Réseaux de Neurones Profonds, Apprentissage de Représentations

Thierry Artières

ECM, LIF-AMU

February 15, 2018









Generative models and Adversarial Learning

- Generative models
- GANs
- Adversarial Autoencoders
- Examples

Outline



- Generative models
- GANs
- Adversarial Autoencoders
- Examples

Generative models

Generative models

Goal

- Learn to generate complex and realistic data
- $\bullet\,$ 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 $Z(x) = \sum_{x} F(x)$



Maximum Likelihood Estimation (MLE)

Generative models and Adversarial Learning ○● ○○○○○○○○○○ ○○○○ ○○○○

Generative models

Adversarial learning principle



GANs

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 $z \to$ 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

GANs

- 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:

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

- 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



GANs

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 $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_q(z)$.
- Sample minibatch of *m* examples $\{x^{(1)}, \ldots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
- · 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 {z⁽¹⁾,..., z^(m)} from noise prior p_g(z).
- · Update the generator by descending its stochastic gradient:

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

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

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

 \bullet with JSD \geq 0 and JSD = 0 \rightarrow $p_{\textit{data}} = p_{\textit{g}}$

Convergence proof

GANs

Convergence proof

Proposition 2. If G and D have enough capacity, and at each step of Algorithm I, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion $\mathbb{E}_{\boldsymbol{x} \sim p_{dota}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))]$

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.

Generative models and Adversarial Learning 00 000000000000 0000 0000 0000

GANs

Good Examples



T. Artières (ECM, LIF-AMU)

Deep Learning

Generative models and Adversarial Learning OO OOOOOOOOOOOO OOOO OOOO

GANs

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

1111335555777999911111

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

Generative models and Adversarial Learning

Adversarial Autoencoders

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



Learning criterion

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

$$\begin{split} \min_{g} \max_{d} \mathsf{v}(\theta_{g}, \theta_{d}) &= \mathsf{E}_{x \sim \rho_{data}} \left[\| D_{c}(E_{c}(x)) - x \|^{2} \right] + \mathsf{E}_{z \sim \rho_{z}} \left[\log D(z) \right] \\ &+ \mathsf{E}_{x \sim \rho_{data}} \left[\log (1 - D(q(z|x))) \right] \end{split}$$

Adversarial Autoencoders

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 Autoencoders

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

Label conditioned Variant

Goal: Better shape of the hidden code space







Adversarial Autoencoders

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



Conditional GANs [Mirza and al., 2014]

Learning criterion

Citerion

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



Examples

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



T. Artières (ECM, LIF-AMU)