Travail de lecture et de rédaction scientifique sur le Federate Learning

Bal Sébastien

23 mai 2021

Remerciements

TABLE DES MATIÈRES

CHAPITRE

ONE

MACHINE LEARNING

1.1 Les concepts du 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.

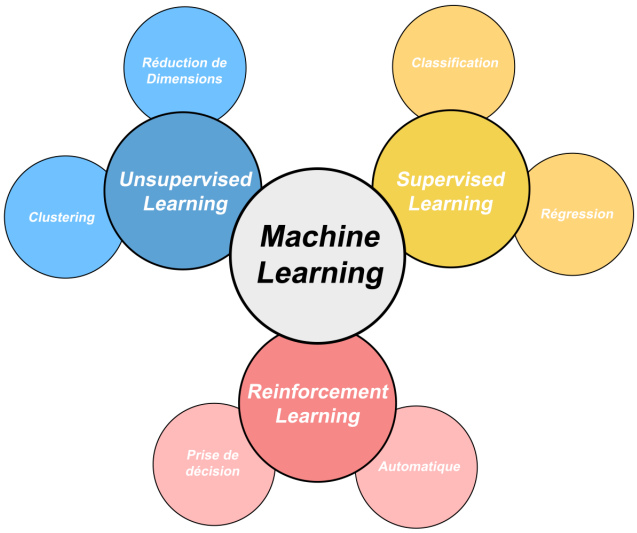

FIGURE 1.1: Familles d'algorithmes les plus utilisés

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]

CHAPITRE

TWO

DEEP LEARNING

2.1 Le concept du Deep Learning

Le Deeo Learning est basé sur un système d'un réseau neuronal inspiré des systèmes cérébraux. Ce type d'apprentissage est d'y supervisé car c'est le développeur qui va décidé sur quel type d'apprentissage il va lancer le Deep Learning. Cette technique à besoin d'énormément de données, on parlera donc de Data Lake.

Pour que le modèle mathématique deviennent performant, il faudra l'entrainer à reconnaitre un élément en particulier. Prenons le cas de la reconnaissance d'un animal. Pour la phase d'apprentissage nous passerons au système plusieurs images d'animaux. On précisera dans la partie d'entrainement les éléments auquels le système devra être conscient.

2.2 L'entraînement du Deep Learning

Pour notre exemple, 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

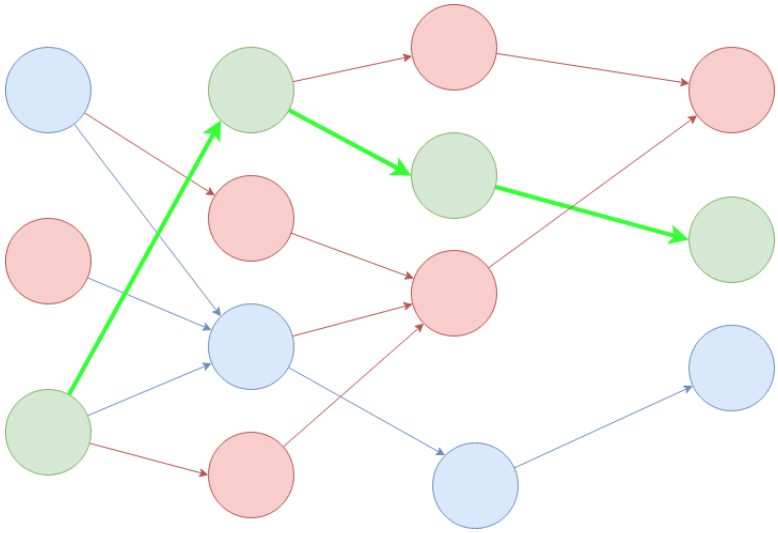

FIGURE 2.1: Autoapprentissage Deep Learning

On peut constater que ce type d'algorithme peut être très vite devenir énergivore. De plus, de 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.

CHAPITRE

THREE

FEDERATE LEARNING

3.1 Définition du Federate Learning

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 téléphones portables(pour notre cas ici).

3.2 Algorithme de Federate Learning

3.3 Implémentation

Dans notre modèle, nous allons utilisé le système TensorFlow pour former notre système neuronal. TensorFlow est une bibliothèque open source pour le Machine Learning. C'est un petit couteau Suisse qui contient ici des outils pour permettre de résoudre des problèmes mathématiques.

3.4 Deep Learning, cas d'utilisation

3.4.1 Smart Building

3.4.2 Industrie

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.