Generative Adversarial Networks (GAN)

강원대학교 IT대학
이창기

I got a lot of content and pictures from http://blog.aylien.com/introduction-generative-adversarial-networks-code-tensorflow/

Introduction

- Yann LeCun (AI research at Facebook)
  - *There are many interesting recent development in deep learning…*
  - *The most important one, in my opinion, is adversarial training (also called **GAN** for **Generative Adversarial Networks**).*
  - *This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion"*

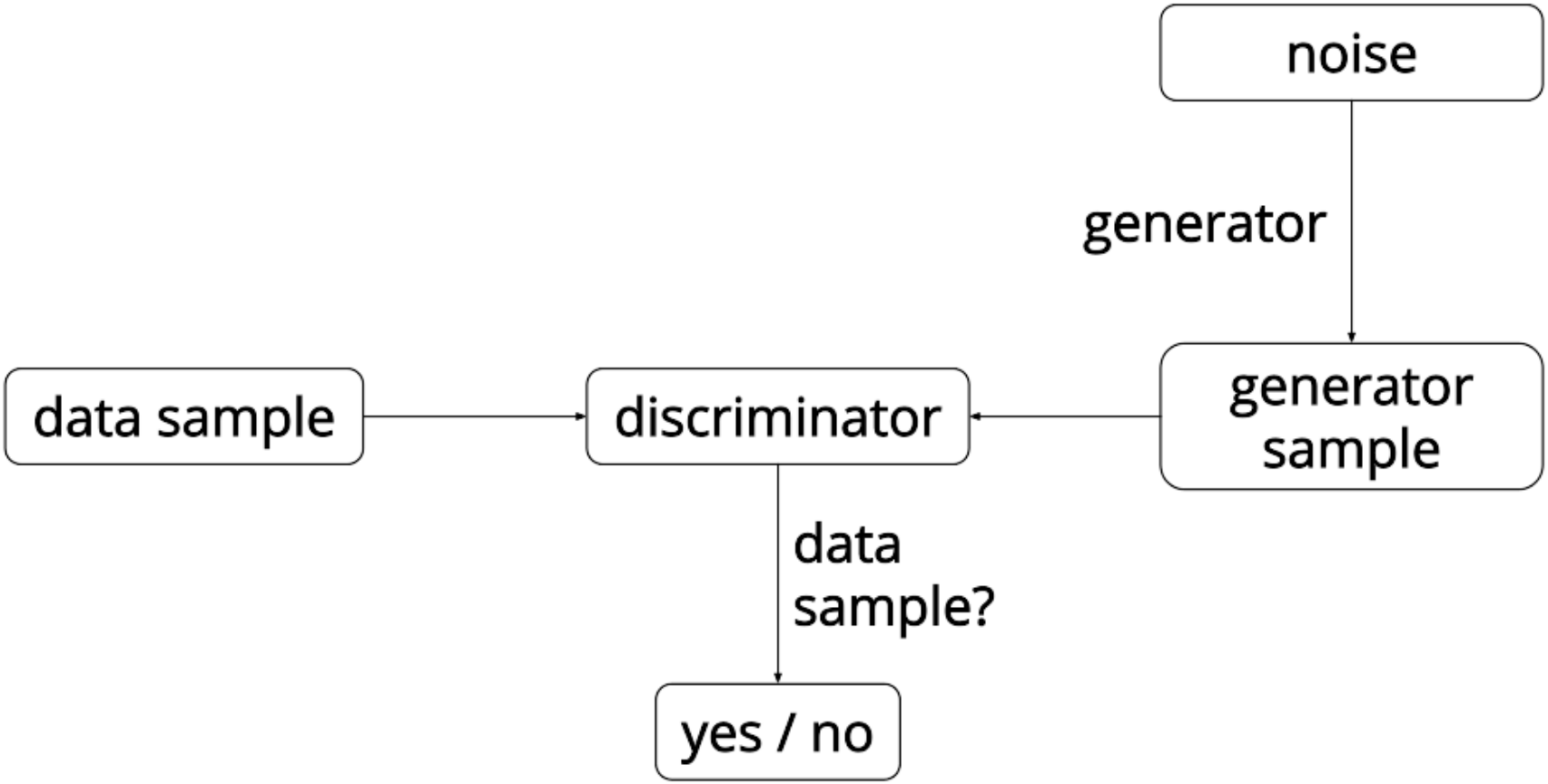
Discriminative vs. Generative models

- **Discriminative** models
  - Learn a function that maps the input data (*x*) to some desired output class label (*y*)
  - In probabilistic terms, they directly learn the conditional distribution *P(y|x)*
- **Generative** models
  - Try to learn the joint probability of the input data and labels simultaneously, i.e. *P(x,y)*
  - This can be converted to *P(y|x)* for classification via Bayes rule, but the generative ability could be used for something else as well, such as creating likely new *(x, y)* samples.
  - They have the potential to understand and explain the underlying structure of the input data even when there are no labels (i.e. unlabeled data)

Generative Adversarial Networks

- GAN has two competing neural network models
  - The **generator** takes noise as input and generates samples
  - The **discriminator** receives samples from both the generator and the training data, and has to be able to distinguish between the two sources
  - The generator is learning to produce more and more realistic samples, and the discriminator is learning to get better and better at distinguishing generated data from real data
  - These two networks are trained simultaneously, and the hope is that the competition will drive the generated samples to be indistinguishable from real data

GAN overview



Training

input noise variables $p_{\boldsymbol{z}}(\boldsymbol{z})$ represent a mapping to data space as $G(\boldsymbol{z};\theta_g)$
$D(\boldsymbol{x};\theta_d)$ that outputs a single scalar.

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

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.

Training – cont'd

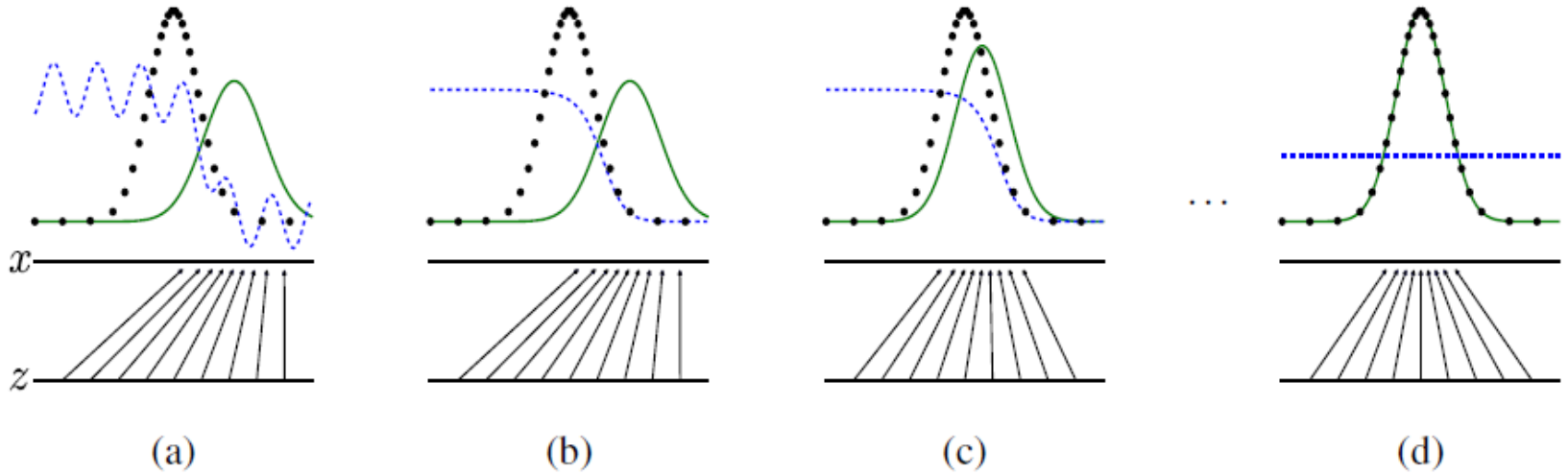


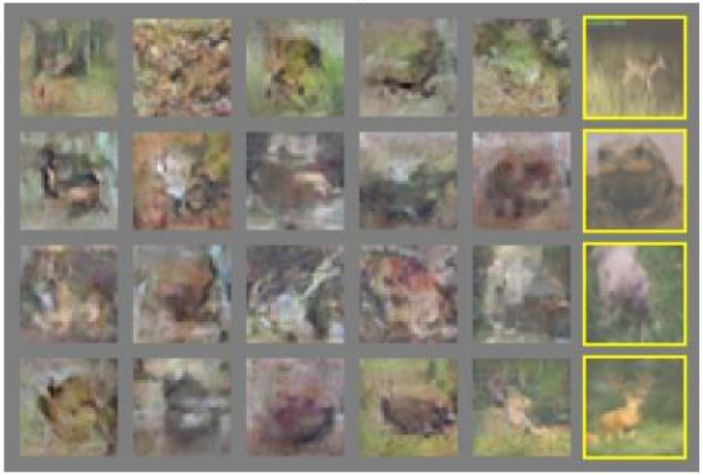
Figure 1: Generative adversarial nets are trained by simultaneously updating the **d**iscriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) p_x from those of the **g**enerative distribution p_g (G) (green, 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 x. The upward arrows show how the mapping $x = G(z)$ 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. (a) Consider an adversarial pair near convergence: p_g is similar to p_{data} and D is a partially accurate classifier. (b) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_g(x)}$. (c) 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. (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(x) = \frac{1}{2}$.

Example

a)

b)

c)



d)

a) MNIST b) TFD c) CIFAR-10 (fully connected model) d) CIFAR-10 (convolutional discriminator and "deconvolutional" generator)

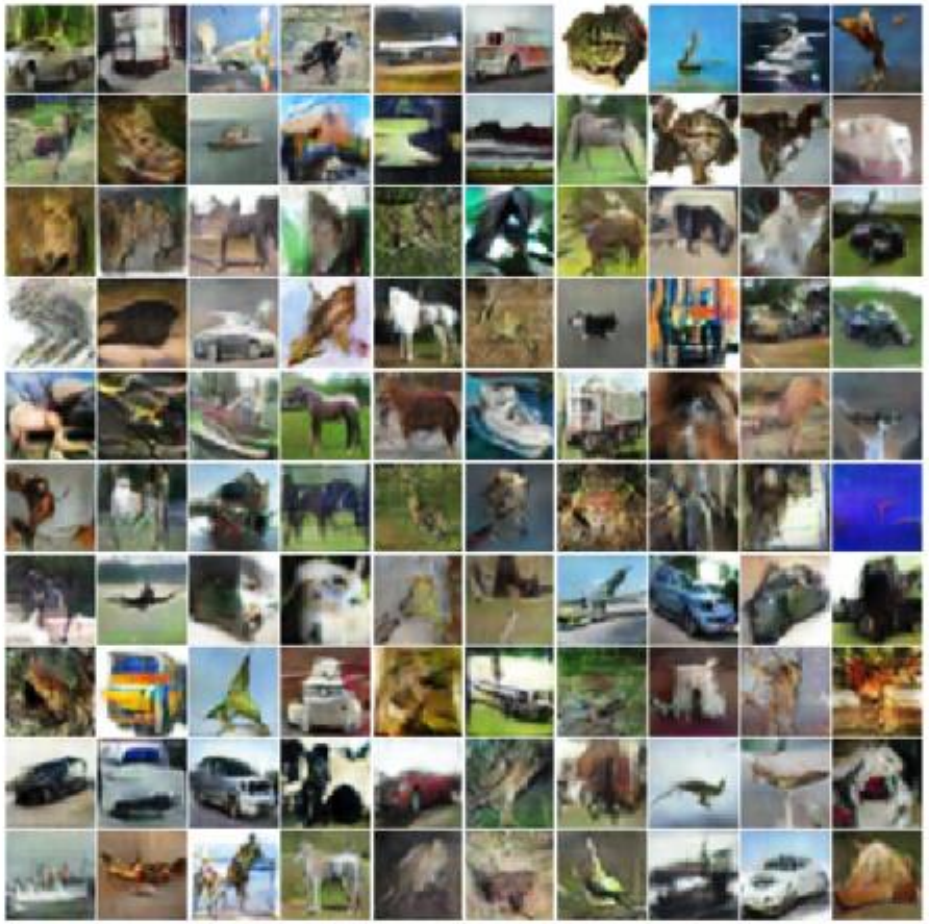
Future Work

- This framework admits many straightforward extensions:
- A **conditional generative** model p(**x**|**c**) can be obtained by adding **c** as input to both G and D.
- **Semi-supervised learning**: features from the discriminator or inference net could improve performance of classifiers when limited labeled data is available.
- …

Unsupervised Representation Learning with Deep Convolutional GAN

Generated bedrooms. Source: "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" https://arxiv.org/abs/1511.06434v2

Improved Techniques for Training GANs



Generated CIFAR-10 samples. Source: "Improved Techniques for Training GANs"
https://arxiv.org/abs/1606.03498