TIPE : Création de patients virtuels par réseaux de neurones antagoniste génératif

LECOQ Raphaël Candidat 20738

2022

La médecine actuelle utilise de plus en plus les bases de données médicales pour entraîner des algorithmes d'analyse de données qui peuvent faire de l'**aide à la décision** pour les docteurs. Par exemple, l'analyse de certains brins d'ADN ou d'ARNm d'un individu atteint de cancer du sein permet de donner un indicateur quand aux soins à déployés [3].

Cependant, l'entraînement de ces algorithmes de données nécessite des **populations équilibrées** pour éviter des biais d'analyses ou des surepresentations de certaines caractéristiques, et les données de la santé ne le sont que rarement. La solution actuelle est d'entraîner ces algorithmes en **tronquant** aléatoirement la population majoritaire, mais d'énormes quantités de données sont perdues [9].

Le travail effectué par notre binôme (ROGNON Junior, LECOQ Raphaël) tente alors de pallier les déséquilibres de population sans perdre d'information, en **générant** des nouvelles données indiscernables des données réelles pour qu'elles puissent être analysées comme telles.

Part I
Discriminant

Définition 0.1 *Couche*

$$X \rightarrow \boxed{\text{layer}} \rightarrow Y$$

Une couche est une fonction $f_w : R^i \rightarrow R^j$ *où* w *représente les paramètres de la couche.*

Définition 0.2 *Réseau*

Un réseau est une successions de couches.
La sortie Y *du réseau correspond à la sortie de la dernière couche et est comparée à une sortie attendue* Y^* *en calculant leur distance par la fonction d'erreur* $E(Y, Y^*)$

Définition 0.3 *Descente de gradient*
On minimise une fonction E par rapport à un paramètre w avec l'algorithme de descente de gradient selon

$$w \leftarrow w - \alpha \frac{\partial E}{\partial w}$$

La convergence et la correction de ce procédé est admise dans le cadre de notre TIPE

Propriété 0.3.1 *Rétropropagation*
Soit $X \in R^i$ l'entrée d'une couche quelconque, $Y \in R^j$ sa sortie, E l'erreur.
Connaître pour chaque couche $\frac{\partial E}{\partial X} = \begin{bmatrix} \frac{\partial E}{\partial x_1} & \cdots & \frac{\partial E}{\partial x_i} \end{bmatrix}$ et $\frac{\partial E}{\partial Y} = \begin{bmatrix} \frac{\partial E}{\partial y_1} & \cdots & \frac{\partial E}{\partial x_j} \end{bmatrix}$ permet d'appliquer l'algorithme de descente de gradient à toutes les couches.

Définition 0.4 *En notant $W = \begin{bmatrix} w_{1,1} & \cdots & w_{1,j} \\ \vdots & \ddots & \vdots \\ w_{i,1} & \cdots & w_{i,j} \end{bmatrix}$ et $B \in R^j$ où X, B sont les paramètres de la couche.*

Couche totalement connectée : $Y = XW + B = \begin{bmatrix} \sum_i x_i \times w_{i1} + b_1 & \cdots & \sum_i x_i \times w_{ij} + b_j \end{bmatrix}$.

Couche d'activation : $Y = f(X) = \begin{bmatrix} f(x_1) & \cdots & f(x_i) \end{bmatrix}$ où f est une fonction non linéaire réelle.

On utilisera par la suite la fonction sigmoid $\sigma : x \in R \to \frac{1}{1+e^{-x}} \in [0; 1]$ qui s'interprète comme une probabilité.

Définition 0.5 *Discriminant*
Un discriminant est un réseau neuronal qui prend en entrée une donnée qui peut appartenir à n classes de données disjointes et renvoie les probabilités de l'entrée d'appartenir à chaque classe.

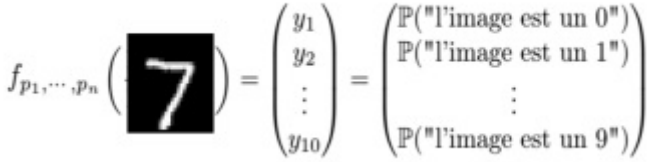


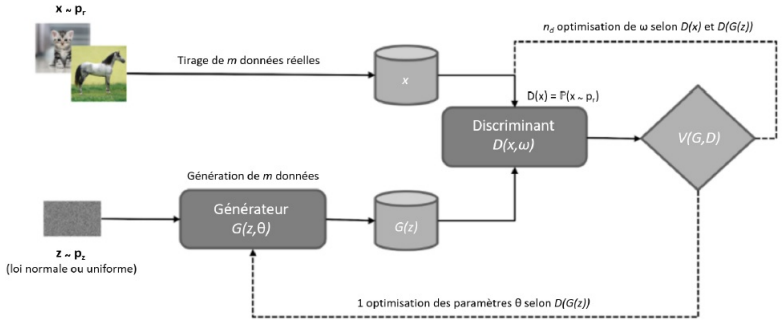
Figure 1: Discriminant appliqué à un chiffre appartenant à la classe 7 et sa sortie

Ici, $Y^* = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$ (1 sur la colonne de la classe 7, 0 ailleurs).
Source principale : [8]

Part II
Réseau de neurone antagoniste génératif

1 Principe

Définition 1.1 *Schéma de principe*



[1]

On entraîne un **Discriminant** défini comme précédement à reconnaître si des données "réelles" sont réelles ou non. On connecte d'autre part un **Générateur** qui prend en entrée un bruit et renvoie une image générée G(z). Le générateur apprend de D(G(z)) la probabilité que G(z) soit réel et de $V(G, D)$ pour optimiser ses paramètres θ.

Fonction de perte : $V(G, D) = E_{x \sim p_{data}(x)}[logD(x)] + E_{x \sim p_z(z)}[log(1 - D(G(z)))] \leq 0$

Le discriminant **maximise** cette expression car il cherche à faire tendre la probabilité qu'une donnée réelle soit réelle vers 1 et la probabilité qu'une donnée générée soit générée vers 0, tandis que le Générateur **minimise** cette expression vers $-\infty$ en cherchant à faire l'inverse:

$$\min_G \max_D V(D,G) = \underbrace{\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}}_{\text{Vraies données}} [\log \overbrace{D(\boldsymbol{x})}^{\uparrow 1}] + \underbrace{\mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}}_{\text{Fausses données}} [\log(1 - \overbrace{D(G(\boldsymbol{z}))}^{\downarrow 0})]$$

[5]

$$\min_G \max_D V(D,G) = \underbrace{\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}}_{\text{Vraies données}} [\log \overbrace{D(\boldsymbol{x})}^{\downarrow 0}] + \underbrace{\mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}}_{\text{Fausses données}} [\log(1 - \overbrace{D(G(\boldsymbol{z}))}^{\uparrow 1})]$$

[5]

L'algorithme calcule alors $\min_G \max_D E_{x \sim p_{data}(x)}[logD(x)] + E_{x \sim p_z(z)}[log(1 - D(G(z))]$

2 Correction de l'algorithme

[4],[2]

Théorème 2.1 *Théorème de transfert généralisé, ADMIS*
$(\Omega, \mathcal{T}, P)$ un espace probabilisé, $X : \Omega \to R$ de loi notée P_X.
Soit $\phi : R \to R_+$ mesurable.
Alors $\int_\Omega \phi(X(\omega))dP(w) = \int_R \phi(x)dP_X(x)$

Corollaire 2.1.1 *$E(\phi(X)) = \int_R \phi(x)f(x)dx$ (existence de V(G,D) et démonstration de **Théorème 2.2**)*

On note G le Générateur et D le Discriminant, p_r la densité de probabilité des données réelles, p_g la densité probabilité des données générées par G.
Par ce qui précède, $V(G, D)$ existe et à G fixé, on cherche D^*_G le discriminant optimal (qui maximise la fonction $D \mapsto V(G, D)$).

Théorème 2.2 $D^*_G(x) = \frac{p_r(x)}{p_g(x)+p_r(x))}$

On pose $C(G) = V(G, D^*_G)$. Le but pour le générateur est donc de trouver : $\min\limits_G C(G)$

Définition 2.1 *Divergence de Kullblack-Leibler*
Soit p et q deux densités de probabilités. on définit : $D(p \mid\mid q) = \int_x p(x) \cdot log(\frac{p(x)}{q(x)})dx$

Caractérise la "distance" entre p et q, mais n'est cependant pas une distance (car non symétrique), éventuellement infini (problématique pour un algorithme).

Propriété 2.1.1 *:*

- $D(p \mid\mid q) \geq 0$
- $D(p \mid\mid q) = 0$ *ssi* $p=q$

Définition 2.2 *Divergence de Jensen-Shannon*
Pour p et q des densités de probabilités, on pose $JS(p \mid\mid q)$: $JS(p \mid\mid q) = \frac{1}{2} \cdot D(p \mid\mid \frac{p+q}{2}) + \frac{1}{2} \cdot D(q \mid\mid \frac{p+q}{2})$

Propriété 2.2.1 *:*

- *$JS(p \mid\mid q)$ est fini donc toujours défini (solution au problème d'existence de $D(p \mid\mid q)$).*
- *$JS(p \mid\mid q)$ est une pseudo-distance.*

Théorème 2.3 *Expression de C(G)*
$C(G) = 2 \cdot JS(p_r \mid\mid p_g) - log(4)$

Théorème 2.4 *Convergence de V(G,D), admise*
Si G et D ont suffisamment de capacité, et qu'à chaque étape de l'algorithme, le discriminant atteint sont optimum étant donné G, et p_g est mis à jour pour améliorer le critère :

$$E_{x \sim p_r}[logD^*_G(x)] + E_{x \sim p_g}[log1 - D^*_G(x)]$$

alors V(G,D) tend vers $\min\limits_G \max\limits_D V(G, D)$

Correction :
Par **Théorème 2.3**, minimiser C(G) c'est minimiser Jensen-Shannon.
Donc minimiser C(G) c'est minimiser Kullblack-Leibler par **Définition 2.2**.
Or minimiser Kullblack-Leibler c'est obtenir $D(p \mid\mid q) = 0$ donc p=q par **Propriété 2.2.1**.
Donc quand l'algorithme optimise ses paramètres (**Théorème 2.4**) pour obtenir $\min_G \max_D V(G, D)$, il obtient bien $p_r = p_g$ et on remarque $D(p_r) = D(p_g) = \frac{1}{2}$ par **Théorème 2.2**.

3 Création de patients virtuels

Données : 20 séquences mutées de la protéine P53 responsable de la destruction des tumeurs chez l'être humain et 20 séquences non mutées caractérisés par 1279 bases azotés trouvées sur la bibliothèque nationale médicale américaine NIH [6].

Représentation des bases azotées :
"A" $\sim \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}$
"T" $\sim \begin{bmatrix}0 & 1 & 0 & 0\end{bmatrix}$
"C" $\sim \begin{bmatrix}0 & 0 & 1 & 0\end{bmatrix}$
"G" $\sim \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}$
On aplatie la liste pour créer un vecteur de taille 1279 * 4

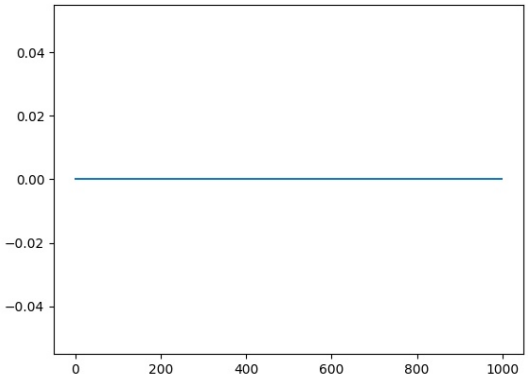


Figure 2: Pourcentage de bonne réponses d'un discriminant seul selon le nombre d'itérations

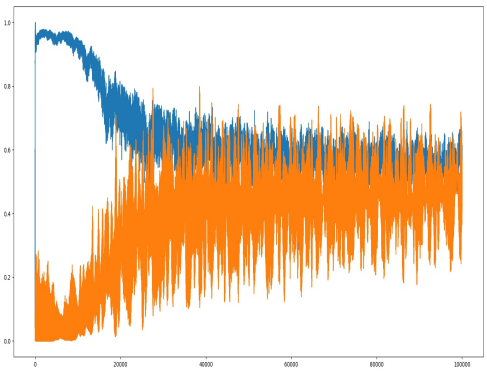


Figure 3: En bleu, score du discriminant sur l'ARNm de la base médicale. En rouge, score des ARNm générées

Figure 1 : Justifie l'utilisation de fonctions d'activation non linéaire.
Figure 2 : On remarque $P(p_r) = P(p_g) = \frac{1}{2}$ ce qui signifie d'après **Théorème 2.3** que la densité de donnée réelle est reproduite par la densité de donnée générée.

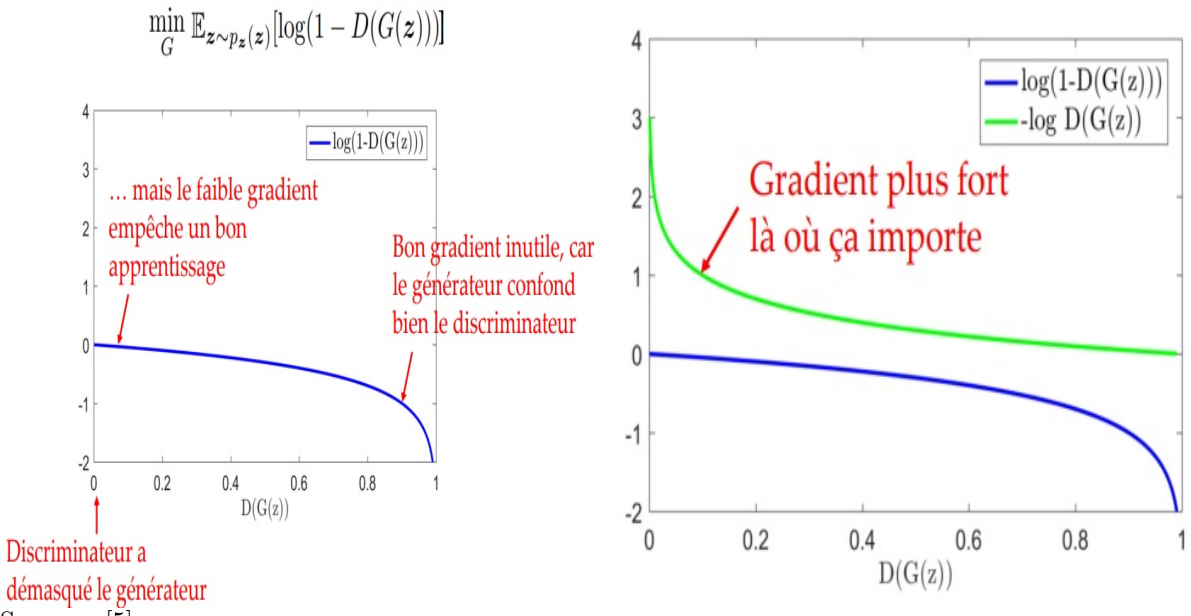
Lorsque l'on compare les données générées dans la bibliothèque de séquence ADN grâce à l'outil BLAST, on obtient **99,7%** de similarité entre les donnée réelles et générées : nous avons bien réussi à générer de faux patients qui présentent une mutation (ou non) de la protéine P53.

Part III
Limites

[4],[5]

- **Le discriminant ne reconnaît pas les données réelles** : le générateur apprend à reproduire les mauvaises données
- **Gradient Vanishing** : Si le discriminant est trop efficace, $P(p_r) = 1$, $P(p_g) = 0$, tout est constant, disparition de gradient et aucun apprentissage
- **Mode collapse** : Le générateur apprend à produire une seule image qui trompe systématiquement le discriminant mais ne reproduit pas la densité de donnée
- **Instabilité** voir non convergence du min max

Le problème de **convergence** du min max peut-être en partie résolu par une modification de V(G,D). Plutôt que de chercher $\min_G E_{x\sim p_z} log[1 - D(G(z))]$ qui présente un faible gradient en D(G(z)) = 0 (là où le Discriminant a démasqué le générateur), on fait une montée de gradient sur $\max_G E_{x\sim p_z} log[D(G(z))]$



Source : [5]

Cette optimisation résout une grande partie des problèmes de **Gradient Vanishing**, mais n'est cependant pas suffisante pour gérer des cas de non convergence dûent à la définition même de la distance de Jensen-Shannon.

Mode collapse :
Application du réseau sur diverses portraits de moi où les expressions faciales sont variées.

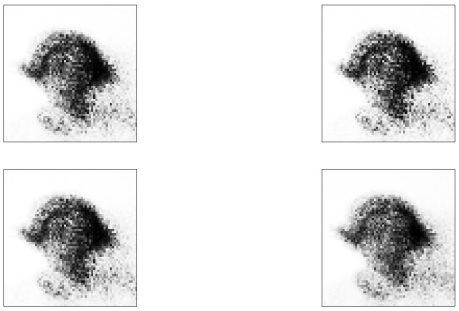


Figure 4: Unique image reproduite par le générateur

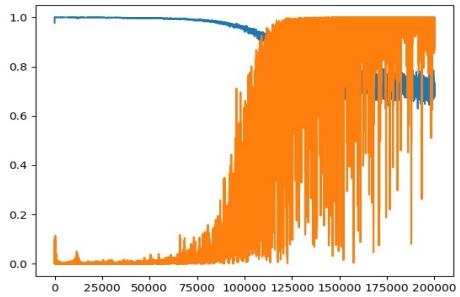


Figure 5: Le générateur trompe systématique le discriminant

Une unique image générée est apprise par coeur et trompe systématiquement le discriminant sans reproduire la distribution des données réelles.
Le générateur ne reproduit alors absolument pas la distribution des données réelles : il s'agit du mode collapse.

Un changement de fonction d'erreur avec la distance dite de **Wasserstein** qui résout les problèmes de convergence du du **min max** et de **Gradient Vanishing** fait l'objet du TIPE de mon binôme ROGNON Junior.

Part IV
Conclusion

Nous avons bien réussi à produire des patients virtuels qui pourraient être utilisés pour l'entraînement des algorithmes d'analyse de données, cependant le GAN évolue déjà vers d'autres variantes plus efficaces (Wassersein-GAN) qui résolvent certaines de ses faiblesses et qui pourraient être appliqué à des données beaucoup plus complexes dont reproduire des séquences bien plus pertinentes qu'un minuscule brin d'ARNm.

References

[1] Quantmetry Aurelia Nègre. DeepFakes et autres générations... que se cache-t-il derrière les GANs, le prochain game-changer de l'IA ? Technical report.

[2] Francis Bach. Entropie, KL, MV, Familles exponentielles.

[3] Fondation de recherche médicale. Tout savoir sur le cancer du sein.

[4] Ian J. Goodfellow, Jean Pouget-Abadie, Bing Xu Mehdi Mirza, David Warde-Farley, Sherjil Ozair†, and Yoshua Bengio Aaron Courville. Generative adversarial nets. *Departement d'informatique et de recherche opérationnelle ´*, 1, 2014.

[5] Philipe Guiguère. GANs : Generative Adversarial Networks. Université de Laval.

[6] https://www.nih.gov/. National Institute of Health.

[7] YouTube Luis Serano. A friendly introduction to GANs.

[8] Medium.com. Mathématiques des réseaux de neurones.

[9] Quantmetry. Classification et déséquilibre de classes.