GANs: Wasserstein

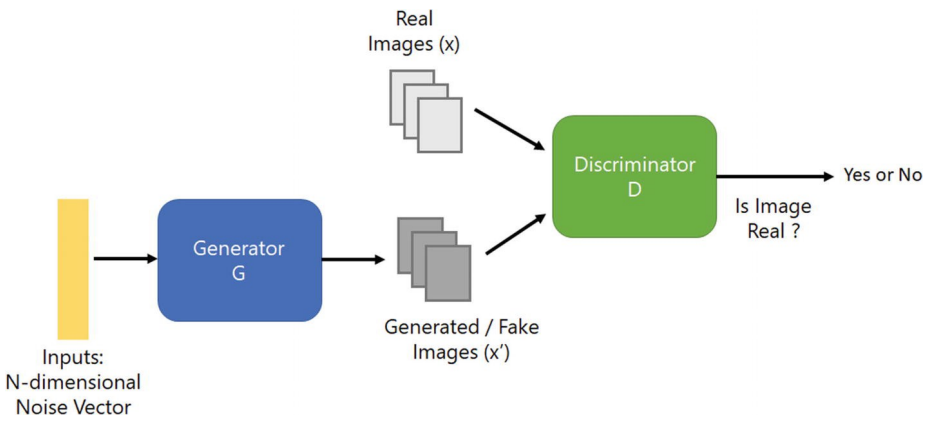
27 февраля 2020 г.

Denis Derkach, Artem Ryzhikov, Maksim Artemev

Contents

JSGAN

Reminder: vanilla GAN



- game between Generator and Discriminator;
- uses backprop for turn-by-turn optimisation;
- optimises Jensen-Shannon-like divergence.

vanilla GAN: main problems

- Stability of training:
  - each step needs optimal discriminator;
  - imbalances in generator / discriminator (in architecture and instance performance).
- Mode collapse:
  - connected to optimisation (JS and others);
  - connected to manifold character.
- Vanishing gradients:
  - JS divergence specific problems.

f-GANs

Constructing the divergence

- Can we change the divergence we use?
- We use f-divergence, for convex function $f(x) : \mathbb{R}^+ \mapsto \mathbb{R}$ and $f(1) = 0$:

$$D_f(P||Q) = \int_{\mathcal{X}} f(\frac{p(x)}{q(x)})q(x)dx.$$

- The inequality holds:

$$\int_{\mathcal{X}} f(\frac{p(x)}{q(x)})q(x)dx \geq f(\int_{\mathcal{X}} \frac{p(x)}{q(x)}q(x)dx) = 0.$$

Name	$D_f(P\|Q)$	Generator $f(u)$
Kullback-Leibler	$\int p(x) \log \frac{p(x)}{q(x)} \, \mathrm{d}x$	$u \log u$
Reverse KL	$\int q(x) \log \frac{q(x)}{p(x)} \, \mathrm{d}x$	$-\log u$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)} \, \mathrm{d}x$	$(u-1)^2$
Squared Hellinger	$\int \left(\sqrt{p(x)} - \sqrt{q(x)}\right)^2 \, \mathrm{d}x$	$\left(\sqrt{u}-1\right)^2$
Jensen-Shannon	$\frac{1}{2} \int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \, \mathrm{d}x$	$-(u+1) \log \frac{1+u}{2} + u \log u$
GAN	$\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \, \mathrm{d}x - \log(4)$	$u \log u - (u+1) \log(u+1)$

Fenchel conjugates

› We know the Fenchel conjugate (convex in t):

$$f^*(t) = \sup_u |ut - f(u)|$$

moreover, if f is convex, than $(f^*)^* = f$.

› Thus, if we know the lower bound (variational estimation):

$$D(P||Q) \geq \sup_{T \in \mathcal{T}} (\mathbb{E}_{x \sim P}[T(x)] - \mathbb{E}_{x \sim Q}[f^*(T(x))])$$

› We can find the the equality holds if:

$$T^*(x) = f' \left(\frac{p(x)}{q(x)} \right).$$

› Here T is a test function/critic/discriminator.

From :Nowozin et al. f-GAN: Training Generative Neural Samples using Variational Divergence Minimization, NIPS16

Variational Divergence Minimization

- Parameterize the generator as Q_θ such that e.g. $x = Q_\theta(z)$, $z \sim \mathcal{N}(0, 1)$. Parameterize the test function T as T_ω. In this case, we have:

$$\min_\theta \max_\omega F(\theta, \omega) = \mathbb{E}_{x \sim P}[T_\omega(x)] - \mathbb{E}_{x \sim Q_\theta}[f^*(T_\omega(x))]$$

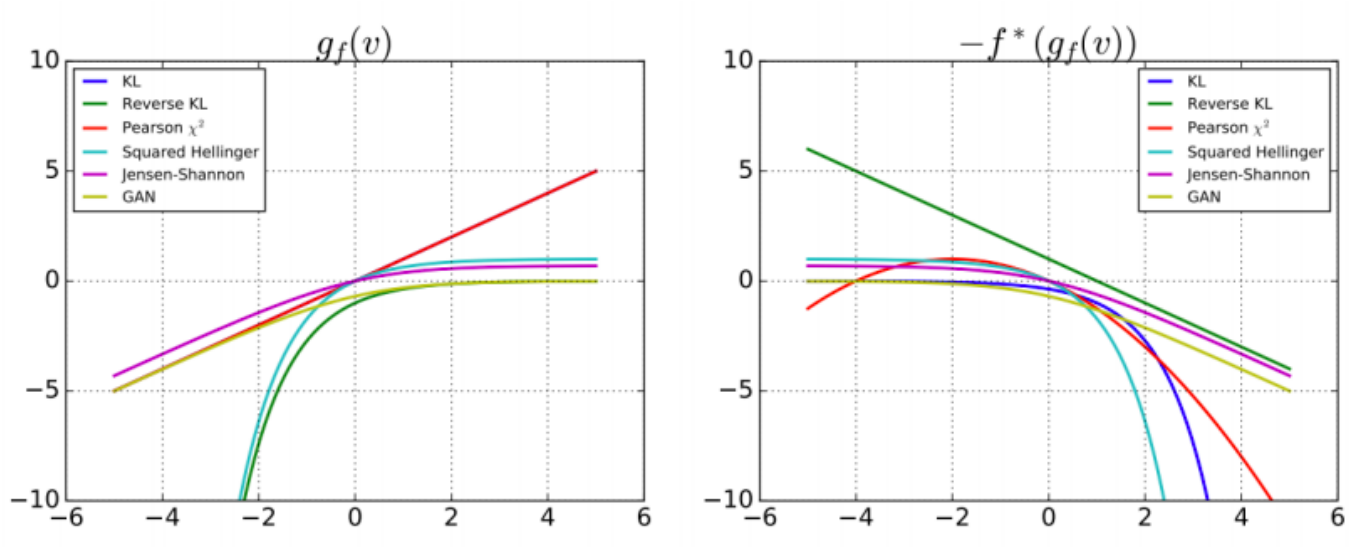
- To simplify the function choice we can use $T_\omega \to g_f(V_\omega)$, where g is nonlinearity chosen by f.
- Convention: choose $g_f(x)$ to be increasing so that a large $V_\omega(x)$ corresponds to samples from P.
- We have again a game with two networks.

Derivation from J. Cunningham course

Game objective

$$\min_{\theta} \max_{\omega} F(\theta, \omega) = \mathbb{E}_{x \sim P}[g_f(V_\omega(x))] - \mathbb{E}_{x \sim Q_\theta}[f^*(g_f(V_\omega(x)))].$$

Given a sample $x \sim P$, expect large $T(x)$ yet small $-f^*(g_f(V_\omega(x)))$.



We classify based on $p(x)/q(x)$ (since $T^* = f'(p(x)/g(x))$, using $f'(1)$ as a proxy threshold for classification. In particular, we classify $x \sim P$ when $T(x) > f'(1)$.

f-GAN results

The known divergences can be summarised in table

Name	$D_f(P\|Q)$	Generator $f(u)$	$T^*(x)$
Kullback-Leibler	$\int p(x) \log \frac{p(x)}{q(x)} \, \mathrm{d}x$	$u \log u$	$1 + \log \frac{p(x)}{q(x)}$
Reverse KL	$\int q(x) \log \frac{q(x)}{p(x)} \, \mathrm{d}x$	$-\log u$	$-\frac{q(x)}{p(x)}$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)} \, \mathrm{d}x$	$(u-1)^2$	$2(\frac{p(x)}{q(x)} - 1)$
Squared Hellinger	$\int \left(\sqrt{p(x)} - \sqrt{q(x)}\right)^2 \, \mathrm{d}x$	$(\sqrt{u}-1)^2$	$(\sqrt{\frac{p(x)}{q(x)}} - 1) \cdot \sqrt{\frac{q(x)}{p(x)}}$
Jensen-Shannon	$\frac{1}{2} \int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \, \mathrm{d}x$	$-(u+1) \log \frac{1+u}{2} + u \log u$	$\log \frac{2p(x)}{p(x)+q(x)}$
GAN	$\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \, \mathrm{d}x - \log(4)$	$u \log u - (u+1) \log(u+1)$	$\log \frac{p(x)}{p(x)+q(x)}$

We can than use to construct any GAN-game out of them.

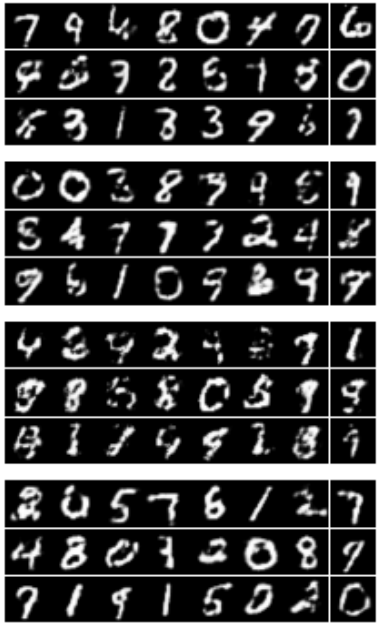


Figure 2: MNIST model samples trained using KL, reverse KL, Hellinger, Jensen from top to bottom.

f-GAN: wrap up

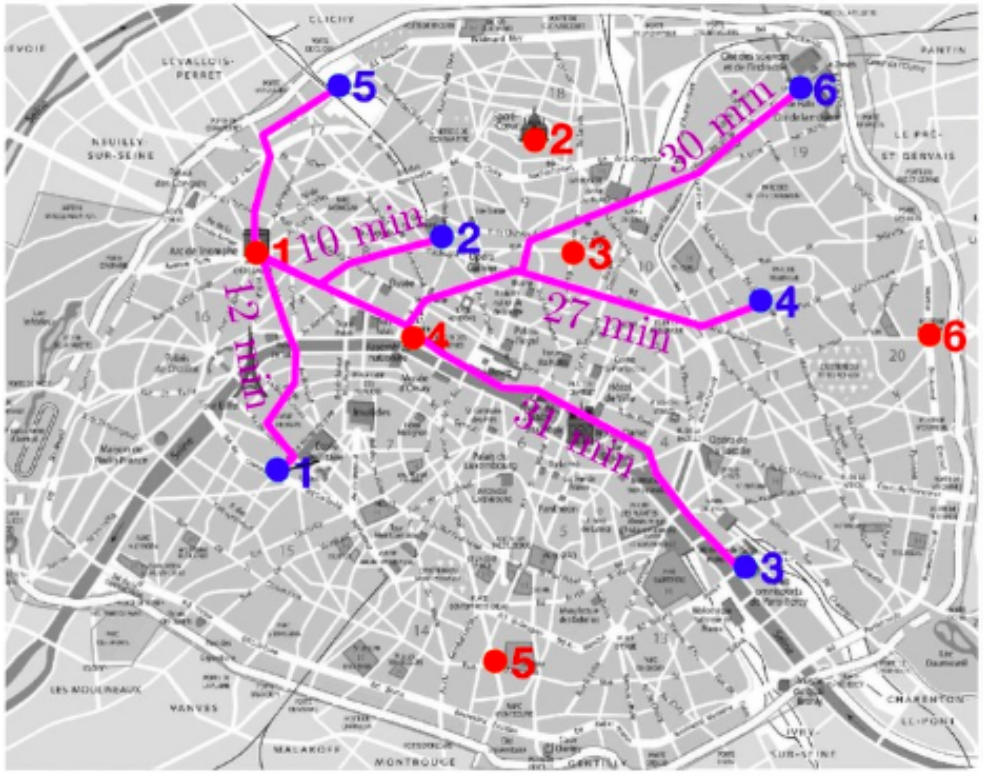
- The framework of distribution learning $\min_Q D(Q, P)$ includes many examples. In particular the specific form:

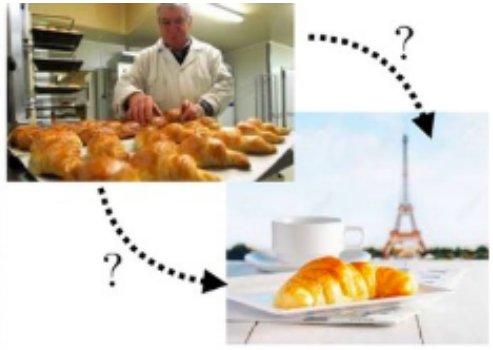
$$\min_\theta \max_T \mathbb{E}_{x\sim P}[T(x)] - \mathbb{E}_{x\sim Q_\theta}[f(T(x))]$$

  includes examples such as the original GAN, f-divergence GAN, MMD-GAN. Each model has its tradeoff.
- Using different f-divergence leads to very different learning dynamics.
- f-divergence generalizes learning using information theoretic divergence.
- Yet, we need a better way to train GANs.

Wasserstein Distance and GAN

French Bakeries



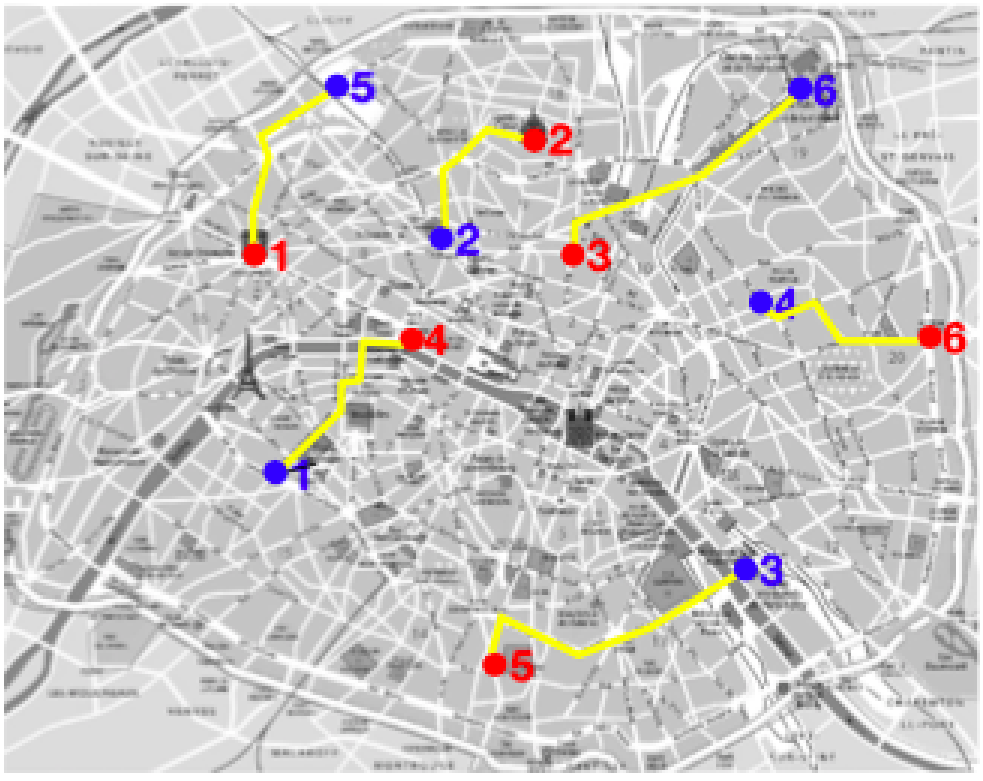


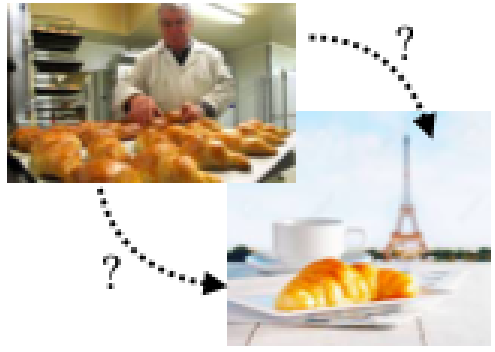
c_{ij}	y_1	y_2	y_3	y_4	y_5	y_6
x_1	12	10	31	27	10	30
x_2	22	7	25	15	11	14
x_3	19	7	19	10	15	15
x_4	10	6	21	19	14	24
x_5	15	23	14	24	31	34
x_6	35	26	16	9	34	15

given a set of N bakeries and M cafes, what is the optimal way to transport loaves of bread between them?

G. Peyre web-site

French Bakeries



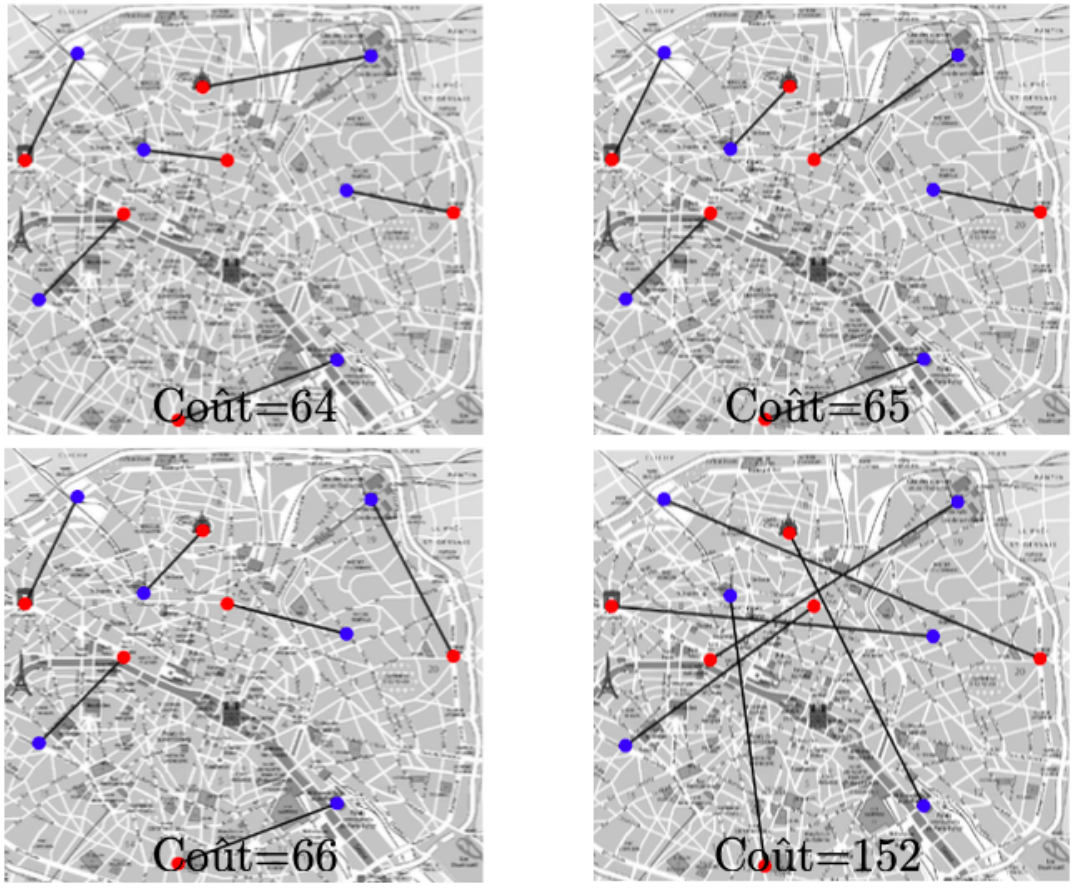


c_{ij}	y_1	y_2	y_3	y_4	y_5	y_6
x_1	12	10	31	27	10	30
x_2	22	7	25	15	11	14
x_3	19	7	19	10	15	15
x_4	10	6	21	19	14	24
x_5	15	23	14	24	31	34
x_6	35	26	16	9	34	15

Price = 10+7+15+10+14+9 = 65 min

G. Peyre web-site

More Bakeries

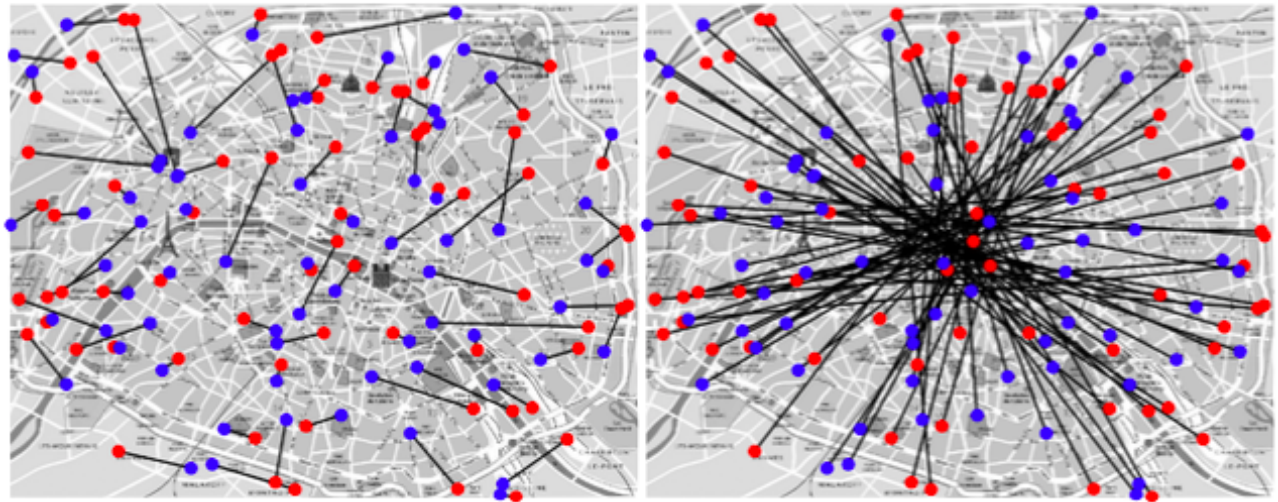


We can estimate different possibilities, using the same matrix of costs.

G. Peyre web-site

Optimal Transport: Monge

The number of calculations rises as factorial.



We thus need to solve a problem:

$$\min_{\sigma \in \mathrm{P}erm_n} \sum_{i=1}^{n} C_{i,\sigma(i)}$$

Formulate the Bakery problem: Kantorovich

What if bakeries can produce different mass of breads?

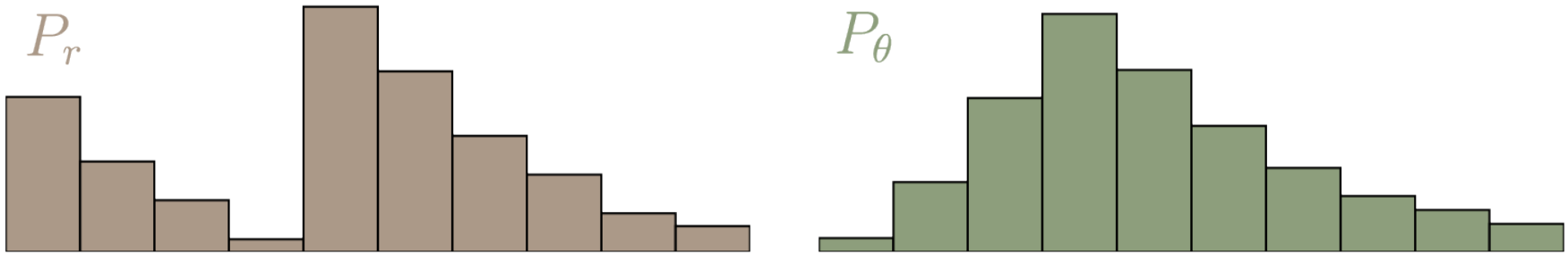
- Let:
  - $p_i \in 1...N$ the mass of bread held by each bakery;
  - $q_j \in 1...M$ the mass of bread desired by each cafe;
  - x_i, y_j the positions of bakeries and cafes
  - $\sum_i pi = \sum_j q_j = 1$, and cost is proportional to work (mass$\times$distance)

Find an optimal coupling $\gamma_{i,j}$ for the mass of bread moved from p_i to q_j. This defines the Earth Mover's (EM) distance:

$$EMD = \inf_{\gamma \in \Pi} \sum_{x,y} ||x - y|| \gamma(x, y) = \inf_{\gamma \in \Pi} \mathbb{E}_{(x,y)\sim\gamma} ||x - y||.$$

Why EMD?

Imagine that we want to move the events from P_r to P_θ. We also want to save effort, that is, not to move large pieces over long distances.

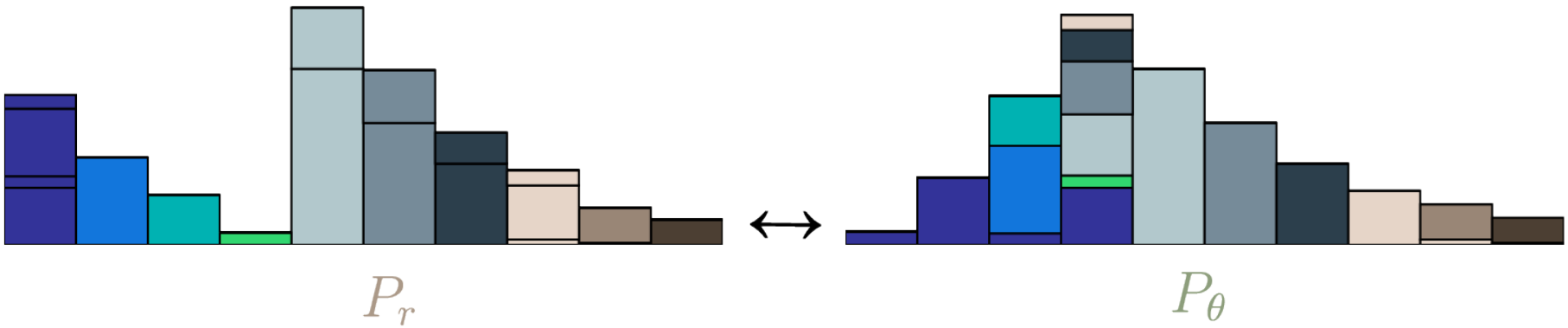


This is related to a problem that is solved by many construction workers every day. In fact, this is the optimal transport problem from P_r to P_θ.

Figure: Vincent Herrmann

Why EMD?

Imagine that we want to move the events from P_r to P_θ. We also want to save effort, that is, not to move large pieces over long distances.



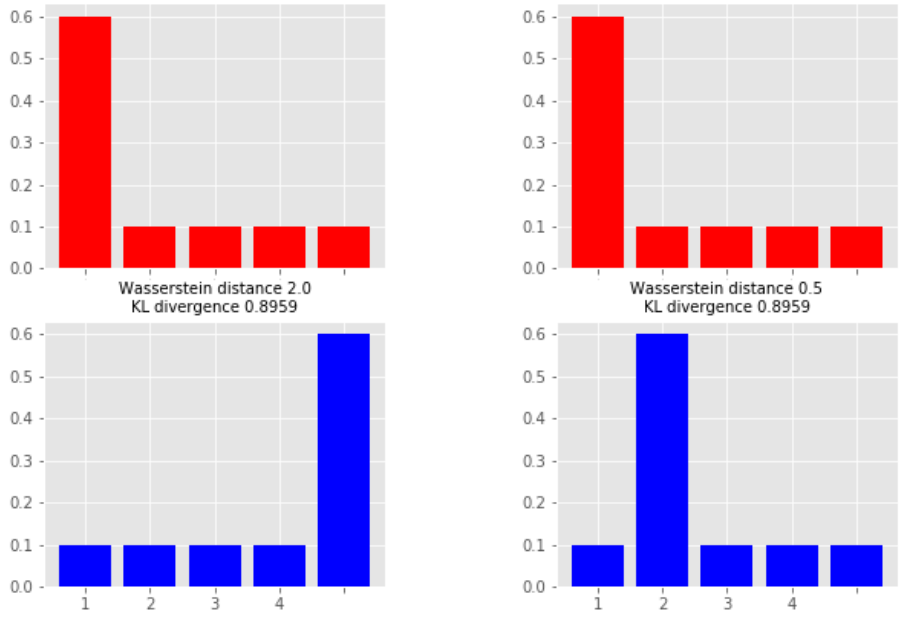
This is related to a problem that is solved by many construction workers every day. In fact, this is the optimal transport problem from P_r to P_θ.

$$EMD(P_r, P_\theta) = \inf_{\gamma \in \Pi} \sum_{x,y} ||x - y|| \gamma(x, y) = \inf_{\gamma \in \Pi} \mathbb{E}_{(x,y)\sim\gamma} ||x - y||,$$

Figure: Vincent Herrmann

W vs KL

EMD also takes into account the distance at which the differences in the distributions are located. This is exactly what we need to take into account multiple solutions!



StackExchange

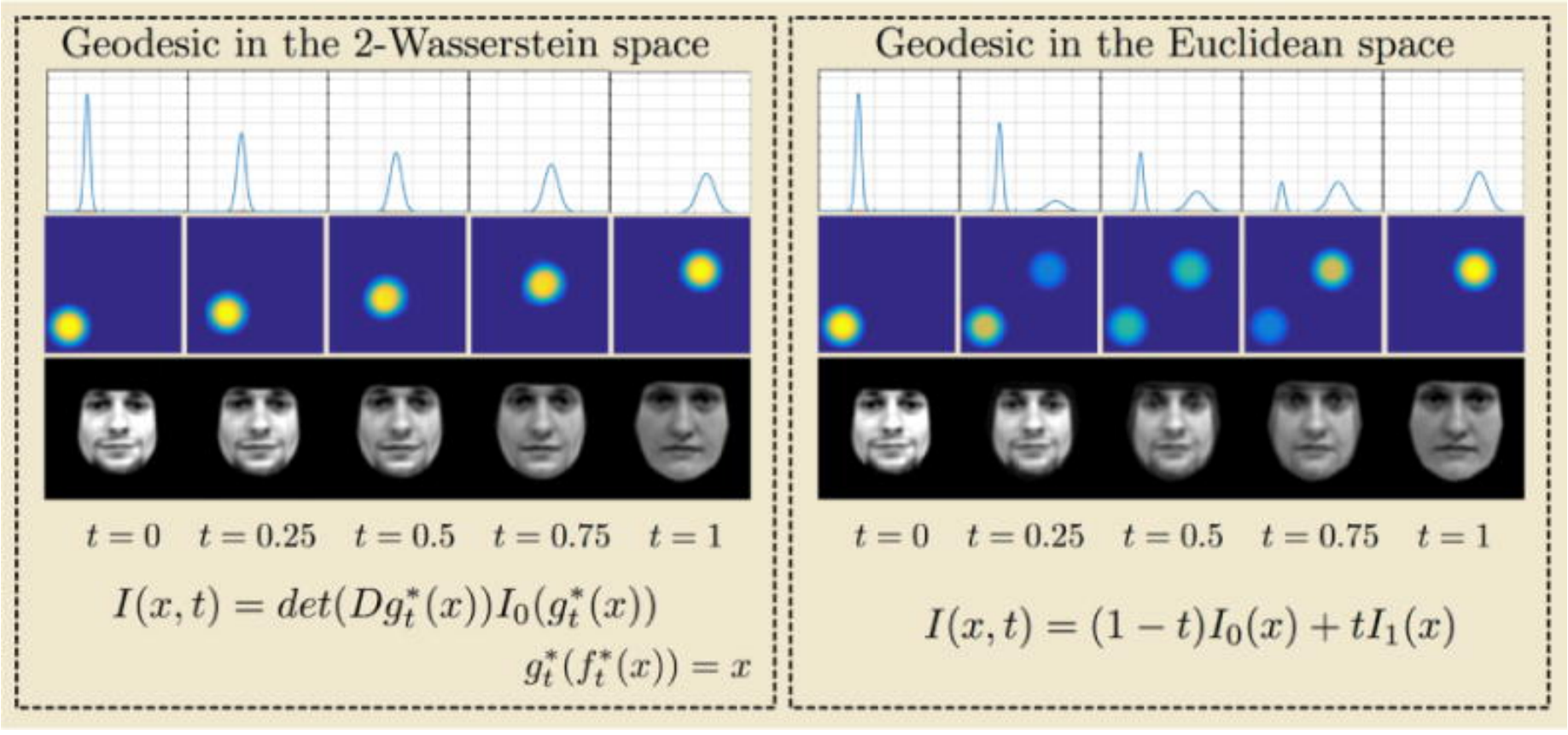
Wasserstein Distance

For continuous case, there are a set of p-Wasserstein distances, with $W_p(p_x, q_y)$ defined with $x \in M$, $y \in M$ and a distance D on x, y:

$$W_p(p_x, q_y) = \inf_{\gamma \in \Pi(x,y)} \int_{M \times M} D(x, y)^p d\gamma(x, y),$$

where $\Pi(x, y)$ is a set of all joint distributions having p_x, q_y as their marginals.

W_2 for interpolation



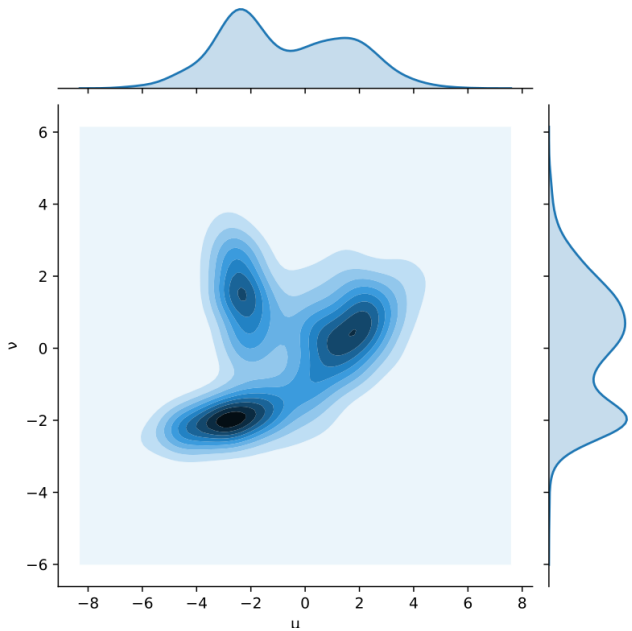
S. Kolouri et al. Optimal Mass Transport: Signal processing and machine-learning applications, IEEE Signal Process Mag. (2017)

W_1 Distance

In particular, W_1 distance with Euclidean norm is:

$$W(p_x, q_y) = \inf_{\gamma \in \Pi(x,y)} \int_{M \times M} D(x,y) d\gamma(x,y) = \inf_{\gamma \in \Pi(x,y)} \mathbb{E}(||x - y||)$$

Which brings an evident connection to EMD.

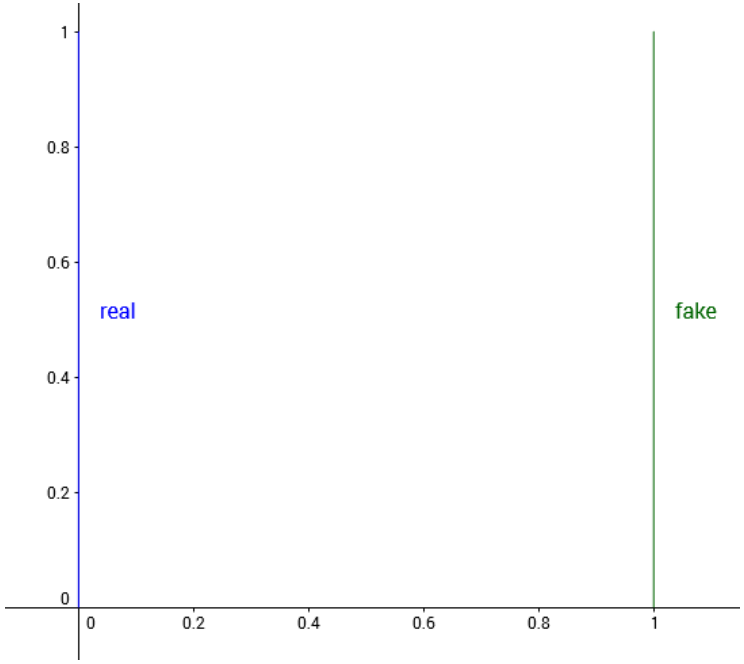


Two dimensional representation of the transport plan between horizontal (μ) and vertical ν pdfs. Note, that this is not unique plan. The inf must be taken over all possible plans.

Picture source:Wikipedia

An Illustrative Example

Let $y \sim U[0, 1]$ be the uniform distribution on the unit interval. Let P_0 be the distribution of $(0, y) \in \mathbb{R}^2$. uniform on a straight line centered at the origin. Let P_θ be the distribution of (θ, y) on R^2.



Let's check the known distances.

Figure from :Alex Irpan blog

An Illustrative Example

- Total variation:

$$TV(P_0; P_\theta) = \begin{cases} 0 & \text{if } \theta \neq 0 \\ 1 & \text{if } \theta = 0 \end{cases}$$

- Kulback-Leibler:

$$KL(P_0; P_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$$

- Jenson-Shannon:

$$JS(P_0; P_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$$

- Wasserstein:

$$W(P_0; P_\theta) = |\theta|.$$

W Properties (not strict)

From the illustrative example we can guess the properties for P being true PDF:

- For a sequence of distributions P_n: $KL(P_n||P) \to 0 \Rightarrow JS(P_n||P) \to 0 \Rightarrow W(P_n||P) \to 0$, with the latter equivalent to $P_n \overset{\text{distr}}{\to} P$.
- For $P_\theta = g_\theta(Z)$ (with Z - rv and g - deterministic: if g_θ is continuous, $W(P_\theta, P)$ is also continuous, with some more restrictions $W(P_\theta, P)$ is differential almost anywhere.

These nice properties (even nicer than KL) seem to be good enough to try W as a GAN divergence: but one problem - calculation takes time.

From: Arjovsky et al. Wasserstein GAN

Reminder: More intuition

- Since we might have disjoint support, let's introduce random noise $\varepsilon \sim \mathcal{N}(0; \sigma^2 I)$:

$$\mathbb{P}_{x+\varepsilon}(x) = \mathbb{E}_{y \sim \mathbb{P}(x)} \mathbb{P}_{\varepsilon}(x - y).$$

- This will make noisy supports, that makes it difficult for discriminator to overlap.
- This type of noise gives interesting result:

$$W(P_r||Q_g) \leq 2V^{1/2} + \sqrt{JS(P_{r+\varepsilon}||Q_{g+\varepsilon})},$$

where V is the variance of noise.

From :Arjovsky et al. Towards principled methods for training generative adversarial networks, ICLR17

Kantorovich-Rubinstein Duality

The Wasserstein distance above is rather difficult to calculate in practice, luckily we have the duality of this metrics:

$$W(p_r, p_\theta) = \sup_{f \in \mathrm{Lip}_1(\mathrm{X})} \left(\mathbb{E}_{x \sim p_r}[f(x)] - \mathbb{E}_{x \sim p_\theta}[f(x)] \right),$$

where $\mathrm{Lip}_1(\mathrm{X})$: $|f(x) - f(y)| \leq d(x, y)$.

The form reminds of f-GAN!

We can try to use a neural network as f.

Back to Bakeries

In fact, the duality works also for EMD, we can say:

$$EMD = \sup_{||f||_L \leq 1} \sum_i f_i q_i - \sum_j f_j p_j$$

For example of bakeries, f can be interpreted as a price of buying or selling at points x_i and y_j.

From :Villani, Optimal Transport

WGAN: Approximations

- **Restriction to a parametric family of functions.** We will optimize over a family that are all K-Lipschitz for some K:

$$\min_{\theta} W(p_r, p_\theta) = \min_{\theta} \max_{w} \left(\mathbb{E}_{x \sim p_r}[f_w(x)] - \mathbb{E}_{x \sim p_\theta}[f_w(g_\theta(z))]\right),$$

  - Evaluation gives $W(p_r, p_\theta)$ up to a multiplicative constant.
  - Differentiation w.r.t θ gives $\frac{\partial W(p_r, p_\theta)}{\partial \theta}$ up to a multiplicative constant.
- **Implementation of W via clipping.** If the weights of the network are in a compact space, the network will be K-Lipschitz for some K.
  - Clip the weights of the network to a fixed box after each gradient update
  - Not the same as updating within the constraints.

WGAN Algorithm

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

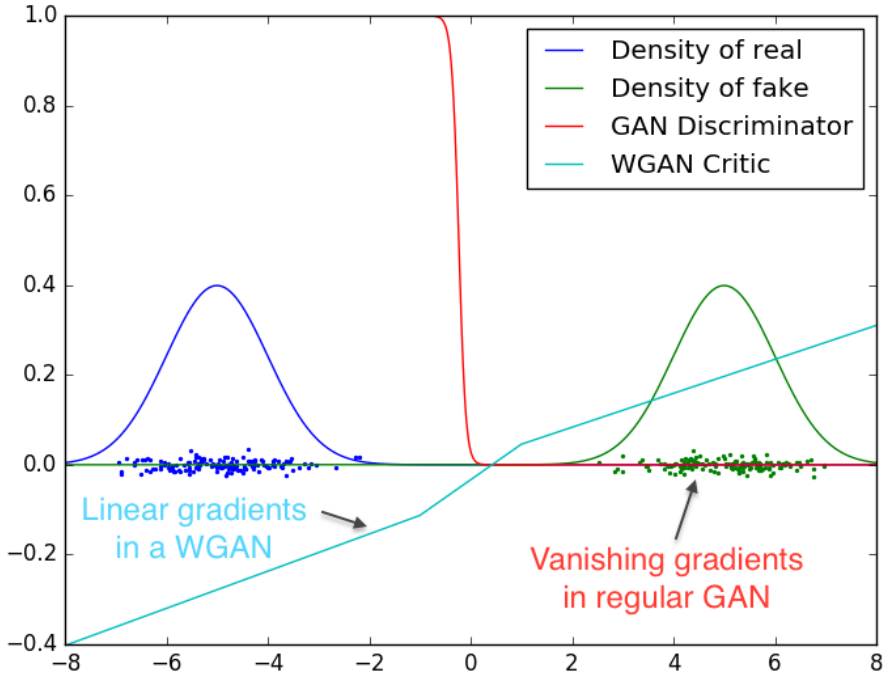
WGAN vs JSGAN

	Discriminator/Critic	Generator
GAN	$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$	$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$
WGAN	$\nabla_w \frac{1}{m} \sum_{i=1}^{m} \left[f\left(x^{(i)}\right) - f\left(G\left(z^{(i)}\right)\right) \right]$	$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f\left(G\left(z^{(i)}\right)\right)$

- Unified loss.
- We can now train the critic to optimality, no concerns about saturation and loss of the gradient if the critic becomes too good.
- W correlates better well with objective.
- WGAN simplifies architectures.

WGAN: problems solved

- mode collapse problem is addressed;
- the vanishing gradient problem is solved;
- from authors: Weight clipping is a clearly terrible way to enforce a Lipschitz constraint.



From: Arjovsky et al. Wasserstein GAN

WGAN: results

Figure 5: Algorithms trained with a DCGAN generator. Left: WGAN algorithm. Right: standard GAN formulation. Both algorithms produce high quality samples.

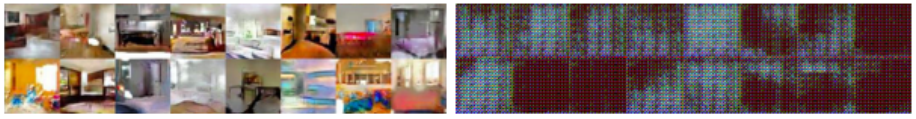


Figure 6: Algorithms trained with a generator without batch normalization and constant number of filters at every layer (as opposed to duplicating them every time as in [18]). Aside from taking out batch normalization, the number of parameters is therefore reduced by a bit more than an order of magnitude. Left: WGAN algorithm. Right: standard GAN formulation. As we can see the standard GAN failed to learn while the WGAN still was able to produce samples.

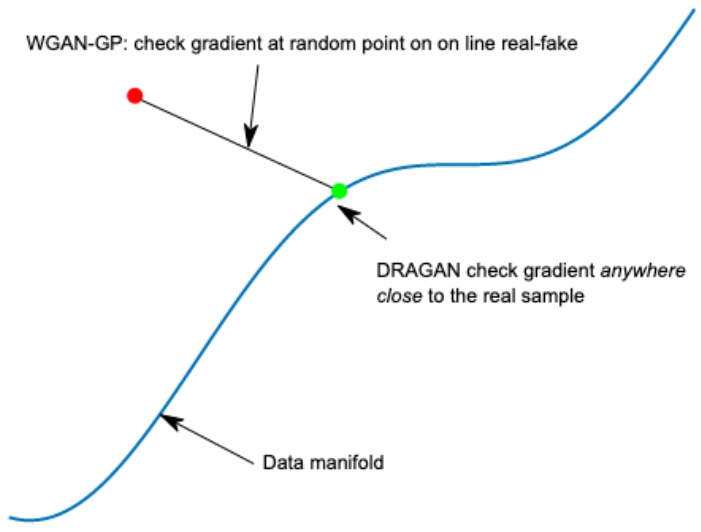
Figure 7: Algorithms trained with an MLP generator with 4 layers and 512 units with ReLU nonlinearities. The number of parameters is similar to that of a DCGAN, but it lacks a strong inductive bias for image generation. Left: WGAN algorithm. Right: standard GAN formulation. The WGAN method still was able to produce samples, lower quality than the DCGAN, and of higher quality than the MLP of the standard GAN. Note the significant degree of mode collapse in the GAN MLP.

Gradient penalty

- a better solution instead of clipping weights is to introduce penalty to gradients in the loss (WGAN-GP):

$$GP = \lambda \mathbb{E}\left[(||\nabla f|| - 1)^2\right]$$

- the gradient is taken at a random point between real and fake example (given the transportation plan).



From: Gulrajani1 et al. Improved Training of Wasserstein GAN Figure: Lernapparat blog

WGAN-GP: results

Figure 2: Different GAN architectures trained with different methods. We only succeeded in training every architecture with a shared set of hyperparameters using WGAN-GP.

Spectral Normalisation

› Spectral normalisation proposes to use normalised weights:

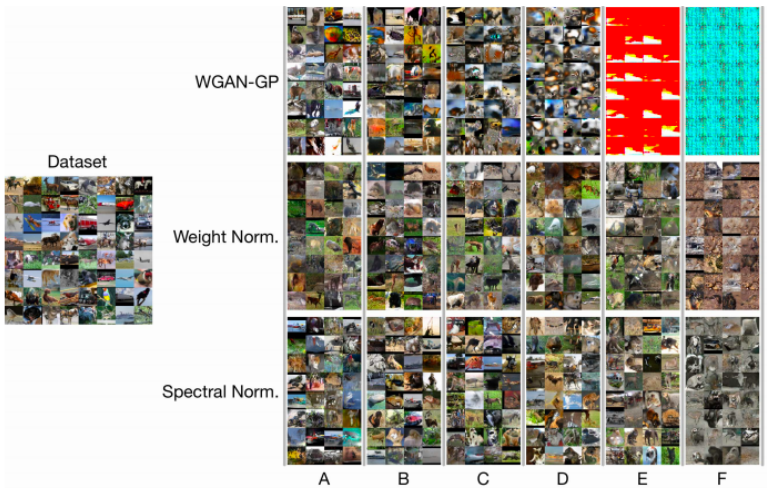
$$W_{SN} = \frac{W}{\sigma(W)}$$

where:

$$\sigma(W) = \max_{h:h\neq 0} \frac{||Wh||_2}{||h||_2}$$

› this gives constraints on gradient:

$$||f||_{Lip} \leq \prod_{i=1}^{l} \sigma(W_l).$$



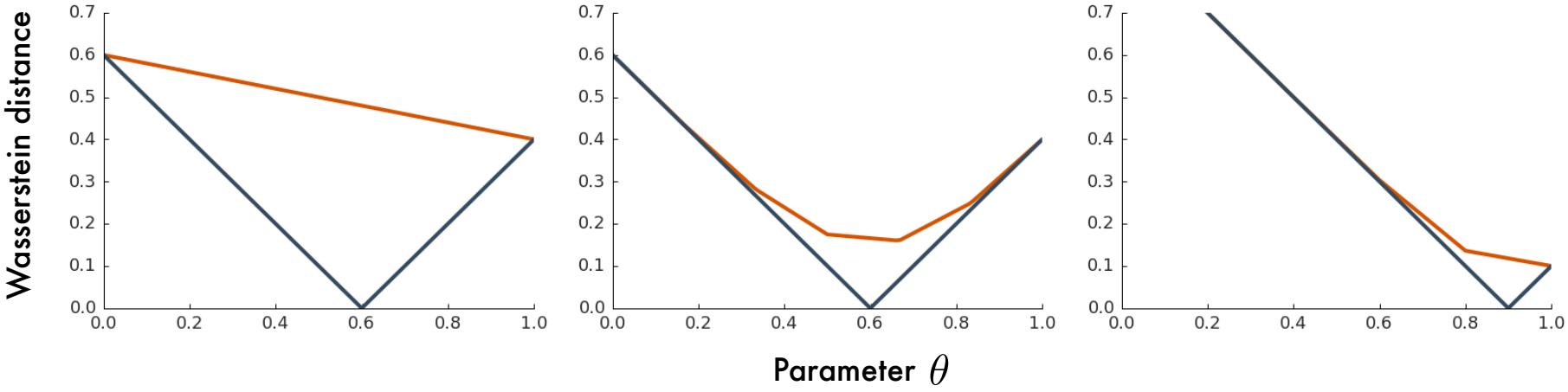
(b) STL-10

Figure 6: Generated images on different methods: WGAN-GP, weight normalization, and spectral normalization on CIFAR-10 and STL-10.

From: Miyato et al. Spectral Normalization for Generative Adversarial Networks, ICLR 2018 Figure: Lernapparat blog

WGAN: more problems

- The expected EMD gradients can differ from the true gradients.
- This leads to problems even for Bernoulli distribution.



Red for sample gradient expectation, blue is for real gradients solution.
Left to right $\theta^* = 0.6; 0.6; 0.9$.

From: M. Bellemare, arXiv:1705.10743

Cramer Distance and GAN

Energy and Cramer Distance

To fight biases, one can introduce Cramer distance:

$$\int_{-\infty}^{\infty} (F(x) - G(x))^2 \, dx,$$

for F and G CDFs of relevant rv.

Or its equivalent in 1D energy distance:

$$D^2(F, G) = 2\,\mathrm{E}\,\|X - Y\| - \mathrm{E}\,\|X - X'\| - \mathrm{E}\,\|Y - Y'\| \geq 0,$$

which in higher dimensions has got better behaviour:

- preserves nice properties of EMD;
- is proven to have unbiased sample gradients.

In general, one can use the same duality as in Wasserstein case.

From: M. Bellemare, arXiv:1705.10743

A. Gretton's post

Cramer GAN: results

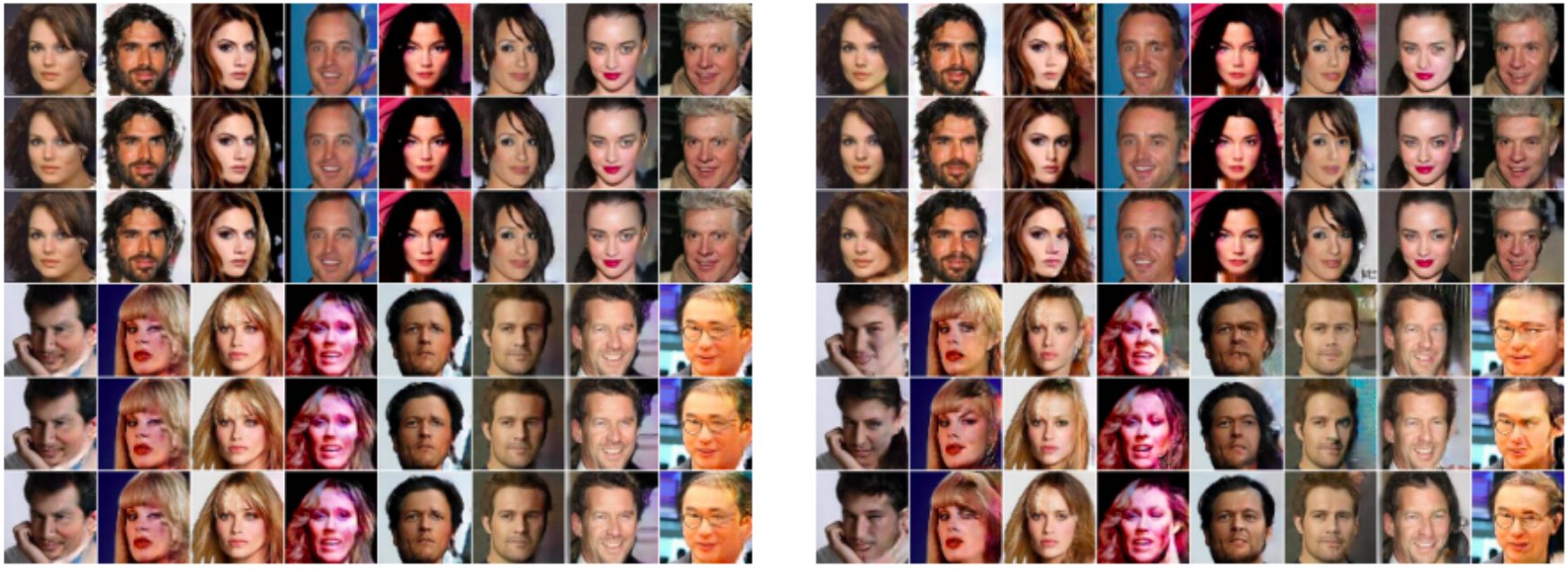


Figure 3: Generated right halves of the faces for WGAN-GP (left) and Cramér GAN (right). The given left halves are from CelebA 64x64 validation set (Liu et al., 2015).

Generally, useful approach, unfortunately paper was not resubmitted anywhere.

Wrap-up

- f-GAN generalizes learning using divergence, W-GAN leverages information from the sample space.
- WGAN allowed to use really deep models for image generation. Some tricks still need to be done:
  - Gradient penalty.
  - Spectral normalisation
- Some things are still missing, like unbiased gradients.

VDM derivation

Variational Estimation [Nguyen et al]: Jensen's inequality and a finite function class $\mathcal{T}$

$$
\begin{aligned}
D_f(P||Q) &= \int_{\mathcal{X}} q(x) f(\frac{p(x)}{q(x)}) dx \\
&= \int_{\mathcal{X}} q(x) \sup_{t \in \text{dom}_{f*}} \{t\frac{p(x)}{q(x)} - f^*(t)\} dx \\
&= \int_{\mathcal{X}} \sup_{t \in \text{dom}_{f*}} \{tp(x) - f^*(t)q(x)\} dx \\
&\geq \sup_{T \in \mathcal{T}} (\int_{\mathcal{X}} p(x)T(x)dx - \int_{\mathcal{X}} q(x)f^*(T(x))dx) \\
&= \sup_{T \in \mathcal{T}} (\mathbb{E}_{x \sim P}[T(x)] - \mathbb{E}_{x \sim Q}[f^*(T(x))])
\end{aligned}
$$

Derivation from J. Cunningham course