

Learning to generate with adversarial networks

Gilles Louppe

June 27, 2016



Problem statement

- Assume training samples $\mathcal{D} = \{\mathbf{x} | \mathbf{x} \sim p_{\text{data}}, \mathbf{x} \in \mathcal{X}\}$;
- We want a generative model p_{model} that can draw new samples $\mathbf{x} \sim p_{\text{model}}$;
- Such that $p_{\text{model}} \approx p_{\text{data}}$.



$\mathbf{x} \sim p_{\text{data}}$



$\mathbf{x} \sim p_{\text{model}}$

Maximum likelihood approach

- Assume some form for p_{model} , as derived from knowledge and parameterized by θ ;
- Find the maximum likelihood estimator

$$\theta^* = \arg \max_{\theta} \sum_{\mathbf{x} \in \mathcal{D}} \log(p_{\text{model}}(\mathbf{x}; \theta));$$

- Draw samples from p_{θ^*} (e.g., with MCMC in case p_{model} is known only up to a constant factor).

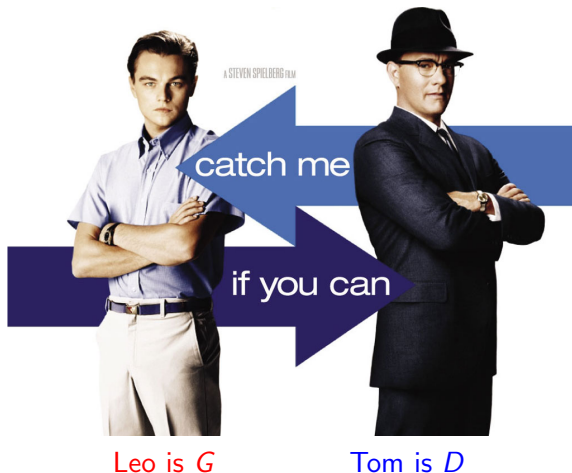
Modern alternatives: Variational Auto-Encoders (VAEs),
Generative Adversarial Networks (GANs)

Generative adversarial nets (Goodfellow et al., 2014)

Do not assume any form, but use a neural network to produce similar samples.

- Two-player game:
 - a discriminator D ,
 - a generator G ;
- D is a classifier $\mathcal{X} \mapsto \{0, 1\}$ that tries to distinguish between
 - a sample from the data distribution ($D(\mathbf{x}) = 1$, for $\mathbf{x} \sim p_{\text{data}}$),
 - and a sample from the model distribution ($D(G(\mathbf{z})) = 0$, for $\mathbf{z} \sim p_{\text{noise}}$);
- G is a generator $\mathcal{Z} \mapsto \mathcal{X}$ trained to produce samples $G(\mathbf{z})$ (for $\mathbf{z} \sim p_{\text{noise}}$) that are difficult for D to distinguish from data.

Catch me if you can



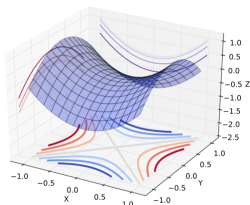
Objective

- Consider the value function

$$V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z} \sim p_{\text{noise}}} [\log(1 - D(G(\mathbf{z})))];$$

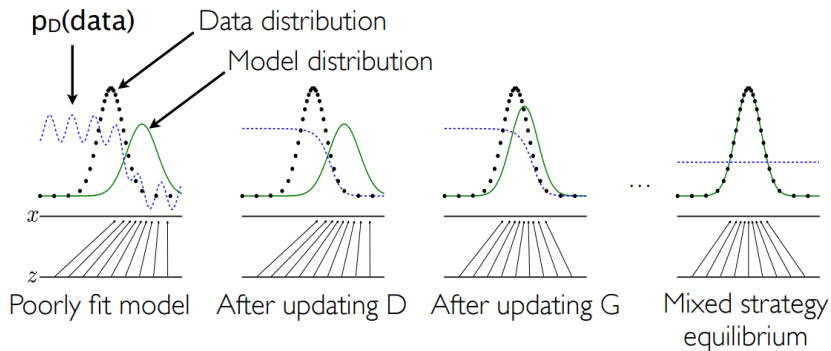
- We want to
 - For fixed G , find D which **maximizes** $V(D, G)$,
 - For fixed D , find G which **minimizes** $V(D, G)$;
- In other words, we are looking for the *saddle point*

$$(D^*, G^*) = \max_D \min_G V(D, G).$$



Learning process

Assuming D and G are neural networks parameterized by θ_D and θ_G , backpropagation can be used to optimize D 's and G 's objectives *alternatively* until convergence.



Theoretical guarantees

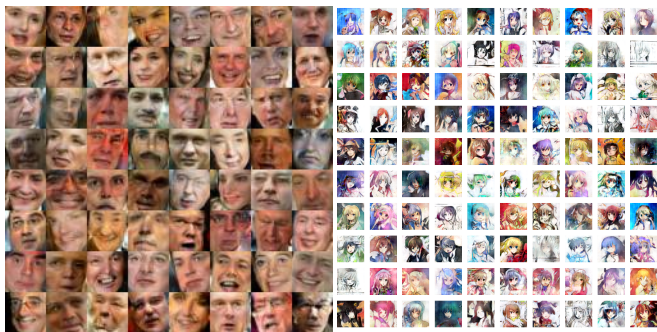
- Unique global optimum ;
- At the optimum, $\rho_{\text{model}} = \rho_{\text{data}}$;
- Convergence guaranteed.

(assuming infinite data and enough model capacity)

Conditional generative adversarial nets (Mirza and Osindero, 2014)

- The GAN framework can be extended to learn a parameterized generator $p_{\text{model}}(\mathbf{x}|\theta)$;
 - D is trained on (\mathbf{x}, θ) pairs,
 - G gets (\mathbf{z}, θ) as inputs;
- Useful to obtain a *single* generator object for all θ configurations;
- Can be used to interpolate between distributions.

In practice



None of these are real pictures!

Software

- Generative adversarial nets are **good old neural nets**;
- Therefore, the **deep learning software stack** can be leveraged
 - E.g. TensorFlow
 - Python and **C++ compatible**,
 - Compatible with single/multi/distributed CPUs/GPUs,
 - Active and strong community, backed by Google and others;
- Disentangle training from predictions for easier integration.
 - E.g. Using **lwttn** to integrate a trained NN into any C++ framework, with minimal dependencies.

Generative models for simulation?

- For **fast approximations** of otherwise **heavy computations**;
 - for either small or longer steps in the simulation pipeline (e.g. a generator for hits in a calorimeter, across one or several layers)
- For unknown processes for which we only have data
 - e.g. some old equipment for which no simulator was ever written;
- For interpolating between parameterized distributions.

Summary

- GANs can be used to learn a generator, from data only;
- GANs come with theoretical guarantees;
- GANs can be used to learn to sample from conditional distributions;

References

- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.
- Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.