

Rapport de Parcours Recherche : Multi-Task Federated Learning

Théo Leleu
Supervisé par Christophe Cerisara
LORIA
Mines de Nancy

29 Mai 2020



Contents

1	Remerciements	2
2	Introduction	2
3	Une rapide présentation du Federated Learning	2
3.1	Le Federated Learning	2
3.2	Approche de Google	3
3.3	Défis de l'apprentissage fédéré	4
3.4	Algorithmes usuels en machine learning	4
3.4.1	Descente de Gradient Stochastique (SGD)	4
3.4.2	Ensemble Averaging	5
4	Federated Curvature	6
4.1	La pénalisation élastique	6
4.2	L'algorithme de Federated Curvature [1]	7
5	Une modélisation sur des données générées	9
5.1	Une modélisation en pytorch	9
5.2	Étude sur trois groupes de données générées	9
6	Étude sur des tâches d'analyse de sentiments	14
6.1	Présentation de Senteval	14
6.2	Installation de Senteval	15
6.3	Étude sur trois sous-tâches	15
7	Conclusion	20

1 Remerciements

Tout d'abord, j'adresse mes sincères remerciements à mon tuteur, M. Christophe Cerisara qui m'a accompagné et aidé durant tout ce parcours jusqu'à la rédaction de ce rapport.

2 Introduction

Le domaine du machine learning est en pleine transformation et toujours en quête de nouvelles données pour son évolution. Des méthodes innovantes émergent pour améliorer les modèles, les rendre plus modulables et plus respectueux de la protection des données. C'est notamment le cas pour les méthodes d'apprentissage fédéré dont nous allons traiter. Ils utilisent des modèles de calculs directement sur les clients sans transmission des données et vont transmettre uniquement le fruit de leur apprentissage au serveur de calcul. Après une rapide présentation des caractéristiques de ces méthodes d'apprentissages, nous allons tâcher d'analyser les caractéristiques d'un algorithme d'apprentissage fédéré nommé *Federated Curvature* ou *FedCurv*⁸ présenté par Edgify en Octobre 2019. Il est décrit par l'auteur de l'article comme le meilleur modèle pour permettre une minimisation globale de la perte sur les modèles dans le cas fédéré et cela même sur des données non identiquement distribuées. Cela est rare pour un programme d'apprentissage fédéré car souvent ils ont des difficultés à atteindre de bons résultats en pourcentage et temps de calcul. Ainsi, nous verrons comment l'algorithme de *FedCurv* se comporte sur des tâches de données artificielles générées puis dans des cas concrets d'analyse de sentiment. L'étude d'analyse de sentiments fera appel à l'outil modulable Senteval conçu par Alexis Conneau et Douwe Kiela en 2018 qui nous permettra d'avoir un comparatif précis. Nous travaillerons en particulier sur des études de commentaires afin de savoir s'ils sont positifs ou négatifs.

Au travers de ce travail, nous étudierons la robustesse et la qualité de cet algorithme et tâcherons de voir s'il peut être supérieur aux autres. Nous comparerons notamment cette méthode à l'algorithme de *Federated Averaging*, aussi appelé *FedAVG* qui lui est proche.

3 Une rapide présentation du Federated Learning

3.1 Le Federated Learning

Le Federated Learning comme nous avons pu le présenter en introduction est une méthode de machine learning qui permet de partitionner l'apprentissage entre plusieurs appareils distincts. Le serveur envoie les informations, en reçoit et combine toutes ces informations pour réaliser un modèle final appris. Il évite la transmission des informations directement au serveur qui pourraient être confidentielles. Tous ces modèles peuvent être fusionnés de manière chiffrée afin de limiter les informations du modèle ou en tout cas d'empêcher l'identification de sa source. Il est possible d'envisager différentes méthodes de transmissions entre clients et serveur qu'elles soient synchrones ou asynchrones. Cela doit limiter l'usage à des supercalculateurs qui réalisent tous les calculs, mais plutôt encourager à l'usage de puissance de calculs non utilisées. Le problème étant que les appareils connectés peuvent être différents à chaque instant et ne sont connectés que pendant une période donnée pouvant être courte. Il faut donc que les cycles d'apprentissage soient relativement courts et qu'un autre appareil puisse y être intégré à tout moment.

En résumé, elle présente de nombreux avantages significatifs :

- Elle est très utile en matière de sécurité puisqu'elle n'oblige pas les clients à fournir ses données. Les modèles clients peuvent être mixés sur le serveur de sorte que l'on ne puisse pas identifier l'émetteur.
- Chaque appareil peut apprendre directement en local. Le client fournit des tâches à effectuer et peut corriger les labels résultants en cas d'erreur.
- Elle est aussi avantageuse en matière de consommation en répartissant les puissances de calculs sur de nombreux appareils.
- Elle permet de réaliser des modèles plus intelligents en faisant collaborer les données de chacun.
- Pas de latence pour l'utilisateur qui a le modèle sur son appareil et peut l'utiliser.
- Elle peut servir à l'optimisation de paramètres comme d'hyperparamètres du modèle.

Elle porte son origine dans un article de Google en 2016[11] qui présente la méthode de Federated Learning. Cette approche est alors présentée avec le schéma suivant :

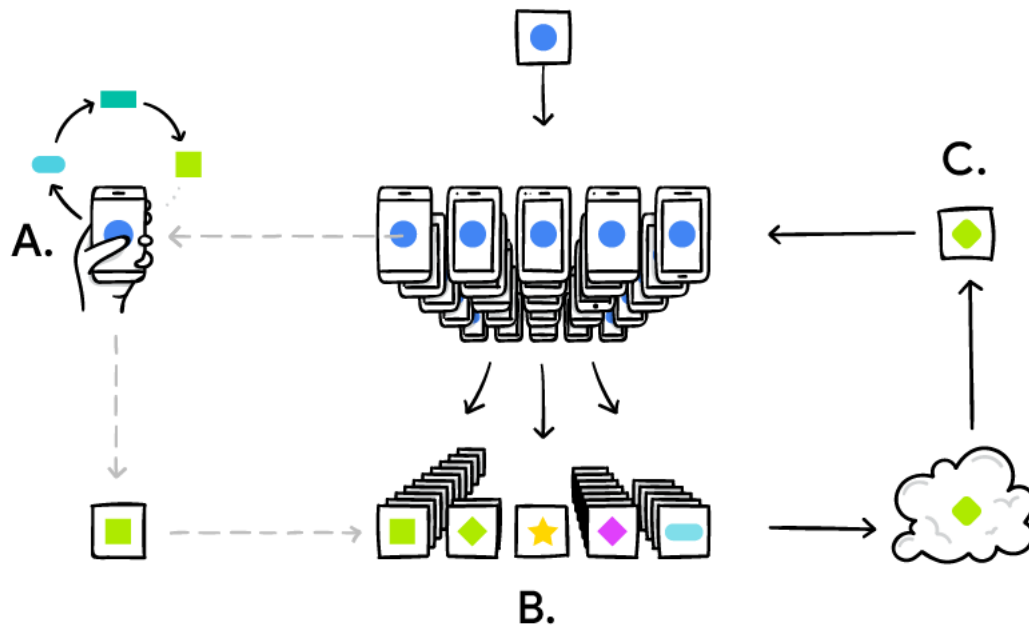


Figure 1: Représentation de Google du Federated Learning

Nous voyons au centre les appareils utilisant un modèle. En A, l'utilisateur peut corriger la suggestion si la recommandation de l'algorithme n'est pas adaptée ou l'accepter.

En B, l'algorithme dispose de tous les résultats appris qu'il va synthétiser dans le nuage afin de renvoyer un modèle unique final en C. Celui-ci sera utilisé par les appareils pour recommencer le processus.

3.2 Approche de Google

L'approche de Google dans le Federated Learning est de faire dialoguer ensemble les appareils Android sans communiquer les informations à l'extérieur. La première idée était d'optimiser les suggestions de mot suivant à travers le clavier Gboard. Chaque partie d'apprentissage se réalise sur l'appareil de l'utilisateur. En cas d'accord de l'utilisateur, le modèle obtenu pourra être utilisé pour enrichir le modèle global de prédiction de Google. Il peut prédire le mot suivant ainsi que des réponses à des questions posées par message. En voici un exemple :

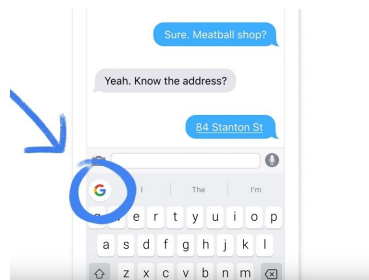


Figure 2: Utilisation du Federated Learning au sein du clavier Gboard de Google.

L'utilisateur cherche l'adresse du restaurant. En utilisant l'outil, il n'a qu'à taper le nom du restaurant pour que l'adresse soit envoyée au destinataire

Avec l'algorithme qui tourne en local, Google est capable d'enregistrer le contexte dans laquelle la requête est effectuée. La mise en place de cette approche sur Gboard a permis d'augmenter la précision de la prédiction de 24%. C'est très significatif !

3.3 Défis de l'apprentissage fédéré

Les défis que peut permettre de relever l'apprentissage fédéré sont donc multiples. Il permet :

- La fusion de modèles de Machine Learning
- La sécurisation des données en réalisant certains calculs directement sur la machine de l'utilisateur
- De réaliser un modèle d'apprentissage sans une collecte de données brutes, mais traitées
- D'utiliser les modèles de chacun pour créer un modèle plus performant En particulier :
- Réduire le nombre de connexions nécessaire entre les différents appareils
- Travailler sur la confidentialité des informations traitées
- Problèmes :
 - Durée de connexion des appareils non régulée et peu stable
 - Possibilité que les appareils soient différents à chaque fois

3.4 Algorithmes usuels en machine learning

Nous présenterons ici les algorithmes nécessaires à l'étude. Ce rapport contrairement au premier se concentre sur l'étude de FedCurv et non sur une vision globale de l'Etat de l'Art.

3.4.1 Descente de Gradient Stochastique (SGD)

C'est la méthode classique de l'apprentissage automatique. Faire une descente de gradient sur toutes les données peut s'avérer coûteux en temps de calcul. La méthode stochastique évalue le gradient sur une seule composante de la somme et réalise donc la descente petit à petit. Soit f la fonction de perte sur l'élément étudié : erreur quadratique moyenne souvent :

$$MSE = \frac{1}{N} \sum_{elem \in E} E(Observation, Prediction)$$

Sur le corpus de test, la pénalisation est appliquée $W_{t+1} = W_t - h_t * f'_i(w_t)$ où $h_t > 0$ est le pas déterminé : fixe, minimisation, section dorée ... La convergence est lente puisqu'il n'y a pas dans la méthode de base d'accélération de convergence. Mais, cette méthode a l'avantage d'être simple et de donner de bons résultats pour l'apprentissage sur une tâche.

Remarque : Il est possible d'accélérer cette convergence en remarquant que si deux gradients successifs sont dans la même direction, il peut être avantageux d'accélérer la vitesse d'apprentissage (learning rate). Lorsqu'ils sont de sens opposé, le minimum commence à se rapprocher, il peut être intéressant de diminuer petit à petit la vitesse d'apprentissage.

Il est possible de représenter la descente sur une composante comme sur la figure 3 ci-dessous.

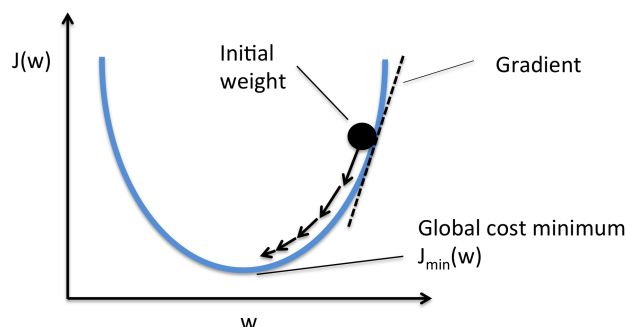


Figure 3: Allure de la descente de gradient suivant une direction donnée

De plus, nous pouvons voir la différence entre descente de gradient et descente de gradient stochastique sur la figure 4.

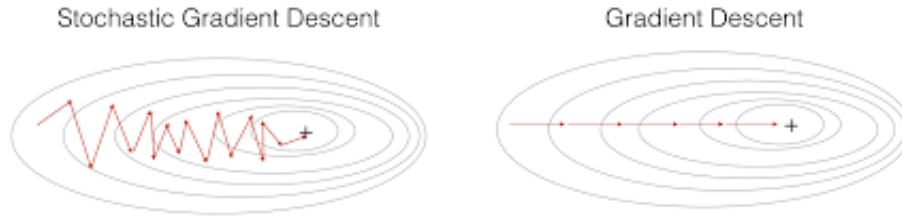


Figure 4: Comparaison entre descente de gradient stochastique et descente de gradient usuelle

Cette méthode permet de se diriger pas à pas vers un point de minimum. Elle se trouve être robuste, mais elle ne peut fonctionner que dans le cas où les objets étudiés sont fortement similaires (même tâche) afin la méthode ne conduise pas à désapprendre ce qui a été appris.

3.4.2 Ensemble Averaging

Cette seconde méthode est indispensable à la compréhension des algorithmes de federated learning que nous utiliserons. L'ensemble averaging consiste à faire une moyenne des poids des modèles qui ont été appris indépendamment afin d'obtenir un modèle global. Cela nous permet de récupérer un modèle qui a appris sur des données et de transmettre le modèle au lieu de devoir transmettre directement les données au serveur. De plus, nous obtenons une distribution résultat qui est proche des modèles d'entrée. Cela n'empêche pas qu'elle peut être dirigée dans une direction peu avantageuse pour limiter la perte globale. Je pense que c'est notamment le cas lorsque la fonction de perte est fortement anisotrope. La formule d'un ensemble averaging est pour des clients i à l'étape t : $\theta_t = \frac{1}{N} \sum_{j \in S \setminus S} \hat{\theta}_{t-1,i}$

C'est, comme son nom l'indique, tout simplement faire la moyenne des coefficients des modèles. Chaque tâche initiale peut alors apporter un surplus d'information au modèle moyenné ainsi, plus le nombre de modèles initiaux est grand plus la représentation sera fidèle.

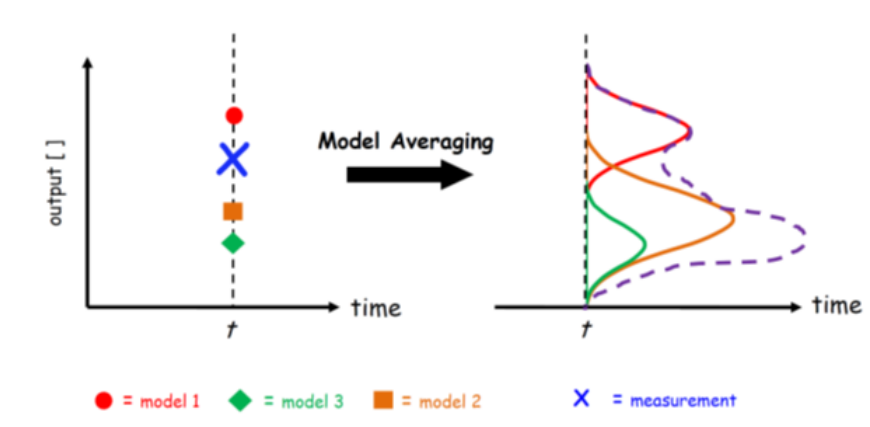


Figure 5: Allure d'une distribution moyenne entre trois distributions sur trois tâches distinctes

Grâce à cet algorithme, nous pouvons avoir une vision globale des zones intéressantes où les différentes tâches ont une valeur d'action importante. Il faut faire attention à moyenner des tâches similaires et capables d'apporter un surplus d'information. Par exemple, ici si nous sommes dans la zone où la valeur est très grande pour la tâche jaune et pour la tâche verte, elle pourra être extrêmement mal représentée pour la tâche rouge. (Voir figure 5) Il faut alors se demander si la tâche rouge représente bien la même caractéristique que les tâches jaunes et vertes. Dans le cas contraire, ces tâches ne devraient pas être moyennées ensemble.

Cette méthode est avec SGD à l'origine de la méthode de Federated Averaging. C'est celle qui est à l'origine du concept de Federated Learning[11].

4 Federated Curvature

Voici l'algorithme central de l'étude. Il s'agit d'un programme d'apprentissage fédéré de Lifelong Learning qui constitue l'idée d'apprendre sur plusieurs tâches sans oublier les précédentes et ainsi d'avoir un modèle qui puisse reconnaître plusieurs éléments simultanément. Cette approche utilise une méthode de pénalisation qui a été étudiée de manière très efficace par Kirkpatrick al. avec leur algorithme nommé *Elastic Weight Consolidation* qui présentait une pénalisation élastique permettant de pénaliser une évolution dans le mauvais sens pour les tâches précédemment apprises. Ainsi, une fois que le modèle est proche de l'optimum pour une tâche, il reste à proximité pour chercher l'optimum de la tâche suivante.

4.1 La pénalisation élastique

Cette pénalisation a été mise en place dans l'algorithme *Elastic Weight Consolidation*[4]. Typiquement, le produit de la valeur de la dérivée seconde au point d'optimum supposé du modèle sur la tâche seule par le carré de la distance à celle-ci est un bon indicateur pour évaluer l'éloignement de l'optimum sur une tâche. En effet, en statistique, pour un nombre suffisant d'observations (x_i) indépendantes et $\hat{\theta}$ l'estimateur du maximum de vraisemblance θ il est possible d'écrire (DL à l'ordre 2) :

$$L(\theta) = L(\hat{\theta}) - \frac{(\theta - \hat{\theta})^2}{2} I(\hat{\theta}) + o((\theta - \hat{\theta})^2)$$

Une telle pénalisation permet en tirant le modèle vers les zones de faible perte de se maintenir à proximité de l'optimum. Elle permet de ne pas s'éloigner de l'optimum et donc de ne pas oublier les modèles précédemment appris.

Graphiquement, cela donne :

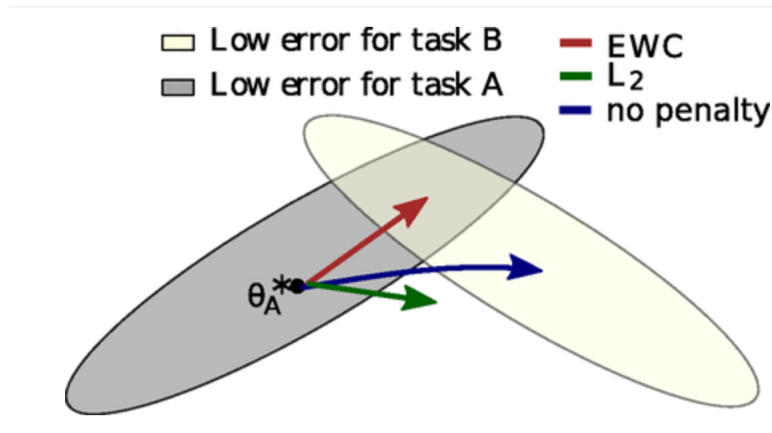


Figure 6: Figure d'illustration pour montrer les propriétés d'EWC

L'algorithme EWC pénalise plus fortement les zones de fortes erreurs. Ainsi, cela permet à l'algorithme de converger vers des zones communes de faible erreur.

C'est extrêmement pratique dans le cadre d'un algorithme qui vise à diminuer l'erreur sans chercher à trouver un minimum absolu pour chacune des tâches

Cela doit alors permettre de rapprocher le modèle d'une zone commune où les deux tâches sont correctement apprises et maîtrisées. Cette zone n'est pas forcément proche des précédents minima sur les tâches, mais les valeurs sont minimisées globalement. Ce graphique illustre le fait que les faibles erreurs sur les tâches ont tendance à se situer à une proximité relative des maximums globaux. C'est ce qui tend à encourager l'usage d'une méthode de moyennage pour avoir une zone intéressante initiale avant l'apprentissage. Cette pénalisation s'écrit alors : $\tilde{L}_B(w) \approx L_B(w) + \lambda(w - \hat{w}_A)^T H_{L_A}(w - \hat{w}_A)$ avec $H_{L_A} = \frac{\partial^2 L_A}{\partial x_i \partial x_j}$

Remarque : Dans l'article de Kirkpatrick et al [4], l'algorithme multitâche avec cette pénalisation apparaît très résistant contre l'oubli de tâches précédemment apprises. Cela pourra nous être utile dans le cas fédéré où de nombreuses informations sont apprises simultanément.

4.2 L'algorithme de Federated Curvature [1]

L'algorithme de FedCurv présenté par Edgify dans son article d'octobre 2019[1] a pour but de maintenir les propriétés intéressantes du Federated Averaging. Elle permet de réaliser un apprentissage et d'y ajouter les avantages de la pénalisation élastique d'EWC.

Le Federated Averaging est un algorithme qui consiste à chaque itération en partant d'un modèle initial sur le serveur à effectuer :

- Copie n fois le modèle et envoie une version à chaque client
- Chaque client apprend avec ses données (en SGD par exemple)
- Le client renvoie son modèle appris au serveur
- Le serveur fait une moyenne des modèles reçus

Les clients peuvent se déconnecter à tout moment. C'est pourquoi il est important de constater que si un client ne renvoie pas son modèle, il y a une perte d'information, mais le modèle résultat reste correct. La convergence de cet algorithme a été prouvée théoriquement sur des données non indépendantes et identiquement distribuées (IID), [3] mais il y a tout de même une baisse de performance constatée lorsque le nombre d'itérations devient trop grand. C'est ce problème que nous tâcherons de résoudre en étudiant l'algorithme qui suit. En effet, cette méthode a tendance à pouvoir se stabiliser assez rapidement. C'est problématique pour apprendre précisément sur un grand nombre d'itérations. Dans ce cas, Federated Curvature pourra être intéressant.

Remarque : Dans la preuve de convergence de Federated Averaging, l'auteur remarque les facteurs de dépendances à la vitesse de convergence. Il trouve :

- E : Si le nombre d'itérations local en une epoch de SGD est trop grand, la convergence peut être plus lente. Il faut donc bien le choisir. (Ce phénomène n'est pas observé sur Mnist équilibré)
- K : Un grand nombre d'appareils améliore légèrement la convergence

De plus, le learning rate doit décliner durant l'apprentissage pour permettre une convergence parfaite.

Les problèmes à résoudre pour implémenter Federated Curvature furent ceux-ci :

- Limiter les oscillations sur l'apprentissage que peuvent causer le Federated Averaging
- À l'aide de la pénalisation élastique, attirer l'apprentissage sur des zones judicieuses pour minimiser globalement les pertes sur l'apprentissage

L'algorithme de Federated Curvature mixe alors les idées de deux algorithmes Federated Averaging et Elastic Weight Consolidation pour combiner leurs avantages. Il rajoute, en partant de Federated Averaging, une pénalisation croisée entre les clients pour tâcher de maintenir le modèle à un niveau de perte relativement proche de celle du modèle initial.

Il est possible de résumer une itération comme cela :

Soit H_t^j la diagonale de Fisher de la négative log vraisemblance (ou hessienne) à l'instant t et pour le client d'indice j. Elle dépend de chaque client qui a son propre modèle et de l'itération dans laquelle il la calcule. Elle est calculée directement par le client qui la renvoie. Nous avons donc à chaque itération :

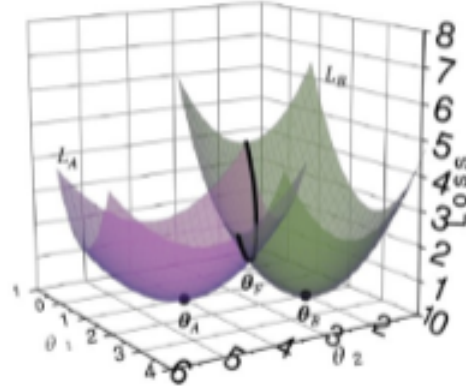
Sur le serveur :

1. Envoyer $\sum_{j \in S \setminus s} \text{diag}(H_{L_{t-1}^j}), \sum_{j \in S \setminus s} \text{diag}(H_{L_{t-1}^j})w_{t-1}^j, w_t$ aux clients
2. Récupérer w_t^k and $\text{diag}(H_{L_t^k})$ for $k=1, \dots, n$
3. Mettre à jour les poids selon la formule $w_t = \frac{1}{n} \sum_{k=1}^n w_{t-1}^k$

Sur la machine du client, effectuer :

1. Pour E itérations, faire : $w_t^k = w_{t-1}^k$:
 $w_t^k := w_t^k - \epsilon \times \sum_j \nabla L_t^k(D_j)$ où D_j sont les données du client et
 $L_t^k(D_j) = L^k(D_j) + \lambda \sum_{j \in S \setminus k} (w - w_{t-1}^j)^T \text{diag}(H_{L_{t-1}^j})(w - w_{t-1}^j)$

On multi-clients model,
we want the results of client distincts but, not too far.



Let's use a penalty method to force that :

$$\tilde{L}_B(w) \approx L_B(w) + \lambda(w - \hat{w}_A)^T H_{L_A}(w - \hat{w}_A) \text{ with } H_{L_A} = \frac{\partial^2 L_A}{\partial x_i \partial x_j}$$

Figure 7: Représentation des valeurs de pénalisations sur les deux tâches en fonction de deux coordonnées caractéristiques

2. Renvoyer w_t^k et $\text{diag}(H_{L_t^k})$

C'est donc pour le client un SGD, avec une fonction de perte qui inclut une pénalisation par rapport aux autres clients. Nous réalisons un moyennage à chaque étape du fonctionnement afin de permettre d'obtenir le modèle du serveur. Pour cela, le serveur doit recevoir les poids des modèles clients. Pour le calcul de la pénalisation, le client doit transmettre la diagonale de Fisher par rapport à ces poids. Ces diagonales doivent être ensuite transmises aux clients pour qu'ils puissent calculer la pénalisation qui les impacte. Afin d'éviter de transmettre toutes ces matrices de Fisher, l'erreur peut s'écrire par développement : $\hat{L}_{t,s}(\theta) = \hat{L}_{t-1,s}(\theta) + \lambda \theta^T \sum_{j \in S \setminus s} \text{diag}(\hat{\tau}_{t-1,j}) \theta - 2\lambda \theta^T \sum_{j \in S \setminus s} \text{diag}(\hat{\tau}_{t-1,j}) \hat{\theta}_{t-1,j} + \text{const}$

Il n'y a alors plus que deux sommes calculées sur le serveur et les poids à transmettre aux clients à chaque étape pour accélérer la transmission et le calcul.

Afin de bien visualiser le fonctionnement de cette pénalisation, nous avons disposé un schéma qui montre la force de la pénalisation sur un modèle en cours d'apprentissage (Figure 7)

Sur cette schématisation, il nous est possible de remarquer que durant l'apprentissage sur le client B, la perte sur le client A est maintenue raisonnable par la pénalisation. La même chose est faite pour tous les clients ce qui permet d'éviter des erreurs trop importantes par oubli des exemples appris par les autres clients.

Les caractéristiques espérées de ce modèle et ses avantages sont multiples :

- Il devrait conserver les propriétés de convergence sur des données non IID du Federated Averaging⁴
- Il devrait éviter l'oubli des données précédemment apprises
- Il devrait converger plus rapidement et avec un bruit plus faible grâce à la pénalisation élastique qui lui est appliquée⁴

Cependant, j'ai pu douter de cette méthode de pénalisation. En effet, à chaque étape, le modèle est fusionné selon les clients et chaque client ne peut donc pas "trop" s'éloigner de ce modèle. Par conséquent, l'importance relative de cette pénalisation pourrait être faible. Il est possible de moduler cette importance en jouant avec le facteur λ , coefficient multiplicateur devant la pénalisation.

Remarque : Il existe d'autres frameworks pour améliorer la convergence de FedAvg comme FedProx présenté par Li al [13] qui permet notamment d'effectuer un nombre variable d'itérations locales sur les clients.

5 Une modélisation sur des données générées

Pour vérifier ces caractéristiques espérées, nous avons commencé par créer des données assez simples et vérifié que le comportement de l'algorithme était conforme aux attentes. Nous avons décidé d'opérer sous Pytorch qui est assez modulable et proche des outils de base de python dans son usage.

5.1 Une modélisation en pytorch

J'ai souhaité modéliser sur une machine les instructions fédérées. C'est-à-dire que dans le cas de la simulation les calculs seront réalisés sur ma machine, mais de manière indépendante comme s'il s'agissait de machines distinctes. Le nombre d'étapes ainsi que le fonctionnement global de l'algorithme sont inchangés par rapport à la méthode fédérée réelle. Cependant, cela ne nous permettra pas de discuter de la répartition des tâches entre les différentes machines d'un réseau et donc du temps effectif de calcul en conditions réelles.

Pour cette modélisation, il a fallu faire une modélisation d'un modèle SGD disposant d'un corpus de développement qui permette de voir la qualité de l'apprentissage. Ensuite, j'ai ajouté la méthode de moyennage sur les modèles appris individuellement en fin d'itération. Enfin, la pénalisation élastique qui nécessite le calcul de la norme 2 du gradient à l'aide des méthodes de Pytorch.

Cette modélisation nous permettra de réaliser un environnement propice pour commencer à mettre en oeuvre l'algorithme de *FedCurv* ainsi que d'en vérifier les caractéristiques générales sur des tâches relativement simples.

5.2 Étude sur trois groupes de données générées

Afin de visualiser le fonctionnement de l'algorithme, nous simulerons des données dans R^2 avec trois groupes de points répartis autour d'une moyenne et d'une variance. Chaque groupe représente une tâche à apprendre. Chaque groupe de point est divisé en quatre couleurs qui sont quatre espèces à classifier. Nous pouvons visualiser les points comme suit :

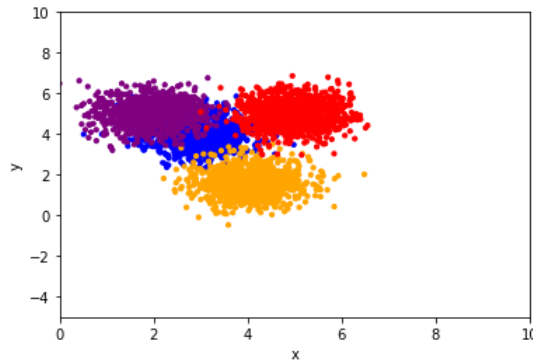
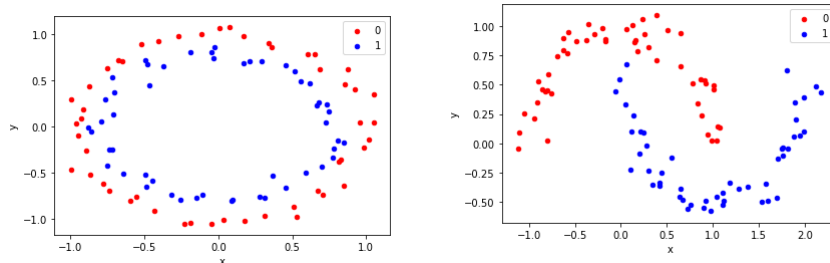


Figure 8: Nous classifions ces points de R^2 selon quatre couleurs distinctes identifiées

Remarque : Nous aurions pu étudier d'autres formes pour voir les différences au niveau de l'apprentissage, par exemple⁷ :



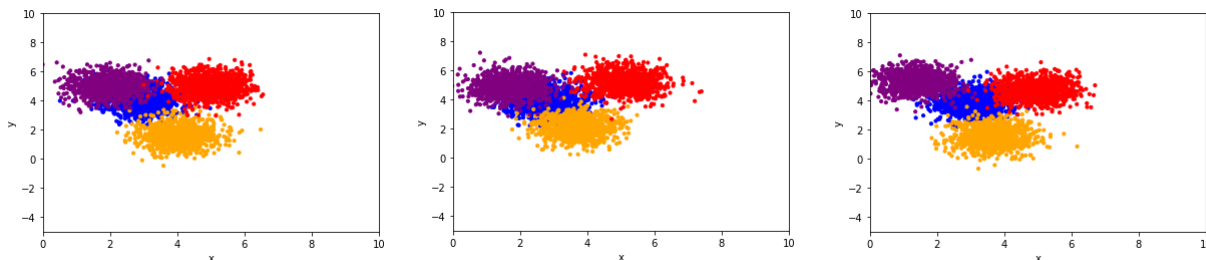
Pour cette étude, nous ferons tourner l'algorithme de *FedCurv* sur une machine unique et simulerons son fonctionnement. Pour cela nous avons dû développer les trois outils principaux pour avoir un algorithme fonctionnel :

- L'apprentissage SGD sur le client
- Le model averaging
- Le calcul de la matrice d'information de Fisher

Afin de réaliser la matrice d'information de Fisher, il faut :

- Initialiser la matrice de la forme des paramètres du modèle.
- Calculer la négative log vraisemblance.
- Calculer le carré du gradient de la negative log likelihood in extenso, l'information de fisher.

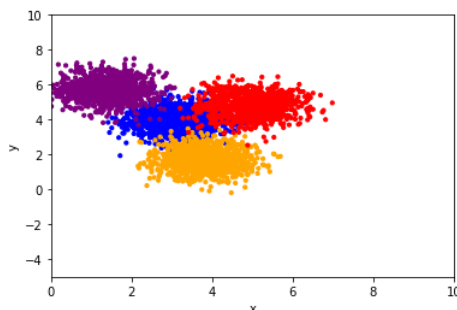
Pour des contraintes pratiques, nous réaliserons les mêmes opérations de la même manière que sur l'algorithme fédéré sauf que l'intégralité des calculs est réalisée de manière séquentielle sur la machine. Les trois tâches pour l'apprentissage sont :



Les 3 tâches d'apprentissage avec les quatre catégories de points en fonction des couleurs

Chaque colonne représente une catégorie. Les tâches sont choisies assez proches. Nous verrons si la classification selon les quatres catégories de points peut se réaliser correctement. Regardons si le résultat fédéré est meilleur que le résultat des tâches prises individuellement.

La tâche de test choisie est :



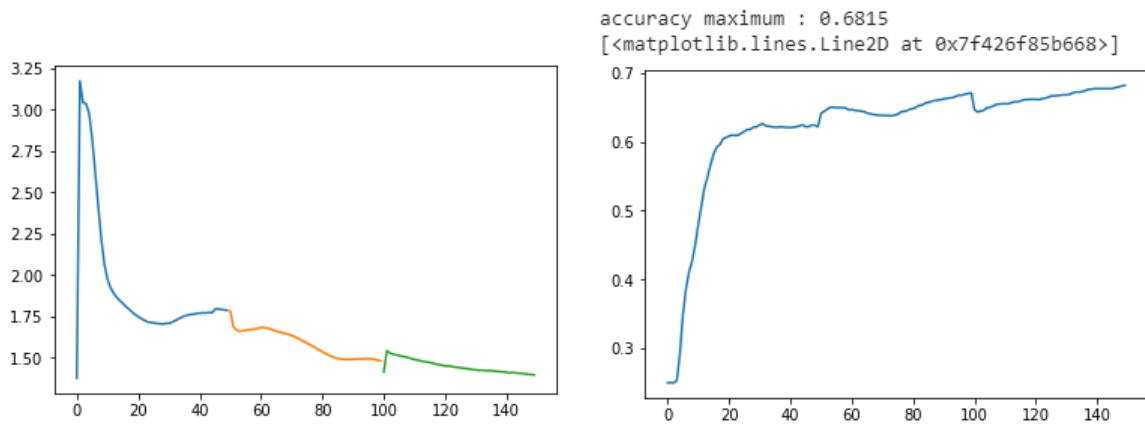
Tâche de test avec les quatres catégories de points

Chacune de ces tâches sera apprise sur différentes méthodes afin de voir leurs caractéristiques.

Montrons d'abord les résultats avec l'algorithme initial / moyennage + pénalisation : Tâchons d'étudier l'impact de lambda, facteur multiplicatif devant la pénalisation pour voir s'il joue un rôle dans l'évolution de l'accuracy sur les différentes tâches.

Prenons en référence l'apprentissage en multi tâche SGD, avec un apprentissage successif sur les trois tâches, le résultat obtenu est :

Soit la perte sur le corpus d'apprentissage (à gauche) et la précision sur le corpus de test (à droite) en fonction du nombre d'itérations:



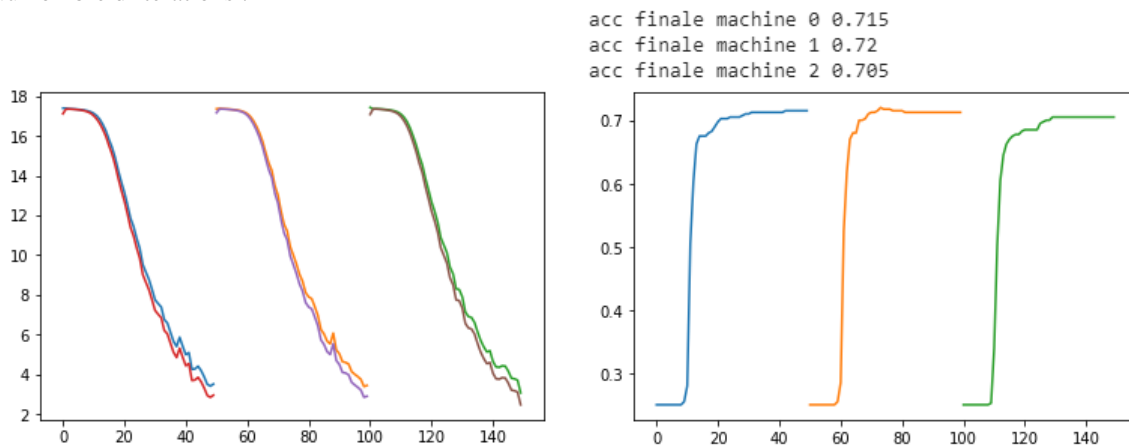
Multi Task SGD Représentation de la perte sur le corpus d'apprentissage puis de la précision sur le corpus de test en fonction du nombre d'itérations effectuées pour les trois tâches présentées apprises successivement (bleu, jaune, vert).

70 % des éléments sont correctement reconnus sur le corpus de test. Dans chaque graphique la tâche la plus à gauche sera la tâche 1, celle au centre la tâche 2 et celle de droite la tâche 3.

Impact de lambda sur l'apprentissage

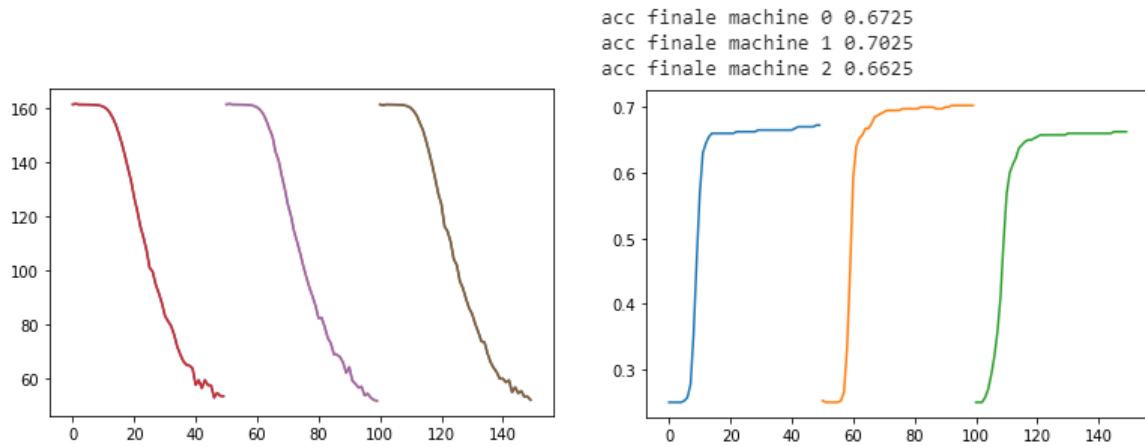
Avec $\lambda = 1$:

Soit les pertes sur le corpus d'apprentissage (à gauche) et la précision sur le corpus de test (à droite) en fonction du nombre d'itérations :



Federated Curvature en séquentiel : Résultat de la perte sur le le corpus d'apprentissage(bleu, jaune, vert) et de test (rouge, violet, marron) (à gauche) et de précision sur le corpus de test (à droite) en fonction du nombre d'itérations pour $\lambda=1$

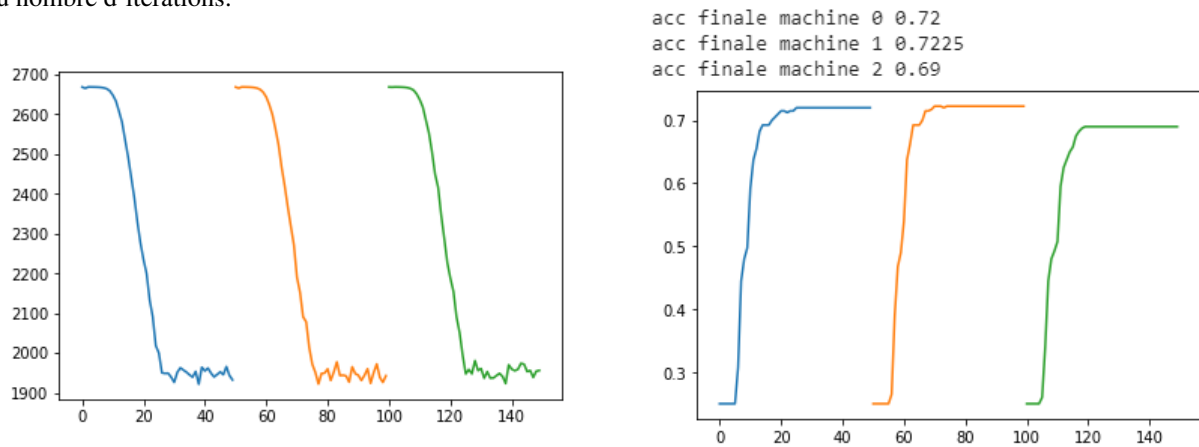
Avec $\lambda = 10$: Soit la perte sur le corpus d'apprentissage (à gauche) et la précision sur le corpus de test (à droite) en fonction du nombre d'itérations:



Federated Curvature en séquentiel : Résultat de la perte sur le corpus d'apprentissage (à gauche) et de la précision sur le corpus de test (à droite) en fonction du nombre d'itérations pour $\lambda=10$

Avec $\lambda=100$:

Soit la perte sur le corpus d'apprentissage (à gauche) et la précision sur le corpus de test (à droite) en fonction du nombre d'itérations:

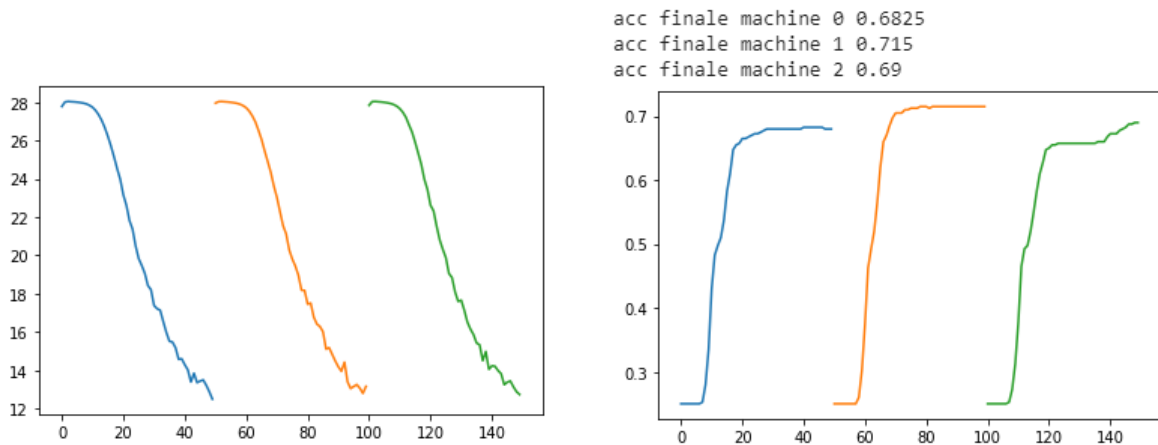


Simulation de Federated Curvature : Résultat de la perte sur l'apprentissage (à gauche) et de précision sur le corpus de test (à droite) en fonction du nombre d'itérations pour $\lambda=100$

Première constatation, la perte sur le corpus d'apprentissage et de test sont très proches. L'apprentissage se réalise correctement. Plus λ augmente, plus la vitesse de convergence est importante. La pénalisation élastique semble efficace pour attirer le modèle vers un optimum commun aux différentes tâches. Cependant, elle ne permet pas probablement à cause du recouplement des tâches d'améliorer l'apprentissage. En tout cas, celui-ci est très rapide. Les tâches choisies sont assez simples.

Vérifions le rôle du moyennage sur la méthode étudiée. Faisons un moyennage jusqu'à la moitié de l'apprentissage afin d'éviter une place trop forte pour la pénalisation sans convergence efficace puis uniquement la pénalisation pour le maintien de la probabilité d'oubli sur les tâches :

Pour la perte sur les corpus d'apprentissage (à gauche) et la précision sur le corpus de test (à droite) en fonction du nombre d'itérations d'apprentissage, nous avons :

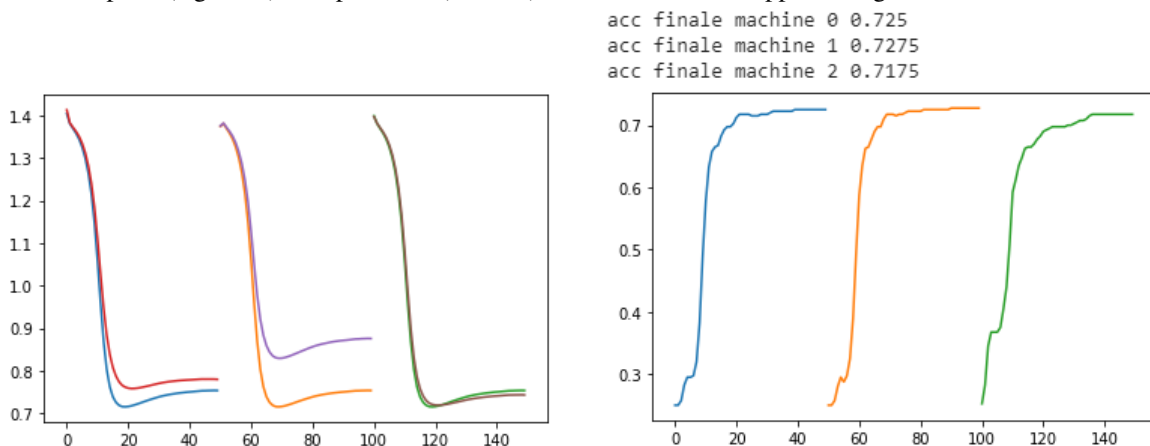


FedCurv avec le moyennage jusqu'à la moitié de l'apprentissage uniquement: Perte sur le corpus d'apprentissage et précision sur le corpus de test en fonction du nombre d'itérations

Dès que le moyennage s'arrête au milieu de l'apprentissage, nous pouvons constater la précision n'augmente plus uniformément sur les 3 tâches, mais tend à privilégier la tâche 2. Le moyennage est donc extrêmement important dans le cadre de l'algorithme pour assurer une optimisation équitable sur les trois tâches.

Avec le multi task SGD + Averaging :

Sur la perte (à gauche) et la précision (à droite) du modèle durant l'apprentissage:



SGD avec averaging : Résultat de la perte sur le corpus d'apprentissage(bleu, jaune, vert) et de développement (rouge,violet, marron) (à gauche) et la précision sur le corpus de test(à droite) en fonction du nombre d'itérations

Sans la pénalisation, il y a une lente remontée sur le corpus d'apprentissage. C'est le Federated Averaging ne permet plus d'obtenir de meilleur résultat. En effet, dès lors que le nombre d'itérations est important, l'article[1] précise que les performances peuvent se dégrader au niveau de l'apprentissage notamment pour FedAVG.

L'algorithme de SGD avec l'averaging est efficace, mais la pénalisation permet d'accélérer la convergence et de maintenir l'apprentissage à un bon niveau. Il y a pas d'oubli des tâches précédemment apprises. La convergence est très rapide. Le facteur lambda est choisi pour modifier la prédominance de la pénalisation. Avec lambda qui vaut 100, le résultat donnant 0.72 est plutôt bon. Pour autant, l'impact de cette pénalisation n'apparaît pas si important que cela notamment en matière de précision où il n'y a pas de distinction majeure finalement.

Si ce n'est la vitesse d'apprentissage, ce corpus ne nous permet pas de visualiser réellement un avantage de l'algorithme de FedCurv par rapport aux autres méthodes notamment FedAVG.

6 Étude sur des tâches d'analyse de sentiments

Il est maintenant nécessaire d'étudier des tâches réelles pour visualiser réellement comment l'algorithme réagit dans le cadre d'une étude de données réelles.

6.1 Présentation de Senteval

SentEval est une boîte à outils mise à disposition par Facebook pour évaluer la qualité d'une méthode de représentation des mots dans un espace de dimension donnée. Senteval inclut une diversité de tâches, permettant de faire de l'apprentissage mono et multi tâches, de l'étude de langages naturels ainsi que l'étude de similarités dans les phrases. Cet outil vient avec des scripts et des méthodes déjà implémentées pour faciliter l'étude de ces corpus en modifiant uniquement la partie que l'utilisateur souhaite étudier (modèle, algorithme, méthode de représentation...) Il permet de faire avancer l'étude vers la recherche d'une meilleure représentation des mots et de meilleures méthodes de traitement de ceux-ci en fournissant des outils efficaces pour l'étude. Les données d'évaluations sont classiques, des méthodes d'optimisations sont proposées et l'algorithme Python est facilement modifiable. Cela permet de réaliser des études dans des conditions permettant une comparaison parfaite avec les résultats précédemment obtenus et fournis dans l'article. Je peux donc espérer évaluer de manière comparative la qualité de mon implémentation par rapport aux résultats précédemment obtenus par les auteurs.

Pour traiter le dictionnaire de vecteur contenant les mots, plusieurs méthodes sont proposées dont "BagOfWord" et "InferSent" notamment.

- BagOfWord est une technique qui prend les mots de la représentation nécessaire à l'étude dans un espace donné et les regroupe dans un sac pour l'étude.
- InferSent étudie l'intégration des phrases pour fournir une représentation sémantique à différentes phrases. Il se base sur les données d'inférence entre les mots et est efficace sur différentes tâches de NLP.

J'utiliserai ici BagOfword pour mon étude sur les tâches. En effet, celui-ci était plus simple à configurer pour effectuer l'étude.

Voici les tâches pour réaliser l'apprentissage comme présenté par l'auteur de Senteval dans son article⁸:

name	N	task	C	examples	label(s)
MR	11k	sentiment (movies)	2	"Too slow for a younger crowd , too shallow for an older one."	neg
CR	4k	product reviews	2	"We tried it out christmas night and it worked great ."	pos
SUBJ	10k	subjectivity/objectivity	2	"A movie that doesn't aim too high , but doesn't need to."	subj
MPQA	11k	opinion polarity	2	"don't want"; "would like to tell";	neg, pos
TREC	6k	question-type	6	"What are the twin cities ?"	LOC:city
SST-2	70k	sentiment (movies)	2	"Audrey Tautou has a knack for picking roles that magnify her [...]"	pos
SST-5	12k	sentiment (movies)	5	"nothing about this movie works."	0

Table 1: **Classification tasks.** C is the number of classes and N is the number of samples.


name	N	task	output	premise	hypothesis	label
SNLI	560k	NLI	3	"A small girl wearing a pink jacket is riding on a carousel."	"The carousel is moving."	entailment
SICK-E	10k	NLI	3	"A man is sitting on a chair and rubbing his eyes"	"There is no man sitting on a chair and rubbing his eyes"	contradiction
SICK-R	10k	STS	[0, 5]	"A man is singing a song and playing the guitar"	"A man is opening a package that contains headphones"	1.6
STS14	4.5k	STS	[0, 5]	"Liquid ammonia leak kills 15 in Shanghai"	"Liquid ammonia leak kills at least 15 in Shanghai"	4.6
MRPC	5.7k	PD	2	"The procedure is generally performed in the second or third trimester."	"The technique is used during the second and, occasionally, third trimester of pregnancy."	paraphrase
COCO	565k	ICR	sim		"A group of people on some horses riding through the beach."	rank

Table 2: **Natural Language Inference and Semantic Similarity tasks.** NLI labels are contradiction, neutral and entailment. STS labels are scores between 0 and 5. PD=paraphrase detection, ICR=image-caption retrieval.

Figure 9: Description des données présentes du package Senteval par Conneau et Kiela [8]

Il faut alors préparer pour les différentes tâches les mots nécessaires à l'apprentissage dans notre fichier de représentation puis les grouper afin de pouvoir effectuer l'apprentissage selon la tâche et la méthode choisie.

Pour réaliser l'apprentissage sur ces tâches, il est proposé de faire une validation croisée à 5 'folds'. La procédure générale est la suivante :

- Mélanger le jeu de données de manière aléatoire.
- Diviser l'ensemble en 5 groupes
- Pour chaque groupe unique :
 - Prendre le groupe comme un ensemble de tests
 - Prendre les groupes restants comme un ensemble d'apprentissage
 - Monter un modèle sur l'ensemble d'apprentissage
 - Conserver le score d'évaluation et jeter le modèle
- Évaluer les compétences du modèle en faisant une moyenne sur les folds

Cela permet dans notre cas de mettre chaque cinquième tour à tour au niveau du corpus de test afin de vérifier que dans chaque cas l'apprentissage sur les 4 parties permet bien d'en déduire la cinquième. Il ne reste plus qu'à moyenner les résultats trouvés dans les 5 cas.

Dans le cadre de l'étude future, la méthode de représentation fasttext MLP qui est une représentation de vecteurs dans un espace de dimension 300 est choisie. L'étude sera réalisée sur un "fold" pour diminuer le temps de calcul. Calculer tous les folds démultiplierait le nombre de possibilités et donc le temps de calcul. Nous étudierons 4 tâches initialement puis 3 en montrant que la quatrième est décorrélée des autres. L'avantage de ces tâches c'est que l'apprentissage et la gestion des données se réalisent de la même façon sur ces quatre tâches. Les algorithmes sont facilement modifiables ce qui permet de mener l'étude.

Ce sont respectivement :

- CR : Détermine si un produit est de bonne qualité ou non
- MR : Détermine si un commentaire de film est positif ou négatif
- MPQA : Polarité de l'opinion évaluée positive ou négative
- SUBJ : Détermine si l'assertion est subjective ou objective

6.2 Installation de Senteval

Afin d'installer et d'utiliser Senteval, il est nécessaire d'être sur une machine Linux disposant d'un GPU cuda de mémoire suffisante pour effectuer les calculs d'apprentissage correctement. La capacité mémoire de ma machine était suffisante pour BagOfWord, mais semblait insuffisante pour Infsent. (Arrêt par CUDA out of memory error)

6.3 Etude sur trois sous-tâches

Nous traiterons quatre tâches CR, MR, MPQA, SUBJ du package Senteval. Nous vérifierons que les résultats correspondent bien à ceux prévus par l'article. Pour chaque étude la première tâche à apparaître est CR, puis MR, MPQA et SUBJ

Nous choisirons l'algorithme SGD avec un learning rate de 0.1 pour l'apprentissage. Ce learning rate est recommandé par les auteurs et donne en effet, des résultats tout à fait corrects. J'ai d'ailleurs pu constater que $lr=0.01$ ne donne pas de meilleurs résultats. Il existe aussi la méthode RMSprop qui est proposée. Elle permet d'accélérer la convergence de SGD en adaptant efficacement le learning rate durant l'apprentissage. Je ne l'ai pas utilisée ici. En effet, j'ai préféré utiliser les outils les plus simples et normalement si le nombre d'itérations nécessaire est plus grand l'avantage de Federated Curvature sur Federated Averaging devrait être plus visible. Nous utiliserons ici le modèle de vecteurs de représentation GloVe MLP de dimension 300. L'objectif sera ici d'essayer d'obtenir des résultats comparables ou meilleurs à ceux de la littérature en travaillant avec notre algorithme d'apprentissage fédéré depuis le modèle de vecteur GloVe MLP. Premièrement, faisons l'essai de l'apprentissage sur les 5 folds différents avec le SGD. L'apprentissage sur réalise indépendamment sur les différentes tâches, c'est l'algorithme SGD tel qu'il est présenté initialement dans Senteval (figure 10)

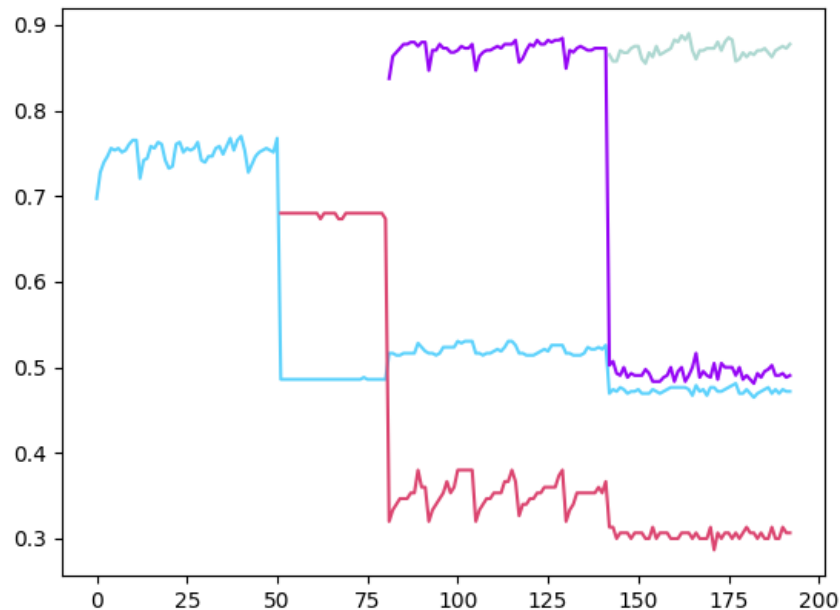


Figure 10: Algorithme de SGD avec 5 folds, $lr=0.1$

Représentation de la précision sur le corpus test en fonction du nombre d'itérations

En bleu, CR est la première courbe qui donne une précision de 70%. Les cinq étapes d'apprentissage associées aux 5 folds sont visibles. En rouge, c'est l'apprentissage sur MR, le taux de réussite est assez faible de 68%. La courbe en violet, la plus haute, est MPQA qui atteint 89 % d'évaluations correctes. Enfin, en bleu gris, la dernière courbe représente SUBJ et atteint 90 % d'évaluations correctes.

Il y a 4 "fold" sur 5 pour l'apprentissage et un pour le test pour chaque tâche. Ici, le premier groupe est choisi comme ensemble de tests et les autres pour l'apprentissage. Seuls les résultats sur MR semblent faibles par rapport aux résultats de l'article qui donne ici des résultats à 0.68 contre 0.78 dans l'article. En effet, l'article donne pour leurs meilleurs résultats obtenus:

- CR : 0.81 contre 0.77 ici
- MR : 0.78 contre 0.68 ici
- MPQA : 0.885 contre 0.88 ici
- SUBJ : 0.93 contre 0.89 ici

SUBJ va être mise de côté puisqu'elle n'apporte pas d'information a priori pour les autres tâches. En effet, elle représente la subjectivité ou non d'une information et non le caractère positif ou négatif de la phrase.

Remarque : Il serait aussi possible de considérer la tâche SST-2 qui détermine aussi si le sentiment sur le film est positif ou négatif. De même pour les différentes tâches SST présentes qui étudient sur 4 ans les similarités entre deux phrases, mais le résultat est un score entre 0 et 5 plus difficile à évaluer qu'avec deux classes positif-négatif.

Voici maintenant les résultats obtenus avec le premier "fold" comme ensemble de tests et les autres pour l'apprentissage sur chacune des tâches.

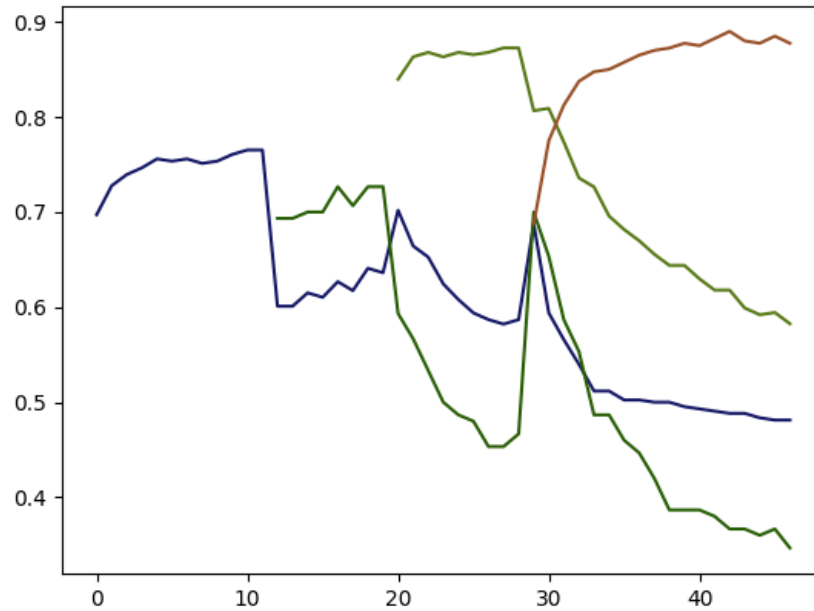


Figure 11: Algorithme SGD multi tâches.

Représentation de la précision sur le test en fonction du nombre d'itérations

En bleu, CR qui atteint 77% . En vert foncé, MR atteint 72 % d'évaluations correctes. En vert clair, la courbe la plus haute est MPQA qui atteint 88 % d'évaluations correctes. Enfin, en marron, SUBJ atteint 88 % d'évaluations correctes sur la tâche.

Sur la figure 11, il y a, au bout d'un moment, une amélioration sur les autres tâches. En effet, en fin d'apprentissage de la troisième tâche nous pouvons voir une remontée significative de la courbe qui montre que cette tâche peut contenir des informations communes avec les tâches précédentes. Ainsi, les trois premières tâches qui qualifient la positivité négativité semblent contenir des informations communes tandis que la quatrième tâche (subjective / objective est décorrélée des 3 autres) Il faut donc se focaliser sur les 3 tâches (CR, MR, MPQA) pour l'apprentissage en parallèle. À la fin de l'apprentissage sur la troisième tâche, le modèle commence à bien reconnaître les deux premières tâches. Nous obtenons 70% de précision sur les deux premières tâches à la fin de l'apprentissage de la troisième tâche.

Rappelons qu'un des objectifs/caractéristiques de l'algorithme de *Federated Curvature* était d'éviter l'oubli de tâches précédemment apprises et de surpasser les résultats de *Federated Averaging* sur un nombre d'itérations important.

Tâchons de montrer que si nous utilisons SGD simultanément sur les trois tâches, l'algorithme oublie rapidement les tâches apprises précédemment. Pour les trois tâches en SGD avec à chaque itération l'apprentissage sur les trois tâches, les résultats sont (en rouge MPQA, en bleu CR et en violet MR) :

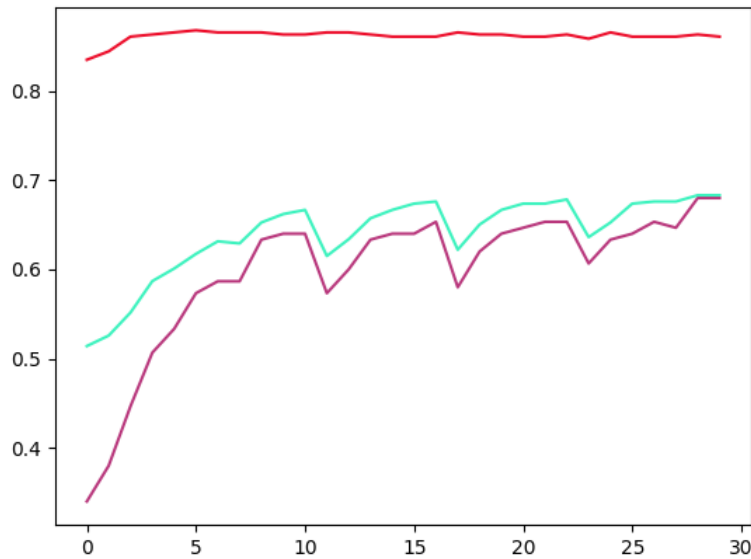


Figure 12: Algorithme séquentiel de SGD avec les trois tâches apprises simultanément. Représentation de la précision sur le corpus de test en fonction du nombre d'itérations. En violet, MR atteint 65% de classifications correctes. En bleu clair, CR fait un tout petit peu mieux avec 68%. En rouge, la courbe la plus haute est MPQA qui atteint 85 % d'évaluations correctes.

À chaque apprentissage dans le cadre de SGD séquentiel, il est possible d'avoir un oubli des tâches précédentes (figure 12). Ainsi, c'est uniquement la troisième tâche de l'itération MPQA est correctement retenue par l'algorithme. Les autres tâches peinent à apprendre et sont oubliées pendant le procédé, nous peinons à avoir 60 % d'évaluations correctes.

Sans le moyennage, les valeurs ne sont pas stabilisées et sans la pénalisation l'oubli des précédentes tâches peut se faire.

Il faut maintenant appliquer le modèle à trois tâches SGD avec moyennage / Federated Averaging ainsi générées :

À chaque étape jusqu'à ce que l'apprentissage ne s'améliore plus (3 fois de suite) ou limite d'itérations :

- Trois copies du modèle sont réalisées
- Chaque tâche est apprise sur un des modèles
- Le modèle est moyenné

Ainsi, testons l'algorithme de Federated Averaging sur ces trois tâches :

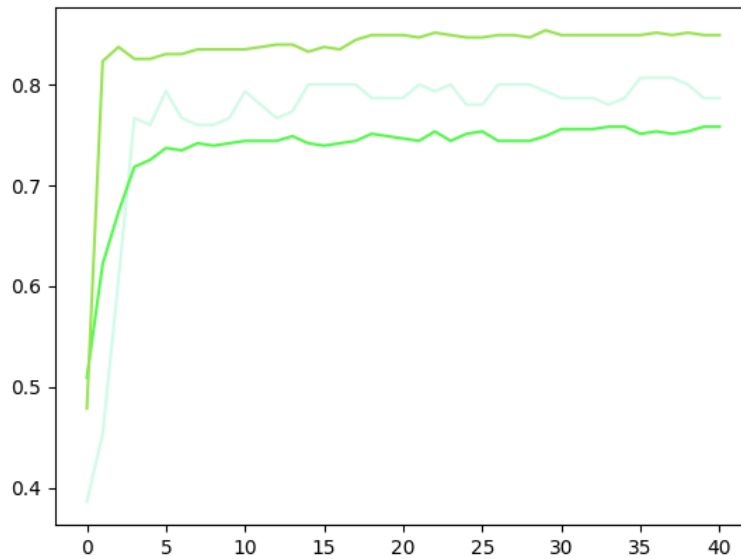


Figure 13: Algorithme séquentiel de Federated Averaging.

Représentation de la précision du corpus de test en fonction du nombre d'itérations d'apprentissage

En bleu clair, les résultats obtenus sur CR montre de légères oscillations et équivalent relativement aux résultats de l'apprentissage mono tâche (76 %). En vert clair en bas, MR donne des résultats lui aussi proches de ceux du SGD sur une tâche soit 72 %. En vert foncé, la courbe la plus haute est MPQA qui atteint 85 % d'évaluations correctes soit de qualité un peu moindre que sur l'apprentissage individuel.

Sur MPQA, l'apprentissage est de moindre qualité que sur un apprentissage unique pour MPQA, mais pour les deux autres tâches les résultats sont relativement bons tout de même. La tâche avec le meilleur taux d'apprentissage est MPQA puis CR et enfin MR.

L'apprentissage sur MR et CR est un peu meilleur avec l'apprentissage multitâche. Par contre, l'apprentissage sur MPQA perd un peu de précision, mais il se stabilise rapidement à un bon niveau. Ce corpus apparaît pertinent pour un apprentissage multitâches. La précision atteint rapidement un plateau. Ce n'est pas la situation idéale pour montrer l'avantage de FedCurv sur FedAVG qui a plutôt tendance à être vraiment avantageux sur le long terme a priori.

Dans le cas de l'algorithme complet avec une pénalisation, il est possible de réaliser la même étude pour observer :

- Une convergence plus rapide ou non de l'algorithme
- Une meilleure résistance au surapprentissage

En réalité, dans ce cadre les résultats sont donnés sur la figure 14 :

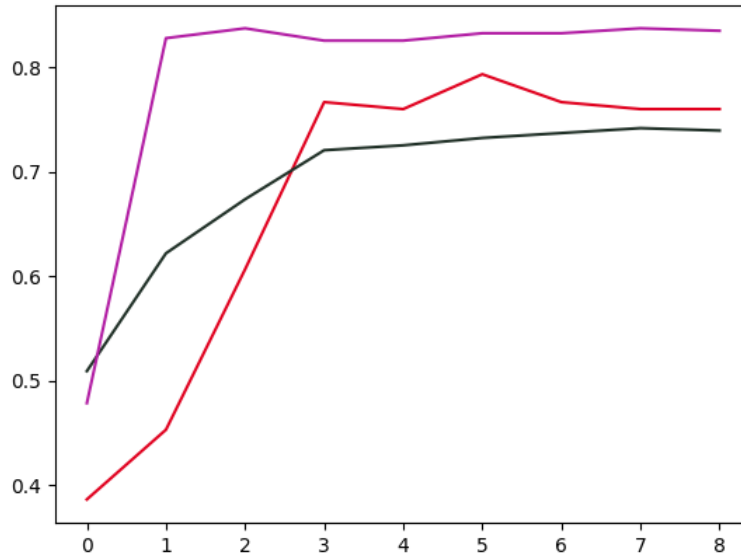


Figure 14: Algorithme séquentiel de Federated Curvature.

Étude de la précision sur corpus de test en fonction du nombre d'itérations sur le corpus d'apprentissage

En rouge, l'évolution de l'apprentissage sur les commentaires de films CR parvient à un taux tout à fait raisonnable de 74%. En noir, l'apprentissage sur MR donne un résultat moindre de 71%. C'est aussi bien que le résultat en SGD. En violet, la courbe la plus haute est MPQA qui dépasse au bout de la première itération les 80% d'évaluations correctes.

Il semble que la convergence avec cette méthode soit assez proche de celle pour la méthode avec moyennage. La convergence est probablement un peu plus rapide et la courbe rouge semble moins "retomber" que dans le cas avec le moyennage, mais ce n'est pas très significatif. Ce graphique ne permet pas d'observer un avantage significatif avec la pénalisation. (figure 14 ci-dessus) Nous n'obtenons pas ici des résultats meilleurs qu'avec les autres méthodes proposées. Probablement que le faible nombre d'itérations ne permet pas de voir le comportement sur le long terme. Cette étude reste partielle et ce sont les premiers résultats que j'ai pu obtenir, mais je pense qu'il est possible d'obtenir des résultats plus significatifs en rallongeant le temps d'apprentissage.

7 Conclusion

Cette étude a permis au travers d'exemples multiples de représenter les caractéristiques de *FedCurv* et d'effectuer une comparaison de cette méthode avec la méthode plus classique de *FedAVG*.

FedCurv permet une convergence rapide en comparaison aux précédents algorithmes proposés. Il est stable et permet de résister à l'oubli des précédentes tâches. C'était un défaut que les algorithmes précédemment évalués pouvaient posséder. Il semble aussi disposer de plus de stabilité que le Federated Averaging dû à la pénalisation qui "tire" rapidement le modèle vers une zone d'étude intéressante et le maintien dans la zone. Par contre, selon l'implantation, le calcul de la pénalisation peut être relativement long notamment sur un modèle avec beaucoup de couches comme celui de Senteval. Il faut donc voir s'il est nécessaire de calculer toutes ces matrices hessiennes et les transmettre pour réduire un peu le temps d'apprentissage et éviter une possible dégradation de l'apprentissage si celui-ci est trop long. Peut-être est-il suffisant de conserver uniquement les composantes majeures de cette matrice. Il faudrait généraliser l'étude sur un vrai système fédéré où les données sont transmises entre les différentes machines. Il serait aussi intéressant d'étudier la vitesse de calcul sur un dispositif fédéré réel et la quantité de données transmises, notamment le comparer avec le Federated Averaging en termes de temps plutôt qu'en matière de nombre d'itérations. Nous aurions aussi pu dans le cas de l'étude sur Senteval augmenter le nombre d'itérations en diminuant le nombre de données par étape pour voir les difficultés qu'aurait FedAVG dans ce cas.

References

- [1] Neta Shoham, Tomer Avidor, Aviv Keren, Nadav Israel, Daniel Benditkis, Liron Mor-Yosef, Itai Zeita, *Overcoming Forgetting in Federated Learning on Non-IID Data*, <https://arxiv.org/pdf/1910.07796.pdf>
- [2] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar and Virginia Smith, *Federated Optimization in Heterogeneous Networks* <https://arxiv.org/pdf/1812.06127.pdf>
- [3] Ismaël Castillo, *Introduction aux Statistiques Bayésiennes*, 2017-2018, http://www.lpsm.paris/pageperso/castillo/bayes/notes_de_cours.pdf
- [4] Kirkpatrick, Pascanu, Rabinowitz, Veness, Desjardins, A. Rusu, Milan, Quan, Ramalho, Grabska-Barwinski, Hassabis, Clopath, Kumaran, Hadsell, *Overcoming catastrophic forgetting in neural networks*, <https://arxiv.org/pdf/1612.00796.pdf>
- [5] Bagdasaryan, Veit, Hua, Estrin, Shmatikov, *How To Backdoor Federated Learning*, <https://arxiv.org/pdf/1807.00459.pdf>
- [6] <https://andrewliao11.github.io/blog/fisher-info-matrix/>
- [7] Jason Brownlee, *How to Generate Test Datasets in Python with scikit-learn*, <https://machinelearningmastery.com/generate-test-datasets-python-scikit-learn/>
- [8] Alexis Conneau and Douwe Kiela, *SentEval: An Evaluation Toolkit for Universal Sentence Representations*, <https://arxiv.org/pdf/1803.05449.pdf>
- [9] Li, Huang, Yang, Wang and Zhang, *On the convergence of FedAVG on non-IID data*, <https://arxiv.org/pdf/1907.02189.pdf>
- [10] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. *Robust and communication-efficient federated learning from non-iid data*, <https://arxiv.org/pdf/1903.02891.pdf>
- [11] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Agueray Arcas. *Communication-Efficient Learning of Deep Networks from Decentralized Data*, <https://arxiv.org/pdf/1602.05629.pdf>
- [12] <https://github.com/yashkant/Elastic-Weight-Consolidation>
- [13] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, Virginia Smith. *Federated Optimization in Heterogeneous Networks*, <https://arxiv.org/pdf/1812.06127.pdf>