Generative adversarial networks

20 февраля 2020 г.

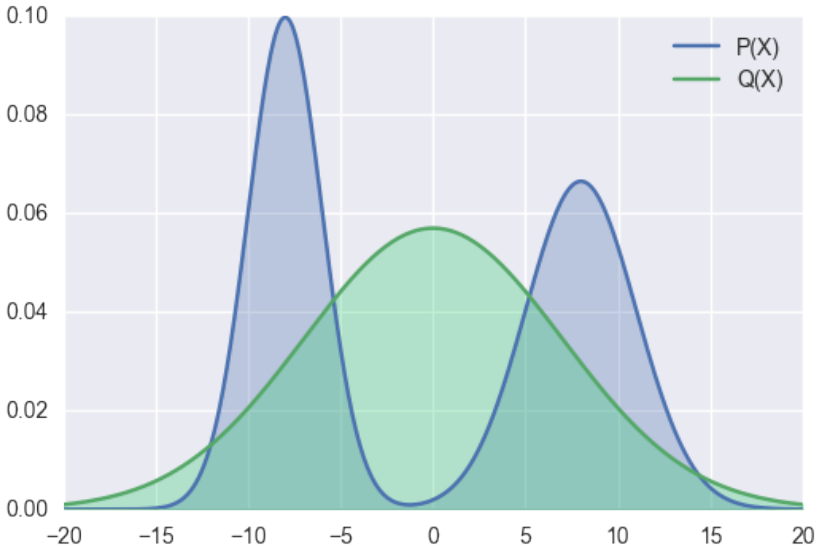
Denis Derkach, Artem Ryzhikov, Maksim Artemev

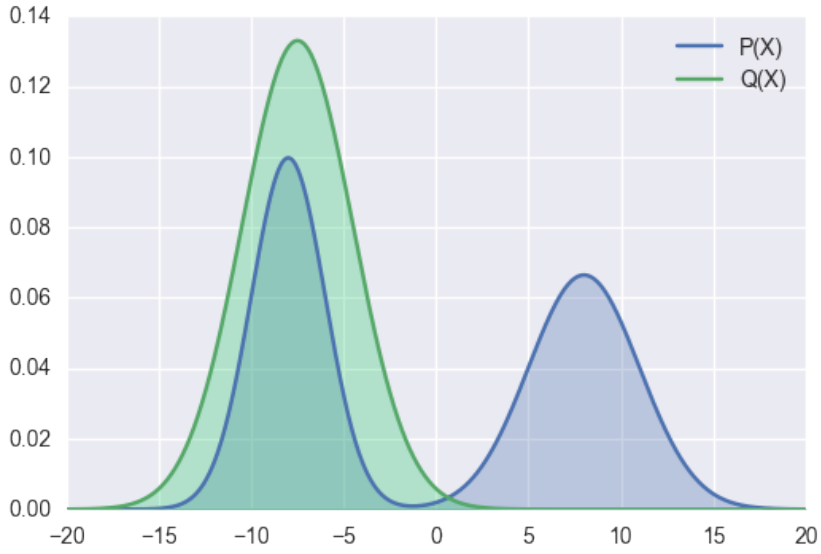
Contents

Divergence
convergence

Reminder: Divergences

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





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

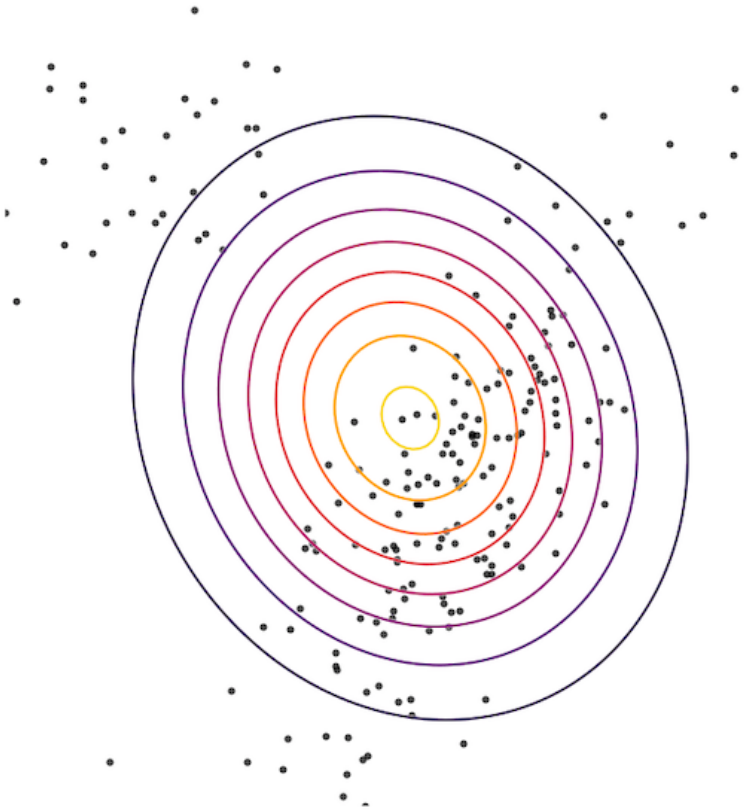
Reverse KL Divergence is known as zero forcing, as it forces $Q(X)$ to be 0 on some areas, even if $P(X) > 0$.

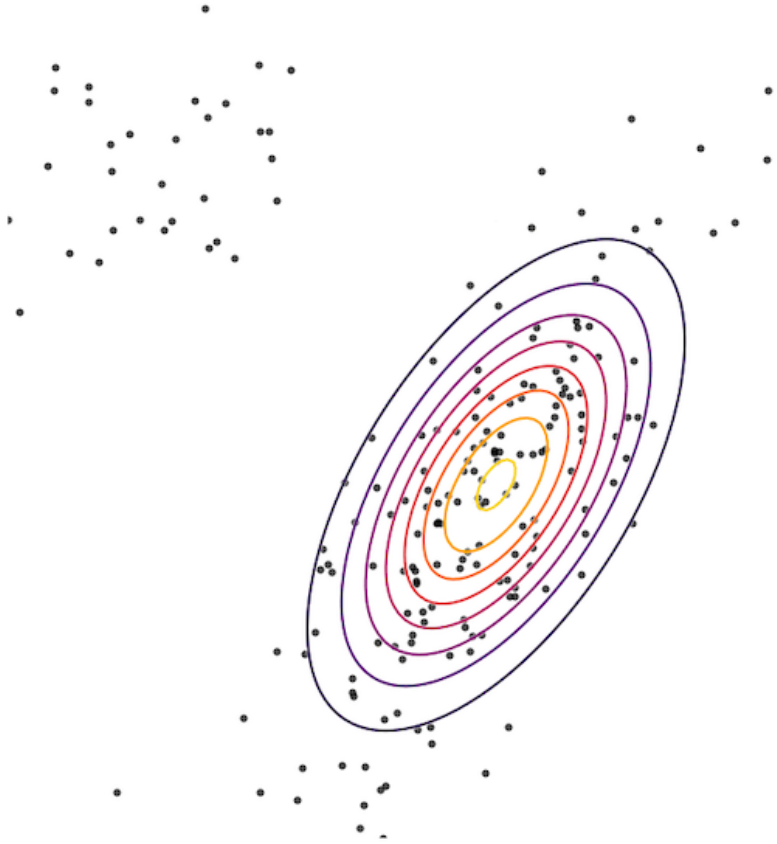
Picture credit:https://wiseodd.github.io/techblog/2016/12/21/forward-reverse-kl/

Reminder: Converging with divergence

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

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





Reminder rKL: optimisation

The rKL is a bit more preferable, but still has a problem:

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

To optimize it properly we need an access to $p(x)$.

Reminder: 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. Still we need access to $p(x)$.

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:

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

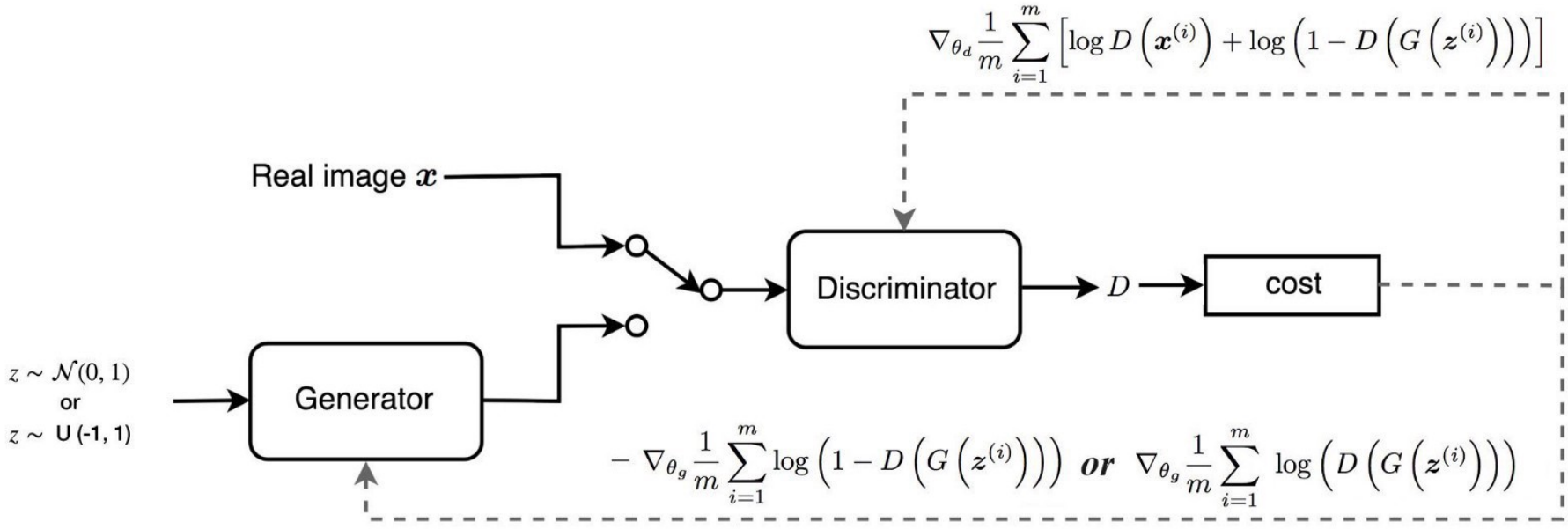
We thus defined a minimax game:

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

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.

GAN algorithm

1. Sample data mini-batch ($x_1, .., x_m \sim D$).
2. Sample generator mini-batch ($z_1, .., z_m \sim q_\theta$).
3. Use SGD to obtain new weights of generator:

$$\nabla_\theta V(G_\theta, D_\phi) = \frac{1}{m} \sum_{i=1}^{m} \log(1 - D_\phi(G_\theta(z_i))).$$

4. Use SGD to obtain new weights of discriminator:

$$\nabla_\phi V(G_\theta, D_\phi) = \frac{1}{m} \sum_{i=1}^{m} \left(\log D_\phi(x_i) + \log(1 - D_\phi(G_\theta(z_i))) \right).$$

5. Repeat with several epochs.

GAN:results

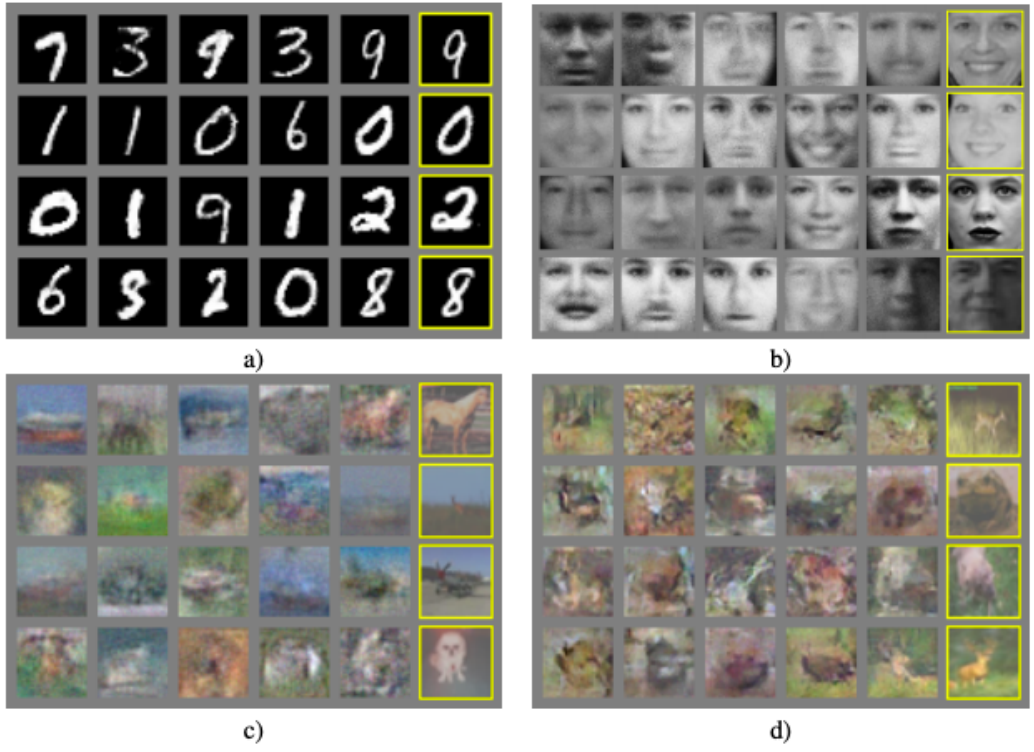
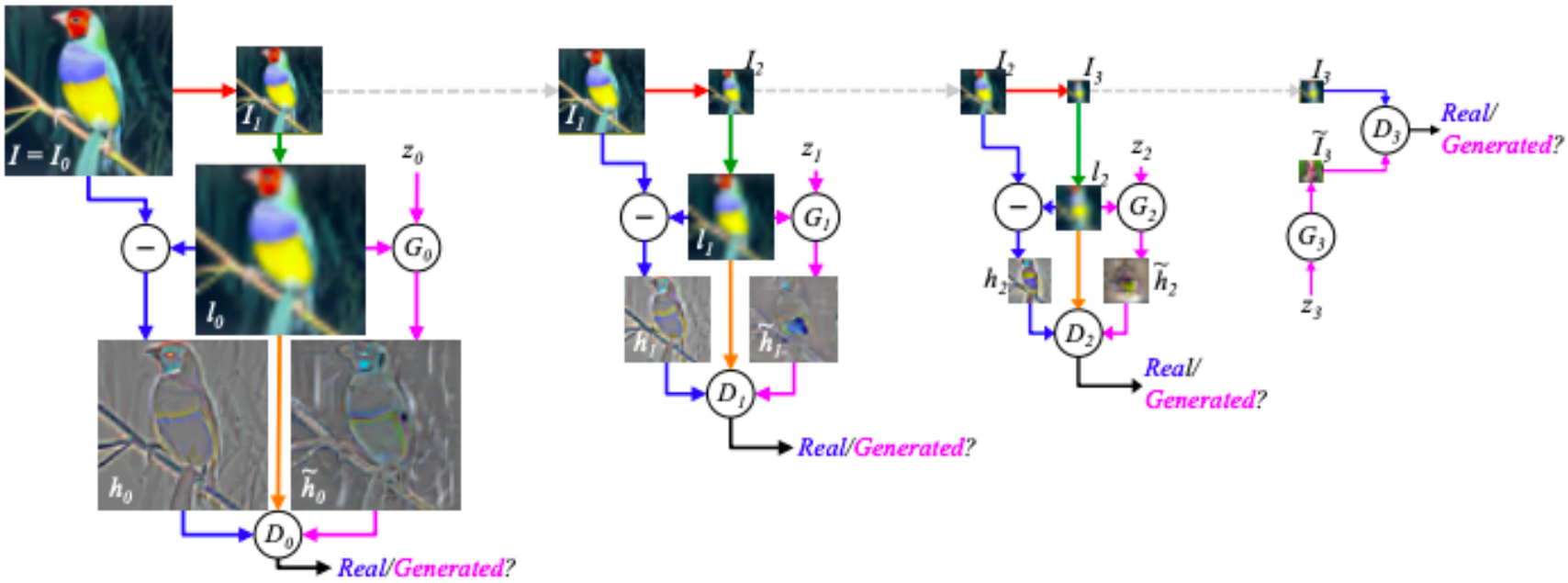


Figure 2: Visualization of samples from the model. Rightmost column shows the nearest training example of the neighboring sample, in order to demonstrate that the model has not memorized the training set. Samples are fair random draws, not cherry-picked. Unlike most other visualizations of deep generative models, these images show actual samples from the model distributions, not conditional means given samples of hidden units. Moreover, these samples are uncorrelated because the sampling process does not depend on Markov chain mixing. a) MNIST b) TFD c) CIFAR-10 (fully connected model) d) CIFAR-10 (convolutional discriminator and "deconvolutional" generator)

From :Goodfellow et al. Generative Adversarial Networks, NIPS14

Unsupervised Feature Learning

We need to decrease the size of images to make a faster convergence.
Let us make a Laplacian pyramid of images.



The representation is learned gradually.

From :Denton et al.Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks, NIPS15

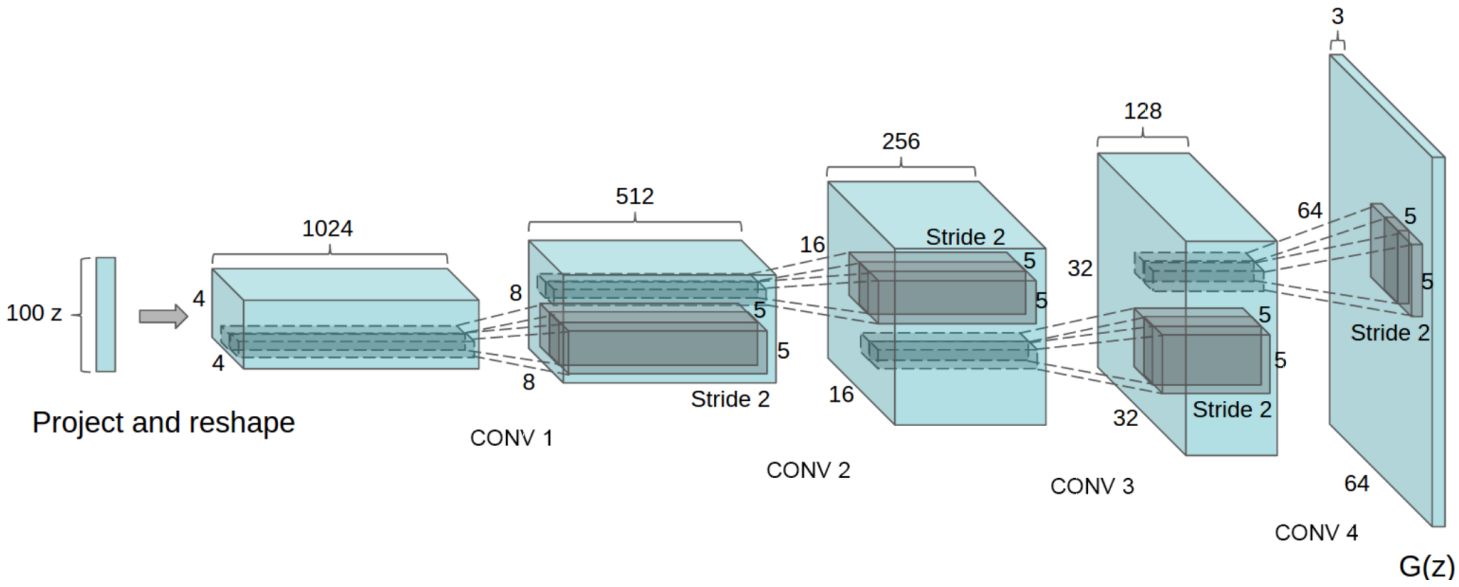
LAPGAN: results

(a) (b)

Figure 4: STL samples: **(a)** Random 96x96 samples from our LAPGAN model. **(b)** Coarse-to-fine generation chain.

Unsupervised Feature Learning

Dealing with images mus involve convolutional neural networks.



With a bit of magic, we can make it work.

From :Radford et al. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR2016

DCGAN: results

Figure 9: Side-by-side illustration of (from left-to-right) the MNIST dataset, generations from a baseline GAN, and generations from our DCGAN .

Latent space walk

In the same paper, authors showed that a walk in latent space can bring a smooth transition in images.

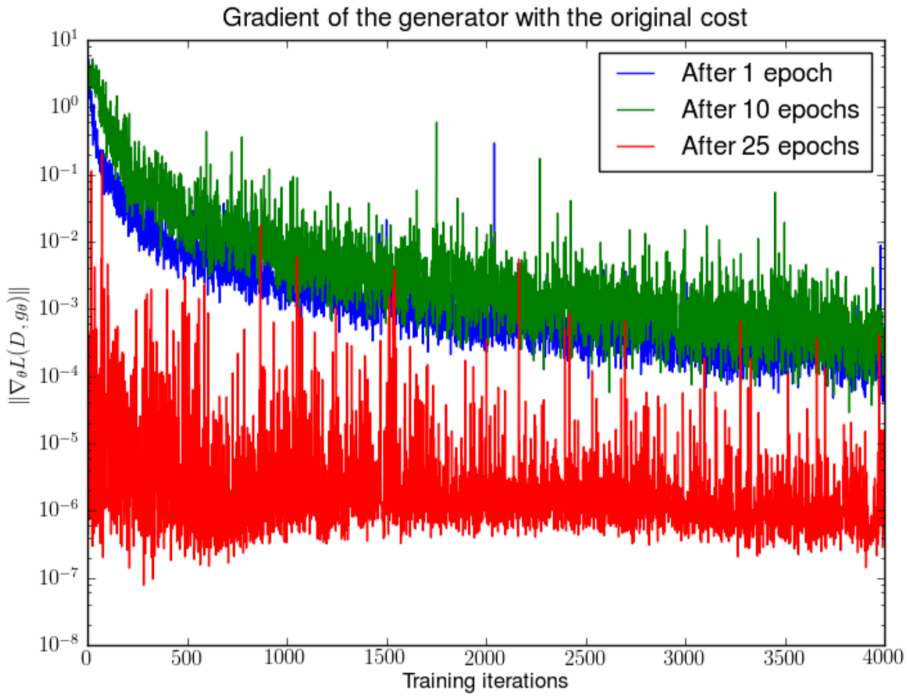
This is interesting, since it means that in latent space we move across the manifold.

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

Game approach problems

- Discriminator must be optimal on every step of convergence.
- This is not true, you should not overtrain discriminator.
- Loss-function can be quite noisy.



No clear stopping rules. No equilibrium can be achieved.

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

Divergence-related 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.

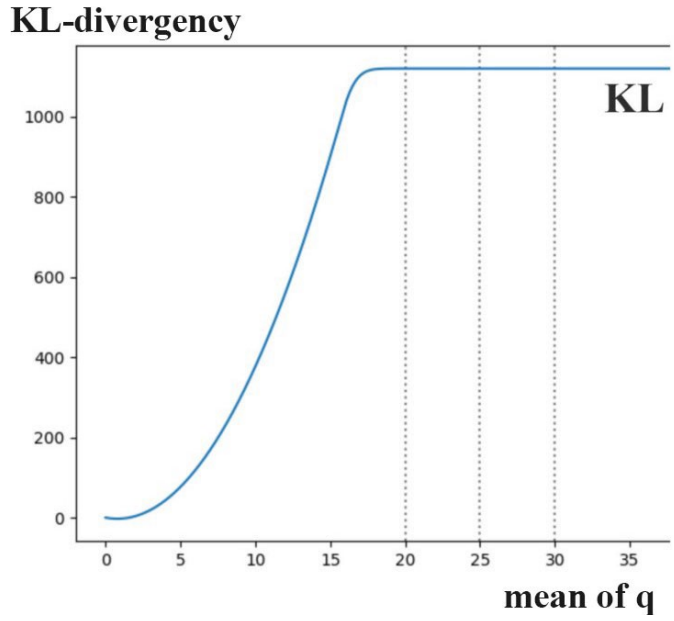
Mode Collapse

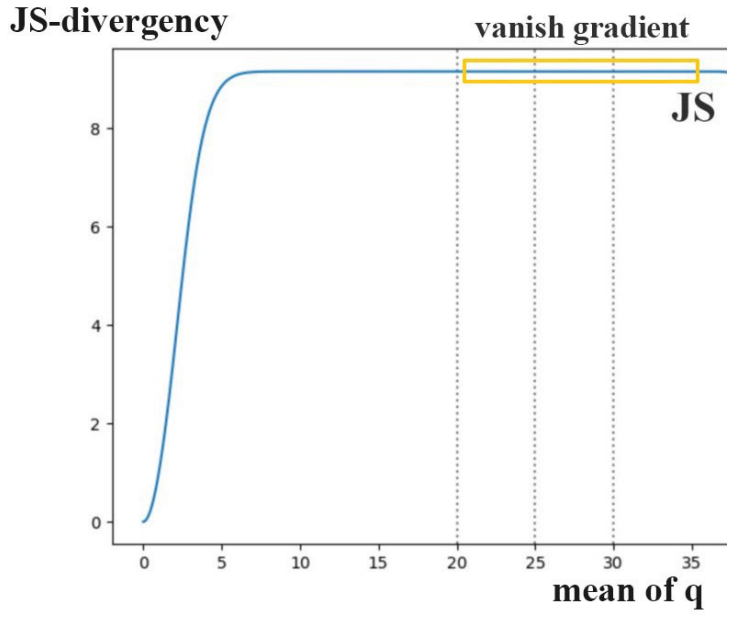
Mode Collapse is the property of JS divergence that is effectively minimised in GAN training.

From :Metz et al. Unrolled Generative Adversarial Networks

JS Diminished Gradients

Diminished gradients: if we start too far away, we risk never get to the solution.

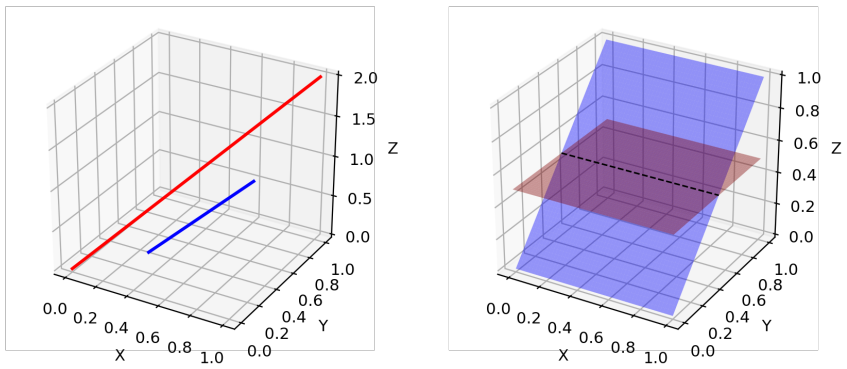




This is further enhanced by noise due to data sample distribution.

Ways to address: making your manifold noisy

› We have seen already that data is normally located on manifold.



› GAN case is in fact more complicated, as we need a discriminator that distinguishes two supports.

› This is easy, if supports are disjoint.

Ways to address: making your manifold noisy

- Let's make the problem harder: 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.

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

Feature Matching

- Change the objective of the generator:

$$||\mathbb{E}_{x \sim p(x)} f(x) - \mathbb{E}_{z \sim p_z(z)} f(G(z))||^2$$

- Here $f(x)$ can be any property we need (including the output of another network.
- No guarantees, but it works.

From :Salimans et al. Improved Techniques for Training GANs, NIPS16

Feature Matching results

Figure 4: Samples generated during semi-supervised training on CIFAR-10 with feature matching (Section 3.1, *left*) and minibatch discrimination (Section 3.2, *right*).

From :Salimans et al. Improved Techniques for Training GANs, NIPS16

Historical averaging

- average with previous parameter values:

$$||\theta - \frac{1}{t}\sum_{i=1}^{t}\theta[i]||^2$$

- this allows to create a fake agent that plays the game.
- and solves the problems only in low dimensions.

From :Salimans et al. Improved Techniques for Training GANs, NIPS16

Look into future

- What if we can look into the future of system?
- We could avoid local optima and optimize better.
- Algorithm:
  - At each step we train discriminator 5-10 steps ahead.
  - We DO NOT introduce it to the system.
  - We show the possible future moves to generator and update it accordingly.

Unrolled GAN: results

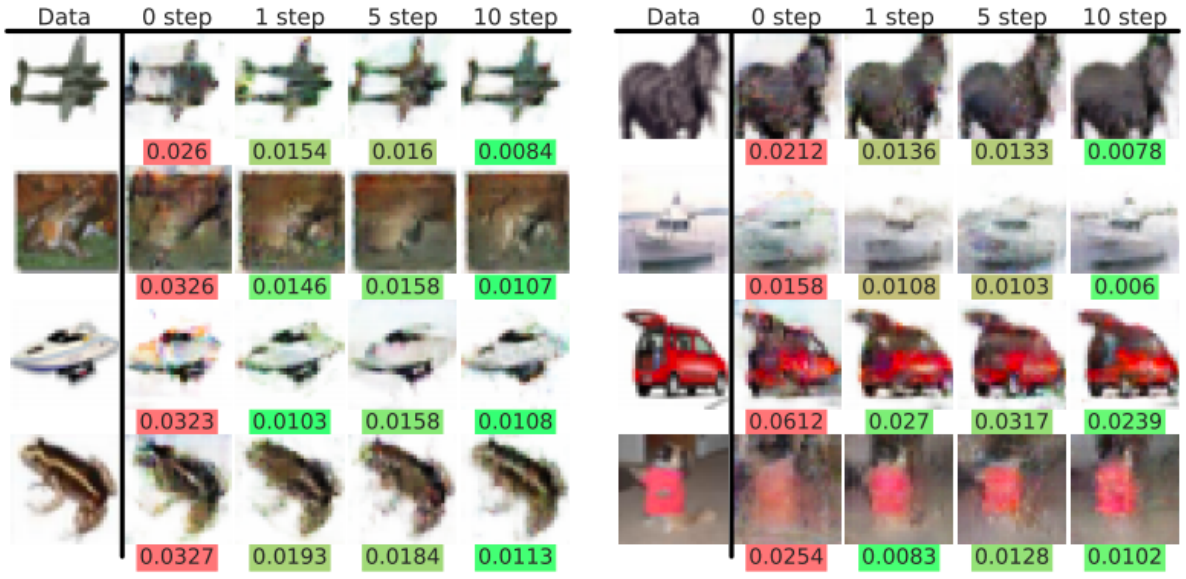


Figure 5: Training set images are more accurately reconstructed using GANs trained with unrolling than by a standard (0 step) GAN, likely due to mode dropping by the standard GAN. Raw data is on the left, and the optimized images to reach this target follow for 0, 1, 5, and 10 unrolling steps. The reconstruction MSE is listed below each sample. A random 1280 images where selected from the training set, and corresponding best reconstructions for each model were found via optimization. Shown here are the eight images with the largest absolute fractional difference between GANs trained with 0 and 10 unrolling steps.

From :Metz et al. Unrolled Generative Adversarial Networks, NIPS16

Constructing the divergence

- Can we change the divergence we use?
- Yes, if we know its lower bound

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

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

- It appears:

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

- From the above equation, we can find $q(x)$ condition on saddle point.

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

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.

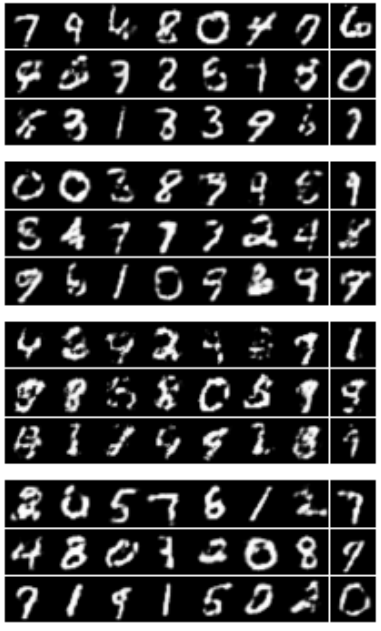


Figure 2: MNIST model samples trained using KL, reverse KL, Hellinger, Jensen from top to bottom.

Summary

- GANs use Generator-Discriminator game to estimate the distance from generated distribution to the true one.
- A zoo of techniques is available.
- The story is still developing.