Generative Adversarial Networks

Matt Olson

December 1, 2017

GAN: Overview

- High dimensional density estimation (i.e. unsupervised learning)
- **Generative**: goal is to create *new* samples that resemble training data (see applications)
- **Adversarial**: training proceeds by a "counterfeiter" pitted against discriminator (more on this later)
- **Network**: the "counterfeiter" and discriminator are neural networks
- Yann Lecun "(GANs are) the most interesting idea in the last 10 years in ML"

Application I: Sampling Images

Application II: Fashion Generation

Application III: Image Completion

Parametric Density Estimation: MLE

Classical MLE:

$$\max_\theta \prod_{i=1}^{n} p_\theta(x_i) \Leftrightarrow \min_\theta \sum_{i=1}^{n} \log \frac{1}{p_\theta(x_i)}$$

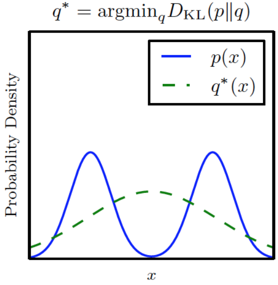
$$\Leftrightarrow min_\theta \int \log \left(\frac{\frac{1}{n}\sum_{i=1}^{n} \delta(x - x_i)}{p_\theta(x)} \right) \frac{1}{n} \sum_{i=1}^{n} \delta(x - x_i)$$

$$= KL(p_{data} || p_\theta)$$

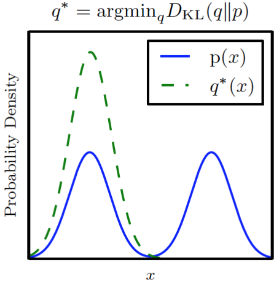
Maximum likelihood can be viewed as minimizing the KL divergence between a parametric family and the empirical distribution. One might consider other loss functions between these two distributions ...

Parametric Density Estimation: Other Approaches

Problem setting: generate data from mixture of Gaussians, fit a univariate Gaussian to the data. For sharp samples, may want to consider other loss functions besides negative log-likelihood ...



Maximum likelihood



Reverse KL

GAN: Shannon-Jensen Divergence

GAN optimization problem:

$$\min_{p_G} KL\left(p_G || \frac{p_G + p_{data}}{2}\right) + KL\left(p_{data} || \frac{p_G + p_{data}}{2}\right)$$

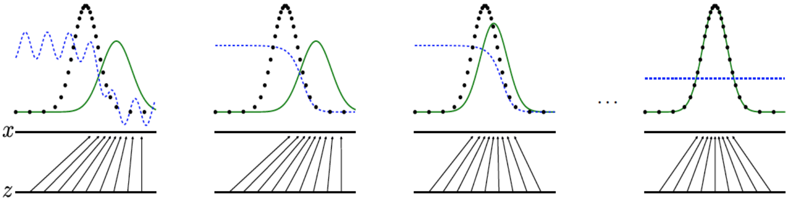
$$\Leftrightarrow$$

$$\min_{p_D} \max_{p_G} -\frac{1}{2}\mathbb{E}_{x \sim p_{data}} \log p_D(x) - \frac{1}{2}\mathbb{E}_{z \sim p_G} \log\left(1 - p_D(z)\right)$$

where p_D and p_G are probability distributions over the input space.

In words, p_G tries to generate data (images, etc.) that look like the training data, while p_D tries to distinguish generated data from training data.

GAN: In Pictures



(a) (b) (c) ... (d)

GAN: Training Procedure

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 $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.

  • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ 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 $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.

  • Update the generator by descending its stochastic gradient:

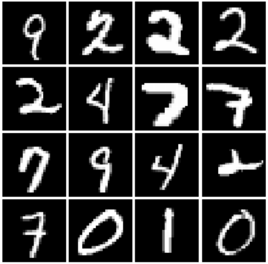
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

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

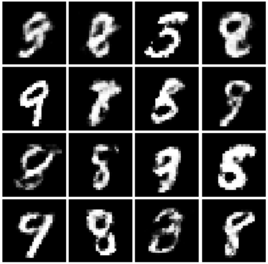
MNIST: Image Generation

- Sampling digits from MNIST data set
- GAN samples: see `gan.py`
- VAE samples: see `Week5/vae.py`



(a) Sample Digits

(b) VAE Samples



(c) GAN Samples