Réseaux de Neurones Profonds, Apprentissage de Représentations

Thierry Artières

ECM, LIF-AMU

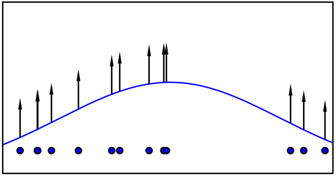
April 6, 2020

Outline

Generative models

Goal

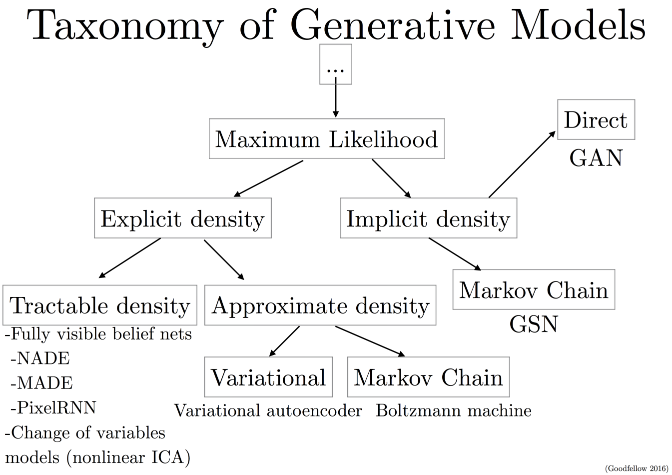
- Learn to generate complex and realistic data
- Statistical viewpoint : learn a model of the density of data / able to sample with this density
  - Postulate a parametric model : Usually not complex enough
  - Postulate a parametric form and perform optimization (e.g. Maximum Likelihood) : Intractable for complex forms $p(x) = \frac{F(x)}{Z(x}$ with $Z(x) = \sum_x F(x)$



$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{x \sim p_{\text{data}}} \log p_{\text{model}}(\boldsymbol{x} \mid \boldsymbol{\theta})$$

Maximum Likelihood Estimation (MLE)

Adversarial learning principle



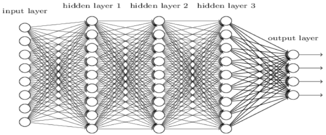
Adversarial learning principle

Principle

- Use a two player game
  - Learn both a generator of artificial samples AND a discriminator that learns to distinguishes between true and fake samples.
  - The generator wants to flue the discriminator
- If an equilibrium is reached the generator produces samples with the true density

Adversarial Learning: Generator

Determinitic NN as a generative model



Using a deterministic NN as a generative model

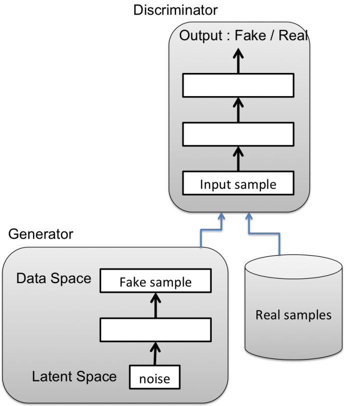
- Let note the function implemented by the model as G
- Let note the input $z \rightarrow$ The NN computes $G(z)$
- Assume z obeys a prior (noise) distribution, p_z, e.g. Gaussian distribution
- then the output x of the NN follows a distribution

$$\Rightarrow p_G(x) = \int_{z \text{ s.t. } G(z)=x} p_z(z)dz$$

Le principe de l'adversarial learning [Goodfellow and al., 2014]

Principle

- Jeu à deux joueurs: un générateur et un discriminateur
  - le discriminateur veut distinguer les exemples générés des vrais exemples
  - Le générateur veut tromper le discriminateur



Adversarial Learning criterion

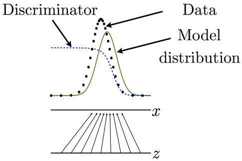
Criterion from [Goodfellow and al., 2014]

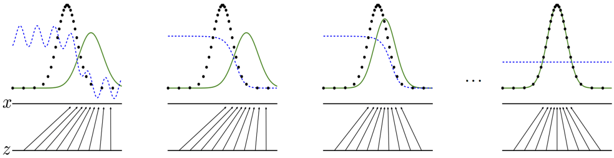
- Generator G and Discriminator D are two NNs
  - Whose parameters are noted θ_g and θ_d
- Distributions
  - p_{data} stands for the empirical distribution of the data from the training set
  - p_z is a prior noise distribution, e.g. a Gaussian distribution
  - On convergence we want $p_g = p_{data}$
- Learning criterion:

$$min_g max_d v(\theta_g, \theta_d) = \mathbf{E}_{x \sim p_{data}} [log D(x)] + \mathbf{E}_{z \sim p_z} [log(1 - D(G(z)))]$$

  - Assume G is fixed: D is trained to distinguish between fake and true samples
  - Assume D is fixed : G is trained to generate samples as realistic as possible

Adversarial Learning theory: What happens during Learning





Learning algorithm

Algo from [Goodfellow and al., 2014]

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**
  for k steps **do**
    • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    • Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Characterizing the solution

Optimal discriminator

- G being fixed

$$D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

- Let note $C(G) = max_D V(G, D)$

$$\Rightarrow C(G) = -log(4) + 2 \times JSD(p_{data}||p_g)$$

- with: $JSD(p_{data}||p_g)$ the Jensen-Shanon divergence

$$JSD(p_{data}||p_g) = KL(p_{data}||\frac{p_{data}(x)}{p_{data}(x) + p_g(x)}) + KL(p_g||\frac{p_{data}(x)}{p_{data}(x) + p_g(x)})$$

- with $JSD \geq 0$ and $JSD = 0 \rightarrow p_{data} = p_g$

Convergence proof

Convergence proof

Proposition 2. *If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D^*_G(\boldsymbol{x}))]$$

then p_g converges to p_{data}

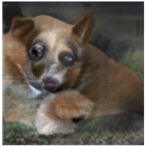
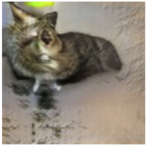
Proof. Consider $V(G, D) = U(p_g, D)$ as a function of p_g as done in the above criterion. Note that $U(p_g, D)$ is convex in p_g. The subderivatives of a supremum of convex functions include the derivative of the function at the point where the maximum is attained. In other words, if $f(x) = \sup_{\alpha\in\mathcal{A}} f_\alpha(x)$ and $f_\alpha(x)$ is convex in x for every α, then $\partial f_\beta(x) \in \partial f$ if $\beta = \arg\sup_{\alpha\in\mathcal{A}} f_\alpha(x)$. This is equivalent to computing a gradient descent update for p_g at the optimal D given the corresponding G. $\sup_D U(p_g, D)$ is convex in p_g with a unique global optima as proven in Thm 1, therefore with sufficiently small updates of p_g, p_g converges to p_x, concluding the proof. □

Good Examples

Bad examples





Interpolating with GANs [Goodfellow and al., 2014]

Idea

- The latent code space is fully occupied
- Any sample drawn by sampling with the generator should be realistic
- One may interpolate between two latent codes and see

Figure 3: Digits obtained by linearly interpolating between coordinates in z space of the full model.

Original GANs' features

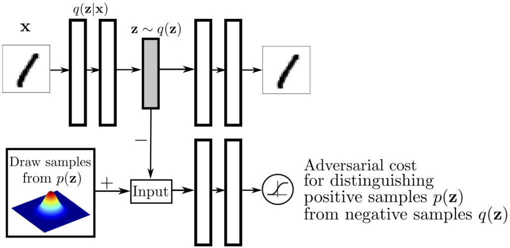
Known problems

- DIfficult learning
- Very long learning
- Missing modes
- Evaluation measures

Many many variants

- Conditional
- Disantangling
- Image editing

Adversarial AE [Makhzani and al., 2014 ou 15]

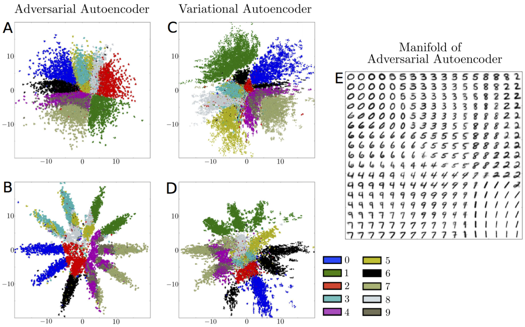


Learning criterion

- Few definitions for $q(z|x)$: simplest = deterministic
- Learning criterion:

$$min_g max_d v(\theta_g, \theta_d) = \mathbf{E}_{x \sim p_{data}} \left[\| D_c(E_c(x)) - x \|^2 \right] + \mathbf{E}_{z \sim p_z} [logD(z)] + \mathbf{E}_{x \sim p_{data}} [log(1 - D(q(z|x)))]$$

Adversarial AE [Makhzani and al., 2014 ou 15]



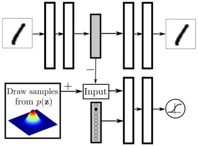
Investigating the hidden code space

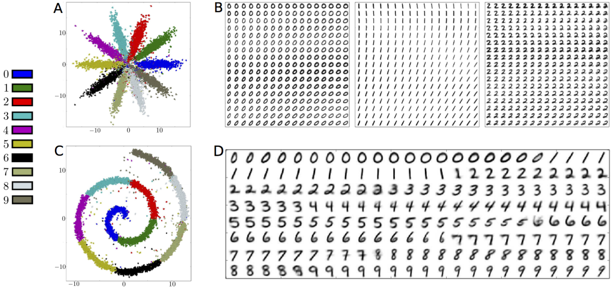
- Using different (2D) prior noise distributions with AAE and VAE (left)
- Sampling uniformly the Gaussian percentiles along each hidden code dimension z in the AAE (right)

Adversarial AE [Makhzani and al., 2014 ou 15]

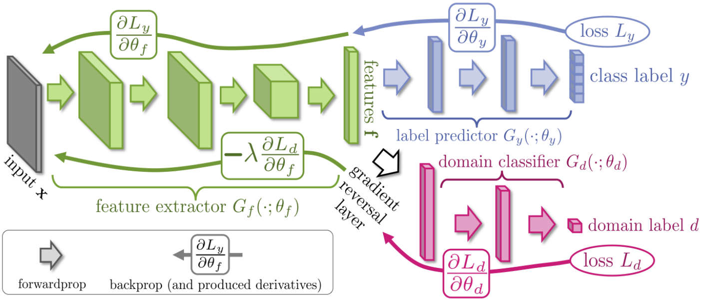
Label conditioned Variant

Goal: Better shape of the hidden code space





About using additional discriminators [Ganin et al, ICML 2015]

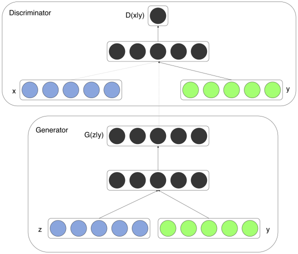


Conditional GANs [Mirza and al., 2014]

Learning criterion

- Citerion

$$min_g\, max_d\, v(\theta_g, \theta_d) = \mathbf{E}_{x,y\ \ p_{data}} [log D(x, y)] + \mathbf{E}_{z\ \ p_z, y'\ \ p_y} \left[log(1 - D(G(z, y'), y'))\right]$$



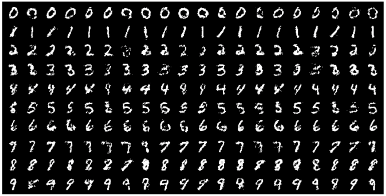
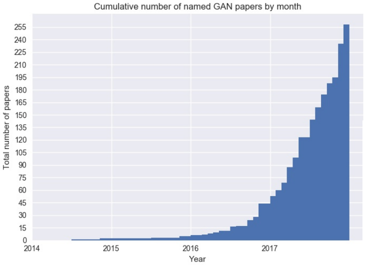
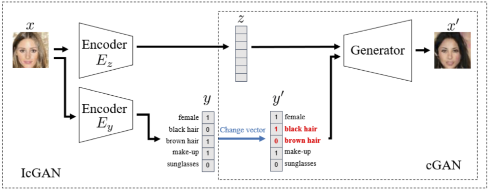
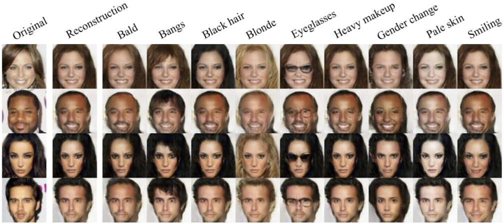


Figure 2: Generated MNIST digits, each row conditioned on one label

Image editing with Invertible Conditional GANs [Perarnau and al., 2016]

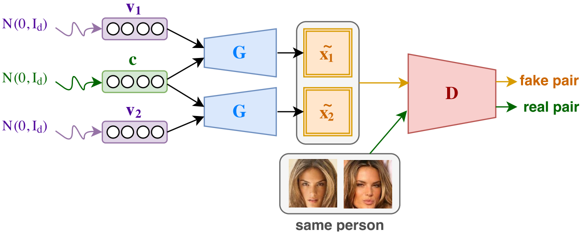


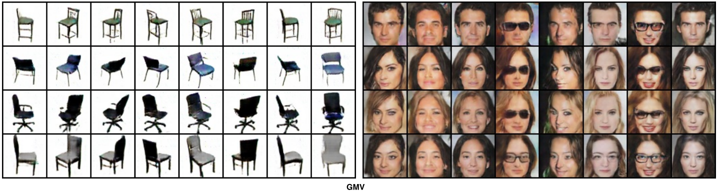




Disentangling factors of variation [Chen et al., 2018]

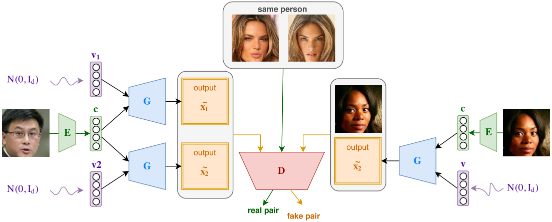
Generating images under various styles

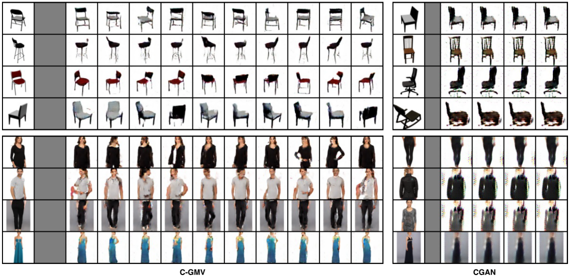




Disentangling factors of variation [Chen et al., 2018]

Transfering styles between images





Disentangling factors of variation [Qi et al., 2017]

Transfering styles between motion capture sequences