De l'ingénierie de la prédiction

Lydie du Bousquet

Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000 Grenoble

Résumé

Depuis quelques années, l'usage de l'intelligence artificielle en général, et des approches *machine-learning* en particulier s'est démocratisé. Désormais des plate-formes "*machine learning as a service*" offrent une multitude d'algorithmes permettant de formater et nettoyer les données, puis de construire un modèle et de l'exporter. Le résultat est un modèle appris, qui se présente comme un composant, similaire à un composant sur étagère (COTS).

Pour autant, un tel composant est différent d'un COTS classique, parce que ses sorties ne peuvent se spécifier : il s'agit de prédictions par rapport à des données de référence (utilisées pendant l'apprentissage). La construction et l'intégration d'un composant aux comportements appris implique des méthodes différentes qui sont à construire et à populariser.

1 Introduction

Le concept de "*machine-learning as a service*" consiste en une plate-forme dotée d'une interface web [2]. Après avoir créé son compte, un utilisateur peut charger ses données sur le site et utiliser les nombreux utilitaires et algorithmes disponibles pour créer et exporter un modèle.

Ces plate-formes sont très faciles d'usage. L'utilisateur spécifie les étapes de traitements de ses données graphiquement sous la forme d'un workflow. Pour cela, il récupère les fonctions dont il a besoin ("filtres") dans une bibliothèque glissé-déposé et les connecte avec sa souris. Tout est fait pour que les filtres se composent facilement, de sorte qu'il n'y a pas besoin d'être un programmeur, ni même *data-scientist* pour obtenir un résultat qui peut s'intégrer dans une application quelconque.

L'étonnante facilité de production d'un modèle démocratise de facto l'usage des algorithmes de *machine-learning*, et de l'intelligence artificielle en général. S'en suivent un certain nombre d'idées reçues qui fragilisent la qualité du logiciel à venir. La communauté du génie logiciel doit lutter contre ces idées reçues en apportant des solutions et former nos futurs ingénieurs.

2 Idées reçues

Idée reçue 1

> On n'a pas besoin d'être un expert pour construire un modèle.

Malheureusement, c'est vrai. Ces plate-formes permettent de construire un modèle tellement facilement que quiconque peut construire un modèle. Pour autant, tout modèle produit n'est pas forcément *pertinent*.

En effet, un modèle appris est une représentation d'un ensemble de données. Si les données sont de "mauvaise qualité", le modèle le sera aussi [3]. Par mauvaise qualité, on entend des données qui ne seraient pas représentative de l'ensemble du domaine à modéliser, incomplètes, incorrectes ou corrompues. C'est pourquoi, il est nécessaire de préparer les données avant de commencer l'apprentissage. Cela peut représenter jusqu'à 80% du travail de production d'un modèle et nécessite un niveau d'expertise métier important [1].

Il faut aussi une expertise relative aux algorithmes de *machine-learning*. Il existe en effet des dizaines d'algorithmes différents. Chacun fait des hypothèses sur le type de données sur lequel il peut ou non être appliqué. Après obtention du modèle, il faut être capable d'évaluer la pertinence des tests [1] : est-ce qu'une précision de 70% est acceptable ? Dans quelle situation ?

Idée reçue 2

> On peut intégrer un composant appris comme n'importe quel autre composant sur étagère.

En pratique, c'est techniquement possible. En revanche, ce n'est pas souhaitable. En effet, les sorties un composant sur étagère classique sont sont le résultat d'un calcul (d'une suite d'instructions). A l'inverse, les sorties d'un modèle issu de l'apprentissage sont des "prédictions" par rapport à un ensemble de données de référence, et sont souvent accompagnées d'une probabilité ou d'un "degré de certitude". Or rien ne garantit qu'à l'exécution, les entrées du composant appris correspondront à ces données de références. Rien ne garantit que le degré de certitude soit assez élevé pour que la sortie puisse être utilisable à un instant donné.

L'intégration d'un composant appris dans une architecture logicielle plus générale doit donc être réinventée. Il faut fournir des méthodes et des outils permettant aux concepteurs de s'interroger sur les actions à conduire si les entrées du composant appris sont significativement éloignées ou si les produites ne sont pas assez "certaines" (ingénierie des besoins). Au-delà du questionnement, il faut proposer des solutions de conception [2] qui encouragent :

1. On parle ici des tests du modèle effectué sur un sous-ensemble des données.
2. par exemple sous la forme de patrons de conception ou de patrons d'architecture.

— l'usage de composants supplémentaires en amont et en aval du composant appris pour filtrer les entrées et/ou les sorties du composant appris ;
— la définition et l'usage d'un composant (non-appris) implémentant un comportement par défaut si la sortie produite n'a pas un degré de certitude suffisant ;
— l'implication de l'utilisateur dans certaines décisions quand le degré de certitude est trop faible (*human in the loop).*

Idée reçue 3

Il n'est pas possible de tester/valider une application contenant un composant appris puisqu'on ne peut pas spécifier l'application.

Traditionnellement, les tests sont générés à partir de spécification ou du code. Or un composant issu de l'apprentissage n'est pas livré avec sa spécification [3] et il ne se présente pas comme une suite d'instructions.

Il n'en reste pas moins que l'ensemble d'une application ne repose pas forcément sur le composant appris. Le reste de l'application peut/doit être validé. Par ailleurs, même s'il n'est pas forcément facile de tester les fonctionnalités impactées par le composant appris, il est quand même souvent possible de décrire ce que l'application ne doit pas être en mesure de faire, ainsi que les exigences non-fonctionnelles.

3 Des idées reçues aux défis

Dans cet article, nous n'avons recensé qu'un petit nombre d'idées reçues : il en existe d'autres. Le premier défi de la communauté passe par la lutte contre ces idées reçues passe au travers de la formation de nos étudiants. Le second défi consiste en la production des méthodes et outils apportant des solutions aux problèmes soulevés à tous les niveaux des processus de développement : ingénierie des besoins, conception-implémentation, validation.

Références

[1] Cleaning big data : Most time-consuming, least enjoyable data science task, survey says, March 2016. https ://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/ [Retrieved July, 2019].

[2] Jane Elizabeth. Top 5 Machine Learning-as-a-Service providers, Feb. 2018. https ://jaxenter.com/top-5-machine-learning-service-providers-141275.html [Retrieved dec, 2019].

3. D'une certaine façon, sa spécification correspond à l'ensemble des données utilisées pendant l'apprentissage.

[3] Ki Hyun Tae, Yuji Roh, Young Hun Oh, Hyunsu Kim, and Steven Euijong Whang. Data cleaning for accurate, fair, and robust models : A big data - ai integration approach. In *Proceedings of the 3rd International Workshop on Data Management for End-to-End Machine Learning*, DEEM'19, pages 5 :1–5 :4, New York, NY, USA, 2019. ACM.