Deep Learning : Generative Adversarial Networks and Cycle-GAN

Melchior Prugniaud

May 2020

1 Unsupervised representation learning with deep convolutional generative adversarial networks

In this paper, authors propose a novel idea to build good image representations : training Generative Adversarial Networks (GANs) and then reuse some elements of the generator and discriminator networks as feature extractors for supervised learning. They name this architecture Deep Convolutional GANs (DCGAN).

First of all, GANs are neural networks that use two neural networks fighting each one versus the other to create new synthetic data that can pass for real one. The main topics of GANs are video, audio and image generation. The first network is the generator network. It purpose is to generate new data instances from random noise. And the discriminator network is a classifier which is trying to find real data versus data generated by the generator.

1.1 Improvements of DCGAN

Authors propose to five elements to produce a stable DCGAN :

- Discriminator: Use strided convolutions instead of pooling layers (learn its own spacial downsampling)
  Generator: Use fractionnal-strided convolutions (learn its own spacial upsampling)
- Normalize batch in order to level off learning (zero mean, and unit variance). This batchnorm is used in generator and discriminator
- In the generator network use ReLu activation function for all except the output layers. Use Tanh for the output layer
- In the discriminator network use LeakyRelu action function for all layers
- Remove fully connected hidden layers

	Value
Mini-batch size	128
Weight init	N(0,0.02)
Optimizer	Adam
Learning rate	0.0002
β_1	0.5

Table 1: Parameters of adversarial training

They mostly proposed DCGAN to reduce the difficulty of learning using elements described above. But as they used in their article, the DCGAN can serve as a feature extractor that reduces the dimensionality of input with a semantically-preserving way. Then you could use model like SVM on top of it to do other machine learning experiences.

2 Wasserstein GAN

2.1 Introducing distances and find the optimal one

They first introduce different distances to qualify the similarity between two probability distributions:

- Total variation (TV) distance

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \mathbb{P}_{A\in\Sigma}\mathbb{P}_r(A) - \mathbb{P}_g(A) \tag{1}$$

- Kullback-Leibler (KL) distance

$$KL(\mathbb{P}_r\|\mathbb{P}_g) = \int log(\frac{\mathbb{P}_r(x)}{\mathbb{P}_g(x)})\mathbb{P}_r d\mu(x) \tag{2}$$

- Jensen-Shannon (JS) distance

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r\|\mathbb{P}_m) + KL(\mathbb{P}_g\|\mathbb{P}_m) \tag{3}$$

  Where $\mathbb{P}_m = \frac{\mathbb{P}_r+\mathbb{P}_g}{2}$

- Earth-Mover (EM) distance or Wasserstein 1

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma\in\Pi(\mathbb{P}_r,\mathbb{P}_g)} \mathbb{E}_{(x,y)\ \ y}[\|x-y\|] \tag{4}$$

  Where $Pi(\mathbb{P}_r, \mathbb{P}_g)$ is the set of all joint distributions (x, y).

So, the Wasserstein Distance is also called Earth Mover's distance (EM) because in a informal way it can be defined as the minimum energy cost of moving a pile of dirt in the shape of one probability distribution to the shape

of the other distribution.
They then define two theorems and one corollary. The first theorem with the corollary is used to prove that learning with minimizing EM distance is useful when using neural networks. The following results proves that Wasserstein is better than JS or KL divergence. The first theorem is :

Theorem 1. *Let $\mathbb{P}_r$ be a fixed distribution over . Let Z be a random variable over another space ζ. Let $g : \zeta \times \mathbb{R}^d \to \chi$ be a function, that will be denoted $g_\theta(z)$ with z the first coordinate and θ the second. Let $\mathbb{P}_\theta$ the distribution of $g_\theta(Z)$. Then :*

- *If g is continuous in θ, so is $W(\mathbb{P}_r, \mathbb{P}_\theta$*
- *If g is locally Lipschitz and satisfies regularity assumption 1, then $W(\mathbb{P}_r, \mathbb{P}_\theta)$ is continuous everywhere, and differentiable almost everywhere*
- *Both statement above are false for the JS divergence $JS(\mathbb{P}_r, \mathbb{P}_\theta)$ and all the KLs*

And it's corollary is :

Corollary 1. *Let g_θ any feed-forward neural network parameterized by θ and $p(z)$ a prior over z such that $\mathbb{E}_{z\ p(z)}[\|z\|] < \infty$. This assumption is satisfied and therefore $W(\mathbb{P}_r, \mathbb{P}_\theta)$ is continuous everywhere and differentiable almost everywhere*

This result show that using the EM distance will be more useful for the problem than the others defined before.
The second theorem is showing that only the EM distance is sensible when learning distributions supported by low dimensional manifolds.

Theorem 2. *Let $\mathbb{P}$ be a distribution on a compact space χ and $(\mathbb{P}_n)_{n\in\mathbb{N}}$ be a sequence of distribution on χ. Then, considering all limits as $n \to \infty$,*

- *The following statements are equivalent*
  - $\delta(\mathbb{P}_n, \mathbb{P}) \to 0$ *with δ the total variation distance*
  - $JS(\mathbb{P}_n, \mathbb{P}) \to 0$ *with JS the Jensen-Shannon divergence*
- *The following statements are equivalent*
  - $W(\mathbb{P}_n, \mathbb{P}) \to 0$
  - $\mathbb{P}_n \to \mathbb{P} where \to$ *represents convergence in distribution for random variables*
- $KL(\mathbb{P}_n \| \mathbb{P} \to 0$ *or* $KL(\mathbb{P} \| \mathbb{P}_n) \to 0$ *imply the statements in the first bullet point*
- *The first statements imply the second statements*

2.2 Improvements of WGAN

Using all of this results and theorem, they find that choose the Wasserstein distance as a GAN cost function is great idea instead of using noise. In that maner, WGAN learns whether the generator is performing well or not. Using this, the gradient is smoother in every points and learn better.

Then they find a function f which solves the maximization problem of

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f`\|_{L\leq 1}} \mathbb{E}_{x\ \mathbb{P}_r}[f(x)] - \mathbb{E}_{x\ \mathbb{P}_\theta}[f(x)] \tag{5}$$

This function need just to be 1-Lipschitz to calculate the Wasserstein distance. In order to approximate this function, a solution could be training a GAN but a different one which they called the Wassertein Generative Adversarial Network (WGAN). The algorithm of WGAN is the following :

Algorithm 1: WGAN. $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_critic = 5$

```
while $\theta$ has not converged do
    for $t = 0, ..., n_{critic}$ do
        Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data
        Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples
        $g_w \leftarrow \nabla_w[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)} - \frac{1}{m}\sum_{i=1}^m f_w(g_(\theta(z^{(i)}))]$
        $w \leftarrow w + \alpha \cdot RMSProp(w, g_w)$
        $w \leftarrow clip(w, -c, c)$
    end
    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples
    $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
    $\theta \leftarrow \theta - \alpha \cdot RMSProp(\theta, g_\theta)$
end
```

Where :

- α is the learning rate
- c the clipping parameter
- m the batch size
- n_{critic} the number of iterations of the critic per generator iteration
- w_0 the initial critic parameters and θ_0 initial generator's parameters

The main differences between GAN and DCGAN is that the discriminator is the DCGAN output a scalar score instead of a probability. This score tell us how real are the images. The discriminator is also renamed the critic. Remembering that the functionf f has to be 1-Lipschitz, WGAN uses a clipping to restrict the maximum weight value, this means that the weight of the discriminator network are within in a certain range controlled by c.