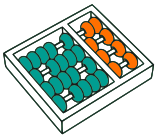


Generative Adversarial Networks

Lesson No.8 L5c

Ivan Lima

1 Introduction

One common aspect among Autoregressive models (MADE, PixelRNN/CNN, Gated PixelCNN, PixelSNAIL), Flow models (Autoregressive Flows, NICE, RealNVP, Glow, Flow++) and, Latent Variable Models (VAE, IWAE, VQ-VAE, VLAE, PixelVAE) is that they all are Likelihood-based models. They follow the sequence: sampling, evaluation the likelihood, training and representing in the latent space.Those type of model are called explicit because they explicitly represent the density estimation.

By doing differently we can focus our attention in sampling. We build a generative model that can understand the distribution behind the data points and interpolate along the training samples and output samples similar to the training data samples (similar, though not the same). Those models are called implicit because we implicitly represent the density estimation.

In Implicit Models, figure 1, we sample z from a fixed noise source distribution (uniform or gaussian), pass the noise through a deep neural network to obtain a sample x. The deep neural network $without$ explicit density estimation. There is no explicit form for p_{data} or p_{model}. It can only draw samples. Given samples from data distribution $p_{\text{data}} : x_1, x_2, \cdots, x_n$. And a given a sampler $q_\phi(z) = \text{DNN}(z;\phi)$ where $z \sim p(z)$, then $x = q_\phi(z)$ induces a density function p_{model}.

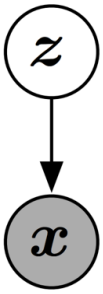


Figure 1

The idea in the implicit approach is to learn an appropriate ϕ to make p_{model} as close as possible to p_{data}. In order to do that, we need some measure of how far apart p_{data} and induced p_{model} are.

This idea diverges from the density models (explicit models) concept. In explicit models we use $KL\left(p_{\text{data}} \| p_{\text{model}}\right)$ which gave us the objective $\mathbb{E}_{x\sim p_{\text{data}}}\left[\log p_\theta(x)\right]$ (discarding the term independent of θ) where we explicitly modeled p_{model} as $p_\theta(x)$.By minimizing KL we maximize the likelihood.

As we do not have an explicit $p_\theta(x)$ in implicit models, it requires us to come up with distance measures. Those measures can potentially behave differently from maximum likelihood. Maximum Mean Discrepancy (MMD); Jensen Shannon Divergence (JSD) ,and Earth Mover's Distance are some examples that have been using for this purpose.
GAN [1], Generative Adversarial Networks, is considered a landmark in the implicit models approach. It was published in 2014. The main author is Ian J. Goodfellow. Also Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio were members of the same research group - Computational Science Department of Montreal's University.

2 Generative Adversarial Networks

2.1 Origin of GAN

The paper GAN proposed revolutionary framework at the time. The idea was estimating generative models via an adversarial process, that simultaneously trains two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples.

Until GAN came out, deep learning with discriminative models dominated the scenario. Deep generative models have had been less popular back then. It has changed since GAN came up. The paper explored the special case when the generative model generates samples by passing random noise through a multilayer perceptron, and the discriminative model is also a multilayer perceptron. They called this adversarial nets. In this case, they train both models using only the highly successful backpropagation and dropout algorithms and sample from the generative model using only forward propagation.

2.2 Math

The adversarial modeling framework idea is summarized in equation 1

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_x(z)}[\log(1-D(G(z)))] \tag{1}$$

We have the generative process G and a discriminator D. We try to minmax the game they are playing to each other. Notice that G appears only in the term $G(z)$ and it is our x (x could be our image space, text space, speech space, everything we are trying to generate), so $G(z)$ generates. Meanwhile the discriminator is trying to classify (discriminate) which of x are real versus generated. $\mathbb{E}_{x\sim p_{\text{data}}}$ is our real data distribution, so we try to maximize the log probability under the discriminator which can have 0 or 1 sigmoid output. Discriminate will try to drive the output to 1 if it is real. Simultaneously, $\mathbb{E}_{z\sim p_x(z)}$ does the opposite. It is trying to drive the discriminator to 0 because it is not real. In other words, the discriminator is trying to assign 1 to real and 0 to fakes as accurate as possible. If we reach the Nash Equilibrium of this game, none of them can improve anymore, the generator should generate things that looks real. The discriminator at that point outputs 50/50. The game is defined to try to generate things as realistic as possible.

Figure 2 illustrates the process. We have two pipelines. In the pipeline on the left we have X coming from the data, pass to D. It tries to assign the probability of 1 of data being real. It is a supervised learning problem. On the right pipeline at the same time there is a not supervised learning going on that try to assign 0 to the probability of artificial generated samples being real. The algorithm is as follows:

3 Evaluation

Evaluation for GANs is still an open problem. Unlikely density models, you cannot report explicit likelihood estimates on test sets.

3.1 Parzen-Window density estimator

If we can generate samples, we can consider them as a mixture of Gaussian where around every sample we put a Gaussian blob that yields a density model. Now we can evaluate for some test data what is the likelihood under this mixture Gaussian kernel density estimator model and report the numbers. The Parzen-Window density estimator, also known as Kernel Density Estimator (KDE) is an attempt to do that. It is a non-parametric way to estimate the probability density function of a random variable with kernel K and bandwidth h:

$$\hat{p}_h(x) = \frac{1}{nh}\sum_i K\left(\frac{x-x_i}{h}\right) \tag{2}$$

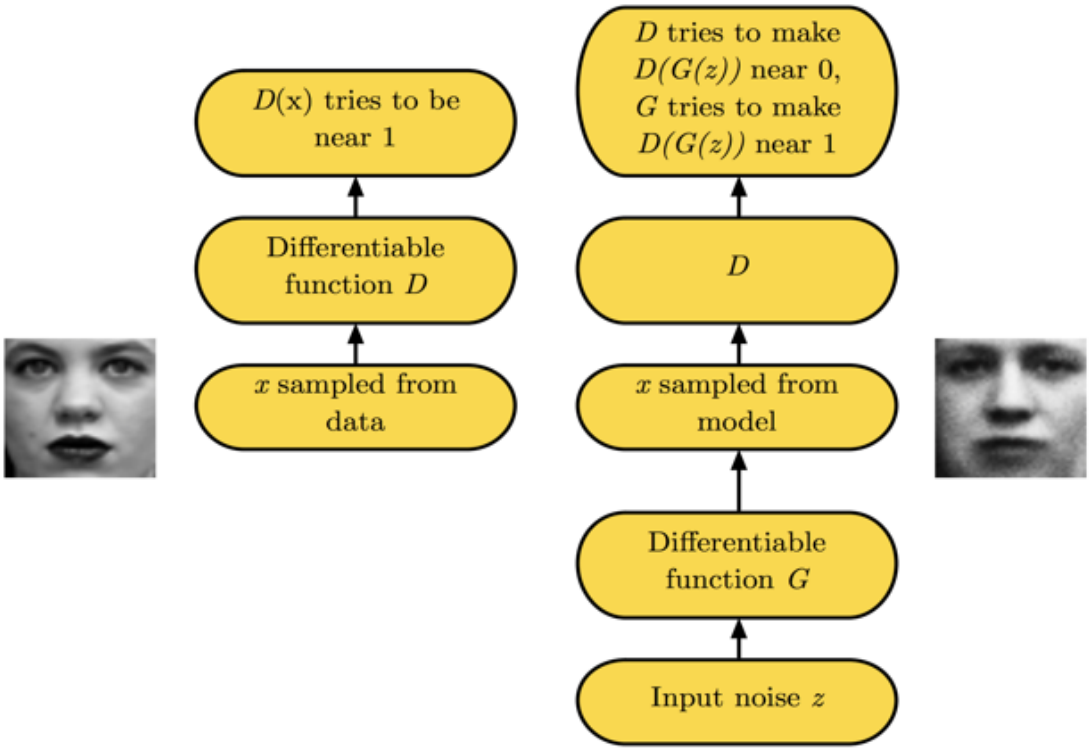


Figure 2. Figure from NeurIPS 2016 GAN Tutorial (Goodfellow)

In generative model evaluation, K is usually density function of standard Normal distribution The bandwidth h is relevant and is chosen according to the validation set. Figure 3 show the effect on kernel density of 3 different bandwidths

In principle we can use it, but it is just not a great metric. As shown in Figure 4, Parzen Window estimator can be unreliable. It is just a 6x6 dimension (36). For megapixel image it would be even worse.

3.2 Inception Score

A currently more popular density estimator is the inception score. The idea behind is that good generators generate samples that are semantically diverse. Semantics predictor could be a trained Inception Network v3. We have the probability of y given x, $p(y \mid x)$, where y is one of the 1000 ImageNet classes. Each image x should have distinctly recognizable object, so $p(\mathrm{y} \mid \mathrm{x})$ should have low entropy. There should be as many classes generated as possible, so p(y) should have high entropy.

Inception model is represented by $p(y \mid x)$ and the marginal label distribution by $p(y) = \int_x p(y \mid x) p_g(x)$. The higher is the inception score, more realistic the generated samples become.

$$\begin{aligned}\mathrm{IS}(x) &= \exp\left(\mathbb{E}_{x \sim p_g}\left[D_{\mathrm{KL}}[p(y \mid x) \| p(y)]\right]\right) \\ &= \exp\left(\mathbb{E}_{x \sim p_g, y \sim p(y|x)}[\log p(y \mid x) - \log p(y)]\right) \\ &= \exp(H(y) - H(y \mid x))\end{aligned}$$

3.3 Frechet Inception Distance

Another popular density measure is known as Frechet Inception Distance. Inception Score does not sufficiently measure diversity: a list of 1000 images (one of each class) can obtain perfect Inception Score. FID was proposed to capture more nuances. The idea is to to embed image x into some feature space (2048-dimensional activations

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.

of the Inception-v3 pool3 layer), then compare mean (m) and covariance (C) of those random features:

$$d^2\left((\boldsymbol{m}, \boldsymbol{C}), (\boldsymbol{m}_w, \boldsymbol{C}_w)\right) = \|\boldsymbol{m} - \boldsymbol{m}_w\|_2^2 + \operatorname{Tr}\left(\boldsymbol{C} + \boldsymbol{C}_w - 2\left(\boldsymbol{C}\boldsymbol{C}_w\right)^{1/2}\right) \tag{3}$$

4 GAN's Theory

GAN theory is not completely understood.

4.1 GAN: Bayes-Optimal Discriminator

What's the optimal discriminator given generated and true distributions? $V(G,D)$ is objective that we are looking for our discriminator. Using the integrals and changing the variable to write both integrals with respect to x we have the following expressions:

$$\begin{aligned} V(G,D) &= \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p(z)}[\log(1 - D(G(z)))] \\ &= \int_x p_{\text{data}}(x) \log D(x) dx + \int_z p(z) \log(1 - D(G(z))) dz \\ &= \int_x p_{\text{data}}(x) \log D(x) dx + \int_x p_g(x) \log(1 - D(x)) dx \\ &= \int_x \left[p_{\text{data}}(x) \log D(x) + p_g(x) \log(1 - D(x)) \right] dx \end{aligned} \tag{4}$$

By calculation the derivative with respect to $D(x)$ and set it equal to zero we have:

$$\nabla_y [a \log y + b \log(1-y)] = 0 \Longrightarrow y^* = \frac{a}{a+b} \quad \forall \quad [a,b] \in \mathbb{R}^2 \backslash [0,0] \tag{5}$$

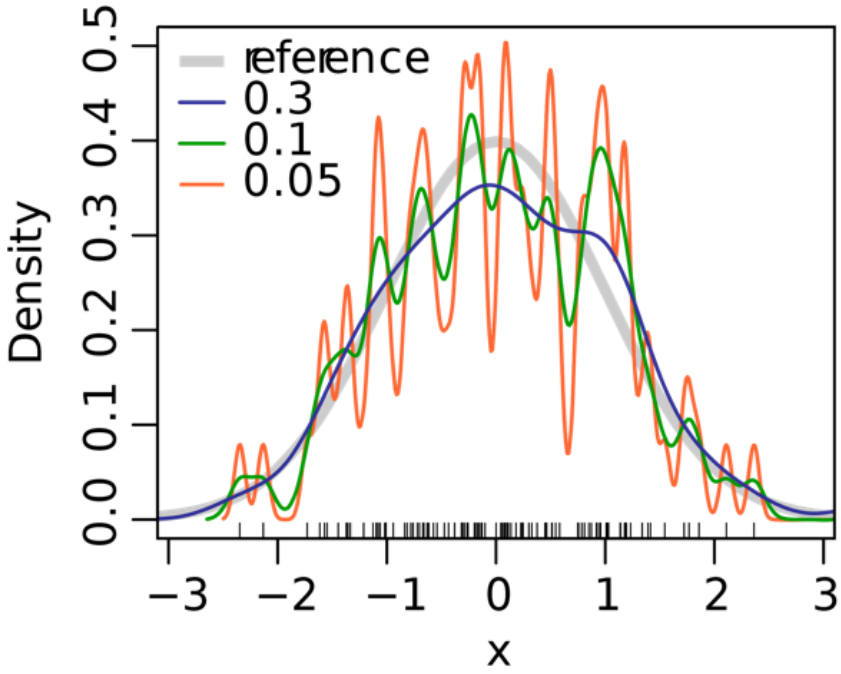


Figure 3. Kernel density estimation of 100 normally distributed random numbers using different smoothing bandwidths

So, the optimal discriminator will take the form of equation 6.

$$D^*(x) = \frac{p_{\text{data}}(x)}{\left(p_{\text{data}}(x) + p_g(x)\right)} \tag{6}$$

See Figure 5 for a illustration of the approach. Generative adversarial nets are trained by simultaneously updating the discriminative distribution D (blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) p_x from those of the generative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which z is sampled, in this case uniformly. The horizontal line above is part of the domain of $\boldsymbol{x}$. The upward arrows show how the mapping $\boldsymbol{x} = G(\boldsymbol{z})$ imposes the non-uniform distribution p_g on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g

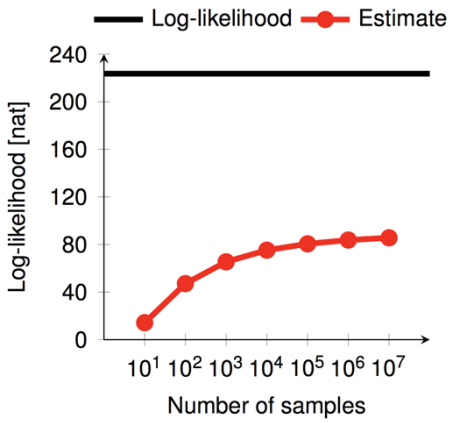
By inserting equation 6 in the original objective expression, we have:

$$\begin{aligned} V\left(G, D^*\right) &= \mathbb{E}_{x\sim p_{\text{data}}}\left[\log D^*(x)\right] + \mathbb{E}_{x\sim p_g}\left[\log\left(1 - D^*(x)\right)\right] \\ &= \mathbb{E}_{x\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}\right] + \mathbb{E}_{x\sim p_g}\left[\log \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)}\right] \\ &= -\log(4) + KL\left(p_{\text{data}} \,\|\, \left(\frac{p_{\text{data}} + p_g}{2}\right)\right) + KL\left(p_g \| \left(\frac{p_{\text{data}} + p_g}{2}\right)\right) \end{aligned} \tag{7}$$

where the KL terms are the Jensen-Shannon Divergence (JSD) of p_{data} and p_g, which is ≥ 0. Notice that $V(G^*, D^*) = -\log(4)$ when $p_g = p_{\text{data}}$

4.2 Saturation

In practice, equation 1 may not provide sufficient gradient for G to learn well. Early in learning, when G is poor, D can reject samples with high confidence because they are clearly different from the training data. In this case, $\log(1 - D(G(z)))$ saturates. Rather than training G to minimize $\log(1 - D(G(z)))$ we can train G to maximize $\log D(G(z))$. This objective function results in the same fixed point of the dynamics of G and D but provides much stronger gradients early in learning. Once the discriminator become very confident there will be no gradients left and you cannot improve your generator. The objective become flat and the derivative is zero. No sign from the discriminator how to improve the generator to get futures better samples. Two alternatives are commonly used



(a)

Model	Parzen est. [nat]
Stacked CAE	121
DBN	138
GMMN	147
Deep GSN	214
Diffusion	220
GAN	225
True distribution	**243**
GMMN + AE	282
k-means	313

(b)

Figure 4. A note on the evaluation of generative models (Theis, Van den Oord, Bethge 2015)

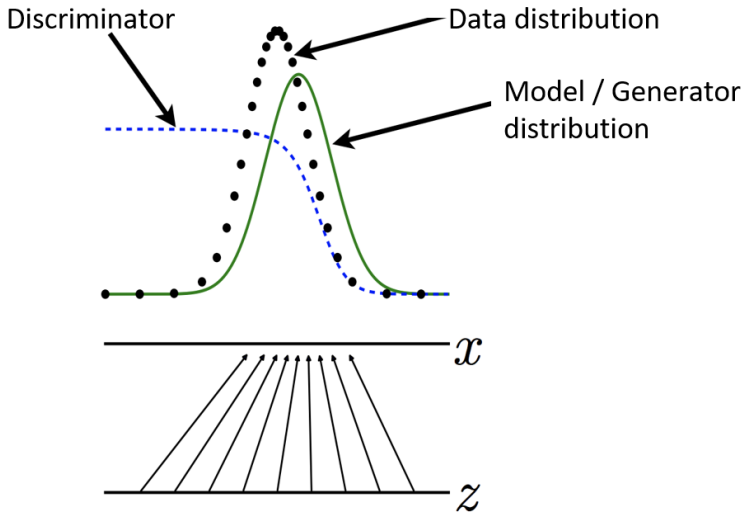


Figure 5. After an update to G, gradient of D has guided $G(z)$ to flow to regions that are more likely to be classified as data.

to avoid discrimination saturation. (1) do not optimize the gradient all the way to the end, and (2) reformulate the objective to be non-saturated.

5 Final remarks

Since the idea behind GAN was first published in 2014, a lot of progress in its architecture have been made. The essential of GAN is about the contest of two neural networks, What leads to the innovative concept of perceptual sampling.. There is no explicit representation of $p_g(x)$, and that D must be synchronized well with G during training (in particular, G must not be trained too much without updating D, in order to avoid G collapses too many values of z to the same value of x to have enough diversity to model p_{data}). Backpropagation is used to obtain gradients, no inference is needed during learning.

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial networks," *ArXiv*, vol. abs/1406.2661, 2014.