DAHAN Nicolas

Deep Learning
ENSAE Paris
Marco Cuturi

Projet de Deep Learning 2021

1 Introduction

En **apprentissage supervisé**, on dispose de données X et d'un label y et l'objectif est d'apprendre une fonction pour faire correspondre X à y (par exemple la régression, la classification, la détection d'objets,...) tandis qu'en **apprentissage non supervisé**, il n'y a pas de label et l'objectif est de trouver une structure cachée sous-jacente des données (par exemple le clustering, la reduction de dimension,...).
L'objectif des **modèles génératifs** est de générer de nouveaux échantillons de données à partir d'une distribution. Ces modèles sont utilisés dans des problèmes tels que l'estimation de la densité qui est un problème d'apprentissage non supervisé. Ici nous présentons plusieurs variants d'un modèle génératif : Les **GANs**.

2 GAN

L'idée sous-jacente d'un GAN est d'échantillonner selon une loi simple (uniforme ou gaussiene par exemple) ou un bruit et d'apprendre la transformation (paramétres du modèle) pour obtenir la distribution réelle des données.
Par la suite et par simplicité d'expression, on considère que les données traitées sont des images, mais on peut considérer tout type de données (vidéo, audio, texte,...)

Les GANs, inventés par Ian Goodfellow et al. en 2014 dans *Generative Adversarial Nets*, représentent de bons modèles pour effectuer cela.

Ils comportent deux réseaux, en concurrence l'un avec l'autre :

- ◇ un générateur (G) qui essaie de tromper le discriminateur en générant des images d'apparence réelle.
- ◇ un discriminateur (D) qui essaie de distinguer les images réelles des fausses images générées par le générateur.

L'entraînement consiste à alterner entre l'entraînement du discriminateur et du générateur. Pour le discriminateur, on calcule sa perte sur des images réelles d'entraînement. On génère de fausses images puis on calcule la perte du discriminateur sur des images générées fictives. Et on effectue une rétro-popagation et une étape d'optimisation pour mettre à jour les poids du discriminateur. Pour le générateur, on commence par générer de fausses images z selon une loi que l'on choisie p_z, on calcule la perte et on effectue une retro-popagation et une étape d'optimisation pour mettre à jour les poids du générateur.
D et G joue un jeu de *minmax* dans lequel D essaie de maximiser la probabilité qu'il classe correctement les vraies et les fausses images ($D(x)$). Et G essaie de minimiser la probabilité que D prédise que ses produits soient faux ($D(G(z))$).

La perte du GAN est défini par :

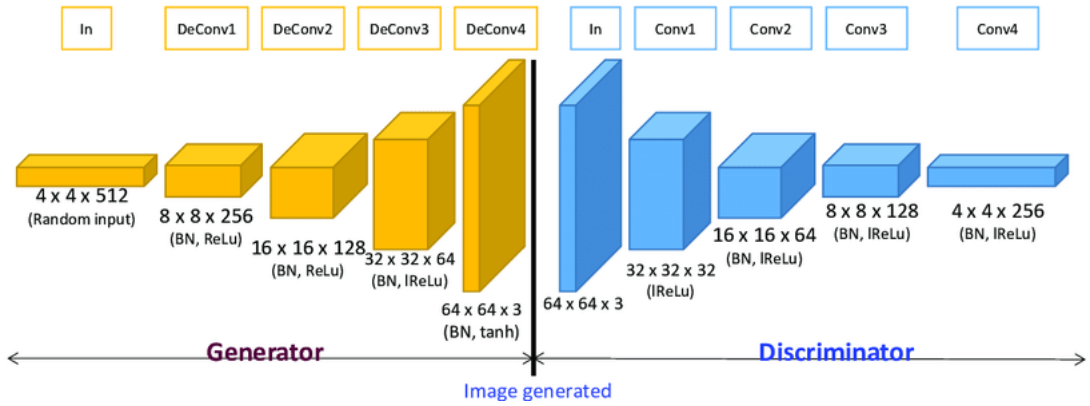
$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}\big[\log D(x)\big] + \mathbb{E}_{z\sim p_z(z)}\big[\log(1-D(G(z)))\big] \tag{1}$$

En pratique, le logarithme de la probabilité est utilisé dans la fonctions de perte au lieu des probabilités brutes, car l'utilisation d'une perte logarithmique pénalise fortement les classifieurs qui sont sûrs d'une classification incorrecte.

3 GAN convolutionnel profond (DC-GAN)

Un GAN convolutionnel profond (*Deep Convolutional GAN*) est une extension directe du GAN, sauf qu'il utilise explicitement des couches convolutionnelles et transposées-convolutionnelles dans le discriminateur et le générateur, respectivement.

Le générateur et le discriminateur du DC-GAN utilisent tous les deux la normalisation par lots (*batch normalization*) et n'utilisent pas de *max pooling*.
De plus, le générateur utilise la *leaky relu* comme fonction d'activation pour toutes les couches, sauf celle de sortie, et le discriminateur l'utilise aussi pour toutes les couches.



4 Wassertein GAN

Dans le papier d'origine [3], les auteurs s'intéressent à l'apprentissage d'une fonction g_θ (le réseau) qui transforme une distribution existante d'un variable Z en P_θ. Le but étant de rapprocher le plus possible la loi P_θ de la loi P_r (loi des données réelles dont on dispose).
Les auteurs considèrent la distance de Wasserstein entre deux loi de probabilité P_r et P_g :

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big] \tag{2}$$

où $\Pi(P_r, P_g)$ est l'ensemble des lois de probabilité jointes γ avec P_r et P_g comme lois marginales.

Les auteurs présentent aussi d'autres distances comme la Variation Total (TV), la divergence Kullback-Leibler (KL) et la divergence Jenson-Shannon (JS) mais celles-ci ne sont pas optimales pour un apprentissage de distribution dans des espaces de petite dimension. Cela est du à des problèmes de régularités de ces distances et au fait que, pour toute suite de loi de probabilité, la convergence sous (KL), (JS) et (TV) implique la convergence avec la distance de Wasserstein.
Le problème est que le calcul exact de la distance de Wasserstein est irréalisable.
Un résultat à partir de la dualité de Kantorovich-Rubinstein montre que (2) est équivalent à :

$$W(P_r, P_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim P_r}[f(x)] - \mathbb{E}_{x\sim P_\theta}[f(x)] \tag{3}$$

où le suprémum est pris sur l'ensemble des fonctions 1-Lipschitzienne.

Avec un modèle génératifde, de type GAN, on souhaite entraîner $P_\theta = g_\theta(Z)$ à correspondre à P_r. On définit $\{f_w\}_{w \in \mathcal{W}}$ un famille paramétrique de fonctions, où où w sont les poids du réseau et W est l'ensemble de tous les poids possibles.
Intuitivement, étant donné un g_θ fixe, on peut calculer le f_w optimal pour la distance de Wasserstein. Nous pouvons alors faire un rétropropagation à travers $W(P_r, g_\theta(Z))$ pour obtenir le gradient en θ.

$$\begin{aligned}\nabla_\theta W(P_r, P_\theta) &= \nabla_\theta(\mathbb{E}_{x \sim P_r}[f_w(x)] - \mathbb{E}_{z \sim Z}[f_w(g_\theta(z))]) \\ &= -\mathbb{E}_{z \sim Z}[\nabla_\theta f_w(g_\theta(z))]\end{aligned}$$

Le processus d'entraînement est divisé en 3 étapes :

- ◇ Pour un θ fixé, on calcule une approximation de $W(P_r, P_\theta)$ en entraînant f_w jusqu'à convergence.
- ◇ Un fois que l'on a le f_w optimale, on calcule le gradient $-\mathbb{E}_{z \sim Z}[\nabla_\theta f_w(g_\theta(z))]$ en échantillonnant plusieurs $z \sim Z$
- ◇ On met à jour θ et on répète le processus.

Toute cette dérivation ne fonctionne que lorsque la famille de fonctions $\{f_w\}_{w \in \mathcal{W}}$ est Lipschitzienne. Pour garantir que cela soit vrai, les auteurs utilisent le bridage (*clamping*) des poids.

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

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

5 Pix2pix

Pix2pix utilise un réseau adversarial génératif conditionnel (**CGAN**) pour apprendre une correspondance entre une image d'entrée et une image de sortie. Il est utilisé pour la traduction d'image à image.

Pour entraîner le discriminateur, le générateur génère d'abord une image de sortie. Le discriminateur examine la paire entrée/cible et la paire entrée/sortie et donne son avis sur le réalisme de ces images. Les poids du discriminateur sont ensuite ajustés en fonction de l'erreur de classification de la paire entrée/sortie et de la paire entrée/cible. Les poids du générateur sont ajustés en fonction de la sortie du discriminateur ainsi que de la différence entre l'image de sortie et l'image cible.

Le générateur se compose d'un encodeur qui convertit l'image d'entrée en une représentation plus petite, et d'un décodeur, qui ressemble à un générateur typique, une série de couches de transposition-convolution

qui inversent les actions des couches d'encodeur. Le discriminateur, au lieu d'identifier une seule image comme vraie ou fausse, examine des paires d'images (image d'entrée et image inconnue qui est soit l'image cible, soit l'image générée), et émet un label pour la paire comme vraie ou fausse. La fonction de perte est la suivante :

$$\min_G \max_D \mathbb{E}_{x,y}\big[\log D(x, G(x))\big] + \mathbb{E}_{x,y}\big[\log(1 - D(G(x, y)))\big] \tag{4}$$

Le problème principal avec pix2pix est l'entraînement car les deux espaces d'images doivent être préformatés en une seule paire d'images $\{X, Y\}$ qui contient les deux images étroitement corrélées.

6 CycleGAN

Le CycleGAN est également utilisé pour la traduction d'image à image. L'objectif du CycleGAN est d'entraîner des générateurs qui apprennent à transformer une image du domaine X en une image qui semble appartenir au domaine Y (et vice versa). Le CycleGAN utilise une approche non supervisée pour apprendre le *mapping* d'un domaine d'image à un autre, c'est-à-dire que les images d'entraînement n'ont pas de label. La correspondance directe entre les images individuelles n'est pas requise (à la différence de Pix2pix). Ainsi les données d'entraînement ne sont pas distribuées en paire de données.

Le modèle inclue 2 générateurs $F : Y \to X$ et $G : X \to Y$, et deux discriminateurs antagonistes D_X et D_Y capables de classifier si une image d'entrée d'un domaine (X ou Y) est réelle ou fausse par rapport à l'autre domaine (resp. Y ou X).

Ainsi les auteurs considèrent plusieurs pertes dans l'objectif :

◇ Deux pertes antagonistes :

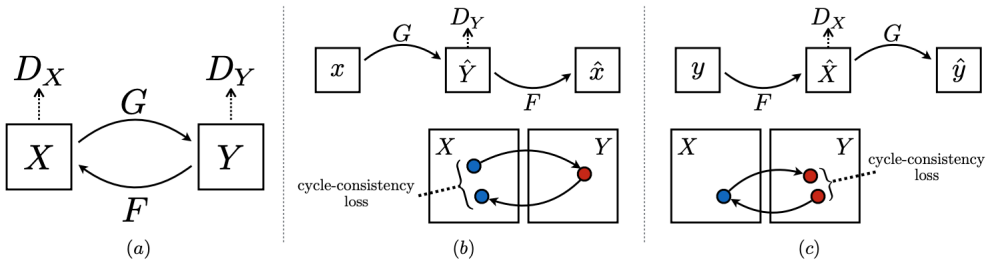
$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}\big[\log D_Y(y)\big] + \mathbb{E}_{x \sim p_{data}(x)}\big[\log(1 - D_Y(G(x)))\big] \tag{5}$$

$$\mathcal{L}_{GAN}(F, D_X, Y, X) = \mathbb{E}_{x \sim p_{data}(x)}\big[\log D_X(x)\big] + \mathbb{E}_{y \sim p_{data}(y)}\big[\log(1 - D_X(G(y)))\big] \tag{6}$$

◇ Une perte de cohérence du cycle :

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}\big[||F(G(x) - x||_1\big] + \mathbb{E}_{y \sim p_{data}(y)}\big[||G(F(y) - y||_1\big] \tag{7}$$

L'intérêt de cette perte est de s'assurer que la "traduction inverse" d'une image fournit l'image d'origine, i.e. $F(G(x)) \approx x$ et $G(F(x)) \approx y$.



Ainsi l'objectif global est :

$$\mathcal{L}_{CycleGAN}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda\, \mathcal{L}_{cyc}(G, F) \qquad (8)$$

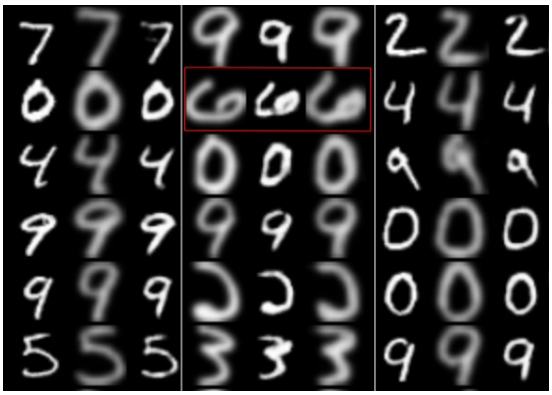
La dernière perte de l'équation est vu comme un terme pénalisé avec λ comme hyperparamètre à choisir. Et le but est de résoudre

$$G^*, F^* = \arg\min_{G,F} \max_{D_X, D_Y} \mathcal{L}_{CycleGAN}(G, F, D_X, D_Y) \qquad (9)$$

Une implantation est réalisée dans le cadre de projet. L'objectif est de "traduire" les images de la base de données "MNIST" en les images de la base de données "USPS". L'architecture du CycleGAN implémenté est détaillée dans le notebook joint à ce rapport.

On expose ici, une image (tableau) illustrant une partie des résultats obtenus sur une base test.

FIGURE 1 – 6 lignes et 3 colonnes. Chaque ligne d'une colonne présente à gauche, un chiffre de la base test de MNIST, au milieu sa traduction dans la base USPS grâce au CycleGAN, et à droite, sa reconstruction dans la base MNIST - exemple dans l'encadré en rouge.



Références

[1] ALEC RADFORD, LUKE METZ, S. C. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. Cornell University, 2016.

[2] JUN-YAN ZHU, TAESUNG PARK, P. I. A. A. E. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. Berkeley AI Research (BAIR), 2017.

[3] MARTIN ARJOVSKY, SOUMITH CHINTALA, L. B. *Wasserstein GAN*. Cornell University, 2017.

[4] PHILLIP ISOLA, JUN-YAN ZHU, T. Z. A. A. E. *Image-to-Image Translation with Conditional Adversarial Networks*. Cornell University, 2016.