Generative models: generative adversarial networks and friends

26 ноября 2019 г.

Denis Derkach, Artem Ryzhikov, Maksim Artemev

Contents

Generative modeling

Generative vs Discriminative Modelling

Discriminative model

- learn $\mathbb{P}(y|x)$
- Directly characterizes the decision boundary between classes only
- Examples: Logistic Regression, SVM, etc

Generative model

- learn $\mathbb{P}(x|y)$ (and eventually $\mathbb{P}(x)$)
- Characterize how data is generated (distribution of individual class)
- Examples: Naive Bayes, HMM, etc.

Taxonomy of Generative Model Techniques

- Nonparametric
  - histograms
  - kernel density estimation
- likelihood-based parametric
  - autoregressive models
  - variational autoencoders
  - normalizing flow models
- likelihood-free parametric
  - Generative Adversarial Networks

Intuition for generation

Choosing the best metric

We need a metric that tells us that our modelled distribution is somewhere close to the real one. Ideally, it should be differentiable and have nice properties for convergence. From now on, lets denote $p(x)$ as true pdf and q_θ as its esimate.

Total Variation Distance

The first idea is to use something very straightforward, like for pdf's $p(x)$ and $q_\theta(x)$, $x \in \mathbb{R}^n$:

$$D(p(x), q_\theta(x)) = \frac{1}{2} \int_{\mathbb{R}^n} |p(x) - q_\theta(x)| dx,$$

which in fact corresponds to the Total Variation distance (using Scheffé's lemma).

Observations

- Symmetric: $D(P, Q) = D(Q, P)$.
- Connected to the hypotheses testing: $1 - D(P, Q)$ is equal to sum of false positives и false negatives.
- With growing number of trials, n, distance $D(f_{X^n}, g_{Y^n}) \to 1$. Moreover, if $D(f_X, g_Y) = \delta$, than for any $k \in \mathbb{N}$:
  $1 - 2e^{-k\frac{\delta^2}{2}} \leq D(f_{X^n}, g_{Y^n})$.
- Too strong. The distance might ignore the growing number of trials: $D(f_{X^2}, g_{Y^2}) = D(f_X, g_Y)$ (for example, $X_1, \ldots, X_n \sim \pm 1$, $S_n = \sum_n X_i$. Than

  $$S_n/\sqrt{n} \to \mathcal{N}(0, 1),$$

  but $D(S_n, Z) = 1$ for any n).

Kullback-Leibler Divergence: Definition

Let p(x) and q(x) are two probability distributions

$$KL(P||Q) = \int_{\mathbb{R}^n} p(x) \log \left(\frac{p(x)}{q(x)} \right) dx.$$

Although the KL divergence measures the “distance” between two distributions, it is not a distance measure.

KL and maximum likelihood estimate

Let θ^* be the true value of θ. Denote

$$M_n(\theta) = \frac{1}{n} \sum_i \log \frac{f(X_i; \theta)}{f(X_i; \theta_*)}$$

and $M(\theta) = -KL(\theta_*, \theta)$.

Let $\sup_{\theta \in \Theta} |M_n(\theta) - M(\theta)| \xrightarrow{P} 0$ and for any $\epsilon > 0$ $\sup_{\theta: |\theta - \theta_*| \geq \epsilon} M(\theta) < M(\theta_*)$.

If $\widehat{\theta}_n$ is the maximum likelihood estimate, than $\widehat{\theta}_n \xrightarrow{P} \theta_*$.

Cross Entropy and KL Divergence

Given two distributions p and q over a given variable X, the cross entropy is defined as

$$H(p, q) = \mathbb{E}_p(\log q)$$

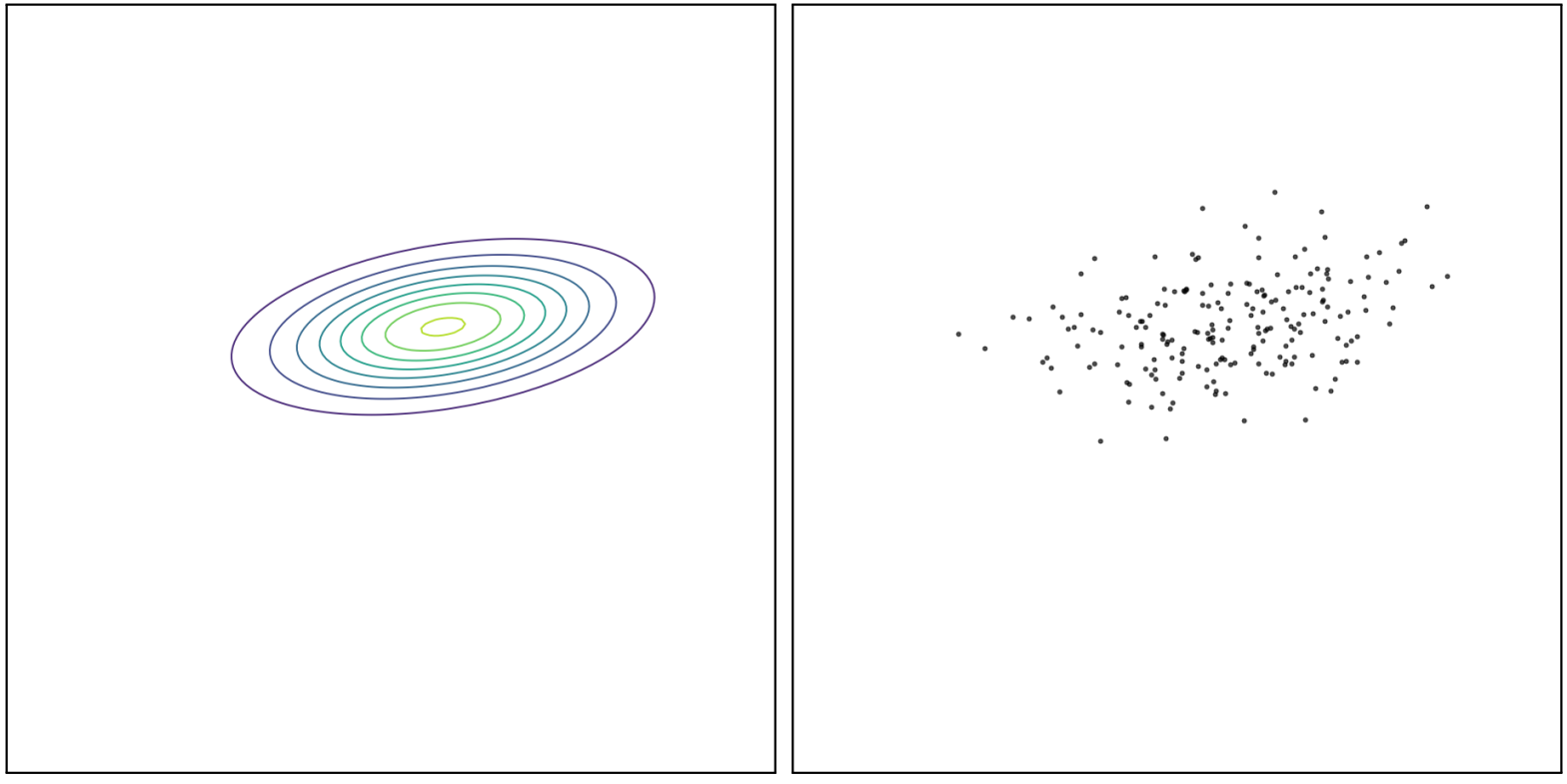
KL divergence is connected to it:

$$KL(p, q) = H(p) + H(p, q).$$

Since we normally optimise $L(\theta) = H(p_{data}, q(x))$, than optimisation of $KL \leftrightarrow$ optimisation of H.

Trying to converge

Let's check the convergence properties.Unfortunately, we do not have access to the true $p(x)$ during the study, so we must sample from it:



In the first part of our study we will using the 2D correlation Gaussian.

Here and later some examples are motivated by this blog.

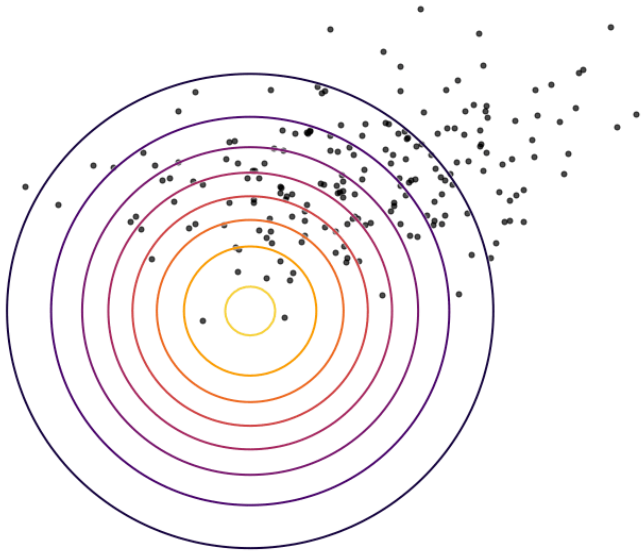
Optimal parameters

We need to get the optimal parameter, θ^* for our study. Let's do it by minimizing KL divergence.

$$
\begin{aligned}
\theta^* &= \arg\min_{\theta} KL(p(x)||q_\theta(x)) = \\
&= \arg\min_{\theta}(\mathbb{E}_{x\sim p}[\log p(x)] - \mathbb{E}_{x\sim p}[\log q_\theta(x)]) \\
&\quad \text{since } p(x) \text{ does not depend on } \theta = \\
&= \arg\min_{\theta} -\mathbb{E}_{x\sim p}[\log q_\theta(x)] = \\
&= \arg\max_{\theta} \mathbb{E}_{x\sim p}[\log q_\theta(x)].
\end{aligned}
$$

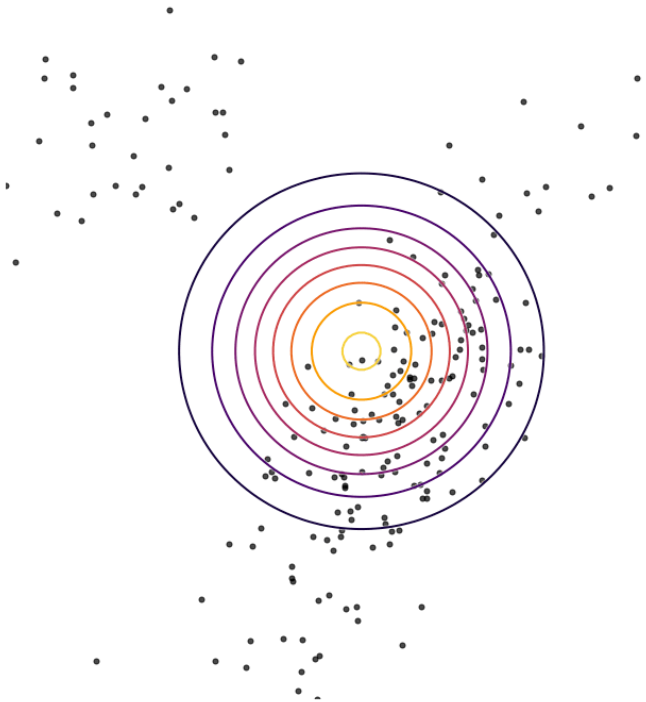
Which means that we want to find θ^* which assigns samples from $p(x)$ the highest possible $\log$ probability under $q_{\theta^*}(x)$.

Converging with KL



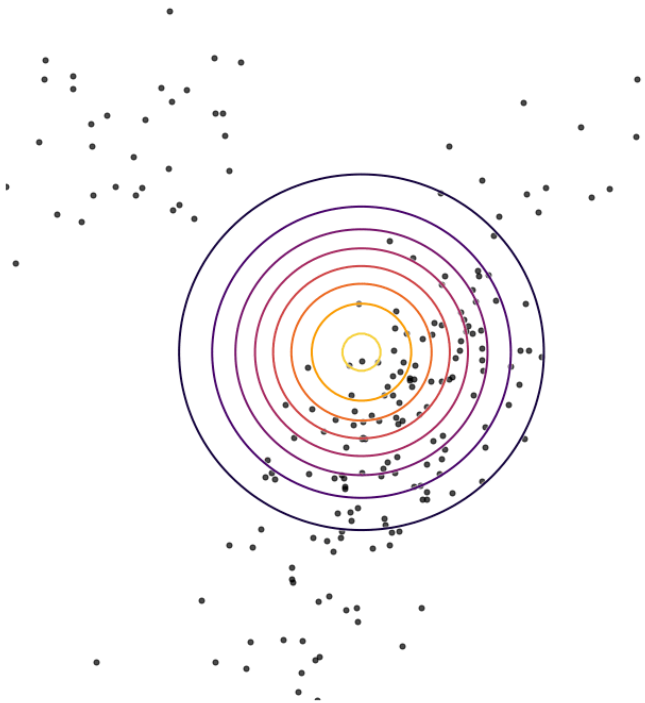
- The procedure works!
- Does it mean that the problem of building a model is solved?

Converging with KL: multimodal case



- The procedure works!
- Does it mean that the problem of building a model is solved?
- Not really, for the multimodal case, we will have problems.

Converging with KL: multimodal case intuition



- It's natural if we look at the quantity we optimize:

$$\arg\max_{\theta} \mathbb{E}_{x\sim p}[\log q_\theta(x)]$$

- If there is no $q_\theta(x)$ support in the place, where we have $x \sim p(x)$ than the optimised function goes to ∞.
- Automatically, we also have $q_\theta(x)$ support in places with no $x \sim p(x)$, which is also bad.

Reverse KL divergence

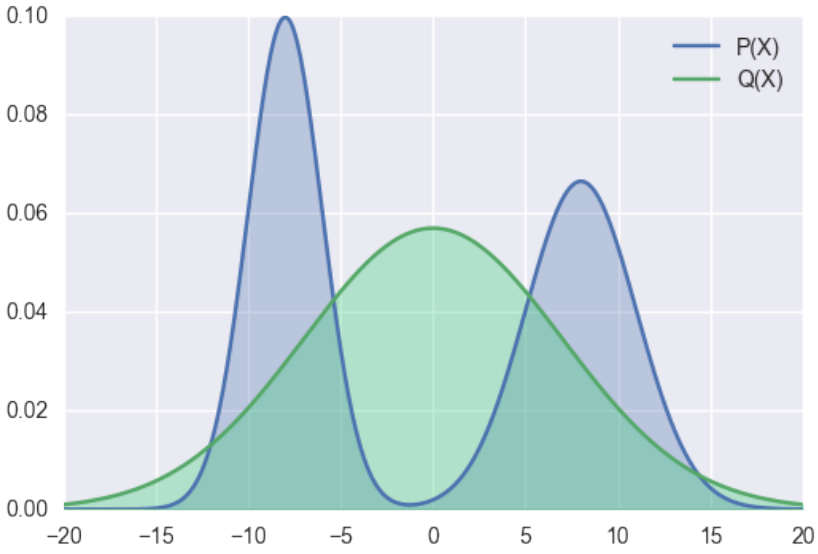
In order to overcome the problems, we can define a reverse divergence:

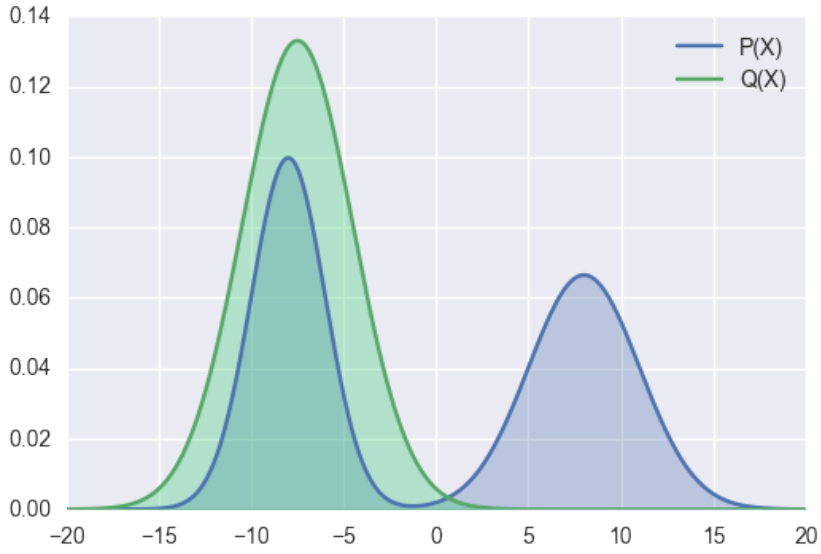
$$rKL(q_\theta||p) = \int_{\mathbb{R}^n} p(x) \log\left(\frac{p(x)}{q_\theta(x)}\right) dx.$$

Intuition

$$KL = \int p(X) \log \frac{p(x)}{q_\theta(x)} dx$$

$$rKL = \int q(x) \log \frac{q_\theta(x)}{p(x)} dx$$





Forward KL is known as zero avoiding, as it is avoiding $q(x) = 0$ whenever $P(x) > 0$.

Reverse KL Divergence is known as zero forcing, as it forces $Q(X)$ to be 0 on some areas, even if $P(X) > 0$.

Picture credit:https://wiseodd.github.io/techblog/2016/12/21/forward-reverse-kl/

rKL: optimisation

In fact, we are optimizing a very similar thing:

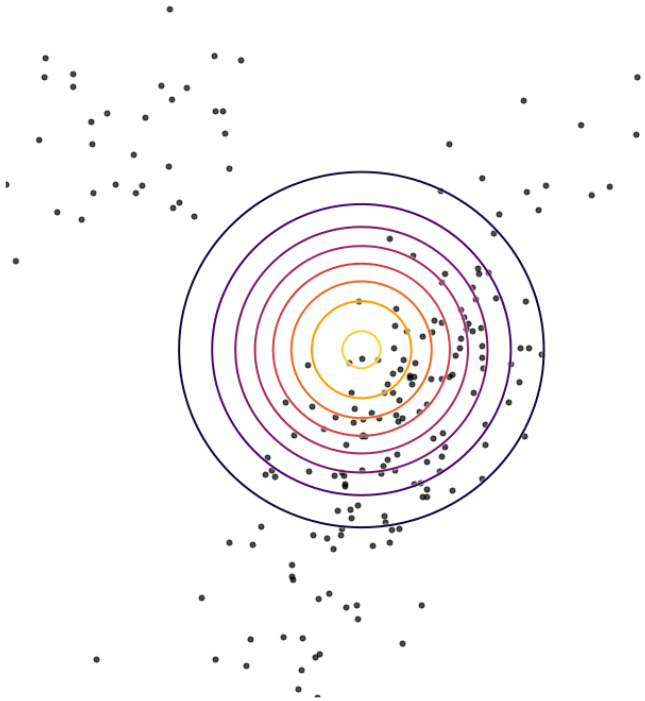
$$\begin{aligned}\theta^* &= \arg\min_\theta KL(q_\theta(x)||p(x)) = \\ &= \arg\min_\theta(\mathbb{E}_{\tilde{x}\sim q_\theta}[\log q_\theta(x)] - \mathbb{E}_{\tilde{x}\sim q_\theta}[\log p(x)]) = \\ &= \arg\max_\theta(-\mathbb{E}_{\tilde{x}\sim q_\theta}[\log q_\theta(x)] + \mathbb{E}_{\tilde{x}\sim q_\theta}[\log p(x)])\end{aligned}$$

But we do not have the previous problem of likelihood going to infinity in unreasonable places.

The first term is related to entropy of the generating model, the second penalises generated samples that are not similar to real distribution.

Let's check whether it works.

Converging with rKL



- We no longer have $q_\theta(x)$ support in the regions with no $x \sim p(x)$ population.
- The converged distribution looks reasonable but only for one solution.

The main problem: optimised expression depends on the $p(x)$

$$\arg\max_\theta(-\mathbb{E}_{\tilde{x}\sim q_\theta}[\log q_\theta(x)] + \mathbb{E}_{\tilde{x}\sim q_\theta}[\log p(x)]),$$

which we normally do not have.

Jensen-Shannon Divergence

We can try to optimize different divergences however, the problems normally stay. A distinguishable attempt is to construct the mixture of KL and rKL:

$$\begin{aligned} JS(p(x)||q_\theta(x)) = & \quad \frac{1}{2} \quad KL(p(x)||\frac{p(x)+q_\theta(x)}{2}) + \\ & + \quad \frac{1}{2} \quad KL(q_\theta(x)||\frac{p(x)+q_\theta(x)}{2}), \end{aligned}$$

It is symmetric and does not ignore zeroes like KL and does not ignore x like rKL.

Generative Adversarial Networks

Rationale

What if we could construct a way to approximate the previous algorithm but without the direct access to the $p(x)$? One of the possible estimates in this case would look like:

$$\theta^* = \arg\min_{\theta} \max_{\phi} \mathbb{E}_{x \sim p, \tilde{x} \sim q_\theta} V(f_\phi(x), f_\phi(\tilde{x}))$$

So, instead of minimizing over some analytically defined divergence, we could minimize over "learned divergence". Let's expand on the way how to obtain it.

Generator

Let's look closely at the generator function. In fact, it should sample from a random noise source. We thus can write:

$$z_j \sim \mathcal{N}(0; 1),$$
$$\hat{x}_j = G_\theta(z_j)$$

where $G_\theta(z_j) : z_j \mapsto x_j$ can be defined in many ways (see talk by M. Borisyak on Friday) but we limit ourselves to neural network. We thus have a sample

$$\{\tilde{x}_j\} \sim q_\theta(x).$$

Discriminator

Following our idea, we add also another network, We add a classifying network D_ϕ (discriminator) to distinguish between the real and generated samples and train both as follows:

$$\max_\phi \left(\mathbb{E}_{x \sim p(x)} (\log(D_\phi(x)) + \mathbb{E}_{\tilde{x} \sim q_\theta(x)} (1 - \log(D_\phi(\tilde{x})) \right) .$$

The first term serves to recognise the real images better. The second term for the generated images.

G+D Recap

We can now put together generator and discriminator.

- objective of discriminator:

$$\max_{\phi} \left(\mathbb{E}_{x \sim p(x)} (\log(D_\phi(x)) + \mathbb{E}_{z \sim \mathcal{N}(0;1)} (1 - \log(D_\phi(G_\theta(z))) \right).$$

- objective of generator:

$$\min_{\theta} \mathbb{E}_{z \sim \mathcal{N}(0;1)} (1 - \log(D_\phi(G_\theta(z))$$

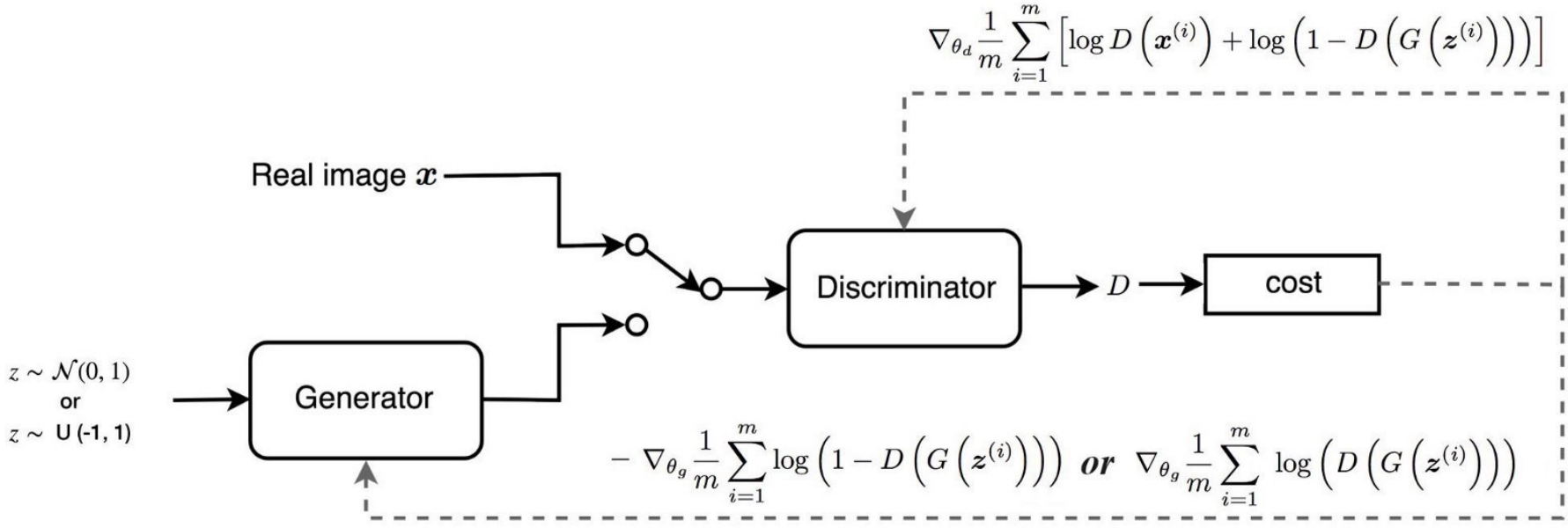
We thus defined a minimax game:

$$\min_{\theta} \max_{\phi} \mathbb{E}_{x \sim p, \tilde{x} \sim q_\theta} V(f_\phi(x), f_\phi(\tilde{x})).$$

In exactly the way we wanted.

Graphical Representation

We can define both generator and discriminator as neural networks:



and use the full power of backpropagation.

Optimal Solution

› For a given generator, the optimal discriminator is:

$$D^*_\phi(G) = \frac{p(x)}{p(x) + q_\theta(x)}.$$

› Incorporating that into the minimax game to yield virtual training criterion:

$$\begin{aligned} C(G) &= \max_D V(G, D) = \\ &= \mathbb{E}_{x\sim p(x)}(\log(D^*_\phi(x))) + \mathbb{E}_{x\sim q_\theta}(1 - \log(D^*_\phi(G_\theta(z)))) = \\ &= \mathbb{E}_{x\sim p(x)} \frac{p(x)}{p(x) + q_\theta(x)} + \mathbb{E}_{x\sim q_\theta} \frac{q_\theta(x)}{p(x) + q_\theta(x)} \end{aligned}$$

Optimal Solution

- In an optimal case $p = q_\theta$, we have $C(G) = -\log(4)$.
- We thus can write out:

$$C(G) = -\log(4) + \frac{1}{2} KL(p(x)||\frac{p(x) + q_\theta(x)}{2}) + \\ + \frac{1}{2} KL(q_\theta(x)||\frac{p(x) + q_\theta(x)}{2}).$$

- In other words, we effectively optimize Jensen-Shannon divergence:

$$C(G) = -\log(4) + JS(p(x)||q_\theta(x)).$$

- Reminder: we did it without access to $p(x)$.
- In general, we can effectively optimize any divergence by constructing correct minimax criteria.

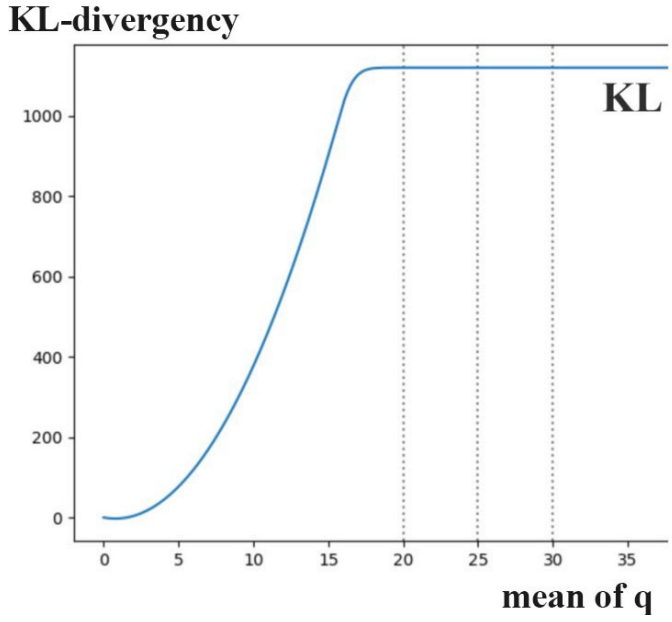
GANs: Pros and Cons

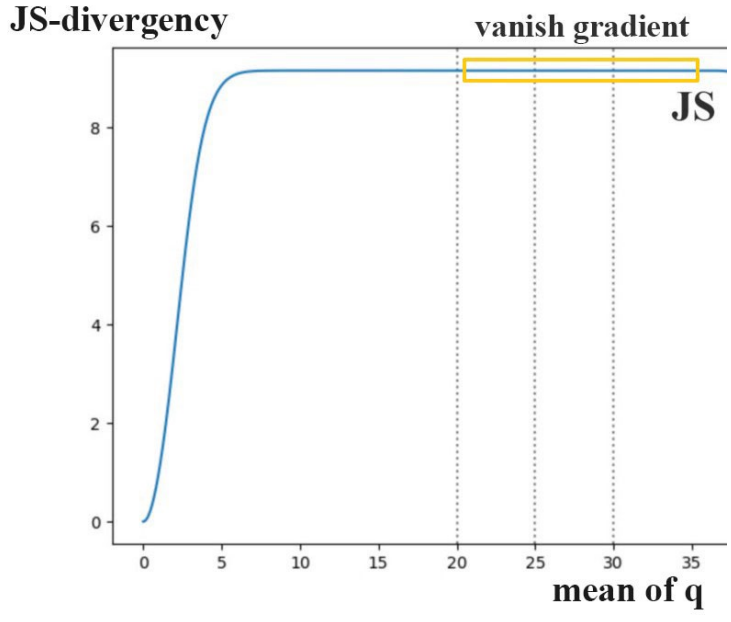
- Pros:
  - Can utilise power of back-prop.
  - No explicit intractable integral.
  - No MCMC needed.
- Cons:
  - Unclear stopping criteria
  - No explicit representation of $g_\theta(x)$
  - Hard to train
  - No evaluation metric so hard to compare with other models
  - Easy to get trapped in local optima that memorize training data
  - Hard to invert generative model to get back latent z from generated x

Even more problems

We effectively optimize JS divergence, this creates nuisances:

- Mode collapse: we explicitly choose one solution over others.
- Diminished gradients: if we start too far away, we risk never get to the solution.



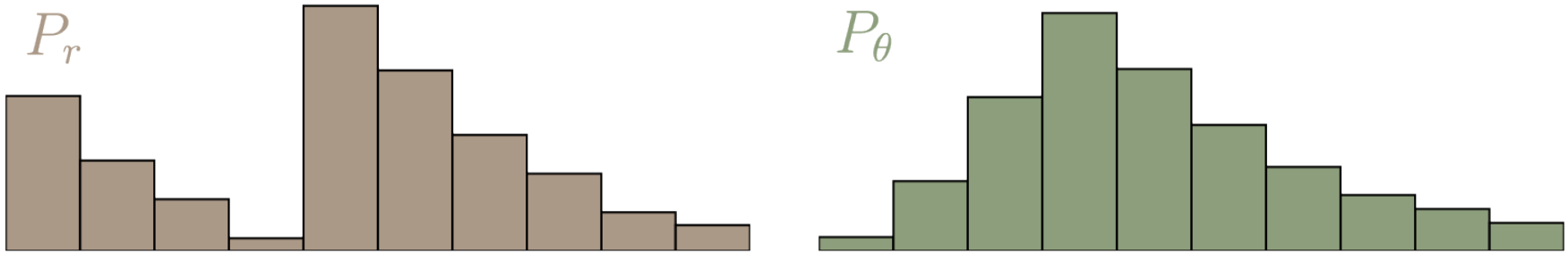


These are connected to the divergences, can we find another one?

Wassertein distance

Motivation

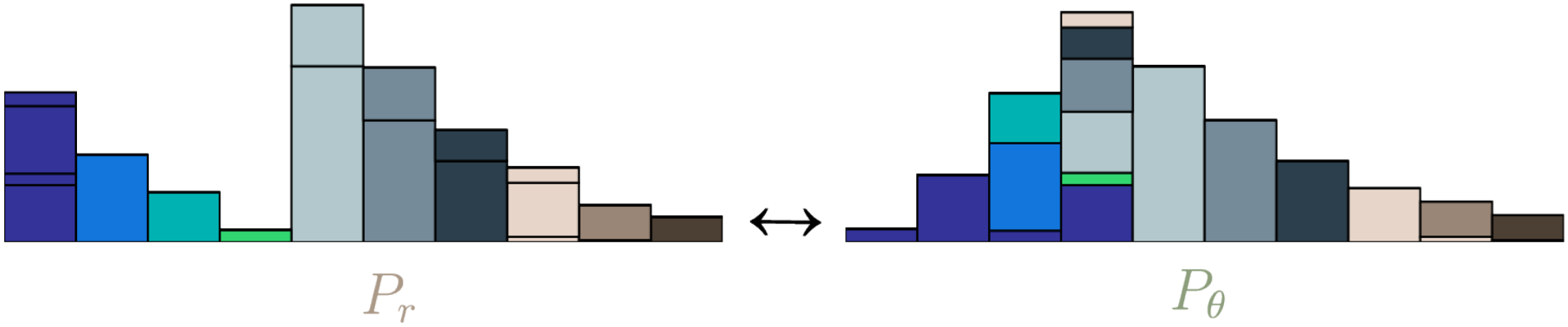
Imagine that we want to move the events from P_r to P_θ. We also want to save effort, that is, not to move large pieces over long distances.



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.

Picture credit:https://vincentherrmann.github.io/blog/wasserstein/

Earth Mover's Distance



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

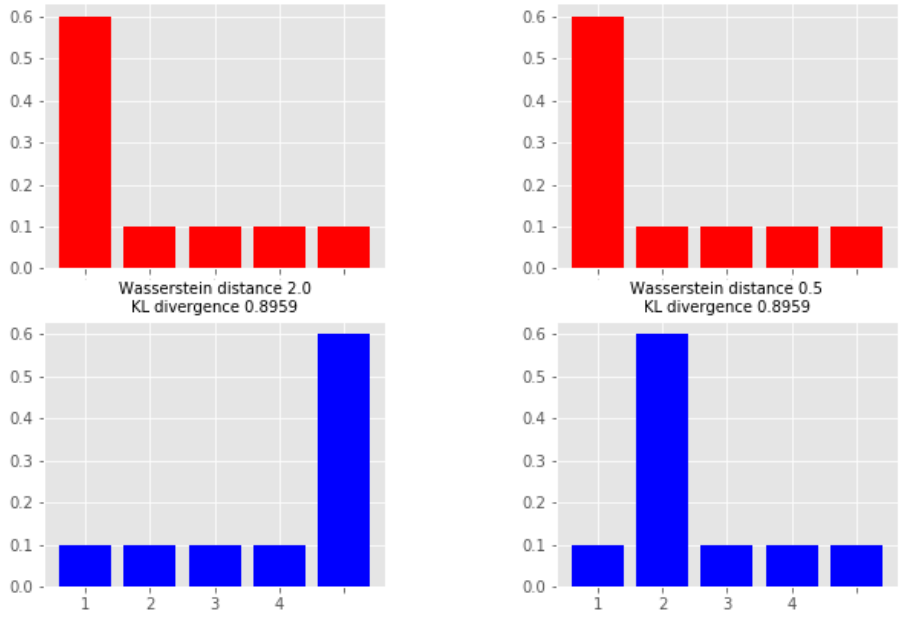
where $\gamma(x, y)$ are the efforts to move from x to y, Π – all possible transfers from P_r to P_θ, $\gamma \in \Pi$.

Easily can be wrote down for continuous observables (and become Wassestein Distance).

Picture credit:https://vincentherrmann.github.io/blog/wasserstein/

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!



Picture credit:https://goo.gl/ncx3gt

Kantorovich-Rubinstein Duality

The EMD metric (or 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 $\mathrm{Lip}_1(\mathrm{X})$: $|f(x) - f(y)| \leq d(x, y)$.
Interestingly enough, we can use a neural network with limited (clipped) weights to fulfil the $\mathrm{Lip}_1(\mathrm{X})$ condition.
We can construct a GAN that effectively fits W-distance.

WGAN vs JSGAN

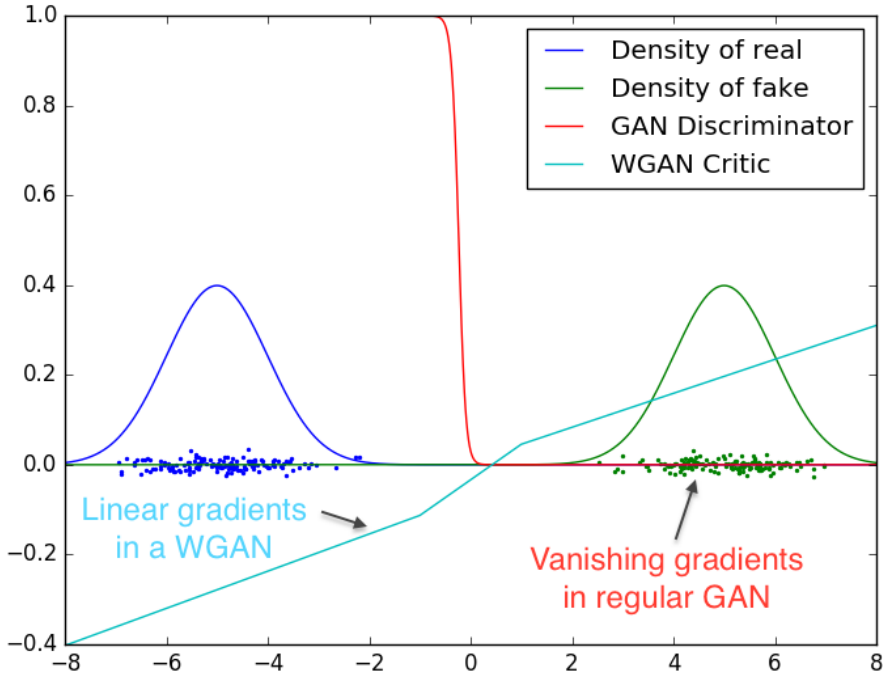
	Discriminator/Critic	Generator
GAN	$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$	$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$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

- WGAN has a simpler way to train.
- The optimisation runs over W-metrics, which has got better properties.

WGAN: problems solved

- mode collapse problem is addressed;
- the vanishing gradient problem is solved.
- a better solution instead of clipping weights is to introduce penalty to gradients in the loss (WGAN-GP):

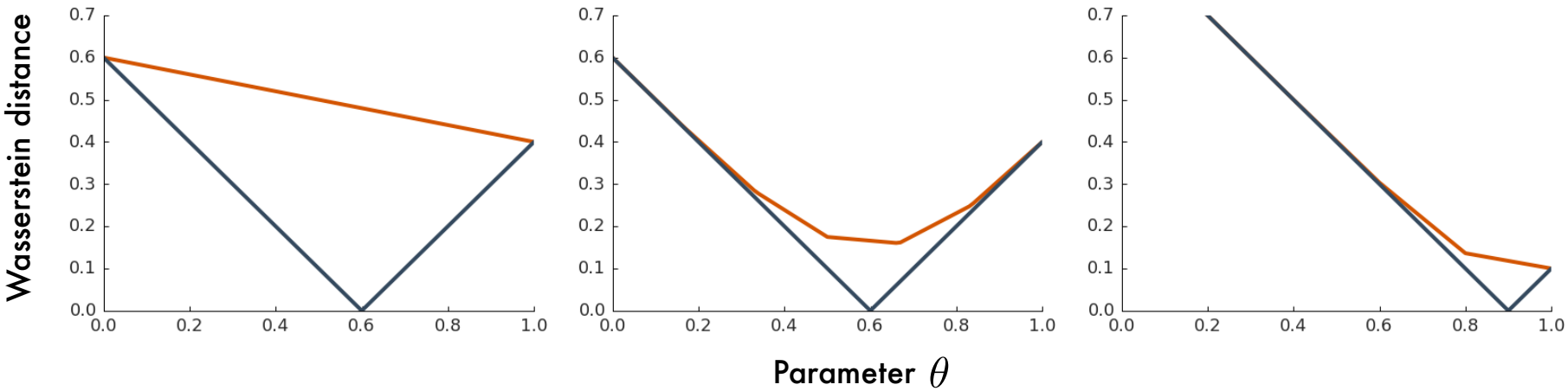
$$GP = \lambda \mathbb{E}\left[(||\nabla f|| - 1)^2\right]$$



From: arXiv:1701.07875

WGAN: more problems

- The expected EMD gradients can differ from the true gradients.
- This leads to problems even for Bernoulli distribution.
- Solution: update W-distance such that it reminds energy distance.



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

From: arXiv:1705.10743

GAN-story so far

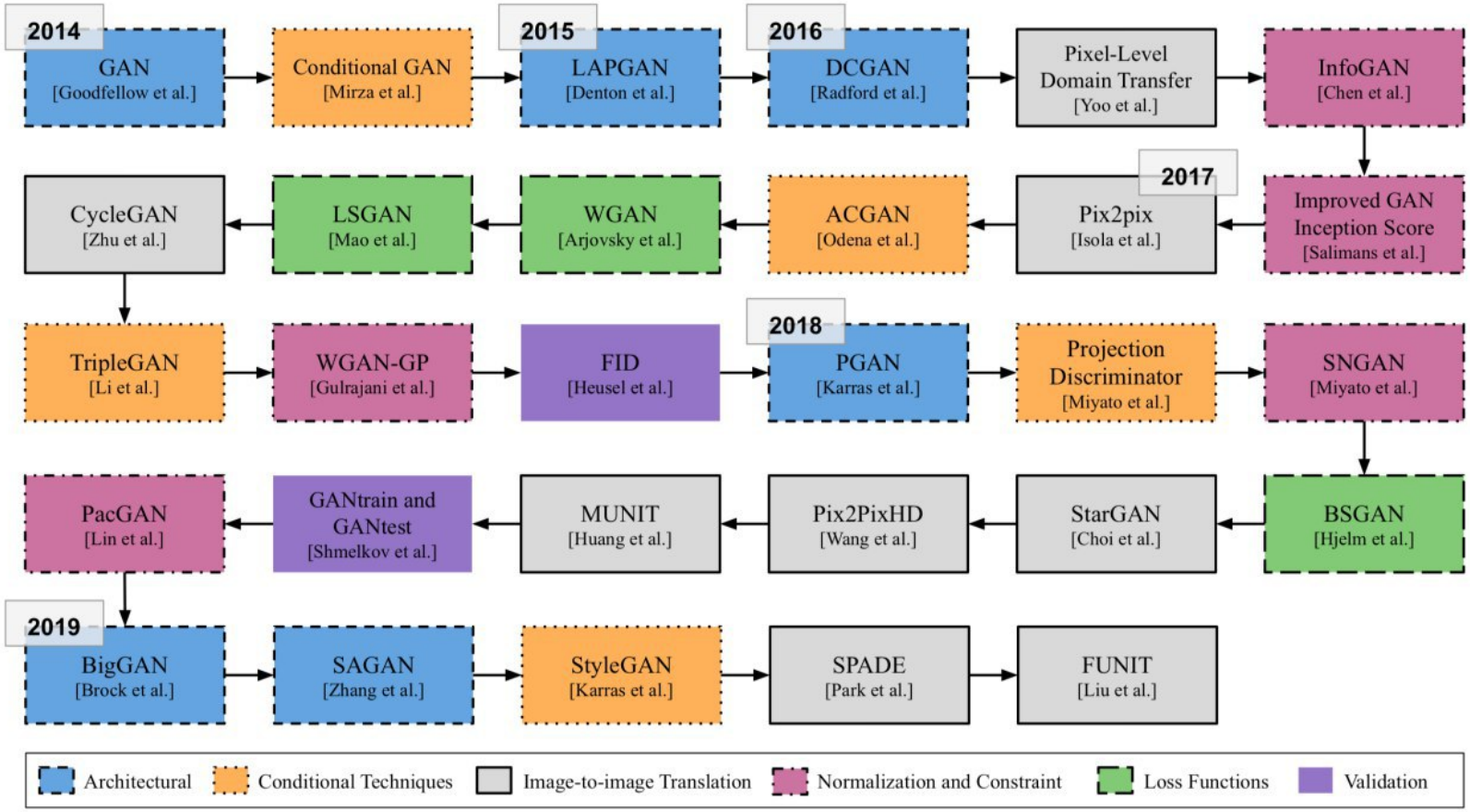


Fig. 1: Timeline of the GANs covered in this paper. Just like our text, we split it in six fronts (architectural, conditional techniques, normalization and constraint, loss functions, image-to-image translation and validation metrics), each represented by a different color and a different line/border style.

From: arXiv:1910.13076

Summary

- GANs use Generator-Discriminator game to estimate the distance from generated distribution to the true one.
- Wasserstein GAN is a useful technique that allows really deep networks to be used for data generation.
- The story is still developing.