Les réseaux de neurones antagonistes génératifs

Jeremie Sayag

10 Janvier 2021

Introduction

De nos jours, les réseaux de neurones antagonistes génératifs sont très connus pour leur prouesses spectaculaires dans le domaine de la génération de contenu. Aussi surprenant que cela puisse paraître, il est désormais possible de construire des modèles d'intelligence artificielle capable de générer un contenu artificiel réaliste. En témoigne les nombreux sites sur lesquels nous pouvons tester ces modèles comme `thispersondoesnotexist.com` ou bien encore `thisartworkdoesnotexist.com`.

Dans le cadre de ce rapport, je propose d'étudier le papier fondateur des réseaux de neurones antagonistes génératifs : **"Generative Adversarial Nets"** proposé par Ian Goodfelllow et al.[**goodfellow˙generative˙2014**] .Nous allons dans un premier temps décrire la stratégie des auteurs pour le problème de génération de contenu. Ensuite, on étudiera plus en détails les concepts mathématiques derrière ce modèle ainsi que son impléméntation. Enfin, nous conclurons sur les avantages et les inconvénients de ce genre de modèle en pratique.

1 Un adversaire à convaincre

L'idée derrière les réseaux de neurones antagonistes génératifs est assez simple. Nous avons à notre disposition deux réseaux de neurones :

- Le premier est appélé **Générateur** et a pour mission de générer un contenu réaliste
- Le second est appelé **Discriminateur** se charge d'estimer quant à lui si le contenu généré par le générateur semble réaliste ou non.

La grande idée de ce modèle est de faire collaborer ces deux réseaux de neurones de telle façon à ce qu'on atteigne un équilibre au sein duquel, le générateur arrive à générer des images tellement réalistes que le discriminateur n'arrive plus à distinguer une image réelle d'une image artificielle.

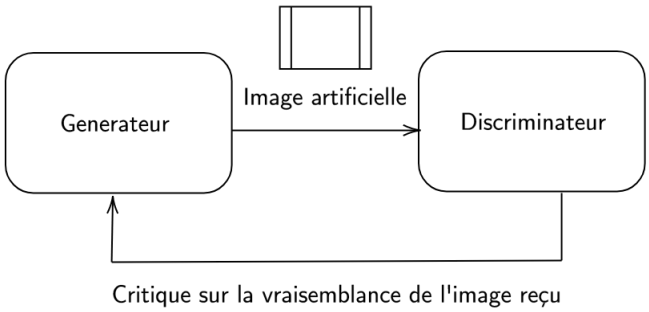

Figure 1: Schema simple du modèle de réseaux de neurones antagonistes génératifs

Derrière cette tactique simpliste se cache en réalité un entrainement difficile. Etant donné que les deux réseaux ont un objectif différents visant à faire du mieux que possible pour ne pas se faire avoir par l'autre, atteindre l'équilibre n'est pas aussi simple que cela puisse paraître. Cet **article** en parle très bien. Néanmoins, il est tout de même possible d'y arriver comme en témoigne les nombreux exploits de Nvidia récemment.

Maintenant que l'idée du modèle est décrite, il est temps de discuter en détails ce qui se cache derrière cet architecture formellement.

2 La théorie mathématique des réseaux de neurones antagonistes génératifs

Nous allons consider ici le même scenario que les auteurs. Nous avons un dataset $\mathcal{D}$ constitué d'observations x générées à partir de la même distribution p_{data}. Par exemple, nous pouvons imaginer un ensemble de visages, chaque visage étant issue d'une même distribution à déterminer.

Le but du générateur va être d'approximer cette distribution du mieux possible. Nous appelerons dans la suite p_g la distribution des observations générées par le générateur G. Maintenant, quel peut être la procèdure pour générer du contenu ? Les auteurs proposent d'imaginer qu'il existe une **variable latente z** de distribution simple et connue p_z, qui permettrait en quelquesorte d'encoder une observation qu'il faudra par la suite décoder en utilisant un réseau de neurones comme le perceptron multi-couche par exemple. Pour mieux comprendre, on peut imaginer dans le cas des visages que la variable latente donne des informations sur la couleur des cheveux,des yeux ou bien encore la forme du nez.

Le discriminateur, quant à lui, prend en entrée une observation **x** potentiellement artificielle et retourne simplement la probabilité **D**(**x**) que cette image soit artificielle ou non selon lui.

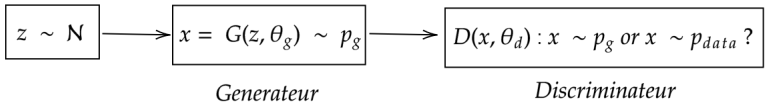

Figure 2: Schema illustratif d'une étape du GAN où la variable latente suit une loi normale

Le Discriminateur a pour objectif de bien distinguer le réel de l'artificiel. Ainsi, on comprend que pour $x \sim p_{data}$, D(x) doit etre maximale tandis que pour $x \sim p_g$, D(x) doit être minimale.

Du côté du générateur, son rôle est simplement de tromper le discriminateur. Dans ce cas, il cherche à maximiser D(G(z)) où $z \sim p_z$.

Ces deux objectifs se traduisent aisément par le problème mathématique suivant :

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

En effet, du point de vue du discriminateur, réussir à distinguer le vrai du faux revient à maximizer D(x) et 1-D(G(z)). Et, du point de vue du générateur, on cherche précisèment à maximiser D(G(z)) pour tromper le discriminateur.

Quand les auteurs introduisent cet objectif, ils expliquent deux choses importantes :

- Il est conseillé d'alterner k entraînement du discriminateur avec 1 entraînement du générateur afin de faire en sorte que le discriminateur soit assez aiguisé pour distinguer le vrai du faux et éviter ainsi **le problème de sur-apprentissage**. En effet, si l'on entraine de façon équilibré les deux modèles, le discriminateur n'étant pas bon au début car non entrainé, risque de laisser entendre rapidement au générateur que ce dernier à réussi à le tromper. La réalité est tout autre : le discriminateur n'étant pas bon initialement pour distiguer le vrai du faux, il est clairement plus facile de le tromper tout en générant du contenu complètement bruité !
- Les auteurs préconisent également de modifier l'objectif défini plus haut étant donné qu'il y a un risque de saturation de ce dernier. En outre, comme le générateur n'est pas bon au début, on s'attend à avoir $D(G(z))$ plutôt faible au départ et donc le terme en $\log(1 - D(G(z))$ risque d'être très proche de zero empechant le bon déroulement de l'entraînement. Pour contrer ce problème, ils proposent simplement de maximiser $\log(D(G(z))$ plutôt que de minimiser $\log(1 - D(G(z))$.

Une chose m'a frappé en lisant l'article. Bien que les auteurs donnent ces précieux conseils, ils ne semblent pas du tout les appliquer dans leurs expériences. En effet, en lisant l'algorithme 1, on se rend compte qu'ils ont parfaitement équilibré l'entraînement du discriminateur avec celui du générateur tout en gardant la même expression de

l'objectif à minimiser/maximiser selon le modèle. Toutefois, même en ne suivant pas leur recommandations, ils ont réussi à obtenir des résultats satisfaisants.

Concernant la phase d'entraînement, elle est tout à fait comparable à une phase d'entraînement classique d'un réseau de neurones. L'idée est d'utiliser l'algorithme de descente du gradient afin de trouver les paramètres permettant le mieux d'arriver à nos fins. Concrètement ici, l'objectif se décompose en un problème de maximisation pour le discriminateur et un problème de minimisation pour le générateur. Ainsi, il y a simplement une alternance entre l'algorithme de montée de gradient et l'algorithme de descente de gradient de telle façon à optimiseer les paramètres respectifs de ces deux modèles selon leurs objectifs respectifs

3 Garantie de l'efficacité d'un réseau de neurones antagonistes génératif

Les auteurs ont montré très simplement que lorsque le générateur est fixé, le discriminateur optimal vérifie l'équation :

$$\boxed{D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}}$$

Ce résultat est assez cohérent avec l'intuition. Notamment, si le générateur à réussi son coup alors $p_g = p_{data}$ et donc le discriminateur optimal renvoie pour n'importe quelle observation x artificielle ou non la probabilité $\frac{1}{2}$. Cela signifie précisèment que le générateur ne sait plus distinguer ce qui est vrai et ce qui est faux. La preuve de ce theorème n'étant pas très compliqué, nous pouvons la rappeller ici :

■ Preuve : Du point de vue du discriminateur, nous devons maximiser la quantité

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

Regardons la plus en détails.

$$\begin{aligned} V(G,D) &= \int_{\boldsymbol{x}} p_{\text{data}}\,(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_{\boldsymbol{z}} p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(G(\boldsymbol{z})))dz \\ &= \int_{\boldsymbol{x}} p_{\text{data}}\,(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx \text{ car lorsque } z \sim p_z,\ D(G(z)) \sim p_g \text{ par définition de } p_g \end{aligned}$$

Maitenant, il se trouve que pour tout $(a,b) \in \mathbb{R}^2 \backslash \{0,0\}$, la fonction f: $y \to a\log(y) + b\log(1-y)$ admet un maximum sur [0,1] en $\frac{a}{a+b}$.

Nous pouvons le voir simplement en dérivant f,

$f'(y) = \frac{a}{y} - \frac{b}{1-y} = \frac{a(1-y) - by}{y(1-y)}$ et cette quantité est positive si et seulement si $y \leq \frac{a}{a+b}$

Ainsi, pour x fixé en identifiant $a = p_{data}(x)$ et $b = p_g(x)$, on obtient le résultat attendu □.

Maintenant, lorsque le discriminateur est enfin le discriminateur optimale, les auteurs ont montré assez simplement que du point de vue du générateur, l'objectif optimale est precisément atteint lorsque $p_g = p_{data}$.

Le coeur de la preuve de ce résultat réside en la réécriture de l'objectif connaissant maintenant D_G^*. Les auteurs ont montré notamment qu'il est possible de l'écrire sous la forme suivante :

$$\begin{aligned} V(G,D^*) &= -\log(4) + KL\left(p_{\text{data}}\,\|\frac{p_{\text{data}}\,+p_g}{2}\right) + KL\left(p_g\|\frac{p_{\text{data}}\,+p_g}{2}\right) \\ &= -\log(4) + 2\cdot JSD\left(p_{\text{data}}\,\|p_g\right) \end{aligned}$$

où KL et JSD sont des métriques permettant de mesurer à quel point deux distributions de probabilités sont proches l'une de l'autre. Elles sont souvent assimilés à des distances entre distributions et sont précisèment minimisées lorsque les deux distributions en leur sein sont égales. D'où le résultat déduit par les auteurs. Nous avons donc maintenant une garantie mathématique du bon fonctionnement de l'algorithme. Cependant, cela ne fonctionne

en pratique que si il y a convergence.

Heureusement pour nous, les auteurs ont prouvé le résultat ci-dessous :

Si les hypothèses suivantes sont vérifiées:

- G et D sont assez complexes compte tenu du problème
- D réussi à atteindre son optimum lorsque G est fixé
- G est correctement actualisé de façon à atteindre son objectif

Alors p_g converge vers p_{data}

La première hypothèse est difficile à traduire mais on peut l'expliquer facilement avec un exemple. Générer les nombres du dataset MNIST est bien plus facile que générer des visages. Ainsi, on s'attend à avoir une architecture plus complexe pour générer des visages que pour générer des nombres. Notamment, si l'on utilise une architecture trop simpliste, il nous sera impossible de générer des visages réalistes. En pratique, les nombres de MNIST peuvent être plutôt bien générer avec des perceptrons multi-couches tandis que pour construire des visages, il faut plutôt se pencher vers des modèles impliquant des convolutions et des déconvolutions comme le DCGAN [**radford2016unsupervised**].

Malheureusement, je n'ai pas très bien compris la preuve de ce résultat mais il me paraît plutôt intuitif.

4 Avantages et inconvénients des réseaux de neurones antagonistes génératifs

Le modèle presenté dans ce papier part d'une idée simple mais pourtant révolutionnaire pour l'époque. L'impléméntation du modèle est également simple et il semble possible d'arriver à des bons résultats assez rapidement pour des datasets de faible complexité comme MNIST. Il y a en fait de nombreux avantages dans ce modèle.

Dans un premier temps, on peut remarquer que l'apprentissage du modèle implique seulement de pouvoir effectuer les rétro-propagation des gradients comme dans tout réseau de neurones. Cela est relativement aisé et beaucoup plus envisageable que de faire de long calculs impliquant des chaînes de Markov comme cela était fait avant d'après l'article. De plus, le modèle est plutôt flexible car nous sommes parfaitement libre sur le design des architectures du discriminateur et du générateur.

Néanmoins, ce genre de modèle génératif est plutôt opaque car nous n'avons aucun moyen d'accéder à une representation explicite de la distribution qui génère les images artificielles. En effet, tout le principe du GAN repose sur la capacité de décoder une variable latente à partir d'un réseau de neurones de telle façon à obtenir un résultat qui a du sens sans pour autant avoir accès à la distribution p_g. D'autre part, comme expliqué précèdemment il est plutôt difficile de trouver un bon équilibrage de D et de G de telle façon à garantir le bon fonctionnement de l'apprentissage.

5 Applications

Dans la vie de tous les jours, ce genre de modèle peut être très utile dans différents cadres. Dans un premier temps, il peut servir à générer de la donnée quand on en manque pour réaliser un algorithme intelligent. En effet, il est courant de voir que des personnes utilisent ce genre de modèle pour artificiellement augmenter leur dataset de façon à pouvoir garantir un meilleur fonctionnement de leur propre algorithme [**antoniou2018data**].

D'autre part, cet outil peut se réveler très utile dans le domaine de l'art et du design afin d'accompagner le processus créatif. Par exemple, on peut imaginer un architecte se servir d'un tel modèle dans le but d'avoir une idée pour le design de son prochain batîment. On peut également imaginer un dessinateur s'inspiré à partir de croquis généré par un GAN. Concrètement, les possibilités sont infinies.

Nous pouvons également parler de l'entreprise Eva Engines qui se sert précisément de ce genre de modèle pour

fournir des mannequins virtuels aux marques de modes, leur permettant de prévoir à l'avance le rendu de leur création et gagner du temps sur leur process.

Conclusion

Comme me l'a fait remarqué notre professeur Mr Laude, à l'aide de TensorFlow il est possible de simuler un Gan ce qui nous permet d'assimiler un peu plus ce concept : Pour cela il suffit de se rendre sur `playground.tensorflow.org`, ou utilisé ce site `https://poloclub.github.io/ganlab/` qui permet de faire toutes sortes de simulation de GAN. Enfin, ce site trés bien fait de tensorflow `https://www.tensorflow.org/tutorials/generative/dcgan` m'a permis d'apprehender les concepts relatifs aux GAN.

Pour conclure, nous avons discuté dans ce rapport du modèle des réseaux de neurones antagonistes génératifs permettant de constuire des images artificielles réalistes. Notamment, nous avons expliqué brièvement son fonctionnement, détaillé quelques aspects mathématiques du modèle et discuté de ses avantages et inconvénients tout en précisant quelques unes de ses applications. Aujourd'hui, ce modèle a été developpé bien plus loin que sa version originelle et continue d'être étudier pour résoudre des problèmes plus en plus complexes. NVIDIA est de loin l'entreprise la plus à jour sur le sujet profitant de ses propres cartes graphiques pour accélérer leurs calculs. Enfin, nous pouvons également rappeler l'existence des deepfakes profitant des prouesses de ce modèle pour tromper notre vigilance. Etant donné ce genre de dérives, il pourrait être intéressant de développer une stratégie permettant de détecter qu'une image réaliste à l'oeil nu a été générée par un algorithme ou non...