Détection de communauté par apprentissage fédéré en cluster

Mickael BETTINELLI[1] Argheesh BHANOT[1] Alexandre BENOIT[1]
[1]LISTIC, Université Savoie Mont Blanc, Annecy

Résumé – Ce travail aborde le défi de la détection de communautés dans l'apprentissage fédéré, où les données ne sont ni indépendantes ni identiquement distribuées (non-IID) entre les clients. Le *Clustered Federated Learning* (CFL) est un ensemble de méthodes du FL visant à s'adapter au cas non-IID. L'avantage de cette approche est sa capacité à identifier des communautés de clients sans nécessiter d'informations préexistantes, tout en améliorant simultanément la performance de la tâche. Cependant, cette méthode présente une forte variabilité des partitionnements. Ce travail propose une solution à ce problème en s'appuyant sur une matrice d'*agreement*. Le jeu de données MNIST a été utilisé pour réaliser une comparaison des résultats entre un CFL classique et un CFL utilisant la matrice d'*agreement*.

Abstract – This research addresses the challenge of community detection in federated learning (FL), where data is neither independent nor identically distributed (non-IID) across clients. Clustered Federated Learning (CFL) is a set of FL methods designed to adapt to the non-IID case. The advantage of this approach is its ability to identify communities of clients without requiring any prior information, while simultaneously improving task performance. However, this method presents a high degree of clustering variability. This work proposes a solution to this problem based on an agreement matrix. The MNIST dataset was used to compare the results between a conventional CFL and a CFL using the agreement matrix.

1 Introduction

L'*Apprentissage Fédéré* (FL) s'est imposé comme une approche prometteuse pour un apprentissage automatique décentralisé, respectueux de la vie privée et sécurisé [7]. Cependant, l'hétérogénéité et la diversité inhérentes des données des clients posent des défis majeurs, limitant l'efficacité d'un modèle fédéré unique [5]. Les avancées récentes en personnalisation des modèles [11], et plus particulièrement en *Clustered Federated Learning* (CFL) [3], ont permis d'aborder ce défi en regroupant les clients selon des critères de similarité, créant ainsi des modèles intermédiaires qui établissent un lien entre les modèles locaux et globaux. Bien que ces méthodes aient démontré une amélioration des performances, elles se concentrent principalement sur l'optimisation des performances plutôt que sur l'exploration de la détection de communautés. De plus, la reproductibilité des résultats est difficile à obtenir car l'instabilité potentielle de la détection de communautés apporte un degré supplémentaire de stochasticité à l'optimisation des modèles. Dans ce travail, nous explorons le potentiel de détection de communautés du CFL à travers l'étude du comportement des clusters. L'identification de communautés tout au long de l'entraînement ouvre de nouvelles perspectives pour l'optimisation des modèles, en particulier dans des scénarios impliquant des données *non-IID* et des flux de données, en optimisant les modèles de communautés pour les clients ayant des distributions de données similaires.

Nous explorons cette direction en proposant une méthode stabilisant la détection de communautés de clients (ci-après nommés *clusters*) au cours de l'entraînement, et nous l'intégrons à une approche CFL. En utilisant un jeu de données contrôlé, nous évaluons l'efficacité de notre méthode en matière de partitionnement tout en analysant la variabilité des résultats. Notre contribution est une méthode de CFL stabilisant les partitionnements et permettant la détection de communautés côté serveur.

Cet article est structuré de la manière suivante : la seconde section donne un aperçu des méthodes CFL de l'état de l'art. Les troisième et quatrième sections présentent respectivement la problématique et nos contributions. Les cinquième et sixième sections détaillent les conditions d'expérimentation et présente les résultats. Enfin, la dernière conclut l'article.

2 Travaux connexes

La proposition initiale du FL [7] suppose qu'un modèle unique peut être appliqué à tous les clients, même lorsque leurs distributions de données sont non-IID. Cependant, des études ultérieures [5] ont démontré que le FL est particulièrement sensible aux problèmes de biais liés à la distribution des données. L'*Apprentissage Fédéré Personnalisé* [11] a récemment été proposé comme un moyen de fournir à chaque client des modèles partagés pertinents. Parmi les différentes méthodologies, celles basées sur le CFL [9] isolent des groupes de clients afin de réduire les biais et d'offrir des modèles adaptés à des sous-ensembles de clients similaires.

L'algorithme IFCA [4] est une méthode de CFL visant à améliorer l'efficacité du FL lorsque les clients possèdent des données non-IID. Contrairement à l'approche dominante dans la littérature où le regroupement des clients est généralement effectué côté serveur, Ghosh et *al.* proposent un processus de partitionnement côté client pour réduire la charge computationnelle du serveur.

FlexCFL [3] est une technique de CFL visant à regrouper les clients en fonction des similarités de leurs mises à jour de gradient. Un mécanisme spécifique est mis en place pour l'attribution d'un modèle aux nouveaux clients, avec pour objectif d'améliorer la scalabilité. Cependant, comme pour IFCA, le nombre de clusters obtenu avec FlexCFL doit être connu à l'avance, et la qualité du regroupement n'est pas évaluée. Dans cet article, les clients sont bipartitionnés après

chaque tour (*round*) en comparant la similarité cosinus de leurs mises à jour de gradient. Ainsi, les clients sont regroupés en fonction de leurs directions de convergence.

Les travaux présentés dans cette section sont axés sur l'optimisation d'un modèle unique ou d'un ensemble de modèles partagés, avec une variété de stratégies employées. Il est à noter qu'aucun des travaux mentionnés ci-dessus n'évalue la qualité des partitionnements. Dans cette étude, nous étendons ce concept et cherchons à atteindre à la fois une performance optimale de la tâche et la découverte de communautés de clients à partir de leurs modèles tout au long du processus FL.

Algorithme 1 Sélection de clients et attribution du modèle

procedure CONFIGURE FIT(K, P, W_t)
Entrée : une fonction de sélection de clients $SelectionnerClients$, kNN utilise la trusted distance.
Sortie : un sous-ensemble $Q \subset K$ commence le tour avec le modèle approprié.
    $Q \leftarrow$SelectionnerClients(K)
    Pour chaque client $q \in Q$ **faire**
        Si q est un nouveau client **alors**
            $w_{t+1}^q \leftarrow W_t$
        Sinon
            $w_{t+1}^q \leftarrow$ kNN(q, P, k=1)
        Fin Si
        FitRound(q, w_{t+1}^q)
        ▷ les clients s'entraînent localement avec w_{t+1}^q
    Fin Pour
Fin procedure

3 Problématique

Nous nous concentrons sur une configuration standard où seules les mises à jour locales des modèles sont communiquées des clients vers le serveur et sans communication entre les clients. Nous nous attaquons à des problèmes d'optimisation non-convexes traités avec des réseaux neuronaux et nous nous appuyons sur du FL centralisé tel que défini par [7]. Nous supposons qu'un ensemble K de clients participent à l'optimisation de la même architecture de modèle en s'appuyant sur les mêmes critères d'optimisation mais sur des distributions de données d'entraînement différentes. Les clients sont connectés à un seul serveur central qui reçoit et agrège les modèles des clients à chaque tour de communication (*round*). Tous les clients participent aux tours de communication et fournissent au serveur leur modèle local mis à jour. Les modèles communautaires W^{C_i} sont calculés comme la moyenne des poids des modèles locaux w^k des sous-ensembles de clients C_i, avec $C_i \subset K$ et $C_i \neq \emptyset$ définis dans l'eq. 1.

$$W_{t+1}^{C_i} = \frac{1}{|C_i|} \sum_{k \in C_i} w_t^k \tag{1}$$

Avec une telle agrégation, à un tour donné, chaque client a la même contribution à son modèle de communauté W^{C^i}. Enfin, la distribution des données de chaque client reste constante au cours d'une expérience donnée (voir section 5 pour les détails à propos de la distribution des données).

4 Détection de communautés

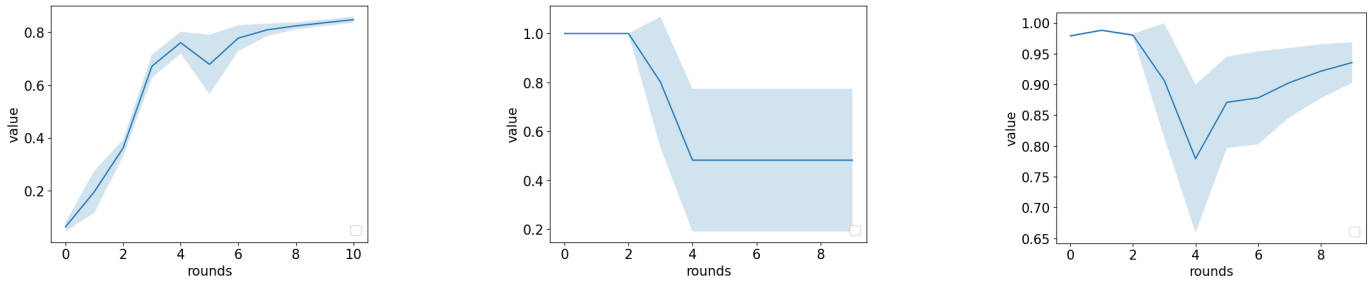
La détection de communautés est une tâche visant à identifier des groupes d'entités qui sont densément connectés en leur sein mais faiblement connectés à d'autres groupes. Les clients doivent être partitionnés sur la base de similarités des paramètres de leur modèle. Nous utilisons une métrique de similarité afin de créer un graphe de clients dans lequel les arêtes représentent un degré de similarité entre les paramètres des modèles des deux clients. Nous présentons ensuite un processus de partitionnement des clients exécuté sur le serveur tout au long du processus d'optimisation. Comme le met en évidence la section 6 de cet article, la détection de communauté en CFL standard présente une certaine instabilité due à l'agrégation et à la redistribution des modèles de communauté à chaque client, ce qui perturbe leurs apprentissages locaux. Cette section présente notre approche de détection des communautés basée sur une matrice d'*agreement* [2] permettant de diminuer l'instabilité des communautés formées.

Algorithme 2 Étape d'agrégation de modèles pour un tour

procedure AGREGATE($modelesClients$)
Entrée : $clients$ le graphe de clients connus, $matriceAgreement$ vide (tous les couples à 0).
Sortie : $clients$, P, W_t et chaque W^{C_i} à mettre à jour
    $clients \leftarrow MAJGrapheClients(modelesClients)$
    Pour r allant de [0.5 :1.5] par pas de 0.05 **faire**
        $P(r) \leftarrow appliquerLouvain(clients)$
        Pour chaque $C_i(r) \in P(r)$ **faire**
            Pour $client_j, client_k \in clients, j \neq k$ **faire**
                Si $client_j$ et $client_k \in C_i(r)$ **alors**
                $matriceAgreement(j,k) += 1$
                Fin Si
            Fin Pour
        Fin Pour
    Fin Pour
    $graphe \leftarrow MatVersGraphe(matriceAgreement)$
    $P \leftarrow suppressionArretes(graphe, 0.6)$
    Pour chaque $C_i \in P$ **faire**
        $W_t^{C_i} \leftarrow \frac{1}{|C_i|} \sum_{k \in C_i} w_t^k$ ▷ Équation 1
    Fin Pour
    $W_t \leftarrow \frac{1}{|P|} \sum_{i \in P} W_t^{C_i}$ ▷ similaire à l'équation 1
Fin procedure

4.1 Métrique de similarité

Des mesures de distance peuvent être employées pour évaluer la dissemblance entre les modèles en fonction de leurs paramètres. Les méthodes les plus courantes basées sur la comparaison des paramètres des modèles sont les *L-normes* et la *distance de cosinus* [6]. Mais ces distances ne comparent pas efficacement le comportement des modèles. Au contraire, la *trusted distance* introduit avec l'optimiseur *Fromage* [1] permet d'exprimer la distance fonctionnelle entre les modèles ayant des structures similaires. En nous basant sur cette métrique, les distances inter-modèles sont ensuite transformées en valeurs de similarité mises à l'échelle sur l'intervalle [1,0] en appliquant l'équation 2. Cela facilite la comparaison et fournit des valeurs normalisées pour les processus suivants :



(a) Précision (b) Adjusted Rand Index (c) Score Silhouette

FIGURE 1 : Résultats du CFL standard sans la matrice d'*agreement*

$$f(v) = 1 - \frac{v - min}{max - min} \quad (2)$$

où v est la valeur de la distance à normaliser tandis que min et max sont respectivement la distance la plus faible et la plus élevée observée parmi toutes les paires de modèles de clients dans un tour donné.

4.2 Partitionnement des clients

À chaque tour, nous considérons l'algorithme de clustering stochastique de Louvain [12] qui est adapté pour un large éventail de nombres de clients tout en présentant peu d'hyperparamètres. Le principal avantage de cette méthode par rapport aux approches traditionnelles telles que K-Mean est sa capacité à partitionner les données sans connaître à priori le nombre de groupes à créer. On utilise la méthode de Louvain avec l'hyperparamètre appelé *résolution* qui régule la taille des communautés. Les faibles résolutions conduisent à de nombreux petits clusters tandis que les plus élevées conduisent à un nombre plus faible de grands clusters. Dans ce qui suit, le partitionnement à un tour donné est appelé P. Chaque communauté i présente un modèle de communauté noté W^{C_i} calculé à partir de l'équation 1. La résolution peut être fixée empiriquement lorsque les données utilisées sont connues et ne changent pas mais son choix devient plus complexe lorsque le nombre de clusters à créer est inconnu ou lorsque celles-ci peuvent varier dans le temps.

Nous proposons une méthode permettant de générer un partitionnement ne nécessitant pas de choisir de valeur de résolution (Algo. 2). Cette méthode applique Louvain pour différentes valeurs de résolution afin d'identifier un consensus de clusters. Une fois que l'on a obtenu des partitionnements pour chaque exécution de l'algorithme, nous observons le nombre de fois que deux clients se trouvent dans la même communauté. Cela est représenté sous forme d'une matrice appelée une matrice d'*agreement*. Pour détecter les communautés, la matrice d'*agreement* est transformée en graphe dans lequel le poids de chaque arête correspond au nombre de fois que les deux noeuds reliés ont été présents dans le même cluster. Lorsque deux noeuds ont été présents dans le même cluster moins de 60% de fois parmi tous les partitionnements, alors leur arête est supprimée. Le partitionnement final est construit en regroupant tous les noeuds encore connectés les uns avec les autres. Ce seuil a été défini empiriquement et peut être sujet à optimisation selon le cas d'étude. Un seuil faible permet de faire moins de grands clusters alors qu'un seuil élevé pourra créer un plus grand nombre de petits clusters. Après la réalisation du partitionnement final, le serveur transmet à chaque client le modèle de la communauté dont il fait partie (Algo. 2).

5 Dispositif expérimental

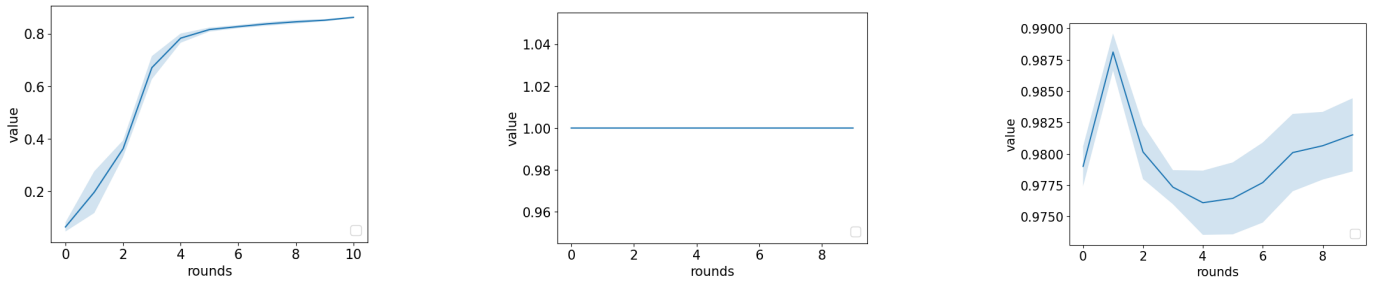
Nous utilisons le jeu de données MNIST, un ensemble d'images à catégoriser, qui regroupe 10 classes de chiffres allant de 0 à 9 écrits à la main. L'ensemble de données MNIST comprend 60.000 images d'apprentissage et 10.000 images de test. Un jeu de données personnalisé construit à partir de MNIST est fourni à chaque client. Ces données sont réparties sans chevauchement de manière déséquilibrée. Les 10 clients du système apprennent chacun sur leur jeu de données local contenant les 10 classes. Chaque client possède deux classes majoritaires. Celles-ci sont partagées avec un autre client. Les deux classes majoritaires comptent pour 30% du jeu de données local (60% au total) tandis que les 8 autres classes comptent pour 5%. L'objectif de la détection de communauté est de détecter les 5 communautés de 2 clients possédant les mêmes classes majoritaires.

Le modèle utilisé pour les expériences est un CNN de 1,663,370 paramètres, composé de deux convolutions et d'une couche dense cachée. Ces expériences ont été menées sur dix processeurs Intel Xeon(R) Gold 5118 à 2,30 GHz, équipé de 40 Go de RAM et utilisant les bibliothèques TensorFlow 2.16.0 et Flower 1.11.0.

La section suivante évalue les performances des tâches et la qualité des partitionnements des modèles au moyen de trois mesures : la *précision* du modèle, le *Adjusted Rand Index* (ARI) et le score de *Silhouette*. ARI [10] produit un score qui évalue un regroupement par rapport à un partitionnement attendu. Le score de Silhouette [8] que nous utilisons est la moyenne des Silhouettes de chaque client. Il évalue à quel point les groupes sont denses et à quel point ils sont éloignés les uns des autres. Chaque expérience est menée sur 10 tours, avec 10 clients, et les résultats sont moyennés sur 5 exécutions avec différentes *seed* d'initialisation nous permettant ainsi de mesurer la variabilité des résultats.

6 Résultats

La figure 1 montre les résultats avec une approche CFL standard. Comme pour notre approche, le partitionnement est réalisé avec l'algorithme de Louvain. Le clustering obtenu est similaire à celui attendu dès le premier tour (Fig. 1b). Cepen-



(a) Précision (b) Adjusted Rand Index (c) Score Silhouette

FIGURE 2 : Résultats avec la matrice d'*agreement*

dant, la qualité du partitionnement est fortement diminuée dès le troisième tour et continue de se dégrader petit à petit tout au long des expériences. Cette dégradation se visualise également avec la Silhouette moyenne (Fig. 1c) qui baisse abruptement au même moment, montrant une diminution de la densité des clusters et un rapprochement des clients à ceux des clusters voisins. En se rapprochant, l'algorithme de Louvain devient moins efficace pour retrouver le bon partitionnement. Au fil des tours de nouveaux clusters se forment et se séparent à nouveau et les clients continuent leur optimisation.

La figure 2 montre les mêmes métriques en utilisant la matrice d'*agreement* pour stabiliser le clustering de Louvain. On observe tout d'abord que l'écart-type est plus réduite pour les trois métriques en comparaison du CFL standard. Les comportements des métriques au cours des tours sont ainsi plus réguliers. La Silhouette moyenne reste plus élevée et sa chute au second tour est nettement plus faible que sans la matrice d'*agreement* (attention à l'échelle). Ces résultats indiquent que la matrice d'*agreement* joue un rôle de stabilisateur du partitionnement. Il faut cependant nuancer cet effet au regard de la distribution de données testée. La matrice d'*agreement* pourrait être bien moins efficace dans les cas où les jeux de données locaux se rapprochent d'une distribution plus homogène (*iid*).

7 Conclusion

Ce papier présente une approche permettant de stabiliser la détection de communauté de clients. L'algorithme basé sur une matrice d'*agreement* partitionne les clients de manière plus régulière d'un tour à l'autre qu'en utilisant uniquement l'algortithme de Louvain. Les résultats préliminaires montrent cette stabilisation du partitionnement et amènent à questionner l'impact de l'optimisation des modèles locaux sur le partitionnement et celui du partitionnement sur les modèles locaux.

Références

[1] Jeremy BERNSTEIN, Arash VAHDAT, Yisong YUE et Ming-Yu LIU : On the distance between two neural networks and the stability of learning. *Advances in Neural Information Processing Systems*, 33:21370–21381, 2020.

[2] Fabrizio DAMICELLI, Claus C HILGETAG, Marc-Thorsten HÜTT et Arnaud MESSÉ : Topological reinforcement as a principle of modularity emergence in brain networks. *Network Neuroscience*, 3(2):589–605, 2019.

[3] Moming DUAN, Duo LIU, Xinyuan JI, Yu WU, Liang LIANG, Xianzhang CHEN, Yujuan TAN et Ao REN : Flexible clustered federated learning for client-level data distribution shift. *IEEE Transactions on Parallel and Distributed Systems*, 33(11):2661–2674, 2021.

[4] Avishek GHOSH, Jichan CHUNG, Dong YIN et Kannan RAMCHANDRAN : An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33:19586–19597, 2020.

[5] Tzu-Ming Harry HSU, Hang QI et Matthew BROWN : Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv :1909.06335*, 2019.

[6] Max KLABUNDE, Tobias SCHUMACHER, Markus STROHMAIER et Florian LEMMERICH : Similarity of neural network models : A survey of functional and representational measures. *arXiv preprint arXiv :2305.06329*, 2023.

[7] Brendan MCMAHAN, Eider MOORE, Daniel RAMAGE, Seth HAMPSON et Blaise Aguera y ARCAS : Communication-efficient learning of deep networks from decentralized data. *In Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[8] Peter J ROUSSEEUW : Silhouettes : a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[9] Felix SATTLER, Klaus-Robert MÜLLER et Wojciech SAMEK : Clustered federated learning : Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 32(8):3710–3722, 2020.

[10] Douglas STEINLEY : Properties of the hubert-arable adjusted rand index. *Psychological methods*, 9(3):386, 2004.

[11] Alysa Ziying TAN, Han YU, Lizhen CUI et Qiang YANG : Towards personalized federated learning. *IEEE transactions on neural networks and learning systems*, 34(12):9587–9603, 2022.

[12] Vincent A TRAAG, Ludo WALTMAN et Nees Jan VAN ECK : From louvain to leiden : guaranteeing well-connected communities. *Scientific reports*, 9(1):1–12, 2019.