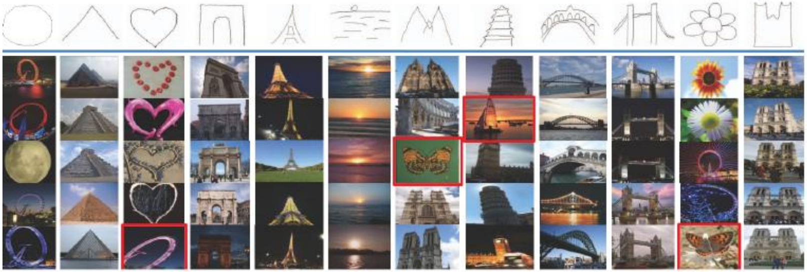
Deep Learning

Caio Corro – Michele Sebag
CNRS – INRIA – LIMSI – LRI

Orsay – Oct. 2019

Credit for slides: Sanjeev Arora; Yoshua Bengio;Yann LeCun; Nando de Freitas; Pascal Germain; Léon Gatys; Weidi Xie; Max Welling; Victor Berger; Kevin Frans; Lars Mescheder et al.; Mehdi Sajjadi et al.

Siamese Networks



Classes or similarities ?

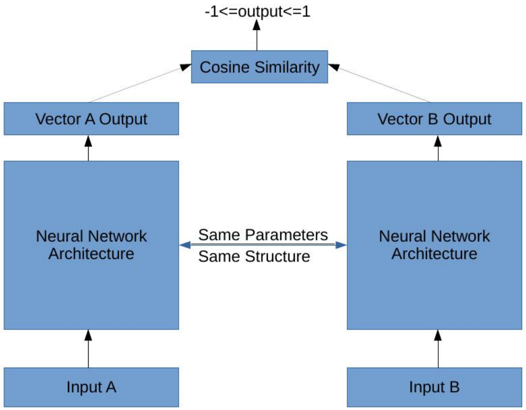
Siamese Networks

Bromley et al. 93

Principle

- Neural Networks can be used to define a latent representation
- Siamese: optimize the related metrics

Schema



Siamese Networks, 2

Data

$$\mathcal{E} = \{x_i \in \mathbb{R}^d, i \in [[1, n]]\}; \mathcal{S} = \{(x_{i,\ell}, x_{j_\ell}, c_\ell) \ s.t. \ c_\ell \in \{-1, 1\}, \ell \in [[1, L]]\}$$

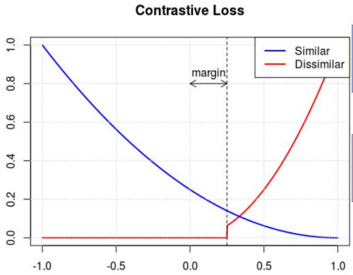
Experimental setting

- Often: few similar pairs; by default, pairs are dissimilar
- Subsample dissimilar pairs (optimal ratio between 2/1 ou 10/1)
- Possible to use domain knowledge in selection of dissimilar pairs

Loss

Given similar and dissimilar pairs (E_+ and E_-)

$$\mathcal{L} = \sum_{(i,j) in E_+} L_+(i,j) + \sum_{(k,\ell) in E_-} L_-(k,\ell)$$



$$L_+ = \frac{1}{4} \cdot (1 - cosine(x_1, x_2))^2$$

$$L_- = \begin{cases} \text{cosine}(x_1, x_2)^2, & \text{if } x \geq \text{margin} \\ 0, & \text{otherwise} \end{cases}$$

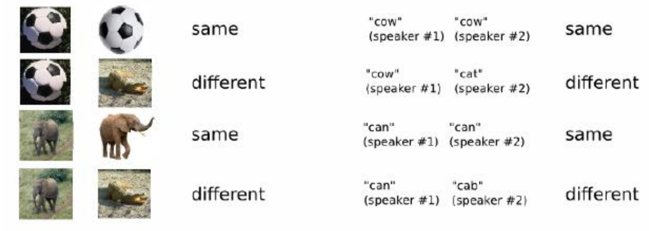
Applications

- Signature recognition
- Image recognition, search
- Article, Title
- Filter out typos
- Recommandation systems, collaborative filtering

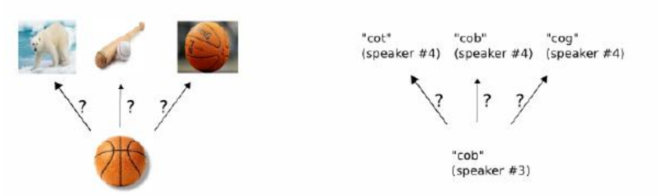
Siamese Networks for one-shot image recognition

Koch, Zemel, Salakhutdinov 15

Training similarity



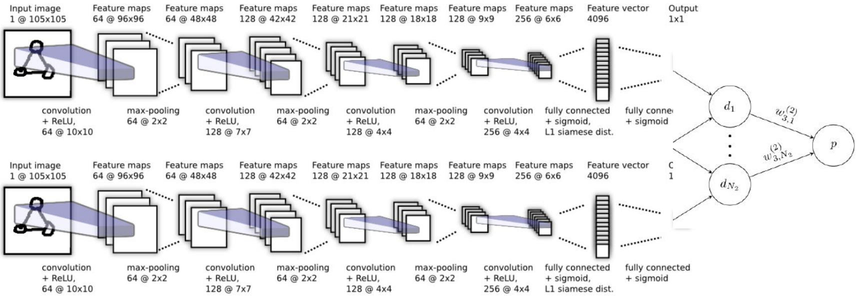
One-shot setting



Ingredients

Koch et al. 15

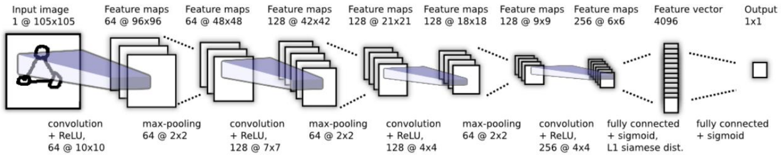
Architecture



Ingredients, 2

Koch et al. 15

Architecture



Distance

$$d(x, x') = \sigma\left(\sum_k \alpha_k |z_k(x) - z_k(x')|\right)$$

Loss

Given a batch $((x_i, x_i'), y_i)$ with $y_i = 1$ iff x_i and x_i' are similar

$$\mathcal{L}(w) = \sum_i y_i \log d(x_i, x_i') + (1 - y_i) \log (1 - d(x_i, x_i')) + \lambda \|w\|^2$$

Results

Koch et al. 15

Omniglot

Aurek-Besh
Futurama
Greek
Hebrew
Korean
Latin
Malay
Sanskrit

Results

Method	Test
Humans	95.5
Hierarchical Bayesian Program Learning	95.2
Affine model	81.8
Hierarchical Deep	65.2
Deep Boltzmann Machine	62.0
Simple Stroke	35.2
1-Nearest Neighbor	21.7
Siamese Neural Net	58.3
Convolutional Siamese Net	92.0

Siamese Networks

PROS

- Learn metrics, invariance operators
- Generalization beyond train data

CONS

- More computationally intensive
- More hyperparameters and fine-tuning, more training

Siamese Networks

Variational Auto-Encoders

Generative Adversarial Networks

Beyond AE

- A compressed (latent) representation
$$x \in \mathbb{R}^D \mapsto z = Enc(x) \in \mathbb{R}^d \mapsto Dec(z) \in \mathbb{R}^D \approx x$$
- Distance in latent space is meaningful $d(Enc(x), Enc(x'))$ reflects $d(x, x')$
- But $\forall z \in \mathbb{R}^d$: is $Dec(z) \in \mathbb{R}^D$ meaningful ?

Beyond AE

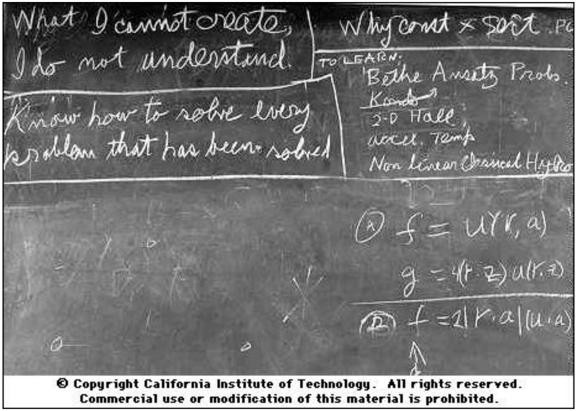
- A compressed (latent) representation

$$x \in \mathbb{R}^D \mapsto z = Enc(x) \in \mathbb{R}^d \mapsto Dec(z) \in \mathbb{R}^D \approx x$$

- Distance in latent space is meaningful $d(Enc(x), Enc(x'))$ reflects $d(x, x')$

- But $\forall z \in \mathbb{R}^d$: is $Dec(z) \in \mathbb{R}^D$ meaningful ?

"What I cannot create I do not understand" Feynman 88

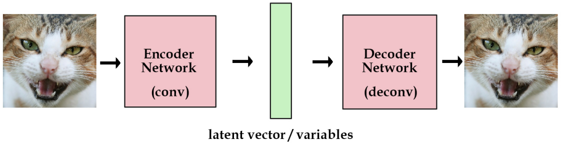

What I cannot create, I do not understand.
Why const × sort. PC
TO LEARN:
Bethe Ansatz Probs.
Kondo
2-D Hall
accel. Temp
Non Linear Classical Hydro
Know how to solve every problem that has been solved
ⓐ f = u(r, a)
g = (r·z) u(r, a)
ⓑ f = 2|r·a|(u·a)
© Copyright California Institute of Technology. All rights reserved.
Commercial use or modification of this material is prohibited.

Variational Auto-Encoders

Kingma Welling 14, Salimans Kingma Welling 15

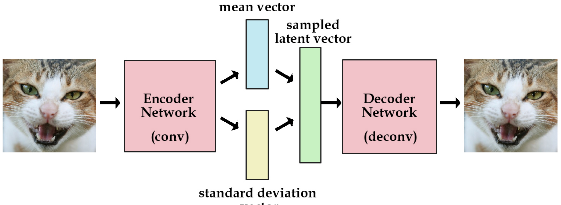
What we have:

- *Enc* a memorization of the data s.t. exists $Dec \approx Enc^{-1}$



What we want:

- $z \sim \mathcal{P}$ s.t. $Dec(z) \sim \mathcal{D}_{data}$



Distribution estimation

Data

$$\mathcal{E} = \{x_1, \ldots, x_n, x_i \in \mathcal{X}\}$$

Goal

- Find a probability distribution that models the data

$$p_\theta : \mathcal{X} \mapsto [0,1] \;\; \text{s.t. } \theta = \arg\max \prod_i p_\theta(x_i)$$

$\equiv$ maximize the log likelihood of data

$$\arg\max \prod_i p_\theta(x_i) = \arg\max \sum_i log(p_\theta(x_i))$$

Gaussian case

$$\theta = (\mu, \sigma) \qquad p_\theta(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

Akin Graphical models

Find hidden variables $\mathbf{z}$ s.t.

$$\mathbf{z} \mapsto \mathbf{x} \text{ s.t. } \quad \text{good} \quad p(\mathbf{x}|\mathbf{z})$$

Bayes relation

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z}|\mathbf{x}).p(\mathbf{x}) = p(\mathbf{x}|\mathbf{z}).p(\mathbf{z})$$

Hence

$$p(\mathbf{z}|\mathbf{x}) = p(\mathbf{x}|\mathbf{z}).p(\mathbf{z}) / \int p(\mathbf{x}|\mathbf{z}).p(\mathbf{z}) d\mathbf{z}$$

Problem:

denominator computationally intractable...

State of art

- Monte-Carlo estimation
- Variational Inference
  choose $\mathbf{z}$ well-behaved, and make $q(\mathbf{z})$ “close” to $p(\mathbf{z}|\mathbf{x})$.

Variational Inference

- Approximate $p(\mathbf{z}|\mathbf{x})$ by $q(\mathbf{z})$
- Minimize distance between both, using Kullback-Leibler divergence

Reminder

- information $(\mathbf{x}) = -log(p(\mathbf{x}))$
- entropy$(\mathbf{x}_1, \dots \mathbf{x}_k) = -\sum_i p(\mathbf{x}_i) log(p(\mathbf{x}_i))$
- Kullback-Leibler divergence between distribution q and p

$$KL(q||p) = \sum_x q(\mathbf{x}) log \frac{q(\mathbf{x})}{p(\mathbf{x})}$$

Beware: not symmetrical, hence not a distance; plus numerical issues when supports are different

Variational inference

$$\text{Minimize } KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

Evidence Lower Bound (ELBO)

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

Evidence Lower Bound (ELBO)

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

use $p(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}, \mathbf{x})/p(\mathbf{x})$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})p(\mathbf{x})}{p(\mathbf{z}, \mathbf{x})} d\mathbf{z}$$

Evidence Lower Bound (ELBO)

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

use $p(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}, \mathbf{x})/p(\mathbf{x})$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})p(\mathbf{x})}{p(\mathbf{z}, \mathbf{x})} d\mathbf{z}$$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}, \mathbf{x})} d\mathbf{z} + \int q(\mathbf{z}) log(p(\mathbf{x})) d\mathbf{z}$$

Evidence Lower Bound (ELBO)

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

use $p(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}, \mathbf{x})/p(\mathbf{x})$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})p(\mathbf{x})}{p(\mathbf{z}, \mathbf{x})} d\mathbf{z}$$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}, \mathbf{x})} d\mathbf{z} + \int q(\mathbf{z}) log(p(\mathbf{x})) d\mathbf{z}$$

as $\int q(\mathbf{z}) d\mathbf{z} = 1$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}, \mathbf{x})} d\mathbf{z} + log(p(\mathbf{x}))$$

Evidence Lower Bound (ELBO)

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

use $p(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}, \mathbf{x})/p(\mathbf{x})$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})p(\mathbf{x})}{p(\mathbf{z}, \mathbf{x})} d\mathbf{z}$$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}, \mathbf{x})} d\mathbf{z} + \int q(\mathbf{z}) log(p(\mathbf{x})) d\mathbf{z}$$

as $\int q(\mathbf{z}) d\mathbf{z} = 1$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}, \mathbf{x})} d\mathbf{z} + log(p(\mathbf{x}))$$

recover $KL(q(\mathbf{z})||p(\mathbf{z}, \mathbf{x}))$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = -\int q(\mathbf{z}) log \frac{p(\mathbf{z}, \mathbf{x})}{q(\mathbf{z})} d\mathbf{z} + log(p(\mathbf{x}))$$

Evidence Lower Bound, 2

Define

$$L(q(\mathbf{z})) = \int q(\mathbf{z}) log \frac{p(\mathbf{z}, \mathbf{x})}{q(\mathbf{z})} d\mathbf{z}$$

Last slide:

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = log(p(\mathbf{x})) - L(q(\mathbf{z})))$$

Hence

Minimize Kullback-Leibler divergence $\equiv$ Maximize $\mathbf{L(q(z))}$

Evidence Lower Bound, 3

More formula massaging

$$L(q(\mathbf{z})) = \int q(\mathbf{z}) log \frac{p(\mathbf{z}, \mathbf{x})}{q(\mathbf{z})} d\mathbf{z}$$

Evidence Lower Bound, 3

More formula massaging

$$L(q(\mathbf{z})) = \int q(\mathbf{z}) log \frac{p(\mathbf{z},\mathbf{x})}{q(\mathbf{z})} d\mathbf{z}$$

use $p(\mathbf{z},\mathbf{x}) = p(\mathbf{z}|\mathbf{x})p(\mathbf{x})$

$$L(q(\mathbf{z})) = \int q(\mathbf{z}) log \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{q(\mathbf{z})} d\mathbf{z}$$

$$L(q(\mathbf{z})) = \int q(\mathbf{z}) log \frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z})} d\mathbf{z} + \int q(\mathbf{z}) log(p(\mathbf{x})) d\mathbf{z}$$

$$L(q(\mathbf{z})) = -KL(q(\mathbf{z})||p(\mathbf{z}|x)) + \mathbb{E}_q[log(p(\mathbf{x}))]$$

Finally

$$\text{Maximize } \mathbb{E}_q[log(p(\mathbf{x})] - KL(q(\mathbf{z})||p(\mathbf{z}|x))$$

make $p(\mathbf{x})$ great under q **akin data fitting**
while minimizing the KL divergence between the two **akin regularization**

Where neural nets come in

Searching p and q

- We want $p(\mathbf{x}|\mathbf{z})$, we search $p(\mathbf{z}|x)$
- Let $p(\mathbf{z}|\mathbf{x})$ be defined as a neural net (encoder)
- We want it to be close to a well-behaved (**Gaussian**) distribution $q(\mathbf{z})$

$$\text{Minimize } KL(q(\mathbf{z})||p(\mathbf{z}|x))$$

- And from $\mathbf{z}$ we generate a distribution $p(\mathbf{x}|\mathbf{z})$ (defined as a neural net, "decoder")
- such that $p(\mathbf{x}|\mathbf{z})$ gives a high probability mass to our data (next slide)

$$\text{Maximize } \mathbb{E}_q[log(p(\mathbf{x}))]$$

Good news

All these criteria are differentiable ! can be used to train the neural net.

The loss of the variational decoder

Continuous case

- $\mathbf{x} \mapsto \mathbf{z}$; Gaussian case, $\mathbf{z} \sim p(\mathbf{z}|\mathbf{x})$
- Now $\mathbf{z}$ is given as input to the decoder, generates $\hat{\mathbf{x}}$ (deterministic)
- $p(\mathbf{x}|\hat{\mathbf{x}}) = F(\exp\{-\|\mathbf{x} - \hat{\mathbf{x}}\|^2\})$
- ... back to the L_2 loss

Binary case

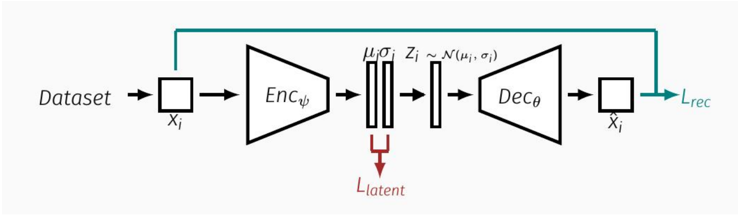
- Exercize: back to the cross-entropy loss

Variational auto-encoders

Kingma et al. 13

Position

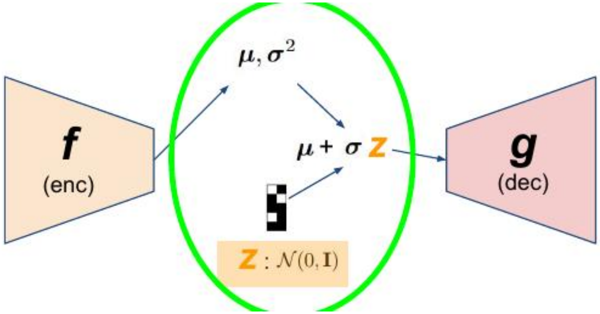
- Like an auto-encoder (data fitting term) with a regularizer, the KL divergence between the distribution of the hidden variables $\mathbf{z}$ and the target distribution.
- Say the hidden variable follows a Gaussian distribution: $\mathbf{z} \sim \mathcal{N}(\mu, \Sigma)$
- Therefore, the encoder must compute the parameters μ and Σ



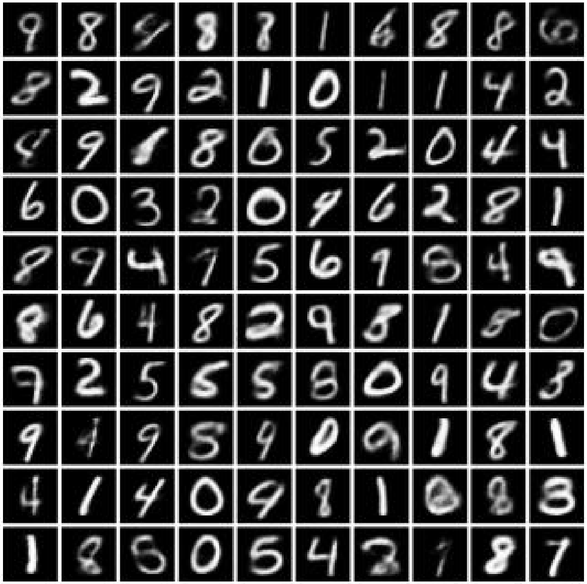
The reparameterization trick

Principle

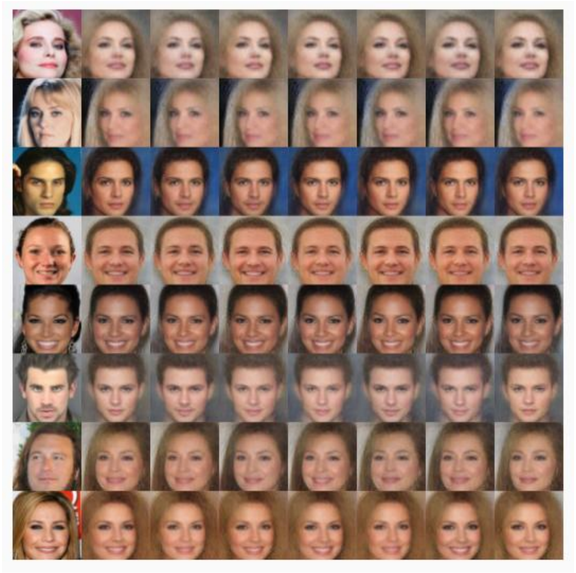
- Hidden layer: parameters of a distribution $\mathcal{N}(\mu, \sigma^2)$
- Distribution used to generate values $z = \mu + \sigma \times \mathcal{N}(0,1)$
- Enables backprop; reduces variances of gradient


$\boldsymbol{\mu}, \boldsymbol{\sigma}^2$
f
(enc)
$\boldsymbol{\mu} + \boldsymbol{\sigma}\,\mathbf{Z}$
g
(dec)
$\mathbf{Z} : \mathcal{N}(0, \mathbf{I})$

Examples



Examples



Also: https://www.youtube.com/watch?v=XNZIN7Jh3Sg

Discussion

PROS

- A trainable generative model

CONS

- The generative model has a Gaussian distribution at its core: blurry

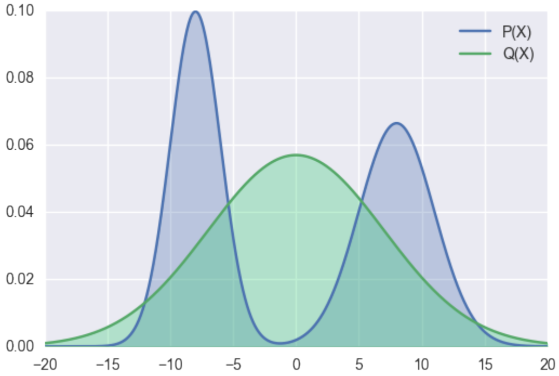
Discussion

PROS

- A trainable generative model

CONS

- The generative model has a Gaussian distribution at its core: blurry



Generative Adversarial Networks

Goodfellow et al., 14

Goal: Find a generative model

- Classical: learn a distribution — hard
- Idea: replace a distribution evaluation by a 2-sample test

Principle

- Find a good generative model, s.t. generated samples **cannot be discriminated** from real samples

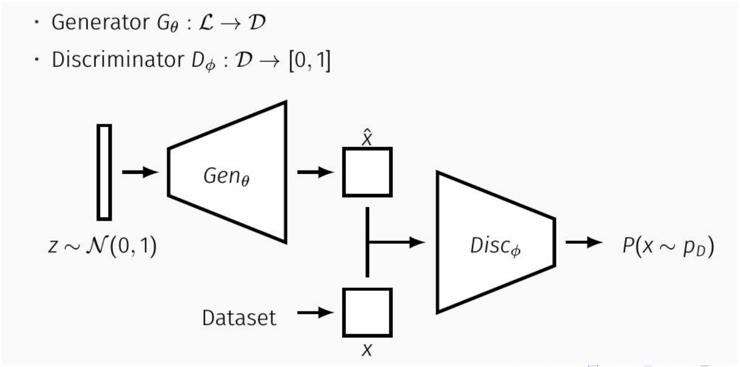
(not easy)

Principle

Goodfellow, 2017

Elements

- True samples **x** (**real**)
- A generator G (variational auto-encoder):
  generates from **x** (**reconstructed**) or from scratch (fake)
- A discriminator D: discriminates *fake* from others (**real** and **reconstructed**)


· Generator $G_\theta : \mathcal{L} \rightarrow \mathcal{D}$
· Discriminator $D_\phi : \mathcal{D} \rightarrow [0,1]$
Gen_θ
$\hat{x}$
$z \sim \mathcal{N}(0,1)$
$Disc_\phi$
$P(x \sim p_D)$
Dataset
x

Principle, 2

Goodfellow, 2017

Mechanism

- Alternate minimization
- Optimize D to tell fake from rest
- Optimize G to deceive D Turing test

$$Min_G \ Max_D \mathbb{E}_{x \in data}[\log(D(\mathbf{x}))] + \mathbb{E}_{z \sim p_x(z)}[\log(1 - D(z))]$$

Caveat

- The above loss has a vanishing gradient problem because of the terms in $\log(1 - D(z))$.
- We can replace it with $-\log((1 - D(z)/D(z))$, which has the same fixed point (the true distribution) but doesn't saturate.

GAN vs VAE

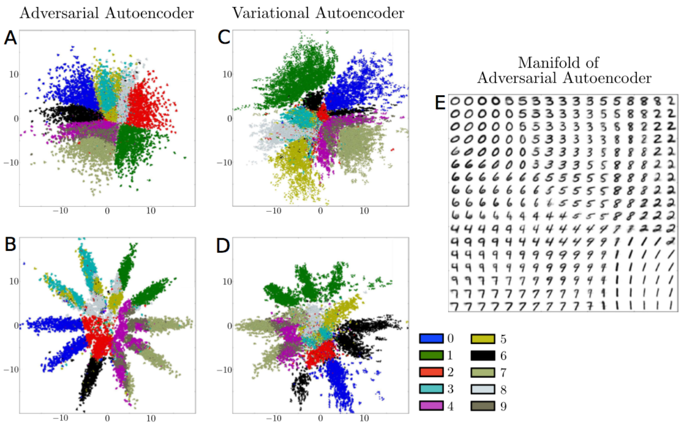


Figure 2: Comparison of adversarial and variational autoencoder on MNIST. The hidden code **z** of the *hold-out* images for an adversarial autoencoder fit to (a) a 2-D Gaussian and (b) a mixture of 10 2-D Gaussians. Each color represents the associated label. Same for variational autoencoder with (c) a 2-D gaussian and (d) a mixture of 10 2-D Gaussians. (e) Images generated by uniformly sampling the Gaussian percentiles along each hidden code dimension **z** in the 2-D Gaussian adversarial autoencoder.

Generative adversarial networks

Goodfellow, 2017

Generative adversarial networks, 2

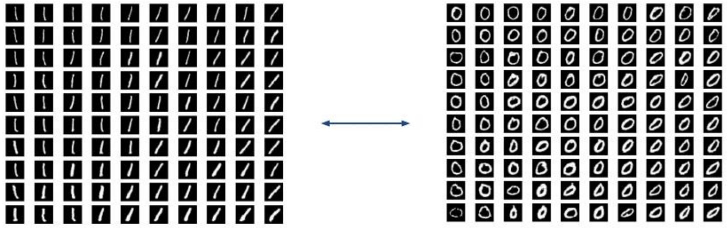
Goodfellow, 2017

Limitations

Training instable

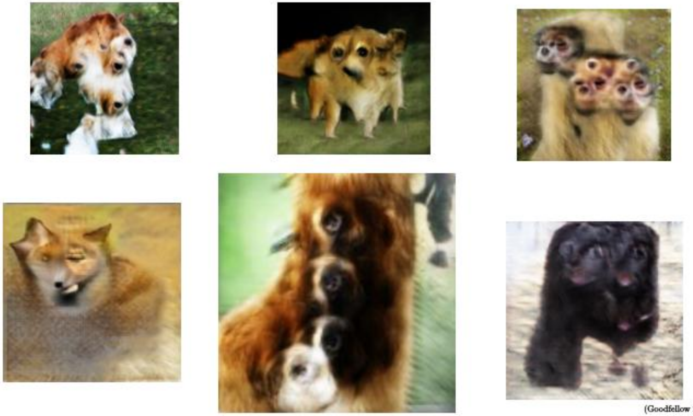
co-evolution of Generator / Discriminator

Mode collapse



Limitations, 2

Generating monsters


(Goodfellow 2016)

Towards Principled Methods for Training Generative Adversarial Networks

Arjovsky, Bottou 17

Why minimizing KL fails

$$\text{Minimizing } KL(P_{real}||P_{gen}) = \int P_{real} \log \frac{P_{real}}{P_{gen}}$$

- For P_{real} high and P_{gen} low (mode dropping), high cost
- For P_{real} low and P_{gen} high (gen. monsters), no cost

The GAN solution: minimizing

$$\mathbb{E}_{x \sim P_r}[\log D(x)] + \mathbb{E}_{x \sim P_g}[\log 1 - D(x)]$$

with

$$D^*(x) = \frac{P_r(x)}{P_r(x) + P_g(x)}$$

i.e., up to a constant, GAN minimizes

$$JS(P_{real}, P_{gen}) = \frac{1}{2}\left(KL(P_{real}||M) + KL(P_{gen}||M)\right)$$

with $M = \frac{1}{2}(P_{real} + P_{gen})$

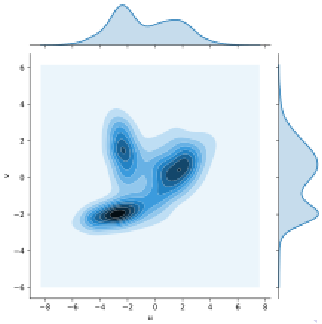
Towards Principled Methods for Training Generative Adversarial Networks, 2

Arjovsky, Bottou 17

Unfortunately
If P_r and P_g lie on non-aligned manifolds, exists a perfect discriminator; this is the end of optimization !

Proposed alternative: use Wasserstein distance

$$min_G\ max_D\ \mathbb{E}_{x\sim P_g}[D(x)] - \mathbb{E}_{x\sim P_r}[D(x)] = min_G\ W(P_r, P_g)$$



Does not solve all issues !

Pb of vanishing/exploding gradients in WGAN, addressed through weight clipping careful tuning needed

New Regularizations

Improved Training of Wasserstein GANs

Gulrajani, Ahmed, Arjovsky, Dumoulin, Courville 17

Stabilizing Training of Generative Adversarial Networks through Regularization

Roth, Lucchi, Nowozin, Hofmann, 17

Which Training Methods for GANs do actually Converge?

Mescheder, Geiger and Nowozin, 18

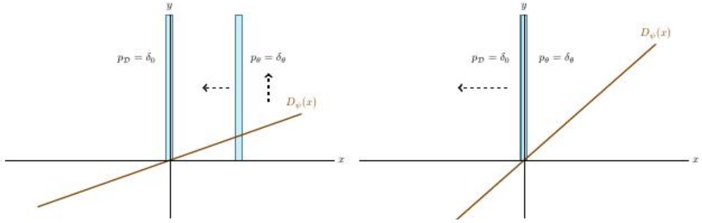
Simple experiments, simple theorems are the building blocks that help us understand more complicated systems. Ali Rahimi - Test of Time Award speech, NIPS 2017

Which Training Methods for GANs do actually Converge? 2

Mescheder, Geiger and Nowozin, 18

Toy example

$$P_r = \delta_0 \qquad P_g = \delta_\theta$$



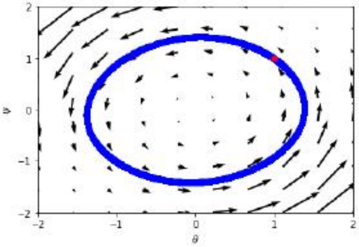
(a) $t = t_0$ (b) $t = t_1$

Figure 1. Visualization of the counterexample showing that in the general case, gradient descent GAN optimization is not convergent: (a) In the beginning, the discriminator pushes the generator towards the true data distribution and the discriminator's slope increases. (b) When the generator reaches the target distribution, the slope of the discriminator is largest, pushing the generator away from the target distribution. This results in oscillatory training dynamics that never converge.

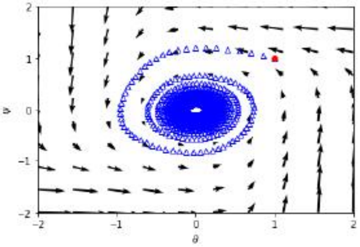
Which Training Methods for GANs do actually Converge? 2

Mescheder, Geiger and Nowozin, 18

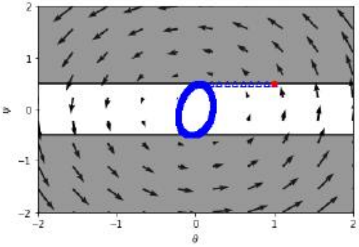
Lesson learned: cyclic behavior for GAN and WGAN



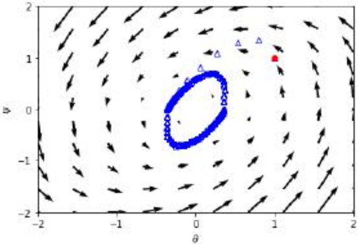
(a) Standard GAN



(b) Non-saturating GAN



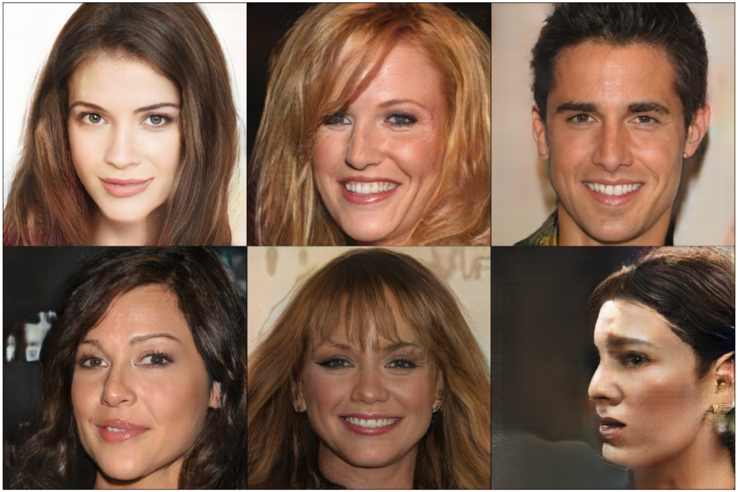
(c) WGAN ($n_d = 5$)



(d) WGAN-GP ($n_d = 5$)

State-of-the-art Generative Adversarial Networks

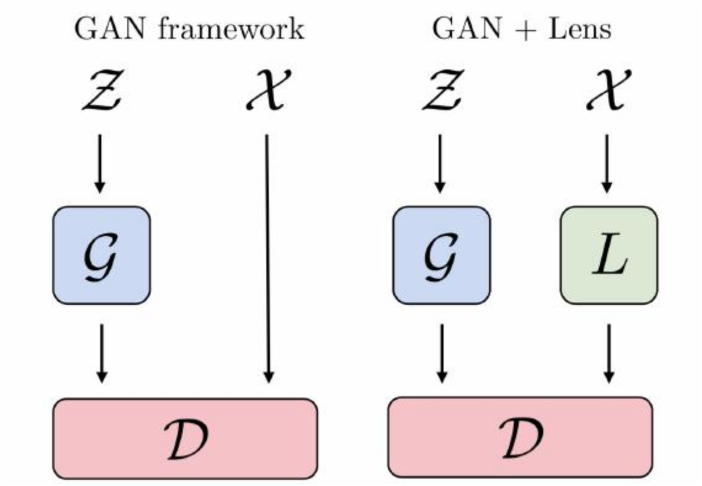
Mescheder, Geiger and Nowozin, 2018



Tempered Adversarial Networks

Sajjadi, Parascandolo, Mehrjou, Scholkopf 18

Principle: Life too easy for the discriminator !



Tempered Adversarial Networks

Sajjadi, Parascandolo, Mehrjou, Scholkopf 18

Principle: An adversary to the adversary

- $\Rightarrow$ Provide $L(X)$ instead, with L aimed at: i) deceiving the discriminator; ii) staying close from original images

$$min_G \, max_D \mathbb{E}_{x \sim P_r}[logD(x)] + \mathbb{E}_{x \sim P_g}[log(1D(x))]$$

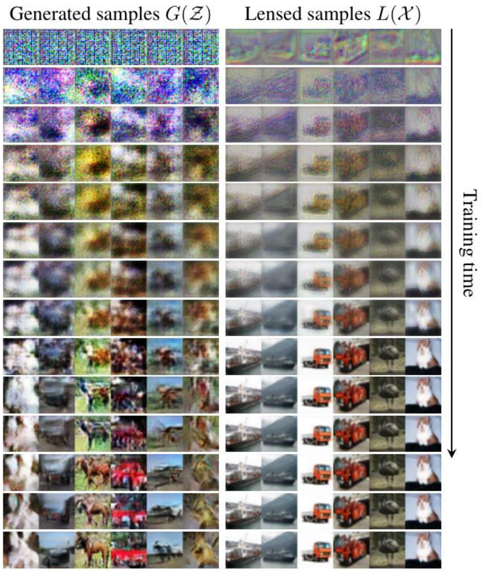
with D trained from $\{(L(x), 1)\} \cup \{G(z), 0\}$ and Lens L optimized

$$L^* = \arg\min -\lambda \mathcal{L}(D) + \sum_i \|L(x_i) - x_i\|^2$$

and $\lambda \to 0$.

Tempered Adversarial Networks, 2

Sajjadi, Parascandolo, Mehrjou, Scholkopf 18



Generated samples $G(\mathcal{Z})$
Lensed samples $L(\mathcal{X})$
Training time

Partial conclusions

- Deep revolution: Learning representations
- Adversarial revolution: a Turing test for machines
- Where is the limitation ?
  VAE: great but blurry
  GAN: great but mode dropping
  the loss function needs more work.

References (tbc)

see: github.com/artix41/awesome-transfer-learning

- Martin Arjovsky, Soumith Chintala, Léon Bottou, Wasserstein GAN, 2017
- Martin Arjovsky, Léon Bottou, Towards Principled Methods for Training Generative Adversarial Networks, 2017
- Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle: Greedy Layer-Wise Training of Deep Networks. NIPS 2006: 153-160
- Yoshua Bengio: Learning Deep Architectures for AI. Foundations and Trends in Machine Learning 2(1): 1-127 (2009)
- Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. ICLR, 2014.
- Gregory Koch, Richard Zemel, Ruslan Salakhutdinov: Siamese Neural Networks for One-shot Image Recognition, ICML 15
- Leon A. Gatys, Alexander S. Ecker, Matthias Bethge: A Neural Algorithm of Artistic Style. NIPS 2015
- Glorot, X., Bordes, A., and Bengio, Y. Domain adaptation for large-scale sentiment classification: A deep learning approach. ICML 11.

References (tbc)

- Ian J. Goodfellow, Yoshua Bengio, Aaron C. Courville: Deep Learning. Adaptive computation and machine learning, MIT Press 2016, ISBN 978-0-262-03561-3, pp. 1-775
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Generative Adversarial Networks, NIPS 2014
- Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville, Improved Training of Wasserstein GANs, 2017
- van der Maaten, L.J.P.; Hinton, G.E.: Visualizing Data Using t-SNE. Journal of Machine Learning Research. 9: 2579–2605. 2008
- Lars Mescheder, Andreas Geiger, Sebastian Nowozin, Which Training Methods for GANs do actually Converge?, ICML
- Portilla, J., Simoncelli, E. P., A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients. Int. J. Comput. Vis. 40, 49-70 (2000)
- Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, Thomas Hofmann, Stabilizing Training of Generative Adversarial Networks through Regularization, NIPS 2017

References (tbc)

- Mehdi S. M. Sajjadi, Giambattista Parascandolo, Arash Mehrjou, Bernhard Schölkopf: Tempered Adversarial Networks, ICML, 2018
- Tim Salimans, Diederik Kingma, and Max Welling. Markov chain Monte-Carlo and variational inference: Bridging the gap. ICML, 2015
- Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.A. Extracting and composing robust features with denoising autoencoders. ICML, 2008.