onepoint. × weave
beyond the obvious

VERS UNE NOUVELLE ALLIANCE HOMME MACHINE ?

IA

Livre blanc

Poids statistiques
Apprentissage supervisé
Validation
Prédictions
Entraînement
Réseaux de neurones profonds
Généralisation
Algorithmes
Sciences cognitives
Savoir inné
Savoir acquis
No Free Lunch Theorem
Traitement du langage naturel
Réseaux de convolution
Architecture
One-Shot Learning
Style Transfert
Variables lattentes
Représentations parcimonieuses
Échantillonage
Intelligence augmentée
Modèles génératifs
Morphing
Compression d'image
Machine Learning bayésien
Adversarial Examples
George Edward Box
TensorFlow
Approximation MAP
Inférence
Intervalle de confiance
Réseaux de neurones bayésiens

Intelligence Artificielle

Vers une nouvelle alliance Homme-Machine ?

Pirmin Lemberger - Nicolas Parent
Thomas Scialom - Fabrice Mauléon

Biographies des auteurs

Pirmin Lemberger

Pirmin Lemberger est directeur scientifique chez weave, physicien théoricien de formation (EPFL), il anime la communauté IA. Ses tribunes paraissent dans IT for Business et le JDN ainsi que sur weave.eu > IA Lab, le site du cabinet. Il est l'auteur principal du livre *Big Data* et *Machine Learning* paru chez Dunod en 2016.

pirmin.lemberger@weave.eu

Nicolas Parent

Nicolas Parent est consultant chez weave. Ingénieur centralien, il est passioné par les nouvelles technologies et par l'effet qu'elles auront sur la société de demain. Cela le conduit à travailler sur des sujets comme la blockchain ou l'Intelligence Artificielle.

nicolas.parent@weave.eu

Thomas Scialom

Thomas Scialom est associé chez reciTAL, ingénieur de formation, il quitte la banque en 2016 pour se consacrer à la recherche en Intelligence Artificielle. Ses travaux portent sur le Natural Langage Processing, l'art de permettre aux machines de comprendre et communiquer avec les humains. L'une de ses applications est le résumé automatique de texte, sujet sur lequel il prépare un doctorat au Laboratoire d'informatique de Paris 6.

thomas@recital.ai

Fabrice Mauléon

Fabrice Mauléon est un expert français en Business Transformation, reconnu en France et à l'international. Auteur, speaker, consultant et professeur, il accompagne les entreprises sur la double problématique de l'innovation et du digital, en adaptant les approches et méthodes du design thinking, business models, etc. aux enjeux de l'Intelligence Artificielle en général et de la révolution digitale en particulier.

fabricemauleon@gmail.com

Préface

Pirmin Lemberger - **weave Business Technology**

Pas un jour sans qu'un article dans la presse quotidienne n'évoque le sujet. Pas un jour sans que dix articles taggés #IA ne soient recyclés dans la grande lessiveuse des réseaux sociaux, trop souvent hélas, sans qu'aucun commentaire ni aucun point de vue personnel ne vienne accompagner l'article proposé à la multitude.

Comment alors s'y retrouver dans un tel capharnaüm ? Comment s'informer dans le brouillard permanent de la « com » et des « likes » de complaisance ?

> *Pour voir un peu plus loin, il faut accepter de creuser un peu plus profond.*

Pour les **chercheurs** en IA, les grandes conférences internationales NIPS, ICML ou ICLR, les blogs spécialisés et la caverne d'Ali Baba qu'est arXiv sont aujourd'hui les principales sources d'information.

À l'autre extrémité du spectre, on trouve une pléthore d'articles destinés à un **public sans connaissances techniques**. Sont mises en avant la présentation d'exemples concrets, une approche « orientée usages » et la description de solutions pragmatiques et « actionables » pour faire « parler la data », comme il convient de dire en novlangue business IT.

Chez *onepoint* x *weave*, nous pensons qu'entre ces deux pôles existe une niche pour des articles destinés à un public de non-spécialistes, qui souhaitent cependant comprendre les principaux enjeux techniques et scientifiques de l'IA au prix d'**un effort modeste.** Celui que l'on consacre par exemple à la lecture d'un article de vulgarisation scientifique dans *Pour la Science*, *Wired* ou *Quanta Magazine*.

Fidèles à notre signature « *Beyond the obvious* », nous estimons qu'il faut **délaisser l'approche usuelle par les usages** au profit d'une approche de l'IA par les **grandes questions.** Elles structurent le sujet, notamment dans le cadre du **Deep Learning**, qui constitue aujourd'hui le cadre prédominant de l'IA. Quelles sont les dernières avancées en matière d'architecture des réseaux de neurones profonds (Deep Learning) ? Quelles en sont les limites ? Quelles sont les questions ouvertes à ce jour ?

> *Comment ces nouvelles avancées techniques dessinent-elles progressivement une nouvelle alliance entre l'homme et la machine ?*

Tels sont les objectifs que nous nous sommes assignés dans nos articles publiés tout au long de l'année 2018 sur **IA Lab**.

En voici une brève rétrospective :

1. **Le Machine Learning décrypté** est une brève introduction qui élucide ce que l'on entend par « apprentissage » dans le cadre du machine learning supervisé et qui décrit les compromis qu'il faut résoudre.

2. **Les pigeons superstitieux, l'inné et l'acquis en IA** aborde la question du **savoir à priori,** qu'il est judicieux d'intégrer dans un modèle prédictif. C'est une question sur laquelle les sciences cognitives ont beaucoup à apprendre à l'IA.

3. **Existe-t-il des IA créatives ?** Voilà un sujet qui a suscité beaucoup de fantasmes et d'articles à sensation. Qu'en est-il vraiment ? En quoi consiste la « **créativité** » d'un réseau de neurones ? En quoi diffère-t-elle de celle d'un humain ?

4. **Construire des machines qui savent douter (raisonnablement)...** Évaluer de manière fiable le **risque d'erreur** d'une prédiction faite par un algorithme prédictif est aujourd'hui chose difficile. L'approche bayésienne du Machine Learning (ML) permettra peut-être de construire des machines qui savent qu'elles ne savent pas, ce qui ouvrira les portes du ML aux applications critiques.

5. **Lorsque le Deep Learning met vos émotions à nu.** Qu'il s'agisse d'analyser un texte écrit, l'expression d'un visage ou les variations dans les intonations d'une voix, le Deep Learning a récemment fait beaucoup de progrès. Sur quelles idées reposent-ils ? Quelles sont les questions éthiques que posent ces systèmes ?

6. **Deep Transfer Learning – le traitement du langage à l'aube d'une révolution ?** L'année 2018 a été particulièrement faste en ce qui concerne les progrès en NLP (traitement du langage naturel). Sur quels principes reposent-ils ? Quelles sont les limites de ces approches ?

7. **L'intelligence artificielle distribuée** décrit une méthode empirique qui permet de tirer profit de la puissance de calcul disponible sur nos terminaux mobiles pour entraîner des modèles de Deep Learning tout en préservant la confidentialité des données.

8. **Soft skills humains et IA : quelle répartition des compétences ?** Si les machines sont dès aujourd'hui capables de se substituer aux humains pour des tâches cognitives simples, on peut imaginer que celles qu'elles assumerons dans le futur, grâce aux progrès conjoints de l'IA et des sciences cognitives, seront de plus en plus complexes. Quelle sera alors la nouvelle alliance entre l'homme et la machine ?

Notre intention pour 2019 est de persévérer dans cette voie, en privilégiant les sujets de fond abordés sous l'angle de la vulgarisation scientifique.

Nos articles sont disponibles sur le site *IA Lab by weave*[1].

(1) URL de IA Lab by weave : weave.eu/le-mur/lab-intelligence-artificielle/

Remerciements

Les auteurs tiennent ici à remercier Olivier Reisse, associé fondateur de *weave Business Technology*, pour ses encouragements tout au long de l'année et, plus encore, pour avoir su créer un environnement de travail à la fois exigeant et bienveillant qui donne envie de se lever le matin. A titre personnel, je souhaite aussi remercier Thibault Hache, notre Digital Marketing Manager, qui, avec une patience à toute épreuve, s'attache à diffuser notre travail et à le mettre en valeur. Je remercie également Quentin Talamoni pour avoir mis sa ténacité et sa sensibilité de Designer Graphique au service de la mise en forme de ce recueil à partir de sources disparates. Je remercie enfin Sandy Malosse pour les nombreuses illustrations, à la fois drôles et décalées, qu'elle a conçues pour la version web de ces articles.

Pirmin Lemberger - **weave Business Technology**

Sommaire


Apprentissage supervisé
Prédictions
Paramètres
Entraînement
Validation
Réseaux de neurones profonds
Généralisation
Algorithmes

Le Machine Learning décrypté

Pirmin Lemberger - weave Business Technology

Résumé

Aujourd'hui, 95% des applications business de l'IA relèvent du machine learning supervisé. C'est ainsi qu'on appelle un ensemble de techniques mathématiques et statistiques qui permettent à une machine d'apprendre à faire des prédictions à partir d'exemples.

1. L'apprentissage supervisé au service du business

Grâce à l'**apprentissage supervisé**, les entreprises peuvent établir des prédictions "business" à partir des données. Les cas d'usage sont de plus en plus nombreux. On peut, par exemple, déterminer la probabilité qu'un individu rembourse un crédit, faire des recommandations de produits et/ou services à telle ou telle cible, évaluer le risque de résiliation, prédire le risque d'être atteint d'une certaine affection à partir de symptômes etc….

Qu'entend-on par apprendre au juste ?

Au préalable et avant toute démarche d'apprentissage, il est nécessaire pour le data scientist de réaliser un travail de préparation. Cela passe par un nettoyage de données – dans le cas où certaines d'entre elles manquent ou sont aberrantes – mais aussi par le croisement de plusieurs sources de données disparates, afin de correctement alimenter le ou les algorithmes qui ne sont capables d'ingurgiter que des chiffres. A ce jour, il existe une petite dizaine d'algorithmes ou de classes d'algorithmes utilisés en pratique. Il faut bien comprendre que le travail d'un data scientist ne consiste pas à inventer de nouveaux algorithmes mais à choisir, en utilisant son intuition et l'expertise métier, quelles sont les données vraiment utiles pour faire des prédictions.

On parle d'entraînement d'un **algorithme**. En fait c'est au moment d'entraîner un algorithme qu'il apprend. L'entraînement consiste pour un algorithme à passer en revue des données pour lesquelles on connaît le résultat qu'un prédicteur idéal devrait produire (c'est l'origine du terme supervisé) et à essayer de faire différentes prédictions. En ajustant petit à petit certains paramètres l'algorithme va faire en sorte que les erreurs qu'il commet diminuent progressivement, pour atteindre finalement une valeur acceptable.

Le compromis fondamental du machine learning

L'étape précédente consiste donc dans un premier temps pour un algorithme à parvenir à reproduire des résultats connus. Mais ce qui est vraiment utile et intéressant c'est naturellement de faire des **prédictions** pour des données que l'algorithme n'a jamais vues au préalable. On parle à ce titre de **généralisation**.

Pour que cette généralisation soit possible, il faut que deux conditions soient réunies. La première est qu'il faut disposer de suffisamment de données dans le jeu d'entraînement pour que les données encore jamais vues soient raisonnablement proches des exemples.

L'ajustement des **paramètres** durant l'entraînement conduira alors l'algorithme à faire une extrapolation raisonnable à partir des données déjà vues. Par raisonnable on entend le fait que la prédiction entre deux données connues va varier de manière assez régulière. La seconde condition est donc que les prédictions soient relativement régulières.

Il s'agit alors d'éviter un double écueil. Si on exige de l'algorithme qu'il fasse des prédictions très régulières, cela se fera au prix d'une rigidité qu'il l'empêchera de parvenir à une erreur faible sur les données d'entraînement. Si en revanche on n'exige aucune forme de régularité de l'algorithme il pourra s'adapter au jeu de données d'entraînement, au prix de contorsions peu vraisemblables ce qui le conduira à commettre d'importantes erreurs sur des données qu'il n'a jamais vues.

Tout l'art du machine learning consiste par conséquent à résoudre ce dilemme : ni trop régulier, ni pas assez !

En pratique on met à l'épreuve un algorithme sur des données dites de validation, distinctes de celles utilisées pour l'entraînent et on ajuste le degré de flexibilité pour que l'erreur commise sur ces données de validation soient aussi faible que possibles. Un fois la flexibilité optimale trouvée, on va évaluer la précision de l'algorithme sur un troisième jeu de données, dites de test, distinctes à la fois des données d'**entraînement** et de **validation**. Cela donnera une bonne estimation de la précision de l'algorithme une fois qu'il aura été mis en production. À condition toutefois que les données de test soient bien représentatives de la réalité à laquelle l'algorithme sera confronté.

Terminons ce bref survol en énumérant quelques qualités que l'on peut attendre d'un algorithme de machine learning. La première est la précision : l'algorithme doit faire peu d'erreurs. Encore faut-il préciser comment on les mesure. On utilise pour cela différentes fonctions de coût qui quantifient les erreurs commises. L'algorithme ne doit pas consommer trop de ressources, ni pour son entraînement, ni pour la phase de prédiction surtout si cette dernière doit se faire en temps réel. Enfin, le RGPD exige dans bien des cas que l'on soit en mesure d'**expliquer** les prédictions que fait un système. C'est là une contrainte forte que ne satisfont pas tous les algorithmes, notamment les réseaux de neurones profonds, le fameux deep learning.

En pratique on peut être amener à choisir entre un algorithme très performant en terme de précision et un autre moins performant mais plus « explicable ».

Enfin, pour être juste, il faut préciser que la réalité est plus complexe que ce que ne le laisse penser notre petite esquisse. La majorité des problèmes de prédictions combinent en réalité les prédictions de plusieurs algorithmes pour parvenir à de bons résultats, mais laissons ces subtilités aux data scientists qui s'en délectent.


Traitement du langage naturel
No free lunch theorem
Systèmes de recommandations
Architecture
Savoir acquis
One shot learning
Machine learning supervisé
Savoir inné
Sciences cognitives
Réseaux de convolution

Les pigeons superstitieux, l'inné et l'acquis en IA

Pirmin Lemberger - **weave Business Technology**

Résumé

Cet article examine le rôle du savoir à priori par rapport à l'apprentissage à partir de données dans le Machine Learning, dit autrement le rôle de l'inné par rapport à celui de l'acquis. Deux exemples tirés du règne animal illustreront notre propos de manière très concrète. La nécessité d'un savoir à priori est formulée de manière théorique (sans formalisme) puis illustrée au moyen de plusieurs exemples de l'IA comme la classification d'image ou la reconnaissance d'écriture manuscrite. Le rôle de ce savoir inné dans la compréhension du langage naturel est évoqué.

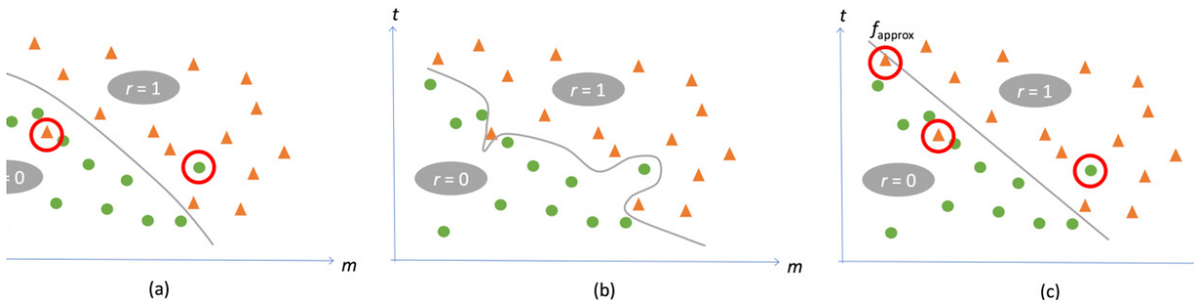
1. Pourquoi les rats sont plus intelligents que les pigeons

Qu'est-ce que la faculté d'**apprendre** sinon la capacité à transformer de l'expérience en une expertise ? Automatiser ce processus est aujourd'hui l'un des principaux objectifs de l'IA dont le **machine learning** (ML) supervisé est la forme la plus courante, la plus solide sur le plan conceptuel et la plus rentable économiquement [MLD, MLN].

A l'heure où nous essayons de construire des systèmes apprenants, il n'est peut-être pas infondé de nous tourner vers la nature pour examiner quelles « solutions » elle a mis en œuvre pour produire des mécanismes d'apprentissage efficaces, notamment dans le règne animal. Apprendre à partir de l'expérience est évidemment vital pour la survie d'un animal, notamment lorsqu'il doit trouver de la nourriture en quantité et en qualité suffisante.

En prenant un peu de liberté avec les termes, on pourrait par ailleurs envisager l'évolution des espèces elle-même comme un gigantesque processus d'apprentissage (par renforcement) étalé sur plusieurs centaines de millions d'années dans la biosphère terrestre. Dans le comportement d'un animal interviennent donc deux processus d'apprentissage. Le premier, à très long terme, pilote en quelque sorte l'évolution de l'espèce par le mécanisme de sélection naturelle qui fabrique des individus aptes à la survie dans leur biotope et à… apprendre efficacement ! Le second mécanisme d'apprentissage, à l'échelle de la vie d'un individu, lui permet d'optimiser son comportement pour maximiser son bien-être durant sa brève existence. C'est ce second processus qui nous intéresse plus directement.

Plusieurs expérimentations en **psychologie cognitive** ont cherché à comprendre les ressorts de l'apprentissage chez différentes espèces. Dans une expérience désormais célèbre, le physiologiste B.F. Skinner [BFS] a enfermé des pigeons affamés dans une cage, la nourriture leur étant distribuée à intervalle régulier. L'observation cruciale de Skinner a été de constater que les pigeons avaient tendance à reproduire l'activité dans laquelle ils étaient engagés (roucouler, secouer la tête, picorer etc…) lorsque la nourriture leur est distribuée pour la première fois. Les pigeons étaient superstitieux dans le sens où ils déduisaient une relation de cause à effet, là où il n'y avait en réalité qu'une concomitance fortuite d'évènements.



Un échantillon d'observations réparties en deux catégories : les points orange et les points verts. Apprendre à classer de nouveaux points revient à tracer une courbe séparant, au mieux ces deux catégories. Trois possibilités sont illustrées.

Chaque nouvelle distribution de nourriture avait dès lors de grandes chances de trouver l'animal occupé à son activité superstitieuse contribuant ainsi à renforcer encore la pauvre bête dans sa fausse croyance.

Le comportement des rats dans leur quête de nourriture est très différent [UML]. On sait qu'ils sont excessivement prudents face à un aliment qu'ils n'ont jamais vu. Ils n'en ingèrent qu'une très faible quantité et en mémorisent toutes les caractéristiques olfactives et visuelles. Si l'aliment devait provoquer un mal-être, même léger, il serait simplement proscrit de toute consommation future. Des chercheurs facétieux se sont alors demandés s'il était possible d'induire des rats en erreur au moyen de faux stimulus. Après chaque ingestion de nourriture par ces animaux ils leurs ont infligés des chocs électriques douloureux. De manière très surprenante, ces chocs électriques n'ont pas réussi à conditionner les comportements des rats face à la nourriture. Des millions d'années d'évolution leur ont visiblement inculqué la ferme conviction qu'aucune nourriture ne peut être à l'origine d'un choc électrique ! La capacité des rats à apprendre correctement à partir des observations tient en l'occurrence à ce **savoir inné**. On peut en revanche interpréter l'ingénuité des pigeons comme un manque de savoir à priori sur ce qui est raisonnable.

De quoi l'on déduit deux vérités fondamentales : d'une part que les rats ont plus de bon sens que les pigeons (CQFD) et d'autre part que la faculté d'apprentissage tient à la confrontation de l'observation avec un **savoir à priori**. C'est le sujet de la suite de cet article.

2. Les biais indispensables du Machine Learning

Pour ceux qui en douteraient, la petite digression animalière qui précède ne relève pas de la simple blague potache. Bien au contraire, la question du savoir à priori se niche au cœur même du Machine Learning, qu'il s'agisse des applications pratiques, de sa théorie ou des questions encore ouvertes de l'IA comme la conception de machines qui comprennent (vraiment) le langage naturel.

Pour concrétiser ce propos, examinons l'exemple élémentaire d'apprentissage automatique illustré dans la figure ci-dessous, ce qui nous permettra de faire pièce à deux idées reçues :

Idée reçue n°1 : « *Le ML finalement c'est avant tout de la data* ». Faux !

Idée reçue n°2 : « *Dans le ML il faut éliminer tous les biais* ». Re-faux !

À partir d'un ensemble d'observations réparties en deux catégories, les points orange et les points verts, il s'agit de prédire la couleur d'un nouveau point à partir de sa position. En d'autres termes il s'agit de tracer une courbe qui sépare au mieux les deux catégories de points. La figure propose trois solutions : (a) semble raisonnable mais commet toutefois 2 erreurs, (b) semble ne commettre aucune erreur mais est vraisemblablement trop spécifique au jeu de données pour être **généralisable**, (c) utilise une simple droite et commet 3 erreurs.

Le **bon sens** ou, si l'on préfère, une certaine « **connaissance** métier » nous incitera probablement à choisir la solution (a) car c'est celle qui semble apte à la généralisation.

Notre capacité à prédire repose donc non seulement sur les données que nous observons mais aussi sur un biais inductif qui nous fait penser que certaines prédictions seront meilleures que d'autres.

Cette observation s'avère vraie en général : sans données, pas de prédictions certes, mais sans spécialisation du modèle non plus ! Cette impossibilité est la conséquence d'un résultat théorique, connu sous le sobriquet de **No-Free-Lunch-Theorem** (NFL), que l'on peut formuler en terme intuitifs ainsi :

Il n'existe pas d'algorithme universel, un algorithme doit nécessairement être spécialisé à un domaine suffisamment restreint pour que l'on soit en mesure de formuler un biais inductif.

Plus précisément on peut montrer que :

« Si quelqu'un prétend avoir découvert un algorithme universel, capable d'apprendre à partir des seules données, on pourra le détromper en construisant un jeu de données que son algorithme ne parviendra pas à exploiter, quel que soit la quantité d'observations qu'il ingurgite, alors qu'un autre algorithme y parviendra ! »

Une version plus fine du résultat précédent [UML] quantifie le nombre d'observations nécessaires à un apprentissage efficace en fonction de la fréquence et de l'amplitude des erreurs de prédiction que l'on est prêt à admettre et, c'est le point essentiel, en fonction de la richesse de notre connaissance à priori[1]. Plus cette connaissance à priori est faible, plus il nous faudra d'observations pour apprendre à faire des prédictions fiables. Dans la situation extrême où nous ne connaissons rien à priori, il nous faudra une infinité d'observations et l'apprentissage sera par conséquent impossible, ce qu'énonce précisément le NFL.

Après ce petit détour indispensable par la théorie du ML, revenons aux implications pratiques de l'importance du **savoir à priori**, avec quelques exemples moins académiques que la classification binaire de la figure 1.

Les systèmes de recommandations

Les systèmes de recommandations sont l'une des applications les plus courantes du ML supervisé. En bref, il s'agit de prédire l'affinité d'un client pour un produit ou un service à partir d'un historique de comportement (d'achat, de location, de consultation etc…).

La connaissance à priori exploitée dans ce cas consiste pour part à faire un choix éclairé des caractéristiques prédictives des personnes à qui l'on souhaite faire des recommandations : comment caractériser leur goût, leurs envies, leur historique d'interaction avec un site d'e-commerce etc. Il consiste aussi à sélectionner un modèle prédictif capable de prédire une affinité pour un produit à partir de ces caractéristiques.

Bref, tout l'**art du data scientist** consiste précisément à trouver le bon biais inductif !

La classification d'images

La classification d'image est aujourd'hui un problème essentiellement résolu de l'IA, grâce notamment aux réseaux de neurones de convolution (CNN) [LMA]. Il va de soi que la catégorisation d'objet ne devrait dépendre ni de sa position (**invariance de translation**), ni de son orientation dans l'image (invariance de rotation). De fait, les CNN incorporent dans leur architecture le principe d'invariance de translation. En revanche ils n'incorporent pas l'invariance par rotation, raison pour laquelle ils devront apprendre ce fait à partir d'images qui

(1) Dans le cadre de la théorie mathématique de l'apprentissage PAC on mesure la complexité de la classe d'hypothèses parmi laquelle notre algorithme sélectionne un prédicteur. Plus la classe est riche moins notre biais inductif est important. Les mesures de complexité sont par exemple la VC-dimension ou la complexité de Rademacher [UML].

représentent un même objet sous des angles différents.

Des recherches récentes [HCN] d'un des pionniers des réseaux neuronaux, G. Hinton, tentent d'incorporer l'invariance de rotation dans une architecture d'un nouveau type appelés **capsules networks**.

L'enjeu est celui que nous avons mentionné précédemment : avec un savoir à priori plus important, on pourra réduire drastiquement le nombre d'observations nécessaires à l'entraînement !

La reconnaissance de l'écriture manuscrite :

Les algorithmes classiques de reconnaissance de l'écriture manuscrite exploitent le ML supervisé et sont entraînés avec des milliers d'exemplaires de chaque lettre de l'alphabet. Ils utilisent les CNN mentionnés précédemment. Dans un travail récent [HLC, SMD], l'équipe du prof. J. Tenenbaum du MIT est parvenu à construire un système qui apprend à lire comme le font les humains, à partir de seulement quelques exemples de chaque lettre et même d'un seul exemple (**one-shot learning**). Pour cela les chercheurs ont construit un modèle détaillé qui explique comment les humains calligraphient un caractère quel qu'il soit, typiquement pas juxtaposition d'une succession de segment de courbes. Après avoir conçu ce **modèle statistique d'écriture** (dont les détails sont complexes), ils l'ont injecté dans le modèle de reconnaissance de caractères. Ce nouveau système, qui sait désormais comment les humains tracent des caractères, s'avère plus fiable dans la tâche de reconnaissance de l'écriture manuscrite que ne le sont les humains et les réseaux de neurones les plus performants !

Là encore, la richesse du savoir à priori incorporé au système l'a en quelque sorte dispensé d'avoir à l'apprendre et a contribué à diminuer d'un facteur mille le nombre d'observations nécessaires à l'apprentissage !

3. Une clé pour les progrès de l'IA

Grâce aux exemples précédents, on constate que le savoir à priori est au cœur de deux problématiques liées et fondamentales de l'IA :

- La performance du processus d'apprentissage comprise comme la capacité d'un système prédictif à exploiter des jeux de données de petite taille (**Small Data**) [SMD].

- L'idée d'incorporer à une IA un ensemble d'éléments de **bon sens** compris comme un ensemble de faits que l'algorithme ne devrait pas avoir à apprendre.

Ce savoir à priori peut revêtir deux formes. La première consiste en une connaissance, plus ou moins détaillée, du mécanisme qui produit les observations, c'est l'exemple du mécanisme d'écriture qui explique comment sont dessinées des lettres. La seconde consiste à postuler que certains types de prédictions sont plus plausibles que d'autres, c'est l'exemple de la classification binaire où l'on présuppose que la courbe séparatrice ne fait pas trop de contorsions.

Ce qui rend fascinante la question du savoir inné en IA, c'est qu'elle est largement ouverte et suscite de vifs débats entre les spécialistes eux-mêmes. Dans le domaine du **traitement du langage naturel** (NLP) en particulier, un débat récent [IPR] a permis de confronter les points de vue de deux géants de l'IA, *Yann LeCun* (YLC), l'inventeur créateur des CNN, et *Christopher Manning* (CM), un linguiste de renom à Stanford. Sans être complètement irréconciliables, leurs points de vue sont cependant très contrastés.

YLC soutient que la tendance ces dernières années dans le modèle de Deep Learning va dans le sens de toujours moins de savoir à priori incorporé dans l'**architecture** des RN et de toujours plus de données. En effet, il faut savoir que les modèles de Deep Learning utilisés à ce jour en NLP ne connaissent rien de la grammaire ni de la syntaxe d'une langue et moins encore du sens des mots qui relève d'une expérience sensorielle de la réalité. Les modèles de traduction automatique se contentent de prédire quel mot succède à telle suite de mots dont ils

ignorent cependant complètement le sens ! À l'appuis de sa thèse, YLC invoque précisément les progrès réalisés récemment dans ce domaine. S'il ne nie pas la nécessité d'inclure un savoir à priori dans les modèles pour construire des machines authentiquement « comprenantes », il estime cependant qu'il doit émerger de progrès en apprentissage non supervisé plutôt que d'être encodé en dur dans l'architecture des RN.

CM ne nie pas ces progrès mais il estime que le NLP a été un peu perverti par cette approche par force brutale (toujours plus de données, toujours plus de GPU !) car elle tend à éluder les questions difficiles sur la nature du savoir inné à l'œuvre dans l'apprentissage d'une langue chez les humains. Il estime que les modèles utilisés en NLP aujourd'hui sont beaucoup trop superficiels en ne permettant pas l'émergence d'une couche de représentation symbolique dans laquelle se niche la capacité d'abstraction. Pour cela, dit-il, il faut se poser la question de la nature du savoir à priori à injecter dans l'architecture des RN. À l'appui de son propos, il invoque le **mécanisme d'attention** [LMA] qui est un parti pris d'architecture qui a fait ses preuves en NLP et en description d'images.

La nature a de tout évidence trouvé un excellent compromis entre savoir à priori et apprentissage à partir de l'expérience. Pour l'instant ce compromis garde largement son mystère. Pour l'élucider, des disciplines comme les **sciences cognitives** ou la **linguistique** pourraient donc jouer un rôle central dans les avancées de l'IA.

Références

- [BFS] **Superstition in the pigeon**, *B. F. Skinner*, Journal of Experimental Psychology 38, p168-172 – juin 1947
  http://psychclassics.yorku.ca/Skinner/Pigeon/

- [UML] **Understanding Machine Learning: From Theory to Algorithms**, *S. Shalev-Shwartz et S. Ben-David*, Cambridge University Press – mai 2014
  www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/

- [HCN] **Understanding Hinton's Capsule Networks**, *M. Pechyonkin*, Medium – novembre 2017
  https://medium.com/ai%C2%B3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

- [IPR] **What innate priors should we build into the architecture of Deep Learning systems?** *Y. LeCun et Ch. Manning*, YouTube – février 2018
  www.youtube.com/watch?v=fKk9KhGRBdI

- [AIL] **AI's Language Problem**, *W. Knight*, MIT Technology Review – août 2016
  www.technologyreview.com/s/602094/ais-language-problem/

- [HLC] **Human-level concept learning through probabilistic program induction**, *B.M. Lake, R. Salakhutdinov, J.B. Tenenbaum*, ScienceMag vol 350 – décembre 2015
  http://web.mit.edu/cocosci/Papers/Science-2015-Lake-1332-8.pdf

- [MLD] **Le Machine Learning décrypté**, *P. Lemberger*, IA Lab weave – avril 2018
  https://weave.eu/machine-learning-decrypte

- [MLN] **Le ML pour les nuls**, *P. Lemberger*, IA Lab weave – août 2017
  https://weave.eu/nouveau-chantier-securite-it-machine-learning#sect_2

- [LMA] **Le mécanisme d'attention en IA**, *P. Lemberger*, IA Lab weave – janvier 2018
  https://weave.eu/mecanisme-dattention-simple-astuce-mecanisme-universel/

- [SMD] **Plus fort que le Big Data : le Small Data**, *P. Lemberger*, IA Lab weave – septembre 2017
  https://weave.eu/plus-fort-big-data-small-data/


Échantillonage
Modèles génératifs
Compression d'image
Auto-encodeurs variationnels
Représentations parcimonieuses
Les Generative Adversarial Networks (GAN)
Style transfer
Morphing
Variables latentes
Intelligence augmentée

Existe-il une IA créative ?

Pirmin Lemberger - weave Business Technology

Résumé

Certains modèles de machine learning sont aujourd'hui capables de générer des images fictives d'un réalisme surprenant. Cet article est une introduction aux modèles génératifs. Il décrit comment fonctionnent les deux principaux modèles utilisés pour créer ou modifier des images : les Variational Auto-Encoders (VAE) et les Generative Adversarial Networks (GAN). Plus largement, il aborde aussi la question de savoir en quoi consiste au juste leur « créativité ».

1. IA – après l'apprentissage, la création ?

Si l'IA désigne globalement les techniques qui visent à automatiser, un jour, l'ensemble des processus cognitifs humains, il faut bien se rendre à l'évidence : l'immense majorité des applications ne reposent aujourd'hui que sur quelques formes rudimentaires d'apprentissage. L'**apprentissage supervisé**, qui présuppose la disponibilité d'exemples de référence pour lesquels on connaît la réponse souhaitée, et l'**apprentissage par renforcement**, qui consiste à optimiser une stratégie par interaction avec un environnement en vue de maximiser une récompense, en sont les deux formes les plus courantes.

Plus récemment, on a vu apparaître des applications du Deep Learning dans le domaine de la création artistique, grâce aux **modèles génératifs** qui implémentent comme nous le verrons une forme simple d'**apprentissage non-supervisé**. Ainsi sait-on aujourd'hui visualiser avec *Deep Dream* les rêves (ou les cauchemars peut-être) d'un réseau de neurones, recréer un Rembrandt plus vrai que nature ou encore appliquer un style Van Gogh à n'importe quelle photo, comme l'illustre la figure 1. Le domaine « littéraire » n'est pas en reste puisque des réseaux de neurones récurrents sont désormais capables de rédiger des pastiches à la manière de Shakespeare. Enfin, sans parvenir pour l'instant à égaler Miles Davis, les réseaux de neurones se sont également essayés à l'improvisation jazz.

Bien que les réseaux génératifs profonds en soient à leurs premiers balbutiements, on pourrait dès à présent y consacrer un ouvrage entier. Notre objectif ici est simplement de répondre à la question figurant dans le titre de cet article. Sans spéculations grandiloquentes, sans envolées lyriques, mais de manière argumentée, en nous appuyant sur une compréhension approfondie des mécanismes à l'œuvre dans les deux principaux modèles génératifs utilisés pour créer des images synthétiques réalistes.

2. Que sont les modèles génératifs ?

Les modèles génératifs sont des modèles de machine learning capables de synthétiser des objets complexes, comme des textes, des images ou de sons, « similaires » à ceux d'une liste d'exemples. En termes un peu plus techniques, on peut dire que les modèles génératifs sont capables d'apprendre une distribution de probabilité sur des objets complexes, distribution que l'on pourra ensuite échantillonner pour produire des exemplaires inédits mais ressemblants aux exemples.

Figure 1 : (a) un Deep Dream (b) un Rembrandt synthétique (c) application d'un style Van Gogh à une photo quelconque.

Les **applications** de ces modèles génératifs sont multiples :

- La **création**, interactive ou non, **d'images ou de vidéos** dans un contexte artistique ou marketing [DMG]. C'est le sujet principal de cet article.
- La simulation d'évolutions vraisemblables d'environnements physiques en vue de planifier des tâches en **robotique** ou dans un contexte d'**apprentissage par renforcement** [IGF].
- La mise à l'épreuve de notre **aptitude à comprendre** la structure d'objets complexes du monde réel, en essayant d'en générer des exemplaires crédibles.
- La génération d'**images en haute résolution réalistes** à partir d'images en basse résolution.
- L'enrichissement d'un jeu de données dans un contexte d'**apprentissage semi-supervisé** où l'on dispose de peu de données étiquetées.

Notre définition intuitive d'un modèle génératif manque hélas de rigueur puisque nous n'avons pas pris la peine de préciser le sens de l'adjectif « similaire ». Si on la prenait au pied de la lettre, on pourrait construire par exemple un algorithme qui se contenterait de régurgiter dans le désordre les images fournies en entrée et arguer (avec un zeste de mauvaise foi) que les images ainsi produites sont bien similaires aux exemples ! On conçoit par conséquent que, pour être utile, un algorithme génératif se doit d'incorporer une **notion de régularité** qui lui permettra d'interpoler entre les exemples qu'il a vu, pour en générer de nouveaux qui ne soient pas strictement identiques. La définition mathématique rigoureuse de cette notion de similarité reste à ce jour un problème ouvert pour les objets complexes du monde physique [MAL]. Par chance, les techniques de Deep Learning sont aujourd'hui en avance sur les développements théoriques. De fait, on sait construire des modèles génératifs qui font l'affaire en pratique, même si nous comprenons encore mal la nature de la régularité qu'ils exploitent implicitement.

Dès à présent nous limiterons notre discussion à la **génération d'images**.

Les auto-encodeurs (AE)

Les auto-encodeurs sont des modèles d'**apprentissage non supervisé** qui vont nous permettre de comprendre dans un premier temps une notion importante dans le cadre des modèles génératifs : la notion de **variables latentes**.

Considérons par exemple une collection d'images de visages humains d'une résolution de 1000 x 1000 pixels et posons-nous la question de savoir combien de paramètres sont nécessaires au minimum pour décrire correctement

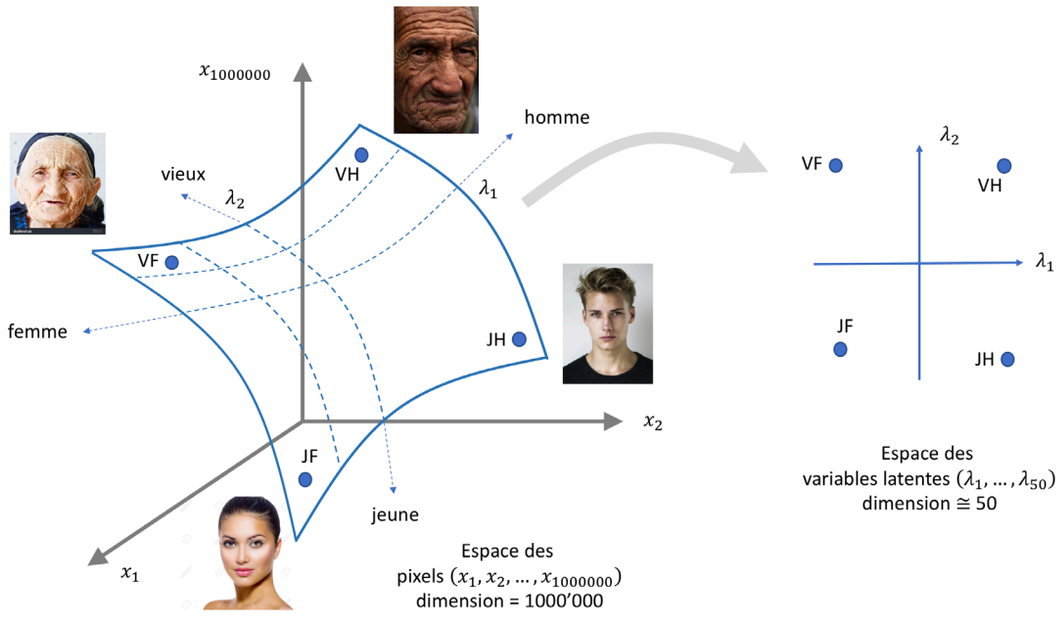


Figure 2 : L'espace des images est de dimension 1000'000 (trois dans la figure). Les images de visages constituent une infime partie de cet espace qui correspond à une surface de dimension beaucoup plus faible, peut-être 50 (deux dans la figure). Les coordonnées sur cette surface de faible dimension sont les variables latentes.

un visage en particulier. L'âge, le sexe, la coupe de cheveux, le teint, l'aspect souriant ou non, … sont probablement indispensables mais, de tout évidence, insuffisants. Peut-être quelques dizaines de paramètres suffiront-ils à condition d'accepter qu'ils n'aient pas tous une interprétation directe évidente. On appelle ces paramètres des variables latentes. La figure 2 illustre la situation : parmi toutes les images carrées d'un million de pixels, seule une infime fraction correspond effectivement à des visages humains.

Les 50 variables latentes $(\lambda_1, \lambda_2, \ldots, \lambda_{50})$ peuvent être envisagées comme une **forme compressée** de l'information visuelle. A chaque point dans l'espace des variables latentes correspond un visage, si bien que parcourir cet espace latent revient à parcourir l'ensemble des visages humains. Les auto-encodeurs sont des systèmes d'apprentissage non supervisé qui permettent :

1. De **construire un espace de variables latentes**. Les différentes variables latentes que l'algorithme découvre n'ont cependant pas vocation à être directement interprétables, contrairement à celles de la figure 2.
2. De **reconstruire une image** à partir d'un point dans l'espace latent.

Leur principe de fonctionnement est illustré dans la figure 3. Il utilise un réseau de neurones.

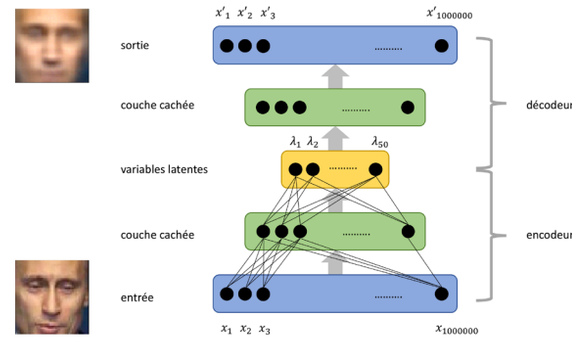


Figure 3 : L'architecture d'un auto-encodeur constitué d'un encodeur et d'un décodeur symétriques. Seules les connexions entre les 3 premières couches sont représentées, les autres étant symétriques.

Un AE est un réseau de neurones constitué de plusieurs couches connectées de manière symétrique comme l'illustre la figure 3[1]. Les couches d'entrée et de sortie sont identiques

(1) En pratique le schéma d'interconnexion pour un AE d'image utilise les éléments d'un réseau de convolution (CNN), maie le principe reste le même.

et sont celles qui possèdent le plus grand nombre de neurones. Pour une image en noir et blanc, chaque pixel correspond à un neurone. Le nombre de neurones de la couche centrale correspondant au nombre de variables latentes que l'on souhaite découvrir. Plus les nombres de couches et de neurones sont importants, plus l'auto-encodeur sera capable d'apprendre à encoder des images complexes. Ces nombres sont des hyperparamètres qu'il faudra ajuster par tâtonnements pour parvenir à une reconstruction optimale sans trop de ressources.

Durant l'**entraînement** de l'AE, on lui présente une liste d'images en entrée et on lui demande d'apprendre à les reconstruire aussi précisément que possible sur la couche de sortie en minimisant une **erreur de reconstruction** qui évalue la différence entre l'image originale et l'image reconstruite[2]. Les couches comprises entre l'entrée et la couche de variables latentes sont alors à envisager comme un **encodeur,** et les couches entre les variables latentes et la sortie comme un **décodeur**. En d'autres termes, on demande à l'encodeur de compresser l'information pour la faire tenir dans un petit nombre de variables latentes et au décodeur de reconstruire cette information du mieux qu'il peut. Sur les données qu'on lui fournit, l'AE ne fait donc globalement… rien ! Cette opération en apparence triviale ne l'est pas en réalité, car le faible nombre de neurones de la couche centrale constitue un goulet d'étranglement qui force l'AE à trouver une **représentation parcimonieuse** des données qu'on lui a présentées[3]. Il faut bien réaliser que ce mécanisme de compression ne fonctionnera correctement que pour des données de même nature que celles avec lesquelles on a entrainé l'AE. Si on présente une image de camion plutôt qu'un visage à l'AE, l'encodage n'aura aucun sens et la reconstruction sera impossible car l'AE cherchera désespérément à interpréter un camion comme un visage !

Utiliser un AE en **mode génératif** revient alors à échantillonner des vecteurs de variables latentes $(\lambda_1, \lambda_2, ..., \lambda_{50})$ puis à reconstruire les images associées au moyen du décodeur. Ce processus est créatif, si l'on veut, dans la mesure où les visages générés de cette manière seront différents de ceux du jeu d'entraînement. Il n'y a aucune raison en effet pour que les vecteurs latents choisis au hasard coïncident avec à ceux des images du jeu d'entraînement.

Les auto-encodeurs variationnels (VAE)

Hélas les auto-encodeurs simples similaires à celui qui est représenté sur la figure 3 fonctionnent mal en pratique. Ils ne permettent pas de réaliser pleinement l'idée illustrée sur la figure 2, à savoir un encodage continu de représentations, qui permettrait de se déplacer sur l'espace des visages pour faire du **morphing** sur des caractéristiques bien identifiées. La seule raison pour parler des AE simples était d'introduire la notion de variables latentes.

Les auto-encodeurs variationnels (VAE) rajoutent **deux ingrédients supplémentaires** par rapport aux AE simples qui leur permettent de réaliser la promesse de la figure 2.

1. Le point de départ est l'idée intuitive qu'une image est toujours le résultat d'un processus aléatoire, si bien que deux versions d'un même visage devraient être encodées par des points proches dans l'espace latent. Dit autrement, deux points dans l'espace latent, dont la distance est inférieure à un certain seuil, devaient être considérés comme indiscernables l'un de l'autre. Dès lors, plutôt que d'encoder chaque image de manière déterministe par un point λ dans l'espace latent, un VAE va choisir au hasard un point $\boldsymbol{\lambda}^*$ proche de ce point λ. Plus précisément, l'encodeur d'un VAE calcule une distribution de probabilité centrée sur λ et d'écart-type σ, les deux étant appris par l'encodeur, qui comporte en l'occurrence deux modules comme l'illustre la figure 4. En ce sens les VAE sont donc des **modèles stochastiques:** ils introduisent une part d'aléa dans le processus de génération. Ce processus de **floutage** délibéré incite le VAE à découvrir des encodages efficaces qui jouissent des propriétés de continuité souhaitées

(2) Comme la somme des carrées des différences $(x_i - x'_i)^2$ par exemple.

(3) Il est possible d'exploiter d'autres contraintes que le goulet d'étranglement pour forcer un AE à découvrir un espace de variables latentes. L'ajout d'un signal de bruit aux données en entrée par exemple ou des contraintes de parcimonie qui limitent le nombre de neurones actifs dans la couche centrale sont deux possibilités [DLP].

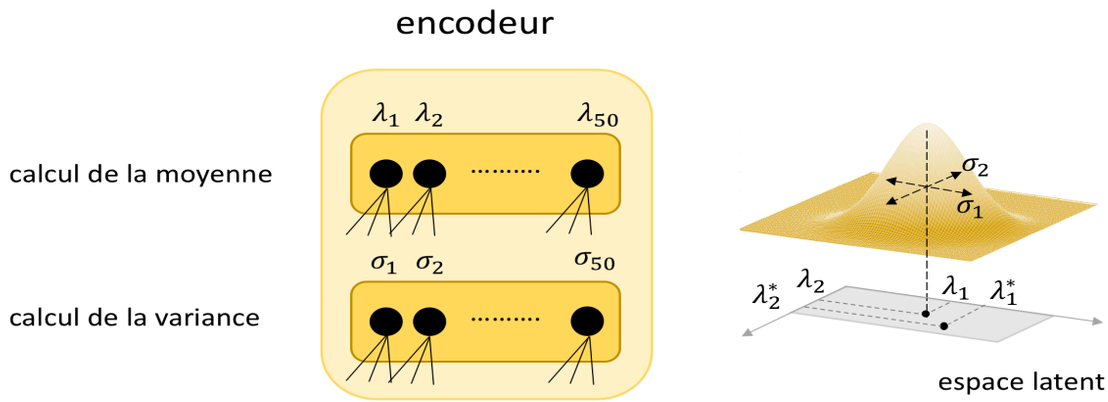


Figure 4 : Le générateur d'un VAE comporte deux parties dont l'une détermine la moyenne λ et l'autre la variance σ d'une distribution normale dont on échantillonne un point λ^* au hasard dans l'espace latent.

et remplace les conditions de parcimonie des AE simples.

2. À chaque image en entrée correspond donc un point tiré au hasard selon le processus décrit précédemment. L'ensemble des points associés à toutes les images de l'ensemble d'entraînement forment donc un nuage de points dans l'espace latent. À priori, la forme de ce nuage est arbitraire. Si toutefois nous parvenions à faire en sorte que ce nuage ait une distribution simple et bien déterminée nous pourrions alors échantillonner des points selon cette même distribution pour créer, grâce au décodeur, des images aux propriétés statistiques identiques à celles de l'ensemble d'entraînement. Pour forcer le nuage de point à avoir une forme simple (une loi normale) les VAE introduisent, en plus du coût de reconstruction, **un terme supplémentaire dans la fonction de coût** qui pénalise les nuages de points dont la distribution s'écarte trop d'une loi normale standard[4].

L'expérience a démontré qu'en plus des propriétés de continuités déjà évoquées, les VAE avaient par ailleurs des propriétés arithmétiques qui les rendent très utiles pour la **retouche d'image avancée**. Ainsi peut-on identifier une direction « sourire » ou une direction « lunettes » dans l'espace latent qui permettent de moduler l'expression d'une photo (figure 5). Ces propriétés sont similaires aux propriétés arithmétiques des **Word Embeddings** dans le domaine linguistique.

Pour trouver ces directions dans l'espace latent, rien de plus simple : pour construire un vecteur « sourire » il suffit de soustraire le vecteur représentatif d'un visage triste à celui d'un visage souriant (ou de faire une moyenne de telles différences).

Les Generative Adversarial Networks (GAN)

L'état de l'art en matière de génération d'images réalistes ne repose plus aujourd'hui sur les VAE mais sur une nouvelle classe de modèles génératifs inventée en 2014 par Ian Goodfellow et appelés **Generative Adversarial Networks** [GAN]. Les GAN sont intéressants pour leurs performances mais, plus encore peut-être, pour l'innovation conceptuelle qu'ils représentent dans le domaine du Deep Learning. Yann LeCun, directeur de recherche du labo Facebook AI Research dit à ce propos : « *Les GAN sont l'idée la plus intéressante (en IA) que j'ai vue depuis 10 ans !* » [YLC].

On a coutume de les introduire au moyen d'une analogie avec l'activité de contrefaçon d'un **faux monnayeur** et, faute d'une meilleure idée, nous ne dérogerons pas ici à cette tradition. À bien y réfléchir un faux monnayeur et un modèle génératif ont effectivement des objectifs similaires : le premier cherche à duper l'autorité en charge d'émettre la monnaie officielle en

(4) Pour les geeks : c'est la divergence de Kullback-Leibler entre la distribution empirique des points représentatifs dans l'espace latent et la loi normale standard en *d* dimension où *d*=dim(espace latent).

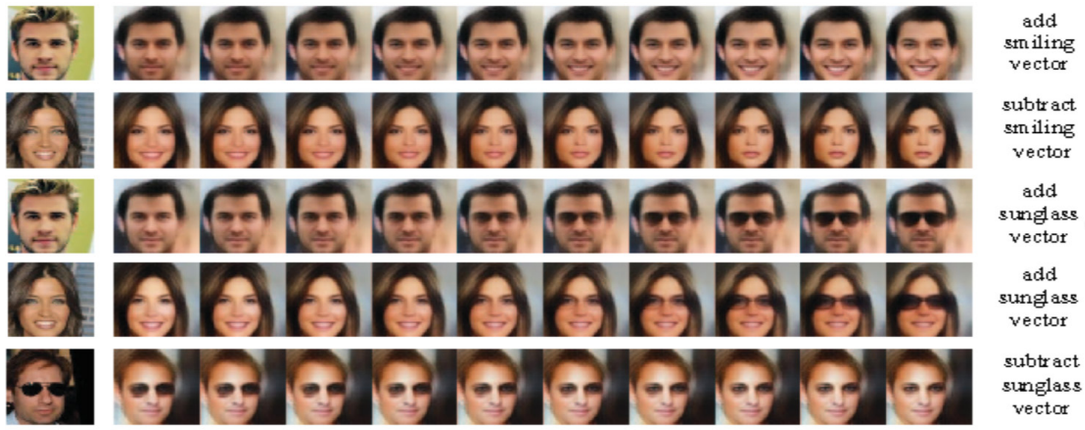


Figure 5 : Application du vecteur « sourire » et du vecteur « lunette » à plusieurs exemples.

confectionnant de faux billets réalistes, alors que le second doit produire des images crédibles capables de leurrer des humains.

Un système GAN est constitué de **deux réseaux de neurones en compétition** comme l'illustre la figure 6.

Le premier, appelé le **générateur**, a pour objectif de créer des images fictives crédibles qui ressemblent à celles d'un ensemble d'entraînement. Ce générateur fonctionne sur le même principe que le décodeur d'un AE ou d'un VAE représenté sur la figure 3 : il apprend à convertir du bruit (un point tiré au hasard dans l'espace latent) en une image. Mais, alors qu'un AE ou un VAE apprenait son « métier » en minimisant une fonction de coût statique, les GAN exploitent un mécanisme d'évaluation dynamique beaucoup plus puissant. Ils délèguent le contrôle qualité à un second réseau de neurones, appelé le **discriminateur**, qui va progressivement apprendre à distinguer les vraies images des fausses sur la base d'exemples connus.

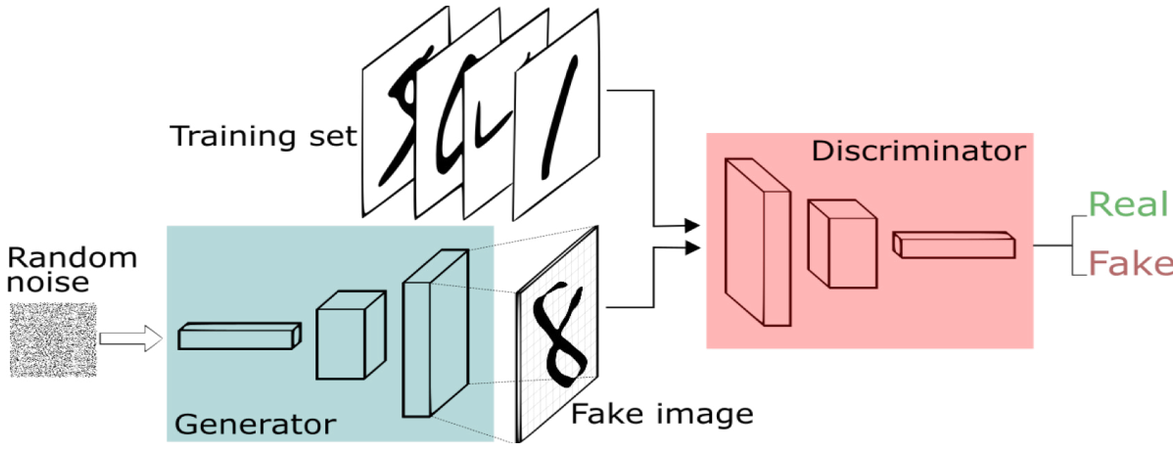


Figure 6 : Un système GAN composé d'un générateur chargé de créer des images réalistes de digits manuscrits par transformation d'un bruit aléatoire et d'un discriminateur chargé de distinguer les vraies images des images fictives. Le discriminateur est basé sur un réseau de convolution (CNN). Le générateur utilise une forme de CNN inversé.

Ce discriminateur est entraîné sur un mode supervisé de la manière suivante :

1. On échantillonne quelques dizaines d'images réelles de l'ensemble d'entraînement auxquelles on attribue l'étiquette « vraie ».

2. On échantillonne ensuite quelques dizaines d'images fictives confectionnées par le générateur à partir de points tirés au hasard dans l'espace et auxquelles on attribue l'étiquette « fausse ».

3. On fusionne les deux échantillons et l'on entraîne le discriminateur à prédire correctement les étiquettes, « vraie » ou « fausse » de chaque image de cet ensemble. Le discriminateur apprend en fait une probabilité pour une image d'être vraie.

On fige ensuite le discriminateur dans cet état, puis on entraîne le générateur à tromper le discriminateur aussi souvent que possible de la manière suivante :

4. On fabrique de nouvelles images fictives grâce au générateur.

5. On soumet ces nouvelles images au discriminateur.

6. On optimise le générateur pour que le discriminateur attribue (à tort) l'étiquette « vraie » aux images fictives aussi souvent que possible.

On itère ce processus. À chaque étape le générateur produira, si tout se passe bien, des images un peu plus réalistes, ce qui contribuera à renforcer le discernement du discriminateur, ce qui obligera le générateur à se surpasser à l'étape suivante et ainsi de suite. L'objectif de cette compétition, ou de ce **jeu** si l'on préfère, est de parvenir à un **équilibre** où le générateur parvient à reproduire des images indiscernables des originaux, tandis que le discriminateur attribue des étiquettes correctes une fois sur deux. La figure 7 montre 20 chambres à coucher synthétisées par un GAN.

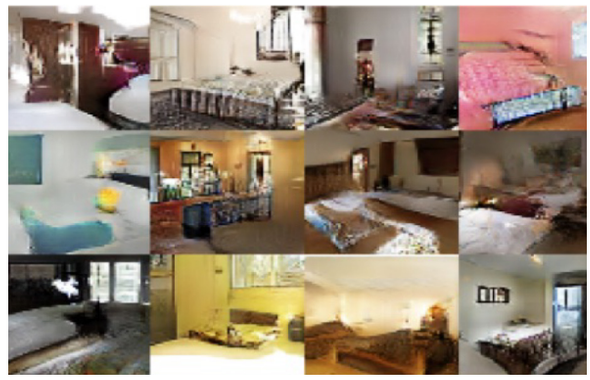


Figure 7 : Des chambres à coucher fictives imaginées par un GAN. À vous de juger !

La recherche d'un équilibre stable[5] dans un jeu est un problème beaucoup plus délicat que l'optimisation d'une fonction de coût, si bien que la description précédente n'est qu'une esquisse conceptuelle qui doit en réalité s'accompagner de nombreux trucs et astuces, d'expérimentations et de patience pour donner des résultats probants. Cette partie du Deep Learning reste à ce stade plus proche de la **magie noire** que de la science fondée sur des justifications robustes. Une partie de la difficulté de l'entraînement des GAN tient à l'inadaptation des outils existants du Deep Learning[6], conçus dès l'origine pour rechercher des minimas et non pas des situations d'équilibre.

En examinant les images de la figure 7 on peut être pris de doute et penser que le GAN s'est simplement contenté de copier des images de chambre à coucher existantes. Il faut cependant garder à l'esprit que le générateur (voir la figure 6) ne « voit » jamais une image directement. C'est indirectement, par le biais des retours que lui fournit le discriminateur, qu'il apprend à construire ces simulacres. En ce sens, l'entraînement d'un GAN s'apparente à une forme d'**apprentissage par renforcement** où le discriminant fournirait une récompense au générateur mais où cette récompense deviendrait de plus en plus difficile à obtenir.

Bien qu'ils produisent des images très nettes, pour des raisons d'ailleurs encore mal comprises, les GAN n'ont pas un espace latent aussi bien structuré et continu que les VAE.

(5) Techniquement il s'agit d'un équilibre de Nash entre deux joueurs qui cherchent chacun à optimiser leur stratégie, ayant connaissance des règles du jeu qui s'appliquent aux deux.

(6) Comme la rétropropagation et descente de gradient stochastique.

Figure 8 : une esquisse et l'image photo-réaliste générée par Pix2Pix qui utilise un cGAN

Notons enfin qu'il existe d'innombrables variantes des GAN pour différents usages. Ainsi les cGAN, ou Conditional GAN, sont-ils capables de générer des images de manière conditionnée. Plutôt que des générer des images aux hasard comme le font les GAN, ils gérèrent des images différentes, selon une information qu'on leur fournit en entrée. Le système Pix2Pix, capable de traduire une esquisse en une image photo-réaliste, utilise un tel modèle cGAN, voir la figure 8 et [PIX].

3. Pas encore de e-Picasso mais...

Nous voilà bien équipés pour répondre à la question initiale : qu'en est-il de la créativité de systèmes génératifs comme les VAE ou les GAN ? Bien qu'ils fonctionnent sur des mécanismes différents, ces systèmes ont en commun l'objectif de base qui consiste, dans un premier temps, à apprendre une distribution de probabilité sur des objets complexes, comme des images, puis, dans un second temps, à générer des objets similaires par **échantillonnage** de la distribution apprise. Voilà donc à quoi se résume toute leur « créativité ». Faut-il préciser dans ces conditions qu'un e-Picasso n'est pas pour demain ? [FPI].

Pour aller un peu plus loin dans l'analyse, dressons un parallèle entre les limitations des modèles d'apprentissage supervisé et celles des modèles génératifs. Les capacités prédictives des modèles actuels reposent, on le sait, sur la possibilité de faire des prédictions par interpolation de la variable cible à partir d'exemples connus, à condition de postuler une **régularité** suffisante dans les données [MAL, LEM]. En ce sens, les prédictions de ces modèles ne sont donc que des **généralisations locales** à partir d'observations similaires à celle pour laquelle une prédiction est souhaitée. On conçoit bien dès lors que les modèles prédictifs actuels sont incapables d'abstraire un apprentissage dans un domaine pour en tirer parti dans un autre domaine éloigné ou encore de planifier une stratégie à long terme dans le cadre d'un apprentissage par renforcement. Ces aptitudes seraient des formes de **généralisation à grande échelle** [DLP] aujourd'hui inaccessibles à l'IA. La situation est analogue pour les modèles génératifs. Eux aussi présupposent une forme de régularité dans la distribution des exemples, dans le sens où un nombre limité d'entre-eux doit permettre d'imaginer de nouveaux exemplaires similaires.

La capacité d'imagination créative des modèles génératifs actuels est aussi limitée que le sont les capacités de généralisation ou d'abstraction des modèles prédictifs et ceci pour les mêmes raisons.

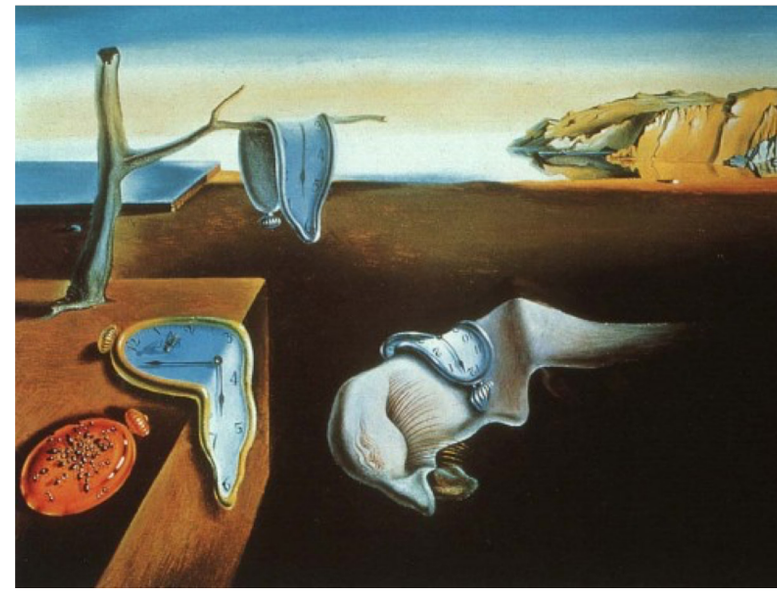


Figure 9 : Les « montres molles » de Salvador Dali.

Sans vouloir nous lancer dans d'aventureuses spéculations sur la nature profonde de la création artistique, essayons malgré tout de la comparer aux balbutiements des modèles génératifs décrits précédemment. Au moins deux caractéristiques distinguent ici l'homme de la machine. La première est que la créativité artistique prend ses racines dans une **expérience multi-sensorielle,** qui s'échelonne sur la vie d'un artiste et va donc bien au-delà d'un simple échantillonnage d'une seule catégorie de données (images, textures, sons, textes etc…). Le seconde est qu'un artiste est habité d'une **intention**, celle de communiquer un monde intérieur ou d'enrichir celui des destinataires de son œuvre, avec qui il partage une communauté d'expériences qui font une vie humaine. Lorsque Salvador Dali peint « *La persistance de la mémoire* » illustrée dans la figure 9, il crée une œuvre qui présuppose que l'observateur sait, non seulement ce qu'est une montre, mais ce qu'est la gravitation et que les montres réelles ne sont pas molles et que de cette incongruité naîtra une allégorie qui tourne en dérision notre obsession du contrôle du temps.

Les affirmations comme quoi le prochain Rembrandt pourrait être un robot ne sont donc que des fantasmes (pour rester modéré) de journalistes en quête de sensationnel.

> *Les machines apprennent des lois statistiques mais n'ont pas accès, pour l'instant, à la sémantique des objets qu'elles manipulent. Si elles savent produire des images réalistes de chambres à coucher elles n'ont en revanche pas la moindre idée des usages possibles d'un lit ou d'un miroir.*

Si l'on veut examiner sérieusement l'apport de modèles génératifs dans un contexte artistique ou créatif, il faut vraisemblablement abandonner le paradigme d'une IA conçue en tant qu'externalisation d'un processus cognitif et envisager plutôt celui d'une **Intelligence Augmentée**. Dans ce cadre, il s'agit souvent pour une « IA » de dispenser le créateur d'un ensemble de tâches considérées comme fastidieuses ou peu créatives justement : coloriser une image, générer des détails ou encore écrire un accompagnement pour une mélodie. Libéré de ces contingences subalternes, celui-ci pourrait alors donner la pleine mesure de son talent créateur ! Est-il bien réaliste toutefois de vouloir dispenser un artiste de toute forme de technicité, qu'elle soit laborieuse à acquérir ou fastidieuse à exécuter ? Rien n'est moins sûr.

Références

- [ROB] **Les robots seront-ils les artistes de demain ?**
D. Bourcier, P. de Filippi, La Tribune – 2 mars 2018
www.latribune.fr/opinions/tribunes/les-robots-seront-ils-les-artistes-de-demain-770046.html

- [FPI] **Et si le futur Picasso était un robot ?**
E. Paquette, L'express – février 2018
lexpansion.lexpress.fr/high-tech/et-si-le-futur-picasso-etait-un-robot_1986771.html

- [PIX] **Image-to-Image Translation in Tensorflow,**
C. Hesse
affinelayer.com/pix2pix/

- [YLC] **An introduction to Generative Adversarial Networks,**
J. Glover – août 2016
blog.aylien.com/introduction-generative-adversarial-networks-code-tensorflow/

- [GAN] **Generative Adversarial Networks,**
I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio [stat.ML] – juin 2014
arXiv:1406.2661v1

- [MAL] **Science des données – cours au Collège de France**,
Stéphane Mallat – janvier 2018.
www.college-de-france.fr/site/stephane-mallat/inaugural-lecture-2018-01-11-18h00.htm

- [DMG] **Dual Motion GAN for Future-Flow Embedded Video Prediction,** *X. Liang, L. Lee, W. Dai, E. P. Xing,* [cs.CV] – août 2017
arXiv:1708.00284v2

- [LEM] **L'IA au-delà des clichés, section II.2**,
P. Lemberger, N. Parent, A. Régent – février 2018
ia2.weave.eu/

- [DLP] **Deep Learning with Python**,
F. Chollet, Manning Publications – novembre 2017

- [TAE] **Tutorial on Variational Autoencoders,**
C. Doersch, [stat.ML] – août 2016.
arXiv:1606.05908v2

- [IGF] **Generative Adversarial Networks**,
I. Goodfellow, vidéo NIPS 2016 – avril 2017
www.youtube.com/watch?v=AJVyzd0rqdc


Inférence
TensorFlow
Intervalle de confiance
Approximation de Laplace
George Edward Box
Réseaux de neurones bayésiens
Adversarial examples
Dropout
Approximation MAP
Démocratisation
Cycle de Box
Machine learning bayésien
Entropie prédictive

Comment construire des IA qui savent douter ?

Pirmin Lemberger – **weave Business Technology**

Résumé

En dépit des progrès récents du Machine Learning (ML) ses applications à des domaines critiques (industrie nucléaire, médecine, algorithmes de trading) restent encore rares. En cause, l'incapacité des modèles de ML actuels à représenter de manière fiable la notion d'incertitude. En un mot : ils ne savent pas douter raisonnablement de leurs prédictions ! Raisonner rationnellement en présence d'incertitudes voilà pourtant ce que permet l'approche bayésienne du ML, une idée déjà ancienne. Combinée à la flexibilité des techniques récentes du Deep Learning, cette approche ouvre un nouveau champ de possibilités. Dans l'immédiat elle offre une panoplie d'outils aux chercheurs en IA pour expérimenter une grande diversité de modèles prédictifs. Cet article peut se lire comme une introduction à ce sujet. Il présente les concepts, quelques applications ainsi que la librairie *Edward* intégrée à *TensorFlow* et développée par *Google* qui concrétise cette approche.

Note : cet article présuppose quelques rudiments de théorie des probabilités et de programmation en Python.

1. Les limitations du ML dans sa formulation usuelle

Quand un chirurgien, la veille d'une opération délicate, prend une décision basée sur l'interprétation qu'il fait d'un cliché radiologique, il engage sa responsabilité professionnelle. Pour assumer ce risque il doit être en mesure de l'évaluer rationnellement. En d'autres termes, il doit mesurer sa propre incertitude quant à l'analyse qu'il fait d'un contexte clinique pour jauger de la pertinence d'une stratégie face à différentes alternatives. De même un trader qui engage des sommes importantes doit évaluer le risque qu'il fait prendre à l'institution pour laquelle il opère. Enfin, un conducteur qui engage son véhicule dans une manœuvre de dépassement doit évaluer le risque de mal interpréter sa perception du paysage routier.

Chacune de ces trois décisions pourrait un jour être confiée à une IA, à condition toutefois que celle-ci sache évaluer **les incertitudes** de la situation à laquelle elle est confrontée pour être en mesure, au-delà d'un seuil de doute raisonnable, de **déléguer** l'analyse à un autre système ou de laisser la décision finale à un humain.

Le problème désormais célèbre des **« Adversarial Examples »** illustre jusqu'à la caricature ces limitations du ML. Comme nous l'avions expliqué dans un article précédent [CSE], rappelons qu'il est possible pour un individu mal intentionné d'induire en erreur un système de classification d'image en superposant aux images originales des perturbations malicieuses mais invisibles à l'œil humains. La figure 1 illustre ce phénomène.

Hormis l'absence d'une prise en compte fiable de l'incertitude, les modèles de Deep Learning souffrent aussi d'autres limitations bien connues des data scientists :

- Ils exigent pour leur entraînement de **grandes quantités de données** (réelles ou créées artificiellement).

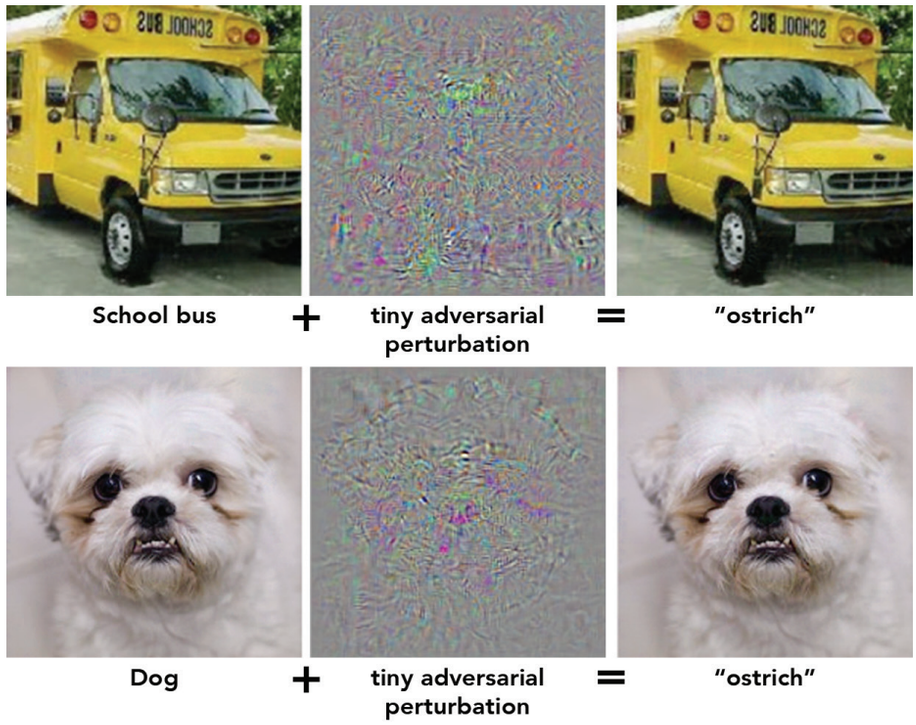


Figure 1 : Un exemple d'« Adversarial Example » où deux images très différentes et légèrement perturbées, sont toutes deux classées avec une grande probabilité comme étant des images d'autruches ! source : [Learning Securely, E. Klarreich]

- Les modèles de DL ont un **fonctionnement en boîte noire** souvent difficile à interpréter.

L'approche bayésienne du ML décrite dans le paragraphe qui suit est une tentative pour apporter des réponses à ces limitations. Nous nous focaliserons ici sur la question de l'évaluation rationnelle l'incertitude.

2. Le ML bayésien, des idées anciennes...

Depuis Aristote, la **logique** est le fondement de tout raisonnement rationnel qui s'appuie sur une information à la fois complète et certaine. Pour raisonner rationnellement en présence de données incomplètes ou incertaines, il faut en revanche recourir à une extension de la logique : la **théorie des probabilités**. Lorsque l'on interprète la notion de probabilité en termes d'incertitude, c'est plus précisément l'**interprétation dite bayésienne** de la théorie qui est utilisée. C'est sur elle que se base le Machine Learning ou le Deep Learning bayésien.

Quelques éléments de vocabulaire pour commencer.

Imaginons que pour expliquer un certain phénomène aléatoire nous formulions une hypothèse dont nous estimons les **chances à priori** d'être vraie à $P(\text{hypotèse})$, un nombre compris entre 0 et 1. La probabilité $P(\text{data} \mid \text{hypotèse})$ d'observer les données *data* si cette hypothèse est vraie est ce qu'on appelle la **vraisemblance** de l'observation *data*. Notons enfin $P(\text{hypotèse} \mid \text{data})$ la **probabilité à postériori** que l'hypothèse soit vraie si l'on a observé les données data. La célèbre formule de Bayes permet de déduire la probabilité à postériori d'une hypothèse incertaine à partir de sa probabilité à priori et de la vraisemblance des données observées :

$$P(\text{hypotèse} \mid \text{data}) = \underbrace{P(\text{data} \mid \text{hypotèse})}_{\text{vraisemblance des données}} \underbrace{P(\text{hypotèse})}_{\text{probabilité d'une hypothèse}} / \underbrace{\text{constante}}_{\text{normalisation}}$$

Équation (1) : la formule de Bayes indique comment mettre à jour la probabilité d'une hypothèse incertaine étant données la vraisemblance d'une observation et la probabilité à priori assignée à cette hypothèse.

Cette probabilité à postériori pourra à son tour jouer le rôle de probabilité à priori dans le cadre d'une nouvelle observation et ainsi de suite. Ce **caractère composable** des déductions est l'un des principaux atouts de l'approche bayésienne. On préfère souvent le terme **inférence** à celui de déduction.

Revenons au Machine Learning et envisageons un problème de classification binaire. Un tel modèle cherche à prédire la probabilité qu'une certaine propriété (un client rembourse son crédit) soit vraie $(y = 1)$ ou fausse $(y = 0)$ en fonction de certaines caractéristiques $\mathbf{x}$ d'une observation (la situation financière du client). Peu importe ici que le modèle soit élémentaire (régression logistique) ou plus évolué (réseau de neurones), il définit toujours la vraisemblance $P(y \mid \mathbf{x}, \mathbf{w})$ qui est la probabilité que *y* soit vraie ou fausse en fonction des caractéristiques $\mathbf{x}$ observées et en fonction des valeurs de certains paramètres internes $\mathbf{w}$ que l'on peut concevoir en l'occurrence comme l'hypothèse incertaine ou le modèle incertain que l'on postule. On utilise souvent le terme de **variables latentes** pour ces paramètres $\mathbf{w}$.

Avant même d'observer quoi que ce soit, on résume l'incertitude que l'on a des paramètres $\mathbf{w}$ par des probabilités à priori $P(\mathbf{w})$. Une fois un échantillon de données $\text{data} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\}$ observé, on utilise la formule de Bayes pour mettre à jour la distribution sur les $\mathbf{w}$ et calculer la distribution à postériori $P(\mathbf{w} \mid \text{data})$ que l'on vient de calculer.

Dans l'approche bayésienne du ML, l'apprentissage consiste donc à mettre à jour de la distribution de probabilités sur les modèles $\mathbf{w}$ après l'observation des données d'entraînement : $\text{data} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\}$

Pour faire une prédiction pour une nouvelle observation $\mathbf{x}_{\text{NEW}}$ on calcul une moyenne des prédictions de chaque modèle individuel $P(y_{\text{NEW}} \mid \mathbf{x}_{\text{NEW}}, \mathbf{w})$ pondéré par sa probabilité à postériori $P(\mathbf{w} \mid \text{data})$:

$$P(y_{\text{NEW}} \mid \mathbf{x}_{\text{NEW}}) = \sum_{\mathbf{w}} P(y_{\text{NEW}} \mid \mathbf{x}_{\text{NEW}}, \mathbf{w}) P(\mathbf{w} \mid \text{data})$$

Équation (2) : Calcul de la distribution de probabilité prédictive comme une pondération des prédictions des modèles individuels par leur probabilité à postériori. Il s'agit d'une forme particulière d' « ensembling».

Voilà pour les principes.

Les choses se gâtent hélas lorsque l'on passe à la pratique. L'approche bayésienne se heurte en effet à deux difficultés principales : d'une part le calcul de la constante de normalisation dans l'équation (1) et d'autre part le calcul de la moyenne pondérée dans l'équation (2) sont le plus souvent impossibles à mener exactement. Pour éviter ce **coût de calcul prohibitif**, on a dès lors recours à différents schémas d'approximations. Sans entrer ici dans les détails, il en existe deux grandes catégories pour le calcul de la distribution à postériori $P(\mathbf{w} \mid \text{data})$:

- Les **méthodes variationnelles** : on renonce au calcul exact de cette distribution pour calculer analytiquement une distribution approximative $P_\theta(\mathbf{w} \mid \text{data})$. On la choisit parmi un ensemble restreint de distributions simples et dépendantes d'un petit nombre de paramètres θ que l'on va ajuster pour rapprocher[1] $P_\theta(\mathbf{w} \mid \text{data})$ de la vraie distribution $P(\mathbf{w} \mid \text{data})$, voir [BIS – chapitre 10].

- Les **techniques d'échantillonnage** : il s'agit de techniques numériques qui permettent d'échantillonner efficacement des distributions de probabilités complexes. La technique de base est la technique dite **MCMC** pour Monte Carlo Markov Chain [BIS – chapitre 11].

Enfin, il existe différentes approximations classiques pour le calcul de la distribution prédictive $P(y_{\text{NEW}} \mid \mathbf{x}_{\text{NEW}})$:

- L'**approximation MAP** (maximum à postériori) est radicale puisqu'elle consiste à remplacer la somme sur $\mathbf{w}$ dans l'équation (2) par un seul terme $\mathbf{w} = \mathbf{w}_{\text{MAP}}$, celui qui maximise la distribution à postériori $P(\mathbf{w} \mid \text{data})$.

(1). La divergence de Kulback-Leibler est la mesure classique adéquate d'écart entre deux distributions de probabilité.

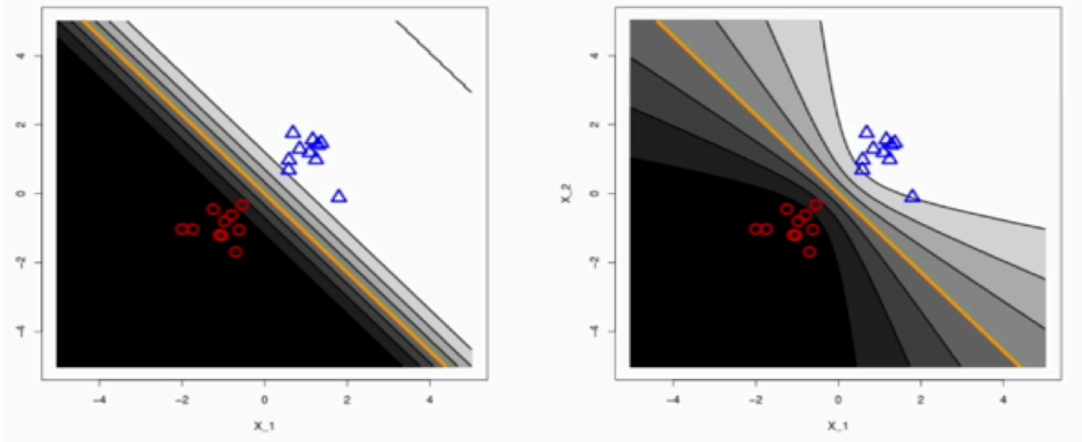


Figure 2 : À gauche l'application du maximum de vraisemblance à un problème de classification au moyen d'un modèle linéaire. Les zones noires (prédiction = rouge) et blanches (prédiction = bleu) correspondent aux régions où le modèle est très confiant dans ses prédictions. A l'évidence cette confiance est excessive dans les zones où aucune donnée n'a été observée. À droite le modèle bayésien qui résulte d'une pondération correcte selon l'équation (2) de nombreux modèles linéaires compatibles avec les données. Ses zones de confiances, noires ou blanches, sont considérablement réduites. Dans un sens ce modèle est plus prudent, il doute.

- L'**approximation de Laplace** est un peu moins radicale, elle consiste à approximer une distribution par une gaussienne autour du mode principal [BIS – chapitre 4].

- L'entraînement d'un modèle de ML par un calcul du **maximum de vraisemblance**, bien connue des data scientists, n'est qu'un cas particulier de l'approximation MAP lorsque l'on suppose que la distribution à priori $P(\mathbf{w})$ ne contient aucune information. Dans ce cas, schématiquement, $P(\mathbf{w})$ se réduit à une constante et la probabilité à postériori $P(\mathbf{w} \mid \text{data})$ est simplement proportionnelle à la vraisemblance $P(\mathbf{w} \mid \text{data})$, si bien que maximiser la première revient aussi à maximiser la seconde. Notons $\mathbf{w}_{\text{ML}}$ ce maximum.

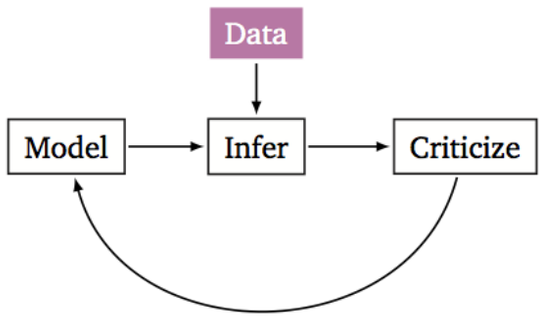


Figure 3 : Le cycle de Box qui résume la réalisation itérative d'un modèle de prédiction probabiliste.

Ce résultat $\mathbf{w}_{\text{ML}}$ présente souvent le défaut du **surapprentissage**, ce qui signifie que le modèle $P(y_{\text{NEW}} \mid \mathbf{x}_{\text{NEW}}, \mathbf{w}_{\text{ML}})$ est exagérément confiant dans les prédictions qu'il fait à propos de données éloignées des données d'entraînement *data*. Intuitivement, ce défaut tient au fait que cette procédure de calcul ne retient qu'un seul modèle, $\mathbf{w}_{\text{ML}}$, celui qui maximise la vraisemblance $P(\text{data} \mid \mathbf{w})$, alors que beaucoup d'autres modèles $\mathbf{w}$ pourraient aussi faire l'affaire, même si leur vraisemblance est moindre.

Pour pallier le surapprentissage les data scientists utilisent différentes techniques ad-hoc de régularisation. Le ML bayésien est plus rigoureux puisqu'il se propose de combiner ces différents modèles prédictifs selon les règles de la théorie des probabilités, si bien que ses prédictions seront plus fiables.

La figure 2 illustre intuitivement la différence entre l'approche usuelle et l'approche bayésienne pour un problème de classification binaire.

On peut résumer l'élaboration d'un modèle prédictif selon l'approche bayésienne en utilisant le **cycle de Box** illustré sur la figure 3 : (1) construction d'un modèle prédictif sous forme d'une fonction de vraisemblance $P(\text{data} \mid \mathbf{w})$, (2) inférence de la distribution à postériori $P(\mathbf{w} \mid \text{data})$ après observation des données (3) évaluation des prédictions.

3. ...désormais combinées au Deep Learning

Expérimenter avec de nombreux modèles

La conception d'un modèle probabiliste et, plus encore, le développement d'une méthode d'inférence appropriée demande aujourd'hui une grande expertise pour soupeser les atouts et les inconvénients des nombreuses méthodes disponibles, qu'elles soient numériques ou analytiques. Les occasions de commettre des erreurs sont donc nombreuses et la lenteur du processus rend difficile l'exploration de toutes les options.

> *L'objectif de la programmation probabiliste (PP) est double. D'une part elle doit permettre de construire par programmation des modèles probabilistes complexes que l'on pourra échantillonner. D'autre part, et de manière indépendante, elle vise idéalement à encapsuler la complexité du processus d'inférence dans un moteur de calcul universel.*

Dis autrement, un système capable d'exécuter un langage de PP comprend d'une part un **simulateur** de phénomènes aléatoires et d'autre part un **mécanisme d'interrogation** qui permet d'en explorer les conséquences.

Aujourd'hui, une part significative des publications et des présentations dans les grandes conférences sur l'IA et le ML (NIPS, ICML, ...) concerne l'étude de nouveaux modèles probabilistes et des méthodes d'inférence adéquates. L'idéal d'agilité que formule la PP, ainsi que le découplage qu'elle promeut entre la conception d'un modèle et le choix d'une technique d'inférence, permettrait d'accélérer considérablement ce genre d'étude. Le conditionnel est ici de rigueur car, et nous y reviendrons, la franchise oblige de le dire, nous n'y sommes pas encore.

Des langages de PP existent depuis belle lurette : WebPPL, Venture, Stan en sont des exemples. Ces systèmes cherchent tous à réaliser un **compromis** acceptable entre d'une part leur **expressivité**, la richesse des modèles qu'ils permettent de formuler, et leur **scalabilité**.

Ces systèmes ne sont hélas plus adaptés aux infrastructures logicielles (TensorFlow) et matérielles (GPU, TPU) conçues pour entraîner des modèles de Deep Learning comportant des millions de paramètres.

De nouveaux systèmes sont en cours d'élaboration avec l'objectif de concilier la flexibilité des modèles de Deep Learning avec les atouts d'un langage de PP universel que sont l'inférence automatique et un traitement rigoureux de l'incertitude.

Nous décrivons ci-dessous succinctement la librairie **Edward** basée sur **TensorFlow** qui est une des plus récentes.

Edward une surcouche de PP pour TensorFlow

Entraîner un modèle de ML classique revient toujours à minimiser une **fonction de coût**. L'opposé du log de la vraisemblance (NLL) mentionnée précédemment en constitue l'exemple le plus courant. Des dizaines d'astuces numériques et de librairies ont été développées durant ces vingt dernières années pour calculer efficacement le minimum de ces fonctions de coût définie par les réseaux de neurones (RN) profonds.

Une librairie comme TensorFlow (TF) développée par Google Brain apporte des services de base pour ce genre de calcul :

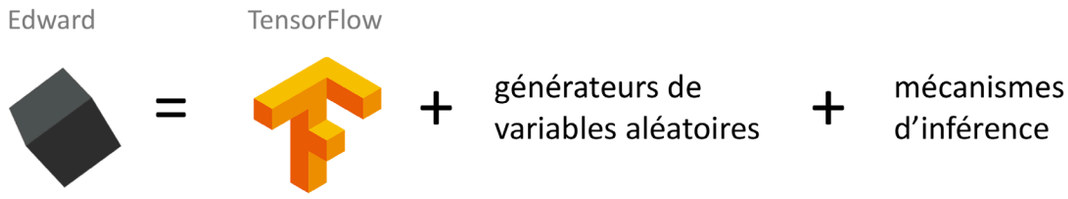


Figure 4 : Edward est une surcouche de TensorFlow.

- TF effectue le **calcul symbolique des dérivées** par rapport aux paramètres $\mathbf{w}$ des fonctions de coût définies par un RN. Ce calcul utilise une version de l'algorithme de **rétropropagation**[2]. Différents **optimiseurs** exploitent ces gradients pour trouver rapidement un minimum approximatif.

- Pour des raisons de performances d'exécution, TF compile en langage C un **graphe de calcul** que le développeur construit à l'aide d'un langage de haut niveau comme Python. L'exécution n'intervient qu'une fois le graphe construit. L'un des avantages de cette stratégie est qu'elle évite les délais occasionnés par les transferts de données répétés entre les couches C et Python qu'exigeraient les invocations successives de fonctions Python vectorisées (comme celles de la librairie `NumPy` par exemple). L'inconvénient est que la construction du graphe conduit souvent à une syntaxe peu intuitive.

- La **compilation** du code est optimisée pour cibler différentes plateformes, distribuées ou non, et notamment pour tirer parti de l'infrastructure de calcul massivement parallèle (CUDA), qu'offre les **GPU** développés par NVIDIA.

La librairie **Edward**, ainsi nommée en l'honneur du statisticien George Edward Box, tire parti de l'infrastructure TF pour créer un système de PP à la fois expressif et scalable. Edward ajoute deux fonctionnalités à TF : (1) une librairie de générateurs de variables aléatoires (VA) pour construire de modèles probabilistes complexes et (2) une librairie d'algorithmes d'inférence pour calculer des distributions à postériori.

Il n'est pas question d'entrer dans le détail de la syntaxe d'Edward, aussi nous nous contenterons de donner deux exemples de construction de modèle et d'inférence dont le code est assez simple pour être intelligible sans commentaires.

Exemple – le jeu de pile ou face biaisé

L'exemple le plus simple est peut-être celui d'une simulation de plusieurs lancer de **pile ou face** avec un pièce potentiellement biaisée dont on ignore la probabilité θ de tomber sur pile. On souhaite apprendre θ, plus précisément calculer sa distribution à postériori, suite à l'observation des résultats de plusieurs lancés. Dans l'exemple ci-dessous on suppose à priori que la pièce est probablement équilibrée[3].

Voici le code qui définit le modèle :

```
from edward.models import Bernoulli, Beta
theta = Beta(1.0, 1.0)
x = Bernoulli(probs = tf.ones(50) * theta)
```

Code (1) : définition du modèle probabiliste pour 50 tirages de pile ou face avec un paramètre de biais θ inconnu.

(2) Pour TensorFlow c'est l'algorithme Reverse-Mode Autodiff qui est implémenté [HOM, appendice D]

(3) La fonction beta (θ) choisie pour la distribution à priori à l'avantage de conduire à des calculs analytiques simples à cause de la relation de dualité avec la distribution de Bernoulli [BIS, section 2.1]. La moyenne de θ vaut ici ½, on estime donc que la pièce est à priori équilibrée.

La première ligne déclare les deux générateurs de VA utilisées. La seconde déclare la distribution à priori sur le paramètre θ , approximativement équilibrée, dont on souhaite apprendre la distribution après observation des lancés. La troisième enfin définit 50 VA binaires `x` indépendantes partageant le même biais θ. Une fois définies, on peut échantillonner ces VA avec une méthode comme `x.sample(100)`, calculer la densité de probabilité avec `x.log_density()` ou encore sa moyenne `x.mean()`.

Pour le calcul d'inférence de la distribution à postériori sur θ il faut choisir une méthode d'inférence appropriée. En l'occurrence, vu la simplicité du problème, une **inférence exacte** est possible.

Voici le code pour ce calcul d'inférence exacte :

```
import edward as ed
theta_cond =
        ed.complete_conditional(theta)
data = np.array([0,1,0,0,0,0,0,0,0,1])
sess.run(theta_cond, {x: data})
```

Code (2) : le calcul d'inférence de la distribution à postériori sur θ.

La première ligne définit une variable aléatoire `theta_cond` qui sera distribuée comme l'est θ une fois que les données `data` ont été observées. La fonction `ed.complete_conditional()` spécifie le choix d'une méthode d'inférence, en l'occurrence un calcul symbolique exact. La seconde définit les données observées. Le troisième enfin effectue le calcul d'inférence. Désormais on pourra échantillonner la VA `theta_cond` et l'utiliser pour faire des prédictions.

Exemple – un réseau de neurones bayésien

Nous illustrons la méthode d'**inférence variationnelle** sur une exemple académique de **RN bayésien** (RNB) qui n'est rien d'autre qu'un RN dont les poids et les biais sont considérés comme des variables latentes $\mathbf{w}$ dont on va calculer la distribution à postériori $P(\mathbf{w} \mid \text{data})$.

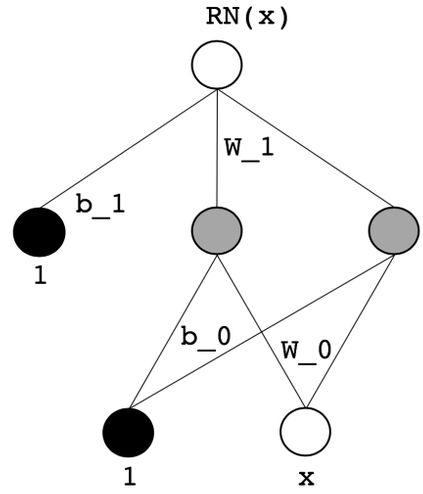


Figure 5 : le RN à une couche utilisé dans l'exemple. Les poids $\mathbf{w}_i$ et les biais b_i, $i = 0$ ou 1 sont collectivement les variables latentes $\mathbf{w}$ du modèle.

Voici le code qui définit des VA normales centrées réduite `W_i` et `b_i` correspondant à la distribution à priori des poids et des biais du RN illustré sur la figure 5 :

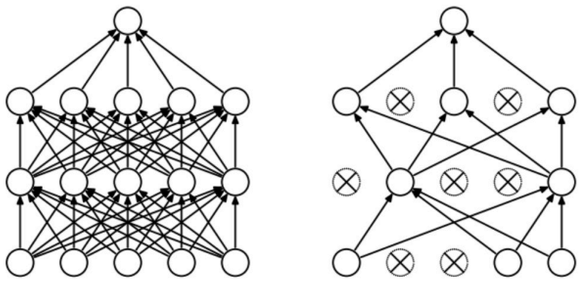
```
import tensorflow as tf
from edward.models import Normal

W_0 = Normal(loc=tf.zeros([1, 2]),
     scale=tf.ones([1, 2]))
W_1 = Normal(loc=tf.zeros([2, 1]),
     scale=tf.ones([2, 1]))
b_0 = Normal(loc=tf.zeros(2),
     scale=tf.ones(2))
b_1 = Normal(loc=tf.zeros(1),
     scale=tf.ones(1))
```

Code (3) : Les distributions à priori sur les poids et les biais du RN représenté sur la figure 5 sont supposées normales et indépendantes.

Étant donnés les poids et les biais, une fonction d'activation `tanh()` sur la couche cachée et une couche de sortie linéaire avec un bruit gaussien, le RN est définit par :

```
def RN(x):
    return tf.matmul(tf.tanh(
        tf.matmul(x, W_0)+ b_0), W_1)+ b_1
y = Normal(loc=RN(data.x), scale=0.1)
```

Code (4) : La VA définie en sortie du RN à une couche cachée.



(a) Standard Neural Net (b) After applying dropout.

Figure 6 : le mécanisme de dropout consiste à supprimer au hasard des liens entre neurones. Les suppressions sont renouvelées aléatoirement pour chaque observation d'un jeu d'entraînement – source : [DML].

Dans le cadre d'une inférence variationnelle avec Edward, la définition des VA `qW_i` et `qb_i` correspondant à la distribution à postériori des VA `W_i`, `b_i` est un peu alambiquée. Il s'agit définir la classe des distributions, en l'occurrence des lois normales, parmi lesquelles on va chercher une solution approximative. On pourra trouver le code ici.

Le code du calcul d'inférence est simple. Il consiste d'une part à associer les VA à postériori (`qW_i` et `qb_i`) aux VA à priori (`W_i` et `b_i`) et d'autre part à associer les observations `data.y` à la VA `y` sortie du RN définie dans le Code (4):

```
import edward as ed

inference =
          ed.KLqp({W_0: qW_0,
                   b_0: qb_0,
                   W_1: qW_1,
                   b_1: qb_1},
                  data={y: data.y})
inference.run(n_iter=1000)
```

Code (5) : définition d'une inférence variationnelle avec Edward. Le premier paramètre de `KLqp()` est un dictionnaire qui associe les poids et les biais du RN aux VA latentes dont on se propose de calculer la distribution à postériori.

Edward propose également plusieurs options d'inférence avec des variantes de la méthode **Monte-Carlo** (MC), voir le tutoriel Edward.

D'autres exemples d'inférences éclairants& sont disponibles dans les tutoriels : régression linéaire, clustering etc…

En résumé : Edward est un système de PP scalable grâce à ses procédures d'échantillonnage de VA, il permet de composer différentes techniques d'inférences et tire profit de l'infrastructure de graphe de calcul de TensorFlow

4. De la R&D aux applications

Calcul d'intervalles de confiance pour un RN régularisé par dropout

Notre première illustration de l'utilité de l'approche bayésienne du ML est un travail de recherche remarquable [DBA] mené par deux chercheurs de l'université de Cambridge. Dans ce travail les auteurs ont dans un premier temps utilisé l'approche bayésienne du ML pour donner une justification rigoureuse de l'efficacité du mécanisme de **régularisation par dropout** des RN. Dans un second temps ils en ont tiré une méthode économique pour calculer des **intervalles de confiance** autour des prédictions d'un RN.

Rappelons que la régularisation par dropout, illustrée sur la figure 6, consiste à supprimer au hasard des liens w_{ij} entre les neurones i et j lors de l'**entraînement** du RN, la probabilité de suppression p étant un hyperparamètre. Les suppressions sont renouvelées aléatoirement pour chaque observation de l'ensemble d'entraînement, si bien que chaque donnée voit un RN légèrement différent. Durant la phase de **prédiction** on ajuste généralement les poids du RN ainsi entraîné pour que l'intensité des liens corresponde à une sorte de moyenne sur les réseaux générés par dropout à l'entraînement. Une meilleure approche consiste cependant à utiliser un processus stochastique également lors de la prédiction. On l'appelle le **MC-dropout**.

Sur un plan intuitif, la méthode MC-dropout revient donc à effectuer une sorte moyenne sur différents RN résultant des différents tirages des liens supprimés ce qui n'est en définitive rien d'autre qu'un RNB !

De fait, il est possible de montrer théoriquement [DBA] qu'il existe un lien précis entre la technique de MC-dropout dans le sens où chaque RN entraîné avec une régularisation par dropout est en fait un RNB !

Cette identification théorique d'un RN régularisé par MC-dropout avec un RNB permet de construire des estimations d'erreurs fiables et économiques en ressources. L'intuition est qu'on peut remplacer le calcul coûteux de la moyenne des prédictions pondérée par la distribution à postériori par le calcul économique du mécanisme de MC-dropout.

Une fois que l'on a compris qu'un RN régularisé avec MC-dropout n'est rien d'autre qu'un RNB déguisé, on peut essayer de comparer ses performances à celles d'un RNB calculé par les techniques d'inférence disponibles dans Edward [BDL]. Les résultats obtenus démontrent que les modèles calculés par Edward sont souvent *moins* performants que ceux calculés à peu de frais par MC-dropout. Les chercheurs pensent cependant avoir compris l'origine de ces faiblesses : les approximations variationnelles utilisées dans Edward sont pour l'instant trop grossières[4] pour que l'approximation calculée s'approche suffisamment de la vraie distribution à postériori. Ces intuitions ouvrent cependant la voie pour de futures améliorations des mécanismes d'inférences implémentés dans Edward.

Des modèles plus prudents face aux Adversarials Examples

Dès lors que l'approche bayésienne prétend intégrer un calcul systématique des incertitudes dans ses prédictions, par la dispersion de la distribution prédictive $P(y_{\text{NEW}} \mid \mathbf{x}_{\text{NEW}})$, il est tentant de la mettre à l'épreuve sur le **cas extrême** que représentent les AE. Ces cas sont extrêmes en effet dans le sens où les perturbations artificielles auxquelles ils correspondent sont, d'une certaine manière, très éloignés des exemples avec lesquels un modèle de RN a été entraîné.

C'est l'objectif que se sont fixés 3 chercheurs de *IBM Research* [AED]. Dans un travail expérimental minutieux, ils comparent la précision des prédictions et l'évaluation de l'erreur pour 4 modèles bayésien de classification d'images (dont le MC-dropout mentionné au paragraphe précédent) sur l'ensemble classique MNIST perturbé par des AE. Les métriques utilisées sont tirées de la théorie de l'information, la plus simple est l'**entropie prédictive** qui mesure l'incertitude associée aux prédictions que fait $P(y_{\text{NEW}} \mid \mathbf{x}_{\text{NEW}})$.

Les 4 modèles sont perturbés avec des intensités variables et de deux manières différentes. D'une part avec des AE, que l'on sait calculer pour être particulièrement « toxiques » grâce à la méthode FSTM[5]. D'autre part avec un simple bruit aléatoire.

La figure 7 résume les résultats obtenus. On constate qu'à intensité comparable, la précision chute plus rapidement pour des perturbations de type AE que pour de simples perturbations aléatoires ce qui correspond bien à l'intuition de toxicité de ces exemples particuliers.

En conclusion :

Les réseaux de neurones bayésiens ne sont pas plus robustes aux AE que leurs confrères non bayésiens. Mais eux au moins savent qu'ils ne savent pas !

(4) Il s'agit d'approximations de champ moyen pour lesquelles les poids associés aux différentes couches sont considérés comme des VA indépendantes.

(5) FGSM = Fast Gradient Sign Method.

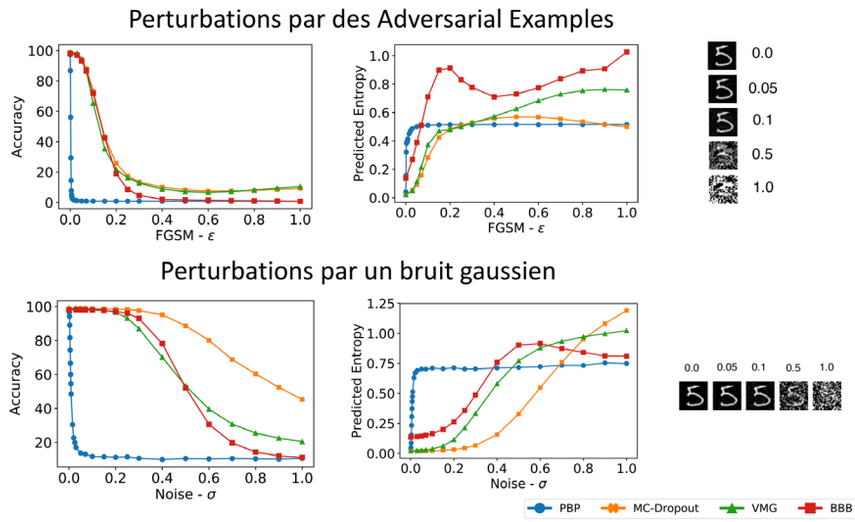


Figure 7 : L'évolution de la précision et de l'entropie prédictive de 4 RNB de classification d'images soumis à deux types de perturbations d'intensité croissante. En haut des AE, en bas du bruit gaussien. On constate que les AE font chuter plus rapidement la précision des prédictions – source : [AED].

5. À quand la démocratisation ?

Réaliser la synthèse de la flexibilité du Deep Learning avec un traitement bayésien de l'incertitude constituerait à n'en pas douter un **progrès majeur pour le ML**. Pour intégrer le ML dans des systèmes critiques, il est en effet indispensable qu'un algorithme sache identifier une situation ambigüe ou une situation jamais vue auparavant pour être en mesure de **déléguer** son analyse à un autre système ou à un humain.

Si les exemples d'applications précédent sont tous deux tirés de travaux de recherches récents, ce n'est pas un hasard. Les applications concrètes en effet se font encore attendre.

> *Pour l'instant les outils du ML bayésiens, comme le langage Edward, sont encore réservés aux chercheurs rompus aux multiples variantes des techniques d'inférence variationnelle ou par simulation MCMC.*

La syntaxe des langages de PP reste encore peu intuitive pour le profane. Elle exigera du data scientist féru de scikit-learn un effort conséquent pour repenser la notion d'apprentissage, non plus comme un problème d'optimisation, mais comme un problème d'inférence probabiliste. Rien n'empêche cependant d'imaginer l'émergence d'ici quelques années d'une nouvelle génération de data scientists ou même d'experts métiers pour qui ce mode de pensée sera naturel, d'autant plus que le paradigme bayésien est, de fait, très intuitif.

D'ici là, il faudra cependant que les outils de programmation probabiliste fassent des **progrès significatifs sur** plusieurs axes :

- L'**intelligibilité** de leur syntaxe
- L'**encapsulation** complète de la logique d'inférence
- Le **temps d'exécution**
- La **précision** des méthodes inférence

Des systèmes comme Edward, qui ambitionnent d'être scalables en tirant parti de l'infrastructure de TensorFlow, en sont vraisemblablement les prémices.

En ce domaine de l'IA comme dans d'autres, nous avons la chance de vivre une époque ou des idées longuement muries sur un plan théorique passent progressivement dans le champ de l'expérimentation puis des applications pratiques. Hier grâce à la loi de Moore. Aujourd'hui grâce aux infrastructures massivement parallèles des GPU et autres TPU. Demain l'informatique quantique prendra peut-être la relève.

Soyons donc patients et curieux. Si l'aptitude au doute raisonnable est un attribut d'une intelligence authentique, il n'est finalement guère surprenant qu'en doter les machines ne soit pas un problème si simple.

Références

- [DBA] **Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning**, *Y. Gal, Z. Ghahramani* – octobre 2016
  arxiv.org/abs/1506.02142

- [DML] **Dropout in (Deep) Machine learning**, *A. Budhiraja*, Medium – décembre 2016
  medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

- [ELP] **Edward: A library for probabilistic modeling, inference, and criticism**, *D. Tran, A. Kucukelbir, A.B. Dieng, M. Rudolph, D. Liang, and D.M. Blei* – février 2017
  arxiv.org/abs/1610.09787

- [AED] **Adversarial Phenomenon in the Eyes of Bayesian Deep Learning**, *A. Rawat, M. Wistuba, M.-I. Nicolae* – novembre 2017
  arxiv.org/abs/1711.08244

- [PAI] **Probabilistic machine learning and artificial intelligence**, *Z. Ghahramani*, Nature **volume** 521, p452–459, mai 2015
  www.nature.com/articles/nature14541

- [BDL] **Bayesian Deep Learning with Edward (and a trick using Dropout)**, *A. Rowan*, Vidéo PyData – mai 2017
  www.youtube.com/watch?v=I09QVNrUS3Q

- [DPP] **Deep Probabilistic Programming Languages: A Qualitative Study**, *G. Baudart, M. Hirzel, L. Mandel* – avril 2018
  arxiv.org/abs/1804.06458

- [TFD] **TensorFlow Distributions**, *J.V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, R.A. Saurous* – novembre 2018 arxiv.org/abs/1711.10604

- [CSE] **Un nouveau chantier pour la sécurité : le Machine Learning**, *P. Lemberger*, IA Lab weave – août 2017
  weave.eu/nouveau-chantier-securite-it-machine-learning/

- [HOM] **Hands-On Machine Learning with Scikit-Learn and TensorFlow**, *A. Guéron*, O'Reilly Media – mars 2017
  http://shop.oreilly.com/product/0636920052289.do

- [BIS] **Pattern Recognition And Machine Learning**, *Ch. Bishop*, Springer-Verlag NewYork Inc – 2006
  www.springer.com/us/book/9780387310732


Séquence vidéo
Émotions
Réseaux de convolution
Intonation vocale
Long short time memory
Sentiment analysis
Éthique
Expressions faciales
Biais culturel
Informatique affective

Le Deep Learning au service de l'informatique affective

Pirmin Lemberger – **weave Business Technology**

Résumé

Les algorithmes de Deep Learning ont récemment contribué à faire progresser les performances des systèmes d'analyse d'émotions, qu'il s'agisse d'analyser des textes écrits, des expressions du visage ou des variations dans les intonations de la voix. Pour chacune de ces trois modalités, nous présentons quelques cas d'utilisation emblématiques et nous décrivons les architectures typiques de réseaux neurones utilisés pour réaliser ces systèmes. Enfin, nous esquissons la réflexion sur les questions éthiques que soulèvent ces nouvelles technologies.

1. 66 % de la communication

C'est la proportion d'information que véhiculent les composantes non-verbales de la communication humaine, qu'il s'agisse des expressions du visage, de l'attitude corporelle ou des intonations vocales. La reconnaissance des émotions y joue un rôle prépondérant, puisque c'est elle qui révèle l'état mental de nos interlocuteurs et nous permet d'adapter notre propre communication au contexte d'un échange. C'est en définitive une aptitude qui est au cœur même de toute vie sociale et, à ce titre, elle est constitutive de notre humanité.

L'**informatique affective**, qui étudie la conception de systèmes capables de reconnaître ou de simuler des émotions, est aujourd'hui en plein essor. À terme, elle nous promet des interactions beaucoup plus naturelles avec les machines. Grâce aux avancées récentes de l'apprentissage profond, les progrès sont particulièrement significatifs dans le domaine de la reconnaissance des émotions. Un champ immense de possibilités s'ouvre désormais aux applications métiers, grâce la prise en compte de ce nouvel élément de contexte dans les échanges d'information. Dans cet article nous décrirons quelques **cas d'utilisation** emblématiques pour chacune des trois modalités suivantes : l'**analyse de texte**, l'analyse des **expressions faciales** et l'analyse des **intonations vocales**.

Nous décrirons ensuite les architectures types des réseaux de neurones utilisés pour chaque modalité. Enfin, nous esquisserons quelques **questions éthiques** que soulève l'informatique affective, sans souci d'exhaustivité.

Une question ouverte que nous n'aborderons pas ici est celle de la conception de machines qui ressentiraient, d'une certaine manière, des émotions, une étape qui pourrait cependant s'avérer incontournable pour le développement d'intelligences artificielles plus évoluées [TEN].

2. Des applications à foison

Reconnaissance d'émotions dans un texte

La détection d'émotions dans le texte est un usage classique du machine learning qu'on désigne parfois par le terme de « sentiment analysis ». Parmi les applications usuelles on trouve par exemple :

- L'analyse de l'impact émotionnel sur les réseaux sociaux d'une campagne de marketing.

Figure 1 Les 7 émotions de base.

- La détection de mouvements d'humeur, de comportements violents ou de harcèlement sur les réseaux sociaux.

- Une solution comme *DeepBreath* de Google propose quant à elle d'évaluer la tonalité d'un mail et d'informer son auteur lorsqu'un mail est jugé trop agressif ou simplement discourtois.

Reconnaissance d'émotions par l'expression faciale

Les expressions du visage peuvent naturellement servirent, elles aussi, à évaluer la satisfaction d'un client aux prises avec un service après-vente ou à face à un produit récemment acquis dont il s'agit de comprendre le fonctionnement. On peut encore mentionner les applications suivantes :

- La détection d'un manque d'attention chez un conducteur en vue d'augmenter la sécurité de la conduite.

- L'évaluation du niveau de stress de passagers à l'atterrissage ou à l'arrivé en gare ou la détection de comportements suspects.

- L'humanisation des robots dans leurs interactions avec les humains dont ils prendraient en compte l'état psychique. Cet usage n'est toutefois pas sans poser de sérieuses questions éthiques, nous y reviendrons dans la conclusion.

- Un système de reconnaissance émotionnel pourrait apporter une forme d'assistance ou de coaching à des personnes atteintes d'autisme, à condition que la technologie puisse être intégrée à objets *wearable* du genre Google Glass [AUT].

Reconnaissance d'émotions dans la voix

L'application la plus fantasmée dans ce domaine est le détecteur de mensonge. Plusieurs analyses scientifiques [LID, TDE] ont toutefois démontré que ces outils ne sont pas vraiment fiables.

- Incorporer les émotions et les intonations dans un outil de traduction simultanée joue un rôle important pour rendre plus la voix synthétique plus naturelle.

Figure 2. Un message d'avertissement à l'attention d'un conseiller lorsque le système détecte des intonations d'insatisfaction chez un client.

- Une application concrète de la détection d'émotion par la voix est le coaching émotionnel d'employés dans des call-center. Un logiciel comme *Cogito* suggère par exemple à un conseiller clientèle d'adapter en temps réel son discours à l'état émotionnel de son client [VDI], voir la figure 2.

La détection multimodale

L'idéal en matière de détection d'émotions consisterait à **intégrer les modalités vocale et faciale** au sein d'un même système. Certaines émotions, comme la colère par exemple, se manifestent en effet de manière prédominante par la voix alors que d'autres, comme la joie, se manifestent plutôt dans l'expression faciale. La conjonction des deux informations a le potentiel d'améliorer de manière drastique la précision de l'identification d'un état émotionnel. Pour l'instant le sujet est encore l'objet de travaux de R&D.

Notons encore qu'il existe de multiples API's de détection d'émotions prêtes à l'emploi.

3. Où le Deep Learning fait la différence

L'IA émotionnelle regroupe un ensemble de cas d'utilisation du machine learning pour lesquelles les techniques d'apprentissage profond sont particulièrement bien adaptées. Pour le comprendre, rappelons qu'un algorithme de machine learning supervisé est entraîné au moyen d'une liste d'exemples, chacun étant décrit pas certaines variables prédictives ou « features ». Ces features sont des données numériques qui caractérisent, par exemple, l'image d'un visage ou un extrait audio qui exprime une émotion spécifique, comme la colère, la joie ou la peur. Deux possibilités se présentent alors pour construire ces features.

La première consiste à recourir à une expertise métier, celle d'un physionomiste par exemple, pour construire manuellement ces variables prédictives à partir des traits d'un visage : inclination des sourcils, angles que forme les commissures des lèvres etc. C'est ce que l'on appelle communément le **feature engineering**. L'inconvénient de ce processus d'extraction manuel des features est qu'il comporte une part d'arbitraire et qu'il n'exploite pas nécessairement l'intégralité de l'information disponible dans une photo ou dans une vidéo.

La deuxième possibilité consiste à utiliser un réseau de neurones profond. Ces algorithmes sont en effet capables de court-circuiter l'étape de feature engineering dans le sens où ils parviennent, pour chaque type de prédiction, à **agréger toute l'information utile** disponible dans les données brutes, qu'il s'agisse en l'occurrence de textes, d'images, de vidéo ou d'échantillons de voix. Pour bénéficier de cet atout il faudra cependant renoncer à interpréter les prédictions d'un tel prédicteur et disposer d'un grand nombre d'exemples étiquetés.

Comme dans beaucoup de problèmes d'IA, c'est là que réside l'une des principales difficultés **:** construire des **jeux de données assez riches** pour entraîner les algorithmes de deep learning à reconnaître des émotions.

À quoi il faut encore ajouter des difficultés d'un autre ordre :

- Le **caractère très privé des données** de type émotionnel fait qu'il est par définition difficile, voire même souvent interdit d'y avoir accès.

- Si, pour contourner la difficulté précédente, on demande à des personnes de jouer une émotion, le **caractère contraint** ou maladroit de leur simulation faussera nécessairement les données.

- À supposer que des échantillons contenant des émotions exprimées spontanément puissent effectivement être récupérés, il faudra encore procéder à leur étiquetage. La difficulté de l'opération tient alors aux différences très ténues qui distinguent les expressions associées à des émotions comme la peur ou la surprise. On estime que les jugements humains coïncident, au mieux dans 70% à 80% des cas [AFF], ce qui confère par conséquent un caractère partiellement subjectif à l'**étiquetage** de données émotionnelles.

- Des **biais culturels** ou liés au genre doivent être pris en compte, aussi bien pour la source des émotions que pour l'interprète en charge de les identifier.

Architecture pour la classification de textes*[1]

L'analyse des émotions dans un texte, souvent appelée aussi « **sentiment analysis** » ou « **opinion mining** », est une application classique du machine learning qui existait bien avant que n'apparaisse le Deep Learning. La stratégie de base consiste à utiliser un algorithme de machine learning supervisé sur un corpus de texte manuellement étiqueté avec les émotions que l'on souhaite identifier. Dans l'approche élémentaire, les variables prédictives sont simplement constituées des fréquences d'apparition, correctement normalisées[2], des principaux termes ou n-grammes qui apparaissent dans le corpus. On appelle cette approche le « **bag of words** » car elle tient compte uniquement de la fréquence d'apparition des mots sans prendre en considération leur ordre dans les phrases. Bien que rudimentaire, cette approche donne cependant souvent de bons résultats dans les cas simples.

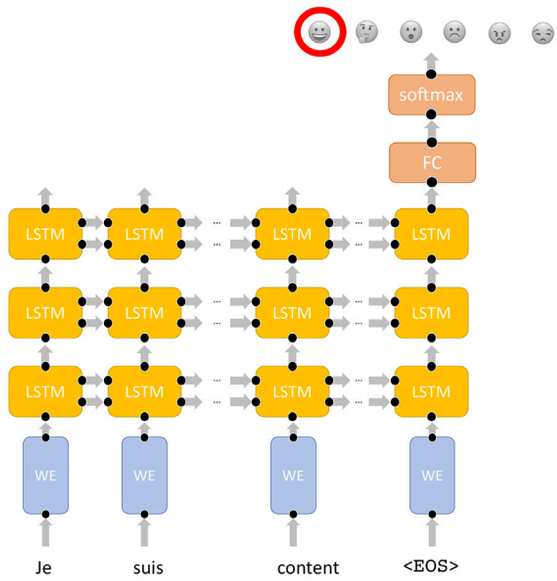


Figure 3 : Une architecture de RNN typique pour la classification de texte par émotion. La couche de Word Embedding (WE) convertit chaque mot en un vecteur. Les couches de LSTM convertissent la suite de ces vecteurs en un vecteur qui est injecté dans une ou plusieurs couches denses (FC) en bout de chaîne avec une activation softmax qui attribue une probabilité à chaque émotion.

Pour aller plus loin et tenir compte aussi bien du sens des mots que de leur ordre dans une phrase, le Deep Learning offre des solutions plus performantes. Les variantes de réseaux profonds sont nombreuses mais utilisent toutes, peu ou prou, deux ingrédients comme l'illustre la figure 3. Le premier mécanisme est ce qu'on appelle le « **word embedding** ». Il encode chaque terme d'une phrase en un vecteur numérique tout en respectant une notion de proximité sémantique au sens où deux termes similaires seront encodés par des vecteurs proches. Le second est l'utilisation d'un **réseau de neurones récurrent** (RNN) avec une cellule mémoire à long terme, comme les LSTM, ce qui permet de tenir compte de l'ordre chronologique des mots même dans des longues phrases. La sortie du RNN en bout de chaîne, en haut à droite dans la figure 3, encode en principe le sens approximatif de la phrase. Un réseau dense terminé par une couche softmax la convertit enfin en une distribution de probabilité sur les émotions que l'on souhaite détecter. Nous ne présenterons pas ici le détail de ces éléments car nous les avions déjà décrits dans le cadre de notre article consacré au mécanisme d'attention.

Architecture pour la détection d'émotions sur le visage*

Deux options se présentent pour détecter une émotion sur un visage : l'analyse d'une **image statique** ou l'analyse d'une courte **séquence vidéo**. L'identification d'une émotion à partir d'une séquence vidéo est plus fiable, on s'en doute, qu'à partir d'une photo mais elle est également beaucoup plus gourmande en ressources. Pour ces deux problèmes de classification, nous décrirons succinctement l'architecture type de réseaux de neurones que l'on peut utiliser. Les variantes sont nombreuses et sont toujours l'objet de travaux de recherche. Tous utilisent les **réseaux de convolution** (CNN) que nous avions succinctement présentés dans l'article consacré au mécanisme d'attention.

(1) Cette section ainsi que les deux suivantes marquées d'un * présuppose une certaine familiarité avec le fonctionnement des réseaux de neurones récurrents.

(2) Les fréquences TD-IDF sont le plus souvent utilisées.

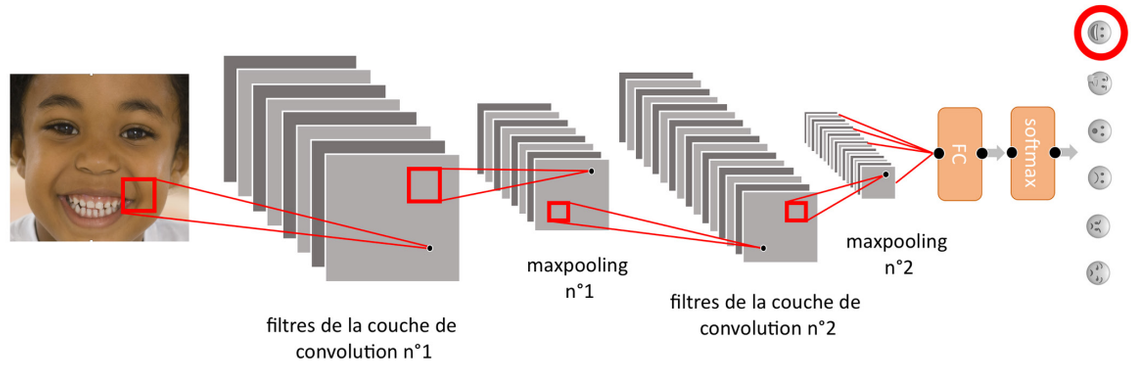


Figure 4 : un CNN classique utilisé comme classifieur d'émotions faciales.

Image statique

La classification d'une image statique d'un visage selon l'émotion qu'il exprime peut se faire au moyen d'un CNN classique, où alternent les **filtres** de convolution et les couches de **maxpooling**. Les filtres apprennent à reconnaître des caractéristiques utiles propres à chaque échelle de l'image, alors que les couches de maxpooling en réduisent progressivement la résolution pour détecter des caractéristiques de plus en plus globales. Comme précédemment, plusieurs couches denses terminées d'une activation softmax achèvent de convertir l'information extraite du CNN en une distribution de probabilité sur les émotions que l'on souhaite détecter.

Séquence vidéo

Classifier en temps réel les expressions d'un visage demande d'utiliser une architecture qui sache tirer parti de l'évolution temporelle de l'expression d'un visage. Pour cela on utilise, comme pour l'analyse d'une phrase écrite, un réseau de neurones récurrent (RNN). L'architecture représentée sur la figure 5 se distingue sur trois points principaux de celle représentée sur la figure 3 :

1. La couche de word embedding (qui convertissait précédemment chaque mot en vecteur) est ici remplacée par un CNN qui convertit l'image à l'instant *t* en un vecteur. L'empilement des LSTM qui consomme ces vecteurs au cours du temps est similaire à celui de la figure 3.

2. On s'intéresse cette fois-ci à la sortie des LSTM à chaque instant et non plus seulement à la fin de la séquence car l'émotion que l'on souhaite détecter peut varier au cours du temps.

3. Les couches denses en charge du calcul de la distribution de probabilité sur les émotions à l'instant *t* sont alimentées à la fois par la sortie des couches de LSTM, ce qui permet de prendre en compte l'évolution l'expression du visage avant cet instant *t* et, directement, par la sortie du CNN, ce qui permet de tenir compte de l'expression à cet instant même.

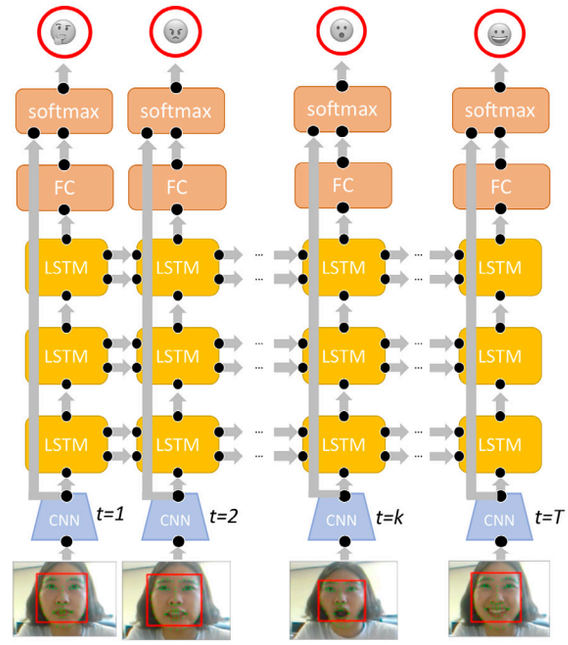


Figure 5 : L'architecture type d'un RNN qui analyse en temps réel les émotions dans une séquence vidéo. Les CNN convertissent l'image en features qui alimentent une couche de LSTM. Ceux-ci détectent les caractéristiques temporelles de l'évolution du visage. Plusieurs couches denses convertissent finalement, à chaque instant t, la sortie des LSTM en distribution de probabilité sur les émotions à cet instant.

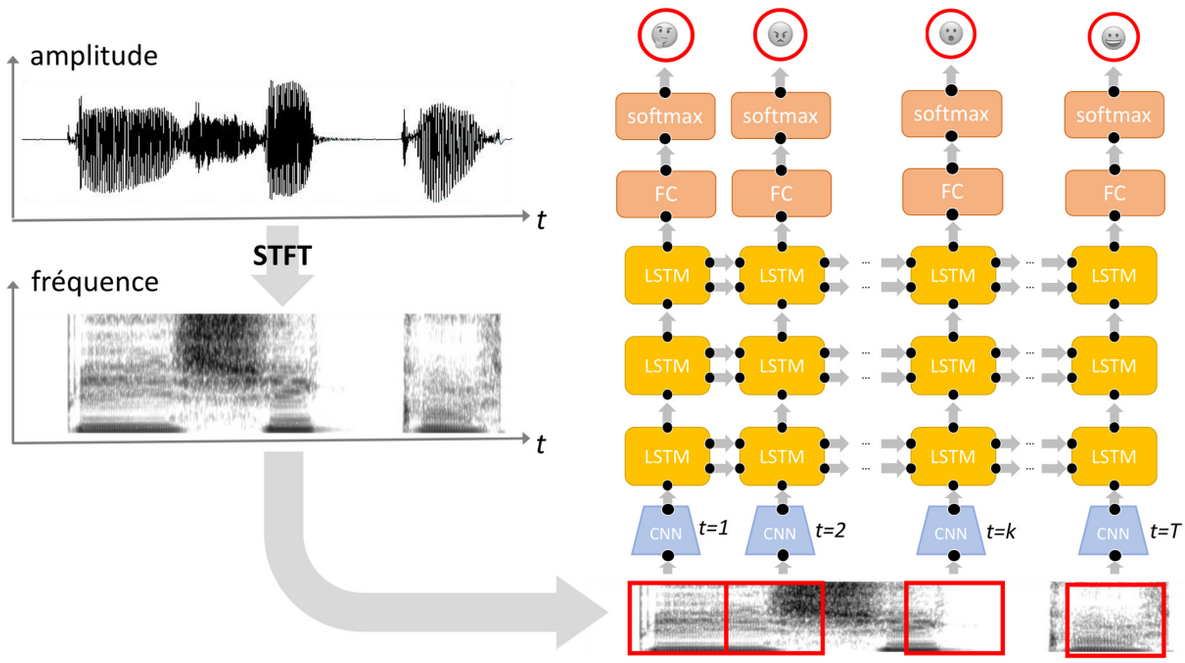


Figure 6 : L'extrait audio est dans un premier temps converti au moyen d'une transformation de Fourier à court terme (STFT) en un spectrogramme. Ce spectrogramme est l'analogue de la vidéo de la figure 5. Il alimente un RNN dont l'architecture est similaire à celle utilisée pour l'analyse de séquences vidéo.

Architecture pour la détection d'émotions dans la voix*

L'analyse d'une séquence audio (SER)[3] est très similaire à l'analyse d'une séquence vidéo présentée précédemment. Dans un premier temps, le signal audio est converti en un **spectrogramme** temps fréquence au moyen d'une transformation de Fourier à court terme (STFT). Ce spectrogramme peut s'envisager comme l'équivalent de la séquence vidéo de l'exemple précédent. Cette image 2D qui varie au court du temps va alimenter un RNN dont l'architecture est similaire [SER] à celle utilisée pour la détection d'émotions dans des séquences vidéo. Le canal de communication direct entre les CNN les couches denses n'a pas d'utilité dans ce cas.

La petite taille des jeu données disponibles est l'une des principales difficultés à surmonter pour construire un système SER. Certains progrès récents reposent davantage sur des techniques d'enrichissement de données (**data augmentation**) que sur des améliorations algorithmiques. Une équipe chinoise prétend avoir mis au point une telle technique qui ferait passer la précision de reconnaissance des émotions de 70% à 99%. La phrase est ici mise au conditionnel car on peut légitimement douter qu'un score aussi stalinien ne soit en réalité symptomatique d'un simple surapprentissage. Leur technique consiste à distordre chaque spectrogramme de différente manière pour en créer plusieurs équivalents en termes de contenu émotionnel [ABR].

Une autre difficulté, propre à la modalité vocale, tient au fait qu'un algorithme audio n'analyse que l'intonation de la voix sans tenir compte du sens des mots que contient le message. Un humain intègrera quant à lui l'information **sémantique** de la phrase à son appréciation, si bien qu'il saura déceler sans peine la colère contenue d'un message d'insultes prononcé à voix basse. Cette interférence entre le sens des mots et l'intonation d'un message risque de fausser une partie des jeux de données.

(3) SER = Speech Emotion Recognition

4. Une vaste duperie ?

Tout système de traitement de l'information pose des **questions éthiques** car nos actions et nos jugements sont influencés par l'information à laquelle nous accédons. L'élaboration de systèmes capable de détecter des émotions humaines ne fait pas exception et pose même certaines questions inédites que nous décrivons brièvement.

Le cœur du problème en l'occurrence réside dans l'**asymétrie fondamentale** qui existe entre une machine capable d'identifier une émotion humaine sans toutefois en ressentir elle-même. Dis autrement, on peut se poser la question de savoir si une information émotionnelle ne devrait pas être traitée uniquement par des êtres capables d'éprouver les émotions qu'ils détectent chez leurs semblables. Sans recours possible à un mécanisme d'empathie, l'usage de cette information ne risque-t-elle pas d'être une forme de tromperie ou d'usurpation de conscience ?

La question prend d'autant plus d'acuité qu'une émotion est souvent associée, explicitement ou implicitement, à un **jugement moral**. Ainsi, dans nombre de circonstances, la colère est assimilée à un manque de maîtrise de soi. La tristesse est parfois associée à de la faiblesse, la surprise à de la naïveté etc… Dès lors, déléguer à une machine, par essence faillible, la détection d'une information susceptible d'influer notre jugement moral, et par conséquent nos actions à l'égard d'une personne, peut poser question. Le caractère opaque des réseaux de neurones, qui est en fait la contrepartie de leurs capacités prédictives, complique encore la donne car il rend difficile, voire impossible, toute justification de leurs analyses en termes humainement intelligibles. L'exemple extrême est celui du **détecteur de mensonge** dont plusieurs études [LID, TDE] ont démontré qu'ils étaient plus enclins à stigmatiser des innocents qu'à détecter d'authentiques mensonges.

L'une des applications emblématiques et d'ores et déjà utilisée de la reconnaissance d'émotion est le coaching de conseillers clientèles dans des call centers; nous l'avons déjà évoqué. On peut imaginer que, généralisé et poussé à l'extrême, ce genre d'usage pourrait conduire à une forme inédite de **taylorisme**. Chaque employé serait relégué au rang de simple nœud de traitement de l'information, un nœud qu'il s'agit d'optimiser, ou peut-être faudrait-il parler plutôt de dressage en l'occurence, grâce à l'IA émotionnelle. Se dessine alors une dystopie par l'uniformisation des comportements au nom de l'efficacité et de la sacro-sainte **satisfaction du client**.

Si l'hypothèse précédente peut paraître un tantinet sombre, d'autres phénomènes d'**uniformisation** ou d'**appauvrissement**, de produits cette fois, paraissent en revanche plus plausibles. Disney est par exemple en train de mettre au point un système capable de mesurer en temps réel les réactions d'une audience face aux blagues du scénario de *Toy Story 5* [DBF]. De là, il n'y a qu'un pas pour entrevoir une normalisation de toute sortes de produits culturels ou de divertissement. Ou alors, à l'inverse, on peut imaginer une segmentation fine de produits, scientifiquement calibrés, pour rencontrer telle ou telle micro-population, un peu à l'image de ces sites de rencontre spécialisés dans une micro-niche socio-culturelle.

Plus subtiles pourrait être les conséquences de l'IA émotionnelle sur notre **libre arbitre**. Imaginons une situation où une IA émotionnelle

aurait détecté du stress ou de la peur chez une personne et serait programmée pour adapter son message afin d'éviter d'accroitre chez cette personne les émotions négatives. L'information qu'elle reçoit serait alors tributaire de son état émotionnel plutôt que de sa faculté de jugement qui lui dicterait, peut-être, un comportement différent si elle avait accès à une information non filtrée. Le courage en effet n'est pas l'absence de peur ou de stress mais bien la capacité à les surmonter pour agir en pleine conscience. Une capacité qui serait en l'occurrence court-circuitée par l'IA, privant un individu d'une part constitutive de son humanité.

On le voit, les questions éthiques sont nombreuses et encore largement ouvertes. En l'absence d'un « serment d'Hippocrate » pour les guider, les concepteurs d'IA et les experts en neuromarketing pourront toujours commencer par méditer, si ce n'est déjà fait, le célèbre **impératif catégorique** kantien :

« Traite toujours autrui comme une fin et jamais seulement comme un moyen ! »

Emmanuel Kant (1724 -1804)

Références

- [FER] **A Brief Review of Facial Emotion Recognition Based on Visual Information**, *Byoung Chul Ko* – janvier 2018
www.ncbi.nlm.nih.gov/pubmed/29385749

- [SER] **Speech emotion recognition using convolutional and Recurrent Neural Networks**, *W. Lim, D. Jang, T. Lee*, IEEE Xplore – janvier 2017
ieeexplore.ieee.org/document/7820699/

- [VDI] **This Call may be Monitored for Tone and Emotion**, *T. Simonite*, Wired – mars 2018
www.wired.com/story/this-call-may-be-monitored-for-tone-and-emotion/

- [DBF] **Disney Is Building Facial Recognition to Figure Out When You'll Laugh During Toy Story 5**, *J. Brown* – juillet 2017
gizmodo.com/disney-is-building-facial-recognition-to-figure-out-whe-1797267294

- [PDL] **Deep Learning with Python**, *F. Chollet*, Manning – octobre 2017
www.manning.com/books/deep-learning-with-python

- [TDE] **The accuracy of auditors' and layered voice Analysis (LVA) operators' judgments of truth and deception during police questioning**, *F. Horvath , J. McCloughan, D. Weatherman, S. Slowik*, US National Library of Medicine – février 2013
www.ncbi.nlm.nih.gov/pubmed/23406506

- [LID] **The Polygraph and Lie Detection**, the National Academies Press – 2003
www.nap.edu/read/10420/chapter/1

- [TEN] **MIT AGI: Building machines that see, learn, and think like people**, *J. Tenenbaum*, YouTube – février 2018
www.youtube.com/watch?v=7ROelYvo8f0&feature=youtu.be&t=1h19m36s

- [ABR] **A breakthrough in Speech emotion recognition using Deep Retinal Convolution Neural Networks**, *Yafeng Niu, Dongsheng Zou, Yadong Niu, Zhongshi He, Hua Tan*, arXiv:1707.09917v1 – juillet 2017
arxiv.org/abs/1707.09917

- [API] **20+ Emotion Recognition APIs That Will Leave You Impressed, and Concerned**, *B. Doerffeld* – août 2016
nordicapis.com/20-emotion-recognition-apis-that-will-leave-you-impressed-and-concerned/

- [AUT] **L'ordinateur enseigne aux personnes autistes à reconnaître les émotions**
nos.nl/artikel/2069009-computer-leert-autisten-emoties-herkennen.html

- [EIA] **Ethical Issues in Affective Computing**, *R. Cowie*, Oxford Handbooks Online – avril 2014.
www.oxfordhandbooks.com/view/10.1093/oxfordhb/9780199942237.001.0001/oxfordhb-9780199942237-e-006


Transfer learning
ELMo
Pré-entraînement
Mécanisme d'attention
Fine tuning
ULMFiT
BERT
Sémantique syntaxe
Réutilisation
Deep Learning
NLP
Word embeddings
Contexte

Deep Transfer Learning - le traitement du langage à l'aube d'une révolution ?

Pirmin Lemberger – **weave Business Technology**

Thomas Scialom – **ReciTAL.**

Résumé

L'année 2018 a été particulièrement faste en ce qui concerne les progrès en traitement du langage naturel. Pour la première fois certains modèles dépassent même les performances humaines sur des problèmes réputés difficiles comme la capacité à répondre à des questions suite à la lecture d'un texte. Ces progrès reposent d'une part sur de nouvelles architectures de réseaux de neurones et d'autre part sur deux grandes classes d'idées : l'idée de modèle de langage et l'idée de transfert d'apprentissage bien connue en vision artificielle. L'objectif de cet article est de présenter trois de ces méthodes : ULMFiT, ELMo et BERT. Pour notre présentation nous nous appuierons essentiellement sur des schémas d'architectures simplifiés.

1. Le vieux rêve de parler aux machines.

Et si le destin de la technologie était de se faire oublier ? De constituer un ensemble d'outils si puissants qu'ils sauraient se faire humbles pour se mettre entièrement au service des hommes sans exiger d'eux ni adaptation, ni laborieux apprentissage ? On peut le souhaiter et, depuis une quarantaine d'années, c'est apparemment le sens de l'histoire avec la démocratisation de l'informatique via des interfaces homme-machine (IHM) graphiques intuitives.

Avec les chatbots [CHL] sont apparus ces dernières années de nouvelles IHM vocales et textuelles qui ambitionnent de permettre des interactions en langue naturelle. Pour l'instant très rudimentaires et réduites à quelques interactions **transactionnelles** sur le mode question-réponse, elles préfigurent peut-être une révolution à plus long terme qui concrétiserait la démocratisation ultime de l'informatique, celle à laquelle ont rêvé les pionniers de la discipline il y 80 ans[1]. Des machines que l'on pourrait interroger sur le mode de la **conversation** pour obtenir une information ou déclencher une action.

Dans cette longue marche vers la conception de machines capables de conversation, l'année 2018 a été particulièrement faste avec des progrès incrémentaux mais très significatifs dans le domaine du **NLP** (Natural Language Processing). Le **Deep Learning** (DL) n'étant pas avare de petits miracles c'est à lui, on s'en doute, que l'on doit ces derniers progrès. L'objectif de cet article est de présenter les idées et les grandes lignes des architectures de réseaux de neurones (NN ci-après pour Neural Network) qui ont permis ces avancées.

(1) Nous pensons ici en particulier au célèbre test de Turing.

Avant cela, donnons quelques exemples de problèmes qui relèvent du NLP et pour lesquels on dispose de jeux de données pour évaluer les performances d'un algorithme.

- La **classification de textes** intervient dans beaucoup d'usages. Le sentiment analysis et l'anti-spam sont des usages classiques pour lesquels des algorithmes de Machine Learning existent depuis une vingtaine d'années. On peut ranger aussi dans cette catégorie l'**extraction d'entité nommées** (« Named Entity Recognition ») qui consiste à reconnaître des entités dans un texte et à les classer en catégories comme des personnes, des organisations, des lieux etc… C'est une tâche utilisée notamment par les chatbots lorsqu'il s'agit de caractériser l'intention d'un utilisateur [CHL]. Enfin, la **déduction à partir d'une hypothèse** (« Textual Entailment »), qui consiste à déterminer si une phrase est la conséquence logique d'une autre, est une autre forme de classification.

- La **réponse à une question** (« Question Answering ») consiste à trouver dans un paragraphe de texte la suite de mots qui constitue la réponse à la question posée. C'est un des problèmes réputés difficile du NLP

- L'**évaluation d'un degré d'équivalence sémantique** entre deux phrases dont l'utilité est évidente dans un grand nombre de cas d'utilisation.

Rappelons ici les raisons principales qui font du NLP un problème ardu de l'IA. La première tient à la difficulté de construire des ontologies conceptuelles et des règles syntaxiques qui encapsuleraient une langue, sa syntaxe et sa sémantique, dans un système formel programmable. Une démarche tentée durant les années 1970 -1980 mais avec un succès très mitigé. Cet échec relatif a conduit les chercheurs à développer depuis une vingtaine d'années des méthodes d'apprentissage statistiques, au premier rang desquelles figure aujourd'hui le Deep Learning (DL). Jusqu'à récemment, ces modèles exigeaient cependant de très gros volumes de données étiquetés pour être entraînés, au point même que la tâche pouvait s'avérer d'une ampleur prohibitive. Enfin, la qualité d'un modèle de NLP (de traduction automatique p.ex.) relève parfois d'une impression subjective qu'il n'est pas toujours aisé d'objectiver[2].

La suite de cet article est organisée comme suit. La ***section 2*** présente les deux idées clés des progrès récents en NLP : l'idée d'apprentissage par transfert ou Transfert Learning et la notion de modèle de langage. La ***section 3*** décrits les trois nouveaux modèles **ULMFiT**, **ELMo** et **BERT**. La ***section 4*** conclu sur quelques problèmes ouverts.

(2) Même si de nombreuses métriques ont été conçues à cet effet comme la métrique BLEU par exemple

2. Les deux idées clés

Bien que différents, les nouveaux modèles de NLP s'appuient sur deux idées qu'il convient d'avoir présentes à l'esprit si l'on veut bien comprendre ces nouvelles architectures : l'idée de **Transfert Learning** et la notion de **modèle de langage**. Les lecteurs familiers avec ces idées peuvent poursuivre la lecture à la section 3.

L'idée d'apprentissage de Transfert Learning

Dans le contexte du NLP, un exemple élémentaire de Transfert Learning est l'utilisation de **word embeddings** (WE) dans la première couche d'un NN profond [MAL]. Le rôle de cette couche, rappelons-le, est de convertir une suite de mots en entrée (ou, plus généralement, de token) en une autre suite de même longueur, de vecteurs. Chaque vecteur encode le contenu sémantique du mot auquel il est associé.

Les composantes des vecteurs de WE peuvent naturellement être apprises lors de l'entraînement du RN. Cependant, la solution habituelle consiste plutôt à réutiliser des versions précalculées de ces WE, *GloVe* et *Word2Vec* étant à ce jour les plus communes. Ces WE désormais classiques ont été construits par apprentissage statistique sur des problèmes artificiels qui n'ont pas forcément d'utilité directe, mais dont l'intérêt dans notre contexte est de permettre l'apprentissage de représentations numériques utiles des mots d'une langue. Ces modèles sont par ailleurs conçus de manière à n'exiger aucun travail d'étiquetage manuel pour leurs données d'entraînement. En fait ils ne font qu'exploiter l'ordre naturel des mots dans un des corpus de textes gratuits disponibles sur le web, comme *One Billion Words for LM* ou *WikiText-103*. À titre d'exemple, *GloVe* est construit à partir d'un modèle de prédiction de cooccurrence de mots dans un corpus de documents alors que *Word2Vec* est construit à partir d'un modèle capable de prédire un mot à partir des mots qui l'entourent. Peu importe ici les détails techniques de ces modèles, le point sur lequel nous voulons mettre l'accent est que ces WE préfabriqués constituent des exemples élémentaires de Transfert Learning.

> *En toute généralité le Transfer Learning consiste à réutiliser tout ou partie d'un modèle, préentraîné sur une tâche d'apprentissage A, comme brique de base pour construire un modèle prédictif pour une tâche B en bénéficiant de la connaissance acquise sur la tâche A.*

Pour que cette stratégie de transfert fonctionne correctement, le problème artificiel devra comporter une part significative de la difficulté du vrai problème. Intuitivement, c'est bien le cas avec les WE. S'ils s'avèrent utiles pour construire un modèle capable de prédire un terme à partir des mots qui l'entourent, cela confirme l'idée que ces WE encapsulent effectivement des représentations sémantiques approximatives de chaque mot, représentations dont on pourra tirer profit dans d'autres problèmes de NLP.

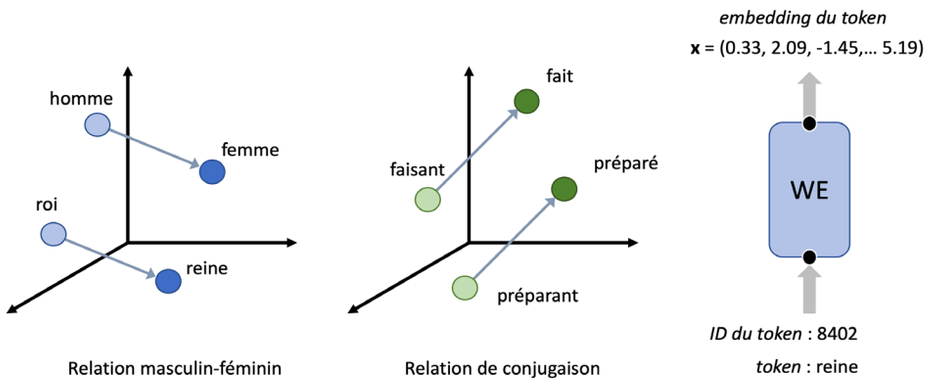


Figure 1 : L'opération classique de Word Embedding (WE) consiste à associer un vecteur à chaque mot ou token d'une liste. Un vecteur de WE comporte typiquement plusieurs centaines de dimensions.

Les trois méthodes présentées ci-dessous (ULMFiT, ELMo, BERT) ambitionnent chacune de généraliser ce mécanisme des WE. Au lieu d'associer un vecteur d'embedding statique à chaque mot, ces méthodes construisent des représentations plus riches qui prennent en compte le **contexte sémantique et syntaxique** de chaque mot. Ces trois méthodes utilisent pour cela différentes variantes de Transfer Learning sur lesquels nous reviendrons.

Un élément d'intuition important à garder à l'esprit lorsqu'on parle de Transfert Learning est l'idée de **couches d'abstraction**. Tous les modèles de Deep Learning sont organisés en couches, chacune d'elle étant responsable d'apprendre une partie de la transformation globale qui convertira l'entrée du modèle dans la prédiction souhaitée.

La sagesse populaire des praticiens du Deep Learning nous enseigne par ailleurs que les couches basses, comprendre proches de l'entrée, encodent une information très **générique** donc aisément réutilisable dans d'autres contextes. Pour un système de classification d'images

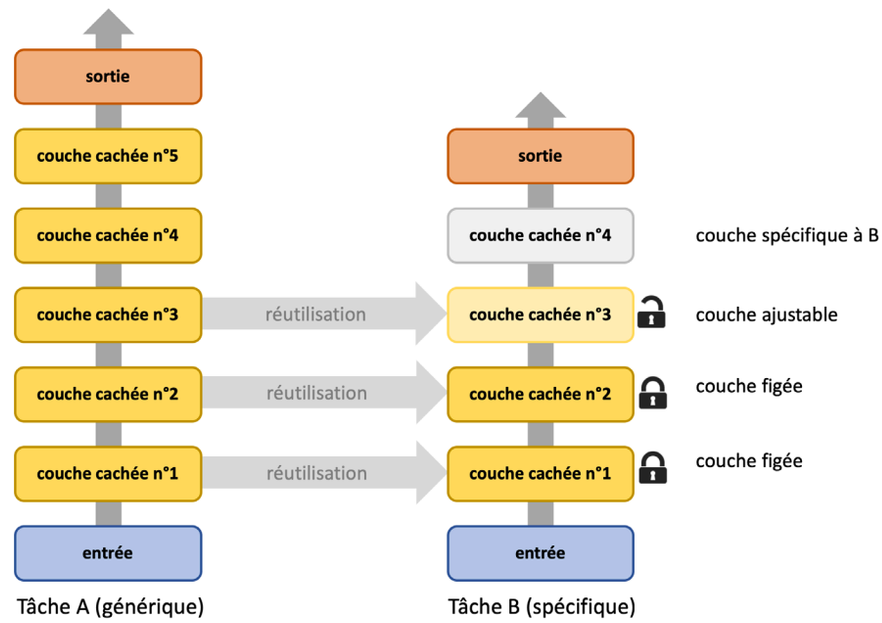


Figure 2 : Le Transfert Learning consiste à réutiliser les couches basses d'un NN profond entraîné sur une tâche A, parfois artificielle, pour résoudre une tâche B similaire mais directement utile. Les couches peuvent être figées ou ajustées à une tâche spécifique.

il s'agit par exemple d'informations de contraste entre zones homogènes, d'indicateurs de la présence de certaines textures etc… Les couches supérieures, plus proches de la sortie, encodent quant à elles une information plus **spécifique** à la tâche considérée (détecter la présence d'un visage p.ex.). Le Transfert Learning réutilise typiquement l'information sur les couches basses d'un réseau profond comme l'illustre la figure 2. Les notions d'abstractions sont similaires en NLP, quoique peut-être moins évidentes à formuler. La couche de WE encode, on l'a vu, une information sémantique et syntaxique, mot à mot. A mesure que l'on s'élève dans les couches la prise en compte de l'information contextuelle croît jusqu'à encoder le sens complet d'un texte.

Remarquons enfin que le Transfert Learning, quel que soit le domaine où il est mis en œuvre, permet de découvrir automatiquement des représentations utiles des données. En ce sens c'est une forme **feature engineering automatisé** qui est l'un des principaux atouts de l'apprentissage statistique par Deep Learning.

L'idée de Transfer Learning n'est pas neuve, elle est même ancienne dans les applications de vision artificielle qui, très souvent, réutilisent plusieurs couches basses d'un réseau de convolutions entraîné sur la classification de millions de clichés tirés de banques d'images comme *ImageNet*. La plupart des librairies de Deep Learning comme *TensorFlow*, *Keras* ou *PyTorch* mettent à disposition des modèles pré-entraînés dans leur arsenal.

Ce qui est inédit en revanche c'est l'application pratique et ultra-performante de cette stratégie de Transfert Learning au domaine du NLP. Pourquoi alors un tel délai dans l'application au NLP d'une idée aussi simple et ayant de surcroît fait ses preuves dans d'autres domaines de l'IA ? La raison est peut-être à trouver dans les détails techniques qui, nous le verrons, sont parfois délicats dans leur mise en œuvre et nécessitent de procéder par tâtonnements et par accumulation d'expériences.

La notion de modèle statistique de langage

Le choix d'une tâche artificielle adéquate sur laquelle on va entrainer un modèle en vue d'appliquer ultérieurement le Transfer Learning est crucial si l'on veut attaquer des problèmes ardus du NLP. Deux contraintes président au choix d'un tel problème. D'une part la tâche doit être suffisamment riche pour que le modèle résultant incorpore différentes facettes de la compréhension du langage comme : l'identification de synonymes, les dépendances à long terme, la résolution d'ambiguïtés lexicales, la capacité de déduction logique, la détection de négation où même peut-être de l'ironie. D'autre part, nous l'avons dit, il est souhaitable que le jeu de données d'entraînement puisse être constitué en court-circuitant le laborieux travail d'étiquetage qu'exige usuellement le ML supervisé.

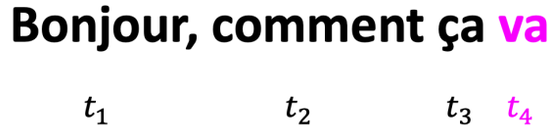


Figure 3 : Un modèle de langage prédit la probabilité d'occurrence d'un mot en fonction des mots qui le précèdent ou qui l'entourent sur une certaine fenêtre temporelle.

Dans les trois modèles présentés dans la section 3 le choix des chercheurs pour cette tâche artificielle s'est porté sur un **modèle statistique de langage** (Language Model, LM). Précisons rapidement ce qu'on entend par là. Un LM n'est rien d'autre qu'une distribution de probabilité empirique $P(t_1,...,t_m)$ sur des suites de mots $(t_1,...,t_m)$ (ou de tokens). Dans le contexte qui nous intéresse, et pour être un peu plus précis, on va apprendre à prédire la probabilité que le mot t_m succède à une suite de mots $(t_1,...,t_{m-1})$. C'est donc une probabilité conditionnelle $P(t_m \mid t_1,...,t_{m-1})$ qu'on va apprendre à l'aide d'un NN profond. Les données d'entraînement d'un tel modèle peuvent évidemment être constituées de n'importe quel corpus disponible gratuitement sur le web dans une langue donnée. L'étiquetage est en l'occurrence gratuit puisque l'étiquette associée à la suite de mots $(t_1,...,t_{m-1})$ n'est autre que le mot t_m.

Nous voilà équipés pour aborder les trois modèles qui ont battus tous les records en NLP !

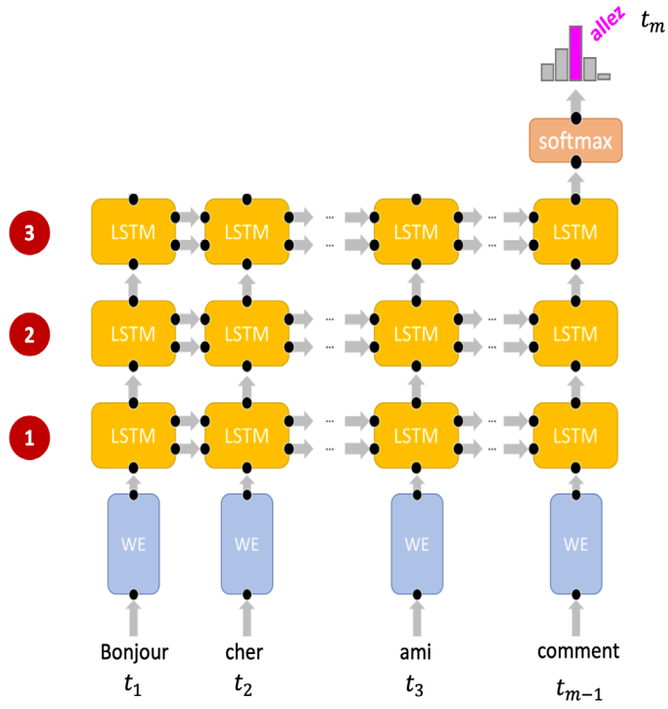


Figure 4 : Pour l'entraînement du LM on utilise un RNN formé de 3 couches de LSTM sans mécanisme d'attention mais doté de mécanismes de régularisation appropriés au NLP qui ne sont pas représentés ici.

3. La cuvée 2018 des modèles de NLP

L'objectif de cette section est de décrire trois techniques de Deep Learning qui ont récemment (2018) contribué à faire progresser l'état de l'art sur plusieurs problèmes parmi les plus ardus du NLP comme la capacité à répondre à des questions à propos d'un texte, dépassant même en cela les humains dans certains cas [BER]. Nous nous appuierons sur des schémas d'architectures[3] simplifiés des NN utilisés par chaque solution.

ULMFiT – le transfert par Fine Tuning

« ***U**niversal **L**anguage **M**odel **F**ine **T**uning* » (ULMFiT) est une méthode générique récente [ULM] qui permet de construire des systèmes de classification de textes extrêmement performants. Dans certains cas elle a permis de réduire le taux d'erreur de 24% par rapport aux meilleurs modèles existants, établissant un nouvel état de l'art sur plusieurs benchmarks. Une amélioration certes incrémentale mais d'une ampleur inédite en NLP. Qui plus est, cette méthode s'est avérée extrêmement économique quant au volume des données à utiliser pour entraîner le modèle, exigeant jusqu'à **100 fois moins de données** que les méthodes antérieures.

Les deux ingrédients principaux de cette avancée, rappelons-le, sont un préentraînement sur un modèle de langage (LM) et l'utilisation d'une forme élaborée de Transfer Learning qui évite le double écueil du **surapprentissage** d'une part et, d'autre part, de ce qu'on appelle l'**oubli catastrophique**, à savoir la destruction lors de l'entraînement sur une tâche spécifique de l'information acquise à prix fort lors du préentraînement sur un modèle générique. L'entraînement du modèle complet procède en trois phases que nous détaillons ci-dessous.

(3) Une notation graphique standardisée pour les architectures de RN, de type ULM, serait ici d'une grande utilité autant pour simplifier la présentation que le décryptage de nouvelles architectures.

Phase 1 – préentraînement d'un RNN sur le LM

La première phase consiste à entraîner le LM. Les auteurs ont utilisé pour cela un réseau de neurones récurrent (RNN ci-après pour Recurrent Neural Network) composé de trois couches de LSTM [LST] alimentés par une couche de WE comme l'illustre la figure 3. Ce réseau est doté des mécanismes de régularisation appropriés au NLP [AWD] mais n'utilise en revanche aucun mécanisme d'attention [NMT, MAL]. Il est vraisemblable qu'un LM plus sophistiqué permettrait d'améliorer encore les performances. Il est à noter que cette première phase de l'entraînement est également la coûteuse en temps de calcul. Elle n'aura cependant à être effectuée qu'une seule fois.

Phase 2 – Fine Tuning du même RNN sur la tâche cible

La deuxième phase initie le **Transfer Learning** proprement dit. On reprend pour cela le modèle préalablement entraîné sur le LM générique et l'on ajuste son entraînement en utilisant cette fois un corpus de texte propre à la tâche cible. Cette phase d'ajustements est typiquement beaucoup plus rapide que la phase précédente. De plus, elle ne nécessite généralement qu'une fraction du volume de données utilisé pour le préentraînement du LM.

Motivés par l'intuition que chacune des couches du modèle de la figure 4 encapsule un type d'information sémantique différent, les auteurs prennent soin d'ajuster séparément la vitesse d'apprentissage pour chacune d'elles. Rappelons qu'on définit ordinairement la **vitesse d'apprentissage** η comme l'amplitude de la mise à jour des paramètres $\mathbf{w}$ du NN à chaque pas d'une descente de gradient : , $\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \nabla L(\mathbf{w}^t)$, L étant la fonction de coût définie sur un mini-batch de données du LM. On aura donc en l'occurrence trois vitesses η_i différentes pour les paramètres $\mathbf{w}_i$ des couches $i = 1,2,3$ avec des mises à jours : $\mathbf{w}_i^{t+1} = \mathbf{w}_i^t - \eta_i \nabla L(\mathbf{w}_i^t)$. Les auteurs ont déterminé empiriquement qu'il est judicieux de fixer la vitesse η_3 de la dernière couche puis de la faire décroître selon $\eta_i = \eta_{i+1}/2.6$ pour $i = 2$ puis $i = 1$. En clair, il s'avère payant de procéder d'autant plus lentement dans la mise à jour des paramètres d'une couche que celle-ci est plus générique (proche de l'entrée).

Ce mécanisme est appelé le **Discriminative Fine Tuning**.

Pour favoriser la convergence rapide du modèle lors du Fine Tuning, les auteurs ont par ailleurs découvert qu'il était judicieux d'utiliser des vitesses d'apprentissage η_i^t dépendantes, non seulement de la couche i, mais également du temps t écoulé (=nombre d'itérations effectuées). L'expérience montre qu'un bon un profil temporel est celui qui est illustré dans la figure 5 où la vitesse croit rapidement durant une brève phase ascendante puis décroît lentement à mesure que l'apprentissage fin de la tâche cible s'incruste dans le modèle. Ce mécanisme est appelé le **Slanted Triangular Learning Rates**.

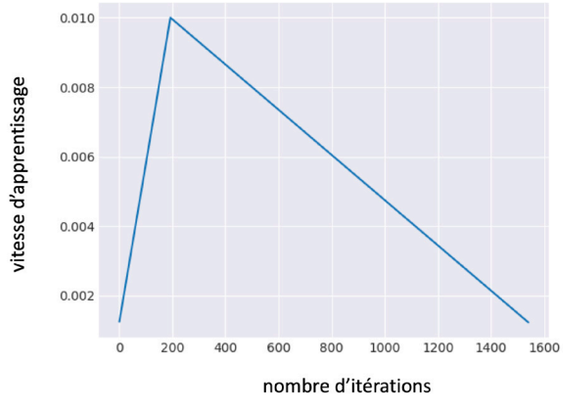


Figure 5 : Profil de la vitesse d'apprentissage en fonction du nombre d'itération t de la DGS. Une première phase d'augmentation rapide favorise la convergence. Une seconde phase de décroissance lente préserve prudemment ce qui a été acquis jusque-là.

Phase 3 – Fine Tuning du classifieur cible par dégel progressif

Le classifieur complet est construit en ajoutant au RNN de la figure 4 deux couches d'un classifieur (en bleu et rouge sur la figure 6), la première avec des activations ReLU, la seconde avec une sortie en softmax, tous deux incorporant les mécanismes de régularisation appropriés (non-représentés ici). Le dernier vecteur caché $\mathbf{h}_T$ contient en principe toute l'information calculée par le RNN. Toutefois, pour éviter de perdre de l'information qui pourrait se nicher dans les vecteurs cachés précédents $(\mathbf{h}_1, ..., \mathbf{h}_{T-1})$ on concatène à ce $\mathbf{h}_T$ deux autres vecteurs qui synthétisent l'information contenue dans

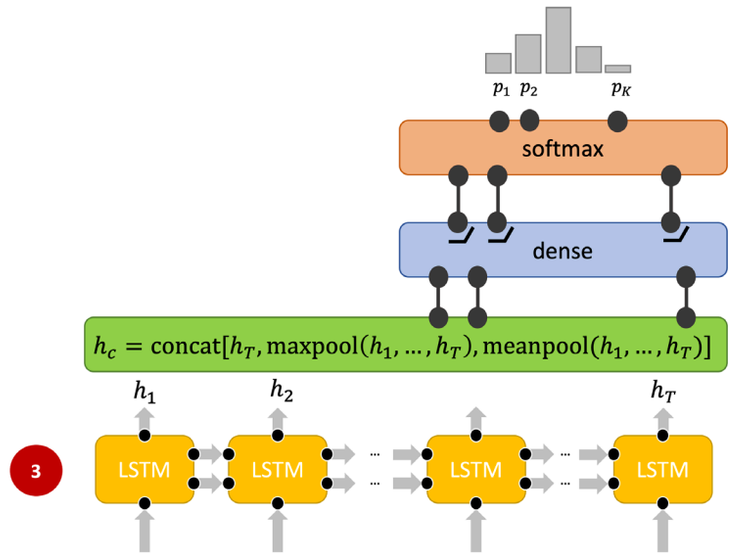


Figure 6 : Le classifieur complet rajoute deux couches supplémentaires (bleue et rouge) à la sortie de la dernière couche du RNN pour prédire une distribution de probabilité sur différentes catégories. La couche verte construit une information qui synthétise l'information contenue dans la séquence des vecteurs cachés en sortie de la dernière couche du RNN.

la suite complète $(\mathbf{h}_1, ..., \mathbf{h}_T)$. On calcule pour cela le maximum et la moyenne, composante par composante, de ces T vecteurs cachés. C'est ce que font les fonctions `maxpool` et `meanpool` dans la couche verte de la figure 6.

Le fine tuning de l'édifice complet est la partie la plus délicate de l'entraînement car il s'agit de préserver l'information engrangée par le RNN durant les deux premières phases d'apprentissage tout en affinant l'apprentissage de la tâche cible. Les concepteurs d'ULMFiT proposent pour cela de dégeler l'une après l'autres les couches du RNN, en commençant par la dernière couche (n°3) puis en procédant couche par couche vers les couches plus génériques. Après chaque dégel, le modèle complet est réentraîné en faisant une passe complète sur les données.

Ce processus de **dégel progressif** (« Chain-Thaw ») des couches de plus en plus basses est illustré dans la figure 7.

L'article [ULM] procède à une **analyse par ablation** rigoureuse, qui démontre la pertinence de combiner tous les mécanismes décrits précédemment : (1) le préentraînement

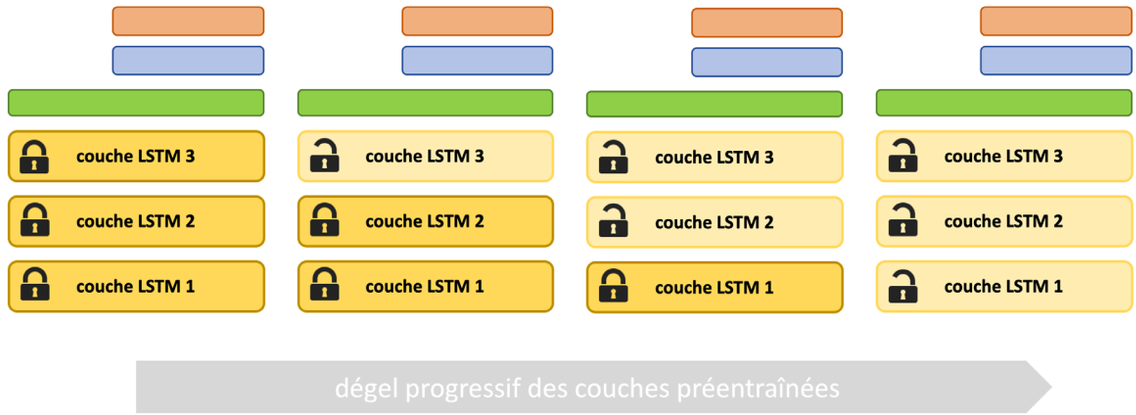


Figure 7 : Le processus de dégel progressif (Chain-Thaw) des couches du RNN pour le fine tuning de l'entraînement du classifieur complet

P.61

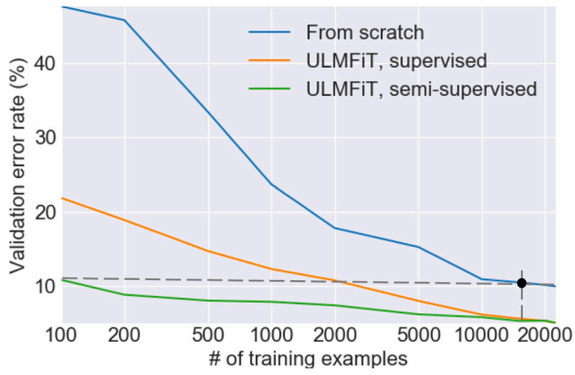


Figure 8 : Comparaison de l'évolution de l'erreur de validation en fonction du volume des données d'entraînement utilisées dans 3 situations. En bleu pour un entraînement sans ULMFiT, en orange pour un entraînement avec ULMFiT mais sans l'étape 2 de fine-tuning spécifique à la tâche cible et en vert pour la procédure ULMFiT complète – source [ULM]

sur un LM, (2) le Discriminative Fine Tuning, (3) Slanted Triangular Learning Rates et enfin (4) le Chain-Thaw.

La figure 8 illustre l'évolution de l'erreur de validation en fonction du volume de données utilisé pour l'entraînement sur un benchmark classique (IMDb). On constate que la mise en œuvre de la procédure ULMFiT complète permet d'économiser un **facteur supérieur à 100** sur le volume des données d'entraînement pour une précision équivalente.

ELMo – le transfert par extraction de features

L'approche développée dans l'article « *Deep Contextualized Word Representations* » [ELM] est conceptuellement très simple puisqu'elle propose de remplacer, dans n'importe quel modèle de NLP, les WE non contextuels usuels par des WE plus riches qui prennent en compte le contexte sémantique et syntaxique de chaque mot que l'on souhaite encoder. C'est donc une approche très générale qui s'applique sans difficultés à toutes sortes de problèmes de NLP à la seule condition que le modèle utilise des WE en entrée, ce qui est presque toujours le cas. Contrairement à ULMFiT cette méthode n'est donc pas restreinte aux seuls problèmes de classification. Elle a été testée avec succès dans des problèmes aussi variés que la déduction d'implications logiques (« **Textual Entailment** ») ou la réponse automatique à des questions (« **Question Answering** ») avec à chaque fois des gains en précision compris entre 6 et 20% sur l'erreur relative et, simultanément, une réduction du volume des données d'entraînement allant jusqu'à un facteur dix.

Là encore, on utilise un modèle de langage pour construire ces word embeddings améliorés que l'on désigne par le sigle **ELMo** pour « ***E**mbeddings from **L**anguage **Mo**dels* ». La méthode ELMo diffère toutefois de

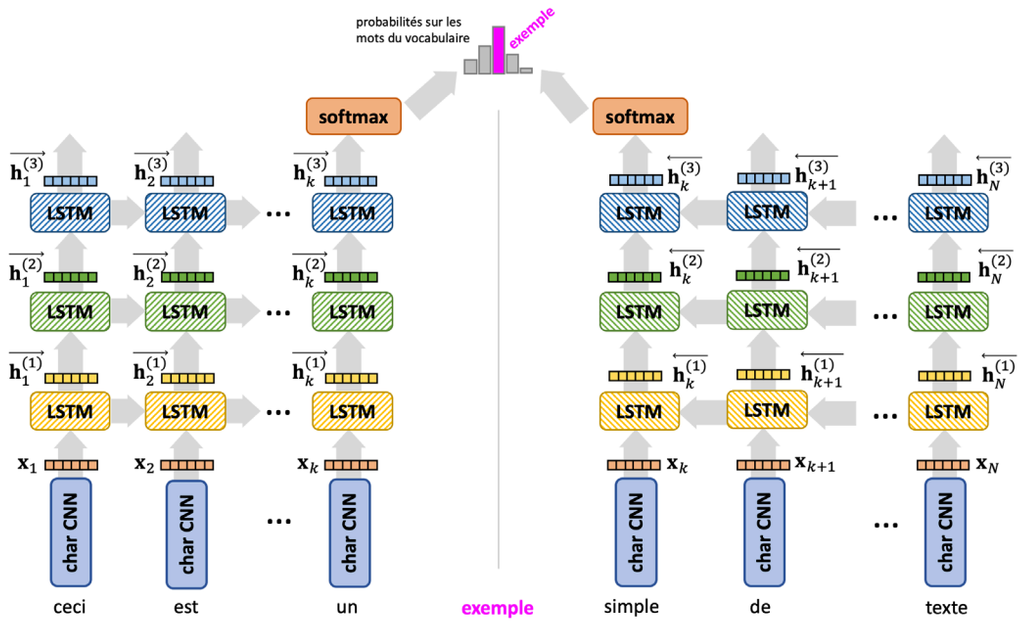


Figure 9 : Le bi-LSTM utilisé comme LM. Les LSTM des parties gauches et droites sont indépendants. Les paramètres des char-CNN et des softmax sont en revanche partagés entre les deux parties.

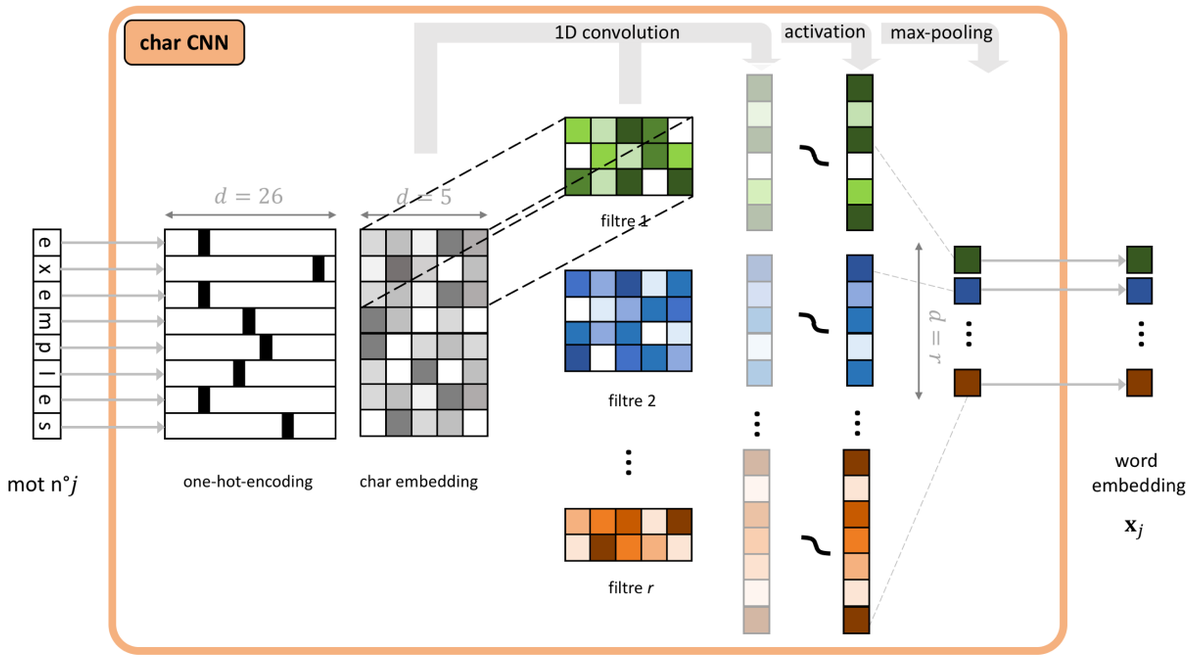


Figure 10 : Zoom sur la partie char CNN qui effectue les WE initiaux. Le nombre de filtres égale la dimension de l'embedding. Voir le texte pour la description complète.

ULMFiT sur deux points principaux. D'une part l'architecture du NN utilisé pour le LM est plus sophistiquée comme nous le verrons puisqu'elle combine les LSTM au niveau des mots et des CNN au niveau des chaînes de caractères [UCN]. D'autre part, le transfert d'apprentissage entre le LM et la tâche cible ne se fait pas par fine tuning du LM mais plutôt par extraction explicite de nouvelles features qui seront précisément les ELMo. Détaillons ces deux points.

Une architecture de LM qui combine LSTM et CNN

Une image valant mille mots, nous nous appuierons sur les figures 9 et 10 pour décrire les principaux éléments de l'architecture du LM.

La figure 9 illustre l'architecture (simplifiée) utilisée pour apprendre le LM. Le modèle est constitué de deux parties associées aux deux directions de traitement possibles. Les paramètres des couches d'embedding char-CNN (décrites plus loin) d'une part et les paramètres des couches softmax d'autre part sont **identiques** pour les deux parties. Les couches de LSTM en revanche sont **indépendantes** l'une de l'autre, ce qui est symbolisé par les hachures dans des directions différentes. Cette structure est appelée un **bi-LSTM**. L'objectif du modèle est de prédire un mot en fonction à la fois des termes qui le précèdent *et* de ceux qui le suivent. La fonction de coût est une simple moyenne des fonctions de coût calculées par les deux parties du bi-LSTM.

Une autre différence avec ULMFiT est que l'on utilise des **réseaux de convolution à une dimension basés sur les caractères** des mots (1D char-CNN) pour réaliser les WE $\mathbf{x}_j$ en entrée de la première couche de LSTM. Le principe des 1D char-CNN appliqués au WE est illustré sur la figure 10 [ELL, UCN]. Les tokens étant ici les caractères d'un mot, on commence par associer chaque caractère à un vecteur (char embedding) de faible dimension (qui sera appris durant l'entraînement). Chaque mot est donc représenté par une matrice comportant autant de lignes que le mot possède de caractères (en gris dans la figure 10). Pour créer des embeddings $\mathbf{x}_j$ de mots de dimension r, on construit autant de filtres (dans le sens CNN du terme [MAL]) dont la largeur (5 dans la figure 10) est égale à la dimension des embeddings. Le résultat du produit de convolution de chacun de ces r filtres s'obtient en faisant glisser verticalement le filtre sur la matrice

P.63

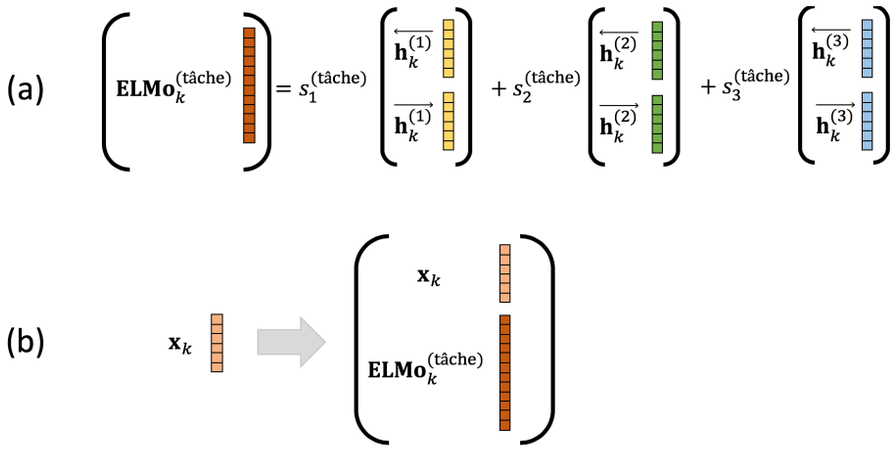


Figure 11 : (a) La fabrication des embeddings boostés $\mathbf{ELMo}_k^{(\text{tâche})}$ par combinaison linéaire des vecteurs cachés associés au mot n° k. (b) Dans le modèle cible, on remplace les embeddings usuels $\mathbf{x}_k$ par la concaténation de chaque $\mathbf{x}_k$ avec les $\mathbf{ELMo}_k^{(\text{tâche})}$, les coefficients $s_j^{(\text{tâche})}$ sont appris sur la tâche cible.

d'embeddings et en calculant la somme des produits des composantes correspondantes. On obtient ainsi r vecteurs de longueurs variables que l'on fait passer par une fonction d'activation sigmoïde usuelle. Le vecteur d'embedding du mot souhaité s'obtient alors par une opération de max-pooling qui sélectionne la composante maximale de chacun de ces r vecteurs.

L'atout principal des 1D char-CNN par rapport aux embeddings classiques est qu'ils permettent d'**encoder n'importe quel mot** contrairement aux WE usuels qui sont associés à un vocabulaire figé [ELL].

Par ailleurs, les 1D char-CNN se révèlent particulièrement efficaces pour des **langues morphologiquement riches** ou lorsqu'un texte retranscrit une **conversation informelle** [ELL].

Extraction des features ELMo

Une fois que le LM a été entraîné sur un grand corpus de texte, on fige ses paramètres. Pour l'utiliser comme un **encodeur contextuel** de mots, on procède de la manière suivante. Si le contexte du mot à encoder est constitué par une phrase, on injecte cette phrase en entrée du bi-LSTM de la figure 9. Si le mot est en position k dans la phrase, on récupère d'une part l'embedding classique $\mathbf{x}_k$ calculé en cette position par le 1D char-CNN et, d'autre part, tous les vecteurs cachés $\mathbf{h}_k^{(1)}, \mathbf{h}_k^{(2)}, \ldots$ situés à cette même position au-dessus des LSTM (voir la figure 9). Ces vecteurs encapsulent une information précieuse car contextuelle à la phrase, grâce, à la double récurrence du bi-LSTM. Les $\mathbf{h}_k^{(j)}$ des couches basses encodent typiquement une information **syntaxique** alors que les $\mathbf{h}_k^{(j)}$ des couches supérieures encodent une information **sémantique**.

On construit ensuite des variables $\mathbf{ELMo}_k^{(\text{tâche})}$ par combinaison concaténation et combinaison linéaire de ces vecteurs cachés, comme l'illustre la figure 11 (a).

Les coefficients $s_j^{(\text{tâche})}$ ne sont pas fixés par avance mais sont appris lors de l'entraînement du modèle sur la tâche cible. Dans le modèle cible, on remplace simplement les embeddings usuels $\mathbf{x}_k$ par ces mêmes embeddings concaténés au embeddings boostés, $\mathbf{ELMo}_k^{(\text{tâche})}$ comme l'illustre la figure 11 (b).

La combinaison linéaire des vecteurs cachés $\mathbf{h}_k^{(j)}$ détruit une part de l'information mais l'expérience montre que l'information reste suffisante. Une combinaison linéaire a par ailleurs l'avantage d'avoir une dimension indépendante du nombre de couches.

Dans certains cas, un fine-tuning du LM sur la tâche spécifique améliorera encore les scores. Au final, c'est l'observation de l'amélioration des scores sur des applications de NLP concrètes qui démontre que les variables $\mathbf{ELMo}_k^{(\text{tâche})}$ incorporent une information contextuelle inaccessible aux WE usuels.

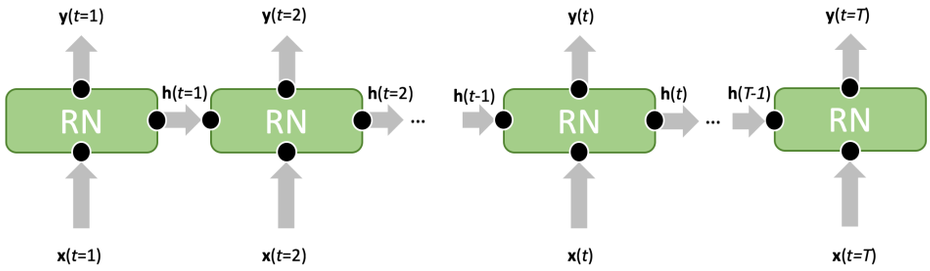


Figure 12 : Le caractère séquentiel d'un RNN empêche la parallélisation des calculs.

BERT – le meilleur pour la fin !

Début octobre 2018, l'article « *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* » [BER] défraie la chronique [BME] dans la communauté du Deep Learning – NLP. Une fois de plus, les équipes de R&D de *Google* donnent le "la" en exploitant une architecture de NN récente de leur cru, le Transformer [AYN]. Les performances de BERT définissent un nouvel état de l'art pour 11 problèmes de NLP et ceci sans exiger aucun ajustement spécifique à une tâche particulière ! Sur le problème difficile du « Question Answering », BERT dépasse même les humains !

Il s'agit là encore d'une variante du Transfer Learning. Le mode principal de fonctionnement de BERT correspond à un transfert par fine-tuning similaire à celui qu'exploite ULMFiT. C'est celui que nous présenterons rapidement, mais il est aussi utilisable en mode transfert par extraction de features comme ELMo.

Schématiquement BERT (***B**idirectional **E**ncoder **R**epresentations from **T**ransformers*) se base sur deux innovations que nous présenterons succinctement ci-dessous :

1. L'utilisation de l'architecture **Transformer** [AYN], qui constitue une alternative récente et performante aux RNN pour réaliser **un préentraînement bidirectionnel profond**, des termes sur lesquels nous reviendrons. C'est l'innovation principale de BERT.

2. L'utilisation de **deux nouvelles tâches pour le préentraînement**, l'une au niveau des mots et l'autre au niveau des phrases.

Pour être juste, mentionnons qu'une équipe d'OpenAI avait aussi proposé quelques mois avant BERT un modèle très performant [OAT] qui s'appuyait également sur le Transformer et ne nécessitait aucun ajustement spécifique.

L'architecture Transformer et l'entraînement bidirectionnel

L'architecture Transformer a été publiée fin 2017 dans un article au titre accrocheur : « *Attention is All You Need !* » [AYN]. A terme, cette architecture a vocation à remplacer les RNN comme encodeurs de séquences. La motivation des concepteurs du Transformer était de pallier un des principaux inconvénients des RNN, à savoir l'impossibilité de paralléliser leurs traitements en raison du caractère séquentiel des calculs qu'ils effectuent. Chaque cellule d'un RNN a en effet besoin d'accéder au résultat calculé par la cellule précédente, voir la figure 12.

Les auteurs ont élaboré une architecture qui tire astucieusement parti du **mécanisme d'attention** [NMT] (d'où le titre) auquel nous avions consacré un précédent article [MAL]. Rappelons que dans le contexte de la traduction automatique ce mécanisme permet, par exemple, d'incorporer des éléments contextuels à l'encodage d'un mot. Pour des phrases comme :

- « *The animal did not cross the **street** because **it** was wide.* » ou
- « *The **animal** did not cross the street because **it** was tired.* »,

il est crucial d'identifier correctement à quel mot se réfère le pronom « *it* » (« animal » ou « street ») pour traduire correctement la phrase en français (« … il était trop fatigué » ou « … elle

P.65

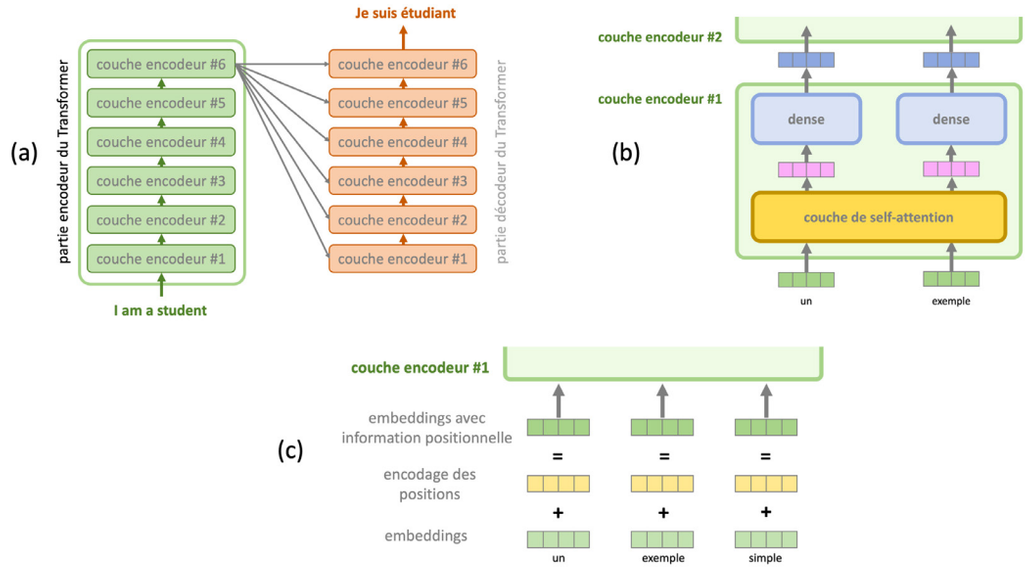


Figure 13 : Les principaux éléments de l'architecture du Transformer. (a) Un encodeur, formé d'un empilement de couches de petits encodeurs, convertit une suite de WE en une suite de vecteurs qui incorporent des informations contextuelles riches pour chaque token. (b) Un zoom sur une couche d'encodeur, voir le texte. (c) Les WE utilisés par le Transformer ajoutent aux WE initiaux une information de position.

était trop large »). Le mécanisme d'attention [NMT, MAL] permet, entre autres, de lever ce genre d'ambiguïté en incorporant dans la représentation numérique du mot « *it* » une information qui indique sa référence à un autre mot, une capacité acquise par entraînement statistique comme pour tout modèle de ML.

Une présentation détaillée, illustrée et pédagogique du Transformer est disponible dans cet excellent article [ITR]. Nous nous limiterons ici à la description très schématique illustrée dans la figure 13.

A l'échelle macroscopique un Transformer fonctionne comme un **encodeur-décodeur** [MAL] utilisé dans un système de traduction automatique, voir la figure 13 (a). Il convertit une suite de WE (ou de token) en une autre suite, de longueur différente, de vecteurs. L'**encodeur** empile plusieurs petits encodeurs (six dans la figure 13) qui sont autant de couches d'abstraction et de même pour le **décodeur** qui empile plusieurs petits décodeurs. Seul l'encodeur du Transformer est utilisé en phase de préentraînement par BERT.

La figure 13 (b) fait un zoom sur un des petits encodeurs. Après entraînement, la couche de self-attention détecte les **relations sémantiques** qui existent entre différents vecteurs qu'elle reçoit en entrée (représentés en vert) et incorpore cette information dans les vecteurs qu'elle produit en sortie (représentés en rose). Ce mécanisme de self-attention intervient à chaque niveau d'abstraction de l'encodeur et du décodeur. Il intervient également entre l'encodeur et le décodeur mais ceci ne nous concerne pas pour BERT. Chaque vecteur en sortie de la couche de self-attention alimente un (même !) simple NN dense, qui ajoute une couche d'abstraction à l'encodeur. Chaque instance de ce NN est indépendante des autres, si bien qu'il est dorénavant trivial de paralléliser ces traitements.

Une difficulté à surmonter est qu'il n'existe pas de notion naturelle d'ordre des tokens en entrée d'un Transformer, chaque token étant équivalent aux autres par contraste avec un RNN qui définit explicitement un tel ordre. Il faudra par conséquent injecter explicitement l'information de position de chaque token dans son embedding, faute de quoi le Transformer percevrait une phrase comme une simple collection désordonnée de mots (un « bag of words »). C'est ce que représente la figure 13 (c). L'encodeur du Transformer ne privilégie cependant aucune des deux directions de lecture d'une suite, en ce sens il

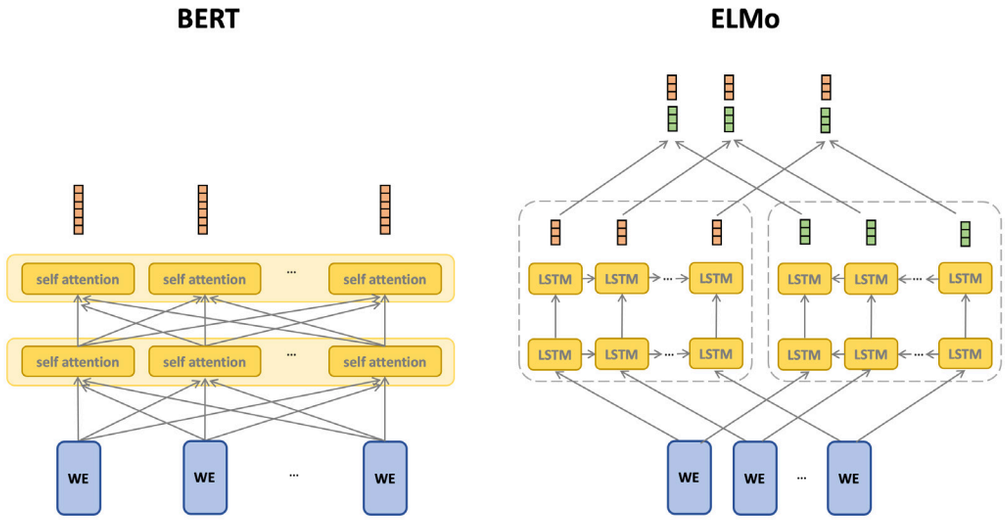


Figure 14 : Comparaison entre l'architecture BERT réellement bidirectionnelle et l'architecture ELMo qui est une concaténation deux RNN unidirectionnels.

est **intrinsèquement bidirectionnel**.

ELMo était aussi bidirectionnel mais d'une manière plus superficielle, puisqu'il était constitué de deux RNN unidirectionnels dont chacun calculait la moyenne des prédictions. La figure 14 illustre cette différence entre ELMo et BERT.

Les deux nouvelles tâches pour le préentraînement

Les LM utilisés jusqu'ici sont des modèles unidirectionnels même si ELMo procède à une moyenne des deux directions. Dans BERT, le Transformer est préentraîné simultanément sur deux tâches NLP qui, d'une part, ne nécessitent aucun travail de supervision manuel et, d'autre part, ne privilégient aucune direction des deux directions de lectures et mettent donc à profit le caractère bidirectionnel du modèle.

La première de ces tâches consiste à entraîner le Transformer à prédire un certain nombre de termes (15% dans [BER]) masqués et choisis aléatoirement dans le corpus source (**Masked Language Model**). Comme pour le LM, cette tâche force le modèle à encapsuler une part importante de NLP, aussi bien sur des aspects de syntaxe que de sémantique. Pour réaliser un tel modèle, il suffit d'ajouter un softmax qui convertit le vecteur caché de la dernière couche du Transformer en une probabilité sur les mots du vocabulaire.

La deuxième tâche a vocation à apprendre à cerner les **relations sémantiques entre deux phrases**, une aptitude essentielle par exemple pour évaluer le degré de proximité sémantique entre deux phrases, pour identifier des relations logiques ou pour à répondre à des questions. On crée à cet effet un ensemble d'entraînement formé de couples de phrases dont 50% sont effectivement consécutives et 50% sont choisies au hasard dans le corpus.

Le Transformer est entraîné simultanément sur les deux tâches avec une fonction de coût égale à la somme des fonctions de coût associées aux deux tâches.

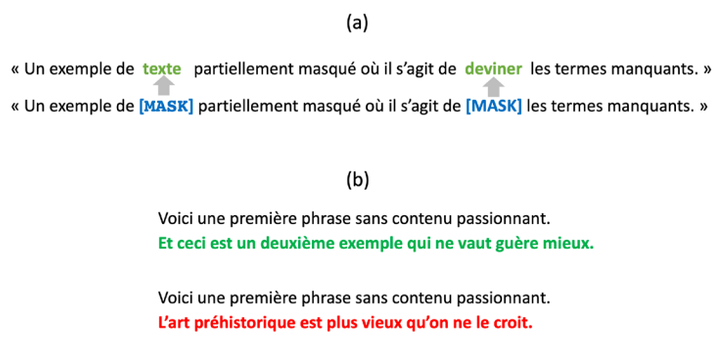


Figure 15 : Les deux tâches utilisées dans BERT pour le préentraînement du Transformer. La première consiste à prédire des probabilités de mots masqués. La seconde consiste à déterminer si deux phrases sont consécutives.

P.67

4. Le début d'une longue marche !

Les progrès en NLP décrit dans cet article marquent assurément un tournant. 2018 restera l'année où l'idée du Transfer Learning [AST] aura été concrétisée dans le domaine du NLP, comme elle l'avait été quelques années plutôt en vision artificielle.

> *L'information encapsulée dans un seul modèle de NLP, entraîné au prix fort sur une tâche générique suffisamment complexe, peut être mise à profit pour entraîner économiquement des modèles différents sur des tâches spécifiques.*

Des implémentations de référence de certains de ces modèles existent d'ores et déjà dans les librairies de Deep Learning comme *TensorFlow* et *PyTorch*.

Sur le plan de l'architecture des NN, ces avancées en NLP reposent aujourd'hui sur quelques intuitions. Le mécanisme d'attention ou la possibilité d'apprendre des structures complexes comme des modèles de langage par empilement de couches d'abstraction sont deux exemples. Bien sûr, ces nouveaux modèles exploitent par ailleurs toute la panoplie de trucs et d'astuces (**bag of tricks**) développés ces dix dernières années pour le Deep Learning en général. Sans oublier pour ce qui est du matériel les GPU désormais incontournables.

A ce jour, le NLP tout comme le Deep Learning en général, reste une discipline très empirique. Les études d'ablations menées dans [ULM, BER], qui cherchent à identifier sans ambiguïté les éléments déterminants des performances d'un modèle, sont en pointe sur cette démarche qui, pour être expérimentale, se veut désormais plus rigoureuse.

Les **fondements théoriques** qui expliqueraient pourquoi ces modèles sont si performants font pour l'instant défaut. Disposer de résultats d'optimalité qui indiqueraient quelles sont les performances optimales que l'on peut espérer d'un algorithme, serait pourtant très utile. Pour guider la conception d'architectures de NN performantes, pour la construction de jeux de données denses (riches et compacts) ou, plus simplement, pour éviter des explorations inutiles et fastidieuses en pointant des voies sans issues[4].

Terminons par une question fascinante que soulève le NLP. Est-il possible d'apprendre une langue entièrement à partir d'un corpus de texte comme prétendent le faire les modèles de contemporains ? Certains chercheurs comme Y. Bengio, les des pères fondateurs du Deep Learning, en doute sérieusement et préconisent d'expérimenter dès à présent une approche dite de **Grounded Language Learning** qui consiste à apprendre un langage par **expérience sensorielle** plutôt que par la seule lecture. L'intuition commune nous suggère qu'un immense réservoir de bon sens, celui qui nous permet de nous mouvoir dans un monde physique et social complexe, n'est disponible dans aucun corpus de textes, aussi vaste soit-il. Par ailleurs, un argument simple tiré de la théorie de l'information [YBC, HTC] démontre que certains langages[5] sont rigoureusement « innaprenables », bien qu'ils puissent encoder une sémantique capable de décrire tout un univers.

La stratégie poursuivie par l'équipe de Y. Bengio consiste à initier un tel « apprentissage linguistique incarné » dans des mondes virtuels simplifiés à l'extrême, où l'apprentissage sera rapide, pour le poursuivre dans un second temps dans le monde réel.

> *En conséquence, il n'est pas impensable que la recherche en NLP redécouvre un jour sur ce vieux principe de sagesse et d'éducation qui nous rappelle qu'on en apprendra toujours davantage de la vie que d'une lecture avide de tous les livres du monde.*

(4) Dans le cadre du Deep Learning en général les travaux de chercheurs comme S. Mallat au Collège de France ou de N. Tishby [OBB] de l'université de Jérusalem sont des tentatives prometteuses.

(5) Si l'on procède à un codage optimal d'un message, avec un codage de Huffman par exemple, le message encodé a l'air d'être un message strictement aléatoire et ne peut donc être appris.

Références

- [ULM] **Universal Language Model Fine-tuning for Text Classification**, *J. Howard, S. Ruder* – mai 2018, arXiv:1801.06146 [cs.CL]
- [ELM] **Deep contextualized word representations**, *M. E. Peters, M. Neumann, M.Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer*, arXiv:1802.05365 [cs.CL] – mai 2018
- [BER] **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**, *J. Devlin, M.-W. Chang, K. L., K. Toutanova*, arXiv:1810.04805 [cs.CL] – octobre 2018
- [BME] **Best NLP Model Ever? Google BERT Sets New Standards in 11 Language Tasks**, *Tony Peng*, Medium – octrobre 2018
- [OAT] **Improving Language Understanding by Generative Pre-Training**, *A. Radford, K. Narasimhan, T. Salimans, I. Sutskever*, OpenAI blog post – juin 2018
- [AYN] **Attention Is All You Need**, *A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin*, arXiv:1706.03762 [cs.CL]– décembre 2018,
- [LST] **Understanding LSTM Networks**, *C. Olah*, blog post – août 2015
- [UCN] **Understanding Convolutional Neural Networks for NLP**, *D. Britz*, WildML blog post – novembre 2015
- [ITR] **The Illustrated Transformer**, *J. Alammar*, blog post – juin 2018
- [ELL] **Exploring the Limits of Language Modeling**, *R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, Y. Wu* – février 2016, arXiv:1602.02410 [cs.CL]
- [AWD] **What makes the AWD-LSTM great?**, *Y. Seth*, blog post– septembre 2018
- [YBC] **Challenges for deep learning toward AI**, *Yoshua Bengio*, vidéo YouTube France is AI – octobre 2018
- [HTC] **How to Teach Artificial Intelligence some Common Sense**, *Clive Thompson*, article Wired – novembre 2018
- [NMT] **Neural Machine Translation by Jointly Learning to Align and Translate**, *D. Bahdanau, K. Cho, Y. Bengio*, arXiv:1409.0473 [cs.CL] – mai 2016
- [AST] **A Survey of Transfert Learning**, *Sinno Jialin Pan, Qiang Yang*, IEEE Transactions on Knowledge and Data Engineering, Volume 22, Issue 10 – octobre 2010

P.69

- [OBB] **Opening the Black Box of Deep Neural Networks via Information**, *R. Shwartz-Ziv, N. Tishby*, arXiv:1703.00810 [cs.LG] – avril 2017

- [CHL] **Chatbots – entre rêves et réalité**, *P. Lemberger*, IA Lab weave – avril 2017

- [MAL] **Le mécanisme d'attention en IA**, *P. Lemberger*, IA Lab weave – janvier 2018


Calcul distribué
Confidentialité
Synchronisation
Moyennes de modèles
Apprentissage fédéré
Compromis CPU réseaux
Descente de gradient stochastique

L'intelligence artificielle distribuée

Nicolas Parent – **weave Business Technology**

Résumé

L'IA distribuée se base sur la notion d'apprentissage fédéré pour permettre l'entraînement d'un modèle de Deep Learning sur les terminaux du grand public, de type smartphone ou ordinateur portable. Cette approche permet un accès direct et respectueux aux données des utilisateurs, tout en mettant à contribution les puissances de calcul souvent peu utilisées de leurs terminaux. Il s'agit cependant d'une approche entièrement empirique, dont la mise en œuvre repose sur un compromis entre puissance de calcul et contraintes réseau

1. Confidentialité et économie de la donnée

La naissance de l'informatique au XXe siècle a permis d'imaginer le concept d'Intelligence Artificielle, des systèmes capables de simuler une intelligence propre. Dès 1950, *Turing* imagine son fameux test, ayant pour but de juger de la capacité d'une machine à imiter un humain. Le Test de Turing a l'avantage de ne pas chercher à décrire une IA, car il ne fait que constater un résultat. Cette approche s'est révélée pertinente : quand on observe les tentatives de simuler l'intelligence humaine depuis cette époque, un constat frappant est la diversité des techniques utilisées. L'IA de *DeepBlue* possède une structure fondamentalement différente de celle d'*AlphaGo Zero*, liée aux progrès des 20 dernières années en intelligence artificielle. En particulier, le Deep Learning, né au début du XXIe siècle, a donné une nouvelle impulsion au concept d'Intelligence Artificielle. Ses concepts ont été adaptés pour créer de nouveaux outils très puissants, utilisables pour de nombreux usages : des assistants vocaux au jeu de Go. Le Deep Learning (et de manière plus large, les technologies du Big Data) est cependant difficile à mettre en place, car il fait apparaître quatre paradoxes.

La puissance de calcul

Un système de Deep Learning requiert des quantités très importantes de puissance de calcul, y compris selon les standards actuels. De tels systèmes nécessitent aujourd'hui des serveurs très puissants (ou un coût élevé si on utilise un Cloud public), ce qui constitue un premier frein à la mise en place d'un projet de Deep Learning.

Pourtant, deux phénomènes viennent challenger ce constat : la loi de Moore et la démocratisation des appareils de calcul. La croissance exponentielle des capacités informatiques prédite par Moore en 1965 s'est montrée correcte jusqu'ici, ce qui a changé l'échelle des capacités informatiques. Par exemple, l'ensemble du programme Apollo 11 a nécessité la puissance de calcul utilisée aujourd'hui par *Google* pour une simple recherche, tandis qu'un smartphone actuel possède la même puissance de calcul que *DeepBlue*, l'ordinateur qui a battu Gary Kasparov aux échecs en 1997 (D'où un détail amusant : si vous gagnez une partie d'échecs contre votre smartphone, il vous a laissé gagner). D'un autre côté, les appareils à forte puissance de calcul ont essaimé. On estime qu'en 2018, il y a autant de smartphones en état de marche que d'humains sur Terre [VMS]. Ces appareils

sont pourtant peu sollicités : nous lisons nos mails et commandons notre nourriture sur les supercalculateurs d'il y a 20 ans.

Ces milliards d'appareils très puissants, utilisés à une fraction de leurs capacités, montrent un premier paradoxe souligné par le Deep Learning : pourquoi investir dans des serveurs surpuissants mais uniques, quand **un océan de puissance de calcul reste inutilisé** par le grand public ?

L'accès aux données

Il existe un constat similaire à celui de la puissance de calcul pour la collecte de donnée. Le Deep Learning repose sur un apprentissage qui exploite de grandes quantités de données, et ces données sont difficiles à récupérer en quantité suffisantes pour des acteurs d'une échelle inférieure à celle d'une grosse entreprise ou celle d'une administration publique.

Pourtant, la production de données par l'humanité semble elle aussi suivre une loi exponentielle depuis les années 60. Cette croissance des données produites, démarrée par l'apparition des contenus "globaux" (Télévision, radio, journaux), a été démultipliée avec la démocratisation d'Internet et l'arrivée des réseaux sociaux. On estime qu'en 2018, nous produisons tous les deux jours autant de données que du début de l'humanité à 2003 [ESC]. Cette explosion de la production de données du XXIe siècle a cependant privilégié des acteurs tels que les GAFAM, NATU, BATX, qui ont la masse critique pour que leurs services attirent un grand nombre d'utilisateurs, permettant de récupérer et valoriser les données. Cela leur a conféré un oligopole sur la donnée personnelle, leur première source de valeur, d'où les difficultés d'accès pour des acteurs plus petits.

Cette quantité de donnée toujours plus importante, mais aussi plus inaccessible, marque un second paradoxe lié au le Deep Learning : si la production de donnée croît de manière aussi importante, pourquoi l'accès à des larges volumes de données est-il aussi complexe pour un particulier ?

La confidentialité

L'une des réponses concernant la difficulté d'accès aux données est la confidentialité. La **protection de la vie privée**, qui passe par la confidentialité des données, est un impératif absolu qui relève des libertés fondamentales.

Pourtant, chaque mois une nouvelle fuite de données est annoncée. *Google+*, *Facebook*, *LinkedIn* (*Microsoft*), *Amazon*... La donnée est difficile à protéger, et c'est désormais un enjeu majeur pour le GAFAM. Du côté des états, il y a eu peu de fuite de données pour l'instant, et on observe chez certains une volonté de protéger les utilisateurs : la CNIL et le RGPD constituent en cela des exemples d'une volonté politique sur ces sujets. Les citoyens restent eux peu engagés sur le sujet de la confidentialité des données. Aucun réseau n'a pour l'instant été boycotté pour une fuite de données et le RGPD, texte européen, n'est pas un sujet pour les élections européennes de 2018. La confidentialité des données est parfois considérée par l'opinion publique comme un enjeu de confort, et non une nécessité absolue. L'exemple de *Cambridge Analytica* [DSM] n'a pas laissé d'impact profond dans l'opinion publique, alors que cette fuite de donnée a eu une influence majeure dans l'élection de Donald Trump, et donc sur la géopolitique mondiale. Il est difficile de mesurer précisément l'impact de *Cambridge Analytica* sur la campagne de 2016, mais le fait que Donald Trump ait remporté la présidentielle américaine de 2016 avec la moitié du budget de campagne d'Hillary Clinton est révélateur d'une certaine efficacité. Cette efficacité est doublée d'un soupçon de cynisme : la frontière entre publicité ciblée et manipulation est poreuse.

La confidentialité est un sujet de société qui va bien au-delà du Deep Learning, qui n'agit qu'en simple "révélateur" de l'importance des données. On peut malgré tout se demander si les fuite de données à répétition ne finiront pas par abimer nos libertés, si des modèles alternatifs de collecte de données ne sont pas envisageables.

L'économie de la donnée

La notion d'économie de la donnée revient souvent dans les médias, de même que dans les rapports d'entreprises ou ceux émis par des états. Il est intéressant de remarquer que cette économie de la donnée est aujourd'hui une économie de bénévoles.

Le premier producteur de données est le grand public. Pourtant, il tire peu d'avantages de cette production (à moins de considérer que les publicités ciblées forment un acquis essentiel...). Les données personnelles sont des objets auxquels il est difficile de donner une valeur, mais certaines approches permettent de se faire une idée : le rachat de *LinkedIn* par *Microsoft*, étant donné le montant et le nombre d'utilisateurs, permet d'**estimer la valeur de votre CV à 60€**, et le même exercice est faisable avec tous les réseaux sociaux cotés en bourse, en gardant à l'esprit qu'il s'agit d'une approche très approximative. Mais cela nous rappelle que l'économie de la donnée se base sur une asymétrie entre producteurs et agrégateurs de données. Un équilibrage paraît peu probable, bien que des exemples existent : la chaîne japonaise *Shiru Café* permet aux étudiants du Japon, d'Inde et des Etats-Unis, de récupérer un café "gratuitement" s'ils fournissent des données personnelles [LBD]. Ces données sont ensuite valorisées par des entreprises sponsor qui chercheront à recruter ces étudiants. Un tel système paraît aujourd'hui entièrement vertueux, mais les problèmes d'éthique ne sont jamais loin : toute donnée personnelle, y compris la plus intime, a-t-elle un prix ?

Au-delà des problématiques éthiques qui ne manqueront pas de surgir, on se trouve face un paradoxe sur cette nouvelle économie de la donnée : le seul acteur indispensable, le grand public, ne récupère que peu de valeur de ces nouveaux modèles.

2. Le fonctionnement de l'IA distribuée

Le modèle d'IA distribuée que nous esquissons ci-dessous est décrit en détail dans cet article [CEL]. Ce modèle repose sur une approche empirique, dont les fondements théoriques restent à établir mais il apporte des éléments de réponse aux paradoxes évoqués au paragraphe précédent.

Le principe

Ce que nous appelons une "IA distribuée" n'est rien d'autre qu'un modèle de Deep Learning dont l'entraînement supervisé (l'optimisation itérative des paramètres) est réalisé sur une multitude d'appareils distincts. Ces différents appareils ne communiquent jamais entre eux leurs données d'apprentissage respectives, mais uniquement les modèles qu'ils élaborent de manière itérative. Les itérations de chacun de ces appareils sont ensuite rassemblées et mise en cohérence par un serveur central : l'**apprentissage est dit "fédéré".**

La distribution de l'entraînement

L'apprentissage supervisé cherche à optimiser un modèle de Deep Learning, dont les paramètres sont assimilables à un vecteur $\mathbf{w}$ de grande dimension. Les données d'entraînement sont un ensemble de n observations, composées de variables prédictives et de valeur cible, notées respectivement $\mathbf{x}_i$ et y_i. Pour chaque observation, on va créer une fonction de coût liée à l'observation, qui prend en paramètre un modèle $\mathbf{w}$, qui mesure la différence entre la valeur cible y_i et valeur prédite par le modèle à partir de la variable prédictive, $\hat{y}(\mathbf{w}, \mathbf{x}_i)$. Cette fonction de coût de l'observation i est notée $l_i(\mathbf{w}) = l(\hat{y}(\mathbf{w}, \mathbf{x}_i), \mathbf{x}_i)$. En prenant une moyenne des fonctions de coût liées à chaque observation, on crée une fonction de coût globale $l(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} l_i(\mathbf{w})$, qui associe à un modèle une mesure de la différence entre prédiction et résultat attendu. L'objectif de l'apprentissage et de trouver des valeurs du modèle pour lesquelles la fonction de coût est minimisée.

Pour minimiser cette fonction de coût, on réalise une opération appelée descente de gradient. Le gradient d'une fonction l est un vecteur noté $\nabla l(\mathbf{w})$, qui indique

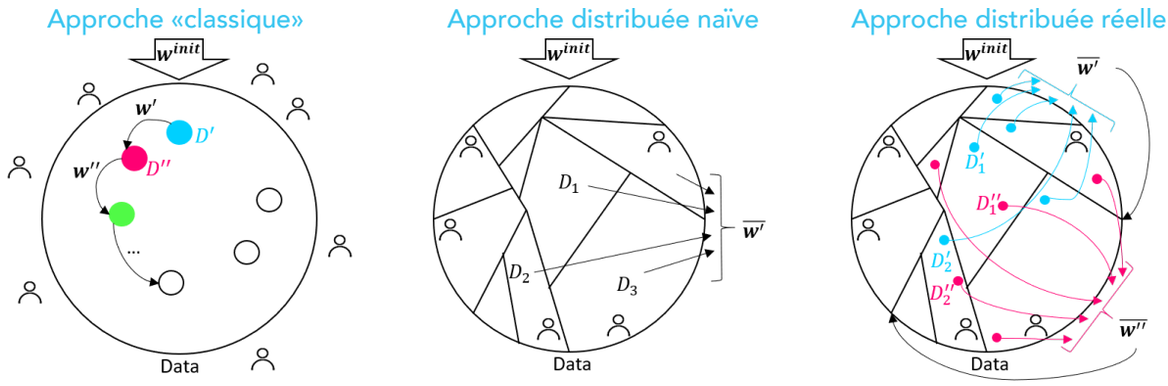


Figure 1 : Le schéma de gauche (1.1) correspond à une descente de gradient stochastique, le modèle progresse de manière séquentielle. Le schéma central (1.2) correspond à une version naïve de l'approche distribuée. Les données sont cloisonnées chez les utilisateurs, et chacun réalise une unique descente de gradient qui est ensuite moyennée. Le schéma de droite (1.3) correspond à la réalité de l'apprentissage supervisé, où les données sont cloisonnées par utilisateur, mais la descente de gradient se fait en différentes étapes.

la direction dans laquelle la fonction l augmente le plus rapidement au voisinage de $\mathbf{w}$. À partir du gradient, on peut calculer de nouveaux paramètres du modèle $\mathbf{w}' = \mathbf{w} - \eta \nabla l(\mathbf{w})$.

On appelle cette opération **descente de gradient** car on fait évoluer les paramètres $\mathbf{w}$ **du modèle** dans la direction opposée au gradient, celle qui va faire « descendre » la valeur de la fonction de coût.

Dans le cas d'un apprentissage supervisé classique, toutes ces opérations sont réalisées sur un serveur central, qui va rassembler l'ensemble des données brutes, en déduire pour chaque observation une fonction de coût prenant en paramètre un modèle, rassembler ces fonctions de coût pour ensuite appliquer des descentes de gradient. Le processus est itératif : à chaque étape, le modèle évolue légèrement grâce à une nouvelle descente de gradient.

Dans le cadre de l'apprentissage fédéré qui nous intéresse, on va reprendre ce fonctionnement en répartissant les tâches entre les appareils des utilisateurs et un serveur central. Chaque appareil contient les données personnelles de l'utilisateur qui seront utilisées pour minimiser localement une fonction de coût. Il est important de noter qu'à aucun moment l'utilisateur n'envoie ni ne reçoit de données des autres utilisateurs. Pour initier la descente de gradient, le serveur central transmet à chaque utilisateur un modèle initial. Retrouver des données personnelles à partir de ce modèle initial est impossible en pratique ce qui garantit la confidentialité de la procédure d'entraînement. Sur la base de ce modèle initial, chaque utilisateur va réaliser des descentes de gradient localement sur son appareil et ainsi améliorer progressivement sa version du modèle avec ses propres données.

L'utilisateur envoie ensuite son modèle mis à jour à un serveur central qui va réaliser une moyenne pondérée des modèles reçus, obtenant un modèle final, dont on peut espérer qu'il permettra des prédictions fiables sur l'ensemble des données de tous les utilisateurs. Les conditions qui garantissent que ce **modèle moyen** sera meilleur que les ceux issus des clients sont détaillées plus loin dans l'article.

Approche séquentielle et parallèle

Une fonction de coût est une somme de termes associés à chaque donnée de l'ensemble d'entraînement. En pratique cette somme comprend beaucoup trop de termes pour être calculée explicitement. La solution usuelle, bien connue en Deep Learning, consiste à utiliser une **descente de gradient stochastique**. On partitionne l'ensemble des données en groupe de taille modeste appelés batchs, puis on applique de manière séquentielle une descente de gradient sur chacun de ces batchs. Sous certaines hypothèses [SAM], un tel procédé permet bien de se rapprocher efficacement d'un minimum de la fonction de coût.

Pour l'approche fédérée comme pour la descente de gradient stochastique, on réalise la descente de gradient sur des sous-ensembles des données. Il s'agit pourtant d'approches différentes, qui se distinguent par le choix de ces sous-ensembles et par la méthode utilisée pour fusionner ces modèles entraînés sur ces sous-ensembles en un modèle global.

L'apprentissage supervisé classique repose sur une approche séquentielle. Étant donné un modèle initial $\mathbf{w}^{\text{init}}$, on choisit un sous-ensemble des données D', sur lequel on réalise des descentes de gradient. On obtient en résultat un nouveau modèle $\mathbf{w}'$, qu'on prendra comme base pour des descentes de gradient sur un nouveau sous-ensemble D'', résultant en un modèle $\mathbf{w}''$. Une fois parcouru tous les sous-ensembles $D', D'', \ldots$ on obtient un modèle $\mathbf{w}^*$, qui est le modèle final. (Figure 1.1)

L'apprentissage fédéré repose lui sur une approche parallèle. Etant donné un modèle initial $\mathbf{w}^{\text{init}}$, chaque utilisateur réalise des descentes de gradient sur ses propres données notées D_k. Cela signifie en particulier que le partitionnement des données est imposé par la répartition de celles-ci sur les terminaux des utilisateurs, vu qu'elles ne sont jamais mises en commun. Chaque utilisateur calcule ainsi son propre modèle $\mathbf{w}_k$, qui sera envoyé au serveur central, pour réaliser une moyenne pondérée : $\overline{\mathbf{w}} = \sum_{k=1}^{K} \frac{n_k}{n} \mathbf{w}_k$ avec n_k le nombre d'observations de D_k et n la somme des n_k. (Figure 1.2)

En réalité, la descente de gradient stochastique est aussi nécessaire pour que l'apprentissage fédéré soit utilisable. Les deux approches vont donc être mises en commun : chaque utilisateur reçoit un modèle initial $\mathbf{w}^{\text{init}}$, qu'il entraîne sur un sous-ensemble D'_k de ses données respectives D_k. Cela donne naissance aux nouveaux modèles $\mathbf{w}'_k$, qui sont envoyés au serveur central pour être moyennés en un modèle $\overline{\mathbf{w}}' = \sum_{k=1}^{K} \frac{n_k}{n} \mathbf{w}'_k$. Ce nouveau modèle est renvoyé à tous les acteurs, qui vont créer les modèles $\mathbf{w}''_k$ à partir de D''_k (Figure 1.3). Par itération, on finit par obtenir un modèle final $\overline{w^*}$.

On voit ainsi apparaître la double approche, séquentielle et parallèle, de l'apprentissage fédéré : à chaque itération de l'approche séquentielle, les différents acteurs travaillent en parallèle, avec une moyenne des modèles qui permet de clôturer l'itération et de démarrer la suivante. On appelle cela la descente de gradient fédérée, qui est une variation de la descente de gradient stochastique.

La moyenne des modèles

La moyenne des modèles est un aspect essentiel de l'apprentissage fédéré, qui paraîtra intriguant à une personne ayant étudié le Deep Learning. En effet, la réalisation de moyennes sur les données se comprend : au travers du calcul de la fonction de coût globale, on réalise une moyenne des écarts. En minimisant cette moyenne, on trouve un modèle qui, dans le cas général, permettra des meilleures prédictions. Faire une moyenne des modèles relève d'un autre procédé : prendre deux intelligences artificielles, les mélanger et espérer que le résultat sera une intelligence artificielle plus évoluée...

On peut prendre une analogie en imaginant deux livres, pris au hasard, que l'on mélange en gardant les pages impaires de l'un et les pages paires de l'autre. Tant que l'on cherche des observations simples, comme la fréquence d'apparition d'une lettre, un tel mélange n'a pas d'importance. En revanche, il est illusoire d'espérer conserver les structures plus complexes, comme le sens d'un récit.

Lorsque l'on réalise l'expérience avec des modèles de Deep Learning distincts, on confirme l'intuition que donne l'exemple des livres : les structures complexes se perdent, et ainsi le modèle obtenu en moyenne de deux modèles ne diminue pas la fonction de coût : il fait même diverger sa valeur (figure 2.1). Cependant, en étudiant de plus près de telles expériences, on se rend compte que sous certaines conditions, la moyenne des modèles permet de diminuer la fonction de coût : il faut pour cela que les modèles aient été entraînés à partir d'un même modèle initial (figure 2.2). Pour reprendre l'exemple des livres, cela revient à faire le mélange avec deux éditions proches d'un même livre : le tout restera compréhensible.

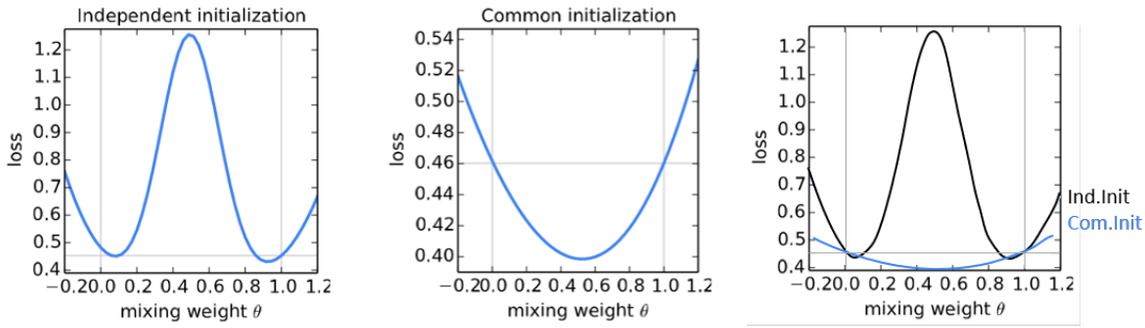


Figure 2 : Les 3 courbes représentent la valeur de la fonction de coût selon un paramètre de pondération θ qui contrôle l'influence de deux modèles que l'on souhaite moyenner. La courbe de gauche (2.1) indique que dans le cas de modèle indépendants, la valeur de la fonction de coût diverge. La courbe centrale (2.2) indique que la valeur de la fonction de coût diminue lors de la moyenne si les modèles ont été initialisés de la même manière. La courbe de droite (2.3) replace les courbes 2.1 et 2.2 à la même échelle. Source [CEL]

Cette approche par la moyenne des modèles ne se justifie pas mathématiquement, du moins pour l'instant. Il s'agit d'une approche empirique dont il convient de vérifier la légitimité expérimentalement.

Le compromis CPU - réseau

On a donc vu qu'une condition indispensable pour réaliser l'apprentissage fédéré était que les moyennes des différents modèles se fassent sur des modèles issus de modèles suffisamment proches. Cela a un impact sur la descente de gradient fédérée : **les différents acteurs doivent se synchroniser fréquemment**, afin d'éviter des écarts trop importants entre leurs modèles, ce qui provoqueraient une divergence lorsque l'on procède au calcul de la moyenne (figure 3).

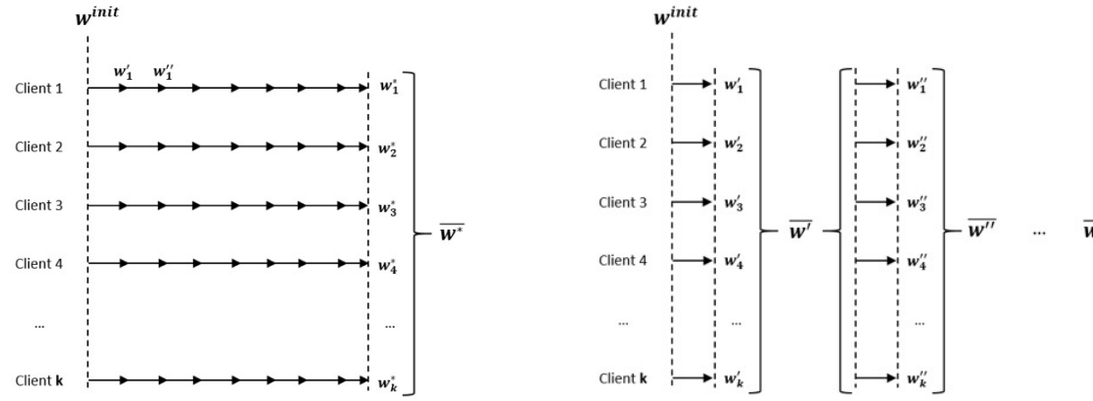


Figure 3 : Ces schémas représentent deux possibilités de synchronisation des acteurs pour l'apprentissage fédéré. Dans le cas de gauche, les clients réalisent l'ensemble des descentes de gradient sur leur appareil, et ne se synchronisent qu'à la fin. Le modèle final ne diminue pas la fonction de coût car il subit une divergence liée au calcul de moyenne de modèles trop éloignés. Dans le cas de droite, les clients se synchronise à chaque étape de la descente de gradient fédérée. Le modèle final correspond aux attentes, mais les synchronisations ont un coût réseau.

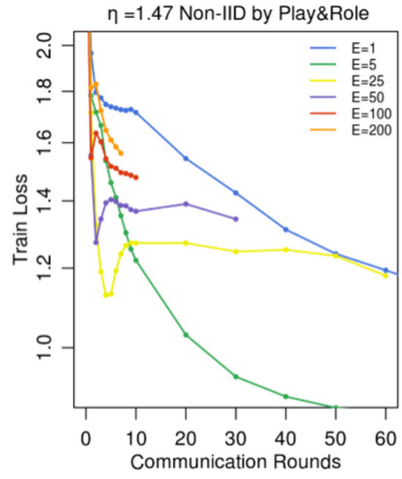


Figure 4 : Les courbes représentent la valeur de la fonction de coût en fonction du nombre de synchronisation. Le paramètre E de chaque courbe correspond à un nombre de descentes de gradient entre chaque synchronisation. Les 5 descentes de gradient entre chaque synchronisation sont dans le cas présenté le meilleur compromis CPU-réseau. Source [CEL]

Ces synchronisations fréquentes ont un impact réseau : chaque synchronisation correspond à plusieurs milliers d'appareils envoyant des Gigaoctets de données (leur modèle personnalisé) à un serveur central, et recevant de celui-ci d'autres Gigaoctets (le nouveau modèle issu de la moyenne, pour l'itération suivante). Il se crée alors un **compromis CPU-réseau** : l'apprentissage fédéré permet un accès à la puissance de calcul phénoménale des appareils du grand public, mais cet accès se fait au prix de fortes contraintes réseaux. Réaliser une synchronisation entre les acteurs à chaque itération de la descente de gradient fédérée paraît excessif, mais les bases théoriques manquent pour optimiser la fréquence des synchronisations. Là encore c'est l'approche empirique qui permet de trouver un bon équilibre comme l'illustre la figure 4.

Les courbes de la figure 4 permettent d'observer l'évolution de la fonction de coût selon la fréquence de synchronisation. On observe qu'une synchronisation à chaque descente diminue peu la fonction de coût (courbe bleue), pour un coût réseau élevé ; une fréquence de synchronisation aussi forte paraît donc excessive. À l'inverse, on observe qu'à partir de 25 itérations par synchronisation, (courbe jaune, violette orange et rouge) les divergences apparaissent et entraînent un effet de palier.

On note de plus que la qualité des modèles ne dépend pas linéairement de la puissance de calcul utilisée : pour connaître la puissance de calcul utilisée pour chaque courbe de la figure 4, il faut multiplier le nombre de synchronisation réalisées par le paramètre E de la courbe. Ainsi, l'apprentissage à 25 itérations par synchronisation de la courbe jaune a nécessité 25 fois plus de puissance de calcul que celui à une seule itération de la courbe bleue, pour un résultat identique au bout de 60 synchronisations.

Cette non-linéarité entre puissance de calcul utilisée et diminution de la fonction de coût complète la notion de compromis CPU-réseau : essayer de diminuer les coûts réseaux en ajoutant de la puissance de calcul doit se faire de manière mesurée, car une telle compensation nécessite des puissances de calcul exponentielles.

Il est essentiel de signaler que l'approche distribuée utilise une puissance de calcul sur des milliers d'appareils, puissance sans commune mesure avec un serveur centralisé. De plus les contraintes réseau ne sont plus bloquantes de nos jours. Les limitations dues au compromis CPU-réseau n'affectent donc pas la viabilité du modèle d'apprentissage fédéré. On peut cependant regretter qu'il n'existe pas de comparaison entre l'approche fédérée et l'approche classique dans la littérature.

3. Conclusion

Une comparaison théoriquement étayée entre l'approche supervisée classique et l'approche fédérée reste à établir. Cependant l'IA distribuée conçue comme solution pragmatique s'avère viable techniquement et apporte des solutions à certains des paradoxes mentionnés dans la section 1. Le point fort de cette méthode d'apprentissage est le **respect total de la confidentialité des données personnelles**. L'accès à ces données et à la puissance de calcul des terminaux d'utilisateurs sont aussi des arguments importants en faveur de cette méthode, en dépit du compromis CPU-réseau.

Le dernier paradoxe, celui de l'économie de la donnée, n'est pas abordé dans l'article [CEL]. Pourtant, l'utilisation de ressources décentralisées permet d'imaginer des modèles de rémunération pour les utilisateurs qui accepte de contribuer à l'apprentissage fédéré d'un modèle prédictif. On peut même imaginer quel les utilisateurs aient la possibilité de choisir les modèles auxquels ils souhaitent offrir un part de leur puissance de calcul.

En particulier, la mise en place d'un tel système de monétisation de ressources distribuées s'articulerait de manière idéale avec un remplacement du serveur central par une approche blockchain : la **Proof-of-Work** correspond déjà à un modèle de rémunération de la puissance de calcul. La monétisation de ressources distribuées est un cas d'usage applicable aux technologies de registre distribué, bien qu'il soit rarement envisagé comme leur intérêt principal. Cela exigerait cependant de surmonter d'importantes difficultés techniques. D'une part il s'agitait d'élaborer une Proof-of-Work adaptée à la descente de gradient (une opération beaucoup plus complexe que les sous-collisions de hash utilisées pour les Proof-of-Work du *Bitcoin* ou d'*Ethereum*). D'autre part il faudrait prendre en compte le compromis CPU-réseau pour à adapter à la notion de difficulté à ce hashrate d'un nouveau type.

Références

- [CEL] **Communication-Efficient Learning of Deep Networks from Decentralized Data**, *H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Agüera y Arcas*, arXiv:1602.05629v3 [cs.LG] – février 2017.

- [SAM] **A Stochastic Approximation Method**, *Herbert Robbins, Sutton Monro*, 1951

- [VMS] **Ventes mondiales de Smartphones**, https://www.planetoscope.com/electronique/728-ventes-mondiales-de-smartphones.html

- [ESC] **Eric Schmidt: Every 2 Days We Create as Much Information As We Did Up To 2003,** https://techcrunch.com/2010/08/04/schmidt-data/?guccounter=1

- [DSM] **Weapons of Micro Destruction: How Our 'Likes' Hijacked Democracy**, *Dave Smith*, https://towardsdatascience.com/weapons-of-micro-destruction-how-our-likes-hijacked-democracy-c9ab6fcd3d02

- [LBD] **Au Shiru Café, on règle l'addition avec ses données personnelles**, https://www.lebigdata.fr/shiru-cafe


Discursive
Processus cognitifs
Communication
AlphaGo
Empathie
Abstraction
Softskills

Soft skills humains et IA : quelle répartition des compétences ?

Fabrice Mauléon – **auteur, speaker, consultant**
Pirmin Lemberger – **weave Business Technology**

Résumé

Y-aura-t-il compétition ou complémentarité entre l'IA et l'homme ? La grande question de l'acceptabilité de l'IA par le grand public n'est-elle pas en réalité celle de la préservation de l'employabilité par les humains, désormais en concurrence avec les nouvelles compétences des machines. Quand on sait que l'opposition entre le progrès technologique et les habitudes sociales se fait toujours en faveur du premier, il est essentiel de poser la question des compétences, au regard de cette nouvelle révolution industrielle. Si la machine devient chaque jour plus autonome et réalise de plus en plus de tâches cognitives complexes, que reste-t-il alors aux humains ? Quelles compétences faudra-t-il développer pour décrocher demain un emploi ou, tout simplement, pour le conserver ?

Les précédents articles de ce recueil ont couvert bien des aspects généraux et particuliers de l'IA. Le terme, on le sait, désigne globalement les techniques qui visent à automatiser, à long terme, l'ensemble des **processus cognitifs** humains. La question des compétences y est donc intimement liée. Quels sont les fondements psychologiques et cognitifs de ces compétences ?

S'il l'immense majorité des applications de l'IA ne reposent aujourd'hui que sur quelques formes rudimentaires d'apprentissage, c'est aussi le cas de certaines tâches que nous réalisons dans un cadre professionnel.

Ainsi : je sais construire un tableau croisé dynamique avec *Excel* pour l'avoir dans un premier temps appris en formation puis, pour en avoir réalisé un certain nombre au cours de mon activité professionnelle. Ceci me permet aujourd'hui de ne plus avoir à y penser consciemment lorsque je dois en réaliser un à nouveau. Il en va de même pour mon plan marketing, pour l'écriture d'un contrat de travail ou encore pour l'analyse d'un processus d'achat.

L'apprentissage supervisé [MLD], qui présuppose la disponibilité d'exemples de référence pour lesquels on connaît la réponse souhaitée, et l'apprentissage par renforcement [ADC, AGZ], qui consiste à optimiser une stratégie par interaction avec un environnement en vue de maximiser une récompense sur le long terme, constituent les fondements de ce qu'on appelle communément l'IA. Cependant, ces processus d'apprentissage sont des formes d'intelligence qu'on pourrait qualifier d'**inconscientes** ou de **non discursives**. Ceci n'est d'ailleurs pas sans poser des questions très pratiques d'ordre juridique et éthique : comment en effet peut-on justifier d'une décision prise par une machine si celle-ci ne peut s'exprimer par un discours intelligible ? [CJD]

Pourtant, mises en œuvre entre des mains expertes, ces technologies d'apprentissage

automatique sont capables de véritables prouesses. Ainsi, tout le monde garde à l'esprit la victoire récente d'**AlphaGo** sur Lee Sedol, l'un des meilleurs experts de cette discipline [AGZ]. Voilà qu'un **jeu de stratégie**, longtemps considéré comme un véritable bastion de l'intelligence humaine venait de tomber sous les coups de boutoirs des algorithmes concoctés par *Deep Mind*.

Mais, et il est important de le noter, certaines de ces techniques d'apprentissage sous-tendent également quelques-unes de nos compétences, professionnelles et même interrelationnelles. Certains algorithmes d'apprentissage par renforcement sont d'ailleurs inspirés pour partie par la psychologie et par les méthodes d'apprentissage dans le monde animal qu'étudient les **sciences cognitives** [RLI].

Dès lors se pose une question simple qui devrait passionner voire obséder tous les professionnels actifs. Si un grand nombre de compétences peuvent effectivement s'apprendre, si par ailleurs une machine est aujourd'hui elle aussi un apprenant, alors :

« Que nous reste-t-il, à nous autres les humains ? »

La question vous fait-elle peur ? Si ce n'est pas le cas, peut-être faites-vous alors partie de ces optimistes qui estiment que l'intelligence humaine restera pour longtemps encore insurpassable. Vous pensez que l'intelligence humaine, multiforme, **discursive**, tout à la fois créative, capable d'**abstraction**, de volonté, de sensibilité et dotée d'empathie n'aura pas d'équivalent artificiel de sitôt. Sans doute avez-vous raison. Mais combien de nos tâches quotidiennes exigent aujourd'hui une telle forme d'intelligence ? Quelles sont les compétences permettant à un individu de préserver son employabilité ? Quelles sont les compétences attendues aujourd'hui dans un monde de l'entreprise en quête d'innovation permanente. Voilà des questions concrètes sur lesquelles nous devrions tous nous interroger.

Assurément, les compétences permettant la réussite professionnelle au 21ème siècle ne seront plus les mêmes qu'au siècle précédent. Alors que les professionnels du 20e siècle faisaient appel à des compétences «automatisables», celles que l'on confie progressivement à l'IA, les prochaines années sont amenées à changer la donne. Et la révolution est déjà en marche. Ces vingt dernières années ont vu croitre l'importance de compétences comme l'aptitude à une **communication** efficace, fondée notamment sur la capacité d'**empathie,** les aptitudes à la collaboration ainsi que les capacités d'analyse. Nous parlons ici des **softskills**. Il est désormais indispensable de savoir mettre en cause la fiabilité des informations que nous lisons, d'être créatif et curieux, de savoir travailler en équipe et de communiquer avec le bon niveau de discours. Face à la rapidité et à la variabilité de l'information, il faut s'adapter, prendre des initiatives faire preuve de discernement et, de plus en plus, être capable de surprendre.

Le concept de softskills était il y a quelques années encore confiné aux rayonnages du développement personnel dans les meilleures librairies. Les ouvrages sur le Design Thinking, l'innovation, la collaboration, l'intelligence émotionnelle, etc. occupent aujourd'hui les linéaires réservés au management et à l'économie. Ces compétences doivent être repensées à l'heure de la nouvelle révolution industrielle. L'IA, le big data, l'internet des objets exigent de repenser les référentiels de compétences existants. Que sont les softskills aujourd'hui ? Très simplement, ce sont les compétences qui seront demain intrinsèquement humaines ! Ou plutôt celles que nous allons volontairement réserver à l'homme, a contrario de celles que nous allons progressivement confier à la machine.

Au-delà du dessin d'une frontière entre intelligence artificielle et humaine, ce sont probablement les contours d'une nouvelle alliance entre l'homme et la machine qui se dessinent.

Soyons un peu optimistes : on peut même imaginer que l'automatisation d'un nombre croissant de processus cognitifs ait pour conséquence de nos pousser dans nos retranchements proprement humains. Bien sûr il faudrait alors s'en réjouir.

Références

- [ADC] **L'art difficile de la conversation artificielle**, P. Lemberger, IA Lab – janvier 2019
  weave.eu/lart-difficile-de-la-conversation-artificielle/

- [AGZ] **De quoi AlphaGo Zero est-il le progrès ?** P. Lemberger, IA Lab – janvier 2018
  weave.eu/de-quoi-alphago-zero-progres/

- [CJD] **Comment justifier les décisions d'une IA pour créer la confiance ?** N. Parent, P. Lemberger , IA Lab – février 2018

- [MLD] **Le Machine Learning déctypté**, P. Lemberger, IA Lab – avril 2018
  weave.eu/machine-learning-decrypte/

- [RLI] **Reinforcement Learning: An Introduction**, R.S. Sutton, A. Barto, Bradford Books – novembre 2018

Conception

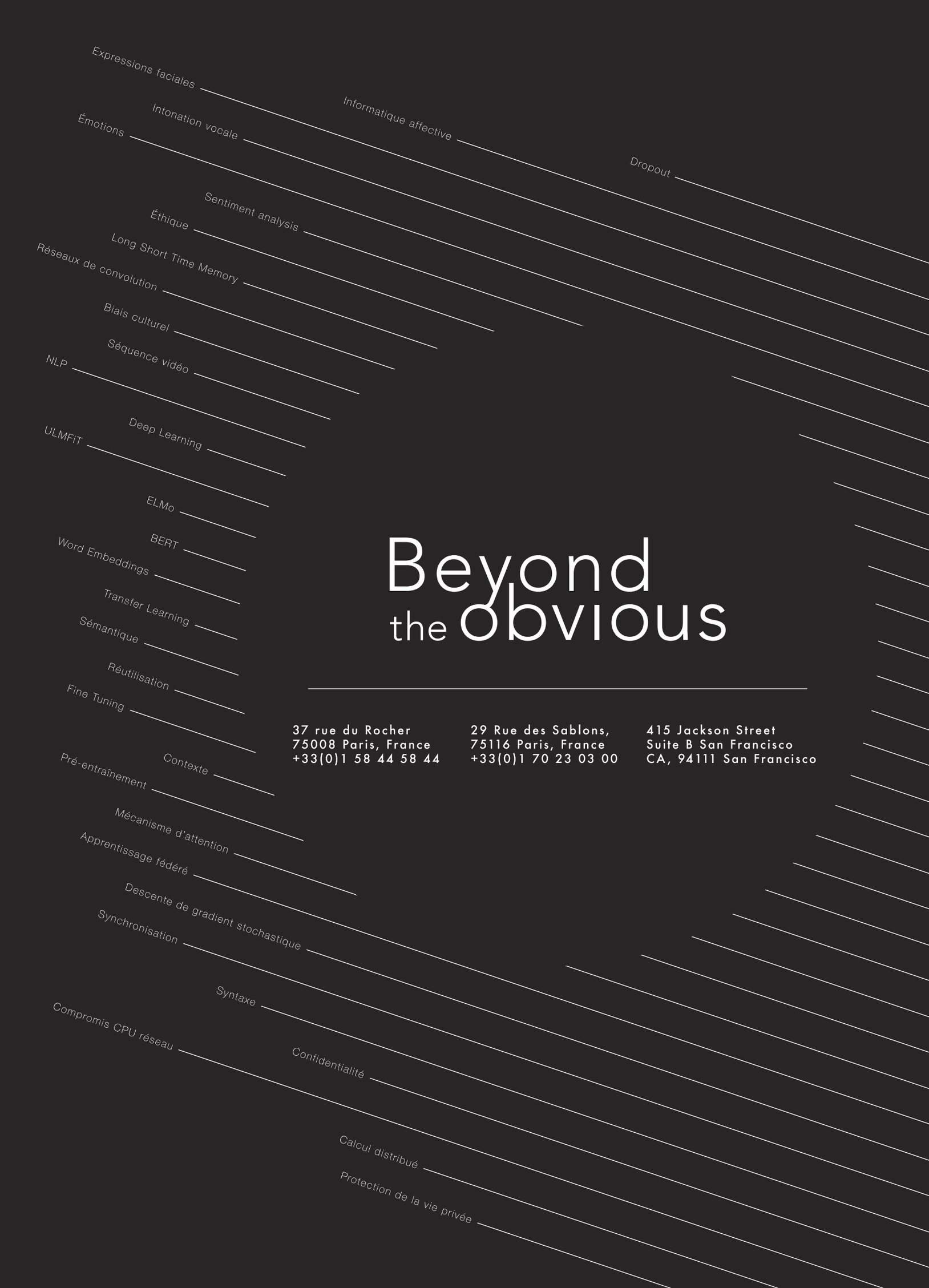
Directeur scientifique : Pirmin Lemberger

Auteurs : Pirmin Lemberger **/** Nicolas Parent **/** Thomas Scialom **/** Fabrice Mauléon

Éditeur : Thibault Hache

Conception Graphique : Quentin Talamoni

Illustrations : Etienne Appert



Beyond the obvious

37 rue du Rocher
75008 Paris, France
+33(0)1 58 44 58 44

29 Rue des Sablons,
75116 Paris, France
+33(0)1 70 23 03 00

415 Jackson Street
Suite B San Francisco
CA, 94111 San Francisco