Deep Generative Models

Chapter 4: Generative Adversarial Networks

Ali Bereyhi

ali.bereyhi@utoronto.ca

Department of Electrical and Computer Engineering

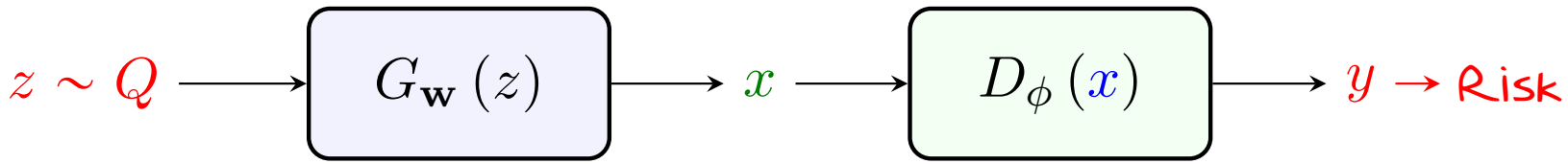
University of Toronto

Summer 2025

W-GAN: *Alternative Paradigm*

In general *a GAN can be built by two networks*

- *Generator that maps latent to data-space*
- *Discriminator that maps generated samples to information for learning*



To train this architecture, we can follow two paradigms

- ✔ *Vanilla GAN which is intuitively inline with a min-max classification game*
  - ↳ *It implicitly maximizes the likelihood ≡ minimizes the JS divergence*
- ➔ *Wasserstein GAN which is less intuitive*
  - ↳ *It implicitly minimizes the so-called Wasserstein distance*

Preliminaries: *Coupling Set*

To understand Wasserstein GAN paradigm: *we need to review a few basic definitions and get some feeling about them*

Coupling Set

Let P and $\hat{P}$ be two distributions defined on data space $\mathbb{X}$: the coupling set of P and $\hat{P}$ is defined as

$$\Pi_{P,\hat{P}} = \left\{ Q\left(x, \hat{x}\right) : \sum_{\hat{x}} Q\left(x, \hat{x}\right) = P\left(x\right) \text{ and } \sum_{x} Q\left(x, \hat{x}\right) = \hat{P}\left(\hat{x}\right) \right\}$$

The coupling set contains all joint distributions whose marginals are P and $\hat{P}$

- *Some of distributions have no correlation, e.g., $Q\left(x, \hat{x}\right) = P\left(x\right)\hat{P}\left(\hat{x}\right)$*
- *Some could be very correlated, e.g.,*
  - ↳ *If $P\left(x\right) = \hat{P}\left(\hat{x}\right)$, we could set $x = \hat{x}$, i.e., $Q\left(x, \hat{x}\right) = P\left(x\right)\delta\left(x - \hat{x}\right)$*

Wasserstein Distance

Wasserstein Distance

Wasserstein distance between P and $\hat{P}$ is defined as

$$D_{\mathrm{W}}\left(P \| \hat{P}\right) = \min_{Q \in \Pi_{P,\hat{P}}} \mathbb{E}_{(x,\hat{x})\sim Q}\left\{\|x - \hat{x}\|\right\}$$

Intuitively, Wasserstein distance does the following

1. *It looks into all possible distributions whose marginals look like P and $\hat{P}$*
2. *It finds the average distance between x and $\hat{x}$ for each distribution*
3. *It takes the smallest average distance as the distance between P and $\hat{P}$*
   - ↳ *Somehow the distance of most correlated case*

\+ *What is special about this metric?*

– *Let's see an example*

Wasserstein vs KL and JS: *Example*

Say we have latent variable z uniformly distributed on $(0, 1)$, i.e.,[1]

$$z \sim \text{Unif}\,(0, 1)$$

Data x is related to z as $x = [0, z]$

> *x lies on a 1D manifold in the 2D space!*

We have access to latent and the following generator

> *For some learnable $\theta \in [-1, 1]$, the generator generates $\hat{x}$ from z as*
>
> $$\hat{x} = G_\theta\,(z) = [\theta, z]$$

Let's see how various divergence metrics between x and $\hat{x}$ look like

[1] *This example is taken from Wasserstein GAN paper by M. Arjovsky et al.*

Wasserstein vs KL and JS: *Example*

Since $\hat{x} = [\theta, z]$ *for* $\theta \in [-1, 1]$ *and latent* $z \in [0, 1]$, *we have*

$$\mathbb{X} = [-1, 1] \times [0, 1]$$

We can also present the data and model distributions as

- *Data is uniformly distributed on the line* $x_1 = 0$

$$P(x) = \boxed{\delta(x_1)}$$

dirac impulse at $x_1 = 0$

- *Model output is uniformly distributed on the line* $x_1 = \theta$

$$\hat{P}_\theta(x) = \boxed{\delta(x_1 - \theta)}$$

dirac impulse at $x_1 = \theta$

Wasserstein vs KL and JS: *Example*

For KL divergence, we could write

$$D_{\mathrm{KL}}\left(P\|\hat{P}_\theta\right)=\mathbb{E}_{x\sim P}\left\{\log\frac{P\left(x\right)}{\hat{P}_\theta\left(x\right)}\right\}=\int_0^1\int_{-1}^1\log\frac{\delta\left(x_1\right)}{\delta\left(x_1-\theta\right)}\mathrm{d}x_1\mathrm{d}x_2$$

- *If* $\theta=0$*; then, KL divergence reads*

$$D_{\mathrm{KL}}\left(P\|\hat{P}_\theta\right)=\int_0^1\int_{-1}^1\delta\left(x_1\right)\log\frac{\delta\left(x_1\right)}{\delta\left(x_1\right)}\mathrm{d}x_1\mathrm{d}x_2=\int_0^1\log 1\mathrm{d}x_2=0$$

- *If* $\theta\neq 0$*; then, KL divergence reads*

$$D_{\mathrm{KL}}\left(P\|\hat{P}_\theta\right)=\int_0^1\int_{-1}^1\delta\left(x_1\right)\log\frac{\delta\left(x_1\right)}{\underbrace{\delta\left(x_1-\theta\right)}_{0\ \textit{for}\ [0^-,0^+]}}\mathrm{d}x_1\mathrm{d}x_2=+\infty$$

Wasserstein vs KL and JS: *Example*

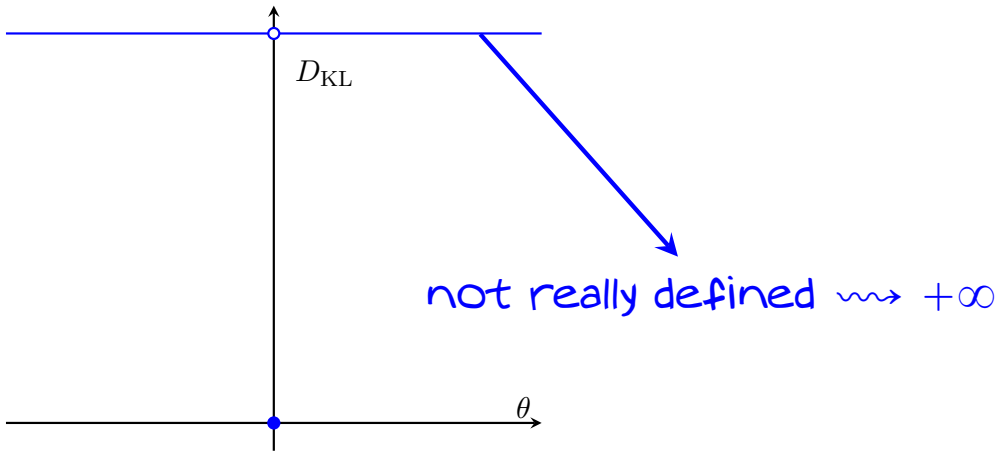
So, KL divergence reads

$$D_{\mathrm{KL}}\left(P \| \hat{P}_\theta\right) = \begin{cases} +\infty & \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$

which is minimized when

$$\textit{data distribution } P \equiv \textit{model distribution } \hat{P}_\theta$$

as we expected!



Wasserstein vs KL and JS: *Example*

For JS divergence, we have

$$D_{\mathrm{JS}}\left(P\|\hat{P}_\theta\right) = D_{\mathrm{KL}}\left(P\|A_{P,\hat{P}_\theta}\right) + D_{\mathrm{KL}}\left(\hat{P}\|A_{P,\hat{P}_\theta}\right)$$

The average distribution in this case reads

$$A_{P,\hat{P}_\theta} = \frac{\delta\left(x_1\right) + \delta\left(x_1 - \theta\right)}{2}$$

- *If $\theta = 0$; then, we have*

$$\begin{aligned} D_{\mathrm{KL}}\left(P\|A_{P,\hat{P}_\theta}\right) &= \int_0^1 \int_{-1}^1 \delta\left(x_1\right) \log \frac{2\delta\left(x_1\right)}{\delta\left(x_1\right) + \delta\left(x_1\right)} \mathrm{d}x_1 \mathrm{d}x_2 \\ &= \int_0^1 \log 1 \mathrm{d}x_2 = 0 \end{aligned}$$

Wasserstein vs KL and JS: *Example*

- *If $\theta \neq 0$; then, we have*

$$D_{\mathrm{KL}}\left(P \| A_{P,\hat{P}_\theta}\right) = \int_0^1 \int_{-1}^1 \delta\left(x_1\right) \log \frac{2\delta\left(x_1\right)}{\delta\left(x_1\right) + \underbrace{\delta\left(x_1 - \theta\right)}_{0 \text{ for } [0^-,0^+]}} \mathrm{d}x_1 \mathrm{d}x_2$$

$$= \log 2$$

We can show that $D_{\mathrm{KL}}\left(\hat{P}_\theta \| A_{P,\hat{P}}\right)$ reads the same, i.e.,

$$D_{\mathrm{KL}}\left(\hat{P}_\theta \| A_{P,\hat{P}_\theta}\right) = \begin{cases} \log 2 & \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$

Wasserstein vs KL and JS: *Example*

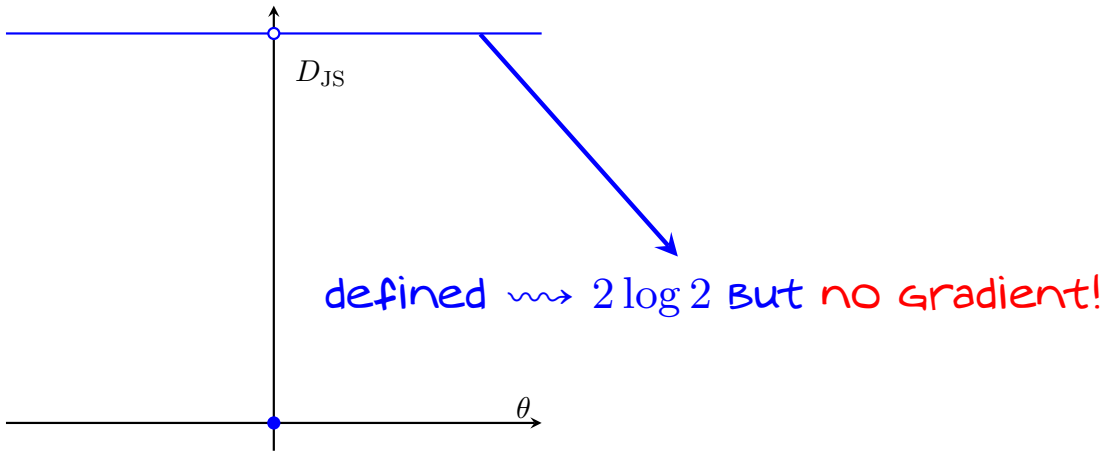
So, JS divergence reads

$$D_{\mathrm{JS}}\left(P \| \hat{P}_\theta\right) = \begin{cases} 2\log 2 & \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$

which is again minimized when

$$\textit{data distribution } P \equiv \textit{model distribution } \hat{P}_\theta$$

as we expected!



Wasserstein vs KL and JS: *Example*

For the Wasserstein distance, we have

$$D_{\mathrm{W}}\left(P\|\hat{P}_\theta\right) = \min_{Q\in\Pi_{P,\hat{P}}} \mathbb{E}_{(x,\hat{x})\sim Q}\left\{\|x-\hat{x}\|\right\}$$

Let us compute the expectation of an arbitrary Q

$$\mathbb{E}_{(x,\hat{x})\sim Q}\left\{\|[0,z]-[\theta,z]\|\right\} = \mathbb{E}_{(x,\hat{x})\sim Q}\left\{|\theta|\right\} = |\theta|$$

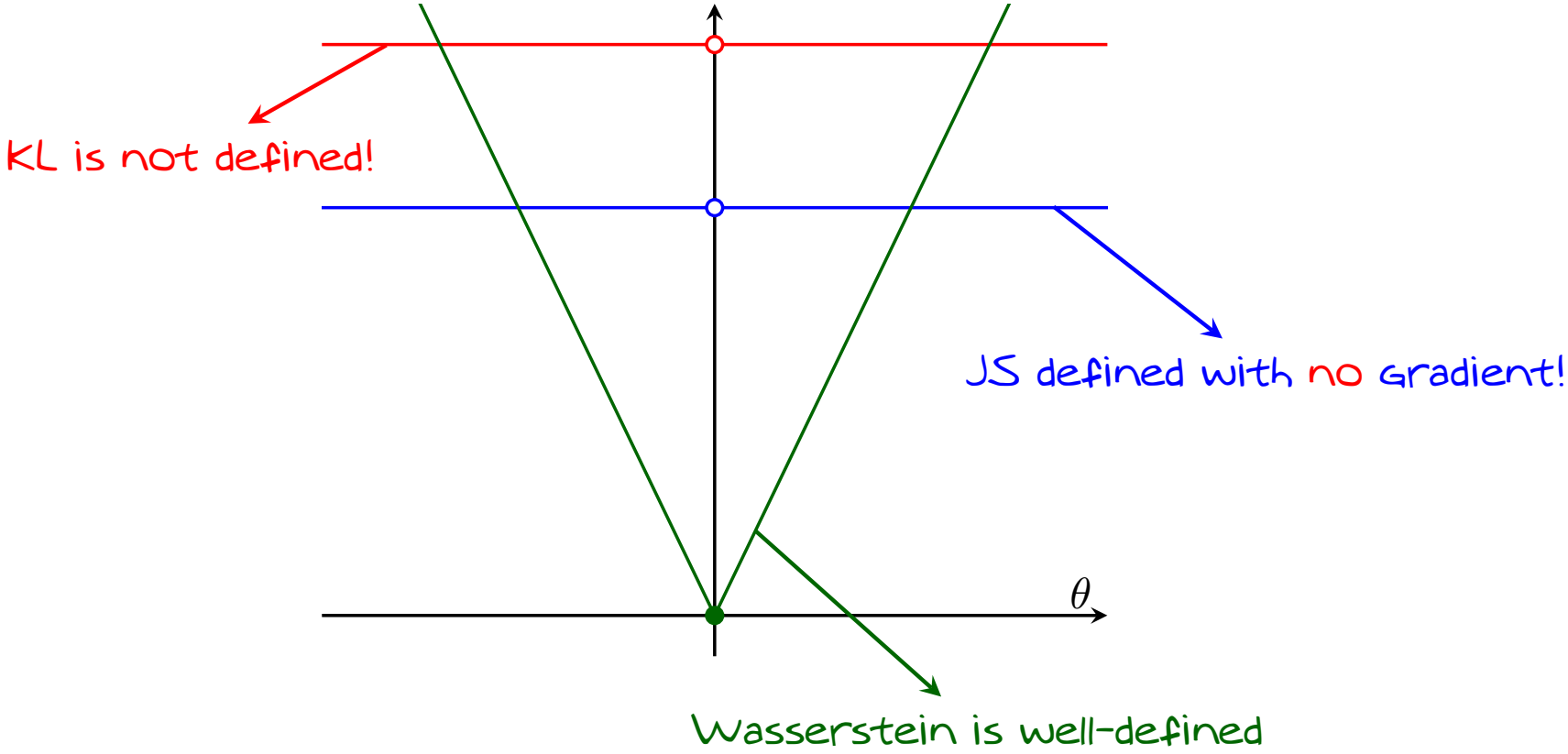
This is independent of Q; thus, we could say

$$D_{\mathrm{W}}\left(P\|\hat{P}_\theta\right) = |\theta|$$

which is again minimized when $P = \hat{P}_\theta$!

Wasserstein vs KL and JS: *Example*

All metrics are minimized when $P = \hat{P}_\theta$ *but look differently against* θ



Key Observations in Example

From the example, we could say: *minimizing all three metrics leads to*

model distribution $\hat{P}_\theta$ *converging to data distribution*

However, they exhibit different analytical behavior

- *KL not defined anywhere but at* $P = \hat{P}_\theta$
  - ↳ *It only returns value when we are exactly on data manifold*
- *JS defined everywhere but gives no* $\nabla_\theta D$ *to use for training*
- *Wasserstein is smooth giving* $\nabla_\theta D$ *everywhere*

Moral of Story

It might be better to minimize the Wasserstein distance between the data and generator distribution instead of KL

Implicit Distance Minimization

+ *But, we did not minimize JS explicitly! Our GAN training was intuitively derived, as we could not compute the generator distribution! Right?!*
- Yes! We did it implicitly with the help of a discriminator
+ *How can we minimize Wasserstein distance then? Shall we do this also implicitly?!*
- Right! We can do it using *Kantorovich-Rubinstein duality!*

Minimizing Wasserstein Distance

We concluded that *GAN implicitly learns*

$$\hat{P}^\star = \operatorname*{argmin}_{\hat{P}} D_{\mathrm{JS}}\left(P \| \hat{P}\right)$$

via generator G: let us know replace this by Wasserstein distance

$$\begin{aligned}\hat{P}^\star &= \operatorname*{argmin}_{\hat{P}} D_{\mathrm{W}}\left(P \| \hat{P}\right) \\ &= \operatorname*{argmin}_{\hat{P}} \min_{Q \in \Pi_{P,\hat{P}}} \mathbb{E}_{(x,\hat{x})\sim Q}\left\{\|x - \hat{x}\|\right\}\end{aligned}$$

This does not seem tractable!

- ↳ *It does not sound feasible to search $\Pi_{P,\hat{P}}$*
- ↳ *Kantorovich-Rubinstein duality lets us do this indirectly*

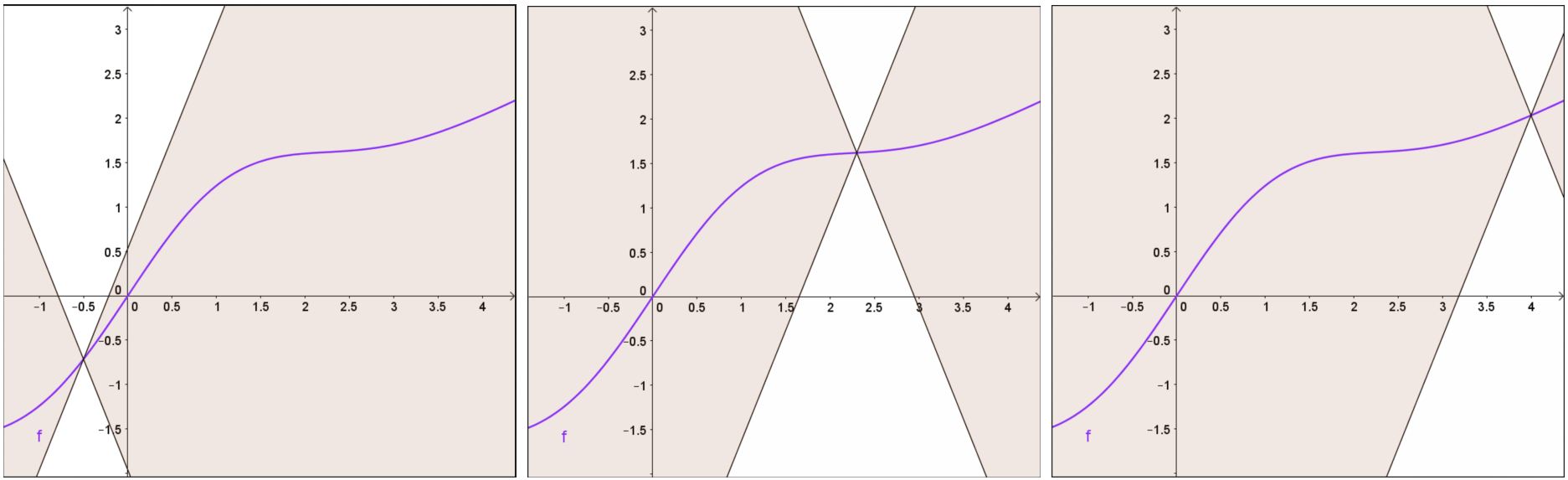
Lipschitz Continuity

L-Lipschitz Function

$f : \mathbb{X} \mapsto \mathbb{R}^m$ *is Lipschitz if for any* x *and* $\hat{x} \in \mathbb{X}$

$$\|f(x) - f(\hat{x})\| \leqslant L\|x - \hat{x}\|$$

with L *being the Lipschitz constant*



Lipschitz Function Space

L-Lipschitz Function Space

The set of all functions $f : \mathbb{X} \mapsto \mathbb{R}^m$ *that are* L*-Lipschitz*

$$\mathbb{L}_L = \{f : f \text{ is } L\text{-Lipschitz}\}$$

! Attention

In practice, we work with computational models of form

$$f_{\mathbf{w}} : \mathbb{X} \mapsto \mathbb{R}^m$$

for such models, we have a set of parameters $\mathbb{W}_L$ *such that*

$$\text{for all } \mathbf{w} \in \mathbb{W}_L \rightsquigarrow f_{\mathbf{w}} \in \mathbb{L}_L$$

Minimizing Wasserstein Distance

Kantorovich-Rubinstein Duality

Wasserstein distance between P and $\hat{P}$ is computed as

$$D_{\mathrm{W}}\left(P \| \hat{P}\right) = \max_{D \in \mathbb{L}_1} \mathbb{E}_{x \sim P}\left\{D\left(x\right)\right\} - \mathbb{E}_{x \sim \hat{P}}\left\{D\left(x\right)\right\}$$

Kantorovich-Rubinstein Duality: *Computational Modeling*

Wasserstein distance between P and $\hat{P}$ is estimated by model D_ϕ as

$$\hat{D}_{\mathrm{W}}\left(P \| \hat{P}\right) = \max_{\phi \in \Phi_1} \hat{\mathbb{E}}_{x \sim P}\left\{D_\phi\left(x\right)\right\} - \hat{\mathbb{E}}_{x \sim \hat{P}}\left\{D_\phi\left(x\right)\right\}$$

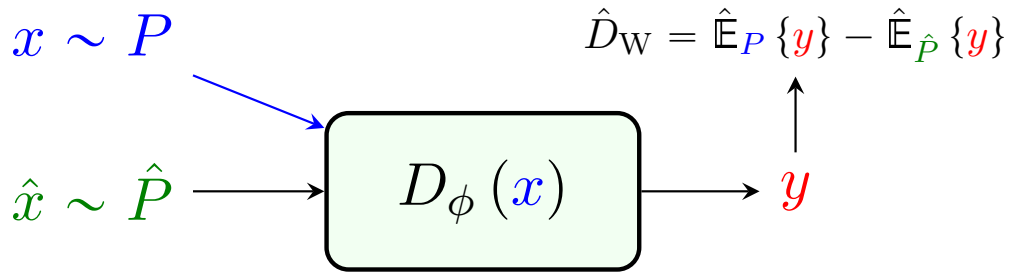
with Φ_1 is the set of parameters with which D_ϕ is 1-Lipschitz

Minimizing Wasserstein Distance

Back to our problem: *we want to minimize Wasserstein distance*

$$\begin{aligned}\hat{P}^{\star} &= \underset{\hat{P}}{\operatorname{argmin}}\, D_{\mathrm{W}}\left(P \| \hat{P}\right) \\ &= \underset{\hat{P}}{\operatorname{argmin}} \max_{D \in \mathbb{L}_1} \mathbb{E}_{x \sim P}\left\{D\left(x\right)\right\} - \mathbb{E}_{x \sim \hat{P}}\left\{D\left(x\right)\right\}\end{aligned}$$

We can estimate it by learning a computational discriminator D_ϕ



as long as we search over Φ_1 to make sure it's 1-Lipschitz

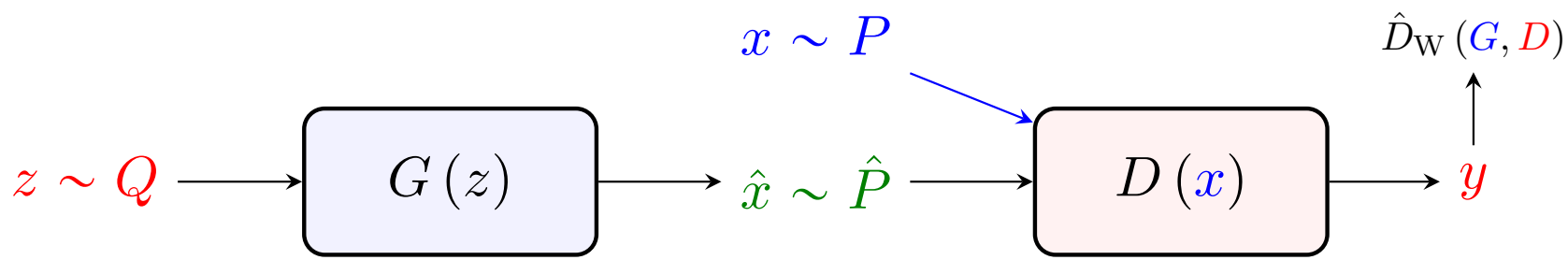
Minimizing Wasserstein Distance

+ *But, we need to do this implicitly! Right?!*
– Exactly! We can use a generator

Sample $z \sim Q(z)$ and let $\hat{x}G(z) \sim \hat{P}$: we find $\hat{P}^\star$ implicitly as

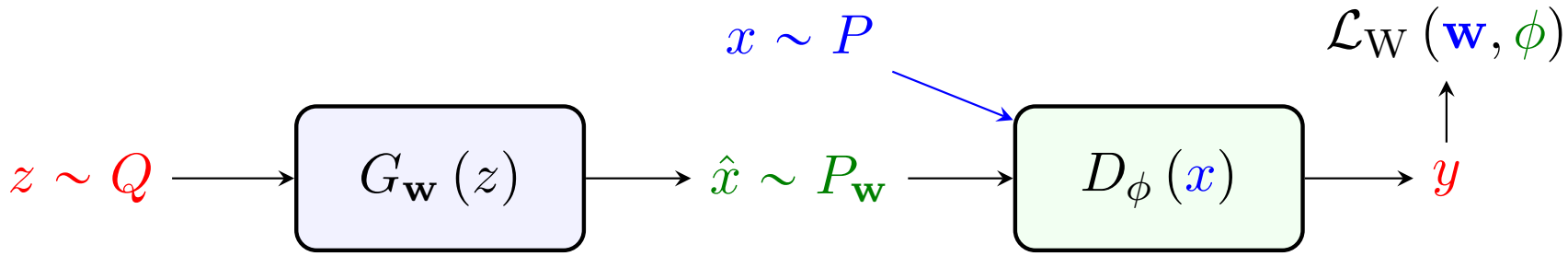
$$G^\star = \underset{G}{\operatorname{argmin}} \max_{D \in \mathbb{L}_1} \mathbb{E}_{x \sim P}\{D(x)\} - \mathbb{E}_{z \sim Q}\{D(G(z))\}$$

***Bingo!** This is a GAN with alternative min-max objective*



This is what we call Wasserstein GAN

Wasserstein GAN: *Alternative Paradigm to Train GAN*



Wasserstein GAN *(Computational)*

Consider a GAN with generator $G_{\mathbf{w}}$ and discriminator D_{ϕ} which is 1-Lipschitz for $\phi \in \Phi_1$: Wasserstein GAN learns through the min-max game

$$\min_{\mathbf{w}} \max_{\phi \in \Phi_1} \mathcal{L}_{\mathrm{W}}(\mathbf{w}, \phi)$$

with objective function

$$\mathcal{L}_{\mathrm{W}}(\mathbf{w}, \phi) = \hat{\mathbb{E}}_{x \sim P}\{D_{\phi}(x)\} - \hat{\mathbb{E}}_{z \sim Q}\{D_{\phi}(G_{\mathbf{w}}(z))\}$$

W-GAN: *Training Loop*

`Train_WGAN(`$\mathbb{D}$`:dataset):`

Initiate the generator $G_{\mathbf{w}}$ *and discriminator* D_{ϕ} *with some* $\mathbf{w}$ and ϕ
for *multiple epochs* **do**
  Sample a batch of data samples $\left\{x^j : j = 1, \ldots, n\right\}$ *from* $\mathbb{D}$
  Sample latents $\left\{z^j : j = 1, \ldots, n\right\}$ *using* Q *and compute* $\hat{x}^j \leftarrow G_{\mathbf{w}}\left(z^j\right)$
  for $\xi = 1, \ldots, \Xi$ **do**
    for $j = 1, \ldots, n$ **do**
      Compute $\mathcal{L}^j = D_{\phi}\left(x^j\right) - D_{\phi}\left(\hat{x}^j\right)$
      Backpropagate over discriminator to compute $\nabla_{\phi}\mathcal{L}^j$
      if $\xi = \Xi$ **then** *Backpropagate over generator to compute* $\nabla_{\mathbf{w}}\mathcal{L}^j$
    end for
    Update ϕ *using* `Opt_avg` $\left\{\nabla_{\phi}\mathcal{L}^j\right\}$
  end for
  Update $\mathbf{w}$ *using* `Opt_avg` $\left\{\nabla_{\mathbf{w}}\mathcal{L}^j\right\}$
end for
return *trained generator* $G_{\mathbf{w}}$

Wasserstein vs Vanilla GAN

Looking at Wasserstein GAN objective

$$\min_{\mathbf{w}} \max_{\phi \in \Phi_1} \hat{\mathbb{E}}_{x \sim P} \left\{ D_\phi \left(x \right) \right\} - \hat{\mathbb{E}}_{z \sim Q} \left\{ D_\phi \left(G_{\mathbf{w}} \left(z \right) \right) \right\}$$

and comparing it with vanilla GAN

$$\min_{\mathbf{w}} \max_{\phi} \hat{\mathbb{E}}_{x \sim P} \left\{ \log D_\phi \left(x \right) \right\} + \hat{\mathbb{E}}_{z \sim Q} \left\{ \log \left(1 - D_\phi \left(G_{\mathbf{w}} \left(z \right) \right) \right) \right\}$$

we can say

1. *Objective does not use cross-entropy anymore, i.e., no* $\log$
2. D_ϕ *in W-GAN does not need to be Sigmoid output anymore*
3. D_ϕ *in W-GAN should be kept* 1*-Lipschitz*

Imposing Lipschitz Continuity

There are various approaches to *restrict discriminator being Lipschitz*

1. *Weight clipping which is the simplest approach*
   - ↳ *We clip the weights periodically throughout training*
   - ↳ *This way we restrict the variation of the model output*
2. *Gradient penalty which is more sophisticated*
   - ↳ *We add* $\left(\|\nabla_x D(x)\| - 1\right)^2$ *as penalty*
   - ↳ *This assures that output variation does not get extremely large*
3. *Spectral normalization which is again an advanced approach*
   - ↳ *We normalize each layer by its spectral norm*

$$\mathbf{W}_\ell \leftarrow \frac{\mathbf{W}_\ell}{\max\left\{\texttt{eig}\left(\mathbf{W}_\ell\right)\right\}}$$

   - ↳ *This way we assure the boundedness of the weights*