

---

# Mémoire de Fin d'étude MDS

**Université de Lille, I.A.E.**

Implémentation d'un moteur de recommandation collaboratif exploratoire utilisant les réseaux de neurones génératif par adversité afin de limiter les effets bulles.

Produit par : **Bucamp Axe** - Dirigé par : **Dominique Crié** - 6 juillet 2022

---



# Remerciements :

Je remercie l'ensemble des personnes ayant participé de manière positive à la construction de la personne que je suis et du mémoire que j'ai construit. Par le biais de leur simple présence, de leur aide, ou de leurs recommandations avisées.

Je remercie particulièrement l'ensemble des professeurs de ce Master MDS de l'IAE de Lille, car ces derniers sont les professeurs qui ont participé à la création et consolidation de ma culture Marketing et Data qui me permet aujourd'hui d'achever ce travail dont j'aurais cru il fut un temps être incapable et cela malgré les temps difficiles de la période du covid-19 et des nombreux confinements qui ont entravé notre apprentissage en tant qu'étudiant de ce master.

Aussi, je remercie très chaleureusement Gérald DHERBOMEZ, Rochdi MERZOUKI ainsi que tous les membres du laboratoire CRISTAL qui m'ont permis lors de ma découverte des réseaux de neurones et du machine learning de commencer mon initiation à ces technologies par le biais d'un CDD et d'un stage post licence et de me donner l'envie de poursuivre dans l'apprentissage de ces dernières dans mon master actuel.

Sans oublier bien sûr ceux qui comme mon tuteur, Monsieur Dominique Crié, ont accepté d'être harcelé par mes questions incessantes sur des parties parfois techniques de l'ensemble des idées qui me passer par la tête concernant l'exercice de création du mémoire comme : Monsieur Mihai CALCIU, Madame Sophie LACOSTE-BADIE, Monsieur Antoine AYACHE, Monsieur Julien HARMONIER et une bonne partie de ma famille ainsi que certains de mes amis.

Ceux qui m'ont aidés dans la mise en forme et la correction de mes nombreuses (très nombreuses) fautes et ayant donc pris du temps pour se faire. Ma mère, ma sœur ainsi que Madame Sophie LACOSTE-BADIE et Monsieur Dominique CRIE.

Je tiens à rajouter une ligne pour tous ces petits métiers bien trop peu estimés et pourtant bien souvent d'une importance capital.

Et finalement, vous, Chères Lecteurs, qui par intérêt, obligation ou par sens de l'exploration vous apprêtez à consulter ce mémoire.

# Table Des Matières :

Introduction :	4
Coeur du mémoire :	7
Systèmes de recommandations déclinaisons :	7
Place du moteur de recommandation dans le marketing :	14
Effet de biais dans la recommandation :	17
Qu'est ce que la créativité, la nouveauté et comment sortir des effets de bulles :	22
Intelligence Artificielle et G.A.N., les Réseaux de neurones créatifs combinatoires et exploratoires :	26
Qu'est que l'intelligence artificielle ?	28
Définition et introduction :	28
Aux origines des réseaux neuronaux : Cas du Perceptron	29
Aux origines des réseaux neuronaux : Cas du multilayer perceptron	31
Evolution vers les Réseaux de neurones et le Deep learning :	34
Les réseaux Créatifs, « Generative adversarial Network » :	35
Réseaux Hybride GAN et Supervisé, PixToPix :	38
Descent de Gradient, Back propagation et Adam optimizer	44
Dataset et Preprocessing, Quelles données utiliser ?	47
Mise en place de l'algorithme :	49
Entrainement :	52
Dataset avec moyenne recentré :	52
Dataset avec uniquement un bruit Gausien :	54
Résultat :	55
Conclusion :	61
Voies de recherche :	64
Bibliographie :	65
ANNEXES :	69

## Introduction :

Les algorithmes de recommandations sont actuellement de par la masse d'informations à gérer une pierre angulaire de notre parcours sur l'internet. Ces derniers ont de très nombreuses déclinaisons selon la particularité technique et les données qu'ils utilisent. À l'heure actuelle, personne ne peut dire qu'il n'est pas influencé par ces derniers dans chacun des nombreux choix qu'il doit faire en ligne. Nous servant de guide, ils nous mènent vers ce qui vraisemblablement selon leurs calculs semble être approprié à nos préférences utilisateurs, or bien que très utile, il se trouve que ces derniers sont soumis à des biais. Impossible pour l'algorithme à la fois d'avoir une représentation parfaite de qui nous sommes, mais aussi de comprendre à partir de cela quels seront les meilleures recommandations. Ces notions subjectives, soumises à une constante évolution, sont effectivement compliquées à interpréter pour de "froids" calculs mathématiques. Ces biais, analyses erronées, filtre par lequel l'algorithme prend des décisions, peuvent entraîner les recommandations dans de mauvaises directions. Parfois, cela est acceptable et l'algorithme prenant de toute manière suffisamment de bonnes décisions, il est préférable de ne les traiter qu'indirectement, mais parfois leurs conséquences sont bien plus importante. En effet les recommandations étant en grande partie responsables du comportement utilisateur, si ces dernières guident dans de mauvaises directions cet utilisateur alors non seulement l'expérience de ce dernier peut être désagréable, mais de plus, c'est bien l'ensemble de son comportement qui peut être déformé. C'est le cas dans le biais menant à la construction des «Filter Bubble» ou «Echo chamber», bulles filtrantes dans lesquelles un utilisateur ayant interagi avec une recommandation d'un ensemble X va être fortement soumis à des recommandations venant de ce même ensemble. Dès lors, il n'y a alors plus que très peu de chance que l'utilisateur sorte de cet ensemble X qui petit à petit le définira de plus en tant qu'utilisateur. Ces dernières sont discrètes et pourtant, leurs conséquences sont grandes, comme à l'heure actuelle le démontre leur rôle dans la consommation et la crédence des utilisateurs envers les « fake news ». Mais dès lors comment produire un algorithme qui aurait la capacité de s'en dégager naturellement, qui limiterait la probabilité que X mène à X. Et bien pour cela cet article se base sur l'idée de la créativité, de ces déclinaisons et des différentes manières dont elle s'exprime afin d'en tirer le concept de la créativité exploratoire dans laquelle il est peu probable que X mène à X car dans cette dernière tout est une nouvelle découverte, un nouveau concept jusqu'alors inexploré au sens spatial du terme, tel un explorateur découvrant une nouvelle carte. Dès lors, il s'agit de parcourir les différents algorithmes créatifs et d'en tirer un capable de s'exercer à cette tâche. Afin de ne pas complètement laisser la recommandation prendre des choix presque

hasardeux, l'algorithme ici ne fonctionnera qu'en partie de manière exploratoire, en fait assez indirectement, mais suffisamment pour que sans aléatoire l'algorithme pousse l'utilisateur naturellement à la découverte de nouvelles expériences. C'est ainsi qu'est venu l'idée de l'utilisation des modèles GAN, réseaux de neurones créatifs capables tel un artiste de peindre de nouvelles œuvres, trouver de nouveaux concepts, concevoir de nouvelles musiques, etc. Et dont, plus particulièrement, le modèle «PIX2PIX» à la fois GAN et supervisé est un bon mélange entre créativité combinatoire et exploratoire permettant d'apprendre des deux concepts.

### **Question de recherche établit :**

Comment se définissent les effets de bulles et quel sont leurs manières de s'exprimer en terme algorithmique ?

Puis, comment quantifier ces biais ?

Et finalement, quels mécanismes permettraient d'en sortir et comment l'exprimer dans notre algorithme ?

### **Hypothèse à vérifier :**

- 1) Notre algorithme résout les effets de bulles filtrantes.

Avec :

$H_0$  : oui, l'algorithme évite des effets de bulles filtrantes ou dit d'echo chamber.

$H_1$  : non, il n'est pas vérifié que l'algorithme évite des bulles filtrantes.

- 2) Notre algorithme est pertinent dans ses recommandations.

Avec :

$H_0$  : oui, l'algorithme a une bonne précision dans la recommandation.

$H_1$  : non, l'algorithme n'a pas une bonne précision dans la recommandation.

- 3) Notre algorithme permet une grande diversité inter recommandation.

Avec :

$H_0$  : oui, l'algorithme diversifie suffisamment les contenus recommandés.

$H_1$  : non, l'algorithme ne diversifie pas suffisamment les contenus recommandés.

### **Liens vers les ressources git-hub :**

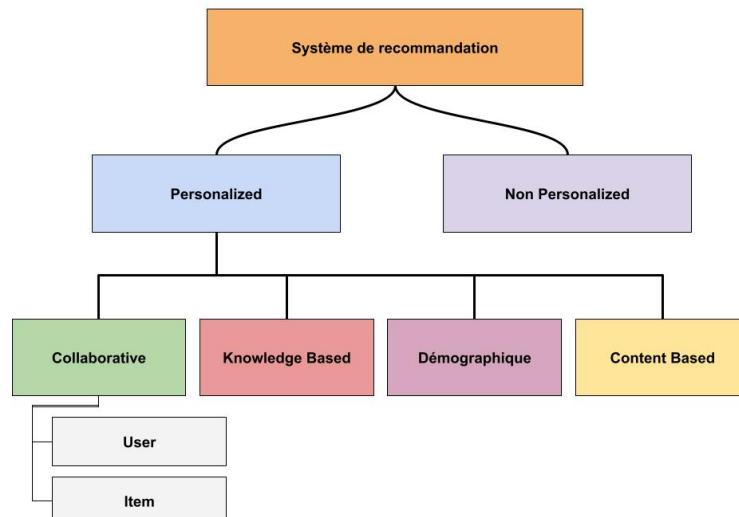
<https://github.com/Axle-Bucamp/Recommandation-System-Gan>

## Coeur du mémoire :

### Systèmes de recommandations déclinaisons :

Désormais, à l'heure d'internet, du big data et de l'intelligence artificielle, nous nous habituons de plus en plus à être inondées par la masse d'informations **Calciu, M., Salerno, F., & Moulin, J. L. (2016)**<sup>1</sup>. Afin de nous aider à trouver notre chemin parmi cette masse de données quotidiennes, il existe ce que l'on appelle des algorithmes de recommandations. Ces derniers servent alors principalement de guide et d'influenceur. Ils nous permettent d'optimiser nos recherches en filtrant les informations qui nous seront présentées. Pour ce faire, ils se basent sur une proximité de l'information filtrée avec notre requête ou avec notre personnalité, nos goûts ou encore nos besoins du moment. **(Das, D., Sahoo, L., & Datta, S. (2017))**<sup>2</sup>.

Pour ce faire, ces algorithmes se basent bien souvent sur des données personnelles, ou à minima la dernière requête que nous avons effectuée. Il existe alors plusieurs familles d'algorithmes de recommandation.



(basé sur la figure 1 de l'article : Das, D., Sahoo, L., & Datta, S. (2017) A Survey on Recommendation System.)

<sup>1</sup> **Calciu, M., Salerno, F., & Moulin, J. L. (2016).** Les calculs sur données-client massives sont-ils trop importants pour être laissés aux analystes marketing?. *Statistique et Société*, 4(3), 37-48.

<sup>2</sup> **Das, D., Sahoo, L., & Datta, S. (2017).** A survey on recommendation system. *International Journal of Computer Applications*, 160(7).

### **Parmi ces derniers il y a :**

- Les algorithmes collaboratifs.
- Les algorithmes basés sur le contenu.
- Les algorithmes plutôt démographiques.
- Les algorithmes basés sur un savoir acquis.

(Das, D., Sahoo, L., & Datta, S. (2017)<sup>3</sup>

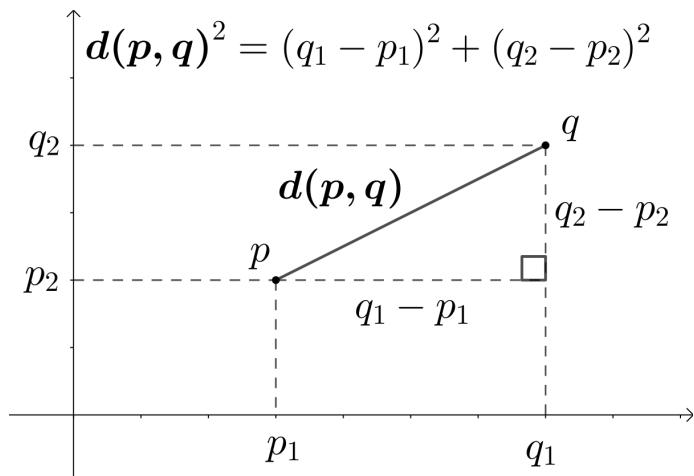
Les algorithmes collaboratifs, prennent des décisions en se basant sur les choix d'un ensemble d'utilisateurs. Ceux-ci ne se contentent pas d'apprendre à vous connaître, ils utilisent une méthode complémentaire en fait assez simple qui permet de savoir à tous les coups quels produits pourraient vous intéresser sans pour autant trop connaître ces derniers. En fait, ils utilisent simplement la comparaison de votre comportement en tant qu'utilisateur avec celui d'autres utilisateurs, si ces derniers sont proches de votre comportement alors, l'algorithme a trouvé quelqu'un qui pourra vous guider lors de son prochain choix.

Pour ce faire, ces algorithmes utilisent couramment les différents votes par item des utilisateurs. On peut alors calculer la proximité d'un utilisateur aux autres et donc de même de la proximité d'un item selon les utilisateurs proches ce qui nous donne dès lors un item recommandé par des gens ayant un comportement similaire. Il peut s'agir ici que d'un simple calcul de distance euclidienne entre chaque utilisateur selon les votes par item. Les utilisateurs proches identifiés, ayant la distance euclidienne la plus faible, peuvent alors recommander de nouveaux produits. Afin d'identifier les nouveaux produits à sélectionner, on utilise une estimation de la note que pourrait donner l'utilisateur donné selon les utilisateurs dit proche. Sois par simple calcul de probabilité (somme des rating des utilisateurs proches/nombre d'utilisateurs) ou par d'autres méthodes prédictives pouvant alors utiliser la distance euclidienne comme poids ou non. Ici, aucune connaissance de l'item n'est nécessaire autre que la perception de celui-ci par les utilisateurs via une notation. Gao, T., Cheng, B., Chen, J., & Chen, M. (2017)<sup>4</sup>.

---

<sup>3</sup> **Das, D., Sahoo, L., & Datta, S. (2017).** A survey on recommendation system. *International Journal of Computer Applications*, 160(7).

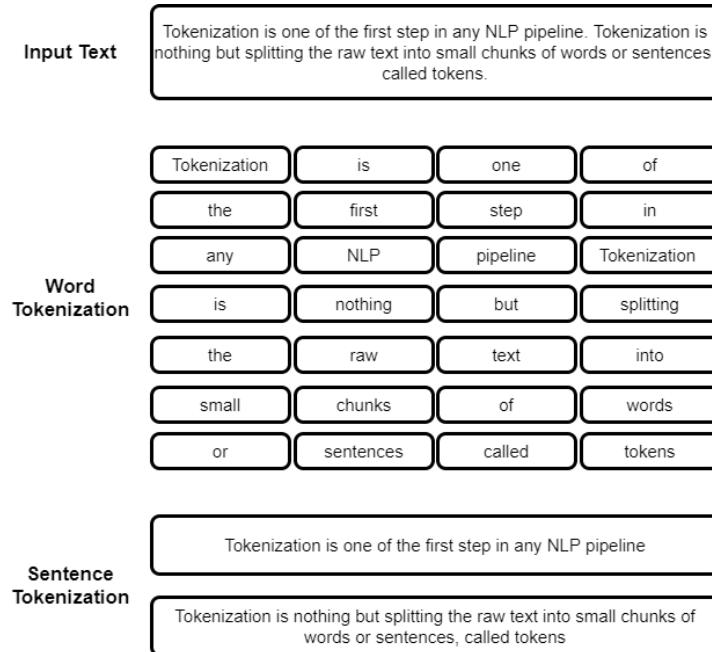
<sup>4</sup> **Gao, T., Cheng, B., Chen, J., & Chen, M. (2017).** Enhancing collaborative filtering via topic model integrated uniform Euclidean distance. *China Communications*, 14(11), 48-58.



### Calcul de distance euclidienne

En ce qui concerne les algorithmes basés sur le contenu, ils vont tendre à recommander des produits proches entre eux ou proches de votre requête. Par exemple, si vous cliquez sur un film romantique, italien avec un acteur nommé mister V. alors il est probable que l'on vous recommande un film du même acteur, romantique aussi, mais cette fois disons espagnol. De même, si vous rentrez dans la barre de recherche les mots "œuf de Pâques, chocolat au lait" alors l'algorithme cherchera parmi les produits ceux possédant une description proche de votre requête.

Une méthode fréquemment utilisée permettant de procéder ainsi est d'utiliser la lecture du texte descriptif du produit et de la requête utilisateur. De tokeniser ces derniers au niveau du mots, manière de séparer les mots de la phrase un à un, puis de les encoder afin de garder uniquement les mots importants et de leur donner un identifiant unique binaire. Palmer, D. D. (2000)<sup>5</sup>.



Puis d'utiliser l'algorithme TF-IDF qui permet de calculer un poids selon la fréquence d'un mot parmi les éléments d'une phrase.

$$w_{i,j} = tf_{i,j} \times \log \left( \frac{N}{df_i} \right)$$

$tf_{i,j}$  = number of occurrences of  $i$  in  $j$   
 $df_i$  = number of documents containing  $i$   
 $N$  = total number of documents

Cela permettant dès lors d'obtenir un premier score par mot par phrase selon les mots de la requête utilisateur par rapport à la description des divers produits. Aizawa, A. (2003)<sup>6</sup>.

<sup>5</sup> Palmer, D. D. (2000). Tokenisation and sentence segmentation. Handbook of natural language processing, 11-35.

<sup>6</sup> Aizawa, A. (2003). An information-theoretic perspective of tf-idf measures. Information Processing & Management, 39(1), 45-65.

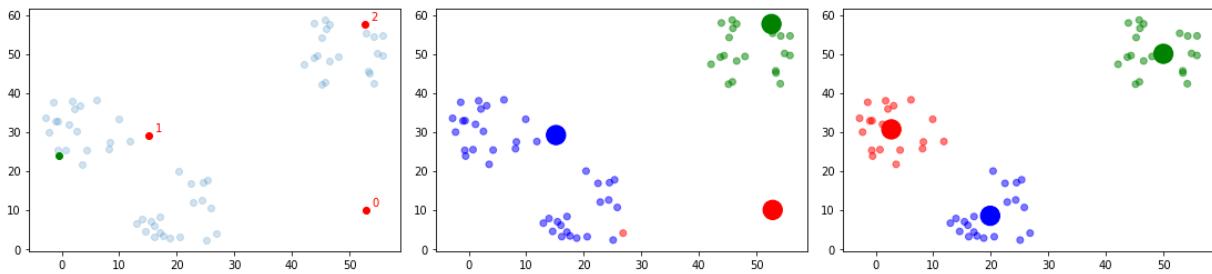
Enfin via un algorithme de calcul de distance spatiale comme la distance euclidienne qui a l'avantage d'être toujours vrai même lorsque de multiples dimensions sont utilisées, nous pouvons alors estimer selon ses scores la proximité d'une requête ou une description par rapport à d'autres nous donnant ainsi un ordre de produit selon cette proximité.

Ceux démographiques, eux, prennent en compte le fait que parfois des produits sont simplement plus ou moins appréciés selon les lieux ou les groupes d'utilisateurs. En effet, ici, le but va être de définir un groupe dans lequel vous appartenez et de trouver ce qui semble approprié à l'identité de ce groupe. Par exemple, un homme va préférer acheter un ballon de foot, généralement, à un tutu de danse bien qu'évidemment, cela lui reste accessible.

Il est possible d'utiliser diverses méthodes de clustering, méthodes faites pour mettre en liens des observations selon leurs ressemblances, ressemblances définies par une règle donnée. Par exemple un équilibre de la variance de la distance intra groupe comme dans l'algorithme des K-means, ce qui signifie que le but de l'algorithme est d'agréger les observations dans K-groupe de manière à obtenir une variance de la distance entre les différents groupes trouvés équilibrés. Pour ce faire, l'algorithme place un centre par cluster dans un premier temps, puis, agrège les données selon le centre le plus proche. Une fois, cela fait, il faut ajuster l'emplacement de chaque centroïde en utilisant comme nouvelle coordonnée du centre de chaque groupe la moyenne des coordonnées des observations agrégées. Ainsi itération après itération les données seront naturellement rassemblées autour des k-centre placés. Cela permet bien souvent d'associer naturellement un ensemble de comportements à un groupe d'observation. Dès lors, les recommandations faites au groupe seront celles liées au comportement du groupe identifié comme une fréquence d'achat, de haute note, etc. On pourra par la suite utiliser une méthode prédictive pour catégoriser les nouveaux utilisateurs dans les segments identifiés. Lu, M., Qin, Z., Cao, Y., Liu, Z., & Wang, M. (2014)<sup>7</sup>.

---

<sup>7</sup> Lu, M., Qin, Z., Cao, Y., Liu, Z., & Wang, M. (2014). Scalable news recommendation using multi-dimensional similarity and Jaccard-Kmeans clustering. *Journal of Systems and Software*, 95, 242-251.



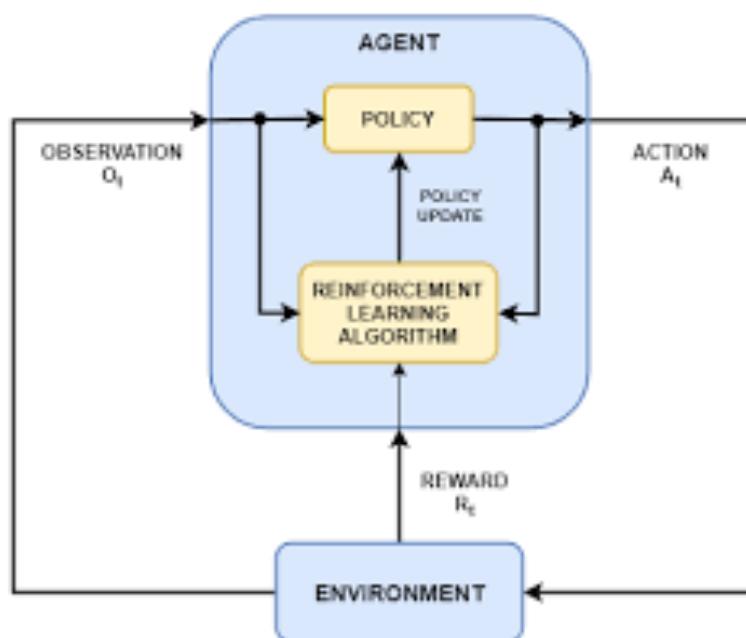
Exemple de clustering K-mean avec  $k=3$  groupes en trois itérations, Les centroïdes de chaque groupe étant les points plus large.

Ceux basés sur un savoir acquis, quant à eux, sont sûrement les plus intuitifs d'entre tous. Ils prennent en compte des règles que l'on a apprises des clients au fur et à mesure telle que l'achat de cet ordinateur correspond bien à l'achat d'une nouvelle souris. Dès lors comme vous vous en doutez, il est très dur de définir toutes les règles qui régissent l'ensemble de ces systèmes, mais ils sont simples et brutalement efficaces. En effet, cette logique imparable permet de facilement interpréter des nouveaux besoins utilisateurs possibles et donc de répondre rapidement et efficacement à ces derniers.

Ici, il peut s'agir de règles métiers connues associé à un comportement logique vis-à-vis d'un item ou d'un utilisateur, par exemple des parfums conçus pour les jeunes femmes ne doivent pas être recommandé à des vieux hommes si celui-ci ne le demande pas spécifiquement. Ou alors une souris d'ordinateur convient bien après l'achat d'un ordinateur. Etc.

Il existe aussi des algorithmes hybrides capables de traiter plusieurs types de données, soit car ils sont un ensemble de méthodes, soit car il s'agit de réseaux de neurones profond généralisées capables d'apprendre des données en toute circonstance et s'améliorant alors communément selon l'expérience apportée par chaque tâche en suivant une règle précise telle que une augmentation des bénéfices, une maximisation du nombre d'interactions des utilisateurs et de leur durée ou encore une augmentation de la satisfaction globale des utilisateurs dans les notations produits les trois règles peuvent d'ailleurs chacune avoir leur poids dans le cadre d'apprentissage par renforcement.

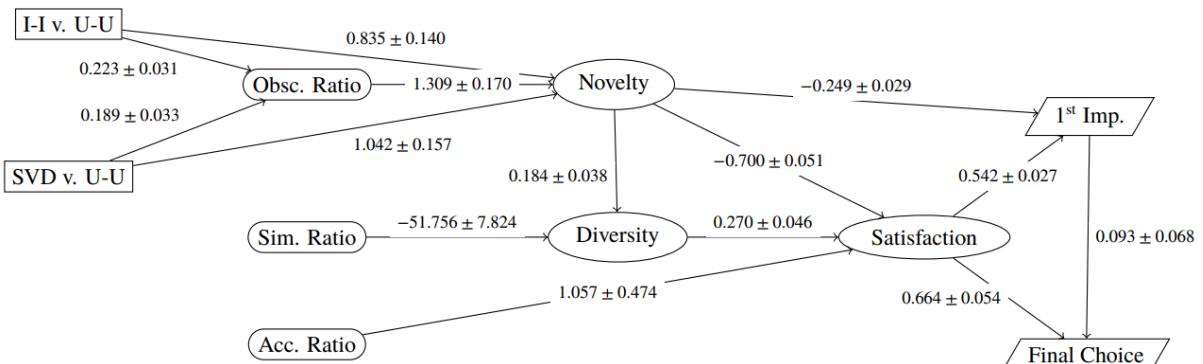
Ainsi, dans ce dernier, l'algorithme se met à jour selon un système de récompense suivant une action qu'il effectue dans un environnement donné. L'algorithme renforce alors la probabilité d'une recommandation dans un contexte défini selon la récompense qu'il obtient, l'utilisateur a-t-il cliqué ? combien de fois ? quel est la valeur de son panier ? etc. De même, ces algorithmes prennent en compte la future et past reward afin de parfois explorer des possibilités moins récompensantes à priori mais dans le future l'étant beaucoup plus. Ces algorithmes peuvent être étonnamment performants dans la finalité, mais nécessitent bien souvent un plus long entraînement. (**Warlop, R. (2018)**<sup>8</sup>)



<sup>8</sup> **Warlop, R. (2018).** Nouveaux algorithmes et méthodes d'exploration-exploitation pour des systèmes de recommandations efficaces.

## Place du moteur de recommandation dans le marketing :

Dans le marketing, ces moteurs de recommandations jouent actuellement un rôle clef dans la digitalisation. En effet, ils permettent à la fois d'augmenter les ventes et la satisfaction du client Jiang, Y., Shang, J., & Liu, Y. (2010).<sup>9</sup> D'ailleurs, il est recommandé de les utiliser de manière à optimiser cette satisfaction plutôt que les ventes directement, celle-ci permettant alors d'augmenter la fidélisation du client ainsi que son image de l'entreprise et donc sa « life time value », rapport durée d'activité des échanges clients/entreprise probables par rapport au bénéfice qu'il engendre. Cela nous démontrant qu'une bonne satisfaction est égale à de plus grands futurs bénéfices dans cet échange. Ho, T. H., Park, Y. H., & Zhou, Y. P. (2006).<sup>10</sup> Ainsi, il est important de comprendre les relations entre un système de recommandation et la perception de ce dernier par l'utilisateur afin de bien comprendre comment maximiser la future value et non directement calculer les bénéfices que ceux-ci apportent. Pour ce faire, des chercheurs en marketing ont cherché à interpréter les relations entre cette perception subjective et différents types de



recommandations. On obtient alors ce genre de modèle :

Ekstrand, M. D., Harper, F. M., Willemsen, M. C., & Konstan, J. A.<sup>11</sup>

<sup>9</sup> Jiang, Y., Shang, J., & Liu, Y. (2010). Maximizing customer satisfaction through an online recommendation system: A novel associative classification model. *Decision Support Systems*, 48(3), 470-479.

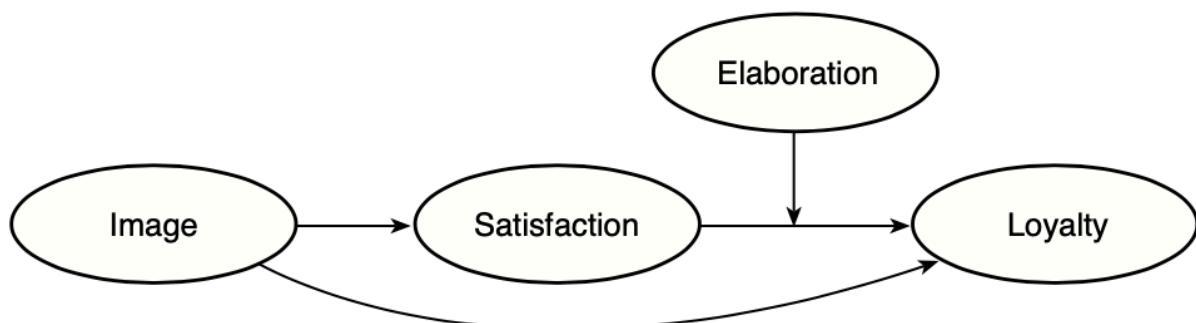
<sup>10</sup> Ho, T. H., Park, Y. H., & Zhou, Y. P. (2006). Incorporating satisfaction into customer value analysis: Optimal investment in lifetime value. *Marketing Science*, 25(3), 260-277.

<sup>11</sup> Ekstrand, M. D., Harper, F. M., Willemsen, M. C., & Konstan, J. A. (2014, October). User perception of differences in recommender algorithms. In *Proceedings of the 8th ACM Conference on Recommender systems* (pp. 161-168).

Ces modèles structurels nous permettent de comprendre les relations de cause à effet entre diverses variables. Ici, les variables observées représentant chacune, un grand concept du marketing sont créées à partir des réponses des utilisateurs à des questions d'un formulaire les engageants plus ou moins sur chacun des concepts utilisés. Dès lors, on observe les variations de l'un par rapport à l'autre selon le contexte qui est ici l'utilisation des moteurs de recommandation.<sup>12</sup>

Ici, ce modèle nous permet de comprendre que la diversité est un facteur important de la satisfaction utilisateur dans le cadre des moteurs de recommandation. Que la nouveauté est plutôt neutre et que la précision de ce dernier, sa capacité à déterminer des choix proches des préférences utilisateurs, est le facteur le plus important à maximiser.

Or, nous avons de manière plus générale ce modèle utilisant la satisfaction de manière à interpréter la fidélisation :



Nous permet de comprendre le liens entre image de la marque et satisfaction du client ainsi que l'effet de médiation de l'élaboration, une combinaison de l'implication du client avec la délibération de ce dernier, sur la satisfaction entraînant alors la fidélité du client envers la marque. Bloemer, J., & De Ruyter, K. (1998).<sup>13</sup>

<sup>12</sup> Bagozzi, R. P., & Yi, Y. (2012). Specification, evaluation, and interpretation of structural equation models. *Journal of the academy of marketing science*, 40(1), 8-34.

<sup>13</sup> Bloemer, J., & De Ruyter, K. (1998). On the relationship between store image, store satisfaction and store loyalty. *European Journal of marketing*.

SAT	INV	DEL	IM	LOY
SAT	0.13	-0.25 <sup>a</sup>	0.35 <sup>b</sup>	0.49 <sup>b</sup>
INV		0.23 <sup>a</sup>	0.15	0.17
DELI			-0.01	-0.10
IM				0.47 <sup>b</sup>

**Notes:**

SAT: store satisfaction; INV: involvement; DEL: deliberation; IM: store image; LOY: store loyalty

<sup>a</sup> One-tailed significance < 0.05

<sup>b</sup> One-tailed significance < 0.001

Tout d'abord ce modèle nous présente une corrélation directe de ces facteurs sur l'implication. Puis nous fait remarquer que le meilleur modèle prédictif est bien l'effet de médiation de ces facteurs sur la satisfaction ainsi que l'utilisation de l'image en tant que prédateur de la satisfaction.

« *H3. Store image has an indirect effect on store loyalty through store satisfaction.* » Ce qui signifie que dans le cadre de cet article c'est bien dans ce sens que l'image agit sur la satisfaction qui elle alors via la médiation de l'implication et la délibération nous donne la fidélisation du client.

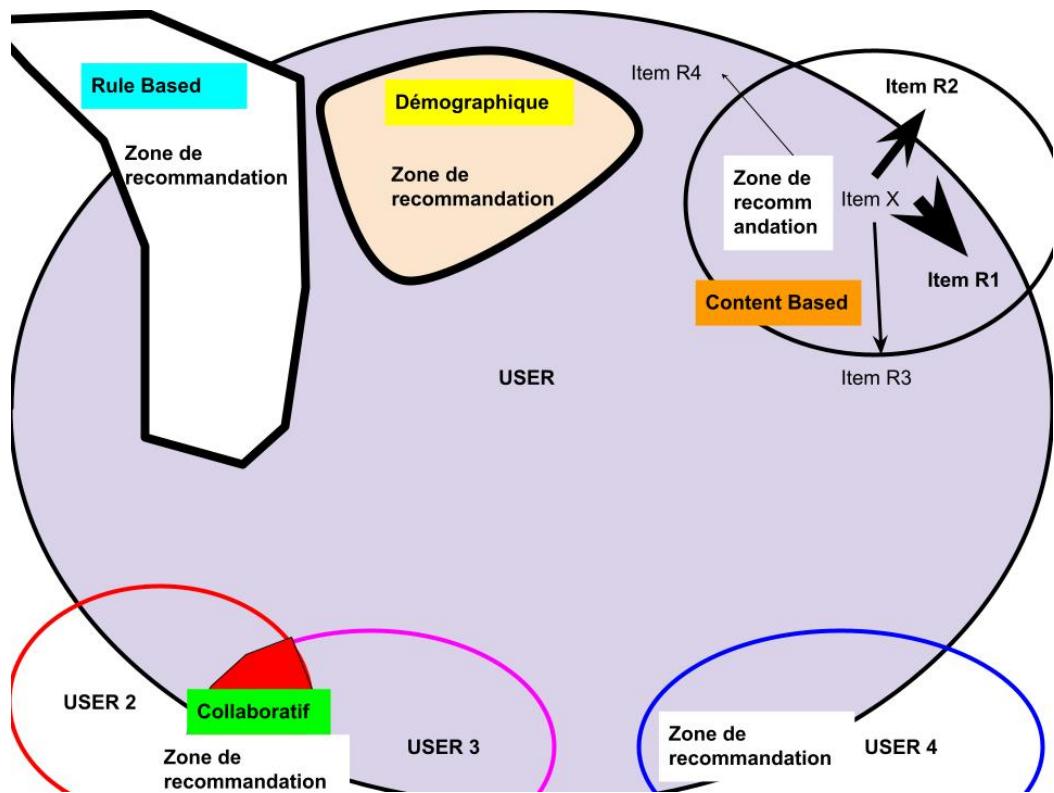
Un autre liens entre satisfaction et fidélisation peut aussi ce constater dans cette article : Javed, F., & Cheema, S. (2017).<sup>14</sup>.

Cela pour dire que cette dernière ne doit pas être négligée, or dans le cadre des moteurs de recommandations, la satisfaction comme nous l'avons vue vient de plusieurs facteurs tels que la diversité des choix proposés et la précision de la recommandation selon les préférences utilisateurs. Ils sont donc les deux facteurs sur lesquels nous devons nous concentrer.

---

<sup>14</sup> Javed, F., & Cheema, S. (2017). Customer satisfaction and customer perceived value and its impact on customer loyalty: the mediational role of customer relationship management. *The Journal of Internet Banking and Commerce*, 1-14.

## Effet de biais dans la recommandation :



En regroupant et en représentant ces différentes méthodes de recommandations par rapport à l'utilisateur voici donc ce que l'on obtient. Une sorte de carte dans laquelle on voit l'ensemble de ce qui représente les préférences l'utilisateur. Ce qui pourrait lui être recommandé est englobé de diverse manière par les divers algorithmes de recommandations suivant leurs méthodes. Ainsi, les algorithmes collaboratifs utilisent bien d'autres utilisateurs ayant un comportement commun pour représenter des zones de recommandations probables. De même que ceux « content based », basé sur le contenu, préfèrent une notion de distance par rapport à un contenu ou une requête connue représentative de la recommandation. Ceux démographiques ne font qu'englober une zone liée à un comportement identifié et ceux « rules based », basé sur des règles, eux sortent tout item répondants positivement aux règles définies.

Bien entendu, chaque algorithme a alors ses défauts, appelés biais. Fortement liés à la manière dont l'algorithme prend des décisions. **Tsintzou, V., Pitoura, E., & Tsaparas, P. (2018)**<sup>15</sup>. Ici, ils sont représentés par le fait que ces algorithmes vont soit trop se limiter par rapport à ce qu'aime vraiment l'utilisateur, soit en sortir dans certains contextes. Cela est représenté par les bordures noires de chaque types d'algorithmes représentant la frontière, limite à laquelle l'algorithme, selon ses règles mathématiques, peut explorer les préférences utilisateurs par rapport au cercle violet, les véritables préférences dont la superficie est inconnue. Lorsque l'on sort de ce cercle on recommande alors quelque chose ne faisant pas partie des préférences utilisateurs et de même la surface englobée par chacun des algorithmes représente la limite de ses capacités à découvrir de nouvelles préférences utilisateurs.

### **Certains de ces biais sont bien connus tel que :**

Le « **cold start effect** » apparaissant lorsqu'un utilisateur est nouveau et donc encore méconnu de l'algorithme **Hannech, A., Adda, M., & Mccheick, H. (2016, October)**<sup>16</sup>. Le **biais du genre**, lorsqu'une recommandation est orientée selon une identité sexuelle **Tsintzou, V., Pitoura, E., & Tsaparas, P. (2018)**<sup>17</sup>. Ou encore le bien connu biais de popularité, dans lequel on attribue naturellement beaucoup de confiance dans la recommandation de produits déjà populaires, le tout nourrissant alors un cercle vicieux dans lequel plus de popularité mène à plus de recommandations qui mène alors à plus de popularité **Fleder, D., & Hosanagar, K. (2009)**<sup>18</sup>.

---

<sup>15</sup> **Tsintzou, V., Pitoura, E., & Tsaparas, P. (2018).** Bias disparity in recommendation systems. *arXiv preprint arXiv:1811.01461*.

<sup>16</sup> **Hannech, A., Adda, M., & Mccheick, H. (2016, October).** Cold-start recommendation strategy based on social graphs. In 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON) (pp. 1-7). IEEE.

<sup>17</sup> **Tsintzou, V., Pitoura, E., & Tsaparas, P. (2018).** Bias disparity in recommendation systems. *arXiv preprint arXiv:1811.01461*.

<sup>18</sup> **Fleder, D., & Hosanagar, K. (2009).** Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. *Management science*, 55(5), 697-712.

Tout en passant par des biais plus complexes et plus profonds, que l'on découvre à force de l'utilisation de ces systèmes comme le fait que ces derniers peuvent globalement influencer le comportement utilisateur et de même la définition des produits **Chaney, A. J., Stewart, B. M., & Engelhardt, B. E. (2018, September)**<sup>19</sup>.

Parfois, ils sont en partie volontaires ou tolérés lors de l'usage et parfois, ils sont gênants et peuvent devenir source de mauvaises recommandations. Allant alors même jusqu'à lasser l'utilisateur, lui donnant ainsi une mauvaise image de l'application. **Lee, G., & Lee, W. J. (2009)**<sup>20</sup>.

Chaque algorithme a alors ses propres biais. Selon sa méthode, il est plus ou moins vulnérable à différentes sources de mauvaises interprétations. Par exemple un algorithme rule based ne sortira pas du carcan de règles qui le définissent. Là où un algorithme démographique se limitera au groupe d'appartenance de l'utilisateur. Un algorithme content based lui aura une forte tendance à ne proposer qu'une seule catégorie de produit correspondant strictement à la requête et enfin un algorithme collaboratif semblera plus aléatoire lorsque l'on connaît peu d'information de l'utilisateur.

Il se trouve qu'il y a un biais qui semble commun à tous ces algorithmes. Les effets de bulles filtrantes **Grossetti, Q. (2018)**<sup>21</sup>. Elles se définissent particulièrement par des groupes de recommandations fonctionnant entre eux, ainsi une recommandation liée à un élément du groupe mène elle-même à une autre recommandation liée au même groupe, comme des bulles dont l'utilisateur aura du mal à s'extraire. Car les recommandations qui lui seront proposées le mèneront toujours à ces bulles et donc l'ouverture des choix qui lui seront proposés seront limités par le filtre que représenteront ces bulles. L'utilisateur aura alors de fortes chances de rester dans ses groupes de recommandations, groupes liés à ses préférences passées. Par exemple, admettons qu'un utilisateur ait consommé une vidéo d'un contenu médiatique, il y a alors de fortes chances que d'autres contenus du genre lui soient proposés, et donc de grandes chances qu'il continu d'en consommer ce

---

<sup>19</sup> **Chaney, A. J., Stewart, B. M., & Engelhardt, B. E. (2018, September).** How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In *Proceedings of the 12th ACM Conference on Recommender Systems* (pp. 224-232).

<sup>20</sup> **Lee, G., & Lee, W. J. (2009).** Psychological reactance to online recommendation services. *Information & Management*, 46(8), 448-452.

<sup>21</sup> **Grossetti, Q. (2018).** Système de recommandation sur les plateformes de micro-blogging et bulles filtrantes (Doctoral dissertation, Sorbonne université).

qui renforce ce lien : « utilisateur » - « contenu médiatique ». Ainsi, une bulle se définit par une très forte probabilité qu'un utilisateur consommant un ensemble X de recommandation consomme d'autres recommandations venant de l'ensemble X lors des itérations futures.



Ainsi, ces dernières poussent les utilisateurs à toujours consommer les mêmes types de contenus ce qui empêche d'explorer de nouvelles informations, de découvrir de nouveaux produits. Cela bloque alors l'algorithme qui aurait besoin d'apprendre de nouvelles préférences potentielles encore inconnues de l'utilisateur pour en sortir, etc. Dans certains cas, ce biais peut même aller plus loin que fausser l'expérience utilisateur. En effet, dans un monde où la recommandation, va nous guider non seulement tout au long de notre expérience utilisateur, mais aussi dans chacun de nos choix, allant parfois même jusqu'à définir notre consommation, **Chaney, A. J., Stewart, B. M., & Engelhardt, B. E. (2018, September)**<sup>22</sup>, il se trouve que les bulles filtrantes peuvent alors vite devenir un véritable fléau comme le montre cet article étudiant leurs causalités dans la propagation des fake news. **DiFranzo, D., & Gloria-Garcia, K. (2017)**<sup>23</sup>. Ainsi donc, lorsque l'utilisateur ne peut s'échapper de sa bulle, si celle-ci se trouve de mauvaise influence, alors il est très difficile pour lui de faire le tri et de s'en sortir. Car chaque choix proposé renforcera cette mauvaise influence.

Ainsi, une question se pose : comment faire en sorte qu'un algorithme de recommandation soit à la fois pertinent dans ses propositions et « créatif », capable de nouveautés, parfois proches d'une exploration aléatoire, permettant à l'utilisateur de continuer de découvrir des données nouvelles, de le sortir de ses bulles filtrantes, en lui proposant des données qui soient peu en lien avec ce qu'il connaît déjà ou ce que l'algorithme sait déjà de lui, sans pour autant lui proposer des recommandations qui ne sont pas en adéquations avec ce qu'il recherche ? Evitant alors à la fois une lassitude future de l'utilisateur et d'autres conséquences possibles liées à la manipulation du comportement utilisateur à cause de ces dernières.

<sup>22</sup> **Chaney, A. J., Stewart, B. M., & Engelhardt, B. E. (2018, September).** How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In *Proceedings of the 12th ACM Conference on Recommender Systems* (pp. 224-232).

<sup>23</sup> **DiFranzo, D., & Gloria-Garcia, K. (2017).** Filter bubbles and fake news. *XRDS: crossroads, the ACM magazine for students*, 23(3), 32-35.

Par ailleurs, des chercheurs, **Nguyen, T. T., Hui, P. M., Harper, F. M., Terveen, L., & Konstan, J. A. (2014, April)**<sup>24</sup>, proposent une méthodologie permettant de les quantifier. On y trouve deux grandes idées, une qui se base plutôt sur une diversification de la consommation utilisateurs au cours du temps par rapport à une utilisation neutre des moteurs de recommandations et une utilisant le génome de la recommandation, sa nature, donnant alors une notion de proximité d'un item par rapport à d'autres items. Cela permettant dès lors d'estimer en utilisant la distance euclidienne, la proximité d'une recommandation faite choisie par l'utilisateur par rapport à l'historique de consommations de ce dernier. On à alors deux groupes, le premier choisit des items, ici des films sans recommandations et le deuxième des films recommandés. En suivant alors leurs historiques de consommation on peut comparer les deux distributions de distance entre les films et donc de diversité de consommation au cours du temps. Ce qui nous permet dès lors de constater l'impact sur la diversité de consommation des effets bulles sur l'utilisateur.

Ensuite, techniquement, dans la nature de nos algorithmes, un effet bulle semble se définir par une incapacité de notre algorithme à trouver de nouvelles préférences utilisateurs au bout d'un certain temps. Il s'agit donc d'une sorte de frontière délimitée par la logique de l'algorithme dans la carte multidimensionnelles des données utilisateurs et produits de recommandations. Il semblerait alors que définir une nouvelle règle, une nouvelle logique ne ferait que les définir autrement. Il s'agit donc de repousser ces frontières de manière algorithmique, mais comment le faire alors qu'un effet bulle ne peut se définir par un seul instant T et est une conséquence d'un ensemble de recommandations ? Difficile donc d'imaginer une règle estimant ces frontières dans un premier temps et ensuite de trouver une nouvelle frontière dont on ne peut garantir qu'elle n'en fait pas partie. Et donc difficile d'imaginer une simple règle d'ajustement, qui de plus serait finalement comme mettre du scotch là où une fissure est déjà apparue et a déjà fragilisé le matériel. Il nous faut donc une autre solution sans pour autant utiliser un aléatoire pouvant avoir un effet complètement contraire à celui souhaité et n'ayant pas forcément en plus une forte chance de sortir l'utilisateur de notre bulle car celui ci aurait peu de chance de choisir ces recommandations et donc d'ajuster l'algorithme. il semblerait qu'il nous faille plutôt un algorithme dont la nature est d'explorer les limites frontalières des méthodes cités et de les repousser à la moindre interaction utilisateur, comme pourrait le faire un algorithme de recommandation basé sur l'apprentissage par

---

<sup>24</sup> **Nguyen, T. T., Hui, P. M., Harper, F. M., Terveen, L., & Konstan, J. A. (2014, April).** Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web* (pp. 677-686).

renforcement avec une subtilité supplémentaire car notre algorithme doit être capable de procéder ainsi même lorsqu'il n'est pas récompensé pour ce faire. Car les algorithmes de renforcement Learning se contenteront finalement comme un algorithme classique de n'explorer des possibilités que si elles semblent le récompenser, par l'interaction de l'utilisateur avec ces dernières et donc encadrera finalement les réponses possibles dans une bulle selon cette règle de la récompense et n'en sortira que probablement peu. Ce qu'il nous faut est alors une capacité naturelle à explorer, à créer de la nouveauté, à surprendre dans les recommandations.

## **Qu'est ce que la créativité, la nouveauté et comment sortir des effets de bulles :**

Afin de mieux comprendre en quoi quelque chose peut être « créatif » sans pour autant être complètement aléatoire, il faut se renseigner sur le concept même de ce que sont la nouveauté ou la « créativité ». Il faut également comprendre comment les algorithmes peuvent y répondre. Dans l'article « **What is Creativity ? Whats it is not ?** de Alfred Balkin<sup>25</sup> » On apprend que la créativité n'est pas un don, mais plutôt quelque chose que l'on peut apprendre ou non, une sorte de savoir-faire. Pour lui, ce savoir-faire est une mise en avant de liens qui existent déjà, mais dont il a fallu une perspective créative pour être mis en avant et donc pas vraiment l'issue d'un simple aléatoire. Ainsi, le nouveau concept existe quelque part, mais pour le faire apparaître, il faut cette créativité. L'article met aussi en avant autre chose, pour lui la créativité ne s'arrête pas au produit. Elle est au contraire le fait d'une personne en interaction avec un processus qui donne alors un produit. Si la personne : l'utilisateur, et le processus : un algorithme, interagissent ensemble alors la recommandation devenant le produit est issue d'un processus de créativité si celui-ci reste innovant. Il faut donc que la mise en liens qui émergent de la perspective créative de ses interactions produisent alors sans cesse de nouvelles recommandations. Cependant, il est difficile de concevoir une recommandation unique à chaque utilisateur, en effet, chaque produit sera bien vu non seulement d'un utilisateur, mais est voué à l'être de plusieurs sans pour autant changer de nature. On peut alors voir une forme d'unicité de par le panel de recommandation, l'ensemble de ces dernières étant ici bien unique à l'utilisateur. Cependant, cette unicité est elle vraiment nécessaire afin de rester créatif ? En effet comme nous l'enseigne **Boden, M. A. (2004)**<sup>26</sup>, dans le livre **The creative mind: Myths and mechanisms. Routledge**. Il existe tout d'abord plusieurs

---

<sup>25</sup> **Balkin, A. (1990).** What is creativity? What is it not?. *Music Educators Journal*, 76(9), 29-32.

<sup>26</sup> **Boden, M. A. (2004).** The creative mind: Myths and mechanisms. Routledge.

types de créativités selon la méthode utilisée pour exprimer celle-ci, mais aussi plusieurs types de créativités selon le groupe d'individus pour qui elles sont créative.

Par exemple dans le cadre du groupe ciblé, la créativité nouvelle pour la personne ou le sujet, mais pas au sens historique P-creativity ou nouveauté psychologique vient comme le décrit M. A. Boden du fait qu'un enfant n'ayant jamais lu Shakespeare, mais se mettant pourtant à en écrire restera quelqu'un de très créatif simplement pas au sens Historique. Cela nous donne aussi la deuxième catégorie celle H-creativity, créatif au sens historique, nouvelle pour l'ensemble des individus. Ici, nos recommandations ne doivent pas particulièrement être uniques au sens historique, mais plutôt aux yeux de l'utilisateur et créative selon le référentiel de notre algorithme. Bien que l'ensemble des recommandations liées à un utilisateur peuvent être créatives au sens historique, car si l'ensemble de recommandations est unique à chaque itération alors très naturellement les effets de bulles seront évités après un nombre d'itération. Cela ne semble pas être la meilleure méthode, en effet est-ce-bien nécessaire de garantir cela ? Et est-ce seulement possible, le nombre d'ensembles de recommandations unique est certes exponentiel au nombre de produits, mais tout de même limité et une vérification de cette unicité semble contre productif. De plus combien d'itérations seraient nécessaire afin de sortir de la bulle ? et l'algorithme serait-il alors pertinent pour l'utilisateur ?

Malgré cela si le processus à une forte tendance à être H-créatif alors il semble bien naturel que ces effets soient minimisés, car alors cela signifierait que la recommandation faite à l'utilisateur aura une forte tendance à sortir des normes établies et donc que les utilisateurs de manière générale auront plus régulièrement des ensembles de produits auxquelles on ne s'attend pas à ce qu'ils aillent de pair.

Mais comment atteindre ce genre de créativité et le faire correctement sans un aléatoire pur, qui serait lui non-souhaitable puisque ce dernier guiderait très peu l'utilisateur et lui fournirait une expérience très peu personnalisée ? Ceci réduirait le biais au prix d'une réduction des interactions utilisateurs avec les produits et très sûrement de sa patience à force de devoir faire lui-même le tri entre les bonnes et les mauvaises recommandations.

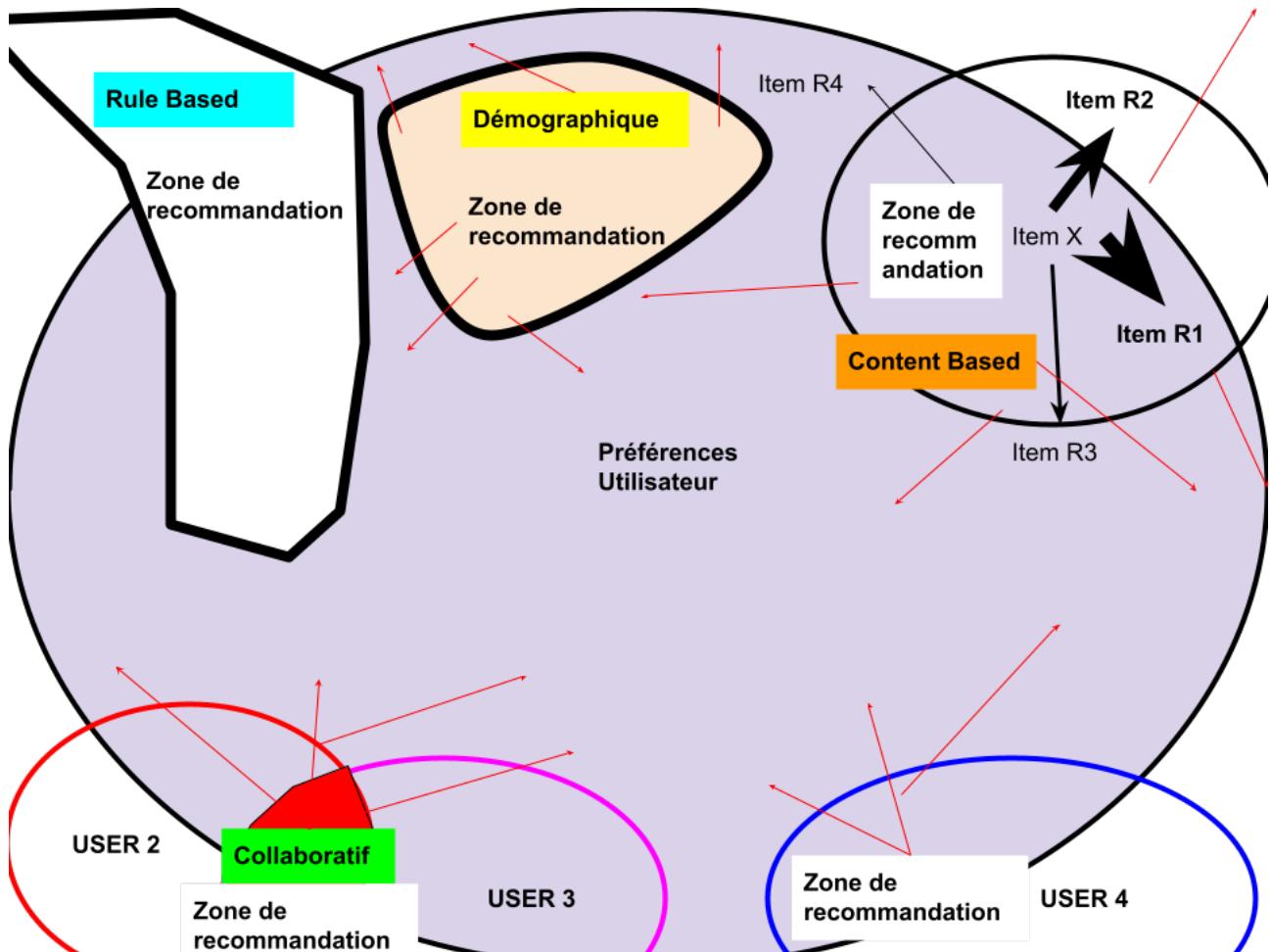
Hé bien, encore une fois dans ce même livre de **M. A. Boden**<sup>27</sup>. ainsi que dans l'article **Dimensions of Creativity.** de **DEARY, V. (1998)**<sup>28</sup>, on nous présente plusieurs autres types de créativités cette fois non pas basés sur la perception du concept selon les individus, mais aussi sur la manière de l'établir. On retient alors trois catégories, celle combinatoire, dans laquelle on fait un lien entre une idée ancienne et une autre afin d'en faire émerger une nouvelle. Il s'agit là d'un processus plutôt suiveur très efficace, car facilement créatif, mais qui a peu de chance de vraiment surprendre, en effet les diverses idées servant de sources au nouveau concept sont déjà l'œuvre d'un processus créatif connu. Puis il y a la catégorie dite transformationnelle qui aura plutôt pour but de transformer quelque chose afin d'en tirer une autre qui sera dès lors créative et finalement la catégorie exploratoire partant de l'idée de totale exploration d'une carte ou d'un espace dimensionnel inconnu. Il s'agit là alors de la méthode utilisée afin de l'explorer, découvrir sans cesse de nouvelles structures jusqu'alors inexplorées, cachées dans la carte multidimensionnelle des données. Parmi ces trois possibilités, celle combinatoire est la plus accessible et exploitée jusqu'alors, cependant elle est moins souhaitable, car cette dernière mène rapidement à des boucles où deux idées ont déjà été combinées auparavant. Celle transformationnelle, elle est pratique, mais nous ne pouvons changer la nature du produit lors de la recommandation, elle ne semble donc pas indiquée ici. Quant à celle exploratoire, cette dernière est bien moins accessible, mais beaucoup plus pratique lorsque l'on cherche à être H-creative ou à surprendre dans le panel de choix, car ici, dans l'exploration, la découverte est totale, nous ne savons pas sur quoi nous allons tomber lors du parcours de la carte multidimensionnelle représentée par nos données. Elle semble donc la plus intéressante.

---

<sup>27</sup> **Boden, M. A. (2004).** The creative mind: Myths and mechanisms. Routledge.

<sup>28</sup> **DEARY, V. (1998).** Dimensions of Creativity. Edited by M. Boden.(Pp. 242.) MIT Press: Cambridge, MA. 1994. Genius: The Natural History of Creativity. By HJ Eysenck.(Pp. 344;£ 37.50.) Cambridge University Press: Cambridge. 1995. *Psychological Medicine*, 28(2), 485-494.

Si l'on reprend notre image donc, ce que nous souhaitons est un algorithme capable de non seulement reprendre les principes des algorithmes précédent, mais en plus capable de véritablement ce lancer dans l'exploration de cette carte de préférences. Contrairement à un simple aléatoire ayant de trop forte chance de sortir des préférences de notre utilisateur, nous souhaitons que notre algorithme soit capable de le faire sans trop s'éloigner rapidement de ce que l'on sait déjà sur l'utilisateur.



Autrement dit, on souhaite que non seulement nos algorithmes suivent les règles classiques permettant de connaître les points de départ des préférences clients, mais aussi, que ces algorithmes soient capables de parfois franchir ces frontières imposées par les règles logiques établies auparavant et d'explorer ce qui se trouve derrière petit à petit. Ainsi, si l'utilisateur montre face à cette exploration un nouvel intérêt alors il sera possible pour nous d'élargir cette frontière, de sortir de cette bulle.

## **Intelligence Artificielle et G.A.N., les Réseaux de neurones créatifs combinatoires et exploratoires :**

Cette recherche de la créativité dans un algorithme sans aléatoire paraît improbable. Au mieux, on a tendance à s'imaginer un algorithme froid mettant en liens des données selon une suite de règles dictées bien précises ou apprises des données dans le cas du machine Learning. **Carbonell, J. G., Michalski, R. S., & Mitchell, T. M. (1983)**<sup>29</sup>. Or, à l'heure actuelle, on parle pourtant de plus en plus du sens artistique de certains algorithmes. **Black, M. J. (2002)**<sup>30</sup>. Et tout particulièrement lorsqu'il s'agit des réseaux du type **G.A.N. (générationneur adversarial network)**<sup>31</sup>. Ces réseaux de neurones faisant partie des technologies du deep Learning sont capables non seulement de faire des liens entre une entrée et un résultat connu, mais aussi de comparer sa production à d'autres résultats, eux n'étant pas en lien direct avec l'entrée. Ainsi, tel un artiste, le modèle compare son œuvre à d'autres sans se soumettre à une seule bonne réponse, simplement en se posant la question, ai-je produit une œuvre d'art ? Cela permet alors de généraliser des tâches plus subjectives auxquelles il n'existe pas qu'une seule solution mais plusieurs dont il est difficile d'en connaître une meilleure que les autres et donc de favoriser la créativité par rapport à une vérité à prédire unique. Depuis, ces modèles sont utilisés dans de nombreux cas pratiques, par exemple dans un but purement créatif, où l'on se pose la question, qu'est-ce que la machine pourrait créer, quelle est sa perception de la musique, ou encore peut-on transformer un morceau de rap en véritable œuvre de musique classique ? **Mor, N., Wolf, L., Polyak, A., & Taigman, Y. (2018)**<sup>32</sup>

Cependant, ces modèles sont aussi capables de sortir du contexte artistique comme le montrent ces cas d'utilisations :

---

<sup>29</sup> **Carbonell, J. G., Michalski, R. S., & Mitchell, T. M. (1983).** An overview of machine learning. *Machine learning*, 3-23.

<sup>30</sup> **Black, M. J. (2002).** The art of code. University of Pennsylvania.

<sup>31</sup> **Aggarwal, A., Mittal, M., & Battineni, G. (2021).** Generative adversarial network: An overview of theory and applications. *International Journal of Information Management Data Insights*, 1(1), 100004.

<sup>32</sup> **Mor, N., Wolf, L., Polyak, A., & Taigman, Y. (2018).** A universal music translation network. *arXiv preprint arXiv:1805.07848*.

- 1) Dans le cas de prédiction dans la physique, des particules ou le modèle permet d'estimer efficacement des dépôts d'énergies. **de Oliveira, L., Paganini, M., & Nachman, B. (2017)**<sup>33</sup>
- 2) Dans la détection de fausses images générées par un algorithme **Hsu, C. C., Zhuang, Y. X., & Lee, C. Y. (2020)**<sup>34</sup>,
- 3) Ou même dans des études pharmaceutiques comme l'utilise **Bian, Y., Wang, J., Jun, J. J., & Xie, X. Q. (2019)**<sup>35</sup> dans le cas de l'étude de drogues.

Dans notre processus de créativité on remarque donc que cela nous situerait facilement dans un cadre à la fois combinatoire, transformationnel, mais aussi et surtout exploratoire, car l'algorithme n'a pas d'idée précise du résultat qu'il doit trouver et explore bien les possibilités afin d'en tirer certaines, suffisamment convenables au sens de ce que l'on connaît du concept de la tâche sans se baser uniquement sur les entrées, mais aussi en explorant les résultats possibles connus. En effet, ici, l'algorithme dans un cadre GAN n'a pas pour but de trouver une solution précise à une problématique donnée, mais doit apprendre à comprendre le concept qu'on lui présente via une analyse exploratoire afin ensuite d'être capable de le reproduire lui-même à partir des données d'entrées lors d'une créativité, elle, plus combinatoire faisant alors le lien de la donnée vers un résultat suivant : le concept apprit.

Avant de rentrer dans les détails du modèle commençons par définir le fonctionnement des réseaux de neurones, qui eux sont plutôt à l'origine des algorithmes dit combinatoires bien que certains comme celui dont on parle ici, ainsi que ceux issus du renforcement Learning trouvent toujours le moyen de nous surprendre avec leurs créativités exploratoires.

---

<sup>33</sup> **de Oliveira, L., Paganini, M., & Nachman, B. (2017).** Learning particle physics by example: location-aware generative adversarial networks for physics synthesis. *Computing and Software for Big Science*, 1(1), 1-24.

<sup>34</sup> **Hsu, C. C., Zhuang, Y. X., & Lee, C. Y. (2020).** Deep fake image detection based on pairwise learning. *Applied Sciences*, 10(1), 370.

<sup>35</sup> **Bian, Y., Wang, J., Jun, J. J., & Xie, X. Q. (2019).** Deep convolutional generative adversarial network (dcGAN) models for screening and design of small molecules targeting cannabinoid receptors. *Molecular pharmaceutics*, 16(11), 4451-4460.

# Qu'est que l'intelligence artificielle ?

---

## Définition et introduction :

Afin de mieux comprendre ce que signifie l'intelligence artificielle, reprenons son idée depuis ses origines. L'I.A au départ en terme d'algorithme, commence par n'importe quel procédé algorithmique ayant pour finalité la simulation d'un comportement dit « intelligent » souvent une tâche associée à un comportement humain ou animal mais très vague. Toutes sortes d'algorithmes se retrouvent alors catégorisés dans l'I.A., de la simple fonction basée sur des règles telles qu'une suite de conditions menant à un comportement précis selon des scénarios bien précis, à une véritable capacité d'apprentissage voire d'adaptations dans certains contextes. Comme on peut le lire dans l'article « **Qu'est ce que l'IA et qu'est ce que ce n'est pas?** »<sup>36</sup> La différence entre un algorithme classique et l'I.A. est surtout basée sur la capacité de l'algorithme à répondre à une tâche demandant habituellement des capacités cognitives. Un des domaines de l'intelligence artificielle auquel nous nous intéressons plus dans ce mémoire est le machine learning. Le machine learning vient de l'idée qu'un algorithme peut à même titre qu'un humain apprendre. Pour ce faire, de nombreux procédés sont utilisés. **Mahesh, B. (2020)**<sup>37</sup>. On peut lister les algorithmes supervisés où un algorithme fait de lui-même le lien entre une donnée d'entrée et un résultat connu, venant de la connaissance de l'utilisateur. Ici, l'algorithme définit lui-même les parfois milliers de règles permettant de « comprendre » la donnée. On peut aussi citer les algorithmes fonctionnant plutôt de manière non supervisées. Méthode où l'algorithme cherchant à optimiser une règle précise va apprendre sans transfert d'expérience à traiter la donnée. Ou encore l'apprentissage par renforcement dans lequel l'algorithme va tester par essais d'erreurs de nombreuses possibilités afin de trouver les relations de causes à effets semblant le rendre plus performant.

---

<sup>36</sup> **Lagarrigue, A., & Viéville, T. (2021).** Qu'est ce que l'IA et qu'est ce que ce n'est pas?. *Lecture Jeune*, 180.

<sup>37</sup> **Mahesh, B. (2020).** Machine learning algorithms-a review. International Journal of Science and Research (IJSR). [Internet], 9, 381-386.

Ce qui est commun à tous ses algorithmes est simplement que dans le machine learning c'est bien toujours le modèle qui apprendra des données les règles qui semblent la régir. Généralement plus la donnée est complexe plus l'algorithme le devient, devenant alors parfois une sorte de boîte noire indescriptible trouvant le résultat sans vraiment expliquer les raisons à cela. C'est d'ailleurs selon Alan Turing considéré comme le père de l'intelligence artificielle, presque un but, car pour lui seul un système aux règles si complexes qu'elles ne peuvent en être décrites peut répondre à la question d'intelligence artificielle forte. **Goutefangea, P. (2017).**<sup>38</sup>

C'est particulièrement le cas lors de l'utilisation des réseaux de neurones. Ces derniers trouvent leurs origines dans l'étude biologique du fonctionnement des neurones. C'est ici ce que nous allons étudier. Pour ce faire, commençons par le début en définissant ce qu'est un « neurone artificiel » dans le cadre du « perceptron » ainsi que les idées qui s'en dégagent.

---

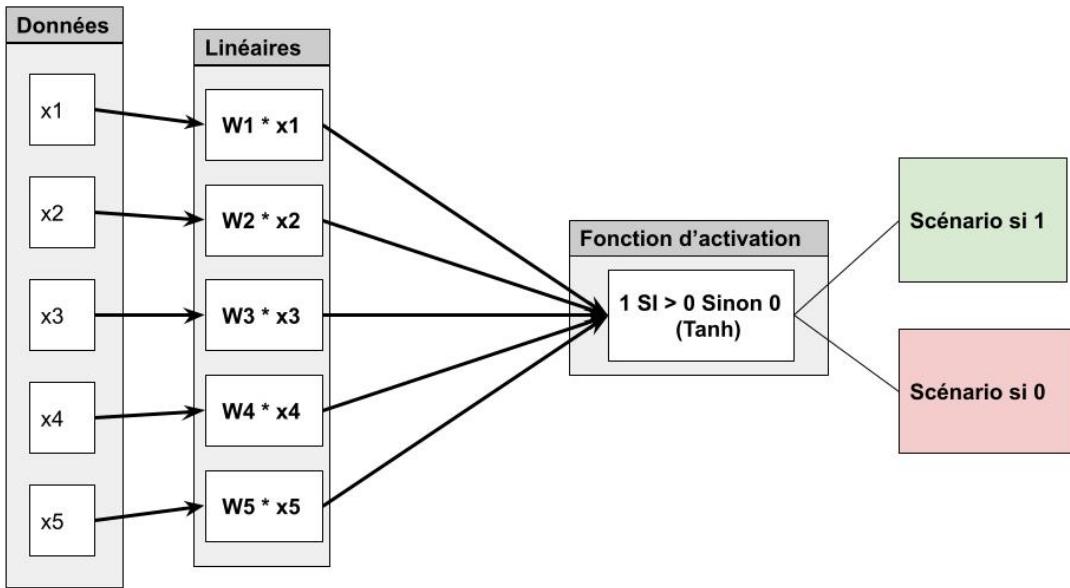
## Aux origines des réseaux neuronaux : Cas du Perceptron

Le perceptron est l'un des premiers algorithmes ayant permis de structurer le fonctionnement d'un neurone **Rosenblatt, F. (1958)**<sup>39</sup>. Le but de l'algorithme est d'obtenir une représentation de la prise de décision binaire selon plusieurs variables. Pour ce faire, le modèle utilise un neurone représenté mathématiquement par une somme d'équations linéaires suivi d'une fonction dite d'activation, fonction permettant la prise de décision selon la sortie des fonctions linéaires. Afin de faire fonctionner le modèle on envoie alors un signal au neurone, ce signal vient des données d'entrées. Il représente alors dans un cadre biologique l'information électrique stimulant le neurone. Soit le neurone est « Trigger », activé par l'information d'entrée, soit il ne l'est pas. C'est l'information de son état qui pourra alors permettre cette prise de décision binaire.

---

<sup>38</sup> **Goutefangea, P. (2017).** Alan Turing et l'intelligence artificielle: le «jeu de l'imitation» et «l'IA forte».

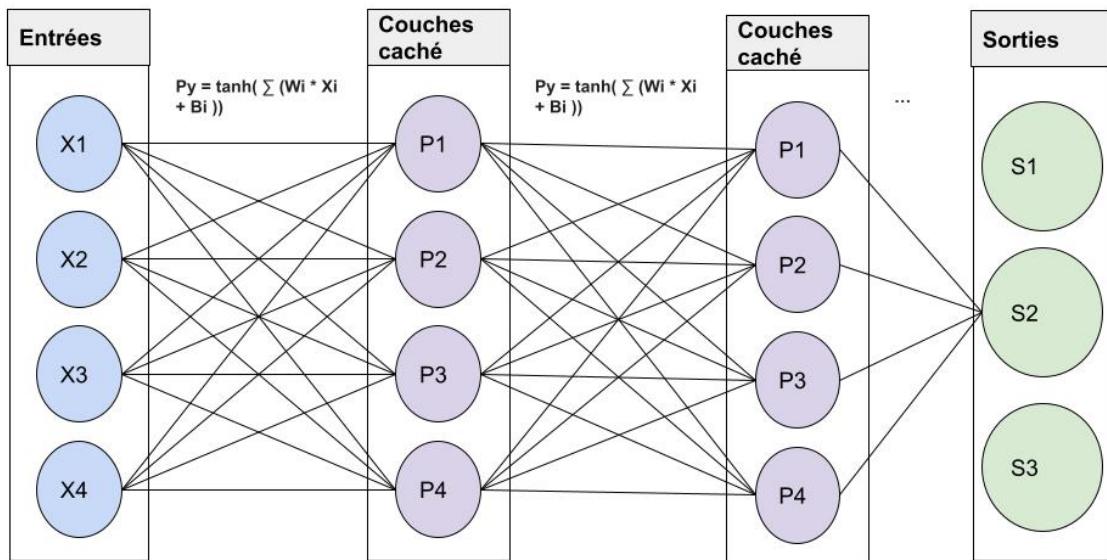
<sup>39</sup> **Rosenblatt, F. (1958).** The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6), 386.



Au départ, le modèle ne connaît pas la solution permettant de savoir à partir des données s'il s'agit d'une représentation du scénario 1 ou 2. Il doit l'apprendre. Segmenter par lui-même l'espace de solution. Pour ce faire, le modèle va d'abord tenter de trouver un résultat, s'il se trompe alors il ne connaît pas la solution ou ne sait pas la trouver donc cela signifie que l'on doit retrouver quels sont les facteurs de son erreur et les quantifier : à quel point le modèle s'est trompé et donc à quel point devons nous le corriger ? Dans ce but, nous allons calculer son taux d'erreur, ici, il s'agit simplement de la différence entre  $Y_t$  la vérité à prédire et  $Y$  la vérité prédictive ( $Y_t - Y$ ). Puis nous allons utiliser ce taux d'erreur afin d'améliorer le modèle, cette étape nommée back propagation est l'étape clef de l'apprentissage d'un réseau de neurones et dans le cadre du perceptron on peut utiliser la loi de Widow-Hoff qui permet de mettre à jour les paramètres du modèle selon ce taux d'erreurs. On a donc  $W'_i = W_i - \alpha (Y_t - Y) X_i$  avec  $\alpha$  le taux d'apprentissage. Cette équation permet étape après les étapes de converger vers une solution possible rendant le modèle alors capable de reconnaître selon les données d'entrées si l'on se trouve dans le scénario 1 ou 2 et donc par la suite de prendre des décisions.

## Aux origines des réseaux neuronaux : Cas du multilayer perceptron

Bien que le perceptron fût très impressionnant lorsqu'il s'agit de simuler une prise de décision, ses limites furent cependant vite atteintes. En effet en liant ainsi les données il n'est possible que de traiter l'information de manière assez linéaire, il faut que les données soient simples et que seul deux scénarios différents s'en dégagent. Raison pour laquelle des chercheurs se sont posés la question de comment rendre ce modèle plus « intelligent », capable de prendre des décisions plus complexes. C'est ainsi qu'est né le multi layer perceptron, un modèle perceptron dit multicouches où cette fois il n'y a pas un, mais plusieurs neurones connectés entre eux et plusieurs couches. Chaque neurone transmet alors de l'information au suivant afin d'affiner les décisions prises, communiquant alors l'information en « feed forward », de l'entrée vers la sortie. C'est alors les premiers réseaux de neurones, on ne se contente plus de simuler un neurone, mais aussi de les mettre en liens.



Cependant, il faut alors revoir la manière dont l'information va se propager dans un sens, lors de la prédiction, comme dans l'autre lors de l'apprentissage. Tout d'abord, la fonction calculant l'erreur n'est pas la même, les sorties pouvant ici être plusieurs et donc les scénarios plus nombreux. La question à laquelle le perceptron répond alors est, les données appartiennent-elles ou non au scénario 1, 2 ou 3, ... Selon le nombre de sorties.

Le plus simple pour calculer l'écart entre le résultat attendu d'un neurone et la vérité à prédire est une fonction MAE, Mean Absolute Error, ou taux d'erreur moyen ou MSE, Mean Squared Error, soit le taux d'erreur carré moyen :

$$\text{MSE}(e_i, y_i) = \frac{1}{n} \sum_{i=0}^n (y_i - e_i)^2 \quad \text{MAE}(e_i, y_i) = \frac{1}{n} \sum_{i=0}^n |y_i - e_i|$$

Avec  $Y(i)$  la vérité à prédire pour les données (i), et  $e(i)$  la valeur prédictive par le réseau pour les données (i).

Mais la plus remarquable nouveauté, celle qui permet vraiment de mettre en place ce réseau est l'algorithme de la descente de gradient. Algorithme qui a pour but de minimiser une fonction mathématique étape après étape, en ajustant les valeurs possibles des paramètres du modèle afin de minimiser l'erreur. La Descente de gradient se base sur l'idée qu'il est possible de modéliser la fonction Gradient du modèle, une fonction qui selon chaque valeur des paramètres ajustables du modèle nous donne les variations de la fonction coût. Minimiser le gradient revient donc à minimiser l'erreur du modèle.

L'équation du gradient s'exprime par la somme des dérivées partielles de la fonction coût selon le modèle. On a donc :

$$\text{Gradient} = \frac{\partial f}{\partial w} + \frac{\partial f}{\partial b} + \frac{\partial f}{\partial y}$$

Soit les dérivées partielles du modèle selon les valeurs des poids, ici, on n'en représente qu'une, suivies de la dérivée partielle du modèle selon les biais, ici de même, on n'en représente qu'une, puis finalement la dérivée partielle du modèle selon la vérité à prédire.

Cette chaîne de montagne est très pratique, car pour chaque valeur possible, elle nous indique comment on influence le taux d'erreur du modèle. Or, représenter cette fonction afin d'ajuster le modèle peut être très coûteux lorsque le réseau possède de nombreux paramètres. Pour compenser ce problème, un algorithme simple a été conçu, la descente de gradient. Ce dernier interprète le Gradient comme un flux liquide, dans ce flux de vague, on cherche alors à suivre le courant afin de petit à petit atteindre un point bas, provocant peu d'erreur. Pour ce faire, l'algorithme utilise la dérivée partielle de la fonction coût selon la valeur du poids du modèle à ajuster. Ainsi, si la dérivée est positive, on part de l'idée qu'augmenter la valeur du poids augmenterait notre taux d'erreur à l'étape suivante et inversement si elle est négative augmenter le poids réduirait l'erreur, car la dérivée nous donne le sens de la variation de la fonction et donc le sens du « courant » du « liquide » représenté par le gradient.

l'équation s'exprime alors ainsi :

$$\mathbf{w}' = \mathbf{w} - \text{alpha} (\frac{\partial f}{\partial w})$$

$$\mathbf{b}' = \mathbf{b} - \text{alpha} (\frac{\partial f}{\partial b})$$

Avec  $w'$  valeur future du paramètre et alpha le taux d'apprentissage, il en va de même pour  $b$ , le biais.

Afin de s'assurer de bien rester dans le sens du courant, il faut se déplacer « lentement » lors de l'ajustement des paramètres, il faut donc un alpha bas. De même, nous ne devons pas avancer trop lentement dans le courant afin de ne pas être pris dans un « tourbillon », une boucle dans laquelle chaque variation du paramètre nous fait revenir sur nos pas à l'étape suivante. Ce pourquoi généralement une bonne valeur d'alpha commence à 0,01 et peut varier durant le processus d'entraînement.

*Détails en Annexe : Calcul du gradient descent et de la backpropagation*

---

## Evolution vers les Réseaux de neurones et le Deep learning :

Comme vous pouvez le remarquer, la force de ces équations est leur flexibilité (particulièrement celles de la backpropagation en annexe), elles ne changent pas selon la complexité du modèle et peu selon les fonctions d'agrégation des entrées et d'activation que l'on souhaite utiliser. Dès lors, il est possible de concevoir des réseaux si profonds, possédant tant de couche et étant si complexes, qu'ils deviennent des sortes de boîte noires gigantesques dont l'interprétabilité, la compréhension de leurs prédictions est très faible, mais dont les capacités à apprendre et à s'adapter sont exponentielles.

Ainsi, suite à l'apparition des multilayers perceptron de nombreux autres modèles ont émergé chacun plus complexes les uns des autres, ajoutant chaque fois plus de profondeur aux réseaux de neurones en espérant ainsi leurs donner plus « d'intelligence », de « capacité d'adaptation ». Petit à petit de nouvelles fonctions d'activations autres que des fonctions « tanh » ont émergé ayant chacune leurs spécificités et leurs spécialités. Même si la philosophie des réseaux de neurones est plutôt dans la finalité d'obtenir un seul modèle capable de tout, actuellement, c'est bien la combinaison de plusieurs types de réseaux ayant chacun des fonctionnalités différentes dans un seul plus grand réseaux qui nous permettent de traiter des tâches complexes.

Dès lors on parle d'apprentissage profond et de deep Learning **LeCun, Y., Bengio, Y., & Hinton, G. (2015)**<sup>40</sup>, méthode utilisant le stochastique gradient descent ou d'autres méthodes proches, permettant de complexifier toujours plus les capacités des modèles dans le but de résoudre des tâches proches de la cognition humaine. Comme : jouer aux échecs, au go, interpréter une image, écrire, interpréter des sentiments, etc.

---

<sup>40</sup> **LeCun, Y., Bengio, Y., & Hinton, G. (2015).** Deep learning. *nature*, 521(7553), 436-444.

Ces algorithmes révolutionnaires semblent à l'heure d'aujourd'hui n'avoir de limites que notre imagination et notre puissance de calcul. Et pourtant, il est encore difficile de parler de super intelligence ou même d'intelligence. En effet, pouvons-nous seulement dire que la machine comprend ? Après tout ne s'agit-il pas plutôt d'un calcul minimisant les erreurs en fonction des données présentées, renforçant au fur et à mesure des liens entre données et résultats ? On aurait envie de dire parfois que, oui, la machine comprend le concept qu'elle traite, qu'elle sait bien différencier le chat du chien, la colère de la tristesse, mais cette machine sait-elle ce qu'est un chat ? La colère ? D'une certaine manière oui et non. L'I.A. en a sa propre interprétation écrite dans son réseau. C'est du moins ce que nous montre certains algorithmes tentant de décrypter les modèles **Montavon, G., Samek, W., & Müller, K. R. (2018)**<sup>41</sup>. Mais même ainsi cette interprétation de l'I.A. reste très limitée aux données qui lui sont présentées. Cette question semble alors insoluble et pourtant n'est-ce-pas nous-même ce que nous faisons au cours de notre vie, apprendre sans cesse de nouveaux concepts ?

---

### Les réseaux Créatifs, « Generative adversarial Network » :

Dans les cas vus auparavant, c'est bien l'algorithme qui met en liens la donnée et le résultat, c'est donc une grosse avancée dans cette recherche de la créativité. Cependant, dans tous les cas précédents, c'est notre expérience du résultat qui est apporté au modèle. Afin de résoudre notre problématique, il nous faut plus que cela. Il nous faut un réseau capable d'interpréter de lui-même ce qu'est le résultat, une bonne recommandation sans se baser uniquement sur notre expérience. Pour ce faire, il faut commencer à explorer les possibilités que nous offrent les réseaux Generative Adversarial Network ou réseaux génératifs par adversité. Ces derniers ont une manière très particulière d'aborder l'apprentissage du modèle qui permet d'éviter l'association d'un unique résultat connu à une donnée et donc de toujours laisser l'algorithme apprendre de lui-même ce qu'est un bon résultat, leur permettant dès lors de mettre en valeur beaucoup plus de valeur créative que de nombreux autres modèles. Ces réseaux partent du principe qu'il n'existe pas un seul tableau possible à un paysage donné, pas une seul solution « vrai » à une problématique artistique, mais un ensemble de solutions. Pour répondre à cette question des chercheurs ont finit par trouver une solution, des réseaux fonctionnant en concurrence l'un tentant de toujours produire un résultat plus « vrai », au sens artistique et l'autre cherchant à toujours « discriminer » l'œuvre produite par

---

<sup>41</sup> **Montavon, G., Samek, W., & Müller, K. R. (2018).** Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73, 1-15.

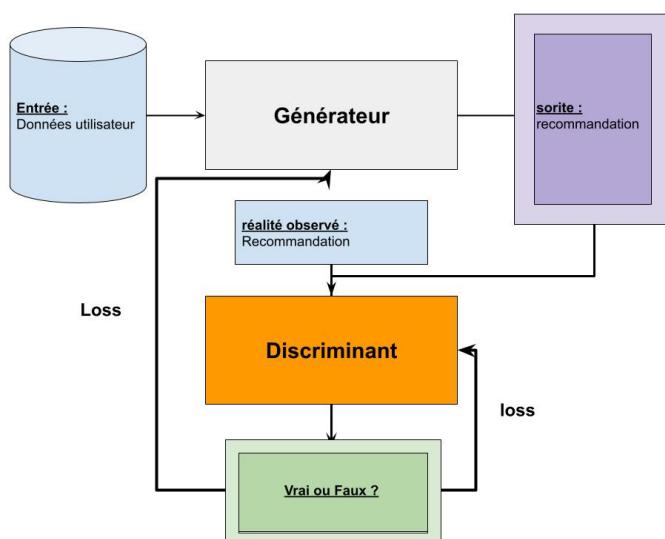
l'I.A. comme s'il s'agissait d'une simulation de la relation entre un critique d'art et un artiste dont l'un essaye de reconnaître l'artiste en question et l'autre à faire des tableaux dans des nouveaux styles.

### **Le modèle procède alors ainsi :**

Un premier réseau prédit une sortie à partir de données d'entrées, dans notre cas les données utilisateurs sont l'entrée et la recommandation la sortie.

Un second réseau compare la sortie du premier réseau, la recommandation, avec des résultats dont on sait qu'ils sont « vrais », et tente de prédire laquelle est issue du traitement d'un réseau de neurones et laquelle est une sortie issue de la réalité des données.

Suite à cela, le premier réseau utilise comme taux d'erreur son incapacité à produire une sortie suffisamment « vrai » pour ne pas être discriminée par son adversaire, le second réseau. Et le second réseau, lui, apprend au contraire de son incapacité à déterminer le « vrai » du « faux ».



Le modèle est alors corrigé uniquement lorsqu'il s'éloigne trop de quelque chose de semblable à un résultat observable et se permet donc d'explorer plus facilement les possibilités d'associations créatives possibles au sein de la donnée, tout en restant cohérent par rapport à la tâche demandée.

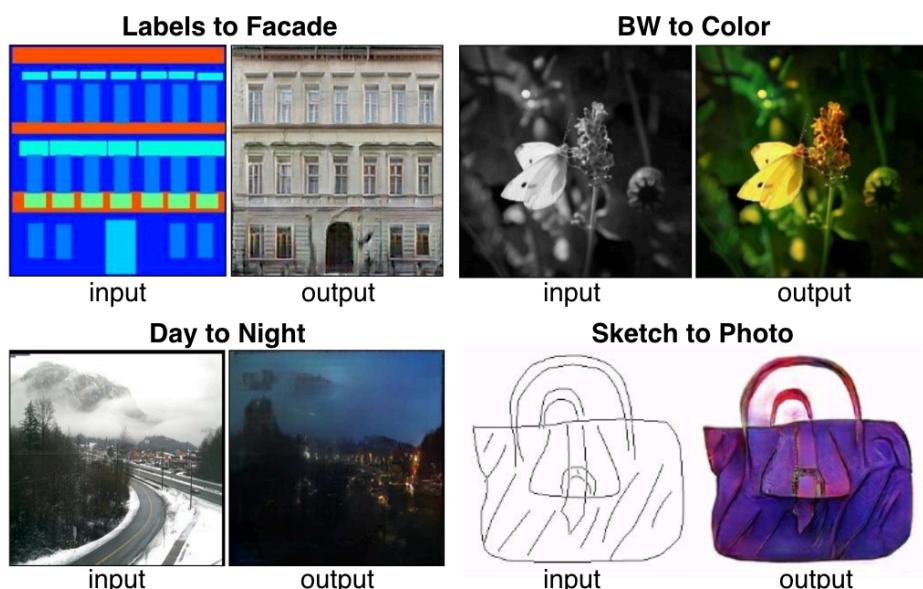
Autrement dit, alors que le discriminant cherche à maximiser une loss semblable à :

$$\text{MinGMaxD } f(D, G) = \mathbb{E}_x [\log(D(x))] + \mathbb{E}_z [\log(1 - D(G(z)))]$$

le Générateur lui cherche à la minimiser. Les deux réseaux évoluent alors en concurrence l'un par rapport à l'autre.

$\mathbb{E}_x$  est ici la proportion de « vrai » dans la prédiction.  $D(\mathbf{x})$  est alors la probabilité estimée du Discriminant pour les « vrais » données. Et inversement  $\mathbb{E}_z$  est la proportion de « faux » et  $D(G(z))$  l'estimation de la proportion de « faux ».

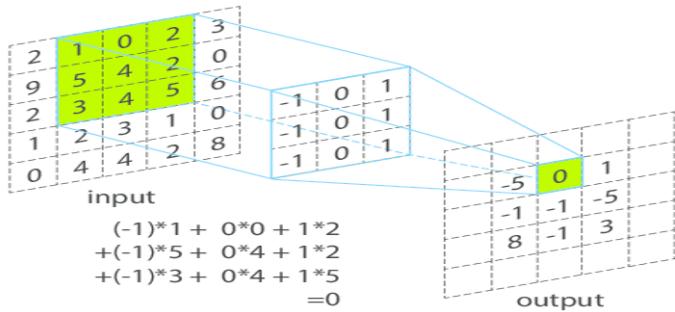
Pourtant, dans le cadre d'algorithme de recommandation, cela pourrait trop dépersonnaliser les données, trop pousser l'algorithme à simplement créer un résultat ressemblant à une recommandation sans pour autant se baser sur les préférences utilisateurs. Pour pallier à cela, il existe des réseaux Hybrides dans le machine Learning entre un entraînement supervisé classique et G.A.N., utilisant les deux méthodes. Cela permet de garder cette cohérence d'une entrée liée à un résultat connu, un utilisateur à ses choix, tout en ajoutant de la créativité. Dans ce cadre le réseau PixToPix originellement utilisé pour créer des images à partir de dessins semble approprié à cet usage. En effet, il suffit alors de traiter les données dans un format matriciel semblable à une image et de faire de même pour le résultat de la prédiction. Ainsi, il est possible de concevoir un algorithme de recommandation dit créatif selon notre définition.



Exemple de l'utilisation de PIX2PIX

## Réseaux Hybride GAN et Supervisé, PixToPix :

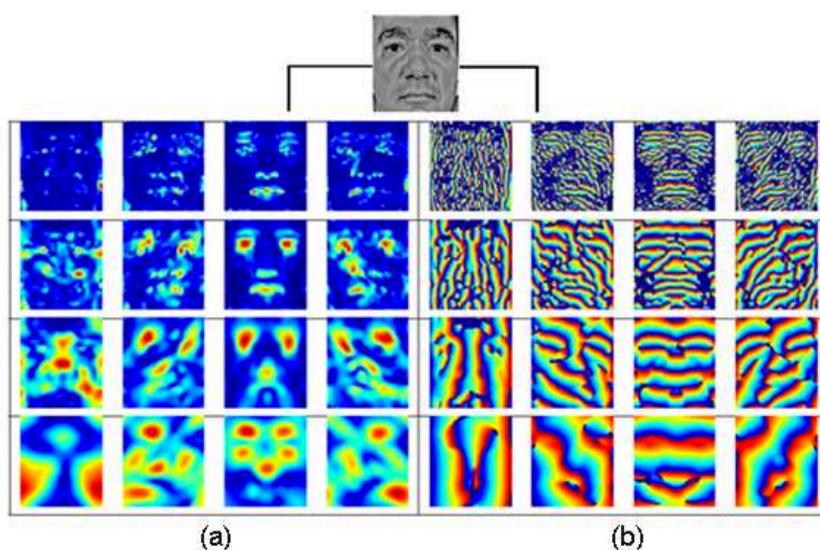
Ce type de modèle utilise des réseaux de convolutions afin de traiter les informations contenues dans des matrices 2D. Les convolutions procèdent ainsi :



**Shi, W., Caballero, J., Theis, L., Huszar, F., Aitken, A., Ledig, C., & Wang, Z. (2016)**<sup>42</sup>

Somme des  $x_{ij} * w_{ij}$  pour i et j dépendant de la taille du kernel. Le kernel étant la taille de la matrice qui parcourt les colonnes et lignes de nos données étape par étape. Ainsi les « pixels » ou données proches entre elles sont mises en liens de manière à mettre en lumière des patterns dans la donnée, des formes, des liens. Plus concrètement une convolution essaie de répondre, dans le cas du deep learning, à la question existe-t-il ce pattern donné ou non dans le bloc exploré et à quel point y est-il semblable ?

Par exemple dans le cas d'une image représentant un visage nous pourrions obtenir ces informations.



<sup>42</sup> Shi, W., Caballero, J., Theis, L., Huszar, F., Aitken, A., Ledig, C., & Wang, Z. (2016). Is the deconvolution layer the same as a convolutional layer?. *arXiv preprint arXiv:1609.07009*.

Ici, l'image est petit à petit décomposée ne laissant plus que les informations vraiment importantes, la personne a-t-elle des yeux et de quelle forme, une bouche et des oreilles, etc. ?

Dans notre modèle, car nous ne traitons pas réellement des images, la mise en liens se fait entre les utilisateurs et les films. Et donc on répond plutôt à la question quel autre utilisateur à vu ses films.

Afin d'éviter des problèmes d'overfitting le modèle combine les convolutions à des couches de type dropouts et des couches de type batch-normalization. Les dropouts correspondent à volontairement perdre de l'information aléatoirement durant l'entraînement sur une couche de neurones donné. Cela pousse le modèle à généraliser au mieux une tâche comme le ferait l'ajout de bruits dans les données. Cependant ici le dropout agit entre deux couches de neurones durant tout l'entraînement.

Quant aux couches de type batch-normalization elles ont pour but de normaliser les sorties de la couche précédente. Les raisons des performances de leurs utilisations restent discutées selon Wikipédia : « Batch Normalisation », bien que mon interprétation reste que la normalisation permet d'éviter un trop grand éparpillement dans les sorties des neurones et donc pousse le modèle à utiliser toutes les données plutôt que uniquement celles ayant le plus gros score, sans pour autant empêcher de l'apparition de scores mettant en avant des variables par rapport aux autres.

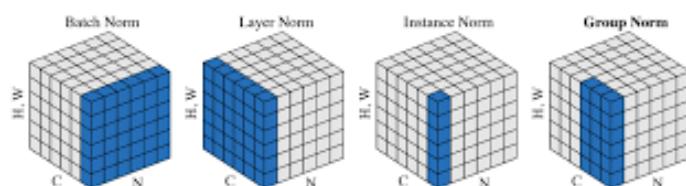


Figure 2. Normalization methods. Each subplot shows a feature map tensor, with  $N$  as the batch axis,  $C$  as the channel axis, and  $(H, W)$  as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

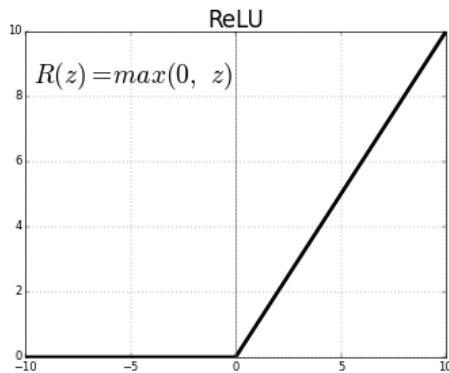
$$\hat{x}_i^{(k)} = \frac{x_i^{(k)} - \mu_B^{(k)}}{\sqrt{\sigma_B^{(k)2} + \epsilon}}$$

Pour ce faire, on effectue ce calcul sur chaque batch de dimension  $(x_1, \dots, x_d)$  ainsi :

Sois la valeur  $x_i$  du batch  $k$  appartenant à  $\{1, d\}$  moins sa moyenne divisée par son écart-type.

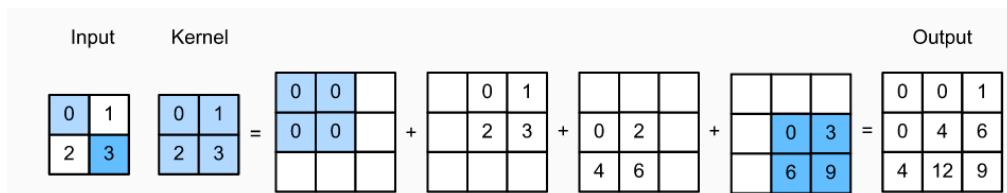
L'epsilon est une petite constante qui peut être utilisée pour ajouter de la stabilité dans la fonction.

Le tout est suivi d'une fonction relu, « rectified linear unit ». Fonction dans laquelle la somme des fonctions linéaires d'agrégations des poids des neurones et des variables d'entrées est simplement corrigée par  $x$ , la valeur d'entrée si  $x$  supérieur au seuil (souvent 0) ou sinon 0. Cela peut permettre d'ignorer des résultats devenant négatifs où de toujours s'assurer d'un résultat positif, elles sont très populaires pour leur efficacité en tant que fonctions d'activation dans les réseaux de neurones, augmentant bien souvent les résultats obtenus.



L'architecture de cet ensemble de couches s'appelle le down sampling. Une fois, plusieurs couches down sampling mises ensemble dans une forme pyramidale le réseau devient capable de rassembler l'information de la matrice et de l'analyser avec une perception plus globale. Les premières couches se contentent alors de rassembler les pixels proches puis les suivantes regardent ce que ces pixels donnent combinés aux autres informations déjà combinées, etc. Jusqu'à atteindre les dernières couches.

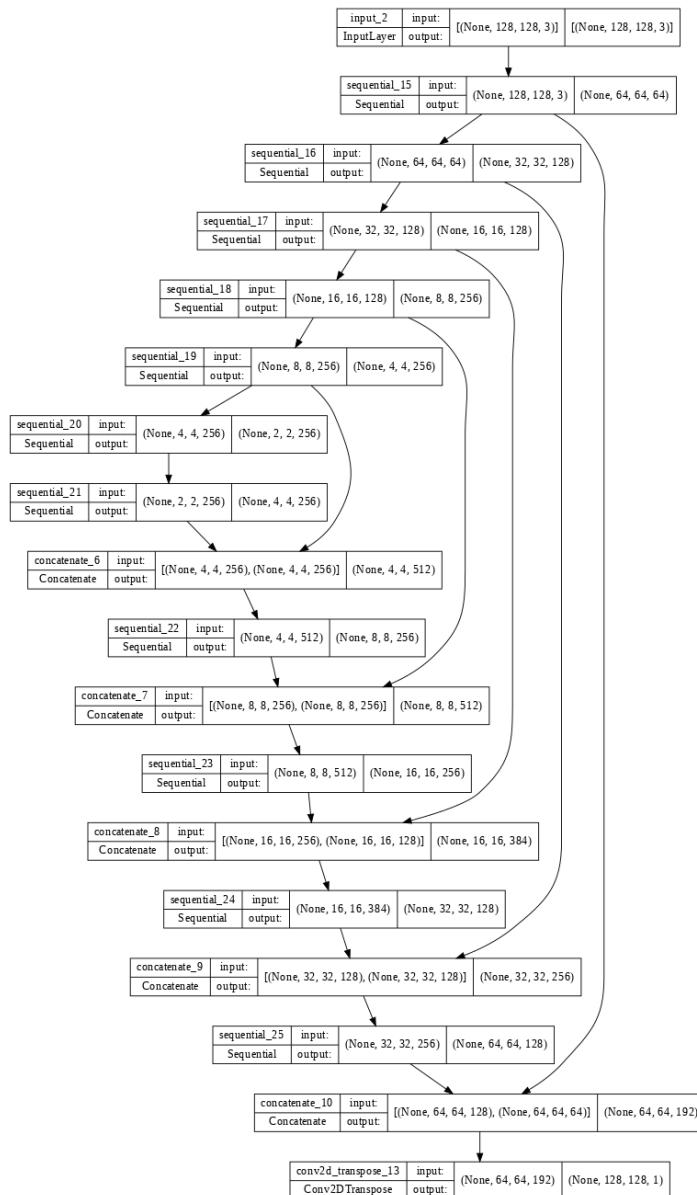
Seul cela ne suffit pas. L'information obtenue de la matrice d'entrée via les couches de down sampling étant compressée, il faut la reconstituer dans une matrice de la taille désirée afin de retrouver notre « image ». Le but est alors de passer de l'information « compressée » en entrée à une information « décompressée » dans une matrice plus large. **Dumoulin, V., & Visin, F. (2016)**<sup>43</sup>



<sup>43</sup> **Dumoulin, V., & Visin, F. (2016).** A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285.

Puis encore une fois, ces convolutions transposées sont suivies d'une couche de batch-normalisation ainsi que d'une couche drop-out et d'une couche de fonction RELU. Ce sous réseau s'appelle l'up-sampling. On obtient donc le modèle suivant, les premières couches étant des couches de Down-sampling et les suivantes d'up-sampling :

Une autre spécificité que l'on peut remarquer est l'interconnection de chaque couche de Down-sampling avec une couche équivalente en terme de « shape », taille de matrice traitée, à une couche d'Up-sampling. Ainsi, les couches d'Up-sampling possèdent non seulement l'information du traitement de la donnée du réseau précédent, mais aussi de ce à quoi ressemble « l'image » lors du traitement des couches supérieures et peut donc interpréter ses deux résultats l'un par rapport à l'autre, et ce, sans perdre trop d'informations suite aux traitements convolutionnels.



Une fois assemblé le modèle générateur est pleinement fonctionnel pour générer des images. Cependant, il lui faut encore apprendre à le faire. Comme dit précédemment, il faudra pour ce faire créer le modèle discriminant.

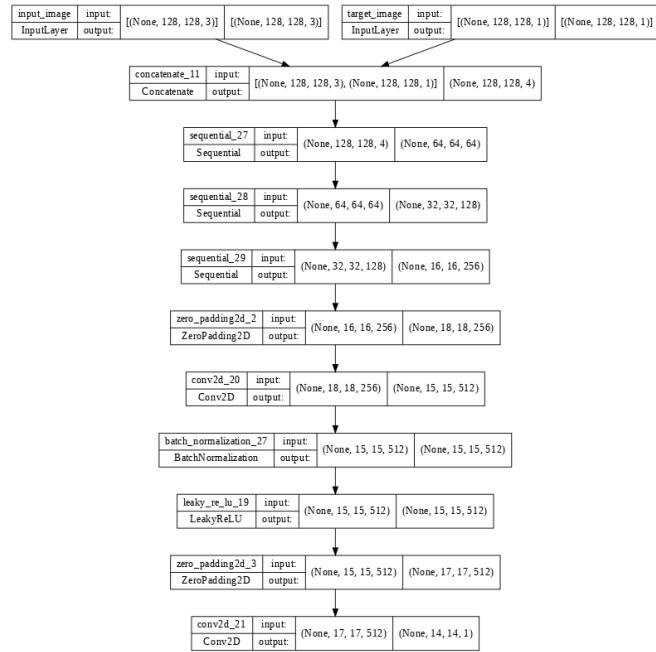
Ce dernier, lui, est constitué de couches de type down-sampling suivies d'une simple couche de convolution. Ce faisant, le modèle met en liens les informations comme le précédent et les « compresses ».

Puis d'une couche RELU, ou plutôt leaky RELU. La différence entre une fonction RELU et leaky RELU étant que si  $x < \text{seuil}$  alors la fonction est  $y = ax$ ,  $a$  étant un paramètre à calibrer lors de l'apprentissage. Il peut aussi y avoir des couches de type zéro « padding », le padding étant une manière de décrire une marge interne à l'objet, la fonction ajoute alors simplement un numéro de valeur 0 à la matrice contenant l'information traitée sur ses « bords ».

Input	Kernel	Output
$\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 \\ 0 & 3 & 4 & 5 & 0 \\ 0 & 6 & 7 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$	$* \quad \begin{matrix} 0 & 1 \\ 2 & 3 \end{matrix} \quad =$	$\begin{matrix} 0 & 3 & 8 & 4 \\ 9 & 19 & 25 & 10 \\ 21 & 37 & 43 & 16 \\ 6 & 7 & 8 & 0 \end{matrix}$

Ce modèle discriminant lui, prendra en entrée deux variables. La première est la matrice d'entrée du modèle générateur, dans notre cas, il s'agit de la matrice (user, item), et le deuxième sa sortie, ici donc les recommandations. Son but comme auparavant est alors de déduire s'il s'agit d'une « image », soit une recommandation bien générée par le réseau précédent ou d'une « image », recommandation, issue de notre jeu de données.

Les deux modèles une fois instanciés, n'ont plus besoin que de deux choses, être mis en liens l'un avec l'autre et apprendre ensemble à résoudre leurs tâches distinctes.



Dans le cadre du modèle PixToPix, celui-ci fonctionnant de manière hybride entre les réseaux GAN et l’entraînement supervisé, il utilise une fonction de perte à minimiser légèrement différente de ces deux méthodes, c’est d’ailleurs plutôt une addition des fonctions de chaque type.

La première loss est celle qui correspond aux réseaux GAN soit comme vu auparavant une « cross entropie » entre les sorties Discriminés et leur taux de discriminations :

$$\text{Gan\_Loss} = \text{Ex} [\log(D(x))] + \text{Ez} [\log(1 - D(G(z)))]$$

Ensuite, on calcule l’écart entre la vérité à prédire et la matrice générée par le modèle. Cela va s’appeler la L1 loss. Elle se calcule ainsi : moyenne ( $| \text{Target} - \text{généré} |$ ).

Sois la moyenne des valeurs absolues des valeurs à prédire - les valeurs prédictes. Minimiser cet écart revient à trouver une sortie générée par le modèle égale à la valeur à prédire.

Ensuite, on pourra calculer la loss totale, qui se calcule part la gan\_loss + (Lambda \* L1\_loss). Lambda étant une constante qui prend par convention la valeur 100, cela équilibre l’impact des deux loss, il est possible de donner plus d’importance à l’une ou l’autre.

Ces loss seront utiles au générateur. Ce sont bien elles qui lui permettront d'apprendre via un calcul de descente de gradient ayant pour but étape après étape de tenter de minimiser les erreurs du modèle, en changeant la valeur des poids et biais du modèle, les paramètres du modèle, en fonction des résultats obtenus.

Mais avant, il faut tout autant permettre au modèle Discriminant d'apprendre.

Pour ce faire, on va commencer par simplement calculer la cross entropie entre les sorties « vraie » et leurs valeurs prédictives lors de la discrimination. Soit  $-1 * \log(y_{\text{prédict}})$ .

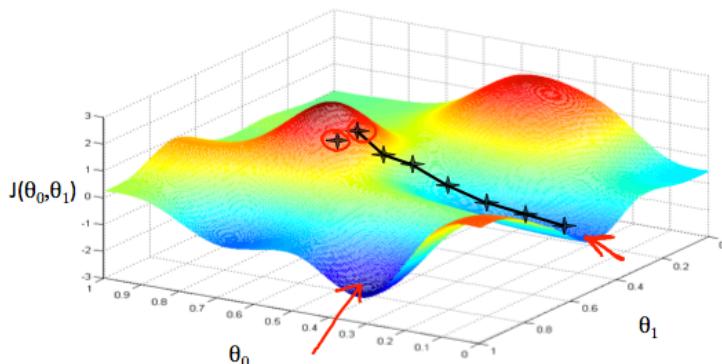
Puis on calcule de même la cross entropie entre les valeurs « fausses » et leurs valeurs prédictives par le modèle. Soit  $-0 * \log(y_{\text{prédict}})$ .

La somme des deux calculs permet d'obtenir la loss totale du Discriminant. La minimiser revient à maximiser la valeur prédictive accordée aux « vrais » par rapport aux « faux ».

---

## Descent de Gradient, Back propagation et Adam optimizer

Afin de minimiser les différentes fonctions loss, ou de perte. Une solution simple est d'utiliser un algorithme de descente de gradient. Ces algorithmes permettent de connaître le sens et la valeur de la variable que l'on doit modifier afin de minimiser une fonction. Leur fonctionnement correspond à une volonté de descendre une chaîne de montagne sans pour autant connaître le point le plus bas. Pour ce faire, on regarde autour de nous les différentes pentes et montés puis on avance jusqu'à atteindre un nouveau point de vue nous permettant de chercher à nouveau un point plus bas et de continuer.

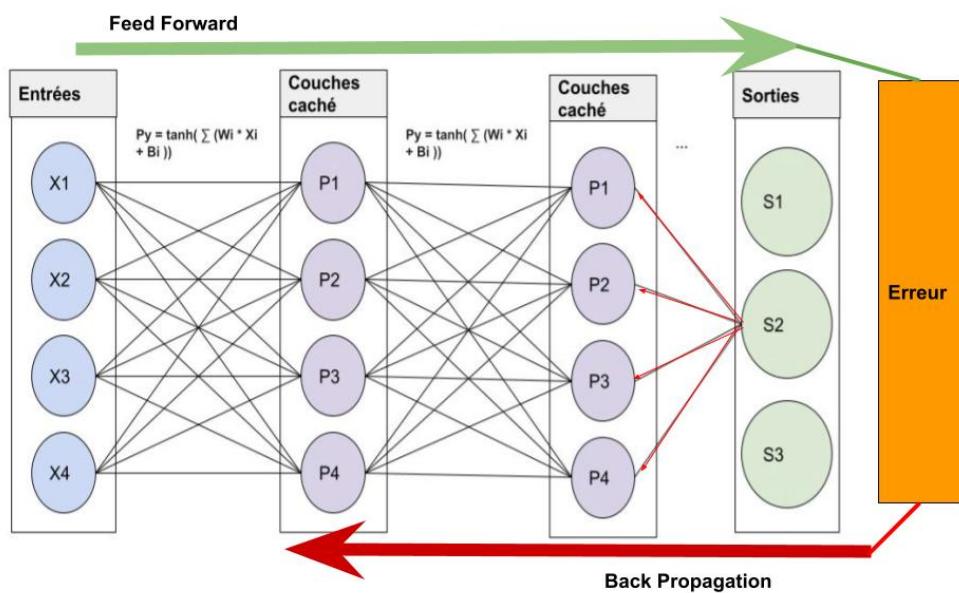


Mathématiquement, cela s'exprime par l'utilisation de la dérivée de la fonction qui nous donne la pente et son ampleur dans la fonction de coût ou de perte selon la valeur des poids, les paramètres du modèle. Ainsi, nous exprimons dans le modèle la question, de combien doit-on modifier un paramètre pour avoir moins d'erreurs ? Il faut aussi exprimer de combien de pas nous souhaitons nous déplacer ?, le taux d'apprentissage qui permet d'avancer.

Cela donne donc :

$$X_j = X_j - \alpha \left( D(f(x)) / Dx \right)$$

Soit, la valeur du poids actuelle d'un des paramètres du réseau - le taux d'apprentissage multiplié par la dérivée de la fonction de perte en fonction du paramètre x (le sens de la pente et son ampleur).



Se basant sur cette idée le deep learning préfère cependant utiliser les équations de la back propagation qui permettent cette fois de calculer le gradient non pas selon la dérivé du modèle selon chaque paramètre, mais celle de la couche n par rapport à la valeur du gradient de la couche n+1. Cela permet de tendre vers la solution en simplifiant largement les calculs, ceux-ci n'ont alors besoin que de propager l'information du gradient, calcul prenant en compte l'erreur de la couche de sortie vers la couche d'entrée par les équations que vous pouvez retrouver expliquées aux pages précédentes ainsi que en annexe reprenant l'idée derrière les réseaux de neurones.

Dans le cas du modèle PixToPix d'ailleurs, il s'agira d'utiliser l'algorithme d'optimisation Adam. Celui-ci tend plus vite vers une solution et est moins coûteux bien que parfois, il ne soit pas optimal pour trouver la meilleure réponse à la problématique. Le but de ce dernier est de minimiser les fonctions nécessaires et d'accélérer la vitesse à laquelle on tend vers une solution lors de l'entraînement. Les détails de cet optimiseur vous sont donnés dans cet article **Kingma, Adam: A method for stochastic optimization**<sup>44</sup>. Il utilise alors la méthode dite des « moments » dans laquelle on garde en mémoire la valeur des gradients de l'itération précédente afin d'avoir plus d'informations sur le sens de la variation de la fonction et de corriger selon cette information « itération - 1 » la valeur d'ajustement de l'itération courante et ainsi d'augmenter la vitesse de la convergence vers la solution. Cependant parfois cet algorithme a tendance à une fois une bonne solution trouvée de continuer sur sa lancé de mettre à jour le modèle. Malgré cela il semble à ce jour le plus apprécié des utilisateurs car il fait gagner un bon nombre d'itérations nécessaires lors de l'entraînement, il permet de réduire considérablement les ressources nécessaires de tout type à l'entraînement du réseau.

On a alors ces équations :

$$\left| \begin{array}{l} m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{array} \right| \quad \left| \begin{array}{l} E[m_t] = E[g_t] \\ E[v_t] = E[g_t^2] \end{array} \right.$$

Ce qui signifie que l'espérance du gradient « t » est égale l'espérance de la moyenne mobile « t » qui elle est égale à Beta1 fois la moyenne m t-1 additionné à 1 moins la valeur de beta1 multiplié par le gradient de l'étape « t ». de même pour la variance et son espérance avec cette fois beta deux et la variance v t-1, v t.

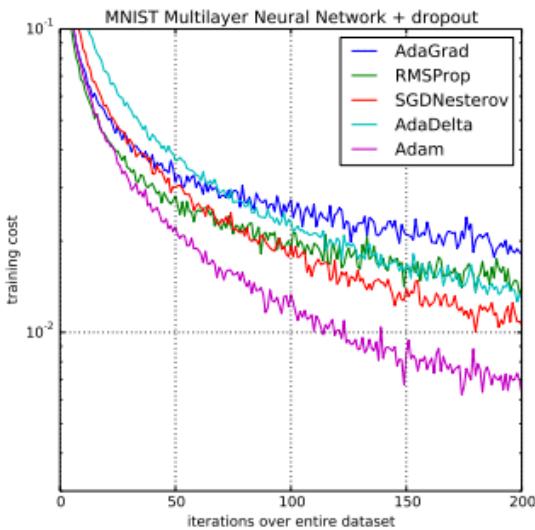
Ainsi, le gradient est calculé non seulement selon le gradient actuel, itération t, mais aussi celui à l'étape t-1, itération précédente.

$$\left| \begin{array}{l} \hat{m}_t = \frac{m_t}{1-\beta_1} \\ \hat{v}_t = \frac{v_t}{1-\beta_2} \end{array} \right| \quad w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

<sup>44</sup> Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Une fois ces calculs faits, on réutilise leurs résultats dans ces dernières équations qui permettent d'ajuster les valeurs  $m_t$  et  $v_t$  afin de devenir estimateur sans biais du gradient puis de mettre à jour la valeur des poids selon ces derniers estimateurs.

Epsilon étant un paramètre configurable du modèle et de même pour le « learning rate nu ».



Résultat de l'optimiser sur un dataset classique de test, mmnist, dataset d'image contenant de nombreux chiffres écrit à la main et leurs équivalents numériques.

## Dataset et Preprocessing, Quelles données utiliser ?

### Selon les principes du modèle il nous faut obtenir plusieurs choses :

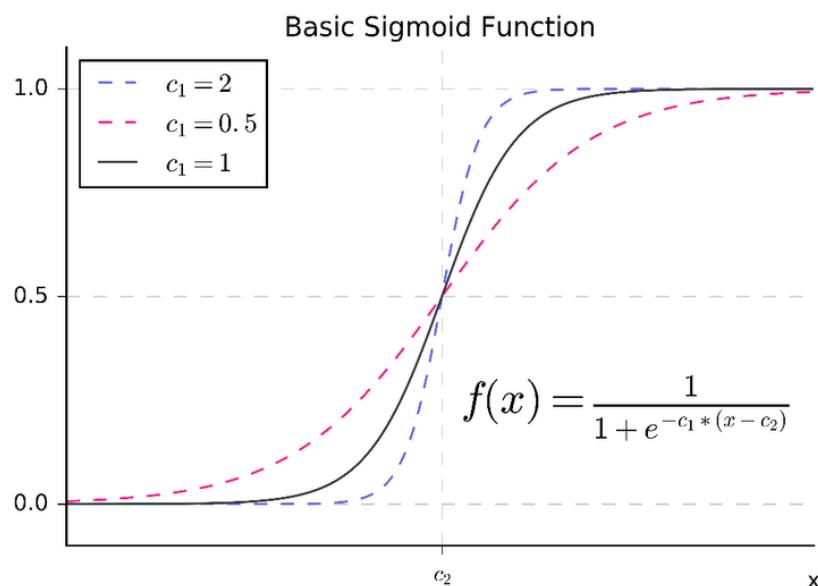
- des données
- une représentation de ce que l'on attend de l'algorithme
- une version du modèle adapté à notre tâche

Pour ce faire, en explorant quelques modèles, celui le plus adapté semble être un algorithme collaboratif. Car les données croisées que l'on obtient des utilisateurs par rapport aux items permettent de facilement obtenir une large matrice bien pratique pour les réseaux convolutionnels. De plus, il est possible d'y ajouter d'autres informations comme, la date de visionnage, le genre des films et autres, afin de compléter les informations de préférences des

utilisateurs. L'article « *Scalable deep learning-based recommendation systems* »<sup>45</sup> montre bien comment il est possible d'utiliser dans un tel format ce genre de données. Dans ce cadre, une matrice similaire est utilisée, bien que le preprocessing de l'article ne soit pas appliqué ainsi.

Ensuite un dataset bien connu des concours Kaggle ayant déjà fait ses preuves et servant régulièrement de base comparative est utilisé. Le dataset Movie 100k dans lequel un grand nombre d'utilisateurs ont noté au minimum 20 films (items) de 0 à 5 étoiles renseignant leurs dates de visionnage et d'autres informations relatives aux films consultés.

Puis finalement les adaptations en lien direct avec le modèle, elles, sont légères. Un réseau moins complexe, car la tâche est plus simple, et une sortie sigmoid afin de donner un résultat semblable à une probabilité (les bons films seront alors notés proches de 1, les mauvais films proches de 0, dans un intervalle 0 à 1).



<sup>45</sup> Lee, H., & Lee, J. (2019). Scalable deep learning-based recommendation systems. *ICT Express*, 5(2), 84-88.

## Mise en place de l'algorithme :

Le dataset movie 100k possède plusieurs versions l'une avec +-1000 utilisateurs notant 1682 films et une autre version avec des millions d'utilisateurs pour ces mêmes films. Pour chacune de ces dernières, les utilisateurs ont noté au moins 20 films. Quant au contenu du jeu de données, il décrit les films, leurs genres (0 ou 1 si le film est du genre associé), la date de visionnage par l'utilisateur, sa note et d'autres infos sur l'utilisateur.

Ce qui nous intéresse ici, c'est principalement. Les notes des utilisateurs sur chacun des films. Bien que les autres données restent exploitables. Dans le cadre d'un algorithme collaboratif, le but reste de comparer les utilisateurs entre eux afin d'en dégager des préférences probables, car d'autres utilisateurs ayant les mêmes préférences recommandent aussi ce film.

Dans le dataset ce qui indique qu'un utilisateur à apprécier un film est la note qu'il lui attribut. Un simple 0 ou 1 si la personne à visionné ou non le film peut suffire, mais il est préférable afin d'avoir des nuances de préférences de garder les notes de 0 à 5 sur chaque film. Il faut, pour obtenir la matrice d'entrée, croiser les données des notes selon les utilisateurs et leurs films associés. On peut faire de même avec la date de visionnage.

item id	1	2	3	4	5	6	7	8	9	10
user id										
1	5.0	3.0	4.0	3.0	3.0	5.0	4.0	1.0	5.0	3.0
2	4.0	NaN	2.0							
3	NaN									
4	NaN									
5	4.0	3.0	NaN							

item id	1	2	3	4	5	6
user id						
1	0.012986	0.116823	0.205703	0.11682	0.80956	0.684587
2	0.744867	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN
5	0.049081	0.049097	NaN	NaN	NaN	NaN

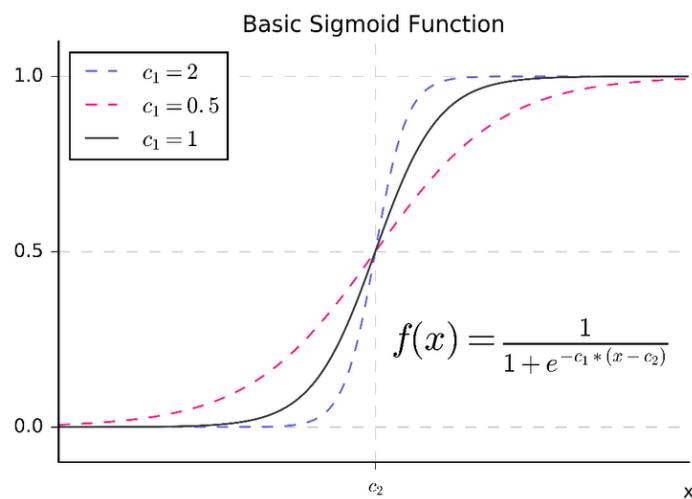
En procédant ainsi, on obtient bien deux matrices de taille (utilisateur, item) contenant nos données de préférences. Cependant, on remarque un grand nombre de vide. Bien que les algorithmes classiques collaboratifs exploitent ainsi les données se basant alors soit sur des statistiques soit sur des modèles hybrides afin d'exploiter ces matrices. Dans notre cas, afin de ne pas manquer d'informations, une matrice contenant la note moyenne des films est utilisée, bien qu'afin d'éviter un biais de popularité, un recentrage de cette moyenne est utilisé (le biais de popularité étant un biais dans lequel plus un film à une note élevée plus il sera recommandé et gagnera en popularité augmentant du même coup ses chances d'être à nouveau recommandé, etc.). Ainsi, un film ayant

une bonne note en moyenne a plus de chance d'être intéressant cependant, il ne le devient pas forcément uniquement pour cette raison.

Ainsi une note de 5 devient une note moyenne de note + seuil / 2 soit, pour un seuil de 2 la note obtenue est de 3,5 alors qu'une note moyenne de 4 devient une note de 3, etc. Ainsi, les fortes notes sont favorisées et connues du modèle sans pour autant être le facteur de décision principal. De plus, cette matrice de moyennes recentrées est utile afin de pousser le modèle à tenter de prendre des décisions même lorsqu'un film est très peu noté comme le ferait un bruit gaussien durant l'entraînement de certains G.A.N., évitant alors des variables d'entrées trop souvent nulles et ajoutant de la créativité aléatoire dans le modèle.

Ensuite, le modèle prend en entrée les données connues de l'utilisateur avec un pourcentage aléatoire de perte d'informations. Puis il prend en sortie la matrice de moyennes recentrées jointe aux informations de préférences perdues aléatoirement ainsi que celles connues du modèle, afin de garder cette probabilité d'apprécier ou non un film. La sortie est remise sous forme sigmoïde par le calcul  $Y_{ij} = 1 / \exp(1 + \exp(-X_{ij}))$  avec X la note connue de l'utilisateur sinon la moyenne recentrée du film -3 / 3 (afin de bien avoir un numérique entre 0 et 1 les notes étant de 1 à 5).

Ici, on remarque que recentrer la moyenne d'un film bien noté permet alors d'avoir en sigmoid une note de recommandation probable (0.5 à 0.7) sans pour autant devenir la priorité du modèle (0.8 à 1).



Une fois, cela fait, il ne reste plus qu'à assembler le dataset de manière à obtenir suffisamment de données distinctes. Le plus simple est de prendre aléatoirement des colonnes et des lignes dans la matrice (utilisateur, item) de manière à toujours garder la même taille de matrice (128,128). Ainsi, il reste possible de toujours pouvoir assembler de nouvelles données qui ne seront pas pour autant semblables à celle déjà présentes dans le dataset et donc d'éviter l'overfitting et l'underfitting **Hawkins, D. M. (2004)**<sup>46</sup>.

Finalement, la matrice des entrées devient avec toutes les informations assemblées une matrice (128,128,3) de (user, item) par rapport aux notes, à la date de visionnage et à la moyenne du film.

Afin de tester plusieurs modèles, d'autres traitements appliqués au dataset ont été utilisés, dont d'ailleurs plutôt qu'une matrice de moyenne un simple bruit gausien en entrée. Ainsi que des dataset ne remplaçant pas la vérité non connue par une moyenne. Il semble dans les résultats que la plus grande précision sur la vérité connue reste le dataset évoqué ci-dessus avec un recentrage des moyennes pour les valeurs manquantes cependant, cela a tendance à rajouter un biais de popularité lorsque l'on étudie uniquement les films ayant les plus gros scores et donc il est largement préférable de finalement plutôt utiliser le bruit gausien ainsi que des valeurs manquantes laissées nulles pour les films non connus de l'utilisateur.



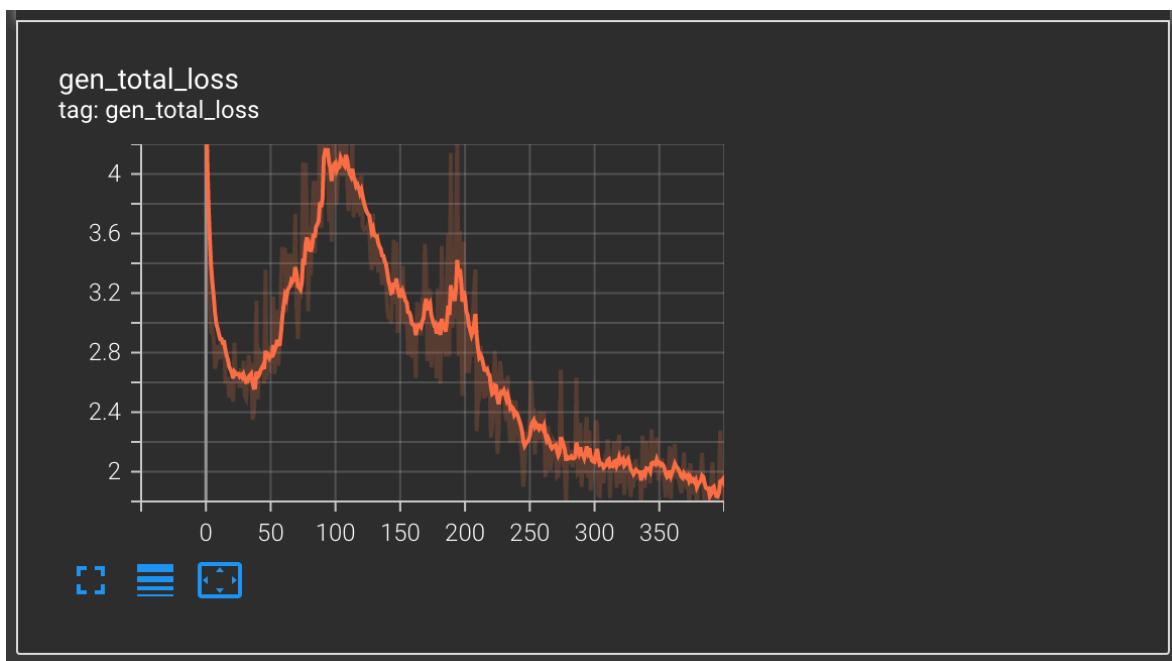
Exemple d'une matrice 2D formant un bruit gausien.

<sup>46</sup> **Hawkins, D. M. (2004).** The problem of overfitting. Journal of chemical information and computer sciences, 44(1), 1-12.

## Entrainement :

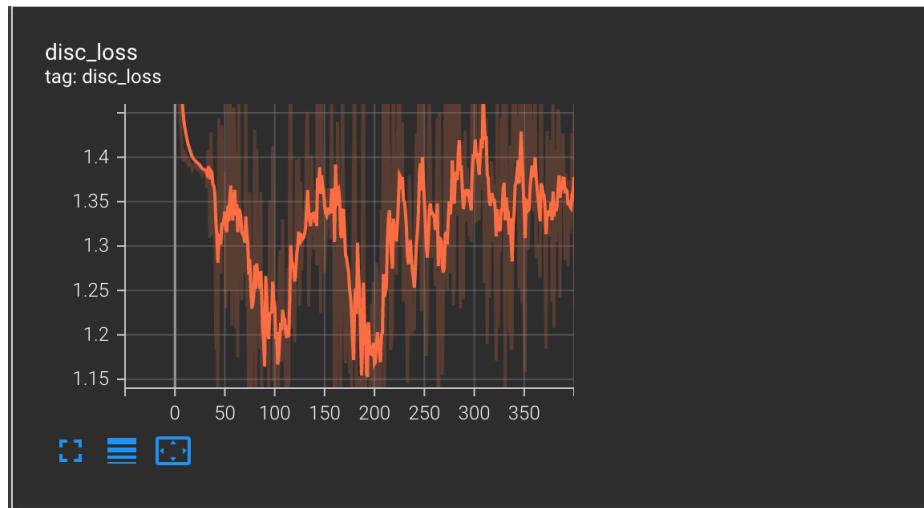
### Dataset avec moyenne recentré :

Pendant l'entraînement, les modèles de type GAN possédant un générateur et un discriminant ont tendance à être en concurrence l'un avec l'autre. En effet alors que la loss de l'un signifie sa capacité à ne pas faire d'erreurs significatives, elle signifie pour l'autre au contraire son incapacité à résoudre la problématique qui lui est posée. Cependant, dans le cadre de Pix to Pix, on trouve aussi la Loss responsable de l'entraînement supervisé du modèle. Les deux modèles ne se contentent donc pas de s'affronter, affrontement faisant augmenter pour chacun des modèles la loss de l'autre modèle au fur de l'entraînement, donnant donc des effets de vagues variant selon le modèle remportant le duel à l'étape donnée. Le but est alors que les deux modèles finissent par converger vers un duel équilibré avec de faibles variations.

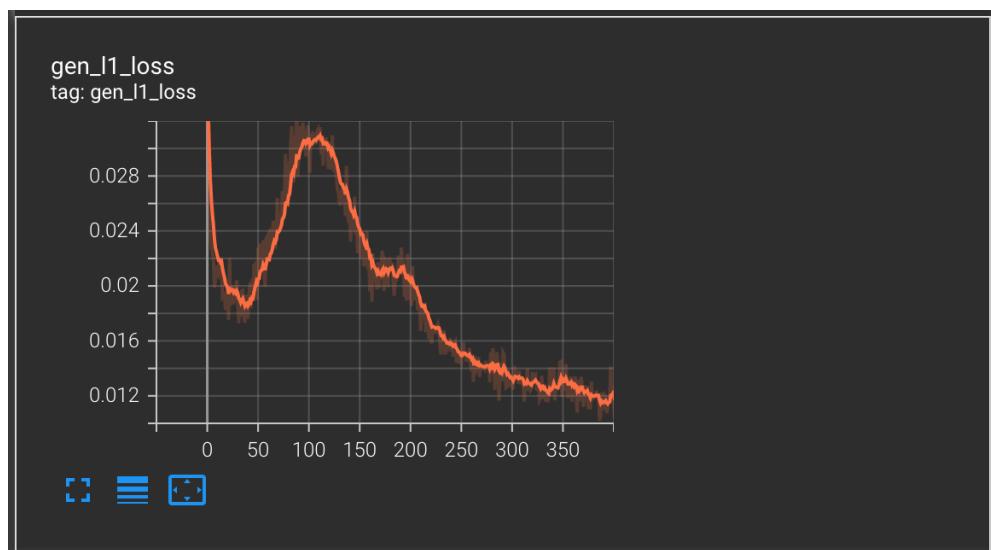


Dans l'image ci-dessus, on peut voir à quoi ressemble la phase d'entraînement du modèle. Ici, il s'agit de la loss totale du générateur, prenant en compte à la fois les résultats du discriminant, mais aussi la partie supervisée. Cela n'est pas le seul résultat observable comme nous l'avons vu, nous avons quatre loss à observer. On remarque que la loss se réduit bien, ici le duel s'éclipse presque, bien que présent au début, la partie supervisée de l'entraînement semble fortement participer à la réduction de l'erreur malgré la loss concurrentielle de type gan. Ici, le but est bien de minimiser l'erreur.

Alors que dans l'image ci-dessus, elle, étant la loss de la partie génératrice de type GAN, on remarque bien la présence du duel, ici la partie supervisée est éclipsée du calcul. Notre modèle tend bien vers une variance plus faible tout en donnant de meilleurs résultats. On peut donc valider cette partie du modèle.



De même, on remarque presque l'opposé de la courbe précédente pour la partie Discriminante du modèle. En effet le calcul de l'un utilisant l'autre dans le sens inverse cela est très logique.



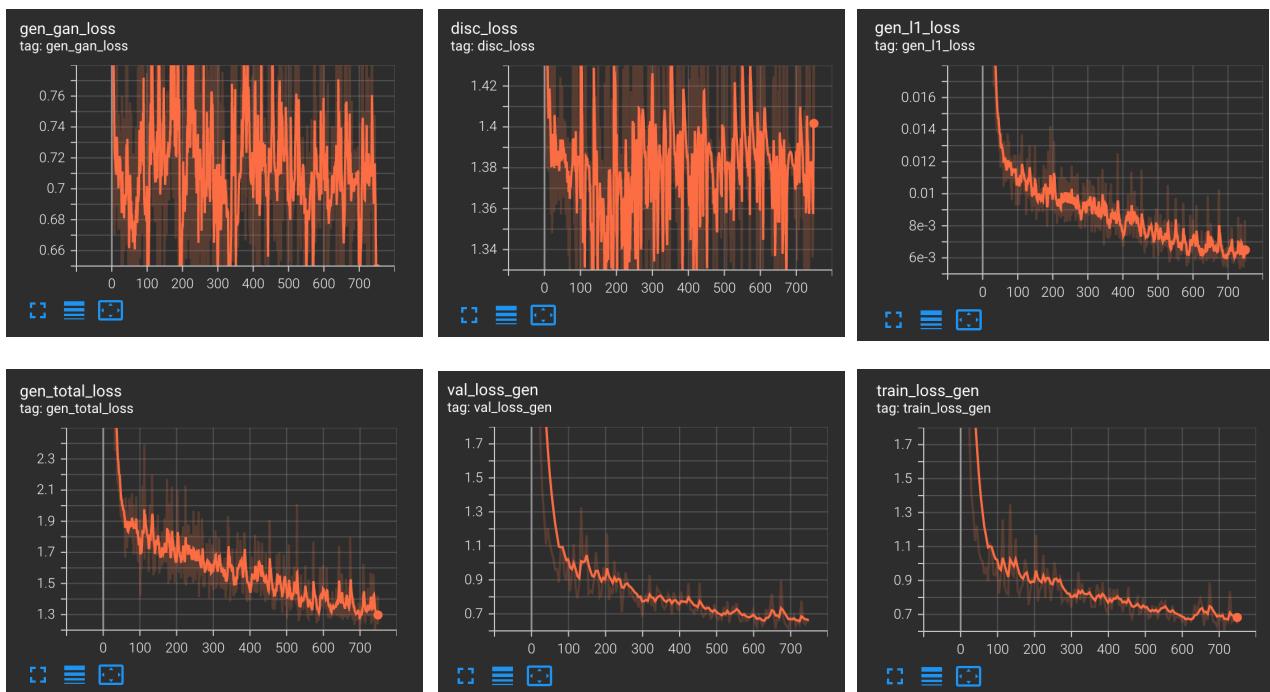
Quant à la dernière loss, celle de la partie supervisée, on remarque bien que cette dernière tend vers une solution de manière assez classique bien que le second pic montre bien que le modèle ne se base pas que sur cette dernière afin de se mettre à jour, sinon il aurait été illogique pour le modèle de continuer dans ce sens durant plusieurs itérations alors que ce dernier augmente l'erreur de type supervisé.

---

## Dataset avec uniquement un bruit Gausien :

Pour le dataset utilisant uniquement un bruit gausien et les données connues de l'utilisateur (avec perte d'information) en entrée et les données connues en sortie (sans perte). Les loss ne sont pas bien différentes à l'exception de l'ajout d'une loss plus classique qui permet de mieux interpréter la finalité du modèle. Celle de validation par rapport à la total GAN loss sur une partie de dataset.

Cela n'a pas forcément de sens dans des modèles GAN cependant PIX2PIX étant un modèle semi GAN semi supervisé, il possède un type de loss nous le permettant.



(Ici ces deux loss n'ont été calculés que toutes les 20 étapes et représentent la total GAN loss et donc pas uniquement la loss Supervisé ce qui explique les variations )

Cela nous permet de constater non seulement que le modèle apprend bien, mais aussi qu'il généralise bien, car il ne se contente pas d'obtenir des résultats sur le jeu de données sur lequel il s'entraîne. Bien souvent, on remarque aussi via ses courbes si le modèle underfit, n'arrive pas à résoudre la problématique ou overfit, généralise très mal. Ces défauts peuvent chacun avoir de nombreuses origines. Parmi elles, des choses classique comme un mauvais jeu de donnée, un algorithme pas assez ou trop complexe, un mauvais paramètre du modèle (learning rate, beta, etc.) ainsi que de nombreuses autres possibilités qu'il faut en cas de problème identifier et traiter. Ce qui ne semble pas être notre cas.

---

## Résultat :

### **Test de l'hypothèse de pertinence des recommandations :**

On a vu que de nombreux algorithmes de recommandations existent. Que chacun d'entre eux brille autant dans sa manière de s'exprimer que dans les données qu'il utilisera pour ce faire. Cependant, nous avons aussi vu que de nombreux biais animent ces derniers pouvant souvent frustrer, mal orienter ou encore restreindre les utilisateurs. Nous nous sommes attardés ensemble sur l'un d'eux que nous traitons ici. Le biais responsable des effets de bulles qui ont tendance à enfermer l'utilisateur dans des recommandations trop similaire entre elles. Dès lors, il devient impossible pour l'utilisateur de sortir de ce cercle vicieux où rien d'original ne lui sera proposé et donc biais dans lequel il continuera de consommer le même contenu. Ils nous faut dès lors des solutions permettant à l'utilisateur d'explorer de nouveaux horizons sans pour autant soumettre les recommandations à un pur aléatoire, ce qui serait non seulement risqué, mais aussi plus probablement ignoré de l'utilisateur. Pour ce faire nous nous sommes attardé sur le concept de créativité, nous avons cherché ensemble ce qui l'anime et ce qui fait la force des liens nouveaux que ce concept crée ainsi que les différentes manières dont elle s'exprime. Nous en avons retenu qu'exploiter la créativité exploratoire semble être un bon moyen de toujours s'assurer que les recommandations puissent sortir du carcan des effets de bulles, car cette dernière ne se baserait pas sur une mise en liens entre l'utilisateur et les films qui pourrait alors nous limiter à ce que l'on sait déjà sur l'utilisateur, mais plutôt sur une exploration de la structure spatiale de la filmographie disponible. Cela poussant donc l'algorithme à découvrir de nouveaux goûts utilisateur encore inconnus tout en restant pertinent.

Dès lors, nous avons cherché quelque chose de semblable à ce concept dans la logique de nos algorithmes. Cela nous a amené vers le deep learning, et principalement vers les réseaux de neurones créés à des fins artistiques. Les generative adversarial network, ou réseaux génératifs par adversité. Ces derniers brillent par leurs capacités à généraliser une problématique en n'associant pas uniquement une entrée à une sortie, mais en associant plutôt une entrée à un ensemble de résultats possibles tout aussi valables, en effet lorsqu'il s'agit d'art difficile de dire qu'un chat doit être roux ou gris, mais il reste une représentation valable d'un chat même de manière abstraite. Dès lors à chaque itération de notre algorithme le panel complet de recommandation peut varier tout en gardant une forme de cohérence artistique répondant à la question, est-ce bien une recommandation

convenable ? Sans uniquement se lier à l'historique de l'utilisateur pour autant, ce qui ajoute alors bien un caractère exploratoire créatif à notre algorithme.

Ainsi une fois les données assemblées et l'algorithme adapté entraîné, il ne nous reste plus qu'à vérifier la cohérence de nos recommandations.

Pour ce faire, la décision qui a été prise est de se baser uniquement sur les données déjà connues, car comme expliqué auparavant une partie des connaissances du comportement de l'utilisateur a été volontairement supprimée ainsi le but du jeu est de constater si ces dernières ont bien été retrouvées de manière cohérente par rapport aux notes que l'utilisateur a réellement mis à ces films. Et donc finalement nous reprenons les bonnes et mauvaises notes non connues de l'algorithme et vérifions le taux de cohérence de chaque proposition liée à ses films selon notre algorithme par rapport à la vérité connue.

```
confusion matrix :  
True negative : 2 False positive : 348  
False negative : 0 True positive : 631
```

```
True negative : 0.00 False positive : 0.35  
False negative : 0.00 True positive : 0.64
```

```
confusion matrix :  
True negative : 0 False positive : 316  
False negative : 0 True positive : 674
```

```
True negative : 0.00 False positive : 0.32  
False negative : 0.00 True positive : 0.68
```

Ainsi on obtient :

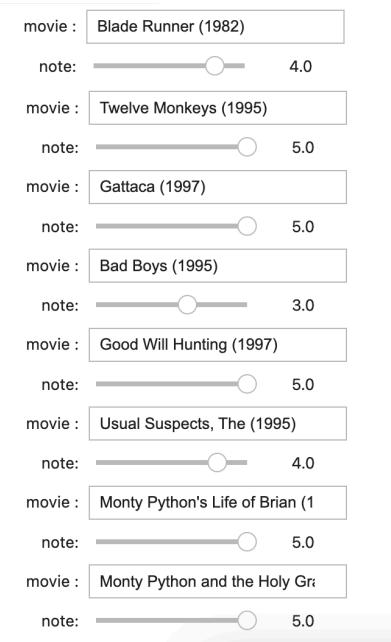
Un taux de correspondance de 65% (avec un pic à 68% lors de l'entraînement avec quelques modifications des paramètres du modèle) pour les films ayant été notés au-delà de 4 étoiles, dont 35% d'erreurs sur des films notés 3 étoiles ou moins par l'utilisateur (généralement 3 étoiles). Bien que ces chiffres semblent bas, il faut rappeler qu'ici tous les films présentés ont obtenu une notation ce qui permet dès lors d'avoir une échelle de préférence sûre de nombreux films, ainsi les films mal notés ne sont au final pas présentés à l'utilisateur le plus souvent, mais permettent tout de même d'obtenir plus de recommandations si nécessaire. Cela n'est pas le cas de la plupart des moteurs de recommandations, eux notés uniquement sur le score de note des recommandations obtenant le meilleur score.

```
confusion matrix :  
True negative : 0 False positive : 62616  
False negative : 0 True positive : 92931
```

```
True negative : 0.00 False positive : 0.40  
False negative : 0.00 True positive : 0.60
```

Ce score est associé au modèle utilisant les moyennes recentrées comme valeur manquante. Il descend à 60%-62% lorsqu'on le teste sur bien plus de données. Quant à celui utilisant un bruit gaussien en entrée et ne remplaçant pas les valeurs manquantes. Nous obtenons pour lui un score de 59%-60%, autrement dit le biais de popularité ne nous apporte que 1% à 3% de précision. On constate donc que ce dernier semble meilleur lorsque l'on recherche la créativité que nous avons décrit.

### Exemple de recommandation possible du modèle :



Admettons par exemple les films notées présent à gauche.

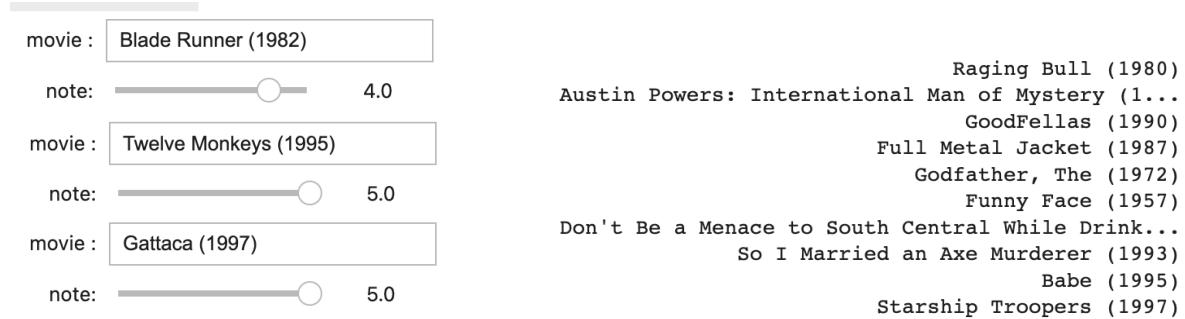
Blue Sky (1994)  
Good, The Bad and The Ugly, The (1966)  
Apollo 13 (1995)  
So I Married an Axe Murderer (1993)  
Starship Troopers (1997)  
Dirty Dancing (1987)  
Wrong Trousers, The (1993)  
Mighty Aphrodite (1995)  
Heat (1995)  
Babe (1995)

Le modèle nous recommande au premier appel les films ci-dessus.

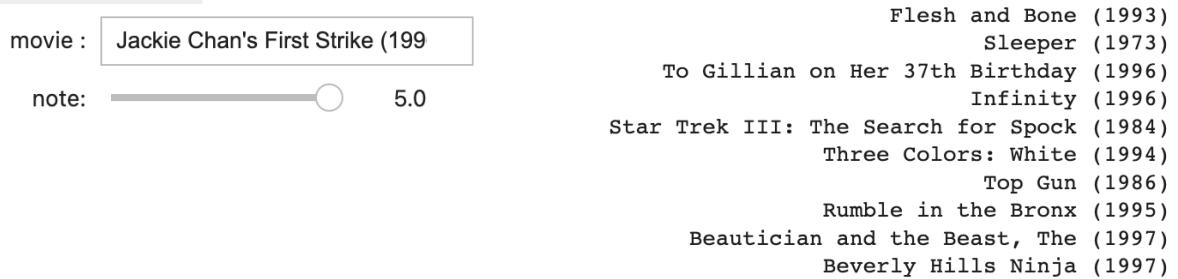
Mais au second appellent ceux-ci :

Ridicule (1996)  
Red Firecracker, Green Firecracker (1994)  
Apollo 13 (1995)  
So I Married an Axe Murderer (1993)  
Babe (1995)  
Starship Troopers (1997)  
Heat (1995)  
Mighty Aphrodite (1995)  
Dirty Dancing (1987)  
Wrong Trousers, The (1993)

On remarque que certains films reviennent, cela arrive bien souvent lorsque de nombreux films notés sont connus de l'utilisateur. Cependant, bien des choix ont changé dans notre top 10.



En utilisant moins de films en entrée, le modèle est tout de même capable de faire des prédictions.



Cela vaut aussi lorsque une seule notation est utilisée. Par ailleurs le coté collaboratif et Gan de notre algorithme permet bien ici d'avoir un panel de recommandation assez vaste. Bien que la nature des algorithmes collaboratifs est combinatoire en terme de créativité, notre modèle Gan lors de son entraînement apprend aussi à explorer de lui-même certaines possibilités.

### **Test de l'hypothèse Diversité des recommandations :**

Cette hypothèse peut être vérifiée en testant comme on peut le faire lors du test proposé par **Nguyen, T. T., Hui, P. M., Harper, F. M., Terveen, L., & Konstan, J. A. (2014, April).**<sup>47</sup> des effets bulles, en utilisant la notion de diversité (distance euclidienne entre les recommandations proposées cette fois) et en les comparant, soit à d'autres moteurs de recommandations, soit à un aléatoire.

En effet si notre répartition de distance intra recommandations est similaire à un aléatoire alors il est possible d'affirmer que le moteur fait preuve d'une grande diversité de recommandation.

Sur 300 observations on obtient :

```
test répartition des distances intra groupes de recommandations
Ttest_indResult(statistic=1.4099084350897388, pvalue=0.15908659836530162)
```

ce qui valide notre hypothèse de diversité de contenu par rapport à la répartition de diversité d'un aléatoire.

### **Test de l'hypothèse correction du biais des « echo chamber » ou « filter bubble » :**

Afin de tester cette hypothèse l'idée de quantification de ces derniers de l'article **Nguyen, T. T., Hui, P. M., Harper, F. M., Terveen, L., & Konstan, J. A. (2014, April).**<sup>48</sup> est utilisé. Cependant par manque d'utilisateurs, ici, un pur aléatoire de choix est associé à l'historique de consommation du groupe de contrôle et le premier choix du moteur de recommandation est associé au groupe de test de notre hypothèse.

Ici l'idée est alors de constater un changement dans la diversité de consommation au fur et à mesure de l'exposition aux recommandations de notre système. Pour ce faire on compare la diversité de consommation d'un groupe non recommandé avec un groupe recommandé. La distance euclidienne sur le dataset dit génome des films est utilisée, ce dataset contient la liste des tags représentant un film et ainsi permet de mieux connaître le contenu d'un film selon le ressentit des utilisateurs en

---

<sup>47</sup> Nguyen, T. T., Hui, P. M., Harper, F. M., Terveen, L., & Konstan, J. A. (2014, April). Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web* (pp. 677-686).

<sup>48</sup> Nguyen, T. T., Hui, P. M., Harper, F. M., Terveen, L., & Konstan, J. A. (2014, April). Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web* (pp. 677-686).

fonction du score de relation fournit dans le dataset entre un film et chacun de ses tags. Dès lors la distance euclidienne de ce films par rapport aux autres nous permet d'interpréter cette notion de proximité d'un film à un autre et donc, si l'on compare ensemble les films de l'historique d'un utilisateur on obtient cette notion de diversité de consommation.

Ensuite grâce à un simple T-test de comparaison des répartitions on peut déterminer si nos variances sont similaires et donc si la diversité de consommation du groupe non soumis au moteur de recherche est comparable à celui du groupe lui soumis à ce dernier, cela validant ou non notre hypothèse.

#### Résultat au test sur 30 observation:

```
number of movie seen 0 :  
Ttest_indResult(statistic=-2.584377999779522, pvalue=0.012294684065305777)  
number of movie seen 1 :  
Ttest_indResult(statistic=-1.6107220588975102, pvalue=0.11266977780313117)  
number of movie seen 2 :  
Ttest_indResult(statistic=-0.7822642818160304, pvalue=0.43724136478901854)  
number of movie seen 3 :  
Ttest_indResult(statistic=-0.9280168217226409, pvalue=0.35724484621363173)  
number of movie seen 4 :  
Ttest_indResult(statistic=-0.6764712279421078, pvalue=0.5014304088542684)  
number of movie seen 5 :  
Ttest_indResult(statistic=-1.065143806972031, pvalue=0.2912250348885435)  
number of movie seen 6 :  
Ttest_indResult(statistic=-0.8033363973451892, pvalue=0.42506035835549194)  
number of movie seen 7 :  
Ttest_indResult(statistic=-0.33525522498201255, pvalue=0.738642156949862)  
number of movie seen 8 :  
Ttest_indResult(statistic=-0.7455240056214761, pvalue=0.4589657501369562)  
number of movie seen 9 :  
Ttest_indResult(statistic=-0.38054595783865797, pvalue=0.704930386796663)
```

Notre test nous permet donc de valider notre hypothèse, bien qu'il faudrait pour être rigoureux tester cela sur des utilisateurs bien réel. On constate que la diversité de consommation du groupe recommandé semble suffisamment proche du groupe de contrôle pour affirmer qu'ici aucun effet bulle ne semble détecté par notre test. (**test réalisé sur 30 observation générée**)

le test semble cependant rejeter l'hypothèse « H0 : répartition similaire » lorsque l'on augmente à 100 le nombre d'observations cependant tant que le moteur n'est pas tester sur des utilisateurs, impossible de rejeter l'hypothèse, et 30 observations étant suffisant pour les conditions du test cela laisse à réfléchir.

```
number of movie seen 9 :  
Ttest_indResult(statistic=-2.952492603640769, pvalue=0.0035332935689339285)
```

## Conclusion :

Bien que les capacités exploratoires de l'algorithme sont améliorées de par la nature des algorithmes GAN, il est toujours possible de tomber dans un effet de bulle de par la partie combinatoire de l'algorithme venant de l'entraînement supervisé. L'idéal serait alors de tester le modèle après plusieurs mois d'utilisation, car seul un véritable comportement client lié aux recommandations faites peut vérifier l'hypothèse de résolution des effets bulles filtrantes. En effet, une simulation de choix, même basé sur nos recommandations, ne peut suivre tout à fait la logique d'une masse d'utilisateurs et rajouterait au fur et à mesure des itérations des biais d'analyse, quant à tester la diversité, cela ne justifierait en rien une dissipation de ce biais, bien que cela rende le moteur de recommandation plus satisfaisant pour l'utilisateur. De plus, l'algorithme présent dans ce mémoire réduit la précision de la recommandation, il est donc possible que cela réduise les interactions des utilisateurs sur le site, bien que la précision semble tout de même suffisante pour garantir un certain niveau de satisfaction et que de plus l'algorithme est calibré pour choisir des recommandations satisfaisant ce dernier de manière optimale et non simplement sur la maximisation des interactions et ou conversion de ce dernier. Afin d'améliorer dans un sens ou dans l'autre l'algorithme, on peut soit changer les données d'entrée, soit changer l'algorithme, en faisant varier le nombre de paramètres ou les hypers paramètres. Il est possible de recalibrer le modèle au besoin. Ce faisant on peut alors imaginer qu'un réseau de neurones spécialisé dans les parcours de graphes serait très efficient pour mettre en liens facilement les données peu importe leurs natures. Le tout pourrait de plus s'adapter à de nombreux types d'algorithmes de recommandations collaboratifs, hybride, content based ou autres. Finalement, il peut être aussi préférable que l'algorithme ne traite qu'une partie des recommandations, ainsi, celles provenant des algorithmes classiques sont mises en avant selon leurs pertinences puis sont combinés à certaines recommandations plus exploratoires ou inversement afin de rajouter de la diversité et une plus grande robustesse probable aux effets bulles. Cela permettant alors d'assurer un fort taux de conversion tout en sollicitant la curiosité de l'utilisateur lorsqu'il le souhaite. On peut alors imaginer des recommandations plus ouvertes sur les pages d'accueil lorsque le client n'a pas encore spécifié la gamme de produits recherchés ou parmi les pages de produits par catégories afin d'éviter un trop fort biais de popularité, le cold start effect et bien d'autres.

Finalement, cet algorithme, détournant le but originel des réseaux GAN (créations artistiques) permet de franchir une limite dans la manière d'explorer de nouvelles recommandations que même des algorithmes d'apprentissage par renforcement éviterait, car à leurs yeux, ses recommandations seraient vite abandonnées par manque de gain à la futur valut, gain par la récompense de l'interaction d'un utilisateur avec l'item recommandé et ceux même en tant que choix secondaire. Mais surtout, il est désormais nécessaire afin de valider le modèle, de le tester à une plus grande échelle, sur des utilisateurs non influencés ou dans la limite du possible peut influencés par d'autres algorithmes. En se basant sur les calculs proposés dans l'article : « Exploring the filter bubble : the effect of using recommender systems on content diversity. In Proceedings of the 23rd international conference on World wide web »<sup>49</sup> il sera alors possible de constater l'évolution du comportement utilisateur dans le contexte de notre algorithme de recommandation.

### **Question de recherche établit :**

~~Comment ce définissent les effets de bulles et quel sont leur manière de s'exprimer en terme algorithmique ?~~ **Résolue**

### **Comment quantifier ces biais ? Trouvé, doit être testé sur des utilisateurs**

Pour ce faire il est nécessaire d'entrainer le modèle sur la version la plus récente de movie 100k contenant 25 millions de films. D'appliquer le calcul de distance euclidienne entre les films de l'historique utilisateur et le vecteur « génome » des films contenant la matrice (film, tag ,score de relation ) nous décrivant pour chaque tag si un film est lié à celui-ci ou non dans un score. Cela nous donne donc une notion de distance moyenne entre les films de l'historique et donc de diversité de consommation des films du dataset. Grace à un AB test entre des utilisateur n'utilisant pas de moteur de recommandations et d'autre utilisant le notre (ou en en comparant plusieurs) il serait alors possible de déterminer la capacité du modèle à éviter les « echo chamber » ou plutôt de ne pas réduire la diversité de consommation des utilisateurs lorsque ce ci ne sont pas recommandés.

( Le code est fournit pour le calculer cependant le manque d'utilisateurs et de possibilités de le tester ne permet pas d'obtenir des résultats avec un minimum de biais, pouvant ici au mieux simuler un utilisateur et ses choix )

---

<sup>49</sup> Nguyen, T. T., Hui, P. M., Harper, F. M., Terveen, L., & Konstan, J. A. (2014, April). Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web* (pp. 677-686).

~~et finalement, quels mécanismes permettraient d'en sortir et comment l'exprimer dans notre algorithme ? Résolue~~

- Notre algorithme résout les effets de bulles filtrante.

**H0 : Oui l'algorithme évite des effets de bulles filtrantes ou dites d'echo chamber.**

H1 : Non il n'est pas vérifier que l'algorithme évite des bulles filtrantes

*Doit être testé sur plusieurs mois lors de X itérations de l'algorithme auprès de Y utilisateurs*

- Notre algorithme est pertinent dans ses recommandations.

**H0 : Oui l'algorithme a une bonne précision dans la recommandation**

H1 : Non l'algorithme n'a pas une bonne précision dans la recommandation

- Notre algorithme permet une grande diversité inter recommandation.

**H0 : Oui l'algorithme diversifie suffisamment les contenus recommandés**

H1 : Non l'algorithme ne diversifie pas suffisamment les contenus recommandés

## **Voies de recherche :**

Les GAN, bien que déjà dépassés dans certains de leurs propres domaines, ont beaucoup à apporter de par leurs manières de traiter les données, car en effet, lorsqu'une tâche ne peut être jugée par l'unicité d'un seul résultat concret, les GAN permettent eux de lier un ensemble de résultats à une même tâche. Ils restent donc de bonnes méthodes permettant d'obtenir des résultats concrets, polyvalents et étonnamment artistiques. Désormais, il peut être intéressant d'appliquer les méthodes GAN à d'autres types d'algorithmes traitants plus que des matrices semblables à des images tels que des algorithmes de parcours de graphes ou les tous récents « transformer » dans le deep Learning qui ont l'incroyable capacité d'apprendre de plusieurs types de données avec le même réseau et les mêmes poids. Car les modèles GAN peuvent être utiles si l'on souhaite éviter de modifier le modèle pour de mauvaises raisons lorsque le résultat est convenable mais éloigné du résultat espéré connu dans le cadre d'un entraînement supervisé classique, ou lorsque plusieurs réponses sont convenables dans des choix subjectifs. De manière plus conceptuelle, on pourrait aussi constater les implications des réseaux GAN dans la recommandation en terme marketing comme par exemple, la créativité a tel un impact significatif sur la recommandation, cela varie-t-il selon le type de créativité et de même l'acceptation de la nouveauté est-elle plus acceptée dans un contexte exploratoire par rapport aux autres types de créativité. Ensuite, il est possible d'étudier sur le long terme si l'utilisation de modèle augmentant la diversité de la culture de l'utilisateur, sa diversité de consommation, ou du moins évitant de tendre vers une limite de cette dernière n'affecterait pas de même d'autres facteurs, comme leur acceptation de la nouveauté ou leur tendance à explorer à leur tour, à être créatif. Il est aussi important d'analyser quel taux de créativité exploratoire par rapport à celle combinatoire de l'algorithme fait un bon combo pour ne pas trop frustrer l'utilisateur, maximiser les potentielles ventes de l'algorithme, la satisfaction et répondre de même aux besoins de diversités de l'utilisateur.

## Bibliographie :

Lee, H., & Lee, J. (2019). Scalable deep learning-based recommendation systems. *ICT Express*, 5(2), 84-88.

Reddy, M. M., Kanmani, R. S., & Surendiran, B. (2020, February). Analysis of Movie Recommendation Systems; with and without considering the low rated movies. In *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)* (pp. 1-4). IEEE.

Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1125-1134).

Lu, H. (2014). Recommendations based on purchase patterns. *International Journal of Machine Learning and Computing*, 4(6), 501.

Tsintzou, V., Pitoura, E., & Tsaparas, P. (2018). Bias disparity in recommendation systems. *arXiv preprint arXiv:1811.01461*.

Das, D., Sahoo, L., & Datta, S. (2017). A survey on recommendation system. *International Journal of Computer Applications*, 160(7).

Hannech, A., Adda, M., & McHeick, H. (2016, October). Cold-start recommendation strategy based on social graphs. In *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)* (pp. 1-7). IEEE.

Calciu, M., Salerno, F., & Moulin, J. L. (2016). Les calculs sur données-client massives sont-ils trop importants pour être laissés aux analystes marketing?. *Statistique et Société*, 4(3), 37-48.

Grossetti, Q. (2018). *Système de recommandation sur les plateformes de micro-blogging et bulles filtrantes* (Doctoral dissertation, Sorbonne université).

Balkin, A. (1990). What is creativity? What is it not?. *Music Educators Journal*, 76(9), 29-32.

Lee, G., & Lee, W. J. (2009). Psychological reactance to online recommendation services. *Information & Management*, 46(8), 448-452.

Aggarwal, A., Mittal, M., & Battineni, G. (2021). Generative adversarial network: An overview of theory and applications. *International Journal of Information Management Data Insights*, 1(1), 100004.

*Carbonell, J. G., Michalski, R. S., & Mitchell, T. M. (1983). An overview of machine learning. Machine learning, 3-23.*

*Black, M. J. (2002). The art of code. University of Pennsylvania.*

*Fleder, D., & Hosanagar, K. (2009). Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. Management science, 55(5), 697-712.*

*Hsu, C. C., Zhuang, Y. X., & Lee, C. Y. (2020). Deep fake image detection based on pairwise learning. Applied Sciences, 10(1), 370.*

*de Oliveira, L., Paganini, M., & Nachman, B. (2017). Learning particle physics by example: location-aware generative adversarial networks for physics synthesis. Computing and Software for Big Science, 1(1), 1-24.*

*Mor, N., Wolf, L., Polyak, A., & Taigman, Y. (2018). A universal music translation network. arXiv preprint arXiv:1805.07848.*

*Bian, Y., Wang, J., Jun, J. J., & Xie, X. Q. (2019). Deep convolutional generative adversarial network (dcGAN) models for screening and design of small molecules targeting cannabinoid receptors. Molecular pharmaceutics, 16(11), 4451-4460.*

*Hawkins, D. M. (2004). The problem of overfitting. Journal of chemical information and computer sciences, 44(1), 1-12.*

*Shi, W., Caballero, J., Theis, L., Huszar, F., Aitken, A., Ledig, C., & Wang, Z. (2016). Is the deconvolution layer the same as a convolutional layer?. arXiv preprint arXiv:1609.07009.*

*Dumoulin, V., & Visin, F. (2016). A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285.*

*Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.*

*Lagarrigue, A., & Viéville, T. (2021). Qu'est ce que l'IA et qu'est ce que ce n'est pas?. Lecture Jeune, 180.*

*Mahesh, B. (2020). Machine learning algorithms-a review. International Journal of Science and Research (IJSR).[Internet], 9, 381-386.*

*Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6), 386.*

*Montavon, G., Samek, W., & Müller, K. R. (2018). Methods for interpreting and understanding deep neural networks. Digital Signal Processing, 73, 1-15.*

*LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.*

*Boden, M. A. (1998). Creativity and artificial intelligence. Artificial intelligence, 103(1-2), 347-356.*

*Boden, M. A. (2004). The creative mind: Myths and mechanisms. Routledge.*

*DEARY, V. (1998). Dimensions of Creativity. Edited by M. Boden.(Pp. 242.) MIT Press: Cambridge, MA. 1994.*

*Genius: The Natural History of Creativity. By HJ Eysenck.(Pp. 344;£ 37.50.) Cambridge University Press: Cambridge. 1995. Psychological Medicine, 28(2), 485-494.*

*Palmer, D. D. (2000). Tokenisation and sentence segmentation. Handbook of natural language processing, 11-35.*

*Aizawa, A. (2003). An information-theoretic perspective of tf-idf measures. Information Processing & Management, 39(1), 45-65.*

*Gao, T., Cheng, B., Chen, J., & Chen, M. (2017). Enhancing collaborative filtering via topic model integrated uniform Euclidean distance. China Communications, 14(11), 48-58.*

*Lu, M., Qin, Z., Cao, Y., Liu, Z., & Wang, M. (2014). Scalable news recommendation using multi-dimensional similarity and Jaccard-Kmeans clustering. Journal of Systems and Software, 95, 242-251.*

*Warlop, R. (2018). Nouveaux algorithmes et méthodes d'exploration-exploitation pour des systèmes de recommandations efficaces.*

*DiFranzo, D., & Gloria-Garcia, K. (2017). Filter bubbles and fake news. XRDS: crossroads, the ACM magazine for students, 23(3), 32-35.*

*Chaney, A. J., Stewart, B. M., & Engelhardt, B. E. (2018, September). How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In Proceedings of the 12th ACM Conference on Recommender Systems (pp. 224-232).*

*Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.*

Goutefangea, P. (2017). Alan Turing et l'intelligence artificielle: le «jeu de l'imitation» et «l'IA forte».

Nguyen, T. T., Hui, P. M., Harper, F. M., Terveen, L., & Konstan, J. A. (2014, April). Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web* (pp. 677-686).

Ekstrand, M. D., Harper, F. M., Willemsen, M. C., & Konstan, J. A. (2014, October). User perception of differences in recommender algorithms. In *Proceedings of the 8th ACM Conference on Recommender systems* (pp. 161-168).

Ho, T. H., Park, Y. H., & Zhou, Y. P. (2006). Incorporating satisfaction into customer value analysis: Optimal investment in lifetime value. *Marketing Science*, 25(3), 260-277.

Ayache, A., Calciu, M., & Salerno, F. Lifetime Value calculations in continuous time buying contexts using Generating functions and Laplace transforms for customer retention models.

Jiang, Y., Shang, J., & Liu, Y. (2010). Maximizing customer satisfaction through an online recommendation system: A novel associative classification model. *Decision Support Systems*, 48(3), 470-479.

Javed, F., & Cheema, S. (2017). Customer satisfaction and customer perceived value and its impact on customer loyalty: the mediational role of customer relationship management. *The Journal of Internet Banking and Commerce*, 1-14.

Liens vers les ressources git-hub :  
<https://github.com/Axle-Bucamp/Recommandation-System-Gan>

## ANNEXES :

### Annexe 1) Calcul du gradient descent et de la backpropagation

Afin de calculer le gradient il y a donc plusieurs équations à résoudre. Pour ce faire il faut d'abord introduire quelques règles des dérivées partielles. Tout d'abord, la règle des chaînes, elle précise que lorsqu'une fonction dérivée  $\frac{df}{dy}$  dépend de plusieurs fonctions alors on peut la décliner en une suite de multiplications de ces fonctions telle que pour une fonction  $\frac{df}{dy}$  dépendant de  $\frac{dy}{dx}$  elle-même dépendante de  $\frac{dx}{du}$  :

$$d(f) = d( f(y(x(u)) ) )$$

$$\frac{d(f)}{d(y)} = \frac{d(f)/d(x)}{d(x)/d(u)} * \frac{d(u)/d(y)}$$

on remarque alors que si l'on simplifie, on retrouve  $\frac{d(f)}{d(y)}$

Ensuite il faut introduire la sum rule, la règle des sommes des fonctions dérivées.

Elle s'exprime ainsi :

Si une fonction  $f(x) = a h(x) + b z(x)$  alors  $f'(x) = ah'(x) + b z'(x)$

Ensuite il nous faut calculer deux fonctions, celle servant à mettre à jour la valeur d'un poids, le paramètre  $w$ , coefficient multiplicateur de notre régression linéaire et celle du paramètre  $b$ , biais de notre modèle.

Nous cherchons donc la valeur des dérivés  $\frac{df}{dw}$  et  $\frac{df}{db}$  ainsi que celle de  $\frac{df}{dy}$  si l'on souhaite calculer notre gradient.

Notre fonction de coût étant dans le modèle précédent une « mean squared error », le résultat de notre dernière couche s'exprime alors ainsi, si on cible la fonction d'activation de la dernière couche du réseau :

$$MSE = \frac{1}{2} * n \text{ somme}(y_{\text{vrai}} - y_{\text{prédict}})^2$$

$$\text{MSE} = 1/2 * \mathbf{n} \text{ somme( } y_{\text{vrai}} - \tanh(\text{somme( } w^*x + b )) )^2$$

et donc  $\mathbf{df/dw} = 1/2 * n * \text{somme( } d/dw ( u - v )^2 )$  selon la « sum rule »;

et ainsi selon la « chain rule » nous obtenons  $\mathbf{df/dw} = 1/2n 2(u - v) * d/dw (u - v)$ .

*Car  $df/dw$  dépend d'une fonction  $z = b^2$  dépendant elle même d'une fonction  $b = (u-v)$ .*

*$df/dw$  est égal à  $z' * b' = 2*b * (u-v)' = 2*(u-v) * (u-v)'$ .*

Donc on obtient :  $\mathbf{df/dw} = 1/2n 2(u - v) * (d/dw(u) - d/dw(v))$ . soit  $-1/n (u - v) * (d/dw u)$  car  $y$ , la vérité à prédire, ne dépend pas de  $w$  ou de  $b$ . il ne nous reste alors qu'à calculer la dérivative de notre fonction d'activation  $\tanh(w^*x+b)$  selon  $w$  et  $b$ .

On a donc une fonction  $\tanh(t)$  avec  $t = w^*x+b$  et donc on peut encore une fois utiliser la règle des chaînes.

$$d/dw u = 1 - \tanh(w^*x+b)^2 * x \text{ et } d/db = 1 - \tanh(w^*x+b)^2$$

En combinant le tout on trouve notre dérivé selon  $w$  ou selon  $b$ .

$$\mathbf{df/dw} = -1/2n 2(u - v) * 1 - \tanh(w^*x+b)^2 * x$$

$$\mathbf{df/db} = -1/2n 2(u - v) * 1 - \tanh(w^*x+b)^2$$

à cela on peut ajouter

$$\mathbf{df/dy} = df/dy ( 1/2n(y - \tanh(w^*x+b))^2 ) = 1/2n 2*y * d/dy (y - \tanh(w^*x+b))$$

$$1/2n 2*y * d/dy (y - \tanh(w^*x+b)) = 1/2n 2*y * (d/dy(y) - d/dy (\tanh(w^*x+b)))$$

$$1/2n 2*y * (d/dy(y) - d/dy (\tanh(w^*x+b))) = 1/2n 2*y * 1 = 1/2n 2*y$$

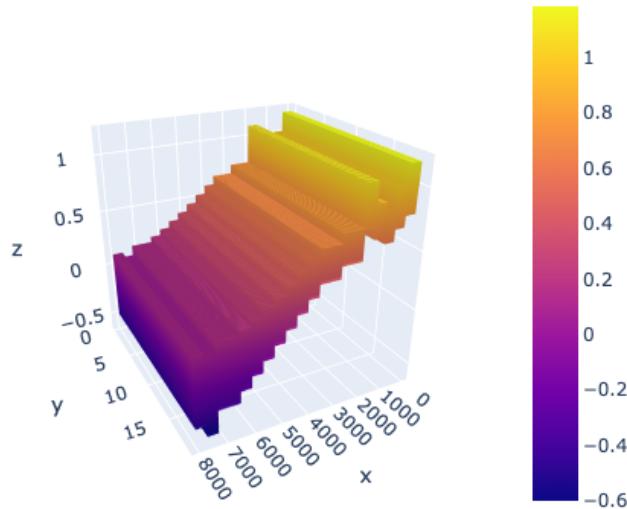
Pour le calcul du gradient qui ce calculera ainsi :

**Gradient(  $y, w, b$  )=  $df/dw + df/db + df/dy$**  en ajoutant la dérivée partielle de chaque paramètre bien évidemment ce qui devient vite très long et très gourmand en ressources.

Notre gradient si l'on utilise alors qu'un neurone pourra ressembler à ceci :

pour  $x = [1, 2, 3, \dots, 20]$

et  $y = [-1, -0.9, \dots, 1]$  (avec des variation)



Cependant notre modèle ne possède pas qu'une seule couche de neurone, ici chaque neurone transmet de l'information et donc chaque neurone applique la fonction vue auparavant. Le dérivé selon  $w$  est alors une encapsulation de chacune des fonctions des neurones, or étant eux-mêmes connectés entre eux cela fait rapidement un nombre exponentiel de fonctions à calculer. Dans le cadre de petit modèle bien que très exhaustif, ce calcul reste possible cependant dans le cadre de modèle plus profond, il faudra utiliser une autre méthode ayant la même philosophie, utilisant le gradient, cela s'appelle la backpropagation soit le rétro propagation du gradient. Lors de cette dernière, le gradient est calculé à partir de l'erreur  $J$  par rapport au poids  $J-1$ . Ainsi, à chaque étape seul le gradient de la couche  $J$  par rapport à la couche  $J-1$  est utilisé. Cela réduit considérablement les calculs nécessaires.

## On à donc ces équations final :

Pour les neurones donnés utilisant ce modèle :

$$x_j^{(n)} = g^{(n)}(h_j^{(n)}) = g^{(n)}\left(\sum_k w_{jk}^{(n)} x_k^{(n-1)}\right)$$

Pour  $x(n)j$  un neurone et  $g(n)$  une fonction d'activation dont  $h(n)j$  est la fonction d'agrégation des entrées associer à cette couche. Soit simplement le calcul vu auparavant lors de la propagation avant (feed forward) ou simplement la prédiction.

$$e_i^{\text{sortie}} = g'(h_i^{\text{sortie}})(y_i - t_i)$$

On obtient le calcul du gradient de l'erreur lié à la dernière couche suivant :

$$e_j^{(n-1)} = g'^{(n-1)}(h_j^{(n-1)}) \sum_i w_{ij}^{(n)} e_i^{(n)}$$

et dès lors celui de toutes les couches ( $n-1$ ) ce calculera ainsi :

donnant alors pour valeur de l'erreur ( $l-1$ ) la valeur de la dérivée de la fonction d'activation de la couche ( $n-1$ ) multiplié par la fonction d'agrégation (fonction linéaire dans notre cas) de la couche ( $n-1$ ) multiplié par la somme des poids fois l'erreur de la couche ( $n$ ). Ainsi, l'erreur calculée à la couche «  $n$  » est bien propagée jusqu'aux premières couches du réseau.

$$w_{ij}^{(l)} = w_{ij}^{(l-1)} - \lambda e_i^{(l)} x_j^{(l-1)}$$

Ce qui nous permet d'actualiser la valeur de nos paramètres via la formule suivante :

soit la nouvelle valeur des paramètres de la couche (l) sont égaux aux poids (l-1) moins le taux d'apprentissage fois l'erreur multiplié par la prédiction de la couche (l-1).

Bien que cet algorithme ne soit pas parfait, il permet tout de même lorsque le réseau est approprié à la tâche demandée de tendre vers une solution minimisant l'erreur associée aux prédictions selon notre calcul de cette dernière.