Travail de lecture et de rédaction scientifique sur le Federate Learning

Bal Sébastien

12 juin 2021

Remerciements

TABLE DES MATIÈRES

CHAPITRE

ONE

CONCEPTS

1.1 Machine Learning

Le Machine Learning appelé en Français apprentissage automatique [x : wikipédiea] a pour objectif de traiter l'information afin de lui donner de la valeur ajoutée.

De nos jours, le Machine Learning est présent partout sur la toile, cela va du moteur de recherche comme Google, aux assistants vocaux comme Siri et Alexa, les fil d'actualités des réseaux sociaux comme Facebook et Twitter. Le point commun entre toutes ces platformes et le stockage massif des données de leur utilisateur appelé Big Data [x : ?]. Le Big Data est une technologie apparue dans les années xxxx, a permis l'essor de l'apprentissage automatique, le Machine Learning. En effet, cet imposant volume de données collectées sur les utilisateurs a permis, dans les exemples cités ci dessus, de mieux cibler le comportement des utilisateurs et donc améliorer les expériences.

Pour fonctionner, le Machine Learning(ML) a besoin de données à ingérer et d'avoir un modèle appris afin de fournir des données à valeur ajoutée comme des algorithmes de prédiction de bourse et la maintenance prédictive. Il est donc nécessaire d'identifier les solutions pour créer ces modèles de ML.

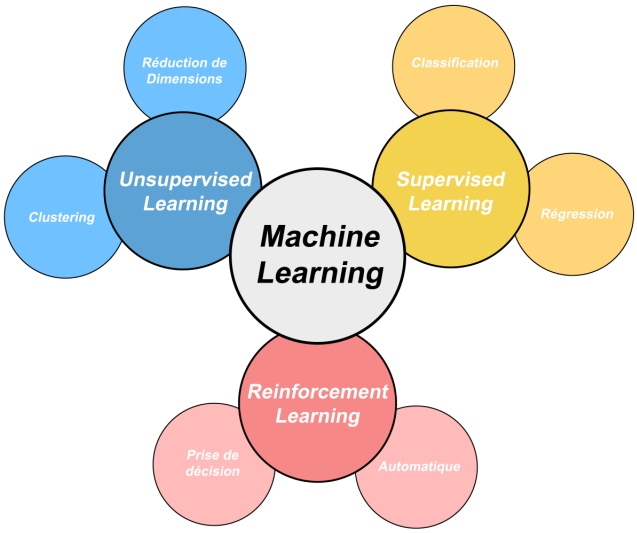

FIGURE 1.1: Familles d'algorithmes les plus utilisés

La première famille, l'appentrissage supervisé (en jaune dans la figure 1.1) consiste à donner des données en entrées, le résultat attendu itéré sur un grand jeu de données afin de trouver la modèle. Pour que le modèle deviennent performant, on fournit un grand volume de données dans le but qu'il se rapproche du modèle attendu.

La deuxième famille,l'apprentissage non-surpervisé (en bleu dans la figure 1.1) consiste à apprendre par reconnaitre des ressemblances et des différences entre les données fournies. L'algorithme rassemble les données en groupe suivant ce qui lui semble le plus pertinant. Ainsi quand on passe un modèle bien précis à l'algorithme, il trouve plus facilement car il a été entrainé.

Pour finir, il existe une catégorie qui gère sa propre expérience. En effet, l'apprentissage par renforcement (en rouge dans la figure 1.1) consiste à générer ses propres expériences. On se rapproche de l'automobile autonome, la machine change ses états suivant les actions qu'elle entreprend de faire. Un système de récompense positive et négative est mis en place pour constituer une nouvelle expérience et rendre la machine attentive pour maximiser ses chances de réussite.

Pour conclure, le ML est une technologie qui vise à trouver des modèles, comprendre des comportements afin de prédire les besoins d'une application suivant un utilisateur.

On peu constater que ce type de besoin se focalise sur une application spécifique cependant, le ML connait des limites au niveau des complexités combinatoire [x :], or certaines applications nécessitent des applications plus complexes avec plus d'entrées, tels que le traitement des images. Pour palier aux limites du ML, le Deep Learning a vu son essort [x : lien vers essort DL]

1.2 Deep Learning

Le Deep Learning(DL) est représentatif d'un système neuronal comme notre système cérébral, il est conçus de plusieurs neuronnes qui intéragissent entre eux. Pour rendre performant tous ces neuronnes, il est nécessaire d'avoir une grande quantité de données, un Data Lake. Celui ci fournit au système plusieurs informations afin de lui constituer une "mémoire". Cette mémoire lui permet de reconnaitre des éléments bien particulier suivant l'entrainement qu'il aura suivit. Cet entrainement est supervisé par des développeurs qui vérifient que le DL ne sort pas de son modèles définis. Si il s'en éloigne, ils corrigent son algorithme mathématique pour le rendre performant et le remettre sur le droit chemin. Pour que le modèle mathématique deviennent performant, il faudra l'entrainer à reconnaitre une donnée en particulier. C'est le sujet de notre prochaine section.

1.2.1 Modèle d'entraînement

Pour entrainer le DL, il lui faut un algorithme mathématique et un grand flux de données afin d'affiner ses recherches et prendre de l'expérience. Dans notre situation, on décider de prendre la reconnaissance d'une image, en particulier celle d'un chat. On fournit à l'algorithme un flux d'images de plusieurs espèces d'animaux, on définit les paramètres qui permettent d'affirmer que l'image que l'on soumet soit bien un chat. Ainsi avec ces critères, l'algorithme devient plus précis car on le guide un peu sur l'objectif qu'il doit atteindre.

Voici un schéma pour l'exemple du fonctionnement du DL,[fig2.1], les boules vertes représente le bon chemin que le système va prendre pour arriver à vérifier le modèle qui était demandé. Les boules bleus sont celles qui ont des caractéristiques avec le modèle mais ne correspondra pas exactement au modèle qui était demandé. Les boules rouges quand à elles, représentent les erreurs que le système a exclu pour pouvoir apprendre le modèle exacte. Les erreurs sont par la suite renvoyées en amont du système pour que le système ajuste son modèle mathématique

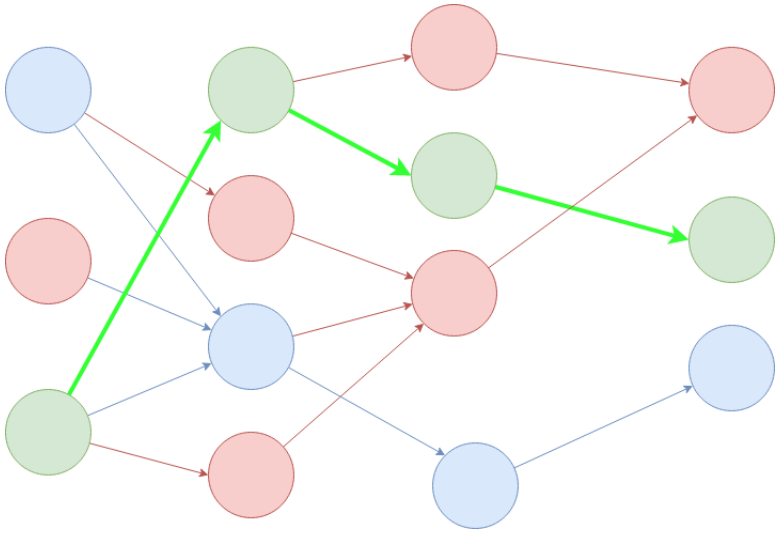

FIGURE 1.2: Autoapprentissage Deep Learning

On peut constater que ce type d'algorithme peut être très vite devenir énergivore. De plus, le DL se focalise principalement sur une application. Or, il peut être intéressant d'étudier plusieurs applications similaires afin d'identifier des modèles plus poussés. C'est dans ce contexte que nous allons introduire le Federated Learning.

1.3 Federated Learning

Notre époque fait face à deux défis importants en terme d'avencer technologique : les respects des données des utilisateurs et l'augmentation du volume des données. L'une des plus importantes est celle de la privatisation de ces données sur le web. Avec la protection des données (RGPD) promulgué en 2018, les données privées font entièrement partie de l'utilisateur, ces données ne peuvent pas être utilisé sans son accord. Ensuite le silo des données met un frein à l'évolution de l'industrie moderne, c'est en ayant plus de données que l'on peut améliorer la qualité de la formation. Cependant le manque de données valides dans le domaine médical vient principalement du faite que les travailleurs doivent être expérimentés dans certains domaines comme celui de l'industrie médicale pour que la qualité des données soient au rendez vous.

C'est grâce au Federated Learning(FL) que l'essort de l'industrie a relevé ces défis. Car le FL est un processus d'apprentissage automatique qui vise a exploité les îlots de données tout en gardant la sécurité des données. Des clients sont coordonnées avec un ou plusieurs serveurs pour faciliter une décentralisation d'un appentrissage automatique. Le FL est lié aux deux concepts que nous avons vu précédements le Machine Learning et le Deep Learning, ce système distribué contient des calculs et des données tous les deux distribués. Le traitement distribué met en place plusieurs appareils connectés à différents endroits via un réseau de communication sous la surveillance d'un serveur centrale afin d'effectuer une partie de la même tâche pour l'accomplir. Le modèle distribué vise à améliorer l'étape de traitement comparer au FL qui lui se base sur un modèle collaboratif afin d'éviter des fuites de confidentialité.

Pour mieux comprendre son fonctionnement, le FL déroule en plusieurs étapes, la formation fédérée est celle qui est la plus courante. Pour commencer, un appareil mobile se procure un modèle en le téléchargeant afin d'avoir un formation suivie en local. En second lieu, le modèle téléchargé est subit à de multiple mises à jours régulière en local afin d'être amélioré, celles ci contiennent des données locales qui appartiennent à différents appareils séparés. Ensuite, ils téléchargent des informations d'un champ de vecteurs présent dans un cloud. En troisième lieu, les modèles locaux effectuent une mise à jour moyenne au sein du cloud et est envoyé à un appareil défini comme le modèle mondial renouvelé. Pour finir, ces étapes précédentes se répètent afin d'arriver à un modèle avec une certaine performance ou qu'une date limite soit atteinte.

Le FL est capable de traiter des données partionnées horizontalement suivant des echantillons et aussi de façon verticale suivant les caractéristiques du modèle d'apprentissage.

En quelques mots, c'est un apprentissage automatique distribué qui permet d'entraîner un modèle mathématique avec un large groupe de données décentralisées qui se trouvent sur des appareils distants.

CHAPITRE

TWO

TECHNOLOGIES EXISTANTES (FRAMEWORK)

2.0.1 TensorFlow Federated (TFF)

Les frameworks ne manquent pas pour manipuler des modèles mathématique et les implémenter pour du ML. Le framework TensorFlow Federated en est un qui est totalement openSource. Il permet aux développeurs de simuler des algorithmes d'apprentissage sur leur modèles et bien entendu tester de nouveau algorithmes. Le TFF est constitué d'un environnement d'exécution de simulation dans lequel on peut simulé nos algorithmes, celle ci est utilisée par une seule machine dédiée aux tests.

2.0.2 Federated AI Technology Enabler (FATE)

CHAPITRE

THREE

CATÉGORISATION

Le FL est caractérisé par trois grands groupes qui surviennet régulièrement, ceux ci sont le FL Horizontal, le FL vertical et le Federated Transfer Learning. Les données stockées sont placées dans différents noeuds, celles ci sont sous forme de matrice de caractéristiques. Les données sont composées de multiples instances, la partie horizontale est représentée comme un client, celle verticale se caractérise par les caractéristiques du client. Pour finir, on peut séparer le FL suivant le mode de partition des données.

3.0.1 Horizontal FL

Pour celui du FL Horizontal, des chevauchements peuvent apparaitrent entre les caractéristiques des données qui sont réparties sur différents noeuds, malgré le fait que les données ne sont pas semblables sur l'espace d'échantillonnage. Les données du point de vue de l'échantillonage sont différentes dans les scénarios des appareils connecté à Internet et les appareils intelligents. Mais ils sont particulièrment similaire dans un espace de fonctionnalité. Les mises à jours, comme nous l'avons vu précédement dans le point du FL, celle si est faite de façon horizontal. En effet, la dimension d'entité est la même pour chaque données. Dans les applications médicales, une forte hausse de travail est nécessaire pour collecté un grand nombre de données. Car il est difficile voir inconcevable pour chaque hôpital de créer une banque de données à partager. Via un réseau fédéral pour les hôpitaux construit par le FL permet d'améliorer le modèle(voir Fig 3.1).

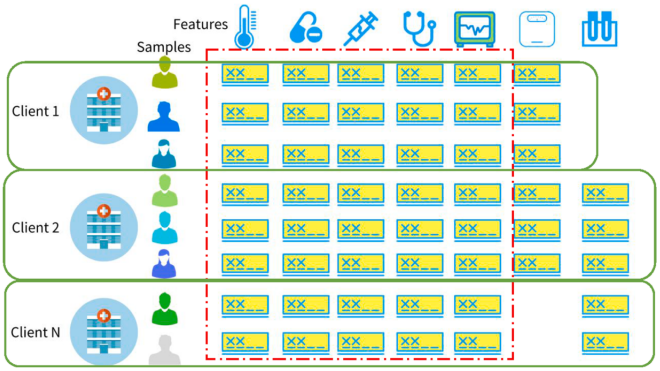

FIGURE 3.1: Horizontal FL

3.0.2 Vertical FL

Lorsque les données sont partitionnées, le FL vertical est plus approprié, chaque parties contient des données homogènes. Ce qui implique que les chevauchement se font en partie sur l'ID de l'échantillon alors qu'elle est différente dans l'espace fonctionnel. Prenons le cas d'un établissement médical, ils souhaitent identifier des maladies telles que le diabète. Il peut être analysé suivant certaines dimensions comme l'âge, le poids et les antécédants médicaux le type de diabète qu'un patient peut avoir. Avec le FL, certaines applications possèdent des données comme le nombre de pas ou la composition des plats qu'un personne a. Ces données peuvent être utilisées pour faciliter cette reconnaissance. La figure 3.2 illustre très bien le FL vertical.

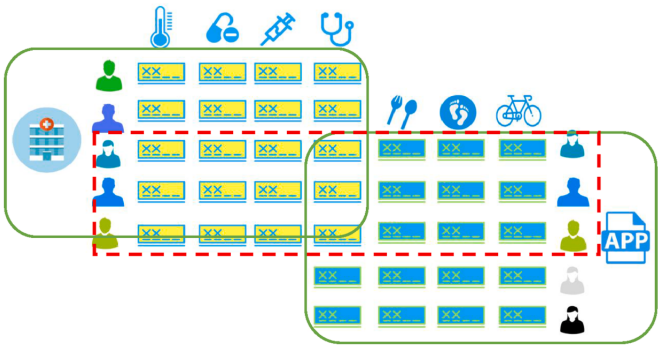

FIGURE 3.2: Verticla FL

3.0.3 Federated Transfer Learning (FTL)

CHAPITRE

FOUR

EXEMPLES D'UTILISATIONS

BIBLIOGRAPHIE

[1] Leslie Lamport, *LaTeX : A Document Preparation System*. Addison Wesley, Massachusetts, 2nd Edition, 1994.

[2] Pour la partie sur le Machine Learning https ://www.lebigdata.fr/machine-learning-et-big-data

ANNEXE

A

ANNEXE

TOWARDS FEDERATED LEARNING AT SCALE: SYSTEM DESIGN

Keith Bonawitz [1] **Hubert Eichner** [1] **Wolfgang Grieskamp** [1] **Dzmitry Huba** [1] **Alex Ingerman** [1] **Vladimir Ivanov** [1] **Chloé Kiddon** [1] **Jakub Konečný** [1] **Stefano Mazzocchi** [1] **H. Brendan McMahan** [1] **Timon Van Overveldt** [1] **David Petrou** [1] **Daniel Ramage** [1] **Jason Roselander** [1]

ABSTRACT

Federated Learning is a distributed machine learning approach which enables model training on a large corpus of decentralized data. We have built a scalable production system for Federated Learning in the domain of mobile devices, based on TensorFlow. In this paper, we describe the resulting high-level design, sketch some of the challenges and their solutions, and touch upon the open problems and future directions.

1 INTRODUCTION

Federated Learning (FL) (McMahan et al., 2017) is a distributed machine learning approach which enables training on a large corpus of decentralized data residing on devices like mobile phones. FL is one instance of the more general approach of "bringing the code to the data, instead of the data to the code" and addresses the fundamental problems of privacy, ownership, and locality of data. The general description of FL has been given by McMahan & Ramage (2017), and its theory has been explored in Konečný et al. (2016a); McMahan et al. (2017; 2018).

A basic design decision for a Federated Learning infrastructure is whether to focus on asynchronous or synchronous training algorithms. While much successful work on deep learning has used asynchronous training, e.g., Dean et al. (2012), recently there has been a consistent trend towards synchronous large batch training, even in the data center (Goyal et al., 2017; Smith et al., 2018). The Federated Averaging algorithm of McMahan et al. (2017) takes a similar approach. Further, several approaches to enhancing privacy guarantees for FL, including differential privacy (McMahan et al., 2018) and Secure Aggregation (Bonawitz et al., 2017), essentially require some notion of synchronization on a fixed set of devices, so that the server side of the learning algorithm only consumes a simple aggregate of the updates from many users. For all these reasons, we chose to focus on support for synchronous rounds, while mitigating potential synchronization overhead via several techniques we describe subsequently. Our system is thus amenable to running large-batch SGD-style algorithms as well as Federated Averaging, the primary algorithm we run in production; pseudo-code is given in Appendix B for completeness.

In this paper, we report on a system design for such algorithms in the domain of mobile phones (Android). This work is still in an early stage, and we do not have all problems solved, nor are we able to give a comprehensive discussion of all required components. Rather, we attempt to sketch the major components of the system, describe the challenges, and identify the open issues, in the hope that this will be useful to spark further systems research.

Our system enables one to train a deep neural network, using TensorFlow (Abadi et al., 2016), on data stored on the phone which will never leave the device. The weights are combined in the cloud with Federated Averaging, constructing a global model which is pushed back to phones for inference. An implementation of Secure Aggregation (Bonawitz et al., 2017) ensures that on a global level individual updates from phones are uninspectable. The system has been applied in large scale applications, for instance in the realm of a phone keyboard.

Our work addresses numerous practical issues: device availability that correlates with the local data distribution in complex ways (e.g., time zone dependency); unreliable device connectivity and interrupted execution; orchestration of lock-step execution across devices with varying availability; and limited device storage and compute resources. These issues are addressed at the communication protocol, device, and server levels. We have reached a state of maturity sufficient to deploy the system in production and solve applied learning problems over tens of millions of real-world devices; we anticipate uses where the number of devices reaches billions.

[1]Google Inc., Mountain View, CA, USA. Correspondence to: Wolfgang Grieskamp <wgg@google.com>, Vladimir Ivanov <vlivan@google.com>, Brendan McMahan <mcmahan@google.com>.

Proceedings of the 2^{nd} SysML Conference, Palo Alto, CA, USA, 2019.