

The final critic output: $T_w(x) : \mathcal{X} \to \text{dom} f^*$

Generative Adversarial Networks (GANs)

GAN as a Specific Case of VDM

GANs represent a specific instantiation of the VDM framework using the Jensen-Shannon divergence.

f-Divergence for GANs

For GANs, the f-divergence is defined as:

$$f(u) = u \log u - (u + 1) \log(u + 1)$$

This is similar to the Jensen-Shannon Divergence (JSD).

Conjugate Function and Activation

The conjugate function is:

$$f^*(t) = -\log(1 - \exp(t)), \quad \text{dom} f^* = \mathbb{R}$$

The activation function becomes:

$$\sigma_f(v) = -\log(1 + e^{-v})$$

GAN Objective Function

General VDM Form

$$J(\theta, w) = \mathbb{E}_{p_x}[\sigma_f(V_w(x))] - \mathbb{E}_{p_\theta}[f^*(\sigma_f(V_w(x)))]$$

GAN-Specific Form

$$J_{GAN}(\theta, w) = \mathbb{E}_{p_x}[\log D_w(x)] + \mathbb{E}_{p_\theta}[\log(1 - D_w(x))]$$

where the discriminator function is defined as:

$$D_w(x) = \frac{1}{1 + e^{-V_w(x)}}$$

This is the **sigmoid function**, which outputs values in $[0, 1]$.

Architecture Diagram

```
z ~ N(0,I) → [Generator g_ (z)] → x̂ ~ p_ (x)
                                        ↓
x ~ p_x(x)                → [Discriminator D_w(x)] → [0,1]
```

The discriminator acts as a **binary classifier** distinguishing between real and generated samples.

GAN Architecture and Implementation

Network Architecture

Generator Network

- **Input:** $z \sim \mathcal{N}(0, I)$ (random noise)
- **Output:** $\hat{x} \sim p_\theta(x)$ (synthetic data)
- **Architecture:** Multi-layer neural networks (MLP, FNN, CNN)

Discriminator Network

- **Input:** x (real or generated samples)
- **Output:** $D_w(x) \in [0, 1]$ (probability that input is real)
- **Architecture:** Neural networks (CNN for images, MLP for other data)
- **Final layer:** Sigmoid activation for binary classification

GAN Loss Function

The complete GAN objective is:

$$J_{GAN}(\theta, w) = \mathbb{E}_{x \sim p_x}[\log D_w(x)] + \mathbb{E}_{\hat{x} \sim p_\theta}[\log(1 - D_w(\hat{x}))]$$

Implementation in Practice

Input Data

Given dataset: $D = \{x_1, x_2, x_3, \dots, x_n\}$ drawn i.i.d. from p_x

Discriminator Optimization

$$w^* = \arg\max_w \left(\mathbb{E}_{p_x}[\log D_w(x)] + \mathbb{E}_{p_\theta}[\log(1 - D_w(x))]\right)$$

Using mini-batches B_1 and B_2:

$$w^* \approx \arg\max_w \left[\frac{1}{B_1}\sum_{i=1}^{B_1} \log D_w(x_i) + \frac{1}{B_2}\sum_{j=1}^{B_2} \log(1 - D_w(\hat{x}_j))\right]$$

where: - $x_1, \dots, x_{B_1} \sim p_x$ (real samples) - $\hat{x}_1, \dots, \hat{x}_{B_2} \sim p_\theta$ (generated samples) - $\hat{x}_j = g_\theta(z_j)$ with z_j sampled from noise distribution

---

Training GANs: Discriminator and Generator

Alternating Optimization Procedure

GAN training involves alternating between optimizing the discriminator and generator:

Discriminator Update Step

Objective: Maximize discriminator's ability to distinguish real from fake

$$w^{t+1} \leftarrow w^t + \alpha_1 \nabla_w J_{GAN}(\theta, w)$$

Keep θ constant and optimize:

$$\theta^* = \arg\min_\theta J_{GAN}(\theta, w)$$

This simplifies to:

$$\theta^* \approx \arg\min_\theta \left[\frac{1}{B_1} \sum_{i=1}^{B_1} \log D_w(x_i) + \frac{1}{B_2} \sum_{j=1}^{B_2} \log(1 - D_w(g_\theta(z_j))) \right]$$

Since the first term is independent of θ:

$$\theta^* \approx \arg\min_\theta \left[\frac{1}{B_2} \sum_{j=1}^{B_2} \log(1 - D_w(g_\theta(z_j))) \right]$$

Generator Update Step

Objective: Minimize generator's loss (fool the discriminator)

$$\theta^{t+1} \leftarrow \theta^t - \alpha_2 \nabla_\theta J_{GAN}(\theta, w)$$

Keep w constant and optimize:

$$J_{GAN} = -\frac{1}{B_2} \sum_{j=1}^{B_2} \log(1 - D_w(g_\theta(z_j)))$$

Update: $\theta^{t+1} \leftarrow \theta^t - \alpha_2 \nabla_\theta J_{GAN}(\theta, w)$

Practical Implementation of GAN Training

Training the Discriminator

Step-by-Step Process

1. **Keep θ constant** - Fix generator parameters
2. **Sample data:** $D = \{x_1, x_2, \dots, x_n\}$

3. **Forward Propagation:**
   - Real samples: $x_1, x_2, ... x_{B_1} \to D_w(\cdot) \to$ classification scores
   - Generated samples: $z_1 ... z_{B_2} \to g_\theta(z) \to g_\theta(z_1) ... g_\theta(z_{B_2}) \to D_w(\cdot) \to$ classification scores
4. **Compute Loss:**

$$J_{GAN}(\theta, w) = \left[\frac{1}{B_1} \sum_{i=1}^{B_1} \log D_w(x_i) + \frac{1}{B_2} \sum_{j=1}^{B_2} \log(1 - D_w(g_\theta(z_j))) \right]$$

5. **Backward Propagation:** Compute $\nabla_w J_{GAN}(\theta, w)$
6. **Update:** $w^{t+1} \leftarrow w^t + \alpha_1 \nabla_w J_{GAN}$

Training the Generator

Step-by-Step Process

1. **Sample noise:** $z_1 ... z_{B_2} \sim \mathcal{N}(0, I)$
2. **Forward Propagation:**
   - $z \to g_\theta(z) \to g_\theta(z_1) ... g_\theta(z_{B_2}) \sim p_\theta$
   - Generated samples through discriminator: $g_\theta(z_j) \to D_w(\cdot)$
3. **Compute Generator Loss:**

$$J_{GAN} = -\frac{1}{B_2} \sum_{j=1}^{B_2} \log(1 - D_w(g_\theta(z_j)))$$

4. **Update θ only with w constant:**

$$\theta^{t+1} \leftarrow \theta^t - \alpha_2 \nabla_\theta J_{GAN}(\theta, w)$$

Training Algorithm Summary

```
1. Initialize  , w
2. For epoch = 1 to max_epochs:
   3. // Train Discriminator
   4. Sample {x , ..., x_B } from real data
   5. Sample {z , ..., z_B } ~ N(0,I)
   6. Generate {x̂ , ..., x̂_B } = {g (z ), ..., g (z_B )}
   7. Compute discriminator loss J_D
   8. w^(t+1) ← w^t +   _w J_D

   9. // Train Generator
   10. Sample {z , ..., z_B } ~ N(0,I)
   11. Compute generator loss J_G
   12.  ^(t+1) ←  ^t -   _  J_G
```

Key Training Insights

Saddle Point Optimization

- GANs solve a **min-max** problem: $\min_\theta \max_w J(\theta, w)$
- This is fundamentally different from standard neural network training
- Requires careful balance between discriminator and generator updates

Training Stability

- **Discriminator too strong:** Generator cannot learn (gradients vanish)
- **Generator too strong:** Discriminator cannot provide useful gradients
- **Balanced training:** Both networks improve together

Implementation Considerations

- **Alternating updates:** Update one network while keeping the other fixed
- **Learning rates:** Often different learning rates for generator and discriminator
- **Update frequency:** Sometimes update discriminator multiple times per generator update
- **Batch sizes:** Can use different batch sizes for real and generated samples

---

Key Takeaways from Week 2

Theoretical Understanding

1. **VDM Framework:** Provides mathematical foundation for adversarial training
2. **f-Divergences:** Allow different types of distance measures between distributions
3. **Variational Representation:** Enables practical optimization using neural networks

Practical Implementation

1. **GAN Architecture:** Two-network adversarial system (generator + discriminator)
2. **Training Procedure:** Alternating optimization between networks
3. **Loss Functions:** Binary cross-entropy for discrimination task

Mathematical Insights

1. **Saddle Point Problem:** Min-max optimization requires special consideration
2. **Neural Network Parameterization:** Both generator and critic are neural networks
3. **Gradient-Based Training:** Standard backpropagation with careful update procedures

Applications and Extensions

1. **Image Generation:** GANs excel at generating realistic images
2. **Data Augmentation:** Generate synthetic training data
3. **Domain Transfer:** Learn mappings between different data domains

This material provides the foundation for understanding more advanced GAN variants and training techniques that will be covered in subsequent weeks.