Comprendre les GPT

GPT signifie Generative Pretrained Transformer.

> En termes simples
>
> *Generative* signifie que le modèle peut générer du contenu, et pas seulement le percevoir. *Pretrained* indique qu'il a déjà été entraîné à apprendre (nous verrons plus loin comment, avec la "rétropropagation"). Quant à *Transformers*, il s'agit du cœur des GPT : une architecture spécifique de réseaux de neurones.

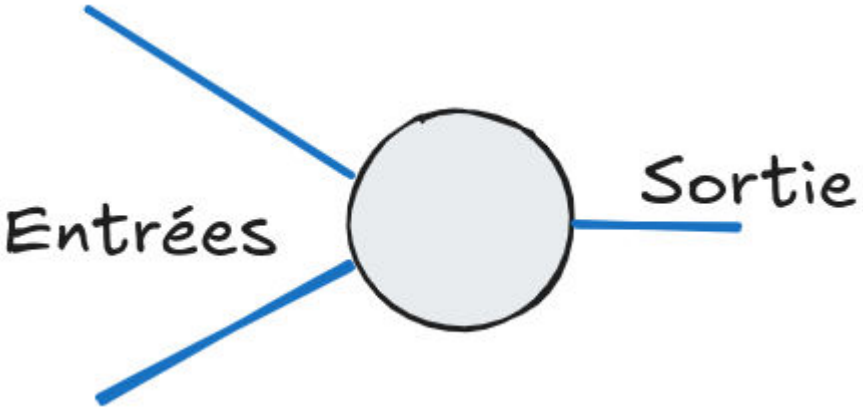
Mais avant d'aller plus loin, découvrons ce qu'est un neurone artificiel et comment ils sont assemblés.

Le terme **GPT** signifie *Generative Pretrained Transformer* (Transformateur Génératif Préentraîné). Chaque mot de cet acronyme résume un aspect clé de la structure et du fonctionnement de ces modèles d'intelligence artificielle.

Qu'est-ce qu'un GPT ?
Il s'agit d'un réseau de neurones artificiel (une simulation simplifiée de nos propres neurones) que l'on peut utiliser pour prévoir un texte, comme un chatbot. Cela rappelle l'autocomplétion dans la barre de recherche Google : si vous tapez "Quel est le résultat de l'élec...", il pourra compléter par "tion". Mais il peut faire bien d'autres choses.

Le perceptron

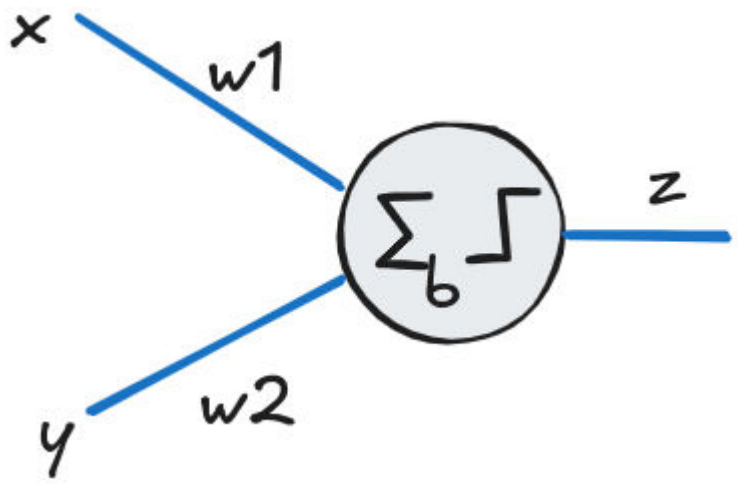
Voici la brique fondamentale des réseaux de neurones, dont les GPT font partie. Il s'agit ici d'un perceptron minimal avec deux entrées (les réseaux peuvent en contenir bien davantage, mais le perceptron a toujours une seule sortie).



En termes simples

Le perceptron effectue une somme pondérée de ses entrées. Si cette somme dépasse un certain seuil, il s'active ; sinon, il reste inactif. En ajustant les poids des entrées, il peut "apprendre". La question qui se pose est de comprendre comment un élément aussi simple, en se multipliant, peut aboutir à ChatGPT !

Description du fonctionnement du perceptron



1. **Entrées (`x` et `y`)** : Le perceptron prend plusieurs entrées, ici symbolisées par `x` et `y`.
2. **Poids (`w1` et `w2`)** : Chaque entrée est multipliée par un poids respectif (`w1` pour l'entrée `x` et `w2` pour l'entrée `y`), qui détermine l'importance de chaque entrée.
3. **Biais (b)** : un biais interne est ajouté.
4. **Somme pondérée (Σ)** : Le perceptron calcule la somme pondérée des entrées :

$$s = x \cdot w1 + y \cdot w2 + b$$

5. **Fonction d'activation (S)** : La somme pondérée `s` passe par une fonction d'activation (symbolisée par le "S" à côté de la somme) qui transforme `s` en une sortie finale `z`. La fonction d'activation permet d'introduire de la non-linéarité et de décider si le perceptron s'active ou non (par exemple, une fonction de seuil, sigmoïde, ou ReLU).
6. **Sortie (`z`)** : La valeur finale `z` est le résultat de la fonction d'activation appliquée à la somme pondérée `s`. C'est un chiffre entre 0 et 1. Dans sa forme simple, c'est 0 ou 1.

Donc, la sortie `z` est calculée par :

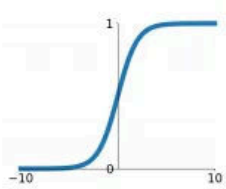
$$z = f(s) = f(x \cdot w1 + y \cdot w2 + b)$$

où `f` est la fonction d'activation.

Voici celles qui sont fréquement utilisées :

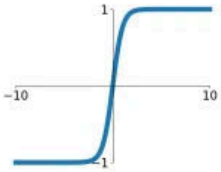
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



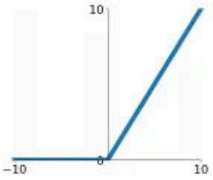
tanh

$$\tanh(x)$$



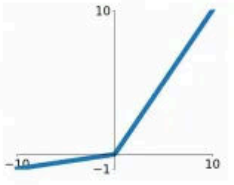
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

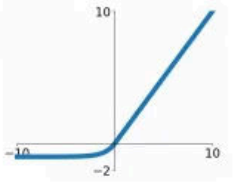


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



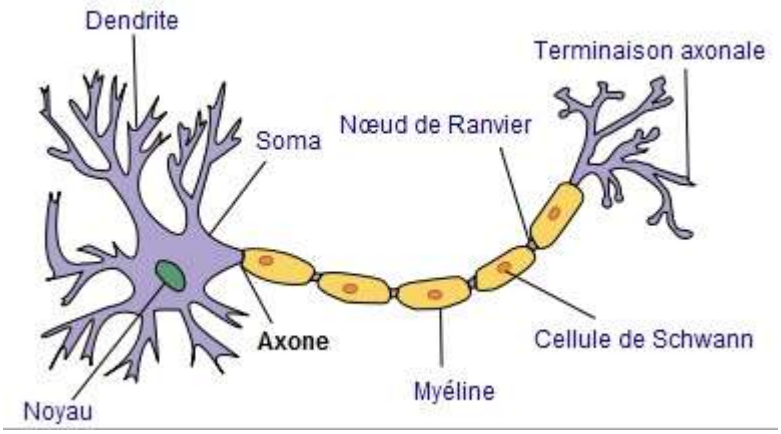
La plus simple de toutes étant : "si le résultat est positif renvoyer 1, sinon 0".

Source historique du Perceptron : Frank Rosenblatt, « The Perceptron, A perceiving and recognizing automaton », *Cornell Aeronautical Laboratory*, janvier 1957

Analogie avec le neurone

Le perceptron est l'une des premières structures de base dans le domaine des réseaux de neurones artificiels. Conçu pour imiter le fonctionnement d'un neurone biologique, il peut effectuer des calculs simples en classifiant des données linéairement séparables. Introduit à la fin des années 1950, le perceptron a marqué un tournant dans la recherche en intelligence artificielle et en apprentissage automatique.

Il représente une modélisation très simplifiée du neurone biologique, qui reçoit des signaux et émet éventuellement une impulsion électrique le long de son axone.

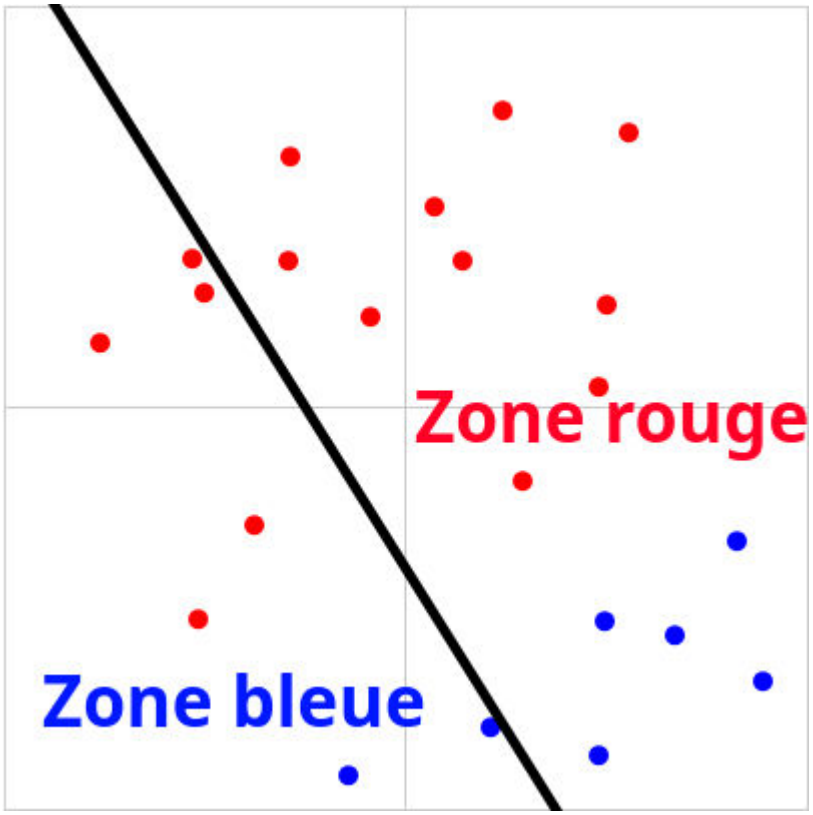


Pour la petite histoire,

Le perceptron à quoi ça sert ?

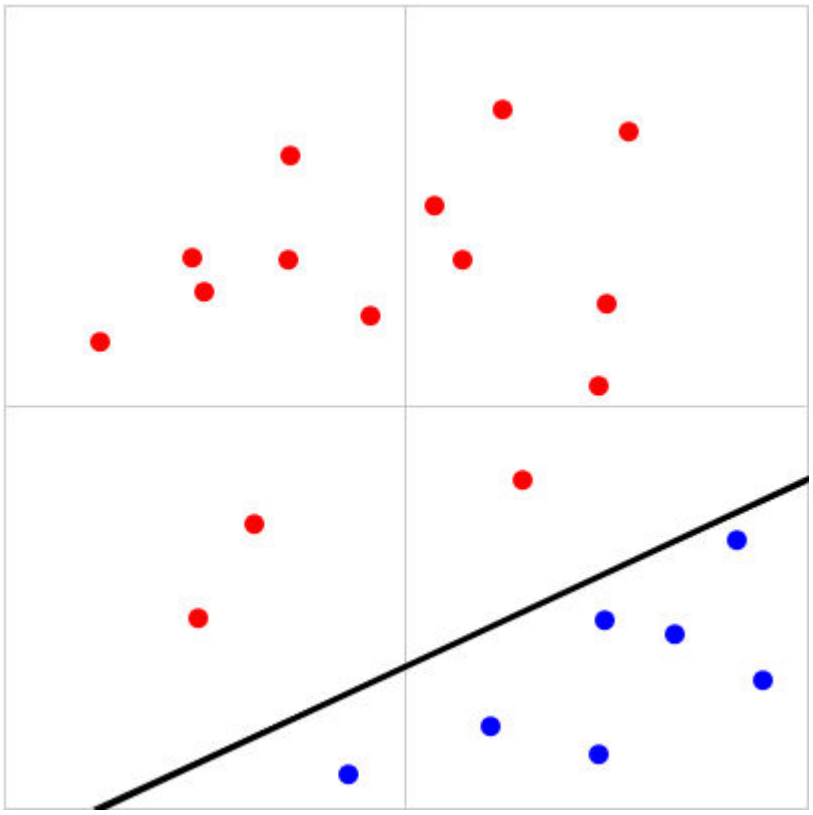
En termes simples

Le perceptron a une tâche simple : distinguer des points de deux couleurs différentes en traçant une frontière pour les séparer. Bien que cela puisse paraître basique, cette capacité est en réalité très puissante, tout comme les transistors dans un microprocesseur. Le perceptron est l'équivalent d'une brique Lego dans la construction de réseaux de neurones complexes.

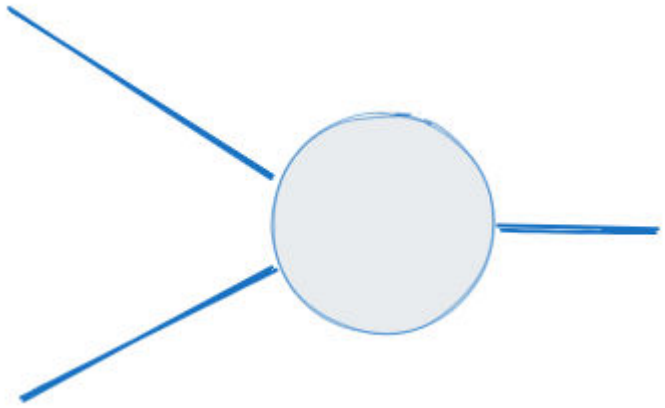


Le rôle du perceptron est de diviser le plan en deux zones : une zone rouge et une zone bleue. Comme on le voit ici, il peut se "tromper" et placer certains points du mauvais côté de la frontière. En effet, le perceptron ne peut pas modifier ses perceptions et se contente de classifier ce qu'il observe.

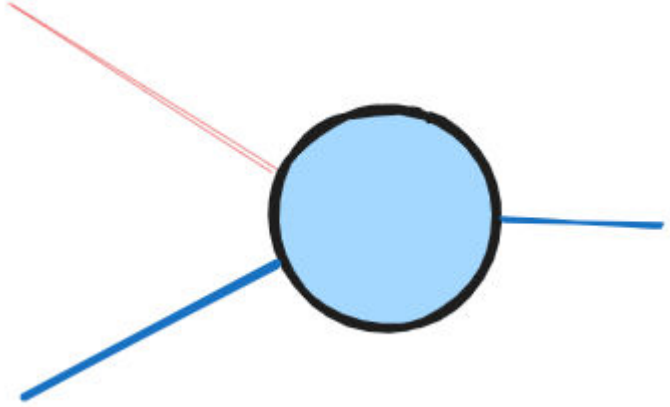
Les erreurs de classification, c'est-à-dire les points mal placés, servent ensuite à ajuster progressivement les poids internes du perceptron, lui permettant ainsi d'"apprendre". Cette méthode d'ajustement est à la base de l'algorithme clé qui permet aux GPT d'apprendre : la rétropropagation (que nous verrons plus tard).



Une fois bien entraîné, le perceptron parvient à classer correctement les points. Pour cela, il a fallu ajuster ses poids (*w1*, *w2*) ainsi que son biais. Par exemple, il est passé de cette configuration initiale :



à cette configuration optimisée :



Ici, les poids *w1* et *w2* ainsi que le biais sont représentés par l'épaisseur et la couleur du trait, avec le rouge pour les valeurs positives et le bleu pour les valeurs négatives (selon une convention).

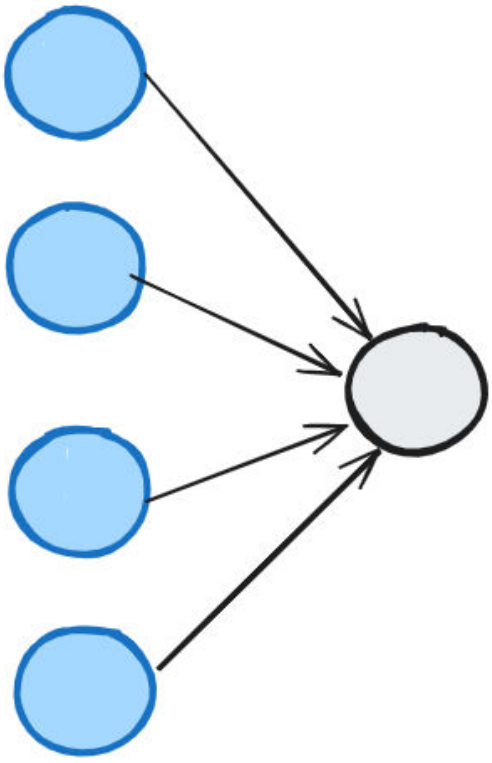
Que peut-on faire avec plusieurs perceptrons ?

En termes simples

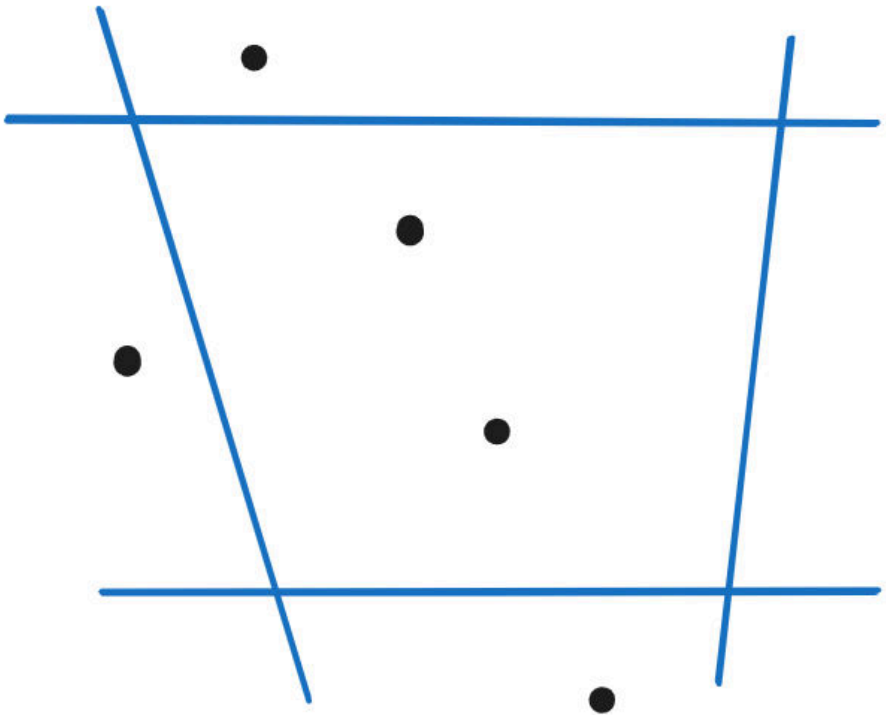
Si le perceptron le plus simple ne peut qu'une ligne de démarcation, en combinant plusieurs perceptron nous pouvons leur faire reconnaître n'importe quelle forme. Plus la forme sera complexe plus il faudra de perceptrons ou de "dentrites" (d'entrées/poids).

Un perceptron, une fois entraîné, peut simplement déterminer si un point se trouve d'un côté ou de l'autre d'une ligne.

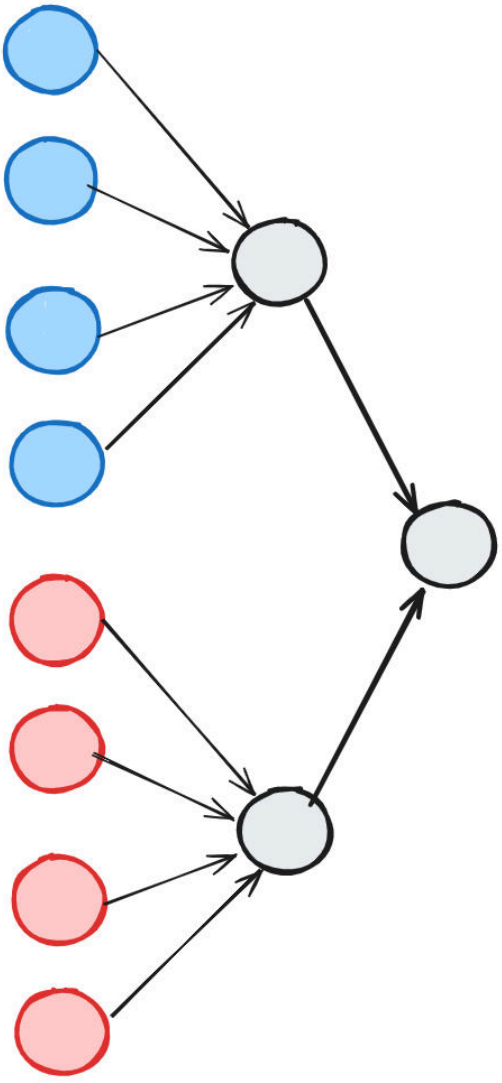
Là où le perceptron devient puissant, c'est lorsqu'on les enchaîne : les sorties de certains perceptrons servent d'entrées pour d'autres.



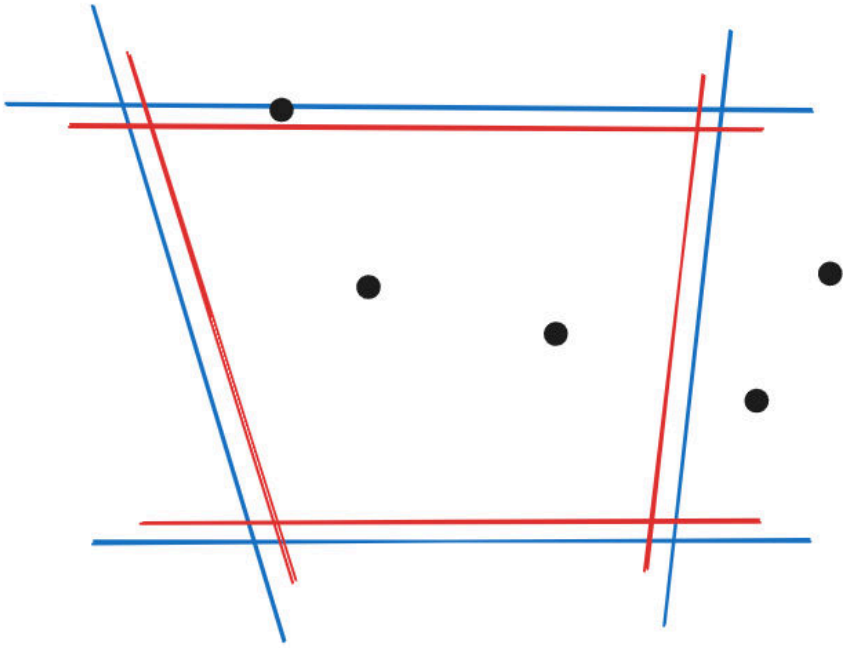
Ainsi, en utilisant quatre perceptrons, on peut facilement savoir si un point se situe à l'intérieur d'un périmètre.



En répétant cette combinaison, on peut également détecter si un point se trouve précisément le long du contour d'un périmètre, avec une certaine précision.



Le perceptron final tout à droite peut nous dire si un point se situe non pas dans un périmètre, mais le long de celui-ci (avec une certaine précision) :



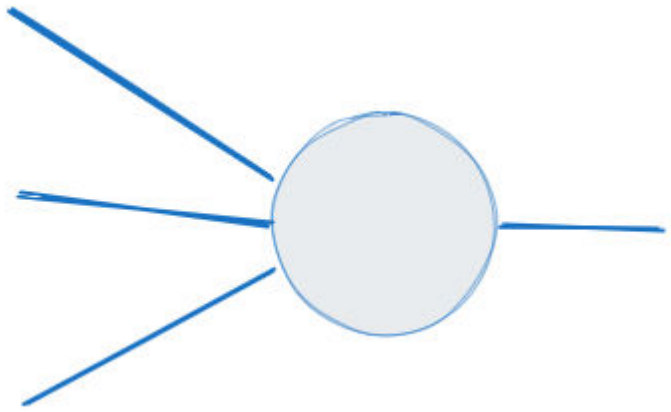
De cette manière, nous apprenons au réseau à reconnaître des formes avec des contours définis !

C'est le principe fondamental : plus on ajoute de couches de neurones successives, plus on peut modéliser des concepts abstraits.

La véritable question devient alors celle de l'organisation des neurones, car en théorie, on peut les structurer en réseau de nombreuses façons pour modéliser des formes complexes.

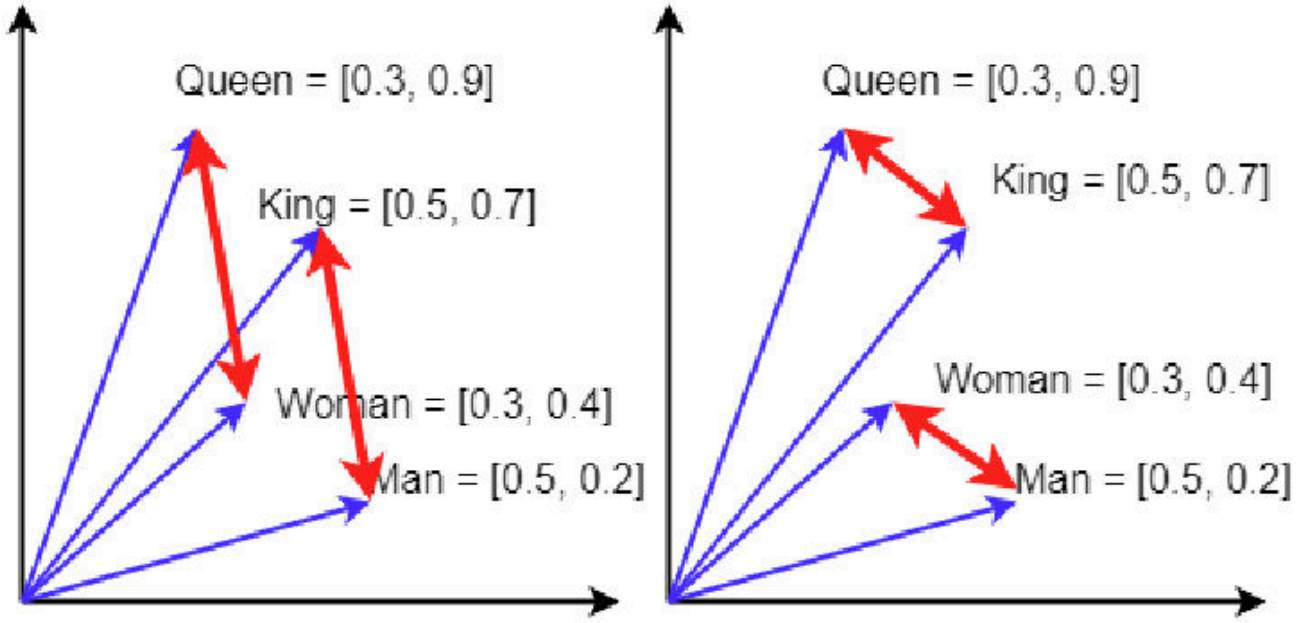
Les entrées du perceptron (embeddings)

Le neurone ne traite que des chiffres et n'a aucune "connaissance" de leur signification. Ces valeurs numériques peuvent représenter la position d'un point, une vitesse, une couleur, un poids, ou tout autre paramètre traduisible en chiffres.

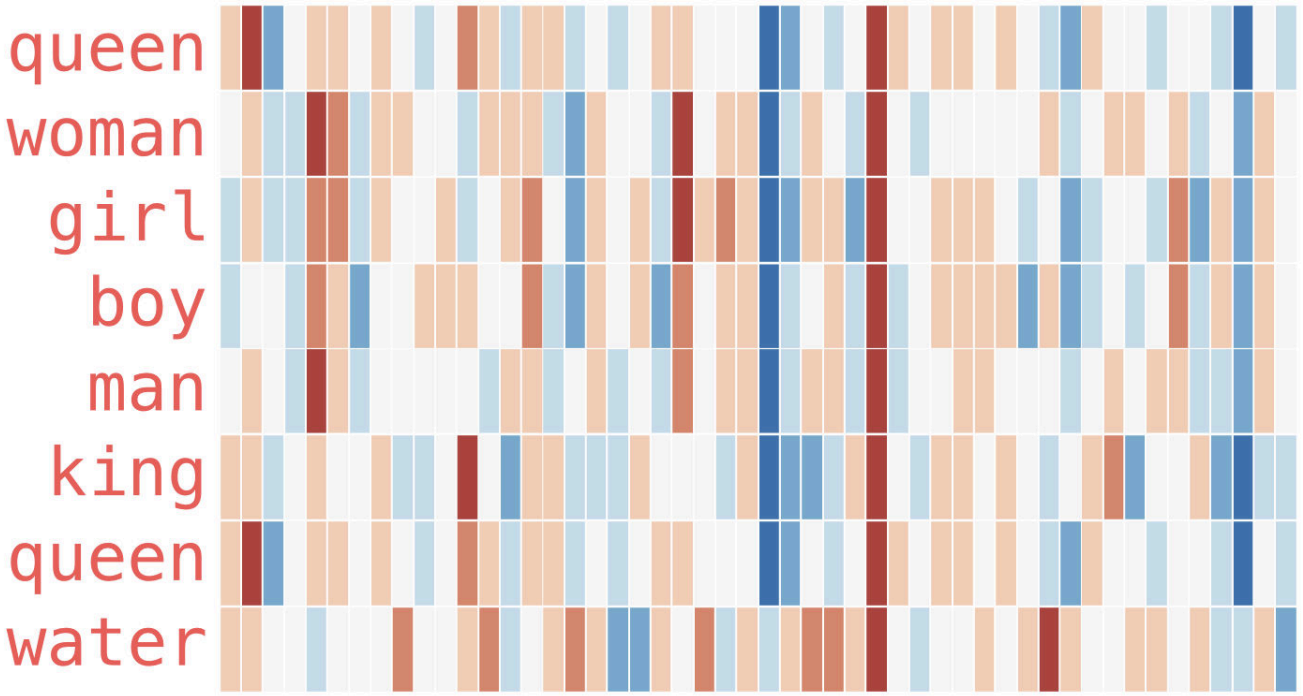


Cela soulève une question : comment organiser les perceptrons ? Un perceptron à deux entrées permet de classer des points en fonction d'une ligne dans un plan. Avec trois entrées, il peut classer des points dans un espace 3D par rapport à un plan. De manière générale, un perceptron à *n* entrées sépare les points dans un espace de *n* dimensions le long d'un "plan" (on parle d'hyperplan), divisant toujours en deux groupes.

Pour représenter des mots, on utilise des vecteurs, c'est-à-dire des ensembles de valeurs numériques, qui placent chaque mot en un point dans un espace multidimensionnel, permettant d'établir des relations entre eux. Par exemple, l'opération : "Roi - Femme + Homme" donne un vecteur proche de celui de "Reine".



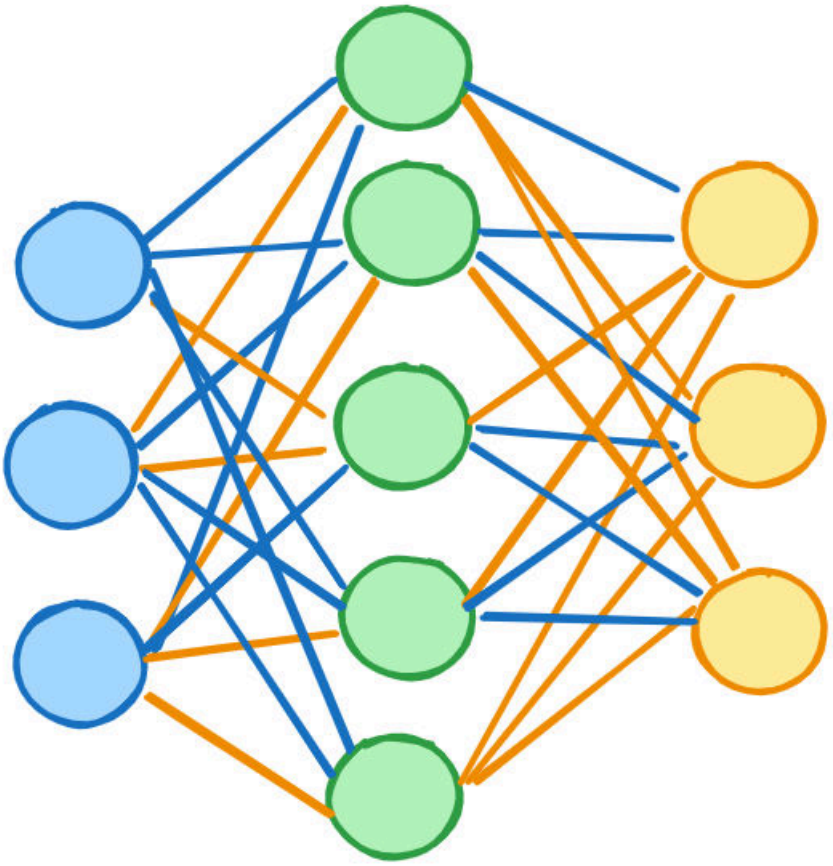
Ces représentations vectorielles, appelées "embeddings", traduisent les mots en vecteurs. Le nombre de dimensions est généralement déterminé de manière empirique. De même qu'une couleur peut être codée en chiffre (rouge ou bleu), un mot peut être représenté par une série de valeurs ou de couleurs :



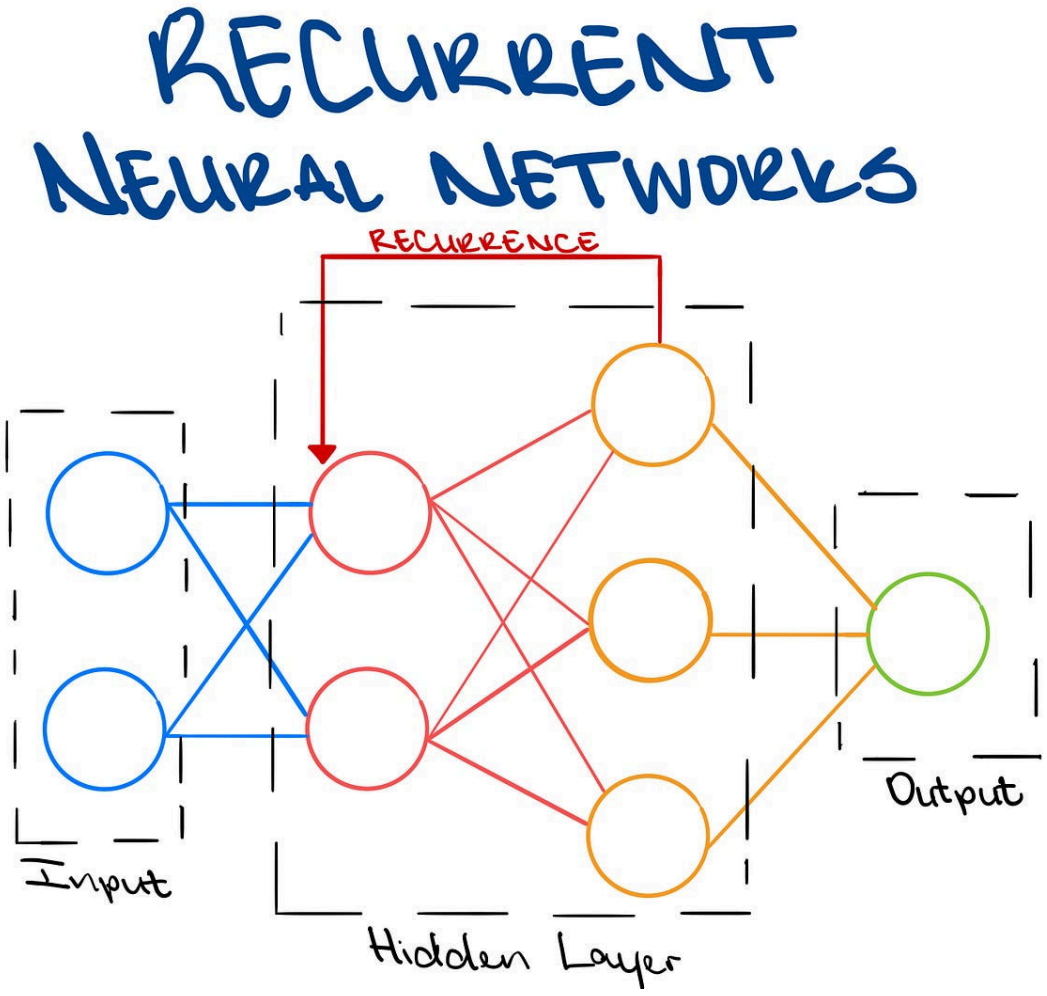
Le MLP : multi-layer perceptron

Dans la pratique, les perceptrons sont souvent organisés en "couches", où les neurones d'une même couche partagent des caractéristiques similaires.

C'est le principe du MLP, ou perceptron multicouche (multilayer perceptron). Dans cette architecture spécifique, chaque neurone est connecté à tous les neurones de la couche précédente. Bien que d'autres configurations soient possibles, cette approche a démontré son efficacité.



Parenthèse technique : les MLP ne sont pas turing complet (ils ne peuvent pas faire ce que fait un ordinateur) même si le perceptron peut simuler des portes logiques, mais ils sont des approximateurs de fonction universel. Les RNN (réseau de neuronnes récurrents) en revanche le sont :



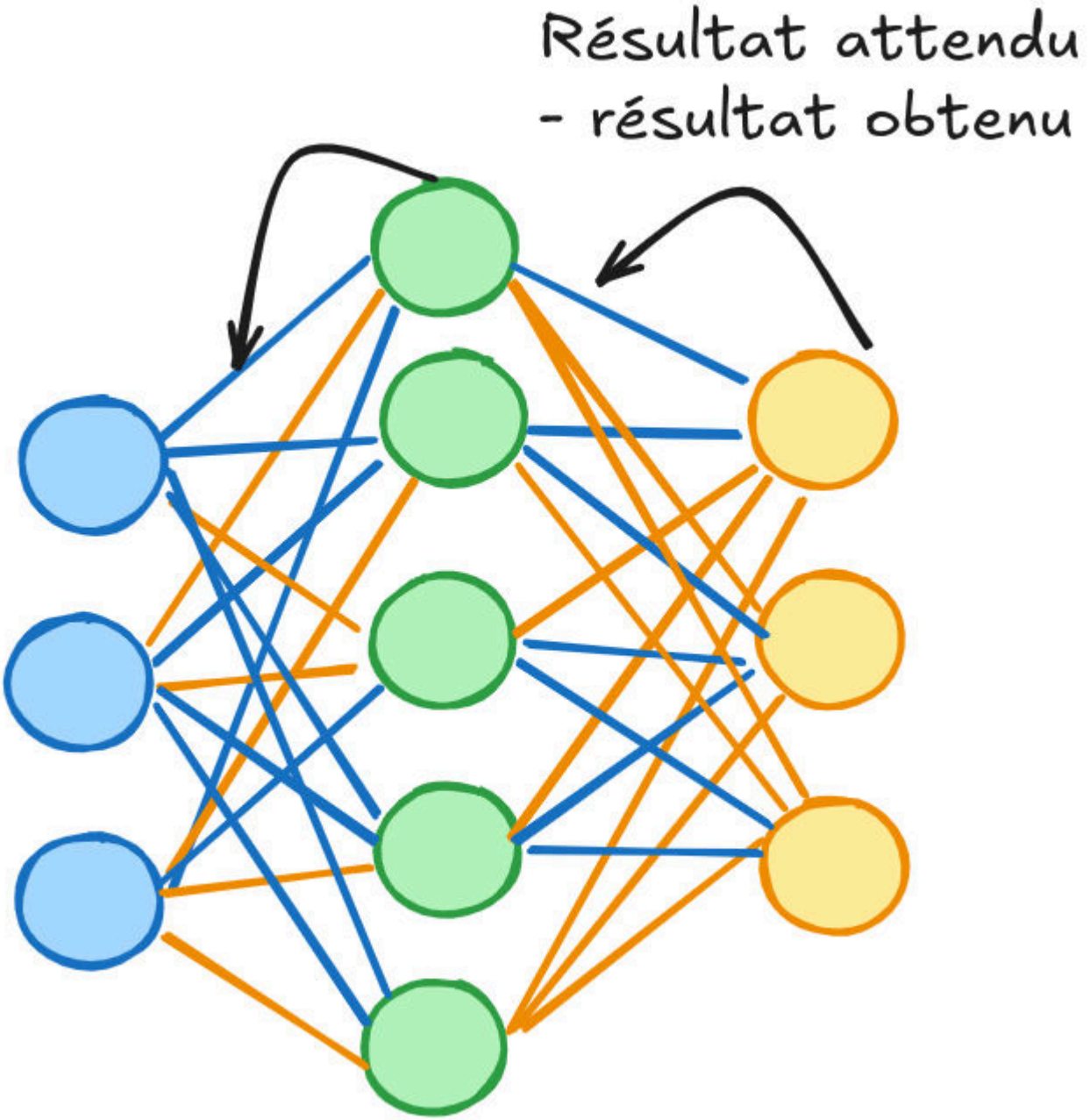
La différence, c'est qu'on leur donne le droit de revenir en arrière.

La rétro-propagation

En termes simples

La rétropropagation consiste à corriger la sortie du réseau de neurone progressivement pour qu'elle se rapproche du résultat voulu.

Nous avons vu qu'un perceptron, ou neurone, apprend en ajustant progressivement le poids de ses connexions et son biais.

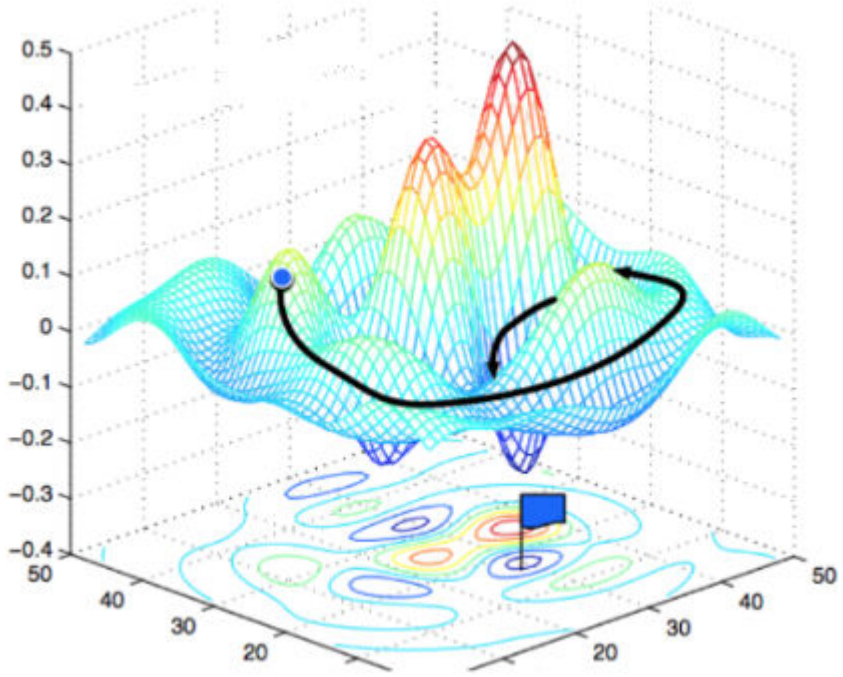


En gros, l'idée est la suivante : les neurones jaunes, qui produisent la sortie finale, sont comparés à la sortie désirée. On en déduit alors quelles auraient dû être les sorties des neurones verts pour obtenir le bon résultat, en tenant

compte de la contribution de chaque neurone à l'erreur. Une fois que l'on dispose des résultats pour les neurones verts, on fait de même pour les bleus. On rétro-propage ainsi l'erreur de la fin vers le début. Ensuite, une fois que l'on connaît l'erreur pour chaque neurone, on ajuste légèrement les poids (note technique : en fonction de la dérivée partielle de l'erreur) pour se rapprocher du résultat voulu.

Source historique pour la rétropropagation : Learning representations by back-propagating errors by David Rumelhart and Geoffrey Hinton, 1986

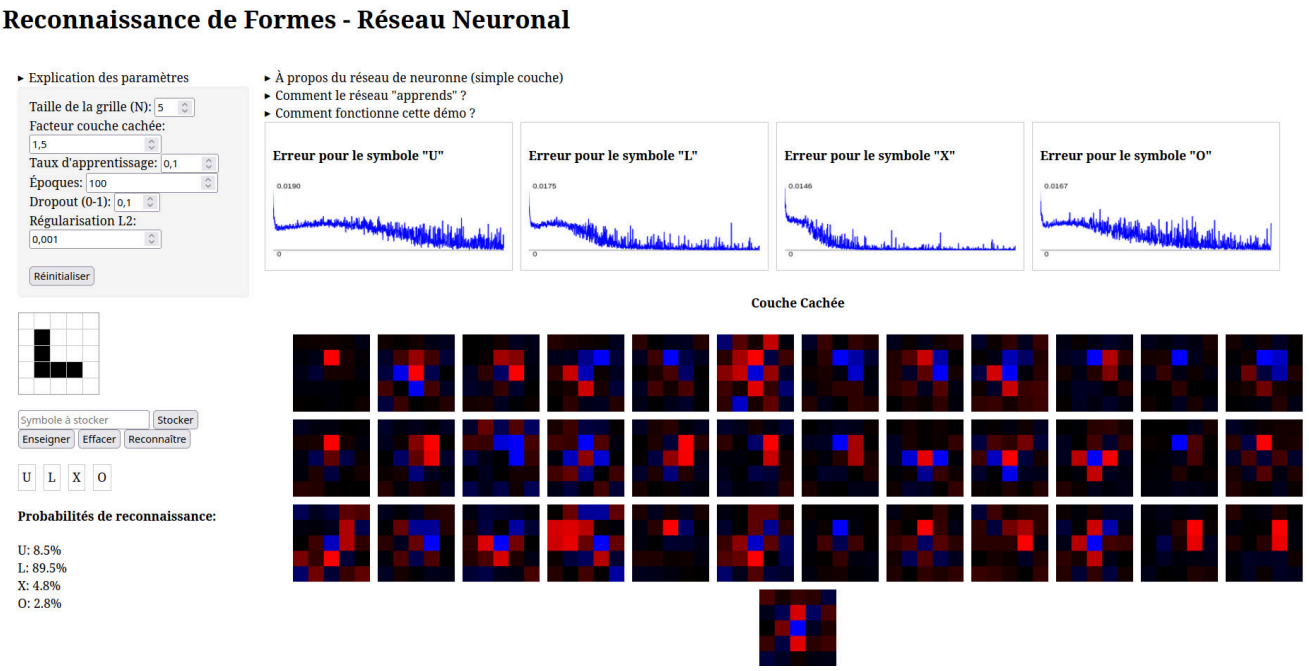
Parenthèse technique : la rétropropagation corresponds à une descente de gradient dans un espace multi-dimensionnel.



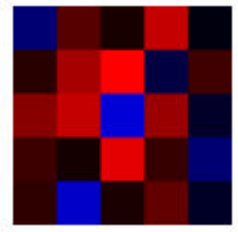
Vous l'aurez compris, il va falloir de nombreux "rounds" d'apprentissage pour que la rétropropagation fonctionne. Ces rounds s'appellent des "époques". Sur les grands modèles, cela peut prendre des jours ou des mois pour les entraîner.

Application : Reconnaître des formes

Voici ci-dessous la représentation visuelle d'un réseau MLP avec une couche cachée (les carrés rouges et bleus disposés en damier). Dans cet exemple, le MLP a appris à reconnaître les lettres U, L, X et O. Nous pouvons observer que l'erreur diminue progressivement au fil des "époques" d'apprentissage. Finalement, il parvient à reconnaître toutes les lettres qu'il a apprises. À l'écran, le score de reconnaissance de la lettre L est affiché à 89,5 %.



Avec un MLP, il est possible, grâce à la rétropropagation, d'apprendre à reconnaître diverses formes, même complexes. La représentation d'un perceptron est souvent illustrée de cette manière :



Chaque carré représente le poids d'une entrée, ou, pour le dire autrement, une dendrite. À mesure que l'erreur est rétropropagée, elle diminue progressivement, et lorsque celle-ci devient suffisamment faible, le réseau de neurones artificiels (ANN) ou MLP (perceptron multicouche), ou encore FFL (feed-forward layer), est capable de reconnaître n'importe quelle forme.

Dans l'exemple ci-dessus, la grille contient seulement 5 pixels, mais la taille peut être ajustée à souhait :

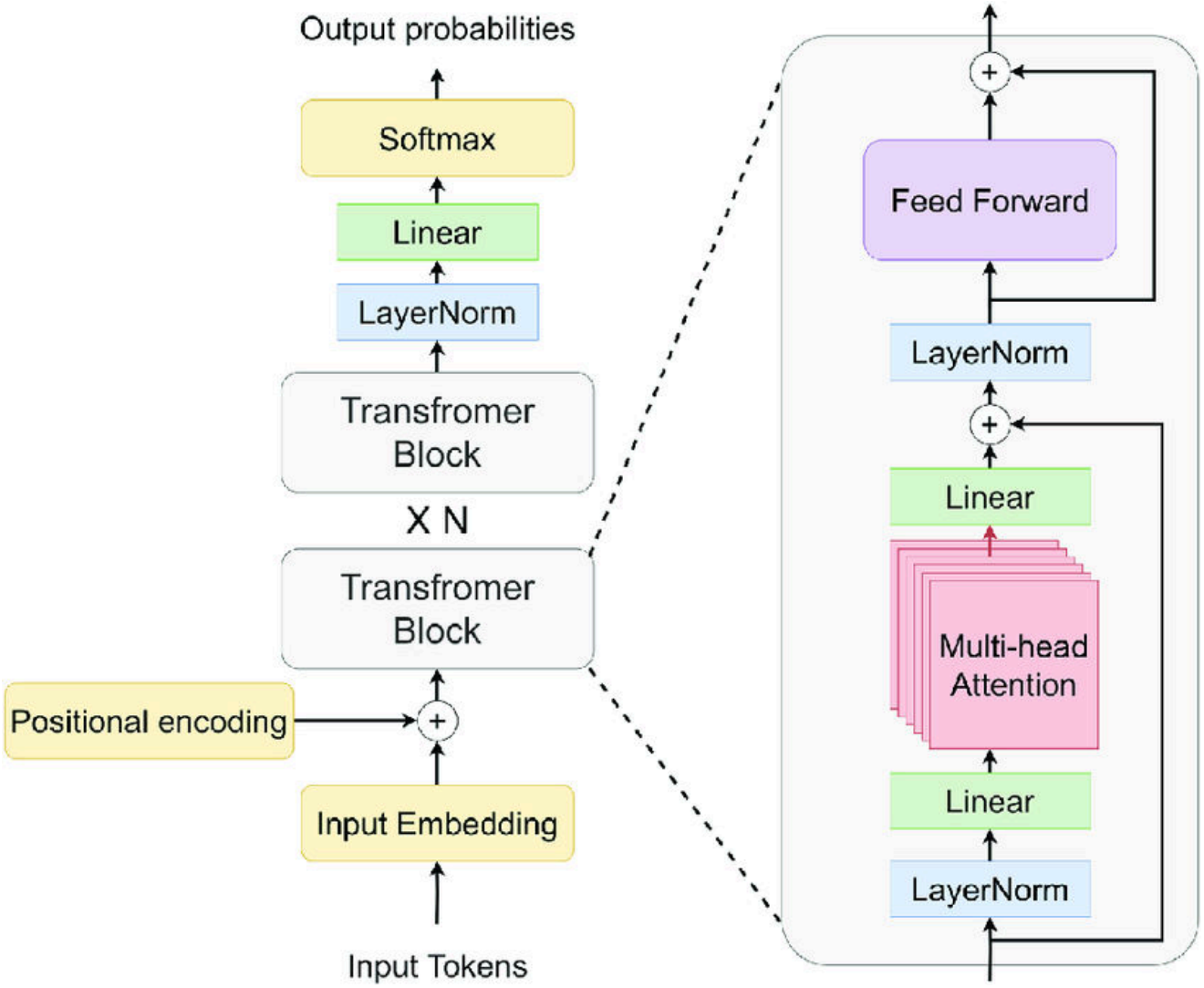
Ici, le réseau estime que le chiffre représenté est probablement un "2", mais il pourrait également s'agir d'un début de "3" ou même d'un "8" si l'on ajoutait une diagonale reliant les deux extrémités.

Mais peut-on simplement, avec un MLP faire apprendre n'importe quoi à un réseau de neurone pourvu qu'il y ait suffisamment de couches ? Ne faut-il pas autre chose que des MLP ? Après tout, ce n'est qu'une architecture possible parmi beaucoup d'autres.

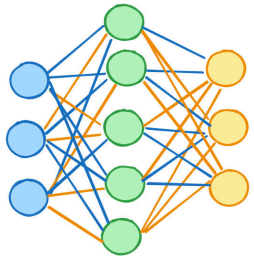
La réponse est empirique. À force de chercher à tâton,, quelqu'un à découvert l'autre brique qui était indispensable : le bloc d'attention.

Le transformers

L'architecture des GPT est relativement complexe, mais on peut l'appréhender de manière progressive. Au cœur de ces modèles, on trouve une série de blocs appelés "Transformers", complétés par des couches de "stabilisation".



Les deux éléments principaux du Transformer (illustrés à droite) sont le "Feed Forward" et l'"Attention multi-têtes". Tout le reste est un ajout de moindre importance (qui sert à stabiliser le réseau et à faire la "colle" entre les couches). Le "Feed Forward" est un MLP, exactement comme celui que nous avons vu précédemment :

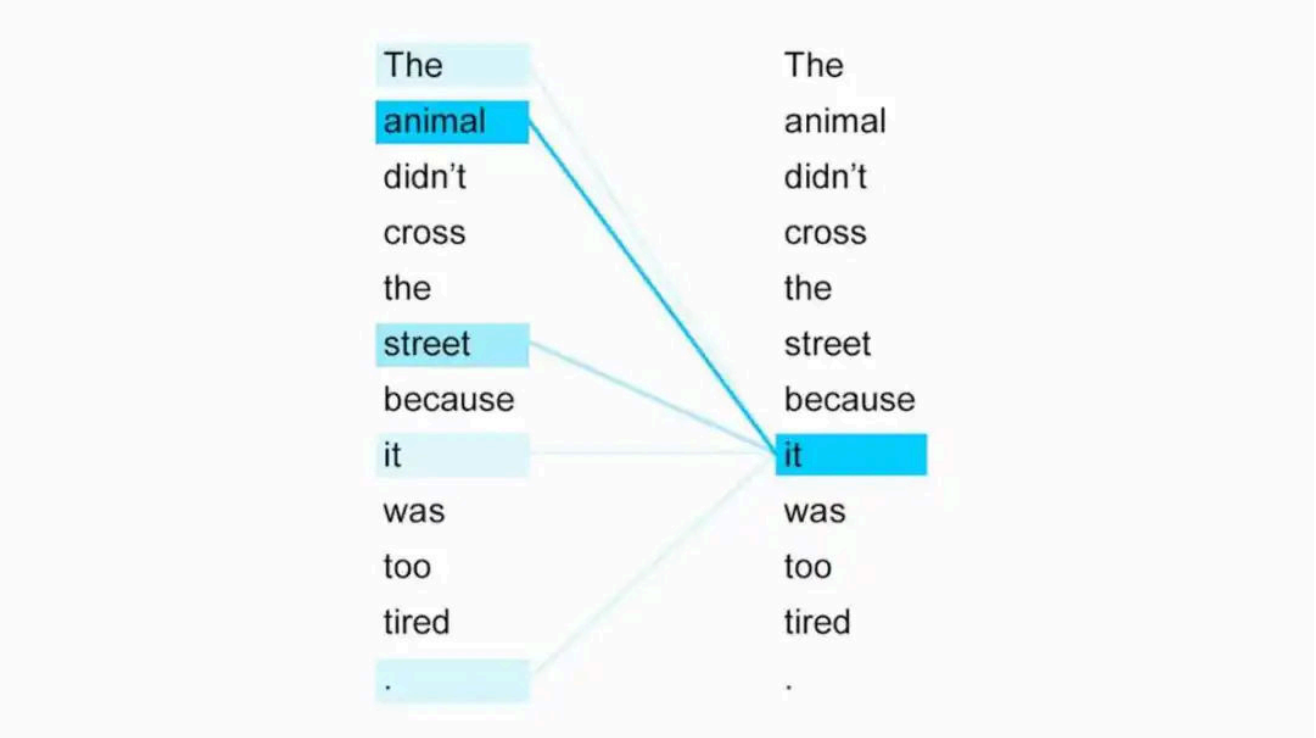


La différence réside dans le fait que le nombre de neurones par couche correspond à la taille de la fenêtre de contexte.

Les recherches sur les LLM (modèles de langage de grande taille) montrent que c'est dans ces couches que s'effectue la mémorisation des "faits". Source: How might LLMs store facts | Chapter 7, Deep Learning - YouTube

La fenêtre de contexte désigne la quantité de texte que le GPT peut traiter à la fois. Par exemple, une taille de contexte de 2k implique que le modèle peut traiter environ 2000 mots (ou tokens, mais je simplifie ici). Au-delà de ce seuil, le modèle oublie le début de la conversation, un peu comme notre propre mémoire de travail.

Ensuite, il y a la couche d'attention. Son architecture est assez complexe. En revanche, il est plus simple de comprendre à quoi elle sert.



Son rôle est de donner du sens, d'établir des liens entre les éléments. Par exemple, dans cette phrase, le mot "it" peut se référer à "animal" ou à "street", avec une préférence pour "animal". Ce mécanisme d'attention, qui semble s'apparenter à notre propre processus de compréhension, permet au Transformer d'attribuer du "sens" aux mots.

Le sens, tout comme la mémoire, est appris par rétropropagation.

Les GPT sont donc construits à partir de chaînes de Transformers (un bloc d'attention/sens + un bloc MLP/mémoire), ce qui explique leur nom. Le fait

d'en enchaîner plusieurs permet d'avoir d'autant plus de niveaux d'abstractions.

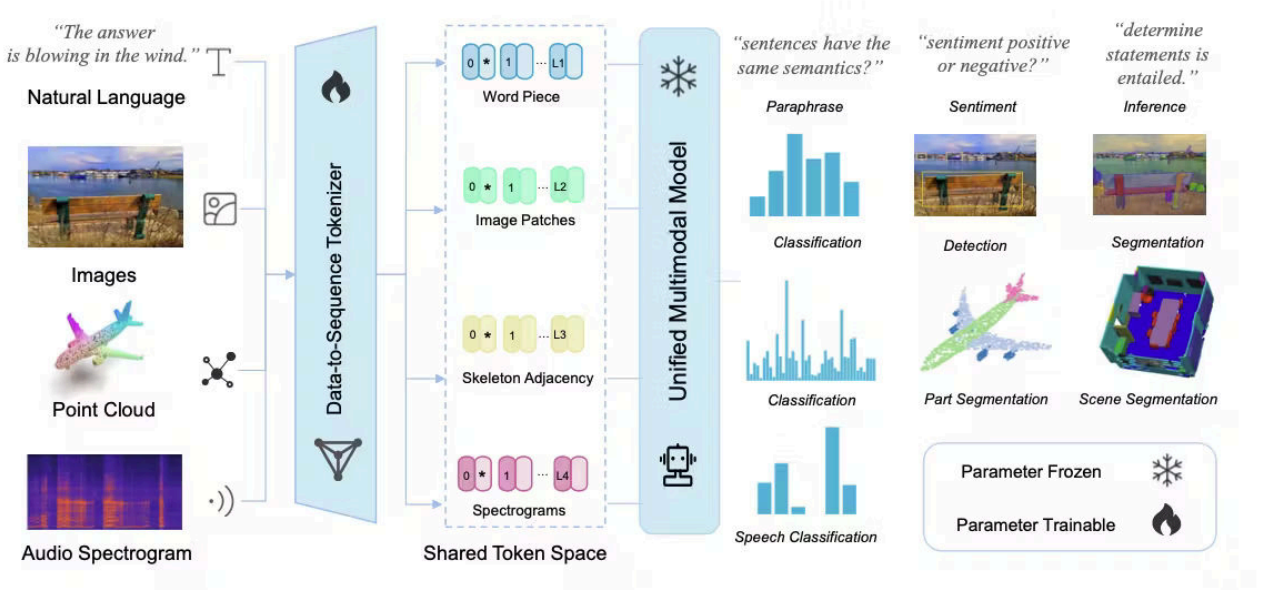
Source historique pour l'Attention : Attention is all you need, 2017

Conséquences pratiques pour les GPT

Les GPT traitent "aveuglément" des séries de chiffres. Ils n'ont pas de "conscience" au sens phénoménologique de ce qu'ils font. Leur fonctionnement repose sur des calculs simples, répétés à grande échelle.

Pour qu'un GPT apprenne correctement, il a besoin d'une quantité massive de données. Il existe une masse critique de paramètres (les poids des perceptrons) et de données d'entraînement à atteindre pour que les résultats générés soient réellement utiles aux humains.

C'est ce qui explique leur capacité à traiter de manière interchangeable du texte, du son, des images : tout est transformé en séries de chiffres.



Le terme "Transformers" fait référence à la capacité du modèle à transformer un chiffre en sortie en texte, image ou son. Par exemple, un texte en entrée peut être transformé en son en sortie, ou un son en entrée peut être converti en image. Cela fait partie de ce que l'on appelle la multimodalité. Le terme "Generative" désigne le processus par lequel les chiffres en sortie sont générés et peuvent être décodés selon la modalité souhaitée, en fonction de la manière dont le GPT a été entraîné.

Le résultat généré par le GPT n'est rien d'autre qu'une tentative d'imitation des données d'entraînement. En réalité, pour le GPT, tout est une forme d'hallucination. Ce que nous appelons des "hallucinations" (lorsqu'il invente des faits faux) ne fait aucune différence pour lui, qu'il s'agisse d'inventer une

histoire fictive, de raconter un événement historique, d'expliquer un concept mathématique ou de créer une information totalement erronée.

En outre, les GPT peuvent être biaisés si les données sur lesquelles ils sont formés le sont. Ils peuvent également mémoriser des informations potentiellement dangereuses, telles que des recettes de bombes. C'est pourquoi il est souvent nécessaire, après un entraînement sur des données collectées massivement sur Internet, de les "brider" par des procédures supplémentaires pour limiter les risques.