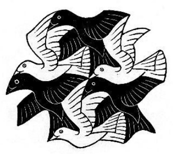
CALEB MOSES & MARK BAI

GANS AND W-GANS



GROUP PROJECT

Math 562 winter term 2023

https://github.com/mathematiguy/gans-and-wgans

Caleb Moses & Mark Bai: *GANs and W-GANs*, Group project, © .

WEBSITE:
https://github.com/mathematiguy/gans-and-wgans

E-MAIL:
caleb.moses@mail.mcgill.ca
hao.bai@mail.mcgill.ca

The titlepage reproduces an engraving of Maurits Cornelis Escher, titled *Plane Filling with Birds* (the picture is obtained from http://www.mcescher.com/).

ABSTRACT

This group project explores the theory behind Wasserstein Generative Adversarial Networks (GANs) which apply useful concepts from Optimal Transport (OT) to the task of image generation. We compare Wasserstein GANs with their predecessor, the GAN and also compare the approach with the more recent phenomenon of diffusion based image generation algorithms.

ACKNOWLEDGEMENTS

With thanks to our professor, Prof Adam Oberman for helping us to formulate our research problem, as well as to the wider community of machine learning practitioners for providing ample reading material on a range of tricky topics that helped us wrap our heads around all of the theory.

CONTENTS

1 OPTIMAL TRANSPORT

Optimal transport theory is concerned with finding the cost minimising map that transfers the mass of one probability distribution to another target distribution. Usually this is described using an "earth moving" analogy, where a pile of dirt (representing a distribution) is moved piece by piece to fill a hole (i.e. the target distribution) elsewhere. This earth moving process has a cost associated with it, which is usually proportional to the distance moved.

Rather than actually being primarily concerned with earth moving, Optimal Transport theory also has applications to partial differential equations, geometry, functional analysis and in data science, to imaging sciences and machine learning. In this paper, we intend to explore examples of the latter case, namely machine learning for image generation.

1.1 THE MONGE–KANTOROVICH TRANSPORTATION PROBLEM

The mathematical formulation of Optimal Transportation takes two forms, called the Monge formulation and the Kantorovich formulation. In both formulations you have a source measure μ and a target measure ν, both of which are measures on a metric spaces X and Y respectively. We refer to the set of all measures on X and Y as $\mathcal{M}(X)$ and $\mathcal{M}(Y)$ respectively. In addition, it is assumed that the total mass of both distributions are equal, in other words $\mu(X) = \nu(Y)$.

Lastly, we also require a continuous cost function $c : X \times Y \to \mathbb{R}$ which represents the cost of moving mass from x to y. We then integrate this cost function over all possible trajectories that transfer mass from source to target and minimise this total cost. Very often, this cost function is related to a metric on the metric spaces X and Y. Cost functions are also generally convex.

1.1.1 The Monge Problem

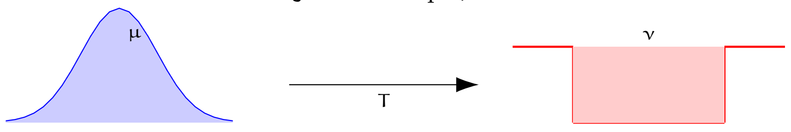
One of the key details in the Monge interpretation is the use of a "push-forward" map. The purpose of this map is to shift the mass from the source distribution to the target distribution in a mass preserving way, and you express this property via a relationship between the transfer map $T : X \to Y$ and the target distribution ν.

Importantly, we need the transfer map to shift the source distribution μ to the target ν, and additionally we require T to be continuous.

Definition 1.1.1 (Push-forward)**.** For $T : X \to Y$, the push-forward measure $\nu = T_{\#}\mu \in \mathcal{M}(Y)$ for some $\mu \in \mathcal{M}(X)$ satisfies $\forall h \in \mathcal{C}(Y)$:

$$\int_Y h(y)d\nu(y) = \int_X h(T(x))d\mu(x)$$

Figure 1: T maps μ to ν



Equivalently, for any measurable set $B \subset Y$ one has:

$$\nu(B) = \mu(x \in X : T(x) \in B) = \mu(T^{-1}(B))$$

Note that $T_\#$ preserves positivity and total mass so that if $\alpha \in \infty_+(\mathcal{X})$ then $T_\#\mu \in \mathcal{M}^1_+(Y)$.

Now that we have finished the relevant setup, we are ready to write down the Monge formulation of Optimal Transport.

Definition 1.1.2 (The Monge Problem)**.** The Monge problem attempts to find the transfer map that minimises the total cost of transferring mass from μ to ν, which we express in the following way:

$$\min_T \left\{ \int_X c(x, T(x)) d\mu(x) : T_\#\mu = \nu \right\}$$

The constraint $T_\#\mu = \nu$ means that T pushes forward the mass of μ to ν. $c(x, T(x))$ is the cost of shifting mass from x to $T(x)$.

Limitations

One limitation of the Monge formulation is that masses are indivisible because the transport map is a function $T(x) = y$. In some situations, it might make sense to split mass into pieces and divide those pieces across the target distribution ν (for example: earth-moving, money, etc). In such cases, a transfer map is not going to exist and we require something more general.

1.1.2 The Kantorovich Problem

Since it does not assume the existence of a transport map, the Kantorovich formulation is more general than Monge and it achieves this using the concept of *product measure*.

Definition 1.1.3 (Product measure)**.** A product measure is a measure defined on the cartesian product of two measurable spaces $\mathcal{X}$ and $\mathcal{Y}$. Letting μ and ν be measures on $\mathcal{X}$ and $\mathcal{Y}$ respectively, the product measure π_{XY} is defined on the product space $\mathcal{X} \times \mathcal{Y}$ in the following manner:

$$\pi(A \times B) = \mu(A)\nu(B)$$

for all $A \in \mathcal{X}$ and $B \in \mathcal{Y}$, where $A \times B$ is the cartesian product of A and B. The measures μ and ν are called the "marginals" of π.

In the Kantorovich formulation, instead of considering transfer maps we consider the set of product measures whose marginals are μ and ν. We denote this set by $\Pi(\mu, \nu)$. Such measures are considered "feasible", and the product measure captures the idea of transporting mass starting at μ and ending at ν.

Definition 1.1.4 (The Kantorovich problem)**.** The Kantorovich formulation of optimal transport is given by the following equation:

$$\inf_\pi \left\{ \int_{X \times Y} c(x, y) d\pi(x, y) : \pi \in \Pi(\mu, \nu) \right\}$$

In this formulation, $\pi(x, y)$ is referred to as a *transport plan*.

In the Kantorovich formulation, we are searching for a product measure whose marginals are μ and ν and for which the total cost over the joint space $X \times Y$ is as small as possible. Because the transport map in the Kantorovich formulation need not be represented by a function, this means it is possible to split mass if necessary.

1.1.3 Transport properties

Transport maps and transport plans have a number of useful properties. 1-D transport maps are monotonic, meaning that they preserve the order of points from their source to their target measure spaces, and in higher dimensions they satisfy a more general property called "cyclical monotonicity".

Transport maps and transport plans are also usually lipschitz continuous, and unique if the cost function is strictly convex. If the metric spaces X and Y are compact (equiv. closed and bounded) then both an optimal transport map and an optimal transport plan is guaranteed to exist, which can be proven using the extreme value theorem.

1.1.4 Cost functions

It is possible to consider a wide range of cost functions in optimal transport problems. The most common examples are the L^2 norm $(x-y)^2$, and more generally the L^p norm $|x-y|^p$. The L^1 norm is not strictly convex, and so it is used less often as the uniqueness of the optimal transport map no longer holds.

You can also use cross-entropy as a cost function, given by $c(x,y) = -\log(p(y|x))$ where $p(y|x)$ is a conditional probability function. This cost function is commonly used in machine learning. You can also define cost functions on graphs/networks, and in that case it things like the length of the shortest path between two nodes are used.

1.2 THE WASSERSTEIN DISTANCE

One important feature of optimal transport is that it can be used to define a metric on a space of probability measures. Usually we compare probability distributions using the Kullback-Liebler divergence, however this has the disadvantage that it is not symmetric which means that the resulting function is not a metric.

However, it turns out that if we compare distributions μ and ν using the minimum L^p cost necessary for an optimal transport plan to shift from μ to ν, this quantity is called the Wasserstein distance $W_p(\mu,\nu)$.

Definition 1.2.1 (The Wasserstein distance)**.** Given two measures μ, ν and metric spaces X and Y respectively and for $p \in [1,\infty)$, we define the p-Wasserstein distance to be:

$$W_p(\mu,\nu) = \inf_{T_\# \mu = \nu} \left(\int_X \|x - T(x)\|_p \, d\mu(x) \right)^{\frac{1}{p}}$$

in the Monge formulation, or in the Kantorovich formulation:

$$W_p(\mu,\nu) = \inf_{\pi \in \Pi(\mu,\nu)} \left(\int_{X \times Y} \|x - y\|_p \, d\pi(x,y) \right)^{\frac{1}{p}}$$

1.2.1 The Wasserstein metric

Given either definition, the metric axioms can all be shown to hold. Namely:

1. For any two probability measures μ and ν, the Wasserstein distance $W_p(\mu,\nu)$ is non-negative: $W_p(\mu,\nu) \geqslant 0$.
2. The Wasserstein distance between a probability measure μ and itself is zero: $W_p(\mu,\mu) = 0$.

3. The Wasserstein distance is symmetric with respect to its arguments: $W_p(\mu, \nu) = W_p(\nu, \mu)$.
4. The Wasserstein distance satisfies the triangle inequality, meaning that for any three probability measures μ, ν, and σ, we have: $W_p(\mu, \sigma) \leqslant W_p(\mu, \nu) + W_p(\nu, \sigma)$.

Non-negativity holds because the Wasserstein distance is an integral of a positive quantity (the L_p norm). (2) holds in the Monge formulation because $x = T(x)$ and so $\|x - T(x)\|_p = 0$ for all x, and similarly for the Kantorovich formulation we have $\|x - x\|_p = 0$.

Symmetry holds for the Monge formulation because cost functions are symmetric, you simply need to do a change of variables to show this, but in the Kantorovich formulation this is more straightforward.

The Triangle Inequality holds in the Monge formulation ultimately because it holds in L_p spaces, and the same is true of the Kantorovich formulation but you need to use more machinery such as the gluing lemma in order to complete the proof.

In addition to these properties, the Wasserstein distance is also continuous and convex. In other words

$$W_p(\lambda\mu_1 + (1-\lambda)\mu_2, \lambda\nu_1 + (1-\lambda)\nu_2) \leqslant \lambda W_p(\mu_1, \nu_1) + (1-\lambda)W_p(\mu_2, \nu_2)$$

for any probability measures μ_1, μ_2, ν_1, ν_2, and any $0 \leqslant \lambda \leqslant 1$.

1.3 KANTOROVICH-RUBINSTEIN DUALITY

The Kantorovich-Rubinstein duality is a connection between two different formulations of the optimal transport problem. In particular, it relates the Wasserstein distance between two probability measures to a supremum of the difference of expectations of certain functions.

Suppose we have a probability measure p and a parametrised model p_θ approximating p on some space X. The Wasserstein distance $W(p, p_\theta)$ is given by:

$$W(p, p_\theta) = \inf_{\pi \in \Pi(p, p_\theta)} \mathbb{E}_{(x,y)\sim\pi}[\|x - y|_2]$$

where $\Pi(p, p_\theta)$ denotes the set of joint probability measures with marginals p and p_θ. This infimum is difficult to compute directly.

The Kantorovich-Rubinstein duality provides an alternative formulation of $W(p, q)$ that is easier to work with. Specifically, it states that

$$W(p, p_\theta) = \sup_{\|h\|_L \leqslant 1} \left[\mathbb{E}_{x\sim p}[h(x)] - \mathbb{E}_{y\sim p_\theta}[h(y)]\right]$$

where the supremum is taken over all L-Lipschitz functions h with Lipschitz constant $L \leqslant 1$. Here, $\mathcal{H}$ denotes the set of all such functions.

Intuitively, the duality expresses the Wasserstein distance between p and p_θ as the largest difference in expectations of certain functions h that satisfy the Lipschitz constraint. The dual formulation is often easier to compute since it only involves optimizing over a set of functions rather than a set of joint probability measures.

To see why the duality holds, note that it follows from a more general result in convex analysis known as Fenchel duality. The key idea is to use the Kantorovich-Rubinstein theorem to construct a convex dual problem whose optimal value is equal to the Wasserstein distance. This allows us to solve the original problem by solving the dual problem instead, which is often simpler or more tractable.

2 GENERATIVE ADVERSARIAL NETWORKS

Generative adversarial networks (GANs) (**NIPS2014_5ca3e9b1**) have demonstrated great performance on producing realistic images, language, and even music. As deep *generative* models, they were traditionally used in the context of unsupervised learning. However, this general technique has also been extended to other problems in semisupervised learning, such as image classification (**denton2015deep**; **radford2015unsupervised**).

GANs learn the high-dimensional distribution of the unlabeled data through a pair of networks (the generator and the discriminator) that are competing against each other. An analogy is used in the original paper: the generator is a team of counterfeiters that produces fake currency, while the discriminator is the police trying to detect forgery. When both networks are optimal, the generator is able to produce counterfeit currency that is indistinguishable from the real ones.

In this section, we first provide an overview of GANs. Then, we will discuss their limitations, which leads to the section of Wasserstein GANs. Since the nature of this review is more mathematically motivated, for details on different GAN architecture and applications, see **creswell2018generative**.

2.1 ADVERSARIAL NETS

We define p_z, p_g, p_r as the distributions of the latent variable z (e.g. $z \sim \mathcal{N}(0, 1)$), generator, and real samples respectively. The discriminator $D(x, \theta_d)$ outputs a scalar that represents the probability that x came from p_r rather than p_g. The generator $G(z, \theta_g)$ then learns a mapping from z to the data space. In other words, it is learning p_g over data x, where $x \sim p_r(x)$.

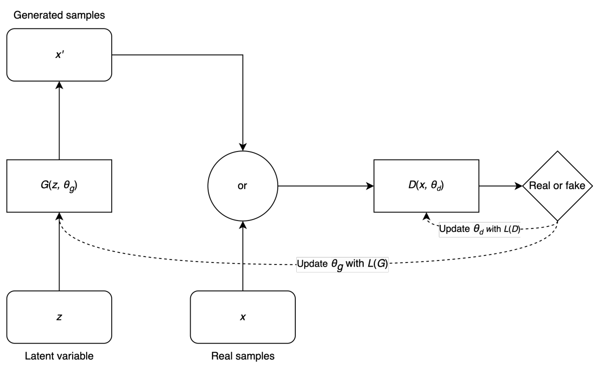


Figure 2: A GAN consists of two models: a discriminator D that outputs the probability that a given sample is real, and a generator G that produces synthetic samples given a latent variable z sampled from the base distribution. The discriminator parameters θ_d are updated to assigns high probability to real samples and low probability to synthetic samples. The generator parameters θ_g are updated to "fool" the discriminator into assigning high probabilities to the generated samples.

D is trained in a binary classification fashion; it maximizes the probability of assigning the correct label to both real samples x and fake samples $x' = G(z)$. This is equivalent to maximizing $\mathbb{E}_{x \sim p_r(x)} [\log(D(x))] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$. The generator then minimizes the probability of the discriminator assigning the correct label to the fake samples: $\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$. In other words, they are playing a minimax game with the following loss function $L(G, D)$:

$$\min_G \max_D L(G, D) = \mathbb{E}_{x \sim p_r(x)}[\log(D(x))] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{1}$$

2.2 TRAINING GANS

Since the discriminator and the generator are playing a two-player non-cooperative game, training them can be thought of as finding the Nash equilibrium, which can be difficult (**salimans2016improved**). In practice, this is done by the following: in each training loop, fix the generator and update the discriminator for k steps, then fix the discriminator and update the generator for 1 step. To update the discriminator, we use the following loss function[1]:

$$\begin{aligned} L(\theta_d) &= -\left(\mathbb{E}_{x \sim p_r(x)}[\log(D(x, \theta_d))] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G^*(z), \theta_d))]\right) \\ &\approx -\frac{1}{m}\sum_{i=1}^{m}[\log(D(x_i, \theta_d))] - \frac{1}{n}\sum_{j=1}^{n}\left[\log(1 - D(G^*(z_j), \theta_d))\right] \end{aligned} \tag{2}$$

where m represents the number of real examples, n represents the number of fake examples, and $G^*(z)$ represents the fixed generator. To update the generator, we use the following loss function:

$$\begin{aligned} L(\theta_g) &= \left(\mathbb{E}_{x \sim p_r(x)}[\log(D^*(x))] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D^*(G(z, \theta_g)))]\right) \qquad (3) \\ &\approx \frac{1}{n}\sum_{j=1}^{n}\left[\log(1 - D^*(G(z_j, \theta_g)))\right] \quad \text{(first term is not related to } G(z, \theta_g)) \end{aligned}$$

where $D^*(x)$ represents the fixed generator. There are also interesting theoretical results related to the loss function of GANs, which have been used by others to analyze their limitations.

2.2.1 Optimal Value for Discriminator and Generator

Given a fixed generator $G^*(z, \theta_g)$, the goal is to maximize the loss function for the discriminator D:

$$\begin{aligned} L(G, D) &= \mathbb{E}_{x \sim p_r(x)}[\log(D(x))] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G^*(z)))] \\ &= \mathbb{E}_{x \sim p_r(x)}[\log(D(x))] + \mathbb{E}_{x \sim p_g(x)}[\log(1 - D(x)] \\ &= \int_x p_r(x)\log(D(x)) + p_g(x)\log(1 - D(x))dx \end{aligned} \tag{4}$$

The maximum value for this function is achieved when $D^*(x) = \frac{p_r(x)}{p_r(x)+p_g(x)}$, and when the *generator* is trained to its optimal, $p_r = p_g$, so $D(x) = \frac{1}{2}$. This is equivalent to the optimal discriminator assigning random labels to any given sample. Although it is trained, it cannot distinguish between real and fake anymore.

2.2.2 Global Optimal and Relationship to JS Divergence

Given $G^*(z, \theta_g)$ such that $p_r = p_g$, and $D^*(x, \theta_d)$ such that $D^*(x) = \frac{1}{2}$,

$$\begin{aligned} L(G^*, D^*) &= \mathbb{E}_{x \sim p_r(x)}[-\log 2] + \mathbb{E}_{x \sim p_g(x)}[-\log 2)] \\ &= -2\log 2 \end{aligned} \tag{5}$$

1 Equation (**??**) is similar to the loss minimization function in **prince2023understanding**. The original paper by **NIPS2014_5ca3e9b1** uses minibatch stochastic gradient ascent for maximization.

Thus, the best possible value of $L(G, D)$ is $-2\log 2$.

KL-divergence is defined as:

$$KL(p\|q) = \int_x p(x) \log \frac{p(x)}{q(x)} dx \tag{6}$$

and JS-divergence is defined as:

$$JSD(p\|q) = \frac{1}{2} KL(p\|\frac{p+q}{2}) + \frac{1}{2} KL(q\|\frac{p+q}{2})$$

Notice that when $D^*(x) = \frac{p_r(x)}{p_r(x)+p_g(x)}$ is used in equation (**??**), it is equivalent to measuring the KL (Kullback-Leibler) divergence between p_r, $\frac{p_r+p_g}{2}$ and p_g, $\frac{p_r+p_g}{2}$, which can be further simplified into the JS (Jensen-Shannon) divergence between p_r, p_g:

$$\begin{aligned}
L(G, D^*) &= \mathbb{E}_{x \sim p_r(x)} [\log(D^*(x))] + \mathbb{E}_{x \sim p_g(x)} [\log(1 - D^*(x)] \\
&= \int_x p_r(x) \log \left(\frac{2 \cdot p_r(x)}{2 \cdot (p_r(x) + p_g(x))} \right) dx + \int_x p_g(x) \log \left(\frac{2 \cdot p_g(x)}{2 \cdot (p_r(x) + p_g(x))} \right) dx \\
&= -2\log 2 + \int_x p_r(x) \log \left(\frac{p_r(x)}{\frac{p_r(x)+p_g(x)}{2}} \right) dx + \int_x p_g(x) \log \left(\frac{p_g(x)}{\frac{p_r(x)+p_g(x)}{2}} \right) dx \\
&= -2\log 2 + KL(p_r\|\frac{p_r+p_g}{2}) + KL(p_g\|\frac{p_r+p_g}{2}) \\
&= -2\log 2 + 2JSD(p_r\|p_g)
\end{aligned}$$

Thus, when the discriminator is optimal, the loss function is equivalent to measuring the difference between p_g and p_r, which is how well the generator is performing. Also, when the generator is optimal, $p_r = p_g$, and $2JSD(p_r, p_g) = 0$, so $L(G^*, D^*) = -2\log 2$, which confirms equation (**??**).

Some have suggested that the use of JS-divergence term in GAN's loss function is the reason behind its success. Since JS-divergence is symmetric, it would not penalize one distribution more than the other and cause "mode dropping". Specifically, consider the terms in equation (**??**) of $KL(p_r\|p_g)$: if $p_r(x) > p_g(x)$, then x is more likely to come from the true data distribution. But when $p_r(x) > 0, p_g(x) \to 0$, KL-divergence becomes very large, so it over penalizes the generator for not covering parts of the true distribution. On the other hand, if $p_r(x) < p_g(x)$, and $p_r(x) \to 0, p_g(x) > 0$, then KL-divergence becomes very small, so it under penalizes the generator for generating fake looking samples.

2.3 LIMITATIONS OF GANS

GANs were known to be difficult to train. Many approached this problem by relying on heuristics, and proposed techniques such as feature matching and minibatch discrimination (**radford2015unsupervised**; **salimans2016improved**). While the techniques improved the stability of GANs in practice, these variants were very sensitive to modifications. **arjovsky2017towards** provided a new perspective on GANs by introducing the theory behind their instability issues.

2.3.1 The Real Cost

Based on the analysis in **??**, the optimally trained discriminator should have maximum cost of $2\log 2 - 2JSD(p_r\|p_g)$, and with the optimal generator, this cost goes to $2\log 2$. However, in practice, this cost approaches 0. This implies that $JSD(p_r\|p_g) = \log 2$, and that the JS-divergence between the two distributions is maxed out. It is shown that this occurs because the supports for p_r and p_g lie on low dimensional manifolds.

To provide some intuition, consider the data space $\mathcal{X}$ which contains all the possible real-world images. Although the dimension of $\mathcal{X}$ appears to be artificially high, there are many pre-existing restrictions placed by nature. For instance, a mammal face usually contains eyes, ears, a nose, a mouth, and a jaw. Thus, the support of p_r is concentrated on a low dimensional manifold. Similarly, the generator $G(z, \theta_g) : \mathcal{Z} \to \mathcal{X}$ is defined by sampling from a prior distribution p_z, which has less dimension (such as 100) than $\mathcal{X}$.

Recall that if $\mathcal{X} : \Omega \to \mathbb{R}^n$ is a random variable, then the support of $\mathcal{X}$ is defined as the set $R_x = \{x \in \mathbb{R}^n : p_r(x) > 0\}$. For instance, the support of p_g has to be contained in $G(z, \theta_g)$. Denote $\mathcal{M}$ as the manifold where the support of p_r lies in, and $\mathcal{P}$ the manifold where the support of p_g lies in. It can be shown that

1. If $\mathcal{M}$ and $\mathcal{P}$ don't perfectly align and don't have full dimensions, then there exists a perfect discriminator $D^{**} : \mathcal{X} \to [0, 1]$ such that it takes the value 1 on a set that contains the support of p_r and value 0 on a set that contains the support of p_g. Also, $\nabla_x D^*(x) = 0$

2. If $\mathcal{M}$ and $\mathcal{P}$ don't perfectly align and don't have full dimensions, then

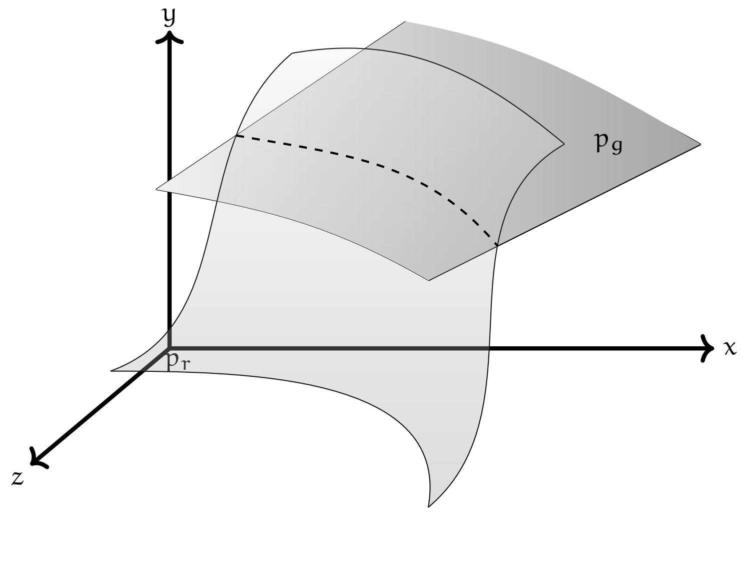
$$\begin{aligned} JSD(p_r \| p_g) &= \log 2 \\ KL(p_r \| p_g) &= +\infty \\ KL(p_g \| p_r) &= +\infty \end{aligned}$$

3. Let $J_{\theta_g} G(z, \theta_g)$ denote the Jacobian of G with respect to θ_g. If **??** is satisfied, and $\|D - D^{**}\| < \epsilon$, and $\mathbb{E}_{z \sim p(z)} \left[\|J_{\theta_g} G(z, \theta_g)\|_2^2 \right] \leqslant C^2$, then

$$\|\nabla_{\theta_g} \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z, \theta_g)))] \|_2 < C \frac{\epsilon}{1 - \epsilon} \tag{7}$$

Note that the perfect discriminator D^{**} in **??** is different from the optimal discriminator in **??**, which assigns equal probabilities to fake and real samples. In fact, this difference is precisely the reason why in practice, the cost of the discriminator approaches 0. In conjunction with the result in **??**, we understand that although the generator is producing realistic looking results, the two underlying distributions p_r, p_g are actually very different and easy for the discriminator to distinguish.

Figure 3: Low dimensional manifolds in high dimension space can hardly have overlaps.



To explain the idea in **??** in simpler terms, imagine the following four cases between p_r and p_g when calculating JSD:

$$p_r(x) = 0, p_g(x) = 0$$
$$p_r(x) \neq 0, p_g(x) \neq 0$$
$$p_r(x) = 0, p_g(x) \neq 0$$
$$p_r(x) \neq 0, p_g(x) = 0$$

The first case does not contribute to JSD. The third case is $\frac{1}{2}\log\left(\frac{p_r}{\frac{p_r+0}{2}}\right) = \frac{1}{2}\log 2$ in JSD calculation. Similarly, the fourth case contributes $\frac{1}{2}\log 2$. For the second case, since $\mathcal{M}$ and $\mathcal{P}$ don't perfectly align and don't have full dimensions, for any parts where $p_r(x)$ and $p_g(x)$ overlap, the resulting integral is negligible, so it also does not contribute to JSD. To provide further intuition, if the data space is $\mathbb{R}^3$, and the supports are planes, then it is very unlikely for them to be perfectly aligned; their intersection is most likely a line which does not contribute to the measure (see **??**).

Finally, the result in **??** provides a bound on the generator's loss when the difference between the discriminator and the perfect discriminator is small. In fact, when the discriminator is perfect, the gradient provided to the generator approaches 0. This is intuitive as the generator loss function (**??**) depends on the discriminator.

$$\lim_{\|D-D^{**}\|\to 0} \nabla_{\theta_g}\mathbb{E}_{z\sim p(z)}\left[\log(1 - D(G(z,\theta_g)))\right] = 0$$

This is also known as the vanishing gradient problem, and explains the instability behaviour in GAN training. Overall, this subsection shows that under the original loss function, there exists a perfect discriminator, which easily distinguishes p_g, p_r, and provides little signal for the generator to learn.

2.3.2 Ways to Address the Instability

In the original paper (**arjovsky2017towards**), the authors first proposed to add noise to p_r and p_g such that their manifolds are able to "spread" to the higher dimensions such that the measure of their overlaps is more significant in higher dimensions. During training, this added noise can be gradually removed, until eventually, the two manifolds are perfectly aligned.

However, they also proposed to use another metric that directly measures the distance between points in the manifolds. This is known as the Wasserstein distance.

3 WASSERSTEIN GANS

Despite their success, GANs suffered from issues such as mode collapse, vanishing gradients, and unstable training dynamics. In this chapter, we introduce Wasserstein Generative Adversarial Networks (WGANs), a variant of GANs that addresses some of these challenges using the Wasserstein distance as the loss function.

3.1 WASSERSTEIN DISTANCE IN WGANS

The main idea behind WGANs is to replace the traditional loss function used in GANs, which is based on Kullback-Liebler or Jensen-Shannon divergences, with the Wasserstein distance. The Wasserstein distance between two probability distributions P_r and P_g is defined as:

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|], \tag{8}$$

By using the Kantorovich-Rubinstein duality, the Wasserstein distance can be reformulated as:

$$W(P_r, P_g) = \sup_{\|h\|_L \leqslant 1} \mathbb{E}_{x\sim P_r}[h(x)] - \mathbb{E}_{x\sim P_g}[h(x)], \tag{9}$$

where the supremum is taken over all 1-Lipschitz functions.

3.2 WGAN ARCHITECTURE AND OBJECTIVE FUNCTION

For WGANs, we refer to the Discriminator D as the "critic". This is because unlike with GANs, which use the Discriminator to predict which of the generated images is sampled from the real data distribution P_r and which are from the generated data distribution P_g by choosing a value in $[0, 1]$, the critic estimates the Wasserstein distance between the generated image and the real (sampled) image.

The objective function for the WGAN can be written as:

$$\min_G \max_{D \in \mathcal{D}} \mathbb{E}_{x\sim P_r}[D(x)] - \mathbb{E}_{z\sim P_z}[D(G(z))] \tag{10}$$

where $\mathcal{D}$ is the set of all 1-Lipschitz functions, G is the generator, and D is the critic. z is sampled from prior distribution P_z (typically a Gaussian) and G is chosen so that $G(z) \sim P_g$. As in Kantorovich-Rubenstein duality, $\mathcal{D}$ is the set of 1-Lipschitz functions.

One way to interpret this loss function, is that the discriminator wants to maximise the Wasserstein distance between the generated data and the real data and at the same time the generator wants to minimise this distance.

3.3 ENFORCING THE LIPSCHITZ CONSTRAINT

To ensure the critic is 1-Lipschitz, WGANs enforce a constraint on the critic's weights. There are two common ways to enforce this constraint: weight clipping and gradient penalty.

3.3.1 Weight Clipping

The original WGAN paper proposed using weight clipping, which simply involves clipping the weights of the critic to a compact set $[-c, c]$. The set of functions satisfying this constraint is a subset of the set of k-Lipschitz functions for some k depending on c and the model architecture. The training procedure is then modified to include a weight clipping step after each update of the critic's weights. This method can be effective but may result in underfitting or other undesired behaviors.

3.3.2 Gradient Penalty

An alternative approach to enforcing the Lipschitz constraint is to use the gradient penalty. This method adds a regularization term to the critic's loss function, which penalizes deviations of the gradient norm from 1:

$$L_{GP} = \mathbb{E}\hat{x} \sim P\hat{x}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2], \tag{11}$$

where x is a point sampled uniformly along a straight line between a real data point and a generated data point, i.e., $x = \alpha x + (1 - \alpha)G(z)$ with $\alpha \sim U(0, 1)$ being a random interpolation coefficient.

The overall objective function for the WGAN with gradient penalty (WGAN-GP) becomes:

$$\min_G \max_D \mathbb{E}x \sim P_r[D(x)] - \mathbb{E}z \sim P_z[D(G(z))] + \lambda L_{GP}, \tag{12}$$

where λ is a hyperparameter that controls the strength of the gradient penalty. The gradient penalty approach has been shown to yield better results than weight clipping and is widely used in practice.

3.4 TRAINING WGANS

Training WGANs follows an alternating update strategy, similar to traditional GANs. However, there are a few key differences:

The critic is updated more frequently than the generator. This is usually done using a ratio, such as 5 critic updates for each generator update. This ensures that the critic provides a more accurate estimation of the Wasserstein distance before the generator is updated.

The learning rate is typically set lower than in traditional GANs, which helps improve the stability of the training process.

The use of batch normalization in the critic is discouraged, as it can interfere with the enforcement of the Lipschitz constraint. Layer normalization or other normalization techniques can be used instead.

3.5 PERFORMANCE GAINS OF WGANS

WGANs offer several performance gains over traditional GANs due to the use of the Wasserstein distance and the enforcement of the Lipschitz constraint. These improvements stem from the following aspects:

Smoother gradient landscape: The Wasserstein distance provides a smoother gradient landscape compared to KL or JS divergences. In traditional GANs, vanishing gradients can occur when the supports of the real and generated data distributions do not overlap or have minimal overlap. This can lead to mode collapse or instability during training. WGANs, on the other hand, provide more informative gradients that help the generator and critic learn more effectively, even when the supports do not overlap.

Meaningful loss function: The Wasserstein distance as a loss function has a more direct relationship with the quality of generated samples. As the Wasserstein distance decreases, the generated samples tend to improve visually and semantically. In contrast, the loss functions based on KL or JS divergences may not exhibit such a clear relationship with the quality of the generated samples.

Improved training stability: Enforcing the Lipschitz constraint on the critic helps prevent the critic from becoming too powerful, leading to a more balanced training process. In traditional GANs, the discriminator can become too strong, overpowering the generator and leading to instability during training. The improved stability in WGANs generally results in higher quality generated samples.

Robustness to network architecture: WGANs are more robust to the choice of network architecture and hyperparameters compared to traditional GANs. This property allows researchers to explore a wider range of architectures and settings, increasing the chances of discovering effective models.

3.6 IMPROVEMENTS AND ALTERNATIVES TO WGANS

While WGANs have made significant progress in addressing some of the challenges of training GANs, there is still room for improvement and exploration of alternative generative models.

Improved Lipschitz constraint enforcement: Current methods for enforcing the Lipschitz constraint, such as weight clipping and gradient penalty, have their limitations. Weight clipping can lead to underfitting, while gradient penalty adds an extra computational cost during training. Developing more efficient and effective methods for enforcing the Lipschitz constraint could further improve the performance of WGANs.

Conditional WGANs: Extending WGANs to conditional settings, where the generated samples are conditioned on additional input information, can enable more controlled generation of samples. This can be achieved by incorporating the conditioning information into both the generator and the critic.

Other GAN variants: There are numerous GAN variants that aim to address different aspects of the traditional GANs' limitations. Some notable examples include Spectral Normalization GANs (SN-GANs), which enforce Lipschitz constraint using spectral normalization, and Self-Attention GANs (SAGANs), which incorporate self-attention mechanisms to capture long-range dependencies within the data.

Diffusion models: Apart from GANs, diffusion models have emerged as another promising class of generative models. These models, such as denoising score matching and contrastive divergence, learn the data distribution by simulating a diffusion process that gradually adds noise to the data points. By learning to reverse this process, diffusion models can generate high-quality samples without the adversarial training dynamics of GANs.

Overall, the field of generative modeling is rapidly evolving, with WGANs being an important step towards more stable and effective generative models. Continued research on improving WGANs and exploring alternative gener

3.7 CONCLUSION

Wasserstein GANs address some of the key challenges associated with training traditional GANs by using the Wasserstein distance as the loss function. This leads to improved training stability, a more meaningful loss function, and a smoother gradient landscape, which in turn results in higher quality generated samples. By enforcing the Lipschitz constraint on the critic, either through weight clipping or gradient penalty, WGANs can achieve better performance than their traditional counterparts, making them a powerful tool for generative modeling tasks.