Comment Saboter l'Apprentissage Collectif par Commérage

Alexandre Pham, Maria Potop-Butucaru, Sébastien Tixeuil, Serge Fdida

Comment Saboter l'Apprentissage Collectif par Commérage

Alexandre Pham[1] et Maria Potop-Butucaru [1] et Sébastien Tixeuil[1 2] et Serge Fdida[1]

[1]*Sorbonne Université, CNRS, LIP6, F-75005 Paris, France e-mail: Prénom.Nom@lip6.fr*
[2]*Institut Universitaire de France, Paris, France*

L'apprentissage collectif par commérage permet l'apprentissage automatique d'un modèle sans utiliser de serveur central. Ce contrôle distribué incite à étudier la possibilité d'attaques malveillantes par les participants eux-mêmes. Alors que les attaques par empoisonnement sur les autres formes d'apprentissage automatique ont été largement étudiées, leurs effets sur l'apprentissage collectif par commérage ne fait pas l'objet du même niveau d'attention. Notre travail est le premier à proposer une méthodologie pour évaluer les attaques d'empoisonnement dans ce contexte dans des scénarios avec ou sans attrition. En outre, afin d'évaluer notre méthodologie sur une étude de cas représentative, nous avons étendu un simulateur existant avec un module d'injection d'attaques par empoisonnement.

Mots-clefs : Apprentissage par Commérage, Attaques

1 Introduction

Dans les systèmes traditionnels reposant sur l'apprentissage automatique (ML), une entité centrale gère à la fois le modèle et les données collectées pour l'entraînement du modèle. Une telle centralisation des données pose problème lorsque les données utilisées pour entraîner le modèle sont sensibles et doivent être maintenues privées, ou simplement pour se conformer à la réglementation. L'apprentissage fédéré introduit par Google en 2016 maintient une entité centrale qui coordonne l'entraînement à travers plusieurs machines disséminées dans le réseau, sans que les données d'entrainement soient communiquées au serveur. Malgré son rôle réduit, l'existence d'un serveur central constitue un point de défaillance et une cible d'attaque évidente. L'apprentissage collectif par commérage (GL) vise à effectuer le FL sans se fier à un serveur central, en utilisant des communications pair-à-pair.

Dans ce travail, nous évaluons l'impact des attaquants (également appelés nœuds *byzantins* ou *malveillants*) qui tentent de corrompre les modèles des participants honnêtes d'un GL. Notre algorithme GL de référence a été proposé par Hegedűs et al. [2]. Proche de notre travail, Giaretta et al. [1] ont étudié l'applicabilité de GL, mettant en évidence des problèmes et proposant des solutions lorsque la distribution des données est corrélée soit à la distribution des degrés soit à la vitesse de communication. Cependant, ils ne considèrent pas les attaques dans leur travail.

Nos contributions. Notre contribution est triple. Tout d'abord, nous proposons une méthodologie pour évaluer les effets d'une attaque par empoisonnement dans un système qui exécute un algorithme GL. Deuxièmement, afin d'évaluer notre méthodologie, nous avons implémenté une extension publique[†] du simulateur populaire *gossipy*[‡] pour simuler des attaques par empoisonnement. Enfin, nous appliquons notre méthodologie sur l'algorithme d'apprentissage par commérage de Hegedűs et al [2] pour évaluer sa résilience aux attaques par empoisonnement selon plusieurs facteurs tels que la topologie, la distribution des nœuds Byzantins et le taux d'attrition. Notre analyse ouvre la voie à la conception efficace de contre-mesures contre les attaques par empoisonnement.

[†] `https://gitlab.lip6.fr/apham/data-poisoning-attacks-in-gossip-learning`
[‡] `https://github.com/makgyver/gossipy/`

2 Methodologie

Dans cette section, nous présentons les différents éléments de notre méthodologie : le choix de l'algorithme de GL à tester, le choix des topologies, de l'ensemble de données et de l'algorithme d'apprentissage automatique, l'attaque, les paramètres d'attrition, et les métriques utilisées pour évaluer l'impact des attaques.

Algorithme GL de l'état de l'art. Récemment, Hegedűs et al [2] ont proposé un algorithme GL qui présente les meilleures performances globales lorsqu'on prend en compte les ressources de communication. Leur algorithme utilise deux mécanismes intéressants. Le premier est une technique de compression, en ne transmettant pas tous les paramètres lors de chaque échange, mais seulement une partie appelée *Partition*, et nous désignons par S le nombre total de partitions que le système utilise, un S élevé implique des messages plus petits. Le deuxième mécanisme permet d'équilibrer l'envoi de messages périodiques et l'envoi de messages en réaction à un événement, ici suite à une mise à jour locale après réception d'une partition.

Topologies pour l'infrastructure GL. Nous étudions les topologies suivantes[§] avec n nœuds : graphes d'arité sortante 20, comme utilisé initialement par Hegedűs et al [2] ; graphes 20-reguliers aléatoires ; graphes Watts-Strogatz pour leur propriété de petit monde ; graphes Erdős-Rényi pour leur propriété équilibrée ; graphes suivant la loi de Zipf, où chaque nœud a au moins un voisin. Pour les deux dernières topologies, nous restreignons nos simulations aux instances où le graphe est connecté.

Ensemble de données pour l'entraînement du modèle d'apprentissage automatique. Nous utilisons la base de données MNIST comme ensemble de données. Elle est constituée de 2 ensembles $\mathscr{D}_{\text{entraînement}}$ et $\mathscr{D}_{\text{test}}$. Une instance de cet ensemble de données est un couple $(x,y) \in \mathbb{R}^{784} \times [\![0,9]\!]$, x représente une image de pixels 28×28 avec le chiffre y écrit dessus. Chaque nœud k aura comme ensemble d'entraînement local $\mathscr{D}^k_{\text{entraînement}}$ de cardinal 250, et tel que $\bigcap_{k=1}^{n} \mathscr{D}^k_{\text{entraînement}} = \emptyset$.

Modèle d'apprentissage automatique. Chaque nœud k entraîne un modèle de régression logistique (multinomiale) pour classer correctement les images de l'ensemble de données décrit précédemment. Nous fixons $\eta = 1$ et $\lambda = 0$, ces valeurs offrant les meilleures performances dans le cas Erdős-Rényi sans attrition, mais le choix de ces 2 valeurs n'est pas trivial et doit être étudié pour toutes les topologies comme cela a été fait par Hegedűs et al [2].

Attaques byzantines. Dans cette étude, les nœuds byzantins insèrent un motif déclencheur de 9 pixels dans 20% des images de leur ensemble de données, qu'ils étiquettent ensuite comme 0. Leur objectif est de faire en sorte que les nœuds honnêtes classent les images marquées comme 0, sans diminuer les performances de classification sur les données propres (c'est-à-dire sans le motif déclencheur). Comme les nœuds byzantins attaquent le système GL en manipulant leur ensemble de données pour perturber le réseau, cela s'appelle une *attaque par empoisonnement de données*, et comme leur objectif est d'introduire un objectif caché, ce type d'attaque est également appelé *attaque à porte dérobée*. Pour évaluer les performances et l'impact de l'attaque sur les nœuds honnêtes, nous tirons 2000 images de l'ensemble de test que nous divisons en deux ensembles de taille égale. La première moitié sera utilisée pour évaluer les performances sur la tâche de classification habituelle. Pour la seconde moitié, nous supprimons tous les (x,y) où $y = 0$, et appliquons les mêmes modifications sur x que celles effectuées par les nœuds byzantins sur les échantillons restants. Ces deux ensembles sont appelés *test* et *porte dérobée*.

Afin de placer les nœuds byzantins, nous utiliserons deux stratégies : *classique* et *aléatoire*. Dans la stratégie *classique*, nous sélectionnons d'abord les nœuds ayant le plus haut degré. Dans la stratégie *aléatoire*, nous sélectionnons les nœuds en échantillonnant aléatoirement sans remplacement.

Métriques. Nous nous intéressons ici à la *précision* moyenne des nœuds honnêtes sur les ensembles test et porte dérobée. La *précision* est définie comme le nombre de prédictions correctes sur le nombre total de prédictions. Pour la première, plus elle est élevée, mieux c'est : les nœuds (honnêtes) doivent bien se comporter sur des données qui n'ont pas été marquées avec le motif, et pour la seconde, c'est l'inverse : les nœuds doivent mal se comporter sur les images empoisonnées.

[§] Ici, nous ne présentons qu'un sous-ensemble des résultats de notre rapport technique [3].

3 Résultats expérimentaux

Nous définissons $n \in \{100, 150\}$ comme le nombre de nœuds et $f = 0.3n$ comme le nombre de nœuds byzantins dans une simulation.

Chaque courbe présentée ici est une moyenne sur au moins 10 exécutions aléatoires. Dans ce qui suit, nous présentons deux scénarios, avec et sans attrition, en commençant par le scénario sans attrition, c'est-à-dire lorsque les nœuds restent présents pendant toute l'exécution.

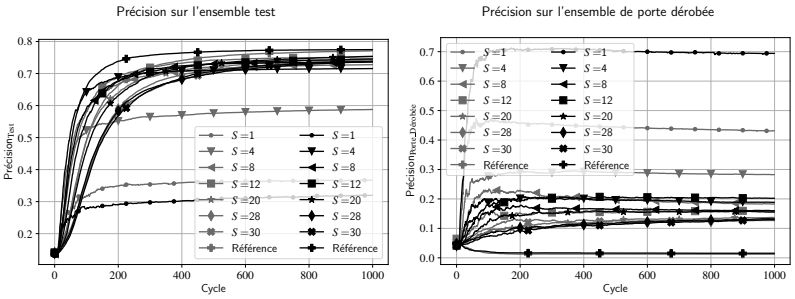
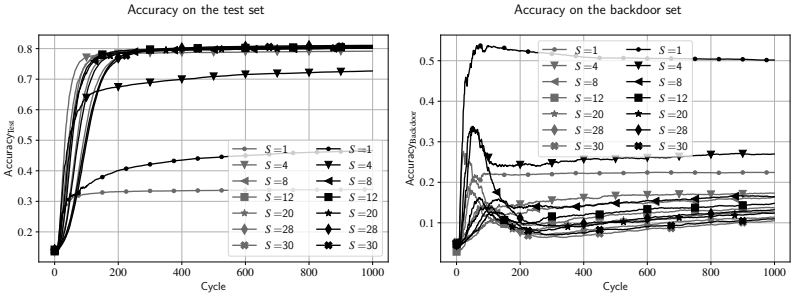


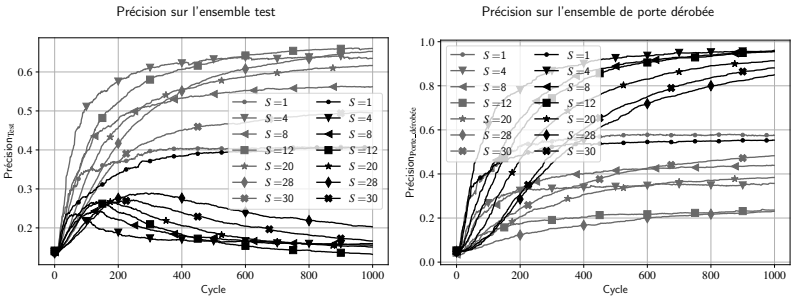
Fig. 1: Précision sur l'ensemble de test (à gauche) et sur l'ensemble de porte dérobée (à droite) pour la topologie d'arité sortante 20 avec $n = 100$ (gris) et $n = 150$ (noir) avec la stratégie de placement aléatoire des Byzantins dans un système sans attrition avec $f = 30$ et 45 respectivement. Les courbes "Référence" représentent les résultats dans des simulations sans Byzantins, avec le meilleur choix de S parmi les valeurs étudiées ici avec Byzantins.

Scénario sans attrition. Dans la Figure 1, nous représentons les performances sur les 2 ensembles définis dans la Section 2 lorsque $n = 100$ et $n = 150$ avec f nœuds byzantins placés selon la stratégie *aléatoire*.

Nous pouvons voir que lorsque S est faible, l'attaque est (trop) puissante pour les nœuds honnêtes, car ils se comportent mal sur l'ensemble de test et classent très bien les entrées altérées par rapport à la référence considérée (meilleur S étudié ici dans des simulations sans Byzantin sur l'ensemble de test). Cela peut s'expliquer par le fait qu'une partition contient très probablement des paramètres qui interagissent avec le motif déclencheur. Lorsque S augmente, nous remarquons que l'attaque est moins puissante, car les nœuds honnêtes se comportent de manière proche de la référence sur l'ensemble de test. Ici, les paramètres qui interagissent avec le motif déclencheur sont trop dilués parmi toutes les partitions, et sont donc moins susceptibles d'être mis à jour. Mais nous remarquons que les nœuds byzantins parviennent toujours à introduire le comportement indésirable (bien que pas autant qu'ils le souhaiteraient).



(a) Watts-Strogatz ($k = 20, p = 0.5$)



(b) Zipf ($\alpha = 2$)

Fig. 2: Précision sur l'ensemble de test et sur l'ensemble de porte dérobée pour différentes topologies avec $n = 150$ et $f = 45$ avec la stratégie de placement aléatoire des Byzantins (gris) et la stratégie de placement classique des Byzantins (noir).

Dans la Figure 2, nous comparons les stratégies *classique* et *aléatoire* pour les nœuds byzantins, dans

des graphes où la distribution des degrés n'est pas uniforme. Nous remarquons que lorsque $n = 150$, les nœuds placés dans un réseau de Watts-Strogatz et les nœuds byzantins sont placés de manière aléatoire, les nœuds honnêtes se comportent de manière similaire à ce qui a été montré dans la Figure 1 lorsque S est élevé (et se comportent mieux lorsque $S = 1$). De manière intéressante, avec la porte dérobée, lorsque les nœuds byzantins sont placés aléatoirement, les nœuds honnêtes placés dans une topologie de Watts-Strogatz gèrent mieux l'attaque que lorsqu'ils sont placés dans un réseau basé sur la loi de Zipf. Nous supposons que ce résultat est dû à des parties locales du graphe où les nœuds honnêtes réagissent très fortement à la porte dérobée, tandis que ce n'est probablement pas le cas pour la topologie de Watts-Strogatz en raison de sa propriété de petit monde. Lorsque les nœuds byzantins sont placés de manière classique, les nœuds honnêtes se comportent beaucoup mieux sur la porte dérobée que sur le test, cela est dû au fait qu'il leur est plus facile d'optimiser avec des données empoisonnées, et lors de l'envoi de partitions, leurs solutions ont un fort impact.

Scénario avec Attrition. Nous étudions maintenant le cas où les nœuds peuvent se déconnecter et se reconnecter et ont une probabilité de 20% d'être en ligne[¶]. Dans la Figure 3, nous étudions $n =$ 100 et 150, avec f nœuds Byzantins placés de manière aléatoire. Nous observons que même si $S = 1$ est spécial, et bien que les nœuds honnêtes ne soient pas résilients contre l'attaque au début, ils parviennent à une solution bien meilleure à la fin, proche des autres valeurs de S étudiées. Cela peut s'expliquer par le fait que les nœuds sont hors ligne la plupart du temps, y compris les nœuds Byzantins, donc les partitions malveillantes sont moins susceptibles d'être traitées. Nous remarquons que pour les autres valeurs de S étudiées, le résultat est très proche de ce qui a été montré dans la Figure 1 malgré la différence de topologie, et s'explique par la distribution uniforme des données : les nœuds honnêtes ayant le même (type) de matériel d'apprentissage, apprennent à accomplir la même tâche (mais plus lentement que quand ils sont toujours présents).

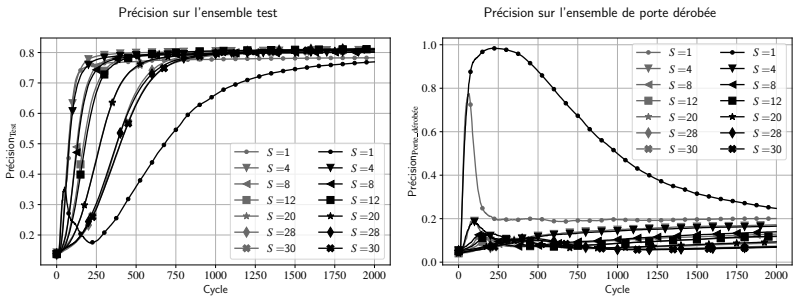


Fig. 3: Précision sur l'ensemble de test et l'ensemble de porte dérobée avec $n = 100$ (gris) et 150 (noir) sur un réseau 20-régulier aléatoire avec $f = 30$ et 45 respectivement.

4 Conclusion

Nous avons proposé une méthodologie pour étudier la résilience de l'apprentissage par commérage en présence d'attaques par empoisonnement. Comme étude de cas, nous ciblons l'algorithme GL de l'état de l'art, et étudions sa résilience dans un large éventail de topologies dans les scénarios sans et avec attrition. Nos résultats montrent que les optimisations de communication et le choix de la topologie sous-jacente ne sont pas toujours favorables aux nœuds honnêtes. De plus, la distribution des nœuds byzantins a également un impact significatif sur la précision des résultats.

Références

[1] L. Giaretta and Š. Girdzijauskas. Gossip Learning: Off the Beaten Path. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1117–1124, Dec. 2019.

[2] I. Hegedűs, G. Danner, and M. Jelasity. Decentralized learning works: An empirical comparison of gossip learning and federated learning. *Journal of Parallel and Distributed Computing*, 148:109–124, Feb. 2021.

[3] A. Pham, M. Potop-Butucaru, S. Tixeuil, and S. Fdida. Data Poisoning Attacks in Gossip Learning. Rapport technique, Sorbonne Université, Jan. 2024.

[¶] Pour se rapprocher du scénario d'attrition utilisé dans [2].