Learning to generate
with adversarial networks

Gilles Louppe

June 27, 2016
Problem statement

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

$x \sim p_{\text{data}}$  \hspace{1cm}  $\rightarrow$  \hspace{1cm}  $x \sim p_{\text{model}}$
Maximum likelihood approach

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

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

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

Modern alternatives: Variational Auto-Encoders (VAEs), Generative Adversarial Networks (GANs)
Catch me if you can

Leo forges fake bank notes
Tom tries to detect them
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 generator $G$;
  - a discriminator $D$,

- $G$ is a generator $\mathcal{Z} \mapsto \mathcal{X}$ trained to produce samples $G(z)$ (for $z \sim p_{\text{noise}}$) that are difficult for $D$ to distinguish from data.

- $D$ is a classifier $\mathcal{X} \mapsto \{0, 1\}$ that tries to distinguish between
  - a sample from the data distribution ($D(x) = 1$, for $x \sim p_{\text{data}}$),
  - and a sample from the model distribution ($D(G(z)) = 0$, for $z \sim p_{\text{noise}}$);
Objective

• Consider the value function

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

• We want to
  - Find \( D \) which maximizes \( V(D, G) \),
  - 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 $\theta_D$ and $\theta_G$, backpropagation can be used to optimize $D$'s and $G$'s objectives \textit{alternatively} until convergence.
Theoretical guarantees

- Unique global optimum;
- At the optimum, $p_{\text{model}} = p_{\text{data}}$;
- 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 \( p_{\text{model}}(x|\theta) \);
  - \( D \) is trained on \((x, \theta)\) pairs,
  - \( G \) gets \((z, \theta)\) as inputs;
- Useful to obtain a *single* generator object for all \( \theta \) 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 lwtnn 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.
Proof of concept: generating shower shape data

\[ \mathcal{D} = \{ x | x = (\text{radius}, \text{angle}, \log(\text{energy})), x \in \mathbb{R}^+ \times [0; 2\pi] \times \mathbb{R} \} \]

\[ x \sim \rho_{\text{data}} \quad \text{and} \quad x \sim \rho_{\text{model}} \]
Proof of concept: generating shower shape data

$p(radius)$

$p(alpha)$

$p(log(energy))$
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;