Generative Adversarial Networks (GANs)

Prof. Dr. Eng. Nícolas de Araújo Moreira

nicolas.araujom@gmail.com

Departamento de Engenharia de Teleinformática
Universidade Federal do Ceará

26 de outubro de 2022

Sumário

UNIVERSIDADE FEDERAL DO CEARÁ

Introduction

- Machine Learning (ML) as the sub-field of AI that uses data to “teach” a machine/program how to perform a new task;
- Most of the supervised learning algorithms are inherently discriminative, which means they learn how to model the conditional probability distribution function (PDF) $p(y|x)$. Despite the fact that one could make predictions with this probability distribution function, one is not allowed to sample new instances (simulate customers with ages) from the input distribution directly;
- It is possible to use discriminative algorithms which are not probabilistic, they are called discriminative functions;
- So discriminative algorithms map features to labels. Instead of predicting a label given certain features, they attempt to predict features given a certain label.

Universidade Federal do Ceará

Generative Models

- Generative models belong to the field of unsupervised learning, a sub-set of ML which aims to study algorithms that learn the underlying structure of the given data, without specifying a target value;
- Generative models learn the intrinsic distribution function of the input data $p(x)$ or $p(x, y)$ if there are multiple targets/classes in the dataset), allowing them to generate both synthetic inputs x' and outputs/targets y', typically given some hidden parameters;
- so...:
  - Discriminative models learn the boundary between classes;
  - Generative models model the distribution of individual classes.

UNIVERSIDADE FEDERAL DO CEARÁ

Generative Adversarial Networks (GANs) - An Overview

- Generative Adversarial Networks (GANs) are used for unsupervised learning: they are able to analyze, capture and copy the variations within a dataset, they can learn to mimic any distribution of data;
- GAN belong to the set of generative models;
- GANs were introduced in a paper by Ian J. Goodfellow and other researchers at the University of Montreal, including Yoshua Bengio;
- The GAN framework is a non-convex, two-player, non-cooperative game with continuous, high-dimensional parameters, in which each player wants to minimize its cost function;

Generative Adversarial Networks (GANs) - An Overview

- GAN is composed of two separate models, represented by neural networks: a generator denoted by G and a discriminator denoted by D. The goal of the discriminator is to tell whether data sample comes from a real data distribution, or whether it is instead generated by G.

Generating Random Variables

- Inverse Transform Method;
- Rejection Sampling: sampling from a known simple distribution and accept or reject;
- Metropolis Hasting: find Markov chain such that it's stationary distribution corresponds to the distribution from which we would like to sample our random variable.

UNIVERSIDADE FEDERAL DO CEARÁ

Inverse Transform Method

- The inverse transform method is a way to generate a random variable that follows a given distribution by making an uniform random variable goes through a well design "Transform Function"(inverse CDF);
  - Generating random variables as function of some simpler random variables;
  - It deforms/reshape the initial probability distribution.

UNIVERSIDADE FEDERAL DO CEARÁ

Inverse Transform Method

- Generating a new image of something can be reshaped into a problem of generating a random vector in the *N*-dimensional vector space that follow "this thing"probability distribution and we have suggested to use a transform method, with a neural network to model the transform function. We need to train (optimise) the network to express the right transform function.

Inverse Transform Method

- **Direct Method:** Compares the true and the generated probability distributions and backpropagating the difference (error) [Generative Matching Networks]
- **Indirect Method:** We train the generative network by making these two distributions go through a downstream task chosen such that the optimization process of the generative network with respect to the downstream task will enforce the generated distribution to the close to the true distribution [Generative Adversarial Networks]

Generative Matching Networks (GMN)

- We do not know how to express explicitly the true probability distribution. However, if we have a way to compare probability distributions based on samples, we can use it to train network.

Maxiumum Mean Discrepancy (MMD)

- Defines the distance between two probability distributions that can be computed (estimated) based on samples of these distributions.

Generative Matching Networks (GMN)

- Given a random variable with uniform probability distribution as input, we want the probability distribution of the generated output to be the "something"probability distribution. The idea of GMNs is then to optimise the network bu repeating the following steps:
  - Generate some uniform inputs;
  - Make these inputs go through the network and collect the generated outputs;
  - Compare the true probability distribution and the generated one based on the available samples. E.g.: compute the MMD distance between the sample of true images of "something"and the sample of generated ones.
  - Use backpropagation to make one step of gradient descent to lower the distance (e.g. MMD) between true and generated distributions.

UNIVERSIDADE FEDERAL DO CEARÁ

Generative Adversarial Networks (GAN) - The Indirect Training Method

- Consists in replacing this direct comparison by an indirect one that takes the form of a downstream task over these two distributions. The training of the generative is then done with respect to this task such that it forces the generated distribution to get closer and closer to the true distribution.

UNIVERSIDADE FEDERAL DO CEARÁ

Generative Adversarial Networks (GAN) - The Indirect Training Method: Discriminator

- Takes samples of true and generated data and that try to classify them as well as possibe, and a generator that is trained to fool the discriminator as much as possible.

UNIVERSIDADE FEDERAL DO CEARÁ

Generative Adversarial Networks (GAN) - The Indirect Training Method

- What we call "direct"training method would then consist in adjusting iteratively the generator (gradient descent interactions) to correct the measured difference/error between true and generated distributions. Finally, assuming the optimization process perfect, we would end up with the generated distribution that matches exactly the true distribution.

UNIVERSIDADE FEDERAL DO CEARÁ

Generative Adversarial Networks (GMN) - The Indirect Training Method

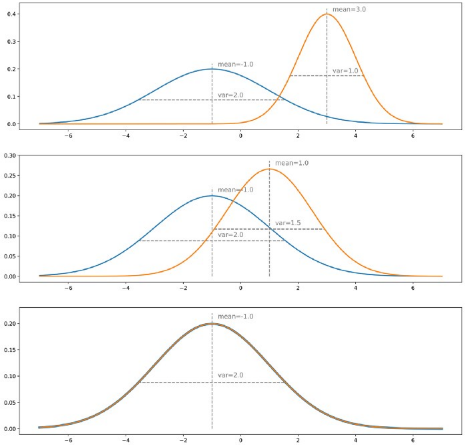


Figura: Generative Adversarial Networks (GMN) - The Indirect Training Method.

Generative Adversarial Networks (GAN)

- **Generative:** to learn a generative model, which describes how data is generated in terms of a probabilistic model;
- **Adversarial:** the training of a model is done in an adversarial setting;
- **Networks:** use deep neural networks as the artificial intelligence (AI) algorithms for training purpose.
- Generative Adversarial Networks are composed of two models:
  - The first model is called a Generator and it aims to generate new data similar to the expected one.
  - The second model is named the Discriminator. This model's goal is to recognize if an input data is 'real' — belongs to the original dataset — or if it is 'fake' — generated.
- Generator as having an adversary, the Discriminator: The Generator needs to learn how to create data in such a way that the Discriminator isn't able to distinguish it as fake anymore. The competition between these two teams is what improves their knowledge, until the Generator succeeds in creating realistic data.

Generative Adversarial Networks (GANs)

- Once generative models create data samples by capturing data distributions of type of things we want to generate, GAN captures the distribution of data and is trained in such a manner that tries to maximize the probability of the Discriminator in making a mistake. The Discriminator is based on a model that estimates the probability that the sample that it got is received from the training data and not from the Generator. The GANs are formulated as a minimax game, where the Discriminator is trying to minimize its reward $V(D, G)$ and the Generator is trying to minimize the Discriminator's reward or in other words, maximize its loss.

UNIVERSIDADE FEDERAL DO CEARÁ

Generative Adversarial Networks (GAN): Advantages

- They currently generate the sharpest images;
- They are easy to train (since no statistical inference is required), and only backpropogation is needed to obtain gradients.

Generative Adversarial Networks (GAN): Drawbacks

- GANs are difficult to optimize due to unstable training dynamics;
- No statistical inference can be done with them (except here): GANs belong to the class of direct implicit density models; they model $p(x)$ without explicitly defining the P.D.F.

Generative Adversarial Networks (GAN): Applications

- Generating realistic artwork samples (video/image/audio);
- Simulation and planning using time-series data.

Generative Adversarial Networks (GAN): Generators and Discriminators

- The generator is a neural network that models a transform function. It takes as input a simple random variable and must return, once trained, a random variable that follows the targeted distribution;
- Another neural network will model a discriminative function for discriminator. It takes an input point and returns as an output the probability of this point to be a "true"one.
- Once defined, the two networks can then be trained jointly (at the same time) with opposite goals:
  - The goal of the generator is to fool the discriminator, so the generative neural network is trained to maximise the final classification error (between true and generated data).
  - The goal of discriminator is to detect fake generated data, so the discriminative neural network is trained to minimise the final classification error.

Generative Adversarial Networks (GAN): Generators and Discriminators

- So, at each iteration of the training process, the weights of the generative network are updated in order to increase the classification error (error gradient ascent over the generator's parameters) whereas the weights of the discriminative network are updated so that to decrease this error (error gradient descent over the discriminator's parameters).
- The opposite goals explains the name "adversarial networks": both networks try to beat each other and, doing so, they are both becoming better and better. The competition between them makes these two networks "progress"with respect to their respective goals.

UNIVERSIDADE FEDERAL DO CEARÁ

Generative Adversarial Networks (GAN): Generators and Discriminators

- Neural networks modelling essentially requires to define two things: an architecture and a loss function. We have already described the architecture of Generative Adversarial Networks and it consists in two networks:
  - A generative network $G(\cdot)$ that takes a random input z with density P_z and returns an output $X_g = G(z)$ that should follow (after training) the targeted probability distribution.
  - A discriminative network $D(\cdot)$ that takes an input X that can be a "true"one (X_t, whose density is denoted P_t) or generated one (X_g, whose density is denoted P_t) or generated one (X_g, whose density is denoted by the density P_z going through G) and that returns the probability $P(X)$ of X to be a "true"data.

Generative Adversarial Networks (GAN): Generators and Discriminators

- The expected absolute error of the discriminator can the be expressed as:

$$E(G,D) = \frac{1}{2}E_{x\sim p_t}[1-D(x)]+\frac{1}{2}E_{z\sim p_z}[D(G(z))] = \frac{1}{2}(E_{x\sim p_t}+E_{x\sim p_g}[D(x)]) \quad (1)$$

- The goal of the generator is to fool the discriminator whose goal is to be able to distinguish between true and generated data. So, when training the generator, we want to maximise this error while we try to minimise it for the discriminator. It gives us:

$$\underset{G}{max}(\underset{D}{min}E(G,D)) \quad (2)$$

- For any given generator G (along with the induced probability density P_G), the best possible discriminator is the one that minimizes:

$$E_{x\sim p_t}[1-D(x)]+E_{x\sim P_g} = \int_{\mathbb{R}}(1-D(x))P_t(x)+D(x)D_g(d)\mathrm{d}x \quad (3)$$

Universidade Federal do Ceará

Generative Adversarial Networks (GAN): Generators and Discriminators

- In order to minimise (with respect to D) this integral, we can minimise the function inside the integral for each value of X. It then defines the best possible discriminator for a given generator.

$$1 I_{P_t(x) \geq P_g(x)} \tag{4}$$

- We then search G that minimises:

$$\int_R (1 - D^*_G(x)) p_t(x) + D^*_G P_g(x) \mathrm{d}x = \int_{\mathbb{R}} min(p_t(x), p_g(x)) \mathrm{d}x \tag{5}$$

- Again, in order to maximize (with respect to G) this integral, we can maximize the function inside the integral for each value of x. As the density P_t is independent of the generator G, we can't do better than setting G such that:

$$P_g(x) \geq P_t(x) \tag{6}$$

Universidade Federal do Ceará

Generative Adversarial Networks (GAN): Generators and Discriminators

- Of course, as P_g is a probability density that should integrate to 1, we have necessarily the best G:

$$P_g(x) = P_t(x) \tag{7}$$

- So, we have shown that, in an ideal case with unlimited capacities generator and discriminator, the optimal point of the adversarial setting is such that the generator procedures the same density as the true density and the discriminator can't do better than being true in one case out of two, just like the intuition told us. Finally, notice also that G maximises:

$$\frac{1}{2}\int_{\mathbb{R}} min(P_t(x), P_g(x))\mathrm{d}x = \int_{\mathbb{R}} \frac{minP_t(x), P_g(x)}{P_t(x)+P_g(x)} \frac{P_t(x)+P_g(x)}{2}\mathrm{d}x \tag{8}$$

Universidade Federal do Ceará

Generative Adversarial Networks (GAN): Generators and Discriminators

- Under this form, we better see that *G* wants to maximise the expected probability of the discriminator to be wrong.
- Each side of the GAN can overpower the other. If the discriminator is too good, it will return values so close to 0 or 1 that the generator will struggle to read the gradient. If the generator is too good, it will persistently exploit weaknesses in the discriminator that lead to false negatives. This may be mitigated by the nets' respective learning rates. The two neural networks must have a similar "skill level."

UNIVERSIDADE FEDERAL DO CEARÁ

Generative Adversarial Networks (GAN) - The Algorithm

- The generator takes in random numbers and returns an image;
- This generated image is fed into the discriminator alongside a stream of images taken from the actual, ground-truth dataset;
- The discriminator takes in both real and fake images and returns probabilities, a number between 0 and 1, with 1 representing a prediction of authenticity and 0 representing fake;

Generative Adversarial Networks (GAN) - The Algorithm

- So you have a double feedback loop: The discriminator is in a feedback loop with the ground truth of the images, which we know; the generator is in a feedback loop with the discriminator.
  - Part 1: The Discriminator is trained while the generator is idle. In this phase, the network is only forward propagated and no back-propagation is done. The Discriminator is trained on real data for *n* epochs, and see if it can correctly predict them as real. Also, in this phase, the Discriminator is also trained on the fake generated data from the Generator and see if it can correctly predict them as fake;
  - Part 2: The Generator is trained while the Discriminator is idle. After the Discriminator is trained by the generated fake data of the Generator, we can get its predictions and use the results for training the Generator and get better from the previous state to try and fool the discriminator.

UNIVERSIDADE FEDERAL DO CEARÁ

Generative Adversarial Networks (GAN) - The Algorithm

- The above method is repeated for a few epochs and then manually check the fake data if it seems genuine. If it seems acceptable, then the training is stopped, otherwise, its allowed to continue for few more epochs.

Generative Adversarial Networks (GAN) - The Algorithm

- The fundamental steps to train a GAN can be described as following:
  1. Sample a noise set and a real-data set, each with size m.
  2. Train the Discriminator on this data.
  3. Sample a different noise subset with size m.
  4. Train the Generator on this data.
  5. Repeat from Step 1.

Generative Adversarial Networks (GAN) - The Algorithm: Training and Modelling

- Since during training both the Discriminator and Generator are trying to optimize opposite loss functions, they can be thought of two agents playing a minimax game with value function $V(G, D)$. In this minimax game, the generator is trying to maximize it's probability of having it's outputs recognized as real, while the discriminator is trying to minimize this same value;

$$\underset{G}{min}\underset{D}{max}\mathbb{E}_{x\sim q_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p(z)}[\log D(G(z))] \quad (9)$$

- Where G denotes Generator, D Discriminator, $P_{data}(x)$ distribution of real data, $P(z)$ distribution of generator, x sample from $p_d ata(x)$, z sample from $p(z)$, $D(x)$ discriminator network and $G(z)$ generator network;

Generative Adversarial Networks (GAN) - The Algorithm: Training and Modelling

- This expression means that the generator optimizes for maximally confusing the discriminator by trying to make it output high probabilities for fake data samples. On the other hand, the discriminator tries to become better at distinguishing samples coming from G from samples coming from the real distribution;

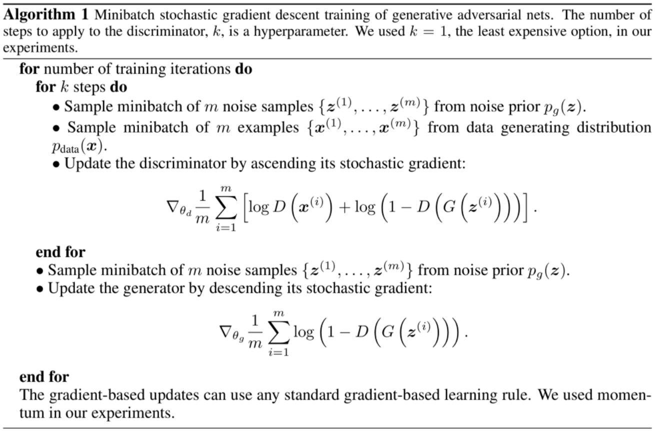
Generative Adversarial Networks (GAN) - The Algorithm: Training and Modelling

- In practice, the logarithm of the probability (e.g. $\log D(\cdot)$) is used in the loss functions instead of the raw probabilities, since using a log loss heavily penalises classifiers that are confident about an incorrect classification.

Generative Adversarial Networks (GAN) - The Algorithm: Stop Criteria

- After several steps of training, if the Generator and Discriminator have enough capacity (if the networks can approximate the objective functions), they will reach a point at which both cannot improve anymore. At this point, the generator generates realistic synthetic data, and the discriminator is unable to differentiate between the two types of input.

Generative Adversarial Networks (GAN) - The Algorithm



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.
```

Figura: Generative Adversarial Networks (GMN) - Algorithm.

UNIVERSIDADE FEDERAL DO CEARÁ

Vanilla GAN

- Is the simplest type of GAN. Here, the Generator and the Discriminator are simple multi-layer perceptrons. In vanilla GAN, the algorithm is really simple, it tries to optimize the mathematical equation using stochastic gradient descent.

UNIVERSIDADE FEDERAL DO CEARÁ

Deep Convolutional GAN (DCGAN)

- It is composed of ConvNets in place of MLPs. The ConvNets are implemented without max pooling, which is in fact replaced by convolutional stride. Also, the layers are not fully connected. Is one of the most popular GAN;
- Introduced convolutions to the generator and discriminator networks. It uses also batch normalization, LeakyRelu and Tanh activations.

Deep Convolutional GAN (DCGAN)

- Let *m* be a measure on the σ-algebra of Borel sets of a Hausdorff topological space X. The measure *m* is called inner regular or tight if, for any open set U, $m(U)$ is the supremum of $m(K)$ over all compact subsets K of U. The measure *m* is called outer regular if, for any borel set B, m(B), is the infimum of $m(U)$ over all open sets U containing B. The measure *m* is called locally finite if every point of X has a neighborhood U for which $m(U)$ is finite. The measure *m* is called a Radon measure if it is inner regular, outer regular and locally finite.

Wasserstein Metric

- **Wasserstein Metric:** Also Knwon as Kantorovich-Rubinstein metric is a distance function defined between probability distributions on a given metric space M. Let (M, d) be a metric space for which every probability measure on M is a random masure. For $p \geq 1$, let $P_p(M)$ denote the collection of all probability measures μ on M with finite p-th moment. Then, there exists some $x_0 \in M$ such that:

$$\int_M d(x, x_0)^p \mathrm{d}\mu(x) < \infty \tag{10}$$

The p-th Wasserstein distance between two probability measures μ and nu in $P_p(M)$ is defined as:

$$W_p(\mu, \nu) := \left(\underset{\gamma \in \Gamma(\mu,\nu)}{inf} \int_{M \times M} d(x, y)^p \mathrm{d}\gamma(x, y) \right)^{1/p} \tag{11}$$

UNIVERSIDADE FEDERAL DO CEARÁ

Wasserstein Metric

- or equivalently:

$$W_p(\mu, \nu) = (inf E[d(X, Y)^p])^{1/p} \tag{12}$$

where $\Gamma(\mu, \nu)$ denotes the collection of all measures on *MxM* with marginals μ and ν on the first and second factors respectively and $E[\cdot]$ denotes the expected value of a random variable. The set $\Gamma(\mu, \nu)$ is also called the set of all couplings of μ and ν.

Other Variations

- **Wasserstein GAN - WGAN:** Propose a new cost function that has a smoother gradient (Wasserstein distance). The discriminator is called critic;
- **Self-Attention GAN:** Convolutions are quite bad at capturing long-term dependencies in input samples;
- **Conditional GAN (CGAN):** Is a deep learning method in wich some conditional parameters are put into place. In CGAN, an additional parameter y is added to the Generator for generating the corresponding data. Labels are also put into the input to the Discriminator in order for the Discriminator to help distinguish the real data from the fake generated data;

UNIVERSIDADE FEDERAL DO CEARÁ

Other Variations

- **Laplacian Pyramid GAN (LAPGAN):**The Laplancian pyramid is a linear invertible image representation consisting of a set of band-pass images, spaced an octave apart, plus a low-frequency residual. This approach uses multiple numbers of Generator and Discriminator networks and different leves of the Laplacian Pyramid;
- **Super Resolution GAN (SRGAN):** A deep neural network is used along with an adversarial network.
- **BigGAN**

Summary of main ideas

- In GANs, there is a generator and a discriminator. The Generator fake samples of data and tries to fool the Discriminator. The discriminator are both neural networks and they both run in competition with each other in the training phase. Generator and Discriminator get better and better in their respective jobs after each repetition;
- One neural network, called the generator, generates new data instances, while the other, the discriminator, evaluates them for authenticity; i.e. the discriminator decides whether each instance of data that it reviews belongs to the actual training dataset or not;

Summary of main ideas

- A neural network $G(z, \theta_1)$ is used to model the Generator mentioned above. It's role is mapping input noise variables z to the desired data space x (say images). Conversely, a second neural network $D(x, \theta_2)$ models the discriminator and outputs the probability that the data came from the real dataset, in the range $[0, 1]$. In both cases, θ_i represents the weights or parameters that define each neural network.

Summary of main ideas

- As a result, the Discriminator is trained to correctly classify the input data as either real or fake. This means it's weights are updated as to maximize the probability that any real data input x is classified as belonging to the real dataset, while minimizing the probability that any fake image is classified as belonging to the real dataset. In more technical terms, the loss/error function used maximizes the function $D(x)$, and it also minimizes $D(G(z))$;
- The generator is a deep neural network. Its input is a vector of random noise (usually Gaussian or Uniform distribution) and outputs a data sample from the distribution we want to capture. The goal of the generator is to generate data samples such as to fool the discriminator;

Universidade Federal do Ceará

Summary of main ideas

- Furthermore, the Generator is trained to fool the Discriminator by generating data as realistic as possible, which means that the Generator's weight's are optimized to maximize the probability that any fake image is classified as belonging to the real dataset. Formally this means that the loss/error function used for this network maximizes $D(G(z))$.

Summary of main ideas

- The generator is an inverse convolutional network, in a sense: While a standard convolutional classifier takes an image and downsamples it to produce a probability, the generator takes a vector of random noise and upsamples it to an image. The first throws away data through downsampling techniques like maxpooling, and the second generates new data;
- The discriminator is a neural network. Its goal is to discriminate between real and fake samples. Its input is a data sample, either coming from generator of from the actual data distribution. The output is a simple number, representing the probability that the input was real. A high probability means that the discriminator is confident that the samples he's being fed is a genuine one. On the contrary, a low probability shows high confidence in the fact that the sample is coming from the generator network.

UNIVERSIDADE FEDERAL DO CEARÁ

Conclusions

- Computers can basically generate simple pseudo-random variables (for example, they can generate variables that follow very closely a uniform distribution);
- There exists different ways to generate more complex random variables including the notion of "transform method"that consists in expressing a random variable as a function of some simpler random variables;
- In machine learning, the generative models try to generate data from a given (complex) probability distribution;
- Deep learning generative models are modelled as neural networks (very complex functions) that take as input a simple random variable and that return a random variable that follows the targeted distribution ("transform method"like)
- These generative networks can be trained "directly"(by comparing the distribution of generated data to the true distribution): this is the idea of Generative Matching Networks;

Conclusions

- These Generative Networks can also be trained "indirectly"(by training to fool another network that is trained at the same time to distinguish "generated"data from "true"data): this is the idea of Generative Adversarial Network.
- GANs they have proven to be really succesfull in modeling and generating high dimensional data, which is why they've become so popular. Nevertheless they are not the only types of Generative Models, others include Variational Autoencoders (VAEs) and pixelCNN/pixelRNN and real NVP.

Future Works

- No statistical inference can be done with them, except with Adversarially Learned Inference

Obrigado pela Atenção!

Contato:

`nicolas.araujom@gmail.com`

UNIVERSIDADE FEDERAL DO CEARÁ