UL : GANs & VAEs

In previous chapters, we discussed three applications of unsupervised learning : density estimation, clustering, and dimension reduction (PCA , random projections). The goal of the unsupervised learning task in this chapter is of a different nature: given an (unlabelled) sample $\mathcal{L}_n = \{ X_1, .., X_n \}$, $X_i \sim \mathbb{P}$ iid, generate new observations that look alike.

↑ Generally, X_i are images, and the task is to generate new images that look like the ones in the training set.

x Remark : In density estimation, the training examples are used to construct an estimate of the unknown distribution $\mathbb{P}$.

↗ Either non-parametrically (e.g. using kernels), or parametrically (give yourself a parametric familly $\mathbb{P}_\theta$ ($\theta \in \Theta$) of distribution, & estimate θ using $\mathcal{L}_n$ (e.g. MLE).

↑

Here, we are not necessarily interested in learning the model. This can be done implicitely. We are mostly interested in generating new (realistic) samples from the unknown distribution.

We discuss two popular approaches : Generative Adversarial Networks (GANs), and Variational Auto-Encoders (VAEs).

I - GENERATIVE ADVERSARIAL NETWORKS (GANs).

GANs ≡ Two-player game :

↙ Generator: Creates images (i.e. Xs) as realistic as possible

↑ Using a latent representation:

↘ Discriminator: Determines if the images presented are real (X produced by the true unknown underlying distribution $\mathbb{P}$) or fake (i.e. produced by our model).

$z \sim \mathbb{P}_z$ ← e.g. uniform / multivariate Gaussian (arbitrary, but fixed)

↓

Non-linear / smooth transform $X := G(z) \sim \mathbb{P}_g$

↗ The distribution of X, induced by G.

[if $\mathbb{P}_z$ & $\mathbb{P}_g$ are AC with densities $f_z(z)$ & $f_g(x)$, & if the transform G is invertible, then

$$f_g(x) = f_z(g^{-1}(x)) \left| \frac{\partial g^{-1}(x)}{\partial x} \right|$$

= change of variable formula]

• Formally, the generator is a function G, parametrized by θ_g , taking z (≡ noise) as an input, producing a candidate image $x = G(z)$. The goal of the generator is to adjust θ_g so that x is as realistic as possible, fooling the discriminator. The discriminator is a function D, parametrized by θ_d. The goal of the discriminator is to correctly classify real images from fake ones, by fine tuning its parameters θ_d.

A binary classification task : label images as $+1$ (real) or 0 (fake).

- Both players have a cost function to minimize :

  Discriminator wishes to minimize $J^{(d)}(\theta_d, \theta_g)$

  & Generator wishes to minimize $J^{(g)}(\theta_d, \theta_g)$

  Several cost functions may be considered to train GANs

- Cost function for the discriminator

  Preliminaries = back to the basics : loss function for logistic regression [A binary classification problem]

  We make use of the notation from the chapters MS : MLE and SL : LINEAR CLASSIFICATION.

  ↳ $\mathcal{L}_n = \{(X_1, Y_1), \dots, (X_n, Y_n)\}$ $\quad Y_i \in \{0, 1\}$

  ↳ Model is $\mathbb{P}_{\beta_0,\beta}(Y=1 \mid X=x) = \sigma(\beta_0 + \beta^t x)$

  discriminative : model only the conditional distribution of $Y \mid X = x$. $\quad \in \mathbb{R}^d \quad$ sigmoid

  ↳ Fit the model using Maximum likelihood = minimizing the KL divergence between the empirical distribution $\hat{\mathbb{P}}_n$ (placing a weight $1/n$ on each observation) & $\mathbb{P}_{\beta_0,\beta}$

- $$KL(\hat{\mathbb{P}}_n \,\|\, \mathbb{P}_{\beta_0,\beta}) = \sum_{(x,y)\in\mathcal{L}_n} \frac{1}{n} \log\left\{\frac{1/n}{\mathbb{P}_{\beta_0,\beta}(y|x)}\right\}$$

  conditional distribution

- Put $H(\hat{\mathbb{P}}_n) = -\frac{1}{n}\sum_{i=1}^n \log\frac{1}{n} = \log n$

  $=$ entropy of the uniform distribution on $\mathcal{L}_n$,

- $$KL(\hat{\mathbb{P}}_n \,\|\, \mathbb{P}_{\beta_0,\beta}) + H(\hat{\mathbb{P}}_n)$$
  $$= -\frac{1}{n}\sum_{i=1}^n \log\left\{\mathbb{P}_{\beta_0,\beta}(Y_i = y_i \mid X_i = x_i)\right\}$$
  $$= -\frac{1}{n}\sum_{i=1}^n \log\left\{\sigma_i^{y_i}(1-\sigma_i)^{1-y_i}\right\},$$

  where $\sigma_i := \sigma(\beta_0 + \beta^t x_i)$.

$\Rightarrow$

$$KL(\hat{\mathbb{P}}_n \,\|\, \mathbb{P}_{\beta_0,\beta}) + H(\hat{\mathbb{P}}_n)$$
$$\underbrace{\boxed{= -\frac{1}{n}\sum_{i=1}^n y_i \log \sigma_i - \frac{1}{n}\sum_{i=1}^n (1-y_i)\log(1-\sigma_i)}}_{\overset{!!}{H(\hat{\mathbb{P}}_n,\, \mathbb{P}_{\beta_0,\beta})}}$$

a.k.a. the cross-entropy between $\hat{\mathbb{P}}_n$ & $\mathbb{P}_{\beta_0,\beta}$.

The cross-entropy consists of two terms :

↳ $\sum_{i \mid y_i = +1} \log\{\sigma(\beta_0 + \beta^t x_i)\}$ = term corresponding to observations labelled $+1$ (i.e. real images in our context)

↳ $\sum_{i \mid y_i = 0} \log\{1 - \sigma(\beta_0 + \beta^t x_i)\}$ = term corresponding to fake images.

In LR, select (β_0, β) such that $\sigma(\beta_0 + \beta^t x)$ is as large as possible for observations x labelled as $+1$,

i.e. close to 1 (recall its probabilistic meaning)

and such that

$\sigma(\beta_0 + \beta^t x) \approx 0$ for observations x labelled as 0.

Inspired by this, consider the following cost function for the discriminator:

DISCRIMINATOR COST FUNCTION

$$\boxed{J^{(d)}(\theta_d, \theta_g) := -\frac{1}{m} \sum_{i=1}^{m} \left\{ \log D(x_i) + \log[1 - D(G(z_i))] \right\}}$$

minibatch of examples

- parametrized by θ_d.
- $x_1, .., x_m$ are a minibatch of true images, subset of $\mathcal{L}_n$
- The discriminator selects θ_d to make $D(x_i)$ as large as possible, i.e. $D(x_i)$ close to 1.

Typically, the discriminator is a deep net.

- $G(z_i)$ are fake images, generated from the noise prior $\mathbb{P}_z$.

The discriminator selects θ_d to make $D(G(z_i))$ close to 0.

↳ With smooth differentiable function D (& G), we gradient ascent to update the parameter θ_d. During this step, the generator (i.e. θ_g) is kept fixed.

The generator then updates its parameters (θ_g) to make $D(G(z_i))$ as close to 1 as possible (θ_d fixed).

x Remark: samples $x_1, .., x_m$ are generated from $\mathbb{P}$
$z_1, .., z_m$ — " — $\mathbb{P}_z$.

⇒ The discriminator cost function is the sample version of the theoretical cost $\mathbb{E}_X\{\log D(X)\} + \mathbb{E}_Z\{\log(1 - D(G(Z)))\}$ $(\times -1)$

- Cost function for the generator.

Several variants exist:

↳ minimax / zero-sum game: the simplest version:
consider $J^{(g)} = -J^{(d)}$ ← zero-sum game

⇒ Consider the value function $V(\theta_d, \theta_g) = -J^{(d)}(\theta_d, \theta_g)$

Then the optimal parameter θ_g^* satisfies:

$$\theta_g^* = \arg\min_{\theta_g} \max_{\theta_d} V(\theta_d, \theta_g)$$

The generator minimizes the log-proba of the discriminator being correct. (theoretically motivated).

↳ Alternatively, we may consider

$$J^{(g)}(\theta_d, \theta_g) = -\frac{1}{2} \mathbb{E}_z\{\log D(G(z))\}$$

(or its empirical version)

The generator maximizes the log-proba of the discriminator being mistaken.
(heuristically motivated)

↳ And more .../...

- A general minibatch SGD algorithm for training GANs is presented on the next page.

SGD for training GANs

```
• Repeat "until convergence"
• For k steps
    – Sample a minibatch $z_1, .., z_m \sim \mathbb{P}_Z$ of size $m$
    – Sample a minibatch $x_1, .., x_m \sim$ from $\mathcal{L}_n$
    – Update $\theta_d$ by ascending its gradient
        $\nabla_{\theta_d} \left\{ \frac{1}{m} \sum_{i=1}^{m} \left( \log D(x_i) + \log[1 - D(G(z_i))] \right) \right\}$
  End For
• Sample a minibatch $z_1, .., z_m \sim \mathbb{P}_Z$
• Update $\theta_g$ by descending its gradient
        $\nabla_{\theta_g} \left\{ \frac{1}{m} \sum_{i=1}^{m} \log[1 - D(G(z_i))] \right\}$
```

• Some theoretical results.

Value function $V(\theta_d, \theta_g) = \mathbb{E}_X\{\log D(X)\} + \mathbb{E}_Z\{\log(1 - D(G(Z)))\}$

$$V(\theta_d, \theta_g) = \int \log D(x) f(x)\, dx + \int \log(1 - D(G(z))) f_Z(z) d$$

Assuming that $\mathbb{P}$ is AC with density f
$\mathbb{P}_Z$ is AC —"— f_Z.

Change of variable formula

$$= \int \log D(x) f(x) + \int \log(1 - D(x)) f_g(x)\, dx$$

$$V(\theta_d, \theta_g) = \int \{ \overset{\uparrow}{f(x)} \log D(x) + \overset{\uparrow}{f_g(x)} \log(1 - D(x)) \} dx$$

$\forall x \in (\operatorname{supp} f) \cap (\operatorname{supp} f_g)$, we see that
$u \mapsto a \log u + b \log(1-u)$ $a, b \neq 0$
achieves its maximum in $[0,1]$ at $\frac{a}{a+b}$

$\Rightarrow$ For a given generator G, the optimal discriminator D^*_G is given by $\boxed{D^*_G(x) = \frac{f(u)}{f(u) + f_g(u)}}$.

Put
$$\begin{aligned}
C(G) &:= \max_D V(G, D) \\
&= V(G, D^*_G) \\
&= \mathbb{E}_X\{\log D^*_G(X)\} + \mathbb{E}_Z\{\log(1 - D^*_G(G(Z)))\} \\
&= \mathbb{E}_{\mathbb{P}}\{\log D^*_G(X)\} + \mathbb{E}_{\mathbb{P}_g}\{\log(1 - D^*_G(X))\} \\
&= \mathbb{E}_{\mathbb{P}}\left\{\log\left(\frac{f(X)}{f(X) + f_g(X)}\right)\right\} \\
&\quad + \mathbb{E}_{\mathbb{P}_g}\left\{\log\left(\frac{f_g(X)}{f(X) + f_g(X)}\right)\right\} \\
&= \mathbb{E}_{\mathbb{P}}\left\{\log\left(\frac{f(X)/2}{[f(X) + f_g(X)]/2}\right)\right\} \\
&\quad + \mathbb{E}_{\mathbb{P}_g}\left\{\log\left(\frac{f_g(X)/2}{[f(X) + f_g(X)]/2}\right)\right\}
\end{aligned}$$

$X \sim \mathbb{P} \rightarrow$ real
$X \sim \mathbb{P}_g \rightarrow$ fake

$$\Rightarrow \mathcal{C}(G) = -\log 4 + \underbrace{KL\left(\mathbb{P} \,\Big\|\, \frac{\mathbb{P}+\mathbb{P}_g}{2}\right) + KL\left(\mathbb{P}_g \,\Big\|\, \frac{\mathbb{P}+\mathbb{P}_g}{2}\right)}_{\text{definition of the Jensen - Shannon divergence } JS(\mathbb{P},\, \mathbb{P}_g).}$$

We obtain $$\boxed{\mathcal{C}(G) = -\log 4 + \underbrace{JS(\mathbb{P},\, \mathbb{P}_g)}_{\text{Always non-negative unless } \mathbb{P}_g = \mathbb{P}}}$$

$\Rightarrow$ The global minimum of $\mathcal{C}(G)$ is achieved iff $\mathbb{P}_g = \mathbb{P}$ & the value of $\mathcal{C}(G)$ at that point is $-\log 4$.

[Ref] Ian J. Goodfellow & al (2014). Generative Adver-sarial Nets. NIPS.

II_ VARIATIONAL AUTO-ENCODERS (VAEs)

VAEs are another popular method for generating new samples from existing ones. The cost function to optimize lies at the heart of variational Bayesian methods. To better understand the optimization problem of VAEs, we make a short digression and discuss the inference problems of VARIATIONAL BAYES & EXPECTATION PROPAGATION.

II.1. KL and Reverse KL.

- Preliminaries: we discuss first the simpler approximation problem of a general distribution with a factorized one. We illustrate the technique on the bivariate normal distribution, which will highlight the main differences between two approximating techniques: minimization of the KL divergence, and minimization of the reversed KL.

- Reference distribution: $p(x) = \mathcal{N}(x \mid \mu, \Lambda^{-1})$

$x = (x_1, x_2)^t \in \mathbb{R}^2$

$\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$ mean vector

precision matrix $\Lambda = \Sigma^{-1}$

Σ = covariance matrix

$\Lambda = \begin{pmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{pmatrix}$

with $\Lambda_{12} = \Lambda_{21}$

$$p(x) = \frac{|\Lambda|^{1/2}}{2\pi} \exp\left\{-\frac{1}{2}(x-\mu)^t \Lambda (x-\mu)\right\}$$

- Approximating distribution: $q(x) = q_1(x_1)\, q_2(x_2)$

Factorizes as a product of two factors. We do not specify a parametric family for q_1 & q_2.

- Idea: Minimize $KL(q \,\|\, p) = \int q(x) \log\left(\frac{q(x)}{p(x)}\right) dx$

$$KL(q\|p) = \int \big(q(x) \log q(x) - q(x) \log p(x)\big)\, dx$$

$$KL(q\|p) = \int \Big\{ q_1(x_1) q_2(x_2) \big[\log q_1(x_1) + \log q_2(x_2) \big] - q_1(x_1) q_2(x_2) \log p(x) \Big\} dx_1\, dx_2$$

$$= \left(\int q_1(x_1) \log q_1(x_1)\, dx_1 \right) \underbrace{\left(\int q_2(x_2)\, dx_2 \right)}_{=1}$$

$$+ \left(\int q_2(x_2) \log q_2(x_2)\, dx_2 \right) \underbrace{\left(\int q_1(x_1)\, dx_1 \right)}_{=1}$$

$$- \int q_1(x_1) \Big(\underbrace{\int q_2(x_2) \log p(x)\, dx_2}_{\text{write this term } \ln \tilde{p}_1(x_1) \,=\, \mathbb{E}_{q_2}\{\log p(x_1, X_2)\}} \Big) dx_1$$

⇒ When minimizing $KL(q\|p)$ with respect to q_1, the quantity to minimize is

$$\int q_1(x_1) \log q_1(x_1)\, dx_1 - \int q_1(x_1) \log \tilde{p}_1(x_1)\, dx_1 \quad (+\text{ cste})$$

$$= \int q_1(x_1) \log \left(\frac{q_1(x_1)}{\tilde{p}_1(x_1)} \right) dx_1 = KL(q_1 \| \tilde{p}_1). \quad (+\text{ cste})$$

The minimum is achieved at $q_1^* = \tilde{p}_1$

$$\Rightarrow \log q_1^*(x_1) = \mathbb{E}_{q_2^*}\{\log p(x_1, X_2)\} + \text{cste}$$

so that $\int q_1^* = 1$

Likewise, by symmetry, q_2^* is given by:

$$\log q_2^*(x_2) = \mathbb{E}_{q_1^*}\{\log p(X_1, x_2)\} + \text{cste}.$$

The so-called MEAN-FIELD EQUATIONS.

The mean-field equations are coupled, and in general must be solved numerically & iteratively (e.g. via Coordinate Descent). In the simple problem of approximating a bivariate normal distribution with a product of densities, there exists an analytical solution.

↳ $p(x_1, x_2) = \mathcal{N}(x \mid \mu, \Lambda^{-1})$

Thus

$$\log q_1^*(x_1) = \mathbb{E}_{q_2^*}\Big\{ -\tfrac{1}{2}(x_1-\mu_1)^2 \Lambda_{11} - (x_1-\mu_1)\Lambda_{12}(X_2-\mu_2) \Big\} + \text{constant}$$

$$= -\tfrac{1}{2} x_1^2 \Lambda_{11} + x_1 \mu_1 \Lambda_{11} - x_1 \Lambda_{12} (\mathbb{E}_{q_2^*} X_2 - \mu_2) + C$$

quadratic expression in x_1
⇒ q_1^* must be normal.

completing the squares

$$= -\tfrac{1}{2} \Lambda_{11} x_1^2 + x_1 \Lambda_{11} \underbrace{\left(\mu_1 - \Lambda_{11}^{-1} \Lambda_{12} [\mathbb{E} X_2 - \mu_2] \right)}_{\text{Denote this term } m_1} + C$$

$$= -\tfrac{1}{2} (x_1 - m_1)^2 \Lambda_{11} + C$$

⇒ $q_1^*(x_1) \sim \mathcal{N}(x_1 \mid m_1, \Lambda_{11}^{-1})$
with $m_1 = \mu_1 - \Lambda_{11}^{-1} \Lambda_{12} (\mathbb{E}_{q_2^*} X_2 - \mu_2)$

& similarly:

$q_2^*(x_2) \sim \mathcal{N}(x_2 \mid m_2, \Lambda_{22}^{-1})$
with $m_2 = \mu_2 - \Lambda_{22}^{-1} \Lambda_{21} (\mathbb{E}_{q_1^*} X_1 - \mu_1)$

Since $\mathbb{E}_{q_j^*} X_j = m_j$, plugging the expression of m_1 back into m_2 yields:

$$m_2 = \mu_2 - \Lambda_{22}^{-1}\Lambda_{21}\left(\cancel{\mu_1} - \Lambda_{11}^{-1}\Lambda_{12}(m_2-\mu_2) - \cancel{\mu_1}\right)$$
$$= \mu_2 + \Lambda_{22}^{-1}\Lambda_{21}\Lambda_{11}^{-1}\Lambda_{12}(m_2-\mu_2)$$

i.e. $m_2(1-\Lambda_{22}^{-1}\Lambda_{21}\Lambda_{11}^{-1}\Lambda_{12}) = \mu_2(1-\Lambda_{22}^{-1}\Lambda_{21}\Lambda_{11}^{-1}\Lambda_{12})$

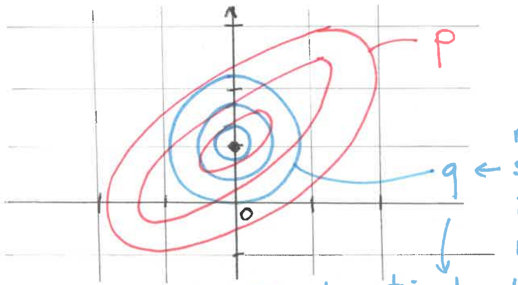
Provided p is non singular,
$|\Lambda| = \Lambda_{11}\Lambda_{22} - \Lambda_{12}\Lambda_{21} \neq 0$ ie $\Lambda_{22}^{-1}\Lambda_{21}\Lambda_{11}^{-1}\Lambda_{12} \neq 1$,
& the unique solution to this equation is $m_2 = \mu_2$.
Likewise, we obtain $m_1 = \mu_1$.

We conclude that the function $q(x_1,x_2) = q_1(x_1)\,q_2(x_2)$ which minimizes $KL(q\|p)$ for $p(x) \sim \mathcal{N}(x|\mu,\Lambda^{-1})$ is given by $q^*(x) = q_1^*(x_1)\,q_2^*(x_2)$,

$$q_1^*(x_1) \sim \mathcal{N}(x_1 \mid \mu_1, \Lambda_{11}^{-1})$$
$$q_2^*(x_2) \sim \mathcal{N}(x_2 \mid \mu_2, \Lambda_{22}^{-1})$$

Ex: Take $\mu = \begin{pmatrix}0\\1\end{pmatrix}$, $\Sigma = \begin{pmatrix}1 & 0.9\\0.9 & 1\end{pmatrix}$, $\Lambda = \frac{1}{1-0.9^2}\begin{pmatrix}1 & -0.9\\-0.9 & 1\end{pmatrix}$

$\Lambda_{11}^{-1} = \Lambda_{22}^{-1} = 0.19$



$q \leftarrow$ most of the support of q is contained in the region where p has most of its mass.

q underestimates the variance of p.

• Alternatively, minimize $KL(p\|q) = \int p(x)\log\frac{p(x)}{q(x)}\,dx$

The roles of p & q are reversed.

$$KL(p\|q) = -\int p(x)\left[\log q_1(x_1) + \log q_2(x_2)\right]dx + \text{cste}.$$

Minimization of this quantity wrt q_1, under the constraint $\int q_1(x_1)\,dx_1 = 1$ $\equiv$ minimization of the Lagrangian

$$\mathcal{L}(q_1) = -\int p(x)\log q_1(x_1)\,dx + \lambda\left(\int q_1(x_1)\,dx_1 - 1\right)$$

Lagrangian parameter

$$= -\int \underbrace{\left(\int p(x_1,x_2)\,dx_2\right)}_{F_1(x_1)} \log q_1(x_1)\,dx_1 + \lambda\left(\int q_1(x_1)\,dx_1 - 1\right)$$

Taking the functional derivative of $\mathcal{L}(q_1)$ wrt q_1 & setting the derivative to zero gives:

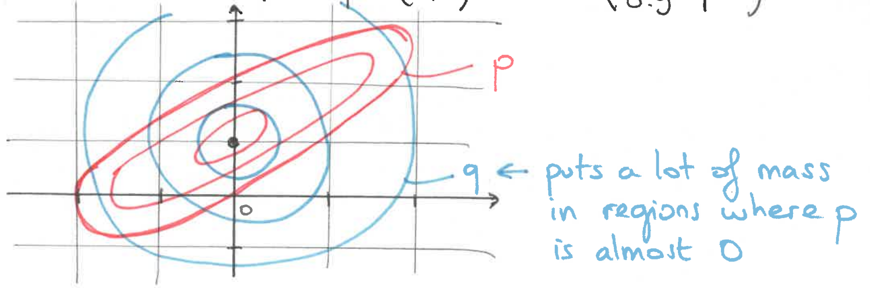
$-\dfrac{F_1(x_1)}{q_1(x_1)} + \lambda = 0$ i.e. $\lambda\, q_1(x_1) = F_1(x_1)$
$\Rightarrow \lambda = 1$

Thus, we obtain the solution
$$\boxed{\begin{aligned} q_1^*(x_1) &= \int f(x_1,x_2)\,dx_2 \\ q_2^*(x_2) &= \int f(x_1,x_2)\,dx_1 \end{aligned}}$$

the marginal densities.

Back to the bivariate approx: $q_j^*(x_j) \sim \mathcal{N}(x_j \mid \mu_j, \Sigma_{jj})$

Back to our example, $\mu = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, $\Sigma = \begin{pmatrix} 1 & 0.9 \\ 0.9 & 1 \end{pmatrix}$,



$\Rightarrow$ Minimizing $KL(q\|p)$ or $KL(p\|q)$ yields two different approximates, with different behaviour. The difference between these two results can be understood more generally.

(i) $\underset{q}{\min}$ $\underline{KL(q\|p)}$ (expectation with respect to q)

- regions in the support of p can be ignored if q puts zero mass on them.
- choose a q that avoids regions where p is small, since otherwise the contributing term $\log \frac{q(x)}{p(x)}$ is large.

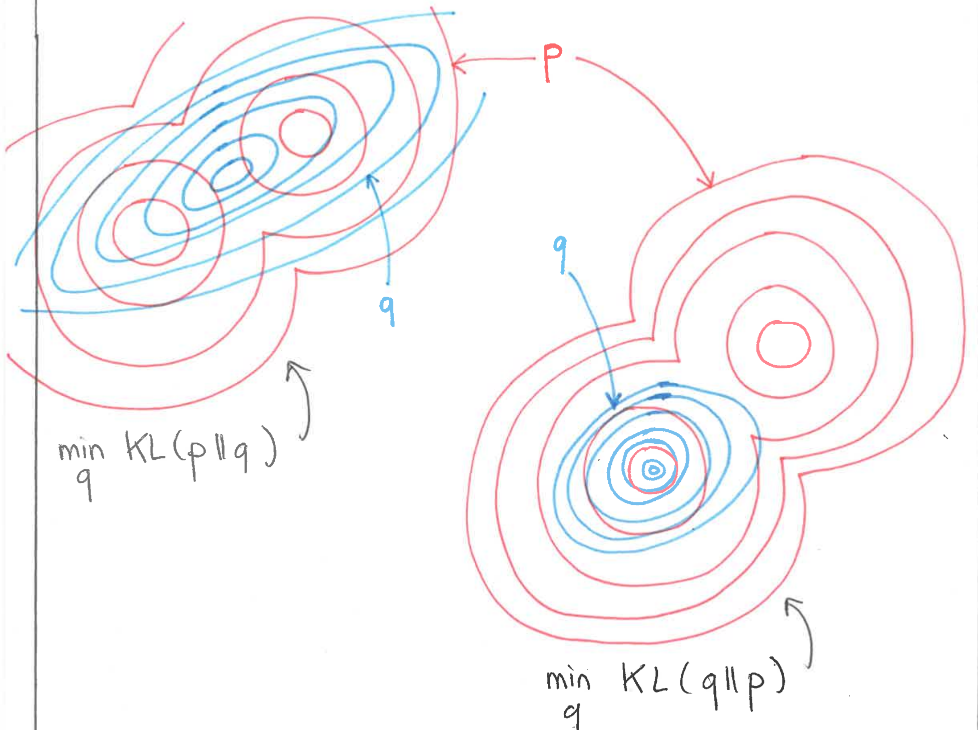
$\Rightarrow$ Tends to return solutions with a support that is too compact.
"local" approximator of p
Usually underestimate the variance.
(in the cases where p is multi-modal, the approximator typically picks one (or a subset of) mode only)
↪ see picture next page.

(ii) $\underset{q}{\min}$ $\underline{KL(p\|q)}$.

- p is the reference distribution, it is fixed. q must be non-negligible in regions where p is non-negligible, to keep the ratio $\log \frac{p(x)}{q(x)}$ close to 0.

$\Rightarrow$ global approximation: q tries to approximate p across the entire support.

Ex: Approximation of a bi-modal distribution with a unimodal one.


p
q
q
$\underset{q}{\min}$ $KL(p\|q)$
$\underset{q}{\min}$ $KL(q\|p)$

(may return a different local minimum centered on the second mode).

II.2. Variational Bayes & Expectation Propagation.

We turn our attention to the problem of evaluating posterior distributions of the form $p(z|x)$. This situation always arises in Bayesian inference (and also in VAEs). The posterior distribution is usually intractable, and approximations are required.

$\to$ numerical: MCMC

$\to$ analytical: Variational Bayes & Expectation Propagation.

- x = observed variable
  z = latent variable (= parameters in Bayesian stats)
- Given: $p(x,z)$.
  (and the marginals $p(x) = \int p(x,z)\,dz$
  the conditional distributions $p(z|x) = \dfrac{p(x,z)}{p(x)}$ ... / ...)
- Goal: Find a good approximation $q(z)$ of the posterior $p(z|x)$.
- Idea: $\forall$ distribution $q(z)$, holds

$$\log p(x) = \sum_z q(z) \log p(x)$$
$$= \sum_z q(z) \log\left(\frac{p(x,z)}{p(z|x)}\right)$$
$$= \sum_z q(z) \log\left(\frac{p(x,z)}{q(z)} \; \frac{q(z)}{p(z|x)}\right)$$
$$= \sum_z q(z) \log\left(\frac{p(x,z)}{q(z)}\right) + \sum_z q(z) \log\left(\frac{q(z)}{p(z|x)}\right)$$

Put $\mathcal{L}(q) := \sum_z q(z) \log\left(\frac{p(x,z)}{q(z)}\right)$

$$KL(q \,\|\, p(\cdot|x)) := \sum_z q(z) \log\left(\frac{q(z)}{p(z|x)}\right)$$

$$\boxed{\Rightarrow \; \log p(x) = \mathcal{L}(q) + \underbrace{KL(q \,\|\, p(\cdot|x))}_{\geq 0}}$$

$\uparrow$ fixed w.r.t. q $\quad\quad$ $\uparrow$ lower bound

$\Rightarrow \log p(x) \geq \mathcal{L}(q)$.

& maximization of the lower bound is equivalent to the minimization of $KL(q \,\|\, p(\cdot|x))$

$\uparrow$
$[\,= 0$ iff $q = p(\cdot|x)\,]$.

- Variational Bayes: approximate $p(z|x)$ using $q(z)$ where q is obtained by minimizing $KL(q \,\|\, p(\cdot|x))$ over a family of tractable distributions (such as the ones that factorize over each variable)

In contrast...

- Expectation propagation: approximate $p(z|x)$ using $q(z)$ where q is obtained by minimizing $KL(p(\cdot|x) \,\|\, q)$ over a family of tractable distributions.

$\uparrow$ Section II.1 gives us an idea of the differences one can expect when using $KL(q \,\|\, p(\cdot|x))$ or $KL(p(\cdot|x) \,\|\, q)$ as a criterion.

Ex: Variational Linear Regression.

- Data distribution: $p(x \mid \beta) = \prod_{i=1}^{n} \mathcal{N}(x_i \mid \beta^t x_i , \tau^{-1})$
- Distribution of weights: $p(\beta \mid \lambda) = \mathcal{N}(\beta \mid 0, \lambda^{-1} I)$
- Distribution of precision: $p(\lambda) = \Gamma(\lambda \mid \nu_0, b_0)$

  $(\propto \lambda^{\nu_0 - 1} e^{-b_0 \lambda})$

We are interested in the posterior distribution

$p(\lambda, \beta \mid x) \propto p(x \mid \beta)\, p(\beta \mid \lambda)\, p(\lambda)$.

Use variational Bayes to approximate the posterior distribution as $q(\lambda, \beta) = q_\lambda(\lambda)\, q_\beta(\beta)$.

[Ref] Section 10.3 in Bishop.

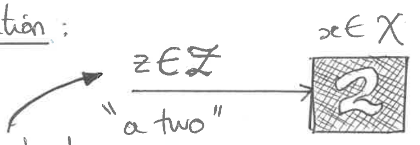
II.3. VAEs.

- VAEs use a latent variable formulation to produce new observations (images) from existing ones.

  x = image / vector of observation $\quad X \sim p(x)$. $\quad x \in \mathcal{X}$

  z = latent vector $\quad Z \sim p(z) \quad z \in \mathcal{Z}$

- Interpretation:



z is some abstract representation of the 'concept' number 2. One z is selected, a picture of the number '2' is generated from the transformation f.

Image of the number '2'.

$x = f(z\,;\, \theta)$

f is deterministic

parameter of f

⇒ Sample $z \sim p(z)$, and with high probability $f(z, \theta)$ will look like the picture of the number two.

To formalize the idea that some z results in samples that 'look like' x, we make the relationship between X and z probabilistic:

$$\boxed{p(x) = \int p(x \mid z\,;\theta)\, p(z)\, dz}$$

arbitrary. Usually $\mathcal{N}(0, I)$

LTP

— instead of $f(z, \theta)$ —

Typically, use

$p(x \mid z; \theta) \sim \mathcal{N}(x \mid f(z,\theta), \sigma^2 I)$

tuning param.

For a binary image, use

$p(x \mid z; \theta) \sim B(x \mid f(z, \theta))$

probability of 'success'.

↳ the class of functions considered to model $f(z, \theta)$ should be sufficiently rich to grasp the relationship between x and z. "powerful function approximator"

— Take $f(z, \theta)$ to be a multi-layer neural network —

- Note that not many z will result in producing the wanted image x. We need to evaluate which z are good latent representations of x. This can be done through the posterior distribution $p(z \mid x)$, which is fixed once the joint $p(x, z) = p(x \mid z)\, p(z)$ is known.

However, $p(z|x)$ is intractable (the mean of $p(x|z,\theta)$ is $f(z,\theta)$, an abitrarily complex function of z. How would you then compute $\int p(x|z,\theta)p(z)dz$

$\rightarrow$ Approximate $\underset{\substack{\uparrow \\ \text{our target}}}{p(z|x)}$ using $\underset{\substack{\uparrow \\ \text{our candidate}}}{q(z|x)}$ using variational inference

(*) $\log p(x) = \mathcal{L}(q(\cdot|x)) + KL(\underset{\substack{\uparrow \\ \text{makes sense to take a function } q \text{ that depends on } x.}}{q(\cdot|x)} \| p(\cdot|x))$

(see page 18 with $q \to q(\cdot|x)$)

where

↘ $\mathcal{L}(q(\cdot|x)) = E_q\left\{\log\left(\frac{p(x,z)}{q(z|x)}\right)\right\}$

↘ $KL(q(\cdot|x)\|p(\cdot|x)) = E_q\left\{\log\left(\frac{q(z|x)}{p(z|x)}\right)\right\} \geq 0$

Thus $\log p(x) \geq \underbrace{\mathcal{L}(q(\cdot|x))}$

make the lower bound as large as possible (indeed, we want to find the parameters of the model which make $p(x)$ the largest).

• Re-writing the lower bound:

$$\mathcal{L}(q(\cdot|x)) = E_{z\sim q}\{\log \underset{\substack{\uparrow \\ \mathcal{N}(x|f(z,\theta),\sigma^2)}}{p(x|z)}\} - KL(q(\cdot|x)\|\underset{\substack{\uparrow \\ \mathcal{N}(0,I)}}{p(\cdot)})$$

$\Rightarrow$ We want to maximize this lower bound over a rich enough class of candidates $q(z|x) \rightarrow$ take $\mathcal{N}(z|\mu(x),\Sigma(x))$.

With this choice of $q(z|x)$, the KL divergence between $q(z|x)$ and $p(z)=\mathcal{N}(z|0,I)$ can be calculated explicitely:

$$KL(q(\cdot|x)\|p(\cdot)) = \frac{1}{2}\left\{\mathrm{Tr}\,\Sigma(x) + \mu(x)^t\mu(x) - \underset{\substack{\uparrow \\ \text{dimension of } z\in\mathbb{R}^k}}{k} - \log\det\Sigma(x)\right\}$$

• To maximize $\mathcal{L}(q(\cdot|x))$, it remains to control the term $E_{z\sim q}\{\log p(x|z)\}$.

$\Rightarrow$ Use Stochastic Gradient Descent (SGD) over both the variables x and z:

full criterion to minimize:

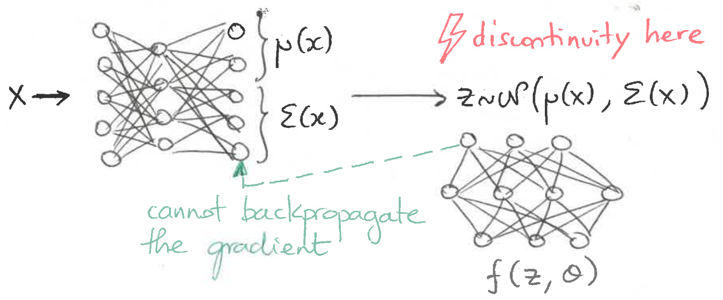
$$\underset{\substack{\uparrow \\ \text{Pick an } X \text{ randomly from the training sample (or a minibatch)}}}{E_{X\sim\mathcal{L}_n}}\left\{ \underset{\substack{\uparrow \\ \text{generate a } z \text{ randomly from the distribution } q(\cdot|x) \text{ once } x \text{ is generated}}}{E_{z\sim q}}\{\log p(X|z)\} - KL(q(\cdot|X)\|p(\ ))\right\}.$$

& compute the gradient of the term inside the expectation

$$\log p(x|z) - KL(q(z|x)\|p(z))$$

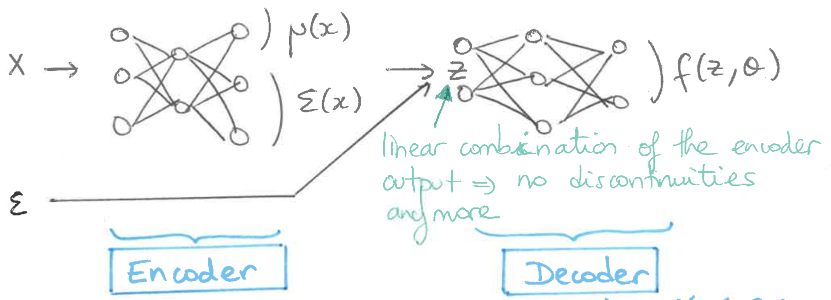
⚠ $\mu(x)$ & $\Sigma(x)$ are usually the output of a deep neural net.

$\Rightarrow$ the gradient of $\log p(x|z) - KL(q(\cdot|x)\|p(\cdot))$ needs to be back-propagated, which in the present form is not possible due to the presence of stochasticity inside the network: a sample z is generated from $\mathcal{N}(\mu(x), \Sigma(x))$ in the middle of the network:


$X \rightarrow$
$\mu(x)$
$\Sigma(x)$
discontinuity here
$z \sim \mathcal{N}(\mu(x), \Sigma(x))$
cannot backpropagate the gradient
$f(z, \theta)$

* ~~Trick~~ Trick: Shift the stochasticity inside the network, outside:

$z = \mu(x) + \Sigma^{1/2}(x)\, \varepsilon \quad ; \quad \varepsilon \sim \mathcal{N}(0, I)$.



$q(z \mid x) \sim \mathcal{N}(z \mid \mu(x), \Sigma(x))$ $\qquad$ $p(x \mid z) \sim \mathcal{N}(x \mid f(z, \theta), \sigma^2)$

Remarks. We have neglected the remaining term $KL(q(\cdot \mid x) \,\|\, p(\cdot \mid x))$, which measures how good our approximation of the posterior is. If q has "high capacity", we expect this term to be small.

. Once the model is trained, we can drop the encoder. Simply use the decoder to generate new samples.

. $\log p(x \mid z) = \|x - f(z, \theta)\|^2 + \text{constant}$.

III_ WASSERSTEIN GANs (WGANs).

We return to GANs, and discuss recent (late 2017) improvements based on a "new" method for evaluating the distance between $\mathbb{P}$ (the real data distribution) and $\mathbb{P}_g$ (model distribution of $x = G(z)$ induced by the mapping G & $\mathbb{P}_z$).

→ GANs ≡ minimize $JS(\mathbb{P}, \mathbb{P}_g)$

→ WGANs ≡ minimize the Wasserstein distance between $\mathbb{P}$ & $\mathbb{P}_g$.

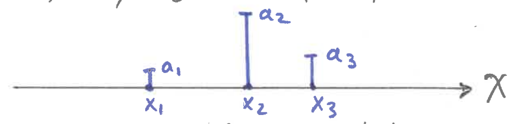
Not "new" & appeared for the first time in the work of Wasserstein in 1969.

III.1. Definition of Wasserstein distance.

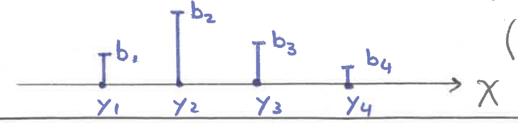
x Motivation: Optimal Transport Plan between two discrete probability distributions.

Let μ = discrete probability distribution placing mass a_1, a_2, a_3 on x_1, x_2, x_3 $\left(\sum_{i=1}^{3} a_i = 1 \quad x_i \in \mathcal{X}\right)$

metric space



ν = discrete probability distribution placing mass b_1, b_2, b_3, b_4 on $y_1, y_2, y_3, y_4 \in \mathcal{X}$ $\left(\sum_{i=1}^{4} b_i = 1\right)$



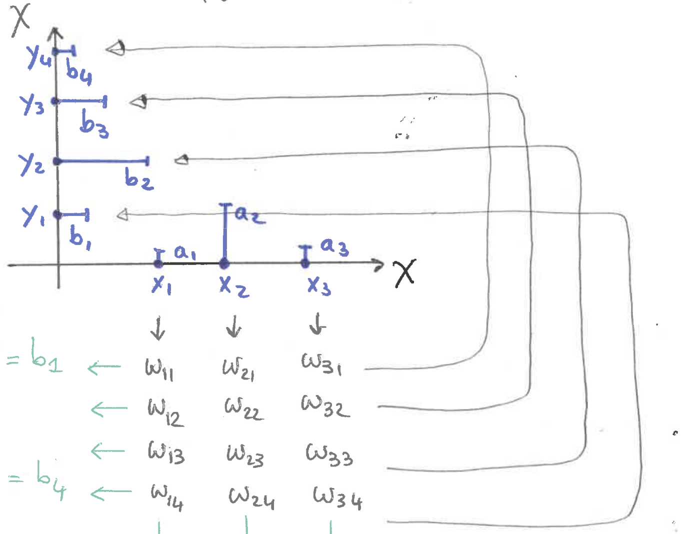
- The goal is to introduce a measure of dissimilarity between μ and ν, based on how much effort / energy is required to move particules x_i of mass a_i to particules y_j of mass b_j.

- Intuitively, the cost of moving a particule of mass w from x to y can be taken as $w\, d(x, y)$.

proportional to its weight

the further away you go, the more expensive

- With more than one particule, the idea is to split each particule x_i into subparticules, and to move them around into their new configuration $\{y_i\}$.



X
y_4
b_4
y_3
b_3
y_2
b_2
y_1
b_1
a_1
a_2
a_3
x_1
x_2
x_3
X
w_{11} w_{21} w_{31}
w_{12} w_{22} w_{32}
w_{13} w_{23} w_{33}
w_{14} w_{24} w_{34}

$\sum_{i=1}^{3} w_{i1} = b_1$

$\vdots$

$\sum_{i=1}^{3} w_{i4} = b_4$

$\sum_{j=1}^{4} w_{1j} = a_1$ ... $\sum_{j=1}^{4} w_{3j} = a_3$

Constraints on w_{ij}

The cost of moving a particule of mass w_{ij} from x_i to y_j is $w_{ij}\, d(x_i, y_j)$.

$\Rightarrow$ The total cost of the split-move-merge procedure is $\sum_{i,j} w_{ij}\, d(x_i, x_j)$.

But there are many ways to split the mass of particules x_i.

$\Rightarrow$ The cost of going from μ to ν is taken to be the minimum possible cost associated with the split-move-merge strategy.

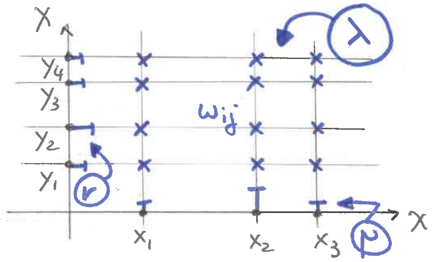
- The non-negative weights w_{ij} ($\sum_{i,j} w_{ij} = 1$) define a distribution on $\mathcal{X} \times \mathcal{X}$, whose marginals are precisely μ and ν.

Call this joint distribution λ.

$\downarrow$

The cost of the split-move-merge procedure coincides with the expected value of $d(X, Y)$ under λ

$= \mathbb{E}\{d(X,Y)\}$ $(X,Y) \sim \lambda$, $X \sim \mu$, $Y \sim \nu$.

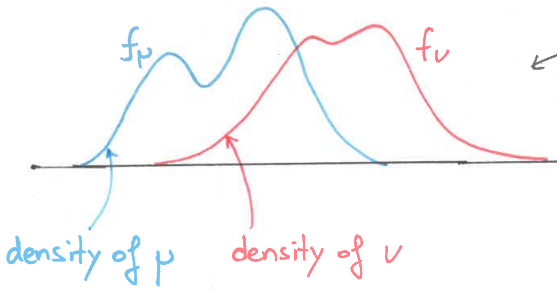


- The infimum of $\mathbb{E}\{d(X,Y)\}$ over all admissible joint distributions λ (given the marginals μ, ν) is precisely the definition of the Wasserstein distance between μ and ν:

$$\boxed{W(\mu,\nu) = \inf_{\substack{\lambda: \\ X\sim\mu \\ Y\sim\nu}} \mathbb{E}\,\|X - Y\|}$$

Take $d(X,Y) = \|X - Y\|$

Remark = The Wasserstein distance is also known as the Earth-Mover (EM) distance. $W(\mu,\nu)$ is the cost of the optimal transport plan. The approach is well suited for distributions that look-alike, but with non-intersecting supports.



← When translated, f_μ and f_ν look similar: it does not take much energy to transform the shape of f_μ to make it look like f_ν

↳ Wasserstein distance captures precisely this.

Differs from usual notions of distance between two functions, mostly comparing the values of $f_\mu(x)$ & $f_\nu(x)$ $\forall x \in X$.

→ $\max_{x\in X} |f_\mu(x) - f_\nu(x)|$ (or sup)

→ $\int_X |f_\mu(x) - f_\nu(x)|\, dx$

Theorem [KANTOROVICH – RUBINSTEIN DUALITY]

$$W(\mu,\nu) = \sup_{\|f\|_L \le 1} \left\{ \mathbb{E}_{X\sim\mu}[f(X)] - \mathbb{E}_{X\sim\nu}[f(X)] \right\},$$

where the supremum is taken over all 1-Lipschitz functions $f : X \to \mathbb{R}$

III.2. Properties of the Wasserstein distance.

There are various ways to measure how close two distributions are. Common definitions include:

(i) Total Variation (TV) $\mathbb{P}, \mathbb{Q}$ = proba measures defined on X ← compact, metric

$$\delta(\mathbb{P},\mathbb{Q}) = \sup_{B\in\mathcal{B}(X)} |\mathbb{P}(B) - \mathbb{Q}(B)|$$

(ii) Kullback-Leibler (KL) divergence (not a distance)

$$KL(\mathbb{P}\|\mathbb{Q}) = \int p(x) \log\left(\frac{p(x)}{q(x)}\right) d\mu(x),$$

where $\mathbb{P}, \mathbb{Q}$ are AC w.r.t some common measure μ defined on X, with densities p, q.

(iii) Jensen-Shannon (JS) divergence (symmetrical)

$$JS(\mathbb{P},\mathbb{Q}) = KL(\mathbb{P}\|\mathbb{M}) + KL(\mathbb{Q}\|\mathbb{M}),$$

where $\mathbb{M} = \frac{1}{2}(\mathbb{P}+\mathbb{Q})$

(iv) Wasserstein distance

$$W(\mathbb{P},\mathbb{Q}) = \inf_{\substack{\lambda\\ X\sim\mathbb{P}\\ Y\sim\mathbb{Q}}} \mathbb{E}_{(X,Y)\sim\lambda} \|X - Y\|.$$

↑ These expressions can be used to quantify the notion of convergence between probability distributions. Depending on the definition used, a sequence of proba distribution may converge under one metric, but not under another one: there are weaker notions of convergence. It turns out that Wasserstein distance defines a weak notion of convergence, as the next theorem states.

Theorem (thm 2 in Arjovsky, Chintala & Bottou (2017))

Let $\mathcal{X}$ = compact metric space
$\mathbb{P}$ = distribution on $\mathcal{X}$
$(\mathbb{P}_n)$ = sequence of distributions on $\mathcal{X}$.

(1) $\delta(\mathbb{P}_n, \mathbb{P}) \to 0 \iff JS(\mathbb{P}_n, \mathbb{P}) \to 0$

(2) $W(\mathbb{P}_n, \mathbb{P}) \to 0 \iff \mathbb{P}_n \xrightarrow{d} \mathbb{P}$ (weak cv.)

(3) $KL(\mathbb{P}_n \| \mathbb{P}) \to 0$ or $KL(\mathbb{P} \| \mathbb{P}_n) \to 0 \Rightarrow (1)$

(4) Statements (1) $\Rightarrow$ Statements (2)

It is easier to converge under the Wasserstein metric, than it is under the JS metric.

III. 3. WGANs

→ GANs The objective is the optimization problem:

$$\min_{\theta_g} \max_{\theta_d} \left\{ \mathbb{E}_X \log D(X) + \mathbb{E}_Z \log(1 - D(G(Z))) \right\}$$

(see top of page 6)

→ WGANs replace this objective with:

$$\min_{\theta_g} \max_{f \in \mathcal{F}} \left\{ \mathbb{E}_X f(X) - \mathbb{E}_Z f(G(Z)) \right\},$$

(from KR duality page 27)

where $\mathcal{F}$:= set of 1-Lipschitz functions.

↳ The optimization problem requires finding the 1-Lipschitz function f that maximizes the inner term. This is a rather tedious problem. We approximate f by training a neural network, resulting in a function f_{θ_d} parametrized by the weights of the network θ_d.

↳ Enforcing Lipschitz continuity of the network output is the main challenge here.
Original solution proposed: clamp the weights to a fix box.

↳ Under the assumption that G and f are any feed-forward neural network, $W(\mathbb{P}, \mathbb{P}_g)$ is continuous everywhere, and differentiable almost everywhere (Corollary 1 in ACB (2017)).
$\Rightarrow$ We are allowed to take gradients of the objective, and to backpropagate the errors through the networks.

[Ref] M. Arjovsky, S. Chintala & L. Bottou
Wasserstein GAN.