

A Primal-Dual Algorithm for Hybrid Federated Learning

Tom Overman¹, Garrett Blum², Diego Klabjan³

¹Department of Engineering Sciences and Applied Mathematics, Northwestern University

²Department of Mechanical Engineering, Northwestern University

³Department of Industrial Engineering and Management Sciences, Northwestern University
tomoverman2025@u.northwestern.edu, garrettblum2024@u.northwestern.edu, d-klabjan@northwestern.edu

Abstract

Very few methods for hybrid federated learning, where clients only hold subsets of both features and samples, exist. Yet, this scenario is extremely important in practical settings. We provide a fast, robust algorithm for hybrid federated learning that hinges on Fenchel Duality. We prove the convergence of the algorithm to the same solution as if the model is trained centrally in a variety of practical regimes. Furthermore, we provide experimental results that demonstrate the performance improvements of the algorithm over a commonly used method in federated learning, FedAvg, and an existing hybrid FL algorithm, HyFEM. We also provide privacy considerations and necessary steps to protect client data.

1 Introduction

Federated learning (FL) has quickly become a top choice for privacy-aware machine learning (Li et al. 2020a). The basic premise of federated learning is that a group of external nodes called clients hold parts of the data and a central server coordinates the training of a model representative of these data but without directly accessing the data itself. This requires the clients to train local models, then pass some information (such as model weights) to the server where the server aggregates the clients' contributions to update its global model. The goal of FL is to build algorithms that result in convergence to a similar objective value as the centralized case, as if the server had access to all data directly, and perform well over various problem settings with minimal communication overhead.

Federated learning can be classified based on how the data are gathered on the clients. In horizontal FL, each client holds a subset of the samples that contain all of their features. In vertical FL, each client holds all of the samples but only a subset of each sample's features. These are both special cases of hybrid FL where each client contains a subset of the samples and a subset of the features.

Hybrid FL is less studied than the case of horizontal and vertical FL, but it is still extremely important in practice. An example of hybrid FL is the case where multiple hospitals wish to build a central model but cannot directly share data between hospitals due to privacy laws. Each hospital has a

subset of all of the patients, and since each patient may have visited multiple hospitals, the patient's features are split between many hospitals. The same situation exists in banking for fraud detection with explainable convex models (Lv et al. 2021).

Another example is in telecommunication where each tower collects data from cell devices that ping the tower. Each cell tower has different specifications and thus collects different measurements than other towers. Therefore, each tower collects different features, and since not every user connects to every tower and most users interact with multiple different towers, the samples are also split across towers.

We introduce a primal-dual algorithm, Hybrid Federated Dual Coordinate Ascent (HyFDCA), that solves convex problems in the hybrid FL setting. This algorithm extends CoCoA, a primal-dual distributed optimization algorithm introduced by Jaggi et al. (2014) and Smith et al. (2017), to the case where both samples and features are partitioned across clients. We provide privacy considerations that ensure that client data cannot be reconstructed by the server. Next, we provide proofs of convergence under various problem settings including special cases where only subsets of clients are available for participation in each iteration. The algorithm and associated proofs can also be utilized in the distributed optimization setting where both samples and features are distributed. As far as we know, this is the only algorithm in the doubly distributed case that has guaranteed convergence outside of block-splitting ADMM developed by (Parikh and Boyd 2014). ADMM has not been designed with FL in mind, but the algorithm has no data sharing. On the down side, block-splitting ADMM requires full client participation which makes it much more restrictive than HyFDCA and essentially impractical for FL. HyFDCA is also the only known hybrid FL algorithm that converges to the same solution as if the model is trained centrally. Finally, we provide extensive experimental results that demonstrate the performance improvements of HyFDCA over FedAvg, a commonly-used FL algorithm (McMahan et al. 2017), and HyFEM, a hybrid FL algorithm (Zhang et al. 2020).

Our main contributions in this work are as follows:

1. Provide HyFDCA, a provably convergent primal-dual algorithm for hybrid FL. The proofs cover a variety of FL problem settings such as incomplete client participation. Furthermore, the convergence rates provided for the spe-

cial cases of horizontal and vertical FL match or exceed the rates of popular FL algorithms designed for those particular settings.

2. Provide the privacy steps that ensure privacy of client data in the primal-dual setting. These principles apply to future efforts in developing primal-dual algorithms for FL.
3. Demonstrate that HyFDCA empirically outperforms both FedAvg and HyFEM in the loss function value and validation accuracy across a multitude of problem settings and datasets. We also introduce a hyperparameter selection framework for FL with competing metrics using ideas from multiobjective optimization.

In Section 2, we discuss work that has been done in the vertical and horizontal settings and the lack of algorithms that exist for the hybrid setting. We then highlight the improvements that HyFDCA provides in theory and practice. In Section 3, we introduce HyFDCA and privacy considerations that protect client data. In Section 4, we analyze convergence of HyFDCA and provide convergence results in a variety of practical FL problem settings. In Section 5, we present experimental results on three separate data sets and compare the performance of HyFDCA with FedAvg and HyFEM.

2 Related Work

There has been significant work in developing primal-dual algorithms using Fenchel Duality for distributed optimization where samples are distributed. One of the leading frameworks on this front is CoCoA. However, these algorithms do not properly handle data that is distributed over both samples and features. This extension from partitioning data over a single axis direction to both directions is not trivial, especially in the primal-dual case where now multiple clients share different copies of the same dual variables and primal weights. D3CA is the first algorithm to extend CoCoA to the case where data are distributed over samples and features (Nathan and Klabjan 2017). However, D3CA has no convergence analysis and has convergence problems in practice with small regularization constant. HyFDCA fixes these issues with D3CA and is altered to ensure that the privacy requirements for FL are met. Block-splitting ADMM is the only other algorithm that can handle distributed samples and features. However, as (Nathan and Klabjan 2017) show, the empirical performance of block-splitting ADMM is poor and full client participation is needed. HyFDCA and the associated proofs, while focused on the federated setting, can also be utilized in the distributed optimization setting where both samples and features are distributed.

There has been substantial work in horizontal FL where samples are distributed across clients but each sample contains the full set of features. One of the most commonly used algorithms is FedAvg which, in essence, computes model weights on each client using stochastic gradient descent (SGD), then averages together these model weights in an iterative fashion. FedAvg can be naively extended to the hybrid FL case by computing client weights locally, as before, then concatenating the model weights and averaging

at the overlaps. From now on, this modified version of FedAvg is what is meant when referencing FedAvg in the hybrid FL setting. Empirical results in Section 5 demonstrate that this naive extension is not satisfactory, and focused algorithms with satisfactory performance specifically for hybrid FL must be developed. The convergence rate of HyFDCA matches FedAvg (Li et al. 2020b) in the special case of horizontal FL. Furthermore, HyFDCA does not require smooth loss functions unlike FedAvg, making the convergence results more flexible.

FedDCD is an approach for using dual methods for FL, but is limited to the regime of horizontal FL (Fan, Fang, and Friedlander 2022). The extension to hybrid FL is not clear as now multiple clients hold copies of the same dual variables and the local coordinate descent that FedDCD performs is no longer valid. Furthermore, the proof results given for FedDCD require smooth loss functions which eliminate many common loss functions such as hinge loss. Our convergence results do not require smoothness of the loss functions. FedDCD does not mention privacy considerations in the case that the mapping between primal and dual variables can be inverted to reveal information about the data held on clients. We address these privacy concerns with suitable homomorphic encryption steps in HyFDCA. We note that to the best of our knowledge, HyFDCA is the only primal-dual algorithm that can handle vertical FL.

There has been substantially less work in vertical FL where each client has all of the samples but only a subset of the features. Some approaches exist such as FedSGD (vertical variant) and FedBCD which rely on communicating relevant information between clients to compute stochastic gradients despite only holding a portion of the features (Liu et al. 2022). However, these algorithms do not work in the hybrid FL case we are exploring. Furthermore, they require communication of this gradient information between clients instead of just passing information through the server. In addition, the convergence rate of HyFDCA in the special case of vertical FL is faster than FedBCD, demonstrating that while HyFDCA is designed to handle hybrid FL, it also enjoys improvements over existing methods in the special cases of horizontal and vertical FL.

To the best of our knowledge, there is only one other algorithm that focuses on hybrid FL, HyFEM. This algorithm uses a feature matching formulation that balances clients building accurate local models and the server learning an accurate global model. This requires a matching regularizer constant that must be tuned based on user goals and results in disparate local and global models. Furthermore, the convergence results provided for HyFEM only claim convergence of the matching formulation not of the original global problem and require complete client participation. In other words, though HyFEM can converge to a solution, it may be substantially worse or simply divergent with incomplete client participation than if the same data are used to train a model centrally as we show in Section 5.2. This work is substantially different than our approach which uses data on local clients to build a global model that converges to the same solution as if the model is trained centrally. Properly tuning the matching constant takes significant computational re-

sources which is not a problem for HyFDCA. Furthermore, due to HyFEM not solving the original optimization problem, we show that the empirical results for convex problems are significantly worse than HyFDCA.

3 The Primal-Dual Algorithm

The goal is to solve the following minimization problem that consists of a strongly convex, L2 regularizer and a sum of convex loss functions

$$\min_{w \in \mathbb{R}^M} P(w) = \frac{\lambda}{2} \|w\|^2 + \frac{1}{N} \sum_{i=1}^N l_i(w^T x_i) \quad (1)$$

where w are the weights of the model, M is the total number of features, N is the total number of samples, λ is the regularization parameter that influences the relative importance of regularization, x_i is the i -th sample, and l_i are sample specific loss functions. This class of problems encompasses many important models in machine learning including logistic regression and support vector machines (SVM).

Our approach takes advantage of the Fenchel Dual of this problem, which is defined as

$$\max_{\alpha \in \mathbb{R}^N} D(\alpha) = -\frac{\lambda}{2} \left\| \frac{1}{\lambda N} \sum_{i=1}^N \alpha_i x_i \right\|^2 - \frac{1}{N} \sum_{i=1}^N l_i^*(-\alpha_i) \quad (2)$$

where α are the dual variables and l^* is the convex conjugate of l . There is a convenient relationship between optimal primal, w^* , and dual variables, α^* , defined by $w^* = \frac{1}{\lambda N} \sum_{i=1}^N \alpha_i^* x_i$. When l_i are convex, we have that $P(w^*) = D(\alpha^*)$.

3.1 The HyFDCA Algorithm

The main idea of HyFDCA, shown in Algorithm 1, is that each client performs a local dual coordinate ascent to find updates to the dual variables. This local method utilizes the inner product of the primal weights and the data, and thus a secure way of finding this inner product across clients that contain sections of each sample is provided in Algorithm 2. The details of the local dual method are shown in Algorithm 3. These dual updates from clients are averaged together and then used to update the global dual variables held on the server. These updated dual variables are then sent back to the clients where they each compute their local contribution of the primal weights. These are then sent back to the server and aggregated. The steps to compute these global primal weights are shown in Algorithm 4. A diagram of HyFDCA that demonstrates each step is shown in Figure 1.

We introduce some additional notation. Set \mathcal{B}_n is the set of clients that contain sample n ; \mathcal{I}_k is the set of samples available on client k ; \mathcal{M}_k is the set of features available to client k ; and \mathcal{K}_m is the set of clients that contain feature m . Furthermore, $x_{k,i}$ is the subset of sample i available to client k , and $x_{k,i,m}$ is the value of feature m of sample i located on client k .

Due to the mapping between primal and dual variables, $w = \frac{1}{\lambda N} \sum_{i=1}^N \alpha_i x_i = \mathbf{A}\alpha$, care needs to be taken to prevent the reconstruction of \mathbf{A} from iterates of w and α . The

Algorithm 1: HyFDCA

```

Initialize  $\alpha^0 = 0, w^0 = 0$ , and  $\hat{w}_0 = 0$ .
Set  $ip_{k,i} = 0$  for every client  $k$  and  $i \in \mathcal{I}_k$ .
for  $t=1,2,\dots,T$  do
    Given  $\mathcal{K}^t$ , the subset of clients available in the given iteration
    Find  $\overline{\mathcal{K}}^t = \{k : k \in \mathcal{K}^t \text{ and } k \notin \overline{\mathcal{K}}^{t-1}\}$ 
    Send  $\text{enc}(\alpha_0^t)$  to clients  $k \in \overline{\mathcal{K}}^t$ 
    PrimalAggregation( $\overline{\mathcal{K}}^t$ )
    SecureInnerProduct( $\mathcal{K}^t$ )
    for all clients  $k \in \mathcal{K}^t$  do
         $\Delta\alpha_k^t = \text{LocalDualMethod}(\alpha_k^{t-1}, w_k^{t-1}, x_i^T w_0^{t-1})$ 
        Send all  $\text{enc}(\frac{\gamma^t}{|\mathcal{B}_n|} \Delta\alpha_k^t)$  to server
    end for
    for  $n=1,2,\dots,N$  do
         $\text{enc}(\alpha_{0,n}^t) = \text{enc}(\alpha_{0,n}^{t-1}) + \sum_{b \in \mathcal{B}_n} \text{enc}(\frac{\gamma^t}{|\mathcal{B}_n|} \Delta\alpha_{b,n}^t)$ 
    end for
    Send  $\text{enc}(\alpha_0^t)$  to clients  $k \in \mathcal{K}^t$  and clients decrypt
    PrimalAggregation( $\mathcal{K}^t$ )
    SecureInnerProduct( $\mathcal{K}^t$ )
end for

```

server could collect w^t and α^t for many t and construct a system of linear equations $\mathcal{W} = \mathbf{A}\mathcal{A}$ where \mathcal{W} collects iterates of w in its columns and \mathcal{A} collects iterates of α in its columns. This would allow for the solution of \mathbf{A} or the approximate solution using least-squares if \mathbf{A} is not square. For this reason, either α or w should be encrypted to prevent this reconstruction of data. Because w is used by the central model for inference on new data, we choose to encrypt α using homomorphic encryption. All aggregations are done on the server, and the server knows the sample IDs of each client through private set intersection (Lu and Ding 2020).

Homomorphic encryption is a technique for encrypting data and preserving certain arithmetic operations in the encrypted form (Gentry 2009). For example, in additive homomorphic encryption the following holds: $\text{enc}(X) + \text{enc}(Y) = \text{enc}(X + Y)$. There are numerous algorithms for homomorphic encryption and new, faster algorithms are invented frequently. For example, the Paillier cryptosystem takes on average 18.882 ms for encryption, 18.865 ms for decryption, and 0.054 ms for addition in the encrypted state (Sidorov, Wei, and Ng 2022). HyFDCA uses homomorphic encryption in several steps to ensure that the server can perform aggregation operations but not reconstruct the underlying data that belongs to the clients.

In addition, the communication of the inner product information poses a similar problem. If we define $b_i^t = (w^t)^T x_i$, then the server could collect iterates of b and w and form a system of linear equations $B = \mathcal{W}x_i$ where \mathcal{W} collects w in its rows and B is a column vector of the corresponding b_i . This system could then be solved for x_i . For this reason, the inner product components passed to the server from the clients must be encrypted using additive homomorphic encryption. So far, we have addressed the server reconstructing data, however, another concern is the clients themselves re-

constructing data from other clients. It is important that the clients are only sent the dual variables corresponding to the samples on that client and only the primal weights corresponding to the features on that client. With this information they would only be able to reconstruct their local data.

The dual and primal variables are sent between the server and clients at the beginning and end of each iteration to ensure that clients are not using stale information because they may not participate in every iteration. Several round-trip communications (RTC) are necessary for the difficult case of hybrid FL with incomplete client participation. For the easier cases of horizontal and vertical FL explored in previous works, HyFDCA can be simplified without changing the framework, resulting in less communications. For horizontal FL with complete client participation, SecureInnerProduct and the first instance of PrimalAggregation can be removed which reduces each iteration to two RTC. Furthermore, if we assume complete client participation, then the first instances of SecureInnerProduct and PrimalAggregation can be removed and each iteration only has three RTC even in the hybrid FL setting.

Algorithm 2: SecureInnerProduct

Input: Set of available clients \mathcal{K}

for all clients $k \in \mathcal{K}$ **do**

for all samples $i \in \mathcal{I}_k$ **do**

 Client k computes local $x_{k,i}^T w_k^t$ and encrypts this scalar using additive homomorphic encryption resulting in $\text{enc}(x_{k,i}^T w_k^t)$.

$ip_{k,i} = \text{enc}(x_{k,i}^T w_k^t)$.

 Send $ip_{k,i}$ to server.

end for

end for

for all samples $i = 1, 2, \dots, N$ **do**

 Server computes $\text{enc}(x_i^T w^t) = \sum_{k \in \mathcal{B}_i} ip_{k,i}$.

 Send to all clients $k \in \mathcal{K}$ all values $\text{enc}(x_i^T w^t)$ for $i \in \mathcal{I}_k$.

end for

Clients decrypt $\text{enc}(x_i^T w_0^t)$ to obtain $x_i^T w_0^t$.

Algorithm 3: LocalDualMethod

Input: $\alpha_k^{t-1}, w_k^{t-1}, x_i^T w_0^t$

D is a set of sample indices available to client k of size H randomly chosen without replacement

Let $\Delta \alpha_{k,i}^t = 0$ for all $i \in \mathcal{I}_k$

for $i \in D$ **do**

 Let $u_i^{t-1} \in \partial l_i(x_i^T w_0^{t-1})$

$s_{k,i} = \arg \max_{s \in [0,1]} \{-l_i^*(-(\alpha_{k,i}^{t-1} + sc_k \gamma_t (u_i^{t-1} - \alpha_{k,i}^{t-1}))) - sc_k \gamma_t (w^{t-1})^T x_i (u_i^{t-1} - \alpha_{k,i}^{t-1}) - \frac{\gamma_t^2}{2\lambda} (sc_k (u_i^{t-1} - \alpha_{k,i}^{t-1}))^2\}$

$\Delta \alpha_{k,i}^t = s_{k,i} c_k (u_{k,i}^{t-1} - \alpha_{k,i}^{t-1})$

end for

Return $\Delta \alpha_k^t$.

Algorithm 4: PrimalAggregation

Input: Set of available clients \mathcal{K}

for all clients $k \in \mathcal{K}$ **do**

for all features $m \in \mathcal{M}_k$ **do**

$\hat{w}_{k,m} = \sum_{i \in \mathcal{I}_k} \alpha_{k,i}^t x_{k,i,m}$

end for

end for

Update global $\hat{w}_{0,k,m}$ from available local $\hat{w}_{k,m}$

for all features $m = 1, 2, \dots, M$ **do**

$w_{0,m}^t = \frac{1}{\lambda N} \sum_{k \in \mathcal{K}_m} \hat{w}_{0,k,m}$

end for

Send w_0^t to clients $k \in \bar{\mathcal{K}}$

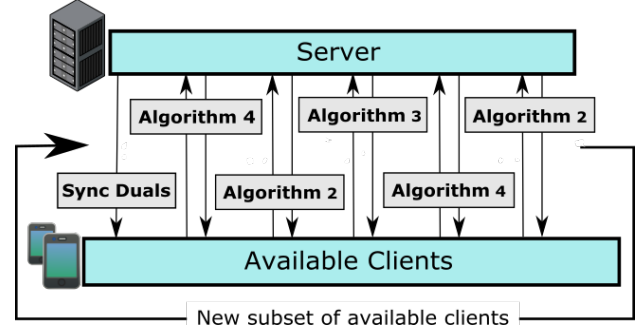


Figure 1: Flowchart of HyFDCA. Each vertical arrow represents a communication of some information between clients and the server.

4 Convergence Analysis

We provide convergence proofs for HyFDCA in various problem settings. The proofs are in the Supplementary Materials of the extended paper (Overman, Blum, and Klabjan 2023). We also note that no assumptions of IID data are made in these proofs, so they apply for non-IID settings.

4.1 Hybrid Federated Setting with Complete Client Participation

We first make the following assumptions of our problem setting.

Assumption 4.1. Loss functions $l_i \geq 0$ are convex and L -Lipschitz functions. This is satisfied by many commonly-used loss functions in practice including logistic regression and hinge loss (support vector machines).

Assumption 4.2. The set of clients, \mathcal{K} , available at a given outer iteration is the full set of clients.

Assumption 4.3. The data are split among clients in the particular way shown in Figure 2. The only assumption we make is that each sample on a particular client has the same features (the data are rectangular).

Theorem 4.4. *If Assumptions 4.1-4.3 are met, $\gamma_t = 1$, and $c_k = N_k/N$ where N_k is the number of samples on client k , then Algorithm 1 results in the bound on the dual suboptimality gap, $\mathbb{E}[\varepsilon_D^t] \leq (1 - \frac{s_t H}{N}) \mathbb{E}[\varepsilon_D^{t-1}] + \frac{s_t^2 H}{N} G$, for any $s_t \in [0, 1]$ and $G \leq \frac{2L^2}{\lambda}$, where $\varepsilon_D^t = D(\alpha^*) - D(\alpha^t)$.*

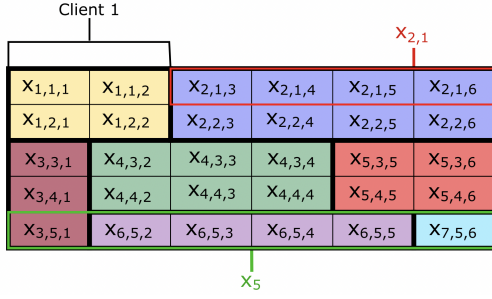


Figure 2: Diagram of a possible data partitioning that follows Assumption 4.3 for the convergence proof. Each client is shaded in a different color.

Next, we find the bound on the suboptimality gap in terms of the number of iterations.

Theorem 4.5. *If $\frac{H}{N} \leq 1$, then for every $t \geq t_0$ we have*

$$\mathbb{E}[\varepsilon_D^t] \leq \frac{2G}{1 + \frac{H}{2N}(t - t_0)} \quad (3)$$

where $t_0 = \max\{0, \lceil \log(\mathbb{E}[\varepsilon_D^0]/G) \rceil\}$. This upper bound clearly tends to zero as $t \rightarrow \infty$.

The requirement of $H \leq N$ places an upper limit on the number of inner iterations that can be performed before aggregation across clients.

4.2 Horizontal Federated Setting with Random Subsets of Available Clients

The case of incomplete client participation for hybrid FL is difficult because of the presence of stale variables due to clients not participating in some iterations affecting several steps of the algorithm. More details on where these stale variables impact the algorithm are given in Section 4.3. The PrimalAggregation and SecureInnerProduct steps before and after the local updates alleviate these issues in practice, but problems still exist for convergence proofs. For this reason, we first approach the incomplete client participation case for horizontal FL where this problem does not exist.

Assumption 4.6. Data are split among clients such that every client has the full set of features but only a subset of samples (definition of horizontal FL). Furthermore, we assume that each client holds N/K samples, where K is the total number of clients.

Assumption 4.7. Each outer iteration, a random subset of clients is chosen to participate and perform updates. Each client has an equal probability of being chosen and the mean number of clients chosen for a given outer iteration is P . Thus the mean fraction of clients participating in a given outer iteration is $\frac{P}{K}$.

Theorem 4.8. *If Assumptions 4.1, 4.6, and 4.7 are met, $\gamma_t = 1$, $c_k = 1$, and $\frac{PH}{N} \leq 1$, then Algorithm 1 results in the dual suboptimality gap for every $t \geq t_0$ as follows*

$$\mathbb{E}[\varepsilon_D^t] \leq \frac{2G}{1 + \frac{PH}{2N}(t - t_0)} \quad (4)$$

where $t_0 = \max\{0, \lceil \log(\mathbb{E}[\varepsilon_D^0]/G) \rceil\}$ and $G \leq \frac{2L^2}{\lambda}$. This upper bound clearly tends to zero as $t \rightarrow \infty$.

The requirement on $PH \leq N$ similarly places a limit on the amount of inner iterations that can be performed before aggregation. This result demonstrates that HyFDCA enjoys a convergence rate of $\mathcal{O}(\frac{1}{t})$ which matches the convergence rate of FedAvg. However, our convergence proof does not assume smooth loss functions whereas FedAvg does. This makes our convergence results more flexible in the horizontal FL setting.

4.3 Vertical Federated Setting with Incomplete Client Participation

We now explore the case of incomplete client participation for the vertical federated setting. We change the assumptions for how subsets of clients are available for participation in Assumption 4.10 because random client subsets impose some issues for vertical FL. If a particular α_i is updated, then w_0 needs to be updated using local data on each client. If one of these clients cannot provide its contribution to w_0 , then these primal weights will be stale and thus $(w_0^{t-1})^T x_i$ used in LocalDualMethod will also be stale. We require a limit on the maximum number of iterations that a particular client can go without being updated. The reason this cannot be extended to the hybrid case is that if w_0^{t-1} is updated, then a particular client will require $(w_0^{t-1})^T x_i$ for any i , but now that samples are also split across clients, it may not be able to access all components of $(w_0^{t-1})^T x_i$ if some of those clients are not available. This is unlike the vertical FL case where all samples belong to each client.

Assumption 4.9. Data are split among clients such that every client has the full set of samples but only a subset of features (definition of vertical FL).

Assumption 4.10. All of the clients are partitioned into $C \geq 2$ sets where each subset of clients has Q/C clients (we assume that $Q \bmod C = 0$). Let $\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_C$ be this partition. We then assume that client subsets are active (participating in a particular outer iteration) in the cyclic fashion. Thus the sequence of active clients is defined as $\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_C, \mathcal{B}_1, \dots, \mathcal{B}_C, \dots$

Theorem 4.11. *If Assumptions 4.1, 4.9, and 4.10 are met, $\frac{H}{N} \leq 1$, $c_k = 1$, and $\gamma_t = \frac{1}{t}$, then Algorithm 1 results in the following bound on the dual suboptimality gap for $t \geq C$*

$$\mathbb{E}[\varepsilon_D^t] \leq \frac{J_1 + J_2(\ln(t - C + 1) + 1)}{t^{H/N}}$$

where $J_1 = C^{H/N} \mathbb{E}[\varepsilon_D^{C-1}]$, $J_2 = \frac{2HL^2(C+1)}{N\lambda} [(C-1)^4 + 2(C-1)^2 + 1]$, and $\mathbb{E}[\varepsilon_D^{C-1}]$ is bounded by a constant with the standard assumption that $l_i(0) \leq 1$. This converges to zero as $t \rightarrow \infty$.

It is clear that for fastest convergence in an asymptotic sense we want H/N to be large, however, this would also increase the magnitude of J_1 and J_2 which would in turn slow convergence in early iterations when t is small and J_1 and J_2 dominate the bound. Furthermore, if we take $H/N = 1$, then HyFDCA exhibits $\mathcal{O}(\frac{\log t}{t})$ convergence whereas FedBCD exhibits a slower $\mathcal{O}(\frac{1}{\sqrt{t}})$ convergence rate and requires

full client participation. Thus, to the best of our knowledge HyFDCA exhibits the best convergence rates for vertical FL even with partial client participation.

We emphasize that Theorems 4.5-4.11 demonstrate that HyFDCA in a particular federated setting converges to the same optimal solution as if all of the data are collected on a centralized device and trained with a convergent method. Furthermore, Theorems 4.8 and 4.11 demonstrate that in the horizontal and vertical FL cases, HyFDCA is still guaranteed to converge to the optimal solution when only a subset of clients participate in each iteration. The convergence rates for the special cases of horizontal and vertical FL match or exceed the convergence rates of existing FL algorithms in those settings.

5 Experimental Results

We investigate the performance of HyFDCA on several datasets and in several different problem settings (number of clients and percentage of available clients). These different problem settings cover the vast number of different environments seen in practice.

Three datasets are selected. MNIST is a database of handwritten digits where each sample is a 28x28 pixel image (Deng 2012). News20 binary is a class-balanced two-class variant of the UCI “20 newsgroup” dataset, a text classification dataset (Chang and Lin 2011). Finally, Covtype binary is a binarized dataset for predicting forest cover type from cartographic variables (Chang and Lin 2011).

We use the hinge loss function for l_i in experiments. A practical variant of LocalDualMethod, shown in Algorithm 5, is used for experiments. Line 6 of Algorithm 5 has a closed form solution of $\Delta\alpha_{k,i}^t = y_i(\max(0, \min(1, \lambda N(1 - x_i^T w_0^{t-1}) + y_i \alpha_{k,i}^{t-1}))) - \alpha_{k,i}^{t-1}$, where y_i is the class label for the i -th sample. Furthermore, the second occurrence of SecureInnerProduct in Algorithm 1 is omitted for experiments because it does not improve empirical performance and incurred more communication cost.

Algorithm 5: LocalDualMethod (Practical Variant)

Input: $\alpha_k^{t-1}, w_k^{t-1}, x_i^T w_0^t$
 D is a set of sample indices available to client k of size H randomly chosen without replacement
Let $\Delta\alpha_{k,i}^t = 0$ for all $i \in \mathcal{I}_k$
for $i \in D$ **do**
 Find $\Delta\alpha_{k,i}^t$ that maximizes $-l_i^*(-(\alpha_{k,i}^{t-1} + \Delta\alpha_{k,i}^t)) - \frac{\lambda N}{2}(\|w_0^{t-1}\|^2 + \frac{2\Delta\alpha_{k,i}^t}{\lambda N}(w_0^{t-1})^T x_i + (\frac{\Delta\alpha_{k,i}^t}{\lambda N})^2)$
 $\alpha_{k,i}^{t-1} = \alpha_{k,i}^{t-1} + \Delta\alpha_{k,i}^t$
end for
Return $\Delta\alpha_k^t$.

5.1 Implementation

The exact details of the implementation are provided in the Supplementary Materials. Data are inherently non-IID in the hybrid FL case because each client stores different sections of the feature space. We emphasize that the experiments are

performed in the hybrid setting where both samples and features are gathered across different clients. Homomorphic encryption is not actually performed; instead, published time benchmarks of homomorphic encryption is used to estimate the encryption time penalty which is added to the overall wall time. The regularization parameter, λ , is found by tuning via a centralized model where the value of λ that resulted in the highest validation accuracy is employed. The resulting choices of λ are $\lambda_{MNIST} = 0.001$, $\lambda_{News20} = 1 \times 10^{-5}$, and $\lambda_{covtype} = 5 \times 10^{-5}$.

Hyperparameter tuning for federated learning is difficult because there are many competing interests such as minimizing iterations to reach a suitable solution while also minimizing the amount of computation performed on clients due to computational limits on common clients such as smartphones. Therefore, we frame this as a multiobjective optimization problem where an optimal solution must be selected from the Pareto-Optimal front. We chose to use Gray Relational Analysis to solve this (Wang and Rangaiah 2017). The exact metrics used are provided in the Supplementary Materials. For FedAvg, we tuned the number of local iterations of SGD performed as well as a, b in the learning rate $\gamma_t = \frac{a}{b+\sqrt{t}}$. For HyFEM, we tune the aforementioned FedAvg hyperparameters in addition to μ which balances the two losses. For HyFDCA, we only need to tune the number of inner iterations. In each problem setting, the number of clients and fraction of available clients had different hyperparameters tuned for that particular problem.

The plots shown use the relative loss function which is defined as $P_R = \frac{P(w^t) - P_C^*}{P_C^*}$ where P_C^* is the optimal loss function value trained centrally. The x-axis of Figure 3 shows relative outer iteration because each dataset requires a different number of outer iterations, but the number of iterations for each algorithm is kept consistent. Further plotting details are provided in the Supplementary Materials.

5.2 Results

We now discuss the results of the experiments. Due to the large number of problem settings we investigate and the various metrics, only selected plots are displayed in the main body. Complete results are included in the Supplementary Materials.

Figure 3 compares the performance of HyFDCA with FedAvg and HyFEM over a variety of settings with respect to outer iterations. Similarly, Supplementary Materials Figure 1 compares performance with respect to time including communication latency and homomorphic encryption time. These plots correspond to varying levels of difficulty. Intuitively, a large number of clients with a low fraction of participating clients is more difficult than a small number of clients with a high fraction of participating clients. Figure 3 and Supplementary Materials Figure 1 show that HyFDCA converges to a lower relative loss function value and a higher validation accuracy in 69 of 72 comparisons made. The poor performance of FedAvg demonstrates that algorithms designed specifically for horizontal or vertical FL cannot simply be lifted to the hybrid case. HyFEM’s similarly poor performance demonstrates that its main utility is

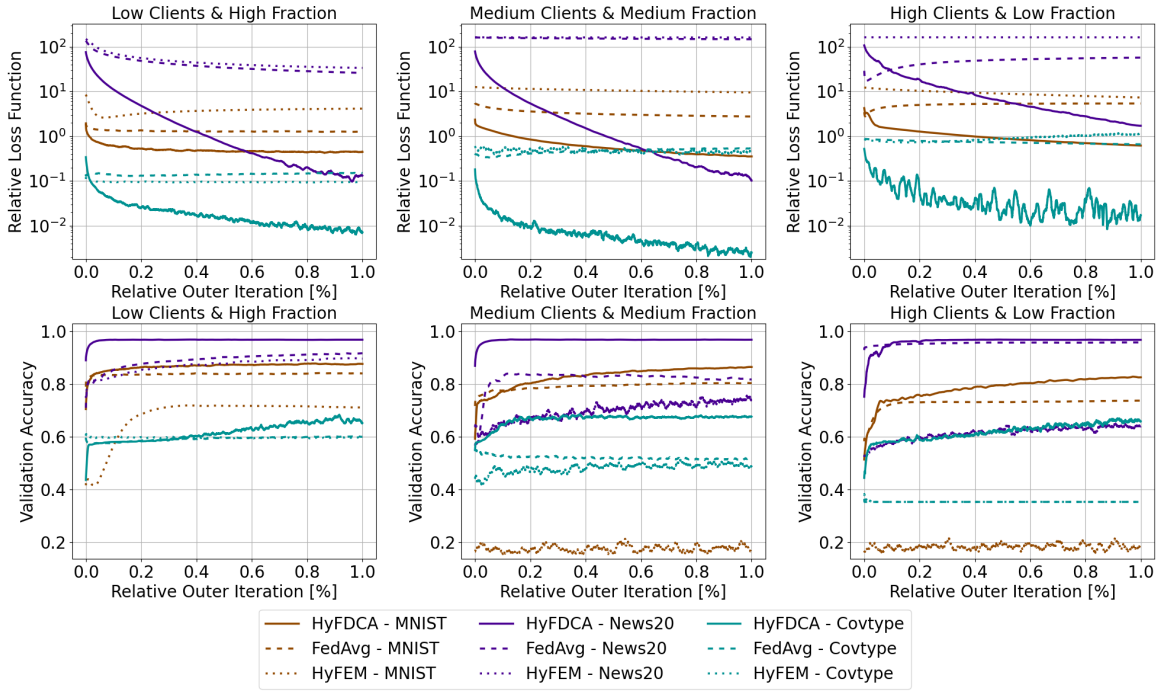


Figure 3: Comparison of HyFDCA, FedAvg, & HyFEM over constant number of outer iterations across client-fraction settings.

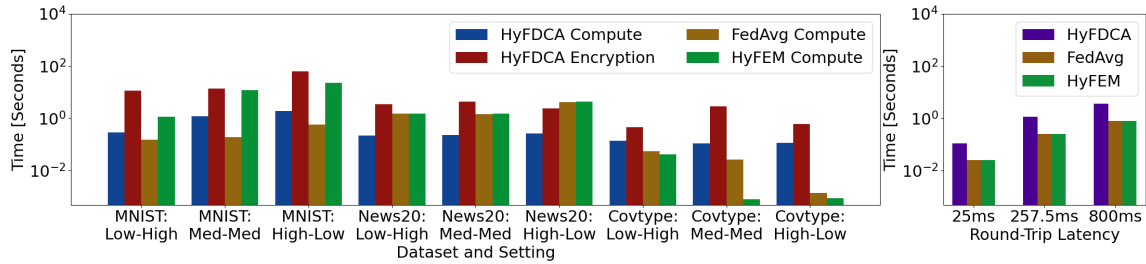


Figure 4: Average time costs of components of each outer iteration for HyFDCA, FedAvg, and HyFEM.

in non-convex problems with significant overlap in feature spaces across clients and where the matching of nonlinear embeddings can be utilized. Moreover, these results are indicative that, though HyFEM can converge, it may result in a substantially worse solution than if trained centrally. Finally, though HyFDCA is a significantly more complex algorithm, HyFDCA often achieves better loss and generalization in a shorter amount of both outer iterations and time even accounting for encryption and latency.

Figure 4 shows the average time cost breakdown per outer iteration of the three algorithms. HyFDCA takes more time per outer iterations than FedAvg or HyFEM. However, the most expensive component of HyFDCA is the homomorphic encryption cost. This is expected to significantly decrease in the future as homomorphic encryption algorithms become much faster due to heavy research efforts. In addition, various methods can be employed to decrease the homomorphic encryption costs such as parallelizing the encryption/decryp-

tion of the vectors or choosing whether to encrypt the primal or the dual variables depending on the dataset.

While HyFDCA is a more complicated algorithm involving more RTC and homomorphic encryption, the clear empirical performance gains over FedAvg and HyFEM make it superior in 69 of 72 comparisons examined. Specifically, HyFDCA converges to a lower loss value and higher validation accuracy in less overall time in 33 of 36 comparisons examined and 36 of 36 comparisons examined with respect to the number of outer iterations. Lastly, HyFDCA only requires tuning of one hyperparameter, the number of inner iterations, as opposed to FedAvg (which requires tuning three) or HyFEM (which requires tuning four). In addition to FedAvg and HyFEM being quite difficult to optimize hyperparameters in turn greatly affecting convergence, HyFDCA’s single hyperparameter allows for simpler practical implementations and hyperparameter selection methodologies.

References

- Adorjan, M. 2020. AWS Inter-Region Latency Monitoring. <https://github.com/mda590/cloudping.co.git>.
- Bergstra, J.; and Bengio, Y. 2012. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13: 281–305.
- Chang, C.-C.; and Lin, C.-J. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3).
- Deng, L. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6): 141–142.
- Fan, Z.; Fang, H.; and Friedlander, M. P. 2022. A dual approach for federated learning. *arXiv:2201.11183*.
- Gentry, C. 2009. Fully Homomorphic Encryption Using Ideal Lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, 169–178. ISBN 9781605585062.
- Harris, C. R.; et al. 2020. Array programming with NumPy. *Nature*, 585(7825): 357–362.
- Jaggi, M.; Smith, V.; Takáč, M.; Terhorst, J.; Krishnan, S.; Hofmann, T.; and Jordan, M. I. 2014. Communication-Efficient Distributed Dual Coordinate Ascent. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, volume 2, 3068–3076.
- Li, T.; Sahu, A. K.; Talwalkar, A.; and Smith, V. 2020a. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine*, 37(3): 50–60.
- Li, X.; Huang, K.; Yang, W.; Wang, S.; and Zhang, Z. 2020b. On the Convergence of FedAvg on Non-IID Data. In *8th International Conference on Learning Representations*.
- Liu, Y.; Zhang, X.; Kang, Y.; Li, L.; Chen, T.; Hong, M.; and Yang, Q. 2022. FedBCD: A Communication-Efficient Collaborative Learning Framework for Distributed Features. *IEEE Transactions on Signal Processing*, 1–12.
- Lu, L.; and Ding, N. 2020. Multi-party Private Set Intersection in Vertical Federated Learning. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 707–714.
- Lv, B.; Cheng, P.; Zhang, C.; Ye, H.; Meng, X.; and Wang, X. 2021. Research on Modeling of E-banking Fraud Account Identification Based on Federated Learning. In *2021 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCCom/CyberSciTech)*, 611–618.
- McMahan, H. B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*, volume 54, 1273–1282.
- Nathan, A.; and Klabjan, D. 2017. Optimization for Large-Scale Machine Learning with Distributed Features and Observations. In *Machine Learning and Data Mining in Pattern Recognition*, 132–146. ISBN 978-3-319-62416-7.
- Overman, T.; Blum, G.; and Klabjan, D. 2023. A Primal-Dual Algorithm for Hybrid Federated Learning. *arXiv:https://arxiv.org/abs/2210.08106*.
- Parikh, N.; and Boyd, S. 2014. Block splitting for distributed optimization. *Mathematical Programming Computation*, 6(1): 77–102.
- Shalev-Shwartz, S.; and Zhang, T. 2013. Stochastic Dual Coordinate Ascent Methods for Regularized Loss. *Journal of Machine Learning Research*, 14(1): 567–599.
- Sidorov, V.; Wei, E. Y. F.; and Ng, W. K. 2022. Comprehensive Performance Analysis of Homomorphic Cryptosystems for Practical Data Processing. *arXiv:2202.02960*.
- Smith, V.; Forte, S.; Ma, C.; Takáč, M.; Jordan, M. I.; and Jaggi, M. 2017. CoCoA: A General Framework for Communication-Efficient Distributed Optimization. *Journal of Machine Learning Research*, 18(1): 8590–8638.
- Telesat. 2017. Real-Time Latency: Rethinking Remote Networks. https://www.telesat.com/wp-content/uploads/2020/07/Real-Time-Latency_HW.pdf.
- T-Mobile USA, I. 2022. Policies: Open Internet. <https://www.t-mobile.com/responsibility/consumer-info/policies/internet-service?INTNAV=fNav:OpenInternet>.
- Virtanen, P.; et al. 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3): 261–272.
- Wang, Z.; and Rangaiah, G. P. 2017. Application and Analysis of Methods for Selecting an Optimal Solution from the Pareto-Optimal Front obtained by Multiobjective Optimization. *Industrial & Engineering Chemistry Research*, 56(2): 560–574.
- Yang, T. 2013. Trading Computation for Communication: Distributed Stochastic Dual Coordinate Ascent. In *Advances in Neural Information Processing Systems*, volume 26.
- Zhang, X.; Yin, W.; Hong, M.; and Chen, T. 2020. Hybrid Federated Learning: Algorithms and Implementation. In *NeurIPS-SpicyFL 2020*.