GRADIENT ASSISTED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In distributed settings, collaborations between different entities, such as financial institutions, medical centers, and retail markets, are crucial to providing improved service and performance. However, the underlying entities may have little interest in sharing their private data, proprietary models, and objective functions. These privacy requirements have created new challenges for collaboration. In this work, we propose Gradient Assisted Learning (GAL), a new method for various entities to assist each other in supervised learning tasks without sharing data, models, and objective functions. In this framework, all participants collaboratively optimize the aggregate of local loss functions, and each participant autonomously builds its own model by iteratively fitting the gradients of the objective function. Experimental studies demonstrate that Gradient Assisted Learning can achieve performance close to centralized learning when all data, models, and objective functions are fully disclosed.

1 INTRODUCTION

One of the main challenges in harnessing the power of big data is the fusion of knowledge from numerous decentralized organizations that may have proprietary data, models, and objective functions. Due to various ethical and regulatory constraints, it may not be feasible for decentralized organizations to centralize their data and fully collaborate to learn a shared model. Thus, a large-scale autonomous decentralized learning method that can avoid data, models, and objective functions transparency may be of critical interest.

Cooperative learning may have various scientific and business applications (Roman et al., 2013). As illustrated in Figure 1, a medical institute may be helped by multiple clinical laboratories and pharmaceutical entities to improve clinical treatment and facilitate scientific research (Farrar et al., 2014; Lo, 2015). Financial organizations may collaborate with universities and insurance companies to predict loan default rates (Zhu et al., 2019). The organizations can match the correspondence with common identifiers such as user identification associated with the registration of different online platforms, timestamps associated with different clinics and health providers, and geo-locations associated with map-related traffic and agricultural data. With the help of our framework, they can form a community of shared interest to provide better Machine-Learning-as-a-Service (MLaaS) (Ribeiro et al., 2015; Wang et al., 2021) without transmitting their private data, proprietary models, and objective functions.

The main idea of Gradient Assisted Learning (GAL) is outlined below. In the training stage, the organization to be assisted, denoted by Alice, will calculate a set of ‘residuals’ and broadcast these to other organizations. These residuals approximate the fastest direction of reducing the training loss in hindsight. Subsequently, other organizations will fit the residuals using their local data, models, and objective functions and send the fitted values back to Alice. Alice will then assign weights to each organization to best approximate the fastest direction of learning. Next, Alice will line search for the optimal gradient assisted learning rate along the calculated direction of learning. The above procedure is repeated until Alice accomplishes a sufficient level of learning. In the inference stage, other organizations will send their locally predicted values to Alice, who will then assemble them to generate the final prediction. We show that the number of assistance rounds needed to approach the centralized performance is often small (e.g., fewer than ten). That is practically appealing, as GAL is primarily developed for large organizations with rich computation resources. A limited number of interactions with others will reduce the communications and networking costs. Our main contributions are summarized below.

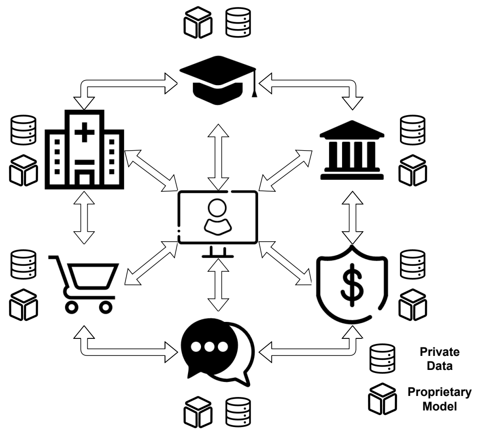


Figure 1: Decentralized organizations form a community of shared interest to provide better Machine-Learning-as-a-Service.

- We propose a Gradient Assisted Learning (GAL) algorithm that is suitable for large-scale autonomous decentralized learning and can effectively exploit task-relevant information preserved by vertically decentralized organizations. Our method enables simultaneous collaboration between multiple organizations without centralized sharing of a model, objective function, or data. Additionally, GAL does not need frequent synchronization of organizations. Moreover, GAL has low communication and networking costs. It typically requires fewer than ten rounds of assistance in our experiments.
- Interestingly, for the particular case of vertically distributed data, GAL generalizes the classical Gradient Boosting algorithms. We also provide asymptotic convergence analysis of the GAL algorithm.
- Our proposed framework can significantly outperform learning baselines and achieve near-oracle performance on various benchmark datasets while producing lower communications overhead compared with the state-of-the-art techniques.

2 Related work

Multimodal Data Fusion Vertically distributed data can be viewed as multimodal data with modalities provided in a distributed manner to different learners/organizations. Standard multimodal data fusion methods include the early, intermediate, and late data fusions (Khaleghi et al., 2013; Lahat et al., 2015). These methods concatenate different modes of data at the input, intermediate representation, and final prediction levels. However, these data fusion methods in decentralized settings often require organizations to share the task labels to train their local models synchronously. In contrast, our method presented below only requires that organizations asynchronously fit some task-related statistics named pseudo-residuals to approximate the direction of reducing the global training loss in hindsight.

Gradient Boosting We are inspired by Gradient Boosting (Mason et al., 1999; Friedman, 2001), where weak learners are sequentially trained from the same dataset and aggregated into a strong learner. In our learning context, each organization uses side-information from heterogeneous data sources to improve a particular learner's performance. Our method can be regarded as a generalization of Gradient Boosting to address decentralized learning with vertically distributed data.

Federated Learning Federated learning (Shokri & Shmatikov, 2015; Konecny et al., 2016; McMahan et al., 2017; Diao et al., 2021) is a popular distributed learning framework developed for edge devices. Its main idea is to learn a joint model by averaging locally learned model parameters. It avoids the need for the transmission of local training data. Conceptually, the goal of Federated Learning is to exploit the resources of edge devices with communication efficiency. Vertical Federated Learning methods split sub-networks for local clients to jointly optimize a global model (Vepakomma et al., 2018; Yang et al., 2019; Liu et al., 2019; Hamer et al., 2020; Thapa et al., 2020; Chen et al., 2020). These methods can be viewed as federated learning with an intermediate data fusion method, and the central server will have access to the true labels. In order to converge, these methods typically require frequent synchronization of backward gradients and a significant number of communication rounds (Chen et al., 2020). Such synchronization is critical as each client contributes to part of the globally shared model, and a client's local update may not decrease the overall loss. In contrast, our proposed method trains multiple local models with pseudo-residuals, each contributing to a small portion of the overarching loss. Each round of updates will decrease the loss, and it does not require frequent synchronizations. Consequently, our method can achieve desirable performance with significantly fewer communication rounds without a global transparent model.

Assisted Learning Assisted Learning (AL) (Xian et al., 2020) is a decentralized collaborative learning framework for organizations to improve their learning quality. In that context, neither the organization being assisted nor the assisting organizations share their private local models and data. Prior work on AL is limited to mean squared loss regression with only two organizations using a sequential exchange of information. Inspired by Gradient Boosting, our proposed Gradient Assisted Learning (GAL) is a general method for multiple organizations to assist each other in supervised learning scenarios. Our technical novelties include 1) generalization from squared loss to any differentiable loss for supervised learning, 2) allowing for private loss functions at each organization, 3) generalization from a sequential protocol between two organizations to parallel aggregation across multiple organizations, and 4) introduction of the assisted learning rate for fast convergence.

The outline of this paper is given next. In Section 3, we introduce and formulate GAL. In Section 4, we provide extensive experimental studies of GAL under various settings. We provide conclusions and final remarks in Section 5.

3 Gradient Assisted Learning

3.1 Notation

Suppose that there are N data observations independently drawn from a joint distribution $p_{xy} = p_x p_{y|x}$, where $y \in \mathcal{Y}$ and $x \in \mathbb{R}^d$ respectively represent the task label and feature variables, and d is the number of features. For regression tasks, we have $\mathcal{Y} = \mathbb{R}$. For K-class classification tasks, $\mathcal{Y} = \{e_1, \ldots, e_K\}$, where e_k is the canonical vector representing the class k, $k = 1, \ldots, K$. Let $\mathbb{E}$ and $\mathbb{E}_N$ denote the expectation and empirical expectation, respectively. Thus, $\mathbb{E}_N g(y, x) \triangleq N^{-1} \sum_{i=1}^{N} g(y_i, x_i)$ for any measurable function g, where (y_i, x_i) are i.i.d. observations from p_{xy}. Suppose that there are M organizations. Each organization m only holds X_m, a sub-vector of X (illustrated in Figure 2). In general, we assume that the variables in $X_1, \ldots, X_M$ are disjoint in the presentation of our algorithm, although our method also allows for the sharing of some variables. For example, one organization may observe demographic features for a mobile user cohort, and another organization holds health-related features of that cohort. Without loss of generality, we suppose that Alice, the organization to be assisted, has local data x_1 and task label y_1, while other $M - 1$ organizations are collaborators which assist Alice and have local data $x_2 \ldots x_M$.

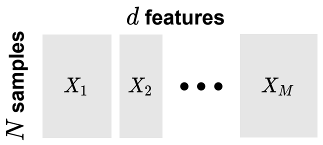


Figure 2: An illustration of the organizations' vertically distributed data.

3.2 Problem formulation

For $m = 1, \ldots, M$, let $\{x_{i,m}\}_{i=1}^{N}$ denote the available data to the organization m. Thus, N objects are simultaneously observed by M organizations, each observing a subset of features from the $x \in \mathbb{R}^d$. Alice also has private task labels $\{y_{i,1}\}_{i=1}^{N}$ for training purposes.

Let $\mathcal{F}_m$ and L_m respectively denote supervised function class (such as generalized linear functions or neural networks) and the private objective function of organization m. We will assume that L_m is differentiable. Without loss of generality, we assume that Alice denotes organization 1, who will be assisted. Without assistance from other organizations, Alice would learn a model that minimizes the following empirical risk, $F_{\text{Alone}} = \arg\min_{F_1 \in \mathcal{F}_1} \mathbb{E}_N L_1(y_1, F_1(x_1))$, Note that the above formulation only involves Alice's local data x_1 and local model (as represented by F_1 and L_1). Without privacy concerns, Alice would be able to operate on other organizations' data $x_2, \ldots, x_M$ as well. Recall that x represents the ensemble of all the available data variables. In general, the most ideal case for Alice is to minimize the following empirical risk, $F_{\text{Joint}} = \arg\min_{F \in \mathcal{F}} \mathbb{E}_N L_1(y_1, F(x))$, where $\mathcal{F}$ is a supervised function class defined on the space of x.

In reality, Alice has no access to the complete data and model resources of other organizations. In this light, she shall be happy to outsource the learning task to other organizations to cooperatively build a model in hindsight, without the need to share any organization's local data or models. In the prediction stage, Alice can collect the pieces of information needed to form a final prediction, to hopefully achieve a performance that significantly improves over her single-organization performance. To this end, we will develop a solution for Alice to achieve such a goal.

We will include detailed derivations and discussions of our solution in Subsection 3.3. For readability, we summarize notations that will be frequently used in the exposition below. Our method will require Alice to occasionally send a continuous-valued vector $r_1 = [r_{i,1}]_{i=1}^N \in \mathbb{R}^{N \times K}$ to each organization m at each communication round. These residuals, to be elaborated in the next subsection, approximate the fastest direction of reducing the training loss in hindsight, namely a sample version of $\partial L_1(y_1, F(x))/\partial F(x)$ given Alice's estimation of F at a particular time (round). Upon the input of these residual vectors, the organization m will locally learn a supervised function f_m that maps from its feature space to the residual space. With a slight abuse of notation, we also refer to f_m as the learned model. To this end, organization m will perform the empirical risk minimization

$$f_m = \arg\min_{f \in \mathcal{F}_m} \mathbb{E}_N \ell_m(r_1, f(x_m)) = \arg\min_{f \in \mathcal{F}_m} \frac{1}{N} \sum_{i=1}^{N} \ell_m \left(r_{i,1}, f(x_{i,m})\right) \tag{1}$$

to obtain a locally trained model f_m. Here, $\mathcal{F}_m$ and ℓ_m respectively denote the supervised function class and loss function of the organization m. We note that ℓ_m are private local regression loss functions for fitting the pseudo-residual r_1 and may not necessarily be the same as L_1 for fitting true labels. For example, L_1 may be the cross-entropy loss for classification of label y, while $\ell_{1:M}$ could be the squared loss for regression of the response r_1. The above local training (optimization) is often performed using the stochastic gradient descent (SGD) algorithm. In our assisted learning context, L_1, $\mathcal{F}_m$, and ℓ_m are proprietary local resources that cannot be shared across organizations.

3.3 The GAL algorithm

We first introduce the derivation of the GAL algorithm from a functional gradient descent perspective. Then, we cast the algorithm into pseudocode and discuss each step. Consider the unrealistic case that Alice has all the data x needed for a centralized supervised function $F : x \mapsto F(x)$. Recall that the goal of Alice is to minimize the population loss $\mathbb{E}_{p_{x,y}} L_1(y, F(x))$ over a data distribution $p_{x,y}$. If $p_{x,y}$ is known, starting with an initial guess $F^0(x)$, Alice would have performed a gradient descent step in the form of

$$F^1 \leftarrow F^0 - \eta \cdot \frac{\partial}{\partial F} \mathbb{E}_{p_{x,y}} L_1(y, F(x)) \mid_{F=F^0} = F^0 - \eta \cdot \mathbb{E}_{p_{x,y}} \frac{\partial}{\partial F} L_1(y, F(x)) \mid_{F=F^0}, \tag{2}$$

where the equality holds under the standard regularity conditions of exchanging integration and differentiation. Note that the second term in (2) is a function on $\mathbb{R}^d$. However, because Alice only has access to her own data x_1, the expectation $\mathbb{E}_{p_{x,y}}$ cannot be realistically evaluated. Therefore, we need to approximate it with functions in a pre-specified function set. In other words, we will find f from $\mathcal{F}_M$ that 'best' approximates $\mathbb{E}_{p_{x,y}} \frac{\partial}{\partial F} L_1(y, F(x))$. We will show that this is actionable without requiring the organizations to share proprietary data, models, and objective functions.

Recall that $\mathcal{F}_m$ is the function set locally used by the organization m, and x_m is correspondingly observed portion of x. The function class that we propose to approximate the second term in (2) is

$$\mathcal{F}_M = \left\{ f : x \mapsto \sum_{m=1}^{M} w_m f_m(x_m), \forall f_m \in \mathcal{F}_m, x \in \mathbb{R}^d, w \in P_M \right\}, \tag{3}$$

where $P_M = \{w \in \mathbb{R}^M : \sum_{m=1}^M w_m = 1, w_m \geq 0\}$ denotes the probability simplex. The gradient assistance weights w_m's are interpreted as the contributions of each organization at a particular greedy

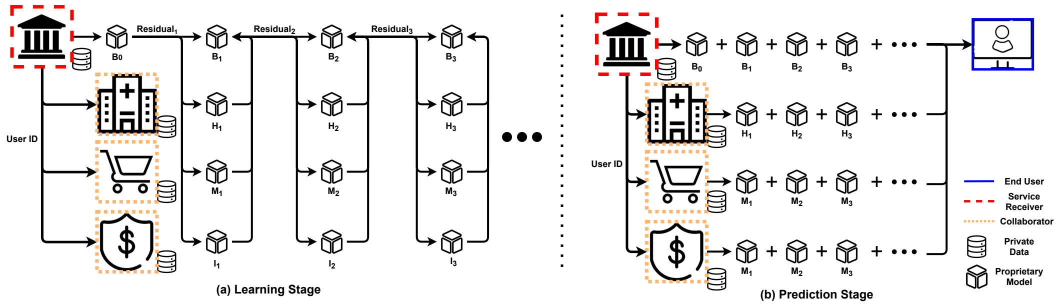


Figure 3: Learning and Prediction Stages for Gradient Assisted Learning.

update step. The gradient assistance weights are constrained to sum to one to ensure the function space is compact, and the solutions exist.

We propose the following solution so that *each organization can operate on its own local data, model, and objective function*. Alice initializes with a startup model, denoted by $F^0(x) = F^0(x_1, y_1)$, based only on her local data and labels. Alice broadcasts r_1 (named 'pseudo residuals') to each organization m, $m = 2, \cdots, M$, who will then fit a local model f_m using r_1. Each organization will then send the fitted values from f_m to Alice, who will train suitable gradient assistance weights w_m. Subsequently, Alice finds the η in (2) that minimizes her current empirical risk. The above procedure is iterated for a finite number of rounds until Alice obtains a satisfactory performance (e.g., on validation data). The validation will be based on the same technique as the prediction stage to be described below. This training stage is described under the 'learning stage' of Algorithm 1. Note that the pseudocode is from the perspective of Alice, the service receiver. For each organization m, it will only need to perform the empirical risk minimization using the label r_1^t sent by Alice at each round t.

In the Prediction/Inference stage (given above in Algorithm 1), other organizations send prediction results generated from their local models to Alice, who will calculate a prediction result $F^T(x)$ that is implicitly operated on x, where T is the number of iteration steps.

We note that the idea of approximating functional derivatives with regularized functions was historically used to develop the seminal work of gradient boosting (Mason et al., 1999; Friedman, 2001). Interestingly, when there is only one organization, the above method reduces to the standard gradient boosting algorithm (Mason et al., 1999; Friedman, 2001).

Organizations in our learning framework form a shared community of interest. Each service-providing organization can provide end-to-end assistance for an organization without sharing anyone's proprietary data, models, and objective functions. In practice, the participating organizations may receive financial rewards from the one to assist. Moreover, every organization in this framework can provide its own task and seek help from others. As a result, all organizations become mutually beneficial to each other. We provide a realistic example in Figure 3 to demonstrate each step of Algorithm 1. We elaborate on the learning and prediction procedures in the Appendix.

We also provide an asymptotic convergence analysis for a simplified and abstract version of the GAL algorithm, where the goal is to minimize a loss $f \mapsto \mathcal{L}(f)$ over a function class through step-wise function aggregations. Because of the greedy nature of GAL, we consider the function class to be the linear span of organization-specific $\mathcal{F}_m$. The following result states that the GAL can produce a solution that attains the infimum of $\mathcal{L}(f)$.

Theorem (Informal): Under suitable assumptions of the loss function and learning rates, the GAL produces an F^t that satisfies $\lim_{t\to\infty} \mathcal{L}(F^t) = \inf_{f\in \text{span}(\mathcal{F}_1,\ldots,\mathcal{F}_M)} \mathcal{L}(f)$.

4 Experimental Studies

Datasets We experiment with several different types of data introduced below. 1) UCI datasets downloadable from the *scikit-learn* package (Pedregosa et al., 2011), including Diabetes (Efron et al., 2004), Boston Housing (Harrison Jr & Rubinfeld, 1978), Blob (Pedregosa et al., 2011), Iris (Fisher, 1936), Wine (Aeberhard et al., 1994), Breast Cancer (Street et al., 1993), and QSAR (Mansouri et al., 2013) datasets, where we randomly partition the features into 2, 4, or 8 subsets, each for an organization. 2) MNIST (LeCun et al., 1998) and CIFAR10 (Krizhevsky et al., 2009) image

Algorithm 1 GAL: Gradient Assisted Learning (from the perspective of the service receiver, Alice)

```
Input: $M$ decentralized organizations, each holding data $\{x_{i,m}\}_{i=1}^N$ (local) corresponding to $N$
    objects, the task label $\{y_{i,1}\}_{i=1}^N$ initially held by the service receiver (Alice local) , model
    class $\mathcal{F}_m$ (local), gradient assistance weights $w$ (Alice local), assistance rate $\eta$ (Alice local),
    overarching loss function $L_1$ (Alice local), regression loss function $\ell_m$ to fit pseudo-residual
    (local), number of assistance rounds $T$.
Learning Stage:
    Intialization:
        Let $t = 0$, and initialize $F^0(x) = \mathbb{E}_N(y_1)$
    for assistance round $t$ from 1 to $T$ do
        Compute pseudo-residual
        $r_1^t = -\left[\frac{\partial L_1(y_1, F^{t-1}(x))}{\partial F^{t-1}(x)}\right]$
        Broadcast pseudo-residual $r_1^t$ to other organizations
        for organization $m$ from 1 to $M$ in parallel do
            $f_m^t = \arg\min_{f_m \in \mathcal{F}_m} \mathbb{E}_N \ell_m (r_1^t, f_m(x_m))$
        end
        Gather predictions $f_m^t(x_m)$, $m = 1, \dots M$, from all the organizations
        Optimize the gradient assistance weights
        $\hat{w}^t = \arg\min_{w \in P_M} \mathbb{E}_N \ell_1 \left(r_1^t, \sum_{m=1}^M w_m f_m^t(x_m)\right)$
        Line search for the gradient assisted learning rate
        $\hat{\eta}^t = \arg\min_{\eta \in \mathbb{R}} \mathbb{E}_N L_1 \left(y_1, F^{t-1}(x) + \eta \sum_{m=1}^M \hat{w}_m^t f_m^t(x_m)\right)$
        $F^t(x) = F^{t-1}(x) + \hat{\eta}^t \sum_{m=1}^M \hat{w}_m^t f_m^t(x_m)$
    end
Prediction Stage:
    For each data observation $x^*$, of which $x_m^*$ is held by organization $m$:
    Gather predictions $f_m^t(x_m^*)$, $t = 1, \dots, T$ from each organization $m$, $m = 1, \dots, M$
    Predict with $F^T(x^*) \triangleq F^0(x_1^*) + \sum_{t=1}^T \hat{\eta}^t \sum_{m=1}^M \hat{w}_m^t f_m^t(x_m^*)$
```

datasets, where we split each image into image patches as depicted in Figure 5. We do not adopt data augmentation such as horizontal flipping. 3) MIMIC3 (Johnson et al., 2016) dataset, where the task aims to predict the length-of-stay with in-hospital data. We split the time series features of MIMIC3 for 4 organizations. 4) ModelNet40 (Wu et al., 2015) dataset, which contains 2D camera views of 3D object data for 12 organizations (following (Su et al., 2015)). For all the datasets, we train on 80% of the available data and test on the remaining for UCI datasets. The summary statistics of each dataset are elaborated in Table 4 of the Appendix. We conduct four random experiments for all datasets with different seeds, and the standard deviation is shown in the brackets.

Model settings We use Linear models for the UCI datasets, convolution neural networks (CNN) for the MNIST, CIFAR10, and ModelNet40 datasets, and long short-term memory (LSTM) for the MIMIC3 dataset. Although we have used the same model architecture for every organization in the experiments, our algorithm does not require the organizations to have the same local learning algorithms and models. Details of the architectures are given in the Appendix.

Learning For regression tasks on datasets such as the Diabetes, Boston Housing, and MIMIC3, we train with l_1 for the residual loss ℓ_m and the overarching loss L_1, and evaluate using the mean absolute deviation (MAD). For classification tasks on the remaining datasets, we train with ℓ_2-loss for the residual loss ℓ_m and cross-entropy loss for the overarching loss L_1, and evaluate using accuracy. We use the SGD optimizer for Linear and CNN with a learning rate of 10^{-1} and the Adam optimizer for LSTM with a learning rate of 10^{-4}. The number of local epochs E is 100 for UCI datasets and 10 for the rest. The number of assistance rounds T is 10 in our experiments.

To optimize gradient assistance weights, we use the Adam optimizer with a learning rate of 10^{-1} and enforce the parameters to sum to 1 by using the softmax function. Furthermore, we perform a line search for the gradient assisted learning rate with the Limited-Memory BFGS optimizer using a learning rate of 1. In the experiments, we found that using the quasi-newton method greatly improves the convergence rates due to better estimation of gradient assisted learning rates compared with SGD and Adam. The cost to optimize the gradient assistance weights w and gradient assisted learning rate η is often negligible compared with the cost to fit the pseudo-residuals since the number of parameters involved in $w \in \mathbb{R}^M$ and $\eta \in \mathbb{R}^1$ is small. Details of learning hyper-parameters are included in Table 6 in the Appendix.

4.1 Baselines

Our experiments are performed with four baselines, including 'Interm', 'Late', 'Joint', and 'Alone'. 'Interm' and 'Late' refer to intermediate and late data fusions (Khaleghi et al., 2013; Lahat et al., 2015), respectively. 'Interm' works for deep learning models such as CNN and LSTM by averaging the hidden representation of each local model. 'Late' also works for Linear models as it averages the output of each local model. 'Joint' is the oracle case where all the data are held by Alice and trained with the Gradient Boosting reduced from GAL. 'Alone' is the single-agent scenario, where only Alice's data are used for learning and prediction.

The experimental results are shown in Tables 1 and 2. We also visualize the performance of CIFAR10 and MIMIC3 at each assistance round in Figure 4(a,d). Our method significantly outperforms the bottom line 'Alone' in all the settings. This is expected since the first organization holds partial data and does not receive any assistance under 'Alone'. Interestingly, the performance of MNIST for $M = 8$ drops significantly under 'Alone' because the organization only holds the left upper image patch, which is usually completely dark (shown in Figure 5 of the Appendix). The results demonstrate that with GAL, an organization with little informative data can leverage other organizations' private data and models and even achieve near-oracle performance (the 'Joint' case). Moreover, we found that the number of assistance rounds needed to approach the centralized performance is small (e.g., often within ten).We point out that although 'Interm', 'Late', and 'Joint' marginally outperform our method, they require training of centralized data. Our GAL algorithm replaces the true label used in 'Interm', 'Late', and 'Joint' oracle case with pseudo-residual to enhance privacy (in terms of data, model, and objective). The results from both regression and classification datasets lead to similar conclusions. More results for different numbers of organizations can be found in the Appendix.

Table 1: Results on the UCI datasets ($M = 8$). The Diabetes and Boston Housing (regression) are evaluated with MAD, and the rest (classification) are evaluated with Accuracy (in percentage).

Dataset	Diabetes	BostonHousing	Blob	Wine	BreastCancer	QSAR
Late	136.2(0.1)	8.0(0)	100(0)	100(0)	96.9(0.4)	76.9(0.8)
Joint	43.4(0.3)	3.0(0)	100(0)	100(0)	98.9(0.4)	84(0.2)
Alone	59.7(9.2)	5.8(0.9)	41.3(10.8)	63.9(15.6)	92.5(3.4)	68.8(3.4)
AL	51.5(4.6)	4.7(0.6)	97.5(2.5)	95.1(3.6)	**98.7(1.3)**	70.6(5.2)
GAL	**42.7(0.6)**	**3.2(0.2)**	**100(0)**	**96.5(3)**	98.5(0.7)	**82.5(0.8)**

Table 2: Results on the MNIST ($M = 8$), CIFAR10 ($M = 8$), MIMIC3, and ModelNet40 datasets. The MIMIC3 (regression) is evaluated with MAD, and the rest (classification) are evaluated with Accuracy. Note that the 'Joint' of ModelNet40 does not perform well because of the known fact that a joint model cannot take into account multiple orientations (Su et al., 2015).

Dataset	MNIST	CIFAR10	MIMIC3	ModelNet40
Interm	98.8(0.1)	78.2(0.2)	92.4(0.3)	85.8(0.2)
Late	98(0.1)	74.4(0.3)	94.3(0.1)	86.6(0.2)
Joint	99.4(0)	80.1(0.2)	96.2(2.2)	46.3(1.4)
Alone	24.2(0.1)	46.3(0.3)	103.2(0.8)	76.4(1.1)
AL	34.3(0.1)	51.1(0.2)	104.1(0.1)	77.3(2.8)
GAL	**96.3(0.6)**	**74.3(0.2)**	**96.4(1.8)**	**83.0(0.2)**

4.2 Comparison with the state-of-the-art

We perform experiments with the MIMIC3 and ModelNet40 data to compare our method with the related works, including Assisted Learning (AL) (Xian et al., 2020) and vertical asynchronous Federated Learning (VAFL) (Chen et al., 2020). The results and comparison of computation and communication cost are shown in Table 3. We also provide benchmark results where all the data are held by Alice (Harutyunyan et al., 2019; Su et al., 2015). We provide some discussions below.

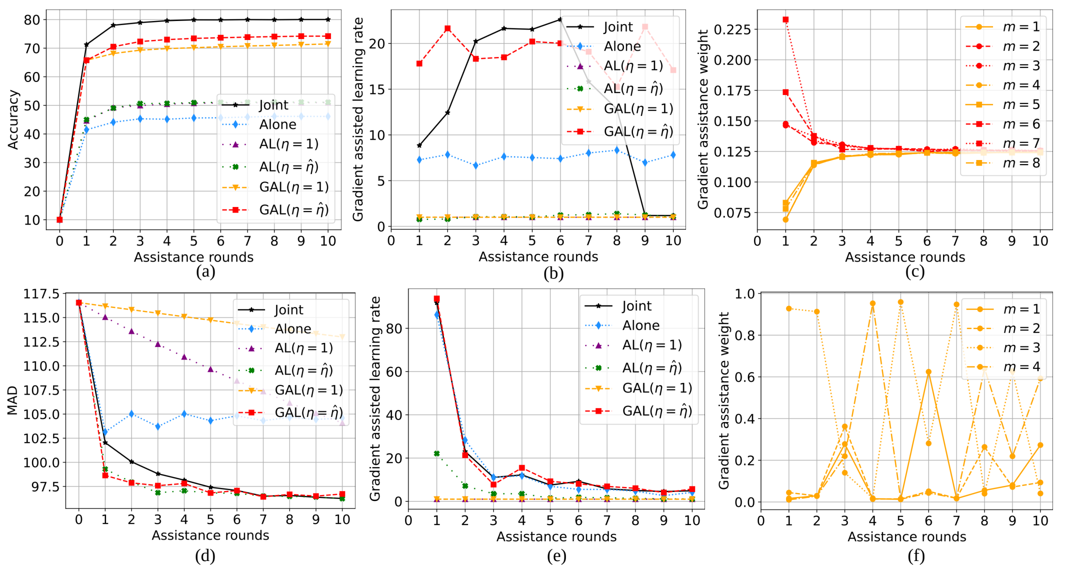


Figure 4: Results from the CIFAR10 (a-c) ($M = 8$) and MIMIC3 (d-f) ($M = 4$) datasets. GAL significantly outperforms 'Alone' and performs close to the centralized case 'Joint.' The gradient assisted learning rate diminishes to zero as the overarching loss converges. A constant gradient assisted learning rate ($\eta = 1$) converges much slower. The gradient assistance weights of the central image patches ($m = \{2, 3, 6, 7\}$) are larger than the boundary ones in the first few rounds.

GAL vs. AL Compared with AL, our method GAL 1) generalizes from the squared loss to any differentiable loss function. 2) introduces the gradient assisted learning rate to reduce the number of communication rounds to achieve satisfactory performance, as discussed in Section 4.3. 3) optimizes multiple organizations in parallel with the help of gradient assistance weights, while AL is restricted to sequential training of data from each organization. We also note that the original AL only applies to regression tasks with $\eta = 1$. We extend it to tackle classification problem with pseudo-residuals for comparison as demonstrated in Tables 1 and 2. Compared with AL under the constraint of the same communication cost, our method requires M^{-1} communication rounds. It is due to that AL trains local models one after another in a sequential manner. In particular, one assistance round of AL requires M communication rounds to traverse M organizations once. However, one assistance round of our method only requires one communication round because we can train every organization and aggregate their outputs in parallel. Moreover, our method outperforms AL in all of our experiments in terms of prediction. Thus, compared with AL, GAL can significantly generalize the problem scope, reduce the computation time and number of communication rounds, significantly outperforms AL, and performs close to the oracle case.

Table 3: Comparison between GAL and state-of-the-art methods. In the table, M represents the number of organizations. The performance metrics are MAD for MIMIC3 and Accuracy for ModelNet40. AL requires M times more computation time and communication rounds because it sequentially trains each organization. VAFL requires far more communication rounds because it requires batch-wise updates from the server. The communication cost is $O(d \cdot E_i)$ for VAFL and $O(K \cdot T)$ for GAL, where d, E_m, K, T represent the size of feature embeddings, the total number of training epochs of organization m, the size of target labels, and the number of assistance rounds, respectively. Here, d is typically greater than K, and E_m is greater than T because GAL allows more local training.

Dataset	Method	M	T	Training Epochs E_m	Computation Time	Computation Space	Communication Round	Communication Cost	Result
MIMIC3	Benchmark (Harutyunyan et al., 2019)	1	N/A	100	1 ×	1 ×	0	0 ×	94.7
	AL (Xian et al., 2020)	4	10	100	4 ×	10 ×	40	1 ×	104.1
	GAL	4	10	100	1 ×	10 ×	10	1 ×	96.4
ModelNet40	MVCNN (Su et al., 2015)	1	N/A	100	1 ×	1 ×	0	0 ×	88.1
	VAFL (Chen et al., 2020)	4	N/A	100	1 ×	1 ×	10000	128 ×	81.0
	AL (Xian et al., 2020)	12	10	100	12 ×	10 ×	120	1 ×	77.3
	GAL	12	10	100	1 ×	10 ×	10	1 ×	83.0

GAL vs. VAFL Vertical federated learning method such as VAFL (Chen et al., 2020) and SplitFed (Thapa et al., 2020) can be viewed as federated learning with intermediate data fusion. They typically require frequent synchronized communications of hidden representations and gradients to optimize a global model in hindsight. In particular, VAFL allocates separate convolution layers for each organization and transmits hidden representations to the server. Compared with VAFL, our method GAL outperforms VAFL in the following aspects. VAFL computes and transmits the (gradients of) hidden representations for every batch-wise update between server and clients. The batch size of VAFL is 0.01 of the whole dataset. Consequently, 100 training epochs would result in 10000 communication rounds for each organization. On the contrary, GAL trains one local model for multiple epochs at each communication round. Each model solves a small part of the overarching loss function (namely, the pseudo-residual multiplied by the gradient assisted learning rate). 1) GAL allows each participant to use its own local model architecture autonomously, while VAFL requires all participants to fit with deep learning model architecture. 2) Under the constraint of the same number of local training epochs, GAL requires a much fewer number of communication rounds than VAFL to achieve satisfactory performance. 3) GAL avoids sharing true labels and objective functions, while the VAFL server will have access to both. GAL is also robust against noise injection-based privacy mechanisms, with the help of the gradient assistance weights, as discussed in Section 4.3.

4.3 Gradient assisted learning rate

We show the gradient assisted learning rate of CIFAR10 and MIMIC3 datasets at each assistance round in Figure 4(b,e). More results are included in the Appendix. Recall that we adopt a quasi-newton method to line search for the gradient assistance rate. Standard first-order optimization methods usually fail to produce accurate line search results and require more assistance rounds to converge. We perform an ablation study of using a constant gradient assisted learning rate ($\eta = 1$). As shown in Figure 4(a,d), the constant gradient assisted learning rate leads to a convergence much slower than line search method. Fast convergence is desirable since the computation and communication cost increases with the number of assistance rounds. To determine the maximum number of assistance rounds T for the service receiver, we can run the GAL procedure until the gradient assisted learning rate becomes small. When the gradient assistance rate is small as shown in Figure 4(e), the overarching loss converges to zero. In this light, an organization may stop receiving assisted learning when the gradient assisted learning rate is below a threshold.

4.4 Gradient assistance weights

We show the gradient assistance weights of CIFAR10 and MIMIC3 datasets at each assistance round in Figure 4(c,f). More results can be found in the Appendix. The results of MIMIC3 show that the contributing organization with the largest assistance weight may change from one round to another. For image datasets MNIST and CIFAR10, it is interesting that the image patches with dominant contributions are $m = (2, 3, 6, 7)$ (colored in red). These image patches correspond to the center of the original image, which matches our intuition appealingly.

We also perform an ablation study of the gradient assistance weights by adding noises (Gaussian with zero mean and σ^2 variance, $\sigma \in \{1, 5\}$) to the transmitted pseudo-residuals to a (randomly chosen) half of the clients during learning and prediction. The purpose of adding noises is to mimic realistic scenarios where some assisting organizations are uninformative, add a moderate amount of noise to enhance data privacy (Dong et al., 2019), or inject adversarial pseudo-residuals. We summarize the results of this ablation study in Table 13 and Figure 6. The results show that the GAL equipped with gradient assistance weights is more robust than the GAL with direct average. We note that as the number of participating organizations becomes large, the gradient assistance weights can also be used for organization (meta-feature) selection.

5 Conclusion

In this paper, we proposed Gradient Assisted Learning, a decentralized learning method for multiple organizations to collaborate without sharing proprietary data, models, or labels. The proposed solution can significantly outperform the learning baselines, state-of-the art methods, and achieve near-oracle performance (as if data were centralized) on various datasets. Our approach enables organizations to form a shared community of interest without compromising their private data and models while achieving near-oracle full collaboration performance. Moreover, this is achieved without any constraints on the models selected by the collaborating organizations.

REFERENCES

Stefan Aeberhard, Danny Coomans, and Olivier De Vel. Comparative analysis of statistical pattern recognition methods in high dimensional settings. *Pattern Recognition*, 27(8):1065–1077, 1994.

Tianyi Chen, Xiao Jin, Yuejiao Sun, and Wotao Yin. Vafl: a method of vertical asynchronous federated learning. *arXiv preprint arXiv:2007.06081*, 2020.

Enmao Diao, Jie Ding, and Vahid Tarokh. HeteroFL: Computation and communication efficient federated learning for heterogeneous clients. In *International Conference on Learning Representations*, 2021.

Jinshuo Dong, Aaron Roth, and Weijie J Su. Gaussian differential privacy. *arXiv preprint arXiv:1905.02383*, 2019.

Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *Ann. Stat.*, 32(2):407–499, 2004.

John T Farrar, Andrea B Troxel, Kevin Haynes, Ian Gilron, Robert D Kerns, Nathaniel P Katz, Bob A Rappaport, Michael C Rowbotham, Ann M Tierney, Dennis C Turk, et al. Effect of variability in the 7-day baseline pain diary on the assay sensitivity of neuropathic pain randomized clinical trials: an acttion study. *Pain®*, 155(8):1622–1631, 2014.

Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7 (2):179–188, 1936.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Ann. Stat.*, pp. 1189–1232, 2001.

Jenny Hamer, Mehryar Mohri, and Ananda Theertha Suresh. Fedboost: A communication-efficient algorithm for federated learning. In *Proc. ICML*, pp. 3973–3983, 2020.

David Harrison Jr and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air. *J. Environ. Econ. Manag.*, 5(1):81–102, 1978.

Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *Sci. Data*, 6(1):1–18, 2019.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Sci. Data*, 3:160035, 2016.

Bahador Khaleghi, Alaa Khamis, Fakhreddine O Karray, and Saiedeh N Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information fusion*, 14(1):28–44, 2013.

Jakub Konecny, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Dana Lahat, Tülay Adali, and Christian Jutten. Multimodal data fusion: an overview of methods, challenges, and prospects. *Proceedings of the IEEE*, 103(9):1449–1477, 2015.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yang Liu, Yan Kang, Xinwei Zhang, Liping Li, Yong Cheng, Tianjian Chen, Mingyi Hong, and Qiang Yang. A communication efficient vertical federated learning framework. *arXiv preprint arXiv:1912.11187*, 2019.

Bernard Lo. Sharing clinical trial data: maximizing benefits, minimizing risk. *JAMA*, 313(8):793–794, 2015.

Kamel Mansouri, Tine Ringsted, Davide Ballabio, Roberto Todeschini, and Viviana Consonni. Quantitative structure–activity relationship models for ready biodegradability of chemicals. *J. Chem. Inf. Model.*, 53(4):867–878, 2013.

Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent in function space. 1999.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proc. AISTATS*, pp. 1273–1282, 2017.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz. Mlaas: Machine learning as a service. In *Proc. ICMLA*, pp. 896–902. IEEE, 2015.

Rodrigo Roman, Jianying Zhou, and Javier Lopez. On the features and challenges of security and privacy in distributed internet of things. *Comput. Netw.*, 57(10):2266–2279, 2013.

Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proc. CCS*, pp. 1310–1321, 2015.

W Nick Street, William H Wolberg, and Olvi L Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Biomedical image processing and biomedical visualization*, volume 1905, pp. 861–870. International Society for Optics and Photonics, 1993.

Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pp. 945–953, 2015.

Chandra Thapa, Mahawaga Arachchige Pathum Chamikara, and Seyit Camtepe. Splitfed: When federated learning meets split learning. *arXiv preprint arXiv:2004.12088*, 2020.

Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

Xinran Wang, Yu Xiang, Jun Gao, and Jie Ding. Information laundering for model privacy. *Proc. ICLR*, 2021.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

Xun Xian, Xinran Wang, Jie Ding, and Reza Ghanadan. Assisted learning: A framework for multi-organization learning. *Advances in Neural Information Processing Systems*, 33, 2020.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *in Proc. TIST*, 10(2):12, 2019.

Tong Zhang and Bin Yu. Boosting with early stopping: Convergence and consistency. *The Annals of Statistics*, 33(4):1538–1579, 2005.

Lin Zhu, Dafeng Qiu, Daji Ergu, Cai Ying, and Kuiyi Liu. A study on predicting loan default based on the random forest algorithm. *Procedia Computer Science*, 162:503–513, 2019.

Appendix for "Gradient Assisted Learning"

We describe the application scenario, further experimental results, and theoretical analysis in the Appendix.

A Application scenario

As shown in Figure 1, Alice, the service receiver (bank) squared in red dashed line, is the organization to be assisted. Before learning, it broadcasts identification (ID) to locate and align vertically distributed data held by other organizations. At the beginning of the Learning Stage, the bank deterministically initializes the values of $F^0(x)$ to be the unbiased estimate of y_1, namely $F^0(x) = \mathbb{E}_N(y_1^0)$. For the regression task, $F^0(x)$ is a single scalar. For classification task, $F^0(x)$ is a point in the K-dimensional simplex P_K.

During the first assistance round in the Learning Stage, the bank computes pseudo-residual r_1^1 and broadcasts it to other organizations (e.g., hospital, mall, and insurance company). Then, all the organizations, including the bank, will fit a new local model with 1) their local data, 2) the pseudo-residual r_1^1, and 3) their local regression loss function ℓ_m (e.g., ℓ_2-loss) to fit the pseudo-residual. We note that organizations have complete autonomy on model fitting. In particular, they can choose their own learning algorithms and models by considering their resources (e.g., computation power). Next, the bank will aggregate all the predictions from each organization's local models by optimizing a weight vector $w_{1:M}$ referred to as gradient assistance weights. As previously discussed in Equation (3), we approximate the oracle gradient (operated on centralized data, in hindsight) with a weighted average of those predictions from organizations. We then numerically search for the learning rate η. This process can be iterated multiple times until the learning rate is low or the validation loss is satisfactory.

During the Prediction Stage, organizations will predict with trained models at every assistance round and transmit their predictions to the bank. Similar to the Learning Stage, *the synchronization of each organization is unnecessary*. The bank computes the final prediction with gradient assistance weights, learning rates, and received predictions.

B Further experimental results

In Table 4, we illustrate the statistics of datasets used in our experiments. In Figure 5, we show how MNIST and CIFAR10 images are split into 2, 4, and 8 image patches. The left upper image patch (labeled $[1]$) of MNIST image is less informative which demonstrates that an organization with little informative data can leverage other organizations' private data and models. The central image patches (labeled $[2, 3, 6, 7]$) of MNIST and CIFAR10 images are more informative than others, which leads to larger corresponding gradient assistance weights. Table 5 summarizes the deep neural network architecture used for the MNIST, CIFAR10, and ModelNet40 datasets. Table 6 shows the hyperparameters used in our experiments. In Tables 7 and 8, we demonstrate the results of our experiments for $M = 2$. In Tables 9 and 10, we demonstrate the results of our experiments for $M = 4$. In Tables 11 and 12, we include ablation studies of the noise injection to half of the organizations for $M = 2$ and $M = 4$, respectively.

From Figures 7 to 17, we show the results from the Diabetes, Boston Housing, Blob, Iris, Breast Cancer, Wine, QSAR, MNIST, CIFAR10, MIMIC3, and ModelNet40 datasets, respectively. All the results indicate that the proposed GAL algorithm allows all the participants to collaboratively and efficiently optimize the objective by the iterative fitting of pseudo-residuals.

Table 4: Detailed statistics used in each data experiment. The variables d and K respectively denote the number of features (or the shape of the image) and the length of the prediction vector (or equivalently, the number of classes in the classification task).

Dataset	N_{train}	N_{test}	d	K	M
Diabetes	353	89	10	1	{2, 4, 8}
BostonHousing	404	102	13	1	{2, 4, 8}
Blob	80	20	10	10	{2, 4, 8}
Iris	120	30	4	3	{2, 4}
Wine	142	36	13	3	{2, 4, 8}
BreastCancer	455	114	30	2	{2, 4, 8}
QSAR	844	211	41	2	{2, 4, 8}
MNIST	60000	10000	(1,28,28)	10	{2, 4, 8}
CIFAR10	50000	10000	(3,32,32)	10	{2, 4, 8}
ModelNet40	3163	800	(12,3,32,32,32)	40	{12}
MIMIC3	16000	8000	76	1	{4}

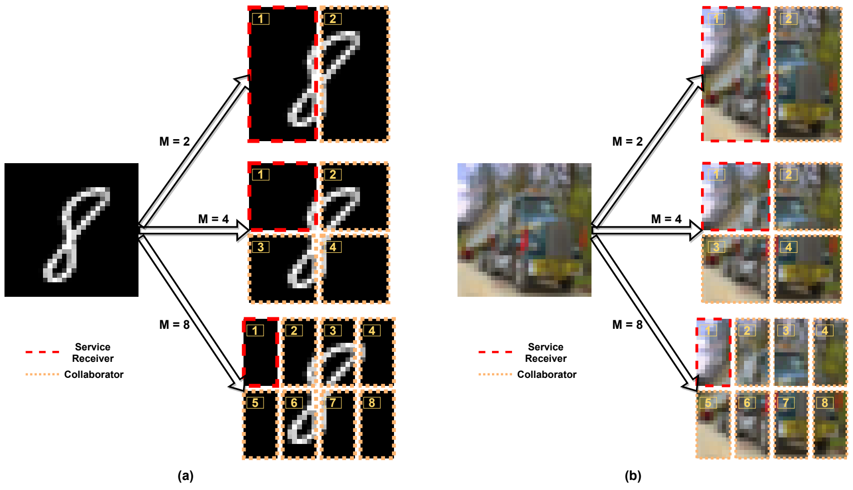


Figure 5: An illustration of (a) MNIST and (b) CIFAR10 data split into 2, 4, and 8 image patches. The left upper image patch (labeled [1]) of MNIST images is less informative in general. In contrast, the central image patches (labeled [2, 3, 6, 7]) of MNIST and CIFAR10 images are more informative.

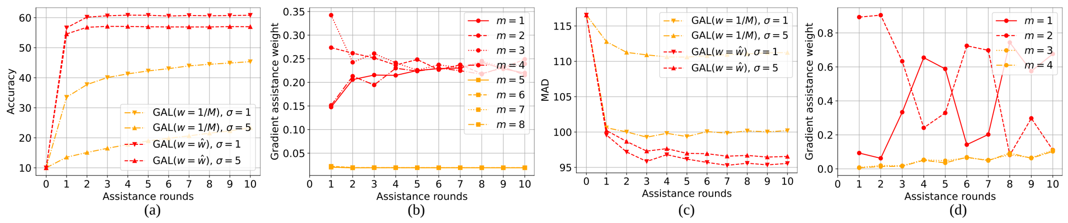


Figure 6: Ablation study results on CIFAR10 (a-b) ($M = 8$) and MIMIC3 (c-d) ($M = 4$) datasets. Plots (a,c) show that the GAL equipped with gradient assistance weight significantly outperforms the GAL with direct average under noise injections ($\mathcal{N}(0, \sigma^2)$, $\sigma = \{1, 5\}$) to the transmitted pseudo-residual to half of the organizations during learning and prediction. Plots (b,d) show the gradient assistance weight of noisy (in orange, $\sigma = 1$) and noise-free organizations (in red).

Table 5: The model architecture of Convolutional Neural Networks (CNN) used in our experiments of the MNIST, CIFAR10, and ModelNet40 datasets. The n_c, H, W represent the shape of images, namely the number of image channels, height, and width, respectively. K is the number of classes in the classification task. The BatchNorm and ReLU layers follow Conv(output channel size, kernel size, stride, padding) layers. The MaxPool(output channel size, kernel size) layer reduces the height and width by half.

Image $x \in \mathbb{R}^{n_c \times H \times W}$
Conv(64, 3, 1, 1)
MaxPool(64, 2)
Conv(128, 3, 1, 1)
MaxPool(128, 2)
Conv(256, 3, 1, 1)
MaxPool(256, 2)
Conv(512, 3, 1, 1)
MaxPool(512, 2)
Global Average Pooling
Linear(512, K)

Table 6: Hyperparameters used in our experiments for training local models, gradient assisted learning rates, and gradient assistance weights.

Model		UCI	MNIST	CIFAR10	ModelNet40	MIMIC3
Architecture		Linear		CNN		LSTM
Local	Epoch	100			10	
	Batch size	1024		512	512	64
	Optimizer			SGD		Adam
	Learning rate			1.0E-01		1.0E-04
	Weight decay			5.0E-04		
Gradient assited learning rates	Epoch			10		
	Batch size			Full		
	Optimizer			L-BFGS		
	Learning rate			1		
Gradient assistance weights	Epoch			100		
	Batch size			1024		
	Optimizer			Adam		
	Learning rate			1.0E-01		
	Weight decay			5.0E-04		
Assistance rounds				10		

Table 7: Results from the UCI datasets ($M = 2$). Diabetes and Boston Housing (regression) are evaluated with MAD and the rest (classification) are evaluated with Accuracy.

Dataset	Diabetes	BostonHousing	Blob	Wine	BreastCancer	QSAR
Late	120.2(0.1)	3.6(0.1)	100(0)	100(0)	100(0)	99.3(0.4)
Joint	43.4(0.3)	3.0(0)	100(0)	99.2(1.4)	100(0)	98.9(0.4)
Alone	46.8(3.5)	4.1(0.7)	100(0)	92.5(6)	93.1(6.4)	99.1(0.6)
AL	63.7(1.5)	3.9(0.6)	98.8(2.2)	95(2.9)	95.1(2.3)	97.6(0.7)
GAL	**43.2(0.8)**	**2.9(0.1)**	**100(0)**	**99.2(1.4)**	**96.5(2.3)**	**98.9(0.4)**

Table 8: Results from the MNIST ($M = 2$) and CIFAR10 ($M = 2$) datasets.

Dataset	MNIST	CIFAR10
Interm	99.4(0)	81.1(0.3)
Late	99.0(0)	81.0(0.2)
Joint	99.4(0)	80.1(0.2)
Alone	96.7(0.2)	72.7(0.2)
AL	96.4(0.1)	74.7(0.3)
GAL	**98.5(0.2)**	**78.7(0.4)**

Table 9: Results from the UCI datasets ($M = 4$). Diabetes and Boston Housing (regression) are evaluated with MAD and the rest (classification) are evaluated with Accuracy.

Dataset	Diabetes	BostonHousing	Blob	Wine	BreastCancer	QSAR
Late	129.5(0.1)	4.7(0)	100(0)	100(0)	100(0)	98.5(0.7)
Joint	43.4(0.3)	3.0(0)	100(0)	99.2(1.4)	100(0)	98.9(0.4)
Alone	56.6(8.2)	4.8(0.6)	80.0(6.1)	79.2(13)	84.7(1.4)	97.1(1)
AL	58.3(2.4)	5.2(0.3)	100(0)	88.3(8.3)	92.4(2.3)	98.9(1.1)
GAL	**43.3(1.1)**	**3.0(0.1)**	**100(0)**	**100(0)**	**97.9(2.3)**	**99.1(0.6)**

Table 10: Results from the MNIST ($M = 4$) and CIFAR10 ($M = 4$) datasets.

Dataset	MNIST	CIFAR10
Interm	99.1(0)	79.8(0.1)
Late	98.4(0.1)	77.5(0.2)
Joint	99.4(0)	80.1(0.2)
Alone	81.2(0.1)	60.0(0.4)
AL	82.5(0.1)	64.8(0.3)
GAL	**96.6(0.2)**	**77.3(0.2)**

Table 11: Ablation study ($M = 2$) of gradient assistance weights by adding noises to the transmitted pseudo-residuals to half of the organizations. The evaluation metrics are the same as Tables 1 and 2.

Dataset	Weight	Diabetes	Boston Housing	Blob	Iris	Wine	Breast Cancer	QSAR	MNIST	CIFAR10
$\sigma = 1$	✗	50.1(1.9)	4.4(0.2)	62.5(2.5)	80.8(6.4)	86.8(2.3)	89.9(3.1)	73.2(1.3)	79.7(0.3)	48.8(0.3)
	✓	**47.8(2.4)**	**3.5(0.5)**	**97.5(4.3)**	**95(3.7)**	**96.5(3.0)**	**98.7(1.0)**	**80.2(0.5)**	**96.8(0.1)**	**71.4(0.1)**
$\sigma = 5$	✗	58.8(1.3)	6.1(0.2)	25.0(9.4)	52.5(10.9)	63.9(3.4)	73.2(1.0)	63.3(0.5)	34.8(0.5)	22.0(0.2)
	✓	**46.5(3.1)**	**4.1(0.8)**	**83.8(7.4)**	**90(4.1)**	**93.1(4.2)**	**97.6(1.1)**	**78.3(1)**	**96.3(0.1)**	**65.9(0.3)**

Table 12: Ablation study ($M = 4$) of gradient assistance weights by adding noises to the transmitted pseudo-residuals to half of the organizations. The evaluation metrics are the same as Tables 1 and 2.

Dataset	Weight	Diabetes	Boston Housing	Blob	Iris	Wine	Breast Cancer	QSAR	MNIST	CIFAR10
$\sigma = 1$	✗	46.7(1.0)	4.1(0.1)	46.3(6.5)	80.0(5.3)	85.4(3.0)	91.2(1.4)	72.6(2.2)	78.7(0.1)	47.6(0.3)
	✓	**45(2.8)**	**3.7(0.5)**	**90.0(5.0)**	**95.8(4.3)**	**94.4(3.4)**	**97.8(1.0)**	**79.1(1.1)**	**94.1(0.1)**	**65.4(0.3)**
$\sigma = 5$	✗	59.4(1.1)	5.7(0.4)	13.8(4.1)	54.2(7.6)	61.1(7.1)	75.9(2.9)	64.1(1.8)	38.4(0.3)	22.6(0.5)
	✓	**49.6(3.7)**	**4.1(0.7)**	**66.3(9.6)**	**93.3(2.4)**	**93.7(3.6)**	**97.8(0.4)**	**76.7(1.6)**	**93.0(0.2)**	**59.9(0.6)**

Table 13: Ablation study ($M = 8$) of gradient assistance weights by adding noises to the transmitted pseudo-residuals to half of the organizations. The evaluation metrics are the same as Tables 1 and 2.

Dataset	Weight	Diabetes	Boston Housing	Blob	Wine	Breast Cancer	QSAR	MNIST	CIFAR10	MIMIC3	ModelNet40
$\sigma = 1$	✗	49.0(1.6)	4.3(0.2)	46.3(6.5)	81.2(5.3)	90.8(2.5)	73.2(1.0)	75.1(0.4)	45.4(0.3)	99.3(0.8)	55.8(1.0)
	✓	**46.4(2.3)**	**4.0(0.2)**	**78.8(8.2)**	**88.9(2.0)**	**96.7(1.0)**	**78.9(1.2)**	**92.7(0.1)**	**61.0(0.4)**	**94.9(1.2)**	**78.3(0.9)**
$\sigma = 5$	✗	61.0(2.4)	5.8(0.2)	12.5(2.5)	54.2(6.9)	78.5(2.0)	61.8(0.5)	33.8(0.3)	23.3(0.6)	110.5(0.6)	24.5(0.4)
	✓	**49.7(3.1)**	**4.7(0.5)**	**62.5(9.0)**	**84.7(1.4)**	**96.9(1.3)**	**77.1(0.8)**	**92.1(0.2)**	**57.3(0.3)**	**96.4(0.8)**	**77.5(0.4)**

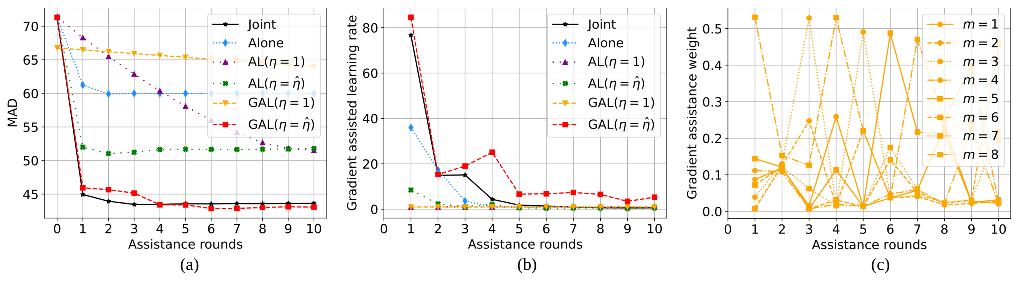


Figure 7: Results from the Diabetes ($M = 8$) dataset.

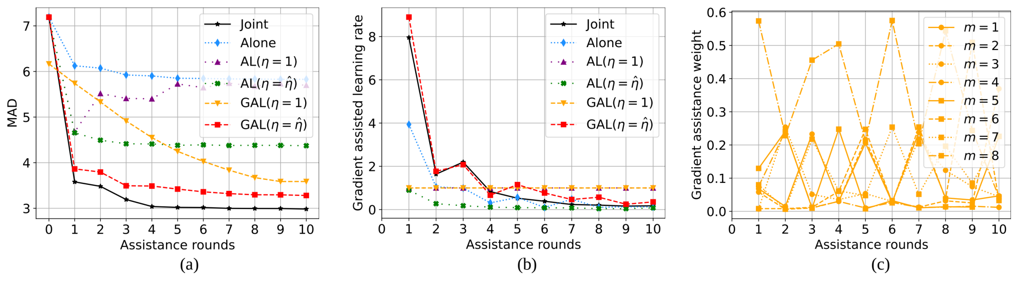


Figure 8: Results from the Boston Housing ($M = 8$) dataset.

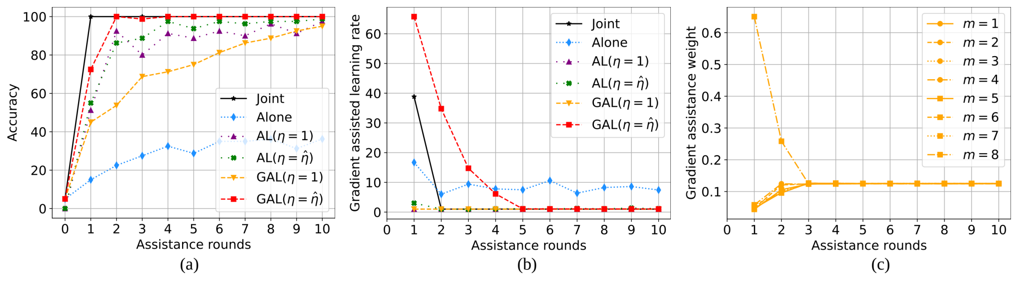


Figure 9: Results from the Blob ($M = 8$) dataset.

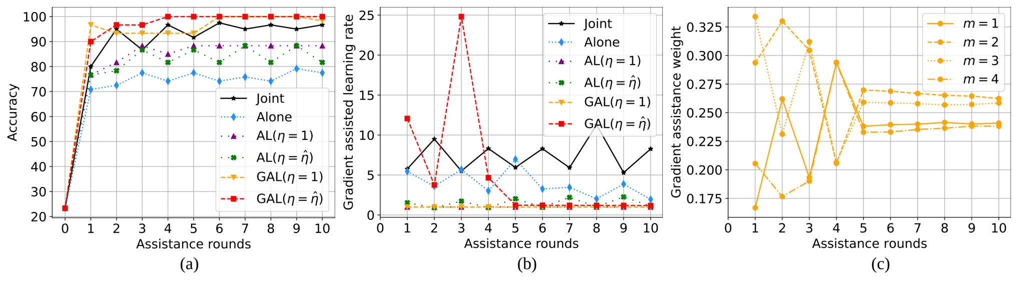


Figure 10: Results from the Iris ($M = 4$) dataset.

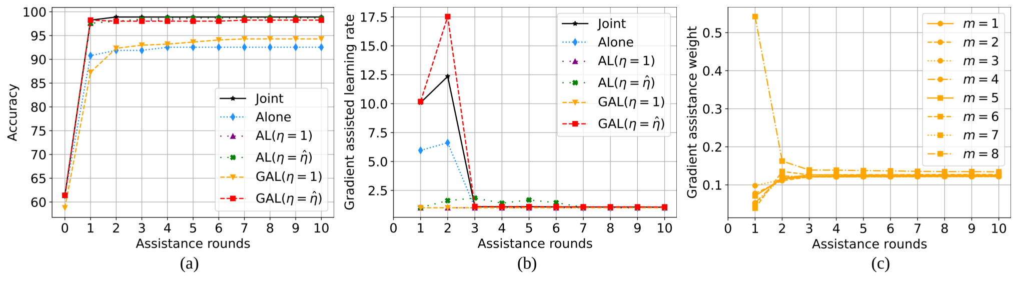


Figure 11: Results from the Breast Cancer ($M = 8$) dataset.

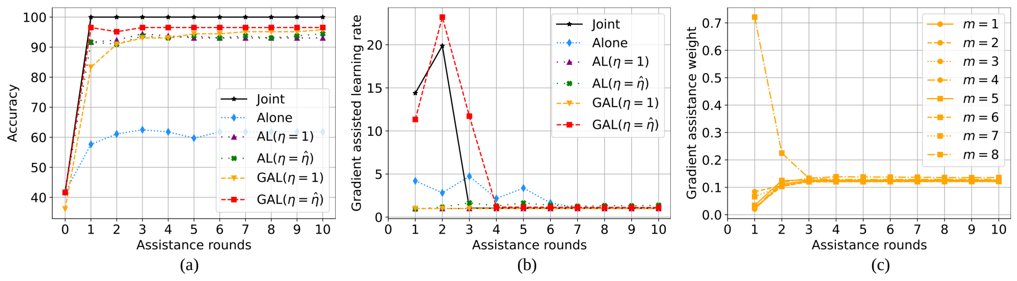


Figure 12: Results from the Wine ($M = 8$) dataset.

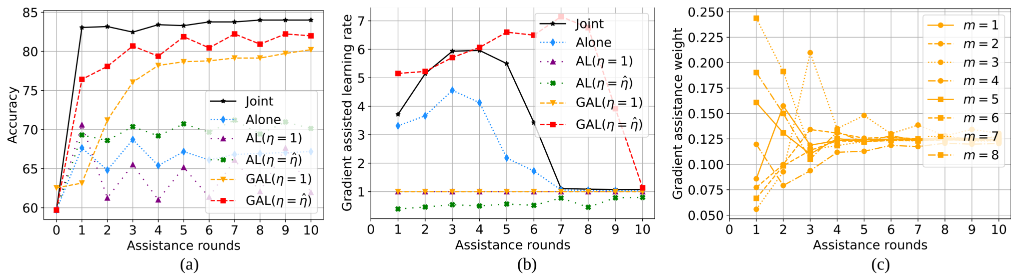


Figure 13: Results from the QSAR ($M = 8$) dataset.

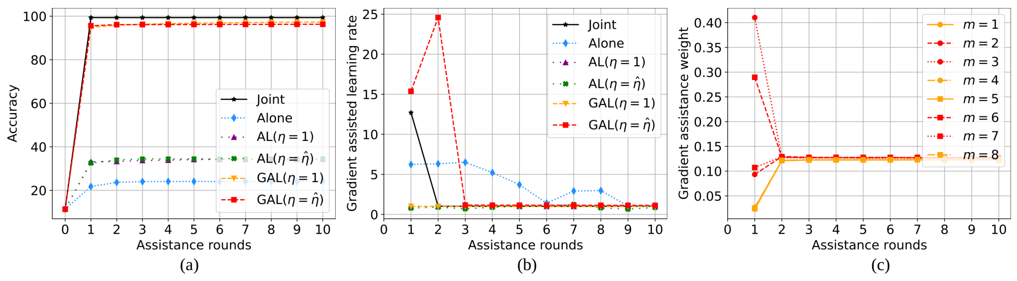


Figure 14: Results from the MNIST ($M = 8$) dataset.

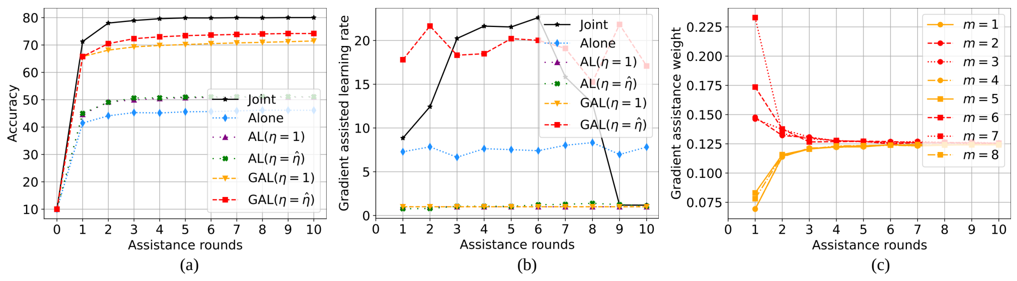


Figure 15: Results from the CIFAR10 ($M = 8$) dataset.

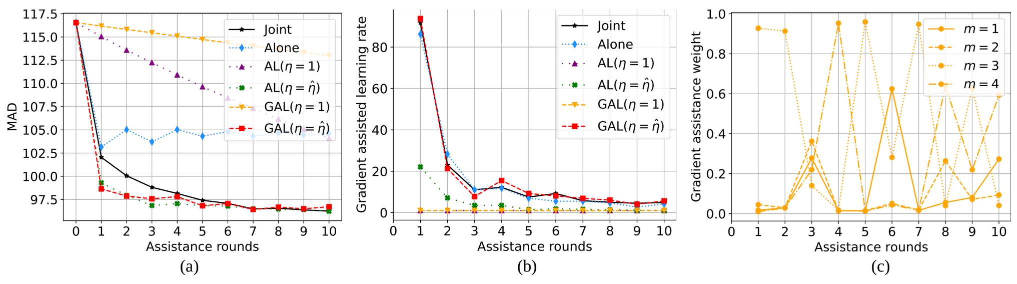
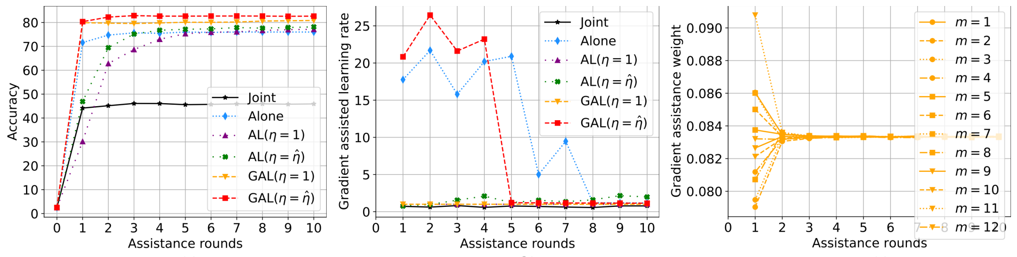


Figure 16: Results from the MIMIC3 ($M = 4$) dataset.



(a) (b) (c)

Figure 17: Results from the ModelNet40 ($M = 12$) dataset.

Algorithm 2 Abstract form of GAL (for theoretical analysis)

```
Learning Stage:
  Intialization:
    Let $t = 0$, and initialize $F^0 \in \mathcal{F}_1$
  for assistance round $t$ from 1 to $T$ do
    for organization $m$ from 1 to $M$ in parallel do
      Optimize each local model by solving
        $$(\alpha_t, f_m^t) = \operatorname*{arg\,min}_{\alpha \in [-a_t, a_t], f_m \in \mathcal{F}_m} \mathcal{L}(F^{t-1} + \alpha f_m) \quad (4)$$
    end
    Gather predictions $f_m^t$, $m = 1, \ldots M$, from all the organizations
    Optimize the gradient assistance weights and learning rate by solving
      $$(\hat{w}^t, \hat{\eta}^t) = \operatorname*{arg\,min}_{w \in P_M, \eta \in [-a_t, a_t]} \mathcal{L}\left(F^{t-1} + \eta \sum_{m=1}^{M} w_m f_m^t\right) \quad (5)$$
    Let $F^t = F^{t-1} + \hat{\eta}^t \sum_{m=1}^{M} \hat{w}_m^t f_m^t$
  end
```

C THEORETICAL ANALYSIS

To develop a convergence analysis of the GAL algorithm, we consider an abstract form of the GAL training procedure as described in Algorithm 2. In particular, we use the following notations. We still let $\mathcal{F}_m$ (for each $m = 1, \ldots, M$) denote a set of real-valued functions defined on organization m's data x_m. For notational simplicity, for each $f_m \in \mathcal{F}_m$, we also treat it as a function of the (artificially) extended variable $x = [x_1, \ldots, x_M]$. So, we may write a function in the form of $f_1 + f_2$, which basically means $[x_1, x_2] \mapsto f_1(x_1) + f_2(x_2)$. Let $\mathcal{L}$ denote the overarching loss function to minimize (for the agent to assist), and P_M the probability simplex.

At round $t = 0$, we initialize with any $F^0 \in \mathcal{F}_1$. At each round t, each organization first runs a greedy boosting step to obtain (α_t, f_m^t). The f_m^t will be sent to us (the organization to assist). Then, we run another greedy step to optimize the assistance weights $\hat{w}^t$ and learning rate $\hat{\eta}^t$, with fixed f_m^t, $m = 1, \ldots, M$. The weighted function will be added to F^{t-1} to generate the latest F^t.

For each m, we let

$$\text{span}(\mathcal{F}_m) = \left\{ \sum_{j=1}^{K_m} \mu_j f_j : \mu_j \in \mathbb{R}, f_j \in \mathcal{F}_m, K_m \in \mathbb{N}^+ \right\},$$

which is the function space formed from linear combinations of elements in $\mathcal{F}_m$. Let

$$\text{span}(\mathcal{F}_1, \ldots, \mathcal{F}_M) = \left\{ \sum_{m=1}^{M} w_m f_m : w_m \in \mathbb{R}, f_m \in \text{span}(\mathcal{F}_m) \right\}$$

denote the linear span of the union of $\mathcal{F}_1, \ldots, \mathcal{F}_M$. An equivalent way to write it is span$(\cup_{m=1}^M \mathcal{F}_m)$.

We will show the following convergence result. With a suitable choice of step parameters a_t and regularity conditions of the loss $\mathcal{L}$, the abstract form of GAL can produce F^t that asymptotically attains the minimum loss within the function class span$(\mathcal{F}_1, \ldots, \mathcal{F}_M)$. We make the following technical assumptions.

(A1) The loss (functional) $f \mapsto \mathcal{L}(f)$ is convex and differentiable on $\mathcal{F}$, with gradient $\nabla\mathcal{L}$. Also, for all $f \in \text{span}(\mathcal{F}_1, \ldots, \mathcal{F}_M)$ and $g \in \cup_{m=1}^M \mathcal{F}_m$, the function $u \mapsto \mathcal{L}(f + ug)$ has a second order derivative $\partial^2\mathcal{L}(f + ug)/\partial u^2$, and it is upper bounded by a fixed constant C.

(A2) The ranges of learning rates $\{a_t\}_{t=1,2,\ldots}$ satisfy $\sum_{t=1}^\infty a_t = \infty$, $\sum_{t=1}^\infty a_t^2 < \infty$.

Theorem 1 *Under Assumptions (A1) and (A2), the GAL solution in Algorithm 2 satisfies* $\mathcal{L}(F^t) \to \inf_{f \in \textit{span}(\mathcal{F}_1, \ldots, \mathcal{F}_M)} \mathcal{L}(f)$ *as* $t \to \infty$.

Remarks on Theorem 1: The result says that with suitable control of the learning rates, the greedy procedure in Algorithm 2 can converge to the oracle one could obtain within span$(\mathcal{F}_1, \ldots, \mathcal{F}_M)$. Suppose that an organization, say the one indexed by $m = 1$, does not collaborate with others. Likewise, we have the convergence for that particular organization, $\lim_{t\to\infty} \mathcal{L}(F^t) = \inf_{f^* \in \text{span}(\mathcal{F}_1)} \mathcal{L}(f^*)$. It can be seen that the GAL will produce a significant gain for this organization as long as

$$\inf_{f^* \in \text{span}(\mathcal{F}_1, \ldots, \mathcal{F}_M)} \mathcal{L}(f^*) < \inf_{f^* \in \text{span}(\mathcal{F}_1)} \mathcal{L}(f^*). \tag{6}$$

It is conceivable that (6) is easy to meet in many practical scenarios since each $\mathcal{F}_m$ is operated on a particular modality of data that belongs to organization m. On the other hand, a skeptical reader may wonder how the GAL solution compares with a function learned from the pulled data. It is possible that the global minimum of $\mathcal{L}$ (over functions that operate on the pulled data) does not belong to span$(\mathcal{F}_1, \ldots, \mathcal{F}_M)$. If that is the case, the best we can do is to find f that attains the limit $\inf_{f \in \text{span}(\mathcal{F}_1, \ldots, \mathcal{F}_M)} \mathcal{L}(f)$. This is a limitation due to the constraint that organizations cannot share data and the additive structure of span$(\mathcal{F}_1, \ldots, \mathcal{F}_M)$. Fortunately, in various real-data experiments we performed, the GAL often performs close to the centralized learning within only a few assistance rounds.

In the technical result, we could allow the approximate minimization of (4) and (5), meaning that the loss of the produced solution is δ_t-away from the optimal loss. In that case, it can be verified that $\sum_{t=1}^\infty \delta_t < \infty$ is sufficient to derive the same asymptotic result in Theorem 1.

The proof of Theorem 1 uses the same technique as was used in (Zhang & Yu, 2005). The technical result here is nontrivial, because f_m^t $(m = 1, \dots, M)$ in each round t are not jointly minimized with $\hat{w}^t$ and $\hat{\eta}^t$ in (5), and thus their linear combination may not be the most greedy solution of minimizing $\mathcal{L}(F^{t-1} + f)$ within $f \in \text{span}(\mathcal{F}_1, \dots, \mathcal{F}_M)$.

A limitation of our theoretical result is that it does not explain our experimental observations that GAL requires only a few rounds of assistance to obtain excellent performance. We leave a more sophisticated convergence analysis as future work.

Proof of Theorem 1:

Let $f^* \in \text{span}(\mathcal{F}_1, \dots, \mathcal{F}_M)$ be an arbitrary fixed function. It is introduced for technical convenience and can be treated as the function that (approximately) attains the infimum of $L(f)$.

For every $f \in \text{span}(\mathcal{F}_1, \dots, \mathcal{F}_M)$, we define the following norm with respect to the basis functions,

$$\|f\|_1 = \inf \biggl\{ \|\mu\|_1 : \sum_{m=1}^{M} \sum_{j=1}^{K_m} \mu_{m,j} f_{m,j} : \mu_{m,j} \in \mathbb{R}, f_{m,j} \in \mathcal{F}_m, K_m \in \mathbb{N}^+ \biggr\}$$

where $\|\mu\|_1$ denotes the abstract sum of its entries, namely $\sum_{m=1,\dots,M, j=1,\dots,K_m} |\mu_{m,j}|$.

For each t, let $S_t \subset \cup_{m=1}^{M} \mathcal{F}_m$ denote the finite set of functions such that
1) $f_m^\tau \in S_t$ for all $0 < \tau < t$, and
2) $f^* = \sum_{g \in S_t} \mu_{f^*}^g g$ $(\mu^g \in \mathbb{R})$, with $\|\mu_{f^*}\|_1 \le \|f^*\|_1 + \varepsilon$.
Note that S_t exists due to the definition of $\|\cdot\|_1$ and the construction of each f_m^τ. Suppose that F^{t-1} admits the representation $F^{t-1} = \sum_{g \in S_t} \mu_{F^{t-1}}^g g$.

From (5), we have

$$\mathcal{L}(F^t) \le \mathcal{L}\bigl(F^{t-1} + \hat{\eta}_t f_m^t\bigr), \quad \forall m = 1, \dots, M. \tag{7}$$

Meanwhile, it follows from (4) that for each m, and each $g \in S_t \cap \mathcal{F}_m$,

$$\mathcal{L}\bigl(F^{t-1} + \hat{\eta}_t f_m^t\bigr) \le \mathcal{L}\bigl(F^{t-1} + a_t s^g g\bigr). \tag{8}$$

where $s^g \triangleq \text{sign}(\mu_{f^*}^g - \mu_{F^{t-1}}^g)$. Combining (7) and (8), we obtain

$$\mathcal{L}(F^t) \le \mathcal{L}\bigl(F^{t-1} + a_t s^g g\bigr), \quad \forall g \in S_t. \tag{9}$$

Applying Taylor expansion to $f \mapsto \mathcal{L}(f)$ at $f = F^{t-1}$, and invoking (9) and Assumption (A1), we have

$$\mathcal{L}(F^t) - \mathcal{L}(F^{t-1}) \le \mathcal{L}(F^{t-1} + a_t s^g g) - \mathcal{L}(F^{t-1}) \le a_t s^g \nabla \mathcal{L}(F^{t-1})^{\mathrm{T}} g + \frac{C}{2} a_t^2 \tag{10}$$

for all sufficiently small $a_t > 0$. Let $\|\mu_{f^*} - \mu_{F^{t-1}}\|_1 \triangleq \sum_{g \in S_t} |\mu_{f^*}^g - \mu_{F^{t-1}}^g|$. Multiplying both sides by $|\mu_{f^*}^g - \mu_{F^{t-1}}^g|$, and add up all the $g \in S_t$, we have

$$\begin{aligned} \|\mu_{f^*} - \mu_{F^{t-1}}\|_1 \cdot \{\mathcal{L}(F^t) - \mathcal{L}(F^{t-1})\} &\le a_t \nabla \mathcal{L}(F^{t-1})^{\mathrm{T}} (f^* - F^{t-1}) + \frac{C}{2} a_t^2 \\ &\le a_t \{\mathcal{L}(f^*) - \mathcal{L}(F^{t-1})\} + \frac{C}{2} a_t^2 \end{aligned} \tag{11}$$

where the last inequality is due to the convexity of $\mathcal{L}$. If $\|\mu_{f^*} - \mu_{F^{t-1}}\|_1 = 0$, F^{t-1} already converges to f^*. Otherwise, we rearrange (11) to obtain

$$\mathcal{L}(F^t) - \mathcal{L}(f^*) \le \biggl(1 - \frac{a_t}{\|\mu_{f^*} - \mu_{F^{t-1}}\|_1}\biggr) \{\mathcal{L}(F^{t-1}) - \mathcal{L}(f^*)\} + \frac{C}{2} a_t^2 \tag{12}$$

$$\le \biggl(1 - \frac{a_t}{\|\mu_{f^*}\|_1 + 1 + \sum_{\tau=0}^{t-1} a_\tau}\biggr) \{\mathcal{L}(F^{t-1}) - \mathcal{L}(f^*)\} + \frac{C}{2} a_t^2, \tag{13}$$

where the last inequality is due to the triangle inequality, the way F^{t-1} is constructed, and the fact that ε can be arbitrarily chosen. Here, we defined $a_0 \triangleq 0$. Let $a_{1:t} = \sum_{\tau=1}^{t} a_\tau$ for each $t \geq 1$. Applying (13) and the Lemma 4.2 in (Zhang & Yu, 2005), we have

$$\max(0, \mathcal{L}(F^t) - \mathcal{L}(f^*)) \leq \frac{\|\mu_{f^*}\|_1 + 1}{\|\mu_{f^*}\|_1 + a_{1:t}} + \frac{C}{2} \sum_{\tau=1}^{t} \frac{\|\mu_{f^*}\|_1 + a_{1:\tau}}{\|\mu_{f^*}\|_1 + a_{1:t}} a_\tau^2. \tag{14}$$

Since f^* is arbitrarily chosen, it can be seen from (14) and Assumption (A2) that $\lim_{t\to\infty} \mathcal{L}(F^t) = \inf_{f^* \in \text{span}(\mathcal{F}_1, \ldots, \mathcal{F}_M)} \mathcal{L}(f^*)$.