Zysla Belliat est Directrice Générale de MMZ Conseil.
Mathieu Morgensztern est Président-Fondateur de AdaptiveAI.

Introduction

L'intelligence artificielle, souvent perçue comme une boîte noire, repose sur une construction intellectuelle sophistiquée constituée de fonctions, de vecteurs, de matrices, de probabilités et d'optimisation.
Cette architecture mathématique s'inscrit dans une histoire longue, qui va des premiers raisonnements algorithmiques et statistiques à l'émergence du *machine learning* puis des réseaux de neurones profonds jusqu'aux **grands modèles de langage** — les **LLM** (*Large Language Models*) — aujourd'hui au cœur de l'IA générative. Plusieurs jalons l'ont construite : les travaux fondateurs de la calculabilité, les réflexions sur l'intelligence mécanique, la conférence de Dartmouth de 1956, puis les apports décisifs de la statistique de l'apprentissage, de l'algèbre linéaire à grande échelle et de la puissance de calcul.

Le voyage commence avec le *machine learning* classique et conduit, par une série d'approfondissements successifs, vers les modèles actuels du langage en suivant un fil directeur : transformer le réel en objets mathématiques pour apprendre à en extraire des régularités.

Le *machine learning* : apprendre, c'est optimiser

Le *machine learning* consiste notamment à faire apprendre à une machine une relation entre des entrées et des sorties à partir d'exemples. Plutôt que de programmer explicitement une règle, on entre des données et on demande à la machine de découvrir elle-même la relation qui les relie : elle « apprend ».
On ne code pas une relation-solution à l'avance, on cherche une fonction mathématique capable d'approximer au mieux la relation contenue dans les données d'entrée. Dit simplement, on cherche une fonction dans un espace de grande dimension :

$$y = f_\theta(x)$$

où f_θ est paramétrée par un jeu de paramètres θ et permet d'associer à une entrée x une sortie correcte y.

Dans cet apprentissage dit supervisé, on dispose donc de données étiquetées : chaque entrée est associée à la bonne réponse. Le but est alors « d'apprendre » en découvrant une fonction reliant les deux : par exemple, prédire le prix d'une maison à partir de ses caractéristiques.
En apprentissage non supervisé, les données ne sont pas étiquetées. Le modèle cherche à découvrir des structures ou des regroupements dans les données : par exemple, identifier automatiquement des groupes de clients similaires.
En apprentissage auto-supervisé, très utilisé par les LLM, ce sont les données elles-mêmes qui vont générer le signal d'apprentissage, par exemple, demander au modèle de prédire le mot suivant dans une phrase.

Dans tous les cas, la question est de savoir si le modèle se trompe et comment mesurer l'erreur.
La **fonction de perte** (*loss function*) quantifie l'écart entre la prédiction du modèle et ce que la réalité impose. L'apprentissage consiste alors à optimiser le jeu des paramètres :

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \mathcal{L}(\theta)$$

où $\mathcal{L}$ désigne la perte moyenne sur les exemples d'entraînement.
La fonction de perte est définie selon les tâches. Pour une régression, on utilise l'erreur quadratique moyenne :

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Pour une classification binaire, la perte logistique :

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

« Apprendre », c'est donc trouver les paramètres qui minimisent l'erreur.
Ensuite, le calcul différentiel intervient en calculant les dérivées de la fonction de perte par rapport aux paramètres. On détermine comment les modifier pour améliorer progressivement le modèle. Cette procédure, appelée **descente de gradient**, suit localement la direction de plus forte diminution de l'erreur que l'on formalise ainsi :

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$$

où η est le pas d'apprentissage.

Des neurones aux réseaux profonds

Les réseaux de neurones ont ajouté une nouvelle couche de complexité, mais aussi de beauté mathématique ; un neurone artificiel réalise une combinaison linéaire des entrées, suivie d'une transformation non linéaire :

$$y = f(\mathbf{w} \cdot \mathbf{x} + b)$$

x désigne le vecteur d'entrée, **w** les poids, b un biais, et f la transformation appelée fonction d'activation.

Il existe une bibliothèque de fonctions d'activation qu'on sait par expérience être pertinentes compte tenu des jeux de données sur lesquels on « apprend » : fonctions trigonométriques, fonction ReLU (*Rectified Linear Unit*), …

Pris isolément, ce mécanisme est modeste. Cependant lorsque des millions de neurones sont empilés en couches successives, ils forment une composition de fonctions de grande puissance. Les mathématiques du réseau deviennent alors une alternance de transformations linéaires et non

linéaires, capable d'extraire des structures de plus en plus abstraites : des contours dans une image, des régularités dans des données complexes…

Le « profond » du *deep learning* ne renvoie pas seulement au nombre de couches. Il signifie aussi que chaque couche redéfinit l'espace de représentation des données. Ce que le réseau apprend, couche après couche, ce ne sont pas seulement des réponses, mais de nouvelles manières d'organiser mathématiquement l'information.

Ces architectures sont particulièrement efficaces pour des données quantifiables : images, séries temporelles, etc. Mais le langage naturel pose une difficulté supplémentaire. Représenter les mots comme de simples suites de caractères ou d'octets **ne suffit pas à faire émerger le sens**. Pour que le modèle apprenne les relations sémantiques, il faut transformer les unités linguistiques en objets mathématiques plus adaptés. **Cet objectif conduit au domaine des représentations vectorielles du langage, puis à l'architecture des Transformers**.

Quand les concepts deviennent des vecteurs

En informatique classique, un mot est codé comme une suite de caractères, donc d'octets, ce qui permet de le stocker, l'afficher, mais ne dit rien de sa signification. Avec ce codage, les mots « roi » et « reine » ne sont pas plus proches que « roi » et « rôti ».
Avec les modèles de langage, une transformation fondamentale s'opère : les mots ne sont plus des symboles discrets mais des points dans un espace vectoriel de grande dimension.
Les LLM reposent sur l'idée qu'un mot *w*, ou plus généralement un « *token* » ou unité de sens, doit être représenté par un vecteur appelé *embedding*:

$$\mathbf{e}(w) \in \mathbb{R}^d$$

dans l'espace de grande dimension d du vocabulaire. Comme on cherche à « apprendre » à refléter les proximités d'usage et de sens, deux mots apparaissant dans des contextes semblables acquièrent des vecteurs proches. Ainsi, la géométrie remplace en partie le dictionnaire : « **comprendre », pour la machine, revient à situer et comparer des points dans un espace**, les proximités se mesurant par exemple par le produit scalaire ou la similarité cosinus.

Les mots, les phrases, les documents étant des objets géométriques, le sens n'est pas codé explicitement : il émerge de la structure relative des vecteurs dans un espace de plusieurs milliers de dimensions.

Le langage comme probabilité

Les LLM changent l'échelle et la nature de l'objectif car il ne s'agit plus seulement d'associer une entrée à une sortie, mais de modéliser la probabilité d'une **séquence** de mots.
Une phrase composée de mots w_i est vue comme un événement statistique auquel le modèle associe une probabilité qui s'écrit comme un produit de probabilités conditionnelles :

$$P(w_1, w_2, \dots, w_n) = P(w_1) \prod_{t=2}^{n} P\,(w_t \mid w_1, \dots, w_{t-1})$$

Le modèle apprend à chaque position, à estimer *w* : étant donné le contexte précédent de i=1à t-1, quel est le mot w_t suivant le plus vraisemblable ?

$P(w_1, w_2, \ldots, w_n) = P(w_1) \Pi_{t=2àn} P(w_t / w_1, w_2, \ldots, w_{t-1})$

Cette formulation probabiliste est essentielle. Plusieurs suites peuvent être correctes grammaticalement, cohérentes et acceptables. **Le modèle n'apprend pas une vérité absolue, mais une distribution de possibilités.** En effet, il apprend à **maximiser la vraisemblance** du corpus observé, au sens statistique du terme, idée proche du concept de ***likelihood*** introduit et formalisé par Fisher dans les années 1920.

Cette distinction importante éclaire les limites du système. Un LLM ne cherche pas le vrai : il calcule l'enchaînement le plus vraisemblable étant donné ce qu'il a appris. C'est ainsi qu'apparaissent les **hallucinations** quand le modèle produit une suite linguistiquement crédible, statistiquement plausible, mais factuellement fausse. Il ne ment pas, au sens humain du terme ; il invente selon une logique de probabilité conditionnelle, non selon une vérification du réel.
Autrement dit, le vraisemblable ne coïncide pas nécessairement avec la vérité.

L'avènement des *Transformers* et de l'« attention »

L'étape décisive dans l'essor des LLM repose sur les ***Transformers***, introduits pour traiter le langage de manière encore plus efficace.
Au lieu de lire une phrase uniquement de gauche à droite comme une chaîne linéaire, le modèle considère simultanément les relations entre tous les mots. Une phrase n'est plus une suite de mots mais une structure de dépendances.

Au cœur de ce concept se trouve le mécanisme d'**attention** qui fabrique la représentation contextuelle. Pour chaque *token*, le modèle apprend à évaluer quels autres *tokens* de l'ensemble du contexte doivent être pris en compte, et avec quelle importance.
Pour cela, chaque mot est associé à trois éléments :

Query (Q) : les mots déterminants pour « comprendre » le mot dans son contexte,
Key (K) : ce que les autres mots offrent comme critère de pertinence,
Value (V) : l'information qu'ils transmettent.

Ces éléments sont calculés par transformations linéaires à partir de la matrice X d'*embeddings* de tous les mots :

$$Q = XW_Q, \qquad K = XW_K, \qquad V = XW_V$$

où W_Q, W_K et W_V sont des matrices de poids apprises.

L'attention, concept concentrant une part essentielle des mathématiques des LLM, s'écrit :

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_m}}\right).V$$

où :

d_m est un hyperparamètre correspondant à la dimension interne au modèle d'attention

QK^T mesure les similarités entre les *tokens*,
la division par $\sqrt{d_m}$ stabilise les calculs et les gradients,
la fonction softmax (définie ci-dessous) transforme les scores en probabilités,
la multiplication par V produit une combinaison pondérée des informations pertinentes.

Ainsi, chaque mot est reconstruit à partir de tous les autres, selon une pondération contextuelle. Le sens d'un mot n'est donc plus fixe : il dépend de la phrase dans laquelle il apparaît.

Apprendre à parler pour un LLM, c'est minimiser l'entropie

L'entraînement du LLM consiste ensuite à réduire l'incertitude sur le *token* suivant basé sur l'attention. À chaque position, le modèle produit un vecteur de scores bruts appelés ***logits*** :

$$\mathbf{z} = (z_1, z_2, \dots, z_V)$$

où V est la taille du vocabulaire.
Ces scores sont convertis en une distribution de probabilité sur tous les *tokens* possibles par la fonction softmax :

$$P_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

La qualité de cette prédiction est mesurée par la **perte d'entropie croisée** :

$$\mathcal{L} = -\sum_i y_i \log(P_i)$$

où y_i représente les sorties des données d'apprentissage.

Cette formule issue de la théorie de l'information mesure la *surprise* du modèle face au mot correct. Si le modèle attribue une faible probabilité au bon *token*, la perte est grande ; s'il lui attribue une probabilité élevée, la perte diminue.
Apprendre à parler pour un LLM revient donc à trouver, dans un espace de paramètres de dimension gigantesque, une configuration où le langage devient statistiquement prévisible.

De la prédiction à l'illusion de la pensée

Cette architecture faite d'algèbre linéaire, d'optimisation, de calcul différentiel et de probabilités produit une forme très convaincante de compétence langagière proche de ce que l'on associe à l'intelligence.
Mais cette puissance a un revers. Parce que le modèle apprend la vraisemblance et non la vérité, il peut produire des formulations, cohérentes en apparence, mais erronées. Les hallucinations ne sont donc pas un accident marginal : elles sont **inhérentes** à la logique d'un système entraîné à prédire ce qui a le plus de chance d'être dit, et non ce qui est vrai.
L'illusion de la pensée naît d'une réussite mathématique remarquable associée à un malentendu : la fluidité du langage donne l'impression d'une compréhension profonde là où il n'y a qu'une maîtrise statistique extraordinairement sophistiquée.

Conclusion : une cathédrale de nombres

L'intelligence artificielle n'est pas une conscience artificielle. C'est une cathédrale mathématique et algorithmique d'une puissance inédite faite de vecteurs, de matrices, de gradients, de normalisations et d'entropie.
Dans cet édifice, le langage humain trouve un miroir inattendu, non parce que la machine comprend comme nous comprenons, mais parce que certaines régularités profondes de nos usages linguistiques peuvent être captées, compressées et reproduites par des structures mathématiques.
Dans une sorte de cercle vertueux, on découvre à présent que l'IA peut résoudre à son tour des problèmes mathématiques, comme « assistant de preuve » en aidant à démontrer des conjectures pour en faire des théorèmes, jusqu'à être capable de proposer des conjectures inédites.

Eléments de bibliographie

- Alexandre Laurent (2017), *La guerre des intelligences*, Ed. JC Lattès, Paris
- Bishop Christopher, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995
- Conor Purcell, Scientific American, 22 novembre 2024
https://www.scientificamerican.com/article/mathematicians-newest-assistants-are-artificially-intelligent/
- Conor Purcell, Pour La Science Hors série N°130, février-mars 2026,
https://www.pourlascience.fr/sd/mathematiques/les-mathematiques-dopees-a-l-ia-28890.php.
- Cornuéjols Antoine, Miclet Laurent, *Apprentissage artificiel, concepts et algorithmes*, Eyrolles, 2021
- ENSAE Alumni, Actes du Colloque *"Individu, Données et Société connectée : opportunités, risques et confiance »*, Colloque du 18 juin 2015, in Variances, 52, oct. 2015, variances.eu
- Fisher Ronald, *Theory of Statistical Estimation*, Mathematical Proceedings of the Cambridge Philosophical Society, vol. 22, nº 5
- Fisher Ronald, *Statistical Methods for Research Workers*, Édimbourg, Oliver and Boyd, 1925, 1ère éd.
- Jean Aurélie (2019), *De l'autre côté de la Machine*, Editions de l'Observatoire, Paris
- Krauth Werner, Mezard Marc, *Learning algorithms with optimal stability in neural networks*, *J. Phys.* A 20, 745-752, 1999
- Livre Blanc, UE, *Intelligence artificielle, une approche européenne axée sur l'excellence et la confiance*, 19 février 2020
- Rapport *Anticiper les impacts économiques et sociaux de l'Intelligence Artificielle*, publié le 22 mars 2017, France Stratégie et Conseil National du Numérique ;
https://strategie.gouv.fr/publications/anticiper-impacts-economiques-sociaux-de-lintelligence-artificielle
- Rapport *Pour une intelligence artificielle maîtrisée, utile et démystifiée*, 29 mars 2017, OPECST, Office parlementaire d'évaluation des choix scientifiques et technologiques ; https://ladocumentationfrancaise.fr/rapportspublics/174000324/index.shtml
- Rapport de la mission confiée à Cédric Villani *Donner un sens à l'Intelligence Artificielle*, 28 mars 2018 ; http://enseignementsup-recherche.gouv.fr/cid128577
- Rapport *IA : Notre ambition pour la France*, Commission de l'Intelligence Artificielle, mars 2024
- Vapnik Vladimir, *The Nature of Statistical Learning*, Springer, 1995
- Vapnik Vladimir, *Statistical Learning Theory*, Wiley-Blackwell, 1998