Global Multiclass Classification from Heterogeneous Local Models

Surin Ahn, Ayfer Özgür, and Mert Pilanci

arXiv:2005.10848v2 [cs.LG] 25 May 2020

Abstract

Multiclass classification problems are most often solved by either training a single centralized classifier on all K classes, or by reducing the problem to multiple binary classification tasks. This paper explores the uncharted region between these two extremes: How can we solve the K-class classification problem by combining the predictions of smaller classifiers, each trained on an arbitrary number of classes $R \in \{2, 3, \ldots, K\}$? We present a mathematical framework for answering this question, and derive bounds on the number of classifiers (in terms of K and R) needed to accurately predict the true class of an unlabeled sample under both adversarial and stochastic assumptions. By exploiting a connection to the classical *set cover* problem in combinatorics, we produce an efficient, near-optimal scheme (with respect to the number of classifiers) for designing such configurations of classifiers, which recovers the well-known one-vs.-one strategy as a special case when $R = 2$. Experiments with the MNIST and CIFAR-10 datasets show that our scheme is capable of matching the performance of centralized classifiers in practice. The results suggest that our approach offers a promising direction for solving the problem of data heterogeneity which plagues current federated learning methods.

I. Introduction

The increasing compute power of edge devices has coincided with the widespread adoption of mobile phones, tablets, and personal wireless devices, which has led to an unprecedented rise in the amount of data generated at the edge. Consequently, there has been a recent surge in research on how best to learn from such vast amounts of decentralized data, much of which is privacy-sensitive (e.g., photos or words typed on a mobile phone). In recent years, *federated learning* (FL) [1], [2], [3] has emerged as a promising decentralized learning paradigm, where a central server trains a shared machine learning model using structured updates from multiple

S. Ahn, A. Özgür, and M. Pilanci are with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (email: surinahn@stanford.edu; aozgur@stanford.edu; pilanci@stanford.edu).

clients. Rather than transmit the actual training data to the server (which would compromise privacy), each client performs some local computation – e.g., several iterations of stochastic gradient descent (SGD) on the shared model using its local training data – and transmits only the resulting model or gradient changes in each round of communication. Typically, these local models are then averaged together by the server and broadcast back to the clients for further updates.

Despite the popularity and success of FL, the presence of *statistically heterogeneous* or *non-i.i.d.* data is known, both theoretically and experimentally, to be detrimental to the convergence of existing federated algorithms [4], [5], [1], [6], [7], [8], [9], [10], [11]. In practice, decentralized data located at different edge devices exhibit very different statistical characteristics, and the sources of such heterogeneity are often quite natural. For instance, users living in different geographical regions are likely to have different types of photos on their mobile phones, or language patterns manifested in text messages. In FL, heterogeneity can be further amplified by irregularities in client participation throughout the learning process.

This paper presents a vastly different perspective on the issue of data heterogeneity in FL, specifically in the context of multiclass classification. We envision having each participating client (or each cluster of similar clients) train a small classifier using just its local training data, and subsequently transmit its fully trained model to the central server. The goal of the server is then to combine the outputs of these heterogeneous, black-box classifiers to solve the larger classification task. This leads us to ask the following fundamental question:

What is the optimal way to construct a global K-class classifier, given only black-box access to smaller R-class classifiers, where $2 \leq R \leq K$?

Note that this approach partially protects the clients' privacy by keeping their training data on the local devices. Moreover, transmitting only the clients' final models to the server may significantly reduce communication costs compared to standard FL. Treating the local classifiers as black boxes allows the clients to use different learning algorithms and architectures (e.g., decision trees, support-vector machines, neural networks).

This is a timely issue which, to the best of our knowledge, has not been previously studied in a systematic manner. We provide many novel angles from which to study this problem. Compared to previously proposed solutions in the FL domain, many of which suggest to maintain separate, personalized models for each client or each cluster of similar clients [7], [12], [13], [6], our

approach seeks to combine local models into a unified, global model, and identify conditions under which the global model works well. Beyond the FL angle, we develop a more general theoretical framework for decomposing classification problems into sub-problems, and present both information-theoretic and graph-theoretic interpretations of our results. The decomposition of classification into binary classification tasks has been studied before, most notably through the ensemble methods of *one-vs.-one* (also known as *all-pairs* or *pairwise coupling*) [14], [15], *one-vs.-all* [16], [17], and *error-correcting output codes (ECOCs)* [18]. A more general framework which encapsulates multiclass to binary reductions using margin-based classifiers was proposed in [19]. One key drawback of one-vs.-all and ECOCs is that they typically require each binary classifier to have access to the entire training set. Importantly, this paper outlines a more general approach to constructing classifiers from not only binary classifiers, but classifiers of arbitrary size which can be trained from only a partial view of the data. Our approach can be viewed as a type of meta or hierarchical ensemble method that combines a diverse set of classifiers which may themselves consist of smaller ensembles. When $R = 2$ (the smaller classifiers are binary), we find that our approach reduces to the one-vs.-one decomposition method.

A. Contributions

This paper makes the following concrete contributions:

- We propose a **mathematical model** for studying the ensembling of smaller R-class classifiers to perform K-class classification, and justify this model through empirical findings.
- Under this model, we derive conditions for achieving **perfect (zero-error) global classification accuracy** under adversarial (worst-case) noise within the local classifiers, and derive bounds which scale as $\Theta(K^2/R^2)$ (up to a $\log$ factor) on the number of smaller classifiers required to satisfy these conditions. We introduce an efficient **voting-based decoding scheme** for predicting the true class of an input given the predictions of the smaller classifiers.
- We show that the conditions for perfect accuracy are intrinsically related to the classical **set cover problem** from the combinatorics and theoretical computer science literature. This connection leads to **efficient, near-optimal algorithms** for designing configurations of smaller classifiers to solve the global problem. We also introduce variations of the original set cover algorithms to address algorithmic questions specific to federated learning.
- We consider a **statistical setting** in which we assume a uniform prior over the classes, uniformly distributed noise within each classifier, and allow a small probability of misclassi-

fication, and show that the required number of smaller classifiers now scales as $\Theta(K/R)$ (up to a $\log$ factor), a significant reduction compared to the perfect accuracy case.

- Through **experiments with the MNIST and CIFAR-10 datasets**, we corroborate our theoretical results and demonstrate the ability of our set covering-based scheme to match the performance of centralized classifiers. Finally, a comparison with the *FedAvg* algorithm [1] shows the potential of our scheme to mitigate the effects of data heterogeneity in federated learning.

II. Problem Formulation

In this paper, we consider black-box access to a set of m classifiers $\mathcal{F}_{m,K} = \{f_i : \mathcal{X} \to \mathcal{Y}_i,\ i \in [m]\}$, with $\mathcal{Y}_i \subseteq [K] \triangleq \{1, 2, \ldots, K\}$ and $2 \leq |\mathcal{Y}_i| \leq K$ for all $i \in [m]$. Each possible input $x \in \mathcal{X}$ belongs to one of the classes in $[K]$. We assume that each classifier f_i was trained to distinguish only between a subset of the classes in $[K]$, i.e. those contained in $\mathcal{Y}_i$, and therefore given an input $x \in \mathcal{X}$ necessarily outputs a class in $\mathcal{Y}_i$. Any classes in $[K] \setminus \mathcal{Y}_i$ are outside of its "universe." This models a distributed setting, where each client has access to data belonging to only a subset of the K classes. Note that f outputs only its final class prediction, rather than confidence scores or conditional probability estimates (we later provide some justification for this modeling decision). We further assume that $\{\mathcal{Y}_i,\ i \in [m]\}$ is known, and we call $|\mathcal{Y}_i|$ the *size* of the classifier f_i. Given a new input $x_k \in \mathcal{X}$ belonging to class $k \in [K]$, we assume that

$$f_i(x_k) = k \ \text{ if } k \in \mathcal{Y}_i.$$

In words, f_i always makes correct predictions on inputs belonging to familiar classes. This model captures the notion that a properly trained classifier should accurately classify a test input whose true class is among its known universe of classes. When $k \notin \mathcal{Y}_i$, then by definition $f_i(x_k) \neq k$ since f_i always maps to $\mathcal{Y}_i$. Hence, f_i's predictions on inputs belonging to classes outside of $\mathcal{Y}_i$ can be considered as undesirable "noise" that we wish to circumvent. We will consider both adversarial (or worst-case) and stochastic models for $f_i(x_k)$ when $k \notin \mathcal{Y}_i$.

In this paper, we study the problem of inferring the class of an unknown input given the "one-shot" outputs of the m classifiers $\mathcal{F}_{m,K}$, i.e., the results of feeding the input a single time to each of the classifiers. Note that this one-shot modeling assumption is fitting in practice, as trained classifiers typically produce the same output when given the same input multiple times. We next define a *K-class classification scheme* constructed out of $\mathcal{F}_{m,K}$.

Definition 1 (Classification Scheme)**.** *A K-class classification scheme is a pair $(\mathcal{F}_{m,K}, g)$ where $\mathcal{F}_{m,K} = \{f_i : \mathcal{X} \to \mathcal{Y}_i,\ i \in [m]\}$ is a set of m local classifiers satisfying $\mathcal{Y}_i \subseteq [K]$, $2 \leq |\mathcal{Y}_i| \leq K$ for all $i \in [m]$, and $g : \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_m \to [K]$ is a decoder which predicts a class upon observing the outputs produced by classifiers in $\mathcal{F}_{m,K}$. Specifically, given an input $x \in \mathcal{X}$, the global class prediction is given by $g(f_1(x), f_2(x), \ldots, f_m(x))$.*

We remark that this problem can be interpreted as an unorthodox communications setting where the transmitter is trying to convey a message $k \in [K]$ to the receiver, but the receiver can only observe the outputs of m channels, each of which is selectively noisy depending on whether k is among its "accepted" symbols. The goal of the receiver is to decode the message k given the m channel outputs.

Broadly, our goal in this paper is to study when and how we can construct an accurate K-class classification scheme given $\mathcal{F}_{m,K}$, or conversely, how we can construct a set $\mathcal{F}_{m,K}$ of m small classifiers of a given size so that accurate K-class classification using $\mathcal{F}_{m,K}$ is possible. In the second case, it is clearly desirable for $\mathcal{F}_{m,K}$ to be minimal. Note that the first problem corresponds to synthesizing a global (bigger) classifier from a given set of local (smaller) classifiers, while the second problem corresponds to decomposing a global classifier into multiple local classifiers. In the rest of the paper, we will study these problems in the following two different settings.

A. Perfect Accuracy Setting

Here, we will require the K-class classification scheme $(\mathcal{F}_{m,K}, \mathcal{D})$ to correctly recover the true class of any input $x \in \mathcal{X}$ for any possible set of outputs from $\mathcal{F}_{m,K}$. More precisely, define the *output set* of a class $k \in [K]$ with respect to a fixed set of classifiers $\mathcal{F}_{m,K} = \{f_i : \mathcal{X} \to \mathcal{Y}_i,\ i \in [m]\}$, denoted by $\mathcal{S}_k$, as the set of all possible classifier outputs given that the true class of the input is k:

$$\mathcal{S}_k \triangleq \left\{ (y_1, \ldots, y_m) \,:\, y_i = k \text{ if } k \in \mathcal{Y}_i,\ y_i \in \mathcal{Y}_i \text{ if } k \in [K] \backslash \mathcal{Y}_i \right\}.$$

Note that $\mathcal{S}_k$ can be constructed using only knowledge of the $\mathcal{Y}_i$. In the perfect accuracy setting, we require the K-class classification scheme to correctly recover k given any observation $y = (y_1, \ldots, y_m) \in \mathcal{S}_k$. Specifically, we say that a scheme $(\mathcal{F}_{m,K}, g)$ achieves perfect K-class classification accuracy if for any $k \in [K]$ and any $y \in \mathcal{S}_k$, we have $g(y) = k$.

Note that this can be regarded as an adversarial or worst-case setting in the sense that for each class $k \in [K]$, the m classifiers are allowed to jointly produce the most confusing output

$y = (y_1, \ldots, y_m)$. In particular, if perfect accuracy can be achieved under this model, it can be achieved under any joint probabilistic model for the behavior of the m classifiers. In the next section, we focus on one probabilistic model, which assumes that the outputs of the m classifiers are independent and uniformly distributed over $\mathcal{Y}_i$ when $k \notin \mathcal{Y}_i$.

B. Statistical Setting

In this setting, given a new test sample $x_k \in \mathcal{X}$ belonging to class $k \in [K]$, we will assume that a local classifier $f_i : \mathcal{X} \to \mathcal{Y}_i$ correctly outputs k if $k \in \mathcal{Y}$, and otherwise will pick a class uniformly at random among those in $\mathcal{Y}_i$. Mathematically,

$$f_i(x_k) = \begin{cases} k & \text{if } k \in \mathcal{Y}_i \\ U \sim \text{Uniform}(\mathcal{Y}_i) & \text{if } k \notin \mathcal{Y}_i \end{cases} \tag{1}$$

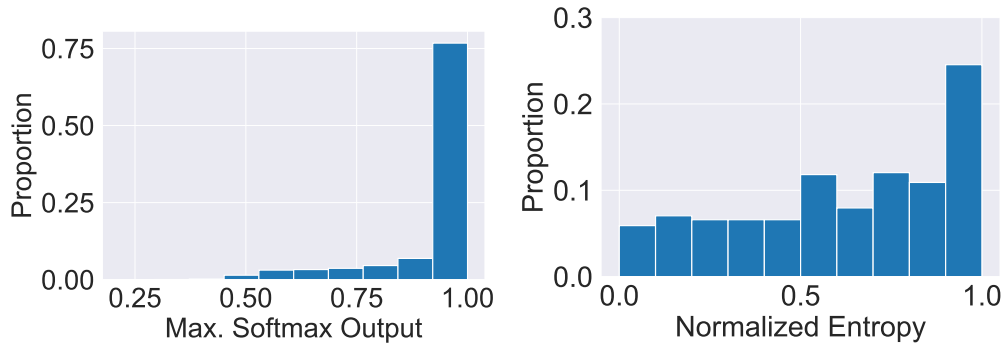
where $U \sim \text{Uniform}(\mathcal{Y}_i)$ denotes a uniform random variable with support $\mathcal{Y}_i$. Note that the output of a classifier in this setting (as well as the earlier perfect accuracy setting) depends only on the true class k corresponding to an input x_k, and does not depend on the input itself. Therefore, we will sometimes write $f(k)$ instead of $f(x_k)$ for notational simplicity.

Given m local classifiers $f_i : \mathcal{X} \to \mathcal{Y}_i$, $i \in [m]$, we assume that the outputs of distinct classifiers [1], denoted by the random vector $Y = (Y_1, \ldots, Y_m) \in \mathcal{Y}_1 \times \mathcal{Y}_2 \times \cdots \times \mathcal{Y}_m$, are conditionally independent given the true class of the input, denoted by Z. This is equivalent to assuming that two distinct classifiers f_1, f_2 have independent noise, i.e., independent sources of randomness U_1, U_2. In this setting, we further assume that Z is chosen uniformly at random from $[K]$, i.e., the prior we impose on the classes is given by $\pi(k) = \frac{1}{K}, \forall k \in [K]$. Let $P_e \triangleq \mathbb{P}(g(Y) \neq Z)$ denote the average probability of error, where $g(Y)$ is the decoder's estimate of Z based on Y. We will now require the K-class classifier to have $P_e \leq \epsilon$ for some fixed $\epsilon \in (0, 1)$. In the sequel, we aim to understand how the decoder can exploit the fact that the true class and the noisy outputs of the classifiers are uniformly distributed and whether this can lead to significant gains with respect to the worst-case setting discussed earlier.

C. Model Justification

While the classification model that we consider may seem stylized, it offers a number of key benefits. First, it is very general, as it makes no assumptions about the underlying classification

[1] We say that classifiers f_1 and f_2 are distinct if $\mathcal{Y}_1 \neq \mathcal{Y}_2$.



(a) (b)

Fig. 1. Histograms of (a) the maximum softmax output of MNIST classifiers when given images belonging to an unfamiliar class, and (b) the normalized entropy of the empirical distribution of predictions produced by classifiers when given images of an unfamiliar class.

algorithm. We consider only "hard" outputs (i.e., final class predictions), rather than "soft" outputs or confidence scores (e.g., outputs of a softmax layer or estimates of conditional class probabilities), as the soft outputs may be incomparable across heterogeneous classifiers.

Figure 1(a) provides further justification for assuming hard outputs. We trained 62 different convolutional neural networks (CNNs) with a softmax layer as the final layer, each trained on a subset of the 10 MNIST digits [20]. In particular, 45 classifiers were each trained on two of the digits, 9 were trained on four digits, 5 were trained on six digits, and 3 were trained on 8 digits (see Section VII for further details). For each classifier, we observed the predictions on $1,000$ images of each *unfamiliar class*, and plotted the resulting histogram of maximum softmax outputs in Figure 1(a). The histogram suggests that in practice, classifiers often give extremely confident predictions of images for which they know nothing about the underlying class, which supports our hard-outputs modeling decision. A related discussion was provided recently in [21], in the context of adversarial machine learning.

In the statistical setting, one reason for assuming that the local classifiers have uniformly distributed noise is that, under the assumption that classifier outputs are conditionally independent given the true class, uniform noise can be viewed as worst-case in a stochastic sense. In Theorem 4, we will see that the lower bound on the minimum number of classifiers needed to achieve $P_e \leq \epsilon$ is maximized by minimizing the mutual information between the true class and the classifier outputs. This is achieved by maximizing the conditional entropy of predictions given the true class, which in turn is achieved by the uniform distribution. While this analysis does not reveal

how the upper bound may be affected by the choice of the noise model, it indicates that uniform noise can make the problem more difficult compared to other random models, assuming that classifier outputs are conditionally independent given the true class.

Empirically, we also observed that classifiers sometimes exhibit behavior similar to uniform noise. Using the same setup as before, for each classifier and each unfamiliar class we generated the empirical distribution, $\hat{\mathcal{P}} = \left(\hat{p}_1, \ldots, \hat{p}_R\right)$ – where R is the size of the classifier – of final class predictions made by the classifier when given images belonging to the unfamiliar class. We then computed the normalized entropy

$$\frac{1}{\log(R)} \cdot H(\hat{\mathcal{P}}) = -\frac{1}{\log(R)} \sum_{i=1}^{R} \hat{p}_i \log \hat{p}_i.$$

Figure 1(b) shows the histogram of normalized entropies that were observed. A normalized entropy close to 1 indicates that the distribution $\hat{P}$ is close to the uniform distribution with the same support. Indeed, the histogram shows that a nontrivial proportion of the MNIST classifiers exhibited behavior similar to the uniform noise in our model.

III. Paper Organization

Throughout this paper, we answer the following questions. Section IV addresses questions (1) and (2), Section V addresses question (3), and Section VI addresses question (4).

(1) What conditions do classification schemes need to satisfy for perfect (zero-error) K-class classification accuracy to be attainable?

(2) Suppose we can design our own set of m classifiers $f_i : \mathcal{X} \to \mathcal{Y}_i$, under the size constraint $|\mathcal{Y}_i| \leq R$, $\forall i \in [m]$ for some fixed integer R satisfying $2 \leq R \leq K$. What is the minimum number of classifiers, m^* (in terms of K and R), needed to satisfy the conditions from (1)?

(3) Is there an efficient algorithm for designing these classifiers given K and R?

(4) In the statistical setting, how does the scaling of m^* change?

Subsequently, we present experimental results in Section VII, and close with a discussion and potential extensions to our work in Section VIII. All omitted proofs are given in the Appendix.

IV. Perfect Accuracy Setting

We begin by answering question (1) above. Here, the goal is to determine the conditions under which exact recovery of the true class corresponding to an input is possible, based solely on observations of the classifier outputs and *a priori* knowledge of $\mathcal{Y}_1, \ldots \mathcal{Y}_m$.

A. Necessary and Sufficient Condition

First, we define the notion of distinguishability between two classes.

Definition 2 (Distinguishability)**.** *We say that two classes $k, k' \in [K]$, $k \neq k'$ are distinguishable with respect to a set of classifiers $\mathcal{F}_{m,K}$ if their output sets are disjoint, i.e., $\mathcal{S}_k \cap \mathcal{S}_{k'} = \emptyset$.*

From a communications perspective, the vectors in a class's output set can be interpreted as the possible channel outputs that the decoder may observe after the class is transmitted over the m noisy channels. Thus, a natural claim is that perfect accuracy is achievable when the output sets of the K classes do not overlap with each other, i.e., each pair of classes is distinguishable. If a set of classifiers $\mathcal{F}_{m,K}$ satisfies these conditions, we will say that it achieves *pairwise distinguishability*. Theorem 1 below says that pairwise distinguishability is a necessary and sufficient condition for achieving perfect accuracy under our classifier model.

Theorem 1. *Given a set of m classifiers $\mathcal{F}_{m,K} = \{f_i : \mathcal{X} \to \mathcal{Y}_i, i \in [m]\}$ with $\mathcal{Y}_i \subseteq [K]$ and $2 \leq |\mathcal{Y}_i| \leq K$ for all $i \in [m]$, perfect accuracy for the K-class classification problem is achievable if and only if all pairs of classes are distinguishable with respect to $\mathcal{F}_{m,K}$.*

Proof. First, suppose that all pairs of classes are distinguishable, and consider the following decoding function, g, for predicting the class of a particular input given the m observed classifier outputs. Initially, we generate a lookup table consisting of $\mathcal{S}_1, \ldots, \mathcal{S}_K$. Note that this can be done using only $\mathcal{Y}_1, \ldots, \mathcal{Y}_m$, which we assume are known *a priori*. Given an input $x \in \mathcal{X}$, we observe the classifier outputs $y = (f_1(x), f_2(x), \ldots, f_m(x))$. We then simply find any $\mathcal{S}_{\hat{k}}$ such that $y \in \mathcal{S}_{\hat{k}}$, and declare $\hat{k}$ as the prediction, i.e., set $g(y) = \hat{k}$. If $k \in [K]$ is the true class, then it must be the case that $y \in \mathcal{S}_k$, by definition of the output sets. Moreover, by the assumption of pairwise distinguishability, $\mathcal{S}_k$ must be the *only* output set containing y. Thus, $g(y) = k$.

Now, suppose without loss of generality that classes $1, 2 \in [K]$ are not distinguishable with respect to $\mathcal{F}_{m,K}$. This means $\exists\, y = (y_1, \ldots, y_m)$ such that $y \in \mathcal{S}_1 \cap \mathcal{S}_2$. For any decoder g, note that $g(y) \neq 1$ or $g(y) \neq 2$, as $g(y) = 1 \implies g(y) \neq 2$. If $g(y) \neq 1$, then since $y \in \mathcal{S}_1$, the classification scheme $(\mathcal{F}_{m,K}, g)$ fails to achieve perfect accuracy. Similarly, failure occurs if $g(y) \neq 2$, since $y \in \mathcal{S}_2$.

□

Theorem 1 says that as long as there is zero ambiguity in the outputs that can result from

each of the K classes, i.e., $\mathcal{F}_{m,K}$ gives an injection from classes to classifier outputs, then the decoder can always determine the correct class. However, this result does not address whether pairwise distinguishability can be achieved by some set of classifiers $\mathcal{F}_{m,K}$. The lemma below gives a condition that is equivalent to distinguishability.

Lemma 1. *Two classes $k, k' \in [K]$, $k \neq k'$ are distinguishable with respect to $\mathcal{F}_{m,K}$ if and only if there exists a classifier $f \in \mathcal{F}_{m,K}$, $f : \mathcal{X} \to \mathcal{Y}$, such that $k, k' \in \mathcal{Y}$.*

Proof. First, suppose there exists a classifier $f_i : \mathcal{X} \to \mathcal{Y}_i$ such that $k, k' \in \mathcal{Y}_i$. Then every $y = (y_1, \ldots, y_m) \in \mathcal{S}_k$ satisfies $y_i = k$, and every $y' = (y'_1, \ldots, y'_m) \in \mathcal{S}_{k'}$ satisfies $y'_i = k'$. It follows that $\mathcal{S}_k \cap \mathcal{S}_{k'} = \emptyset$, so k and k' are distinguishable.

Now suppose there is no classifier $f : \mathcal{X} \to \mathcal{Y}$ such that $k, k' \in \mathcal{Y}$. This means that for every classifier $f : \mathcal{X} \to \mathcal{Y}$, one of three things can happen: 1) $k, k' \notin \mathcal{Y}$, 2) $k \in \mathcal{Y}$ and $k' \notin \mathcal{Y}$, or 3) $k \notin \mathcal{Y}$ and $k' \in \mathcal{Y}$. Consider the i^{th} classifier, $f_i : \mathcal{X} \to \mathcal{Y}_i$, and suppose case 1 holds. Then set $y_i = \tilde{k}$ for some arbitrary $\tilde{k} \in \mathcal{Y}_i$. For cases 2 and 3, assume without loss of generality that $k \in \mathcal{Y}_i$ and $k' \notin \mathcal{Y}_i$. In this case, set $y_i = k$. The resulting vector of outputs $y = (y_1, \ldots, y_m)$ satisfies $y \in \mathcal{S}_k \cap \mathcal{S}_{k'}$, so k and k' are not distinguishable.

□

An immediate consequence of Theorem 1 and Lemma 1 is the following corollary.

Corollary 1. *Perfect accuracy under a set of classifiers $\mathcal{F}_{m,K}$ is achievable if and only if for every pair of classes $k, k' \in [K]$, $k \neq k'$, there exists a classifier $f \in \mathcal{F}_{m,K}$, $f : \mathcal{X} \to \mathcal{Y}$ such that $k, k' \in \mathcal{Y}$.*

To achieve perfect accuracy under worst-case noise, it therefore suffices to have the local classifiers, in aggregate, cover all possible pairwise connections between classes. If this is the case, we will say that $\mathcal{F}_{m,K}$ satisfies the *covering condition*. The question of how to algorithmically generate configurations $\mathcal{F}_{m,K}$ satisfying the covering condition will be addressed in Section V, where we discuss connections to the set cover and clique cover problems. Corollary 1 also makes it clear that when given the choice to design a classifier of size at most R for some $R \in \{2, 3, \ldots, K\}$, one should always choose the maximum possible size, R, as this can only help us get closer to achieving perfect accuracy.

B. Decoding Schemes

We now discuss decoding schemes for predicting the class of an input $x \in \mathcal{X}$ given the m observed classifier outputs $y = (f_1(x), f_2(x), \ldots, f_m(x))$. In the proof of Theorem 1, we considered a naïve approach which requires generating a lookup table consisting of all classes' output sets $\mathcal{S}_1, \ldots, \mathcal{S}_K$. Given the observation y, the decoder finds any output set $\mathcal{S}_{\hat{k}}$ such that $y \in \mathcal{S}_{\hat{k}}$, and declares $\hat{k}$ as the prediction.

When the covering condition from Corollary 1 is satisfied, we can use a more efficient decoding scheme with complexity $\mathcal{O}(mK)$, which works as follows [2]. The main observation behind this scheme is that, instead of storing the entire lookup table, the decoder can store what we call the *authority classifiers*, $\mathcal{C}_1, \ldots, \mathcal{C}_K$, defined for each class $k \in [K]$ as the set of classifiers which were trained on class k:

$$\mathcal{C}_k = \{i \in [m] \,:\, k \in \mathcal{Y}_i\}, \quad k \in [K]. \tag{2}$$

We can think of $\mathcal{C}_k$ as the indices of classifiers whose predictions we trust with respect to inputs belonging to class k. For each $k \in [K]$, the decoder counts the number of votes received from k's authority classifiers:

$$N_k = |\{i \in C_k \,:\, y_i = k\}|. \tag{3}$$

Finally, it predicts the class which received the largest normalized number of votes from its authority classifiers:

$$g(y) = \operatorname*{argmax}_{k \in [K]} \frac{N_k}{|C_k|}. \tag{4}$$

Suppose $\mathcal{F}_{m,K}$ satisfies the covering condition. If the true class of the input is k, then for any $y \in \mathcal{S}_k$, it must be the case that $N_k/|C_k| = 1$, as all of k's authority classifiers will correctly output k. On the other hand, for any $k' \neq k$, by assumption there must exist a classifier $f : \mathcal{X} \to \mathcal{Y}$ such that $k, k' \in \mathcal{Y}$. Note that f is an authority classifier for both k and k', but will output only k. We will therefore observe $N_{k'}/|C_{k'}| < 1$, and hence the decoding scheme will correctly predict k.

C. Binary Matrix Representation

We now start to address question (2) from Section III. We find that there is a one-to-one correspondence between classifier configurations $\mathcal{F}_{m,K}$ and binary matrices with row weight at least 2. This abstraction will somewhat simplify our analysis in Section IV-D.

[2]We note that this decoding scheme can be used even when the covering condition is not satisfied. The covering condition just ensures that perfect accuracy is achieved under our model.

Definition 3 (Classification Matrix)**.** *The classification matrix A corresponding to a set of classifiers $\mathcal{F}_{m,K} = \{f_i : \mathcal{X} \to \mathcal{Y}_i, i \in [m]\}$ is an $m \times K$ binary matrix with*

$$A_{ij} = \begin{cases} 1 & \text{if } j \in \mathcal{Y}_i \\ 0 & \text{otherwise} \end{cases}$$

i.e., $A_{ij} = 1$ if and only if the i^{th} classifier was trained on the j^{th} class. Conversely, a binary matrix $A \in \{0,1\}^{m \times K}$ with row weight at least 2 uniquely defines a set of classifiers $\mathcal{F}_{m,K}$ as follows: the i^{th} row of A defines a classifier $f_i : \mathcal{X} \to \mathcal{Y}_i$ with $\mathcal{Y}_i = \{j \in [K] \,:\, A_{ij} = 1\}$.

The following lemma provides a bridge between the classification matrix and the results of Section IV-A.

Lemma 2. *For a pair of classes $k, k' \in [K], k \neq k'$, there exists a classifier $f \in \mathcal{F}_{m,K}, f : \mathcal{X} \to \mathcal{Y}$ such that $k, k' \in \mathcal{Y}$ if and only if there exists an $i \in [m]$ such that $A_{ik} = A_{ik'} = 1$, where $A \in \{0,1\}^{m \times K}$ is the classification matrix corresponding to $\mathcal{F}_{m,K}$.*

Proof. This follows by construction of the classfication matrix A. □

Lemma 3. *Perfect accuracy is achievable under a set of classifiers $\mathcal{F}_{m,K}$ with corresponding classification matrix $A \in \{0,1\}^{m \times K}$ if and only if for any $k, k' \in [K]$ with $k \neq k'$, there exists an $i \in [m]$ such that $A_{ik} = A_{ik'} = 1$. In this case, we say that A is fully distinguishing.*

Proof. Combine Corollary 1 and Lemma 2. □

This theorem reformulates our exact-recovery problem as the somewhat more concrete problem of designing and analyzing the binary classification matrix. To answer question (2) from Section III, we now impose the size constraint $|\mathcal{Y}_i| \leq R, \forall i \in [m]$ for some fixed integer R satisfying $2 \leq R \leq K$. In light of Lemma 3, we want to understand how the minimum number of rows, m, required to create a fully distinguishing classification matrix scales with the number of columns, K, and maximum row weight, R.

D. Bounds on the Number of Local Classifiers

We now give a lower bound, $m^* \geq \Omega(K^2/R^2)$, on the minimum number of rows in a fully distinguishing classification matrix.

Theorem 2. *For any integer $K \geq 2$, an $m \times K$ classification matrix that is fully distinguishing with maximum row weight $R \in \{2, 3, \ldots, K\}$ must satisfy*

$$m \geq \left\lceil \frac{K(K-1)}{R(R-1)} \right\rceil.$$

Proof. For a classification matrix to be fully distinguishing, it needs to satisfy $\binom{K}{2} = K(K-1)/2$ constraints, namely that every pair of columns needs to share a 1 in some row. However, each row that we add to the matrix can satisfy at most $\binom{R}{2} = R(R-1)/2$ such constraints, as the maximum weight of each row is R. □

The lower bound above is tight at the extreme values of R. When $R = 2$, it is straightforward to show that perfect accuracy is achievable under our model with $m = K(K-1)/2$ classifiers, i.e., using all $\binom{K}{2}$ possible binary classifiers, then performing a majority vote to make predictions. This is the same as the well known one-vs.-one strategy which decomposes multiclass problems into pairwise binary problems. If $R = K$, then perfect accuracy is trivially achieved using just a single centralized classifier.

The following achievability result yields an upper bound of $m^* \leq \mathcal{O}\left(\frac{K^2}{R^2} \log K\right)$ by using a probabilistic argument, where each classifier is drawn independently and uniformly at random from the set of size-R classifiers. [3] Compared to the lower bound in Theorem 2, there is a $\log K$ gap in terms of scaling.

Theorem 3. *For all integers $K \geq 2$, there exists an $m \times K$ binary matrix with maximum row weight $R \in \{2, 3, \ldots, K\}$ that is fully distinguishing with*

$$m = \left\lceil \frac{K(K-1)}{R(R-1)} \log\left(\frac{K(K-1)}{2}\right) + 1 \right\rceil.$$

Resolving the gap between the bounds in Theorem 2 and Theorem 3 is an open problem. We conjecture that the upper bound in Theorem 3 gives sub-optimal scaling, and that the $\log K$ factor can either be reduced to something like $\log R$ or be eliminated altogether. In many combinatorial problems, such $\log$ factors are extraneous and reflect inefficiencies in the sampling procedure. Likewise, Theorem 3 is proved by (inefficiently) sampling classifiers uniformly at random with replacement. In the next section, we give greedy algorithms for selecting a minimal set of local classifiers which form a fully distinguishing classification matrix, i.e., satisfy the covering

[3] Throughout, $\log$ denotes the natural (base e) logarithm.

condition. As we will see, one of these algorithms produces a set of local classifiers which is guaranteed to be within a factor of $O(\log R)$ to optimality.

V. Algorithms via Set Covering and Clique Covering

The proof of Theorem 3 relied on a random configuration of local classifiers to prove the existence of a configuration which satisfies the covering condition. In practice, however, it would be useful to have an explicit algorithm for generating sets of local classifiers which satisfy this condition.

A. Set Covering

Interestingly, the problem of achieving perfect accuracy is a special case of the well-known *set cover problem* from the combinatorics and theoretical computer science literature, originally introduced and shown to be NP-complete in 1972 [22]. The set cover problem consists of

1) the *universe*: a set of elements $\mathcal{U} = \{1, 2, \ldots, n\}$, and
2) a collection of sets $\mathcal{S}$ whose union equals the universe, i.e., $\bigcup_{S \in \mathcal{S}} S = \mathcal{U}$.

The goal is to identify the smallest sub-collection of $\mathcal{S}$ whose union equals the universe. Our classification problem can be reformulated in these terms by setting $\mathcal{U}$ to be the set of all $n = \binom{K}{2}$ pairwise connections between classifiers and setting $\mathcal{S}$ to be the set of all $\binom{K}{R}$ possible sets of $\binom{R}{2}$ pairwise connections that can be made in a single row of the classification matrix. Designing the classification matrix to have as few rows as possible while satisfying the conditions for perfect accuracy is equivalent to finding the smallest sub-collection of $\mathcal{S}$ whose union equals $\mathcal{U}$.

The following greedy algorithm [23] gives a polynomial time (in $n|\mathcal{S}|$) approximation of set covering. In each iteration, the algorithm simply chooses the set in $\mathcal{S}$ that contains the largest number of yet-uncovered elements of $\mathcal{U}$, and adds this set to the partial cover. This step is repeated until the union of the selected subsets covers $\mathcal{U}$. For example, if $K = 5$, $R = 3$ in the classification problem, then the algorithm provides a set covering defined by $\mathcal{Y}_1 = \{1, 2, 3\}, \mathcal{Y}_2 = \{1, 4, 5\}, \mathcal{Y}_3 = \{2, 3, 4\}, \mathcal{Y}_4 = \{2, 3, 5\}$. This set of classifiers satisfies the covering condition from Corollary 1, and thus admits perfect accuracy under our model.

This algorithm identifies a set covering that is at most $\tilde{H}(n)$ times as large as the optimal covering, where $\tilde{H}(n)$ is the n^{th} harmonic number given by

$$\tilde{H}(n) = \sum_{i=1}^{n} \frac{1}{i} \leq \log n + 1.$$

In fact, if $|S| \leq \rho$, $\forall S \in \mathcal{S}$, then the ratio is improved to $\tilde{H}(\rho)$ [24], [25]. In the classification problem, we have $\rho = \binom{R}{2}$, so the greedy algorithm finds a perfect-accuracy classifier configuration with at most $\tilde{H}(\binom{R}{2}) \approx 2 \log R$ times as many classifiers as the optimal configuration. If R is small (which is the expected regime in distributed and federated learning applications), then the algorithm is nearly optimal. Numerically, we also found that the sizes of the covers produced by the algorithm nearly match the lower bound in Theorem 2 in the small R regime.

B. Clique Covering

We note that the greedy algorithm runs in polynomial time *in the parameters of the set cover problem*, n and $|\mathcal{S}|$, but that this translates to exponential time in K and R. However, more efficient algorithms present themselves when we rephrase our problem in graph-theoretic terms. Consider an undirected graph $G = (V, E)$ with vertex set $V = \{1, \ldots, K\}$ and edge set E where for every $k, k' \in [K]$, $k \neq k'$, we have $(k, k') \in E$ if and only if $k, k' \in \mathcal{Y}_i$ for some $i \in [m]$. Thus, each classifier creates a *clique* on the graph. Moreover, a configuration of classifiers can achieve perfect accuracy if and only if their induced graph, G, is the *complete graph* [4] on K vertices. Our problem can be equivalently phrased as: What is the minimum number of cliques of size R needed to cover the edges of the complete graph? This is a special case of the *k-clique covering problem*, which was shown in [26] to be NP-complete in the general case. It is readily seen that clique covering is in turn a special case of set covering. This connection was studied in [27], and approximation algorithms which give better worst-case running times than greedy set covering – at the expense of approximation ratio – were provided. For simplicity, we use the standard greedy set covering algorithm in our experiments in Section VII, but one can alternatively use the more efficient algorithms from [27].

C. Variations on Greedy Set Covering for Federated Learning

By setting appropriate inputs to the greedy set cover algorithm, we can handle two types of situations that may arise in federated learning. Consider a setting with m clients, where the i^{th} client trains a local classifier $f_i : \mathcal{X} \to \mathcal{Y}_i$ and communicates $\mathcal{Y}_i$ to the central server. First, suppose the cliques corresponding to the initial collection $\mathcal{Y}_1, \ldots, \mathcal{Y}_m$ fail to cover the edges of

[4]While the standard definition of a complete graph requires all pairs of vertices to be connected by a unique edge, here we allow for redundant edges and just consider them to be a single edge.

the complete graph on K vertices, and that the server is interested in determining the smallest number of *additional* local classifiers (cliques) that need to be trained to obtain a complete cover. In this case, we can set the universe $\mathcal{U}$ to be the set of uncovered edges, and set $\mathcal{S}$ to be the cliques that have not yet been used. By running the standard greedy algorithm on this problem instance, we obtain a (close to) minimal set of additional cliques needed to form a complete cover. In practice, the central server may leverage this information to seek out clients with the desired missing data.

A second situation that may be encountered is one where there are *more local classifiers than are needed* to cover the graph. This is likely to be the case in large-scale FL settings with millions of participating clients. To minimize communication costs, the server may be interested in selecting a minimal subset of these classifiers which still covers the graph, and asking only the corresponding clients to send their classifiers. Here, we can set $\mathcal{S}$ equal to the cliques available in the pool of users, and keep $\mathcal{U}$ as the set of edges in the complete graph on K vertices. The greedy algorithm will then return a (close to) minimal subset of $\mathcal{S}$ which still covers the complete graph.

VI. Statistical Setting

The perfect accuracy setting from Section IV was combinatorial in nature and gave worst-case bounds on the number of local classifiers needed to achieve exact recovery of the true class. We now investigate the scaling of m^* in a more average-case sense and under less adversarial noise conditions, as described in Section II-B. In practice, classifiers are likely to lie somewhere between these two noise models. In the statistical setting, we are particularly interested in whether one can still achieve a high classification accuracy with fewer classifiers than in the entire set cover. This problem is relevant, for example, when it is difficult to ensure that all of the set-covering classifiers are available.

A. Lower Bound

We first give an information-theoretic lower bound on m^* using Fano's inequality. Our proof will rely on the following lemma, which gives an expression for the conditional entropy of the classifier outputs, Y, given the true class, Z.

Lemma 4. *Under the assumptions of Section II-B, the conditional entropy of* $Y = (Y_1, \ldots, Y_m)$ *given* Z *is*

$$H(Y \mid Z) = m \cdot \frac{(K-R)}{K} \cdot \log R.$$

Theorem 4. *Any* K*-class classification scheme using* m *smaller classifiers of size* R *that achieves* $P_e \leq \epsilon$ *under the assumptions of Section II-B must satisfy*

$$m \geq \left\lceil \frac{K}{R} \cdot \left(\frac{(1-\epsilon)\log K - \log(2)}{\log R} \right) \right\rceil.$$

B. Upper Bound

To prove an upper bound on the number of classifiers required to achieve an ϵ-probability of error, we consider the same probabilistic construction as in Theorem 3, coupled with maximum likelihood (ML) decoding. For a particular output vector $y = (y_1, \ldots, y_m)$, the decoder predicts the class according to the decision rule

$$g(y) = \operatorname*{argmax}_{k \in [K]} \mathcal{L}(y; k)$$

where

$$\mathcal{L}(y; k) = \prod_{i=1}^{m} \mathbb{P}(Y_i = y_i \mid Z = k)$$

and with ties broken arbitrarily.

Theorem 5. *Under the assumptions of Section II-B, the previously described classifier construction and decoding rule achieve a probability of error bounded by an arbitrary* $\epsilon \in (0, 1)$ *using*

$$m = \left\lceil \frac{K(K-1)}{(K-R)(R-1)} \cdot \log\left(\frac{K}{\epsilon}\right) \right\rceil$$

classifiers of size R*.*

For fixed ϵ, the above result gives $m^* \leq \mathcal{O}((K/R)\log K)$ when R is sufficiently smaller than K, which is only a factor of $\log R$ larger than our bound in Theorem 4 in terms of scaling. When $R = \mathcal{O}(1)$, then the upper and lower bounds meet, yielding $m^* = \Theta(K \log K)$. On the other hand, when $R = K^\alpha$ for some $\alpha \in (0, 1)$, then the lower bound scales as $\Omega(K)$, whereas the upper bound scales as $\mathcal{O}(K \log K)$.

When combined, Theorems 4 and 5 reveal that relaxing the criteria for perfect accuracy yields a reduction in the minimum number of required classifiers from roughly $\Theta(K^2/R^2)$ to $\Theta(K/R)$. Note that $\lceil K/R \rceil$ is the minimum number of size-R cliques needed to cover K vertices. Therefore,

under the graph-theoretic interpretation given in Section V-B, the problem now roughly reduces to covering the vertices rather than the edges of the complete graph on K vertices.

VII. Experiments

We present experimental results on the performance of our scheme in practice. For different classifier sizes R, we used the greedy set cover algorithm to design configuration of smaller classifiers. For example, for $K = 10$ and $R = 4$, we trained 9 smaller classifiers, each given access to all training examples corresponding to its own 4 classes. We used the decoding scheme given in Equation (4), Section IV-B. We examined the performance of this scheme on the MNIST [20] and CIFAR-10 [28] datasets, comparing the resulting classification accuracy for $R = 4, 6, 8$ to that of one-versus-one ($R = 2$) and fully centralized classification ($R = 10$). All implementations were done with Keras [29], and our experiments on *FederatedAveraging* [1] additionally utilized the TensorFlow Federated framework [5]. All hyperparameters were set to their default values.

A. Set Covering-Based Classification

1) MNIST: For the MNIST handwritten digit dataset, we used a simple convolutional neural network (CNN) architecture [6]. The same architecture was used for all values of R, i.e., all smaller classifiers as well as the centralized classifier ($R = 10$) were trained with the same architecture, except with different dimensions in the final softmax layer. The batch size was set to 128, and training was done over 12 epochs per classifier. Table I shows the resulting training and testing accuracies. We see that our scheme performs nearly as well as the centralized classifier.

2) CIFAR-10: We next applied our scheme to the CIFAR-10 dataset. We used the ResNet20 v1 [7] [30] architecture for each classifier, with a batch size of 32 and 200 epochs. Table II shows the final training and testing accuracies, demonstrating again the favorable performance of our scheme.

B. Comparison to FederatedAveraging

To assess whether our scheme can effectively address the problem of data heterogeneity in federated learning settings, we compared our approach to the *FederatedAveraging* (or *FedAvg*)

[5] https://www.tensorflow.org/federated.

[6] More details can be found at https://keras.io/examples/mnist_cnn/.

[7] More details can be found at https://keras.io/examples/cifar10_resnet/.

TABLE I

MNIST ACCURACIES OBTAINED BY COMBINING (USING THE DECODING SCHEME FROM EQUATION (4)) LOCAL CLASSIFIERS PRODUCED BY THE GREEDY SET COVER ALGORITHM .

$R =$	2	4	6	8	10
TRAIN (%)	99.71	99.82	99.74	99.78	99.12
TEST (%)	98.98	99.03	98.97	99.07	99.26

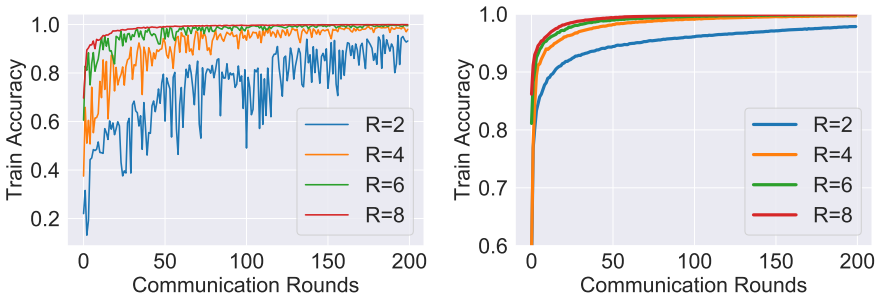
TABLE II

CIFAR-10 ACCURACIES OBTAINED BY COMBINING (USING THE DECODING SCHEME FROM EQUATION (4)) LOCAL CLASSIFIERS PRODUCED BY THE GREEDY SET COVER ALGORITHM .

$R =$	2	4	6	8	10
TRAIN (%)	99.61	99.49	99.55	99.16	98.48
TEST (%)	88.36	90.82	91.95	91.99	91.98

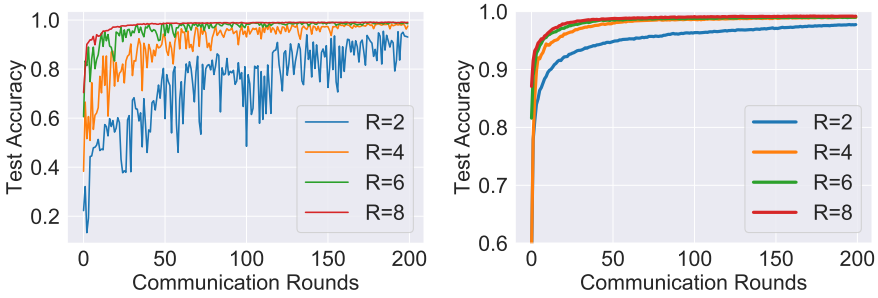
algorithm on non-i.i.d. partitions of the MNIST and CIFAR-10 training data. For each value of $R \in \{2, 4, 6, 8\}$, we partitioned the training data according to the corresponding set cover produced by the greedy algorithm. For example, $R = 2$ corresponds to 45 total clients, each with the training data for two classes. In each communication round, each client performed 1 epoch over their own data.

1) MNIST: The batch size was set to 128 and the number of communication rounds was 200. The model architecture was the same as in Section VII-A1. We first examined the performance of *FedAvg* when two clients are selected uniformly at random to participate in each communication round. The resulting train and test accuracy curves are plotted in Figure 2 (a) and (c). Evidently, the learning process is slower and more volatile for smaller values of R (i.e., more clients). After 100 communication rounds, for instance, the test accuracy for $R = 2$ was 48.5%, and the final test accuracy after 200 rounds for $R = 2$ was 93.1%. In Figure 2 (b) and (d), we show what happens when all clients participate in each communication round. The learning curves are much smoother in this case, although the training accuracy for $R = 2$ after all 200 communication rounds is strictly worse (97.8%) than that of our approach (99.7%).



(a) 2 clients per round

(b) Full client participation



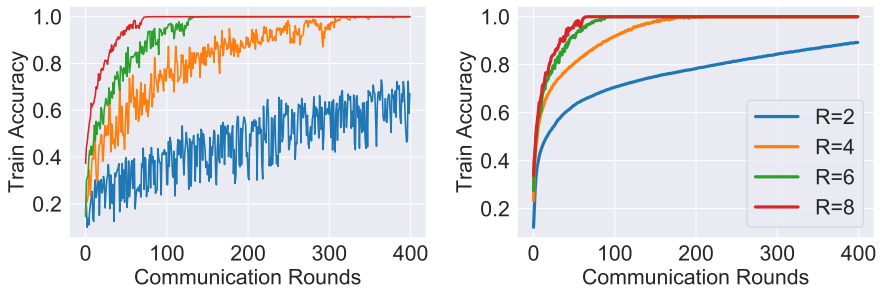
(c) 2 clients per round

(d) Full client participation

Fig. 2. *FedAvg* convergence on non-i.i.d. MNIST data partitions. (a) and (c): 2 clients are selected uniformly at random to participate in each communication round. (b) and (d): All clients participate in each round.

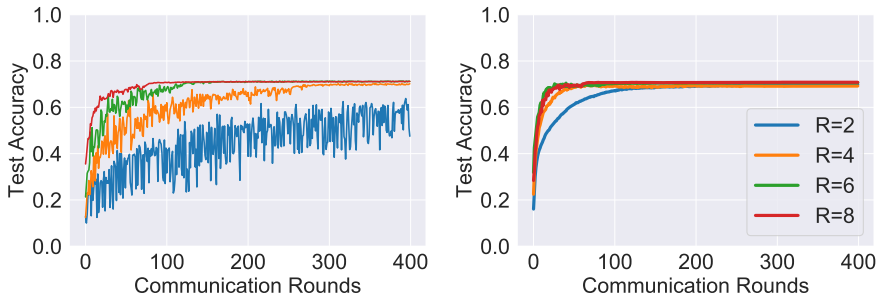
2) CIFAR-10: The batch size was 32 and the number of communication rounds was 400. We used the architecture provided in the TensorFlow tutorial [8]. Figure 3 shows the analogous curves to those in Figure 2. As expected, the higher complexity of the CIFAR-10 dataset compared to MNIST resulted in a slower and more oscillatory learning process. After 200 communication rounds, the train accuracy for $R = 2$ was only 31.52% in the case of 2 clients per round, and 78.37% with full client participation. The final train accuracy after 400 rounds with full participation was 89.24%, in contrast to the final accuracy of one-vs.-one, 99.61%. With 2 clients participating per round, the test accuracy for $R = 2$ after all 400 communication rounds was 47.5% (compared to 67.25% train accuracy), and the maximum accuracy over all values of R was 71.1% (compared to 100% train accuracy). Full client participation improved the speed of convergence, but did not improve the resulting accuracy: The maximum final test accuracy over all values of R was 70.8% in this case. In comparison, our scheme achieved a test accuracy of

[8] https://www.tensorflow.org/tutorials/images/cnn.



(a) 2 clients per round

(b) Full client participation



(c) 2 clients per round

(d) Full client participation

Fig. 3. *FedAvg* convergence on non-i.i.d. CIFAR-10 data partitions. (a) and (c): 2 clients are selected uniformly at random to participate in each communication round. (b) and (d): All clients participate in each round.

88.36% for $R = 2$, and a maximum accuracy of 91.37% over all values of $R \in \{2, 4, 6, 8\}$. It is clear that *FedAvg* is highly sensitive to both the heterogeneity of the data and the amount of client participation per round, whereas our approach enjoys greater robustness to heterogeneity and does not rely on the consistent availability of clients over such long time horizons.

C. Robustness to Missing Classifiers

We tested the robustness of our scheme under incomplete sets of classifiers using CIFAR-10. For each value $R \in \{2, 4, 6, 8\}$, we initially generated a set of classifiers of size R using the greedy set cover algorithm described in Section V. These were the same classifiers used in Section VII-A2. We considered the effect on the training and testing accuracies of removing *(a)* a single classifier, and *(b)* two classifiers from the set cover. For *(a)*, we determined the resulting accuracies after removing each classifier, and then computed the average over all such removals. For *(b)*, we considered all $\binom{m^*}{2}$ removals of two classifiers, where m^* is the size of the set cover.

The average accuracies for each value of R and their corresponding standard deviations are

TABLE III

AVERAGE CIFAR-10 ACCURACIES UNDER CLASSIFIER REMOVALS FROM THE SET COVER. STANDARD DEVIATION IS DENOTED BY σ.

Number of removed classifiers = 1				
$R =$	2	4	6	8
Train (%)	**98.63** ($\sigma = 1.62\%$)	**94.14** ($\sigma = 3.06\%$)	**96.30** ($\sigma = 2.86\%$)	**95.54** ($\sigma = 4.03\%$)
Test (%)	**87.51** ($\sigma = 1.48\%$)	**86.11** ($\sigma = 2.80\%$)	**88.78** ($\sigma = 2.64\%$)	**88.28** ($\sigma = 3.80\%$)
Number of removed classifiers = 2				
$R =$	2	4	6	8
Train (%)	**97.58** ($\sigma = 2.32\%$)	**86.54** ($\sigma = 5.20\%$)	**88.38** ($\sigma = 4.77\%$)	**79.26** ($\sigma = 0.25\%$)
Test (%)	**86.62** ($\sigma = 2.11\%$)	**79.28** ($\sigma = 4.80\%$)	**81.41** ($\sigma = 4.42\%$)	**73.73** ($\sigma = 0.83\%$)

shown in Table III. Surprisingly, the performance under single classifier removals remained quite similar to the values in Table II. The values for $R = 2$ remained closest to those from the complete set cover, and also had the smallest variance among all values of R. This is likely due to the fact that smaller values of R correspond to more classifiers in the set cover, each responsible for fewer classes. Hence, the removal of a single classifier when R is small is likely to have a less detrimental effect on performance. Overall, despite the drop in train and test accuracies, our scheme still performs much better than *FedAvg*.

D. Random Classifier Configurations

A related question to that addressed in VII-C is whether a random classifier construction performs well. In practice, it may not be possible to control the distribution of data across clients or worker nodes; randomly chosen classifiers can serve as a model for such situations. For each $R \in \{4, 6, 8\}$, we sampled uniformly without replacement m times from the set of $\binom{K}{R}$ possible classifiers of size R, for $m \in \{m^*, (m^* + 1), (m^* + 2), (m^* + 3)\}$, where m^* is the number of classifiers specified by the greedy set cover algorithm. That is, we generated slightly more classifiers than those in the set cover. Note that a random set of classifiers, even of cardinality larger than m^*, may fail to comprise a complete set cover.

TABLE IV

CIFAR-10 ACCURACIES WITH RANDOM CLASSIFIER CONFIGURATIONS. THE NUMBER OF RANDOMLY CHOSEN CLASSIFIERS IS DENOTED BY m.

$\mathbf{R = 4}$				
$m =$	9	10	11	12
Train (%)	91.66	92.15	92.44	94.28
Test (%)	84.33	84.98	85.47	87.22

$\mathbf{R = 6}$				
$m =$	5	6	7	8
Train (%)	93.97	98.78	98.87	99.74
Test (%)	86.70	91.60	91.82	92.76

$\mathbf{R = 8}$				
$m =$	3	4	5	6
Train (%)	95.71	99.48	99.75	99.75
Test (%)	88.88	92.60	93.24	93.37

Each randomly chosen classifier was trained on CIFAR-10 with the same architecture and hyperparameters as in Section VII-A2. Table IV gives the resulting training and testing accuracies of our scheme (the values shown are averages over two trials). Overall, we observe that a set of exactly m^* random classifiers tends to yield (except in the case of $R = 8$) lower accuracies than those attained by the complete set cover. For example, with $R = 4$ and $m = m^* = 9$, the random construction achieved train and test accuracies of (respectively) 91.66% and 84.33%, whereas the complete set cover achieved respective accuracies of 99.49% and 90.82%. However, with the addition of a few more classifiers, the accuracy can increase dramatically and can even surpass the accuracy of the set cover. For instance, with $R = 8$ and $m = 6$, the train and test accuracies of the random construction were 99.75% and 93.37%, respectively, in contrast to 99.16% and 91.99% from the full set cover. We speculate that the random construction's ability to outperform the set cover is due to the presence of redundant connections between classes that may be difficult to distinguish, i.e., training more than one classifier to distinguish between two such classes may constitute a form of natural error correction. In summary, these results suggest that our scheme is robust not only to uncovered connections between classes, but also to arbitrary distributions of the data across clients.

TABLE V

CIFAR-10 ACCURACIES WITH LESS TRAINING DATA AND TRAINING TIME PER CLASSIFIER

$R =$	2	4	6	8
Train (%)	71.75	74.55	78.32	78.23
Test (%)	68.56	71.37	75.64	75.60

E. Performance with Less Data and Less Training Time

In practice, it may also be difficult to ensure that each client or smaller classifier has access to sufficiently many training samples. To study the performance of our scheme under such scenarios, we reduced the number of CIRAR-10 training images per class to 500, compared to 5,000 images per class in the entire training set. We also trained each classifier for 100 epochs, compared to 200 epochs in other experiments in this paper. For each $R \in \{2, 4, 6, 8\}$, we trained the classifiers specified by the greedy set cover algorithm. All other details match Section VII-A2. As shown in Table V, there is a noticeable degradation in both training and testing accuracy compared to Table II. However, our scheme still largely outperforms *FedAvg* with respect to testing accuracy in our experiments (see Section VII-B). Hence, even with only a fraction of the original training data and significantly fewer training epochs than usual – two relevant characteristics for FL-type settings – our approach exhibits favorable performance.

VIII. DISCUSSION

While our experiments used the same neural network architecture for the smaller classifiers and centralized classifier, we believe that simpler architectures could be used for the smaller classifiers without sacrificing accuracy. This would further expedite the training process and reduce communication costs between the clients and server. Our approach potentially also has privacy benefits compared to standard FL, as the models are communicated from the clients to the server only once. In principle, the same methods developed to preserve privacy in FL (e.g., adding noise to transmitted gradients to achieve differential privacy) could be applied to our method as well. However, the effect this would have on the resulting classification accuracy is unclear, and is an interesting direction for future inquiry.

One possible objection to our approach is that it assumes each smaller classifier has enough training samples to achieve a sufficiently high accuracy on its own classes. However, our scheme can be seamlessly integrated with FL to address this issue in the following "hierarchical" manner. Rather than having a single centralized server orchestrate the entire learning process, we could instead have one server in charge of each cluster of clients with similar training data. Each sub-server coordinates the training of a smaller classifier via *FedAvg* or a similar technique, thereby simultaneously augmenting the sample size and avoiding the issue of data heterogeneity by including only similar clients. Once all sub-servers have fully trained their classifiers, they transmit their models to the top-level server which synthesizes the smaller classifiers using our scheme.

Other intriguing future directions include using tools from coding theory to increase the robustness of the classification schemes to poorly trained or byzantine classifiers, and performing experiments on large-scale datasets such as ImageNet. Formulating and studying more realistic classifier models in which the local classifiers have sub-perfect accuracy on examples belonging to their familiar classes is also an interesting future direction. Finally, if the learning algorithms used to produce the local classifiers are known, how might we leverage this information to design better global classification schemes?

ACKNOWLEDGEMENTS

This work was supported in part by a Cisco Systems Stanford Graduate Fellowship, the Center for Science of Information (CSoI), the NSF Science and Technology Center under grant agreement CCF-0939370, and NSF award NeTS #1817205.

REFERENCES

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.

[2] Jakub Konečnỳ, 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.

[3] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[4] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra, "Federated learning with non-IID data," *arXiv preprint arXiv:1806.00582*, 2018.

[5] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang, "On the convergence of FedAvg on non-IID data," in *International Conference on Learning Representations*, 2020.

[6] Hubert Eichner, Tomer Koren, H Brendan McMahan, Nathan Srebro, and Kunal Talwar, "Semi-cyclic stochastic gradient descent," *arXiv preprint arXiv:1904.10120*, 2019.

[7] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4424–4434.

[8] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith, "On the convergence of federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, 2018.

[9] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[10] Lumin Liu, Jun Zhang, SH Song, and Khaled B Letaief, "Edge-assisted hierarchical federated learning with non-IID data," *arXiv preprint arXiv:1905.06641*, 2019.

[11] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek, "Robust and communication-efficient federated learning from non-IID data," *arXiv preprint arXiv:1903.02891*, 2019.

[12] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar, "Personalized federated learning: A meta-learning approach," *arXiv preprint arXiv:2002.07948*, 2020.

[13] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh, "Three approaches for personalization with applications to federated learning," 2020.

[14] Jerome Friedman, "Another approach to polychotomous classification," 1996.

[15] Trevor Hastie and Robert Tibshirani, "Classification by pairwise coupling," in *Advances in neural information processing systems*, 1998, pp. 507–513.

[16] Christopher M Bishop, *Pattern recognition and machine learning*, Springer Science+ Business Media, 2006.

[17] Ryan Rifkin and Aldebaro Klautau, "In defense of one-vs-all classification," *Journal of machine learning research*, vol. 5, no. Jan, pp. 101–141, 2004.

[18] Thomas G Dietterich and Ghulum Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of artificial intelligence research*, vol. 2, pp. 263–286, 1994.

[19] Erin L Allwein, Robert E Schapire, and Yoram Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers," *Journal of machine learning research*, vol. 1, no. Dec, pp. 113–141, 2000.

[20] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[21] Gunjan Verma and Ananthram Swami, "Error correcting output codes improve probability estimation and adversarial robustness of deep neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 8643–8653.

[22] Richard M Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*, pp. 85–103. Springer, 1972.

[23] Vasek Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of operations research*, vol. 4, no. 3, pp. 233–235, 1979.

[24] David S Johnson, "Approximation algorithms for combinatorial problems," *Journal of computer and system sciences*, vol. 9, no. 3, pp. 256–278, 1974.

[25] László Lovász, "On the ratio of optimal integral and fractional covers," *Discrete mathematics*, vol. 13, no. 4, pp. 383–390, 1975.

[26] Ian Holyer, "The NP-completeness of some edge-partition problems," *SIAM Journal on Computing*, vol. 10, no. 4, pp. 713–717, 1981.

[27] Oliver Goldschmidt, Dorit S Hochbaum, Cor Hurkens, and Gang Yu, "Approximation algorithms for the k-clique covering problem," *SIAM Journal on Discrete Mathematics*, vol. 9, no. 3, pp. 492–509, 1996.

[28] Alex Krizhevsky et al., "Learning multiple layers of features from tiny images," Tech. Rep., Citeseer, 2009.

[29] François Chollet et al., "Keras," https://keras.io, 2015.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

APPENDIX

A. *Proof of Theorem 3*

We employ the probabilistic method. Consider a random binary matrix $A \in \{0,1\}^{m\times K}$ with each row chosen independently and uniformly at random from the space of possible weight-R binary vectors of length K. We will choose m such that there is a strictly positive probability that A is a fully distinguishing matrix, thereby proving the existence of such a matrix. Let

$$P_e = \mathbb{P}(A \text{ is not fully distinguishing})$$

and let E_j be the event that two columns of A are different or both equal to 0 in the jth place. By the union bound, P_e can be upper bounded by a sum over all $\binom{K}{2}$ pairs of columns of A as

$$\begin{aligned} P_e &\leq \sum_{i=1}^{\binom{K}{2}} \mathbb{P}(\text{the } i\text{th pair of columns of } A \text{ shares no 1 in the same place}) \\ &= \binom{K}{2} \prod_{j=1}^{m} \mathbb{P}(E_j) \\ &= \binom{K}{2} \mathbb{P}(E_1)^m \\ &= \binom{K}{2} \left(1 - \mathbb{P}(E_1^c)\right)^m \end{aligned}$$

where E_1^c is the event that two columns of A are both equal to 1 in the 1st place. By fixing both columns of A to be equal to 1 in the 1st place, the number of ways to assign the remaining $(R-2)$ ones to the remaining $(K-2)$ spots in the same row is $\binom{K-2}{R-2}$. Therefore,

$$\begin{aligned} P_e &\leq \binom{K}{2} \left(1 - \frac{\binom{K-2}{R-2}}{\binom{K}{R}}\right)^m \\ &= \frac{K(K-1)}{2} \cdot \left(1 - \frac{R(R-1)}{K(K-1)}\right)^m \\ &\leq \frac{K(K-1)}{2} \cdot \exp\left(-m \cdot \frac{R(R-1)}{K(K-1)}\right) \end{aligned}$$

where the inequality follows from the fact that $1 - x \leq e^{-x}$ for all $x \in \mathbb{R}$. To ensure that $P_e < 1$, it suffices to choose

$$m = \left\lceil \frac{K(K-1)}{R(R-1)} \cdot \log\left(\frac{K(K-1)}{2}\right) + 1 \right\rceil.$$

□

B. *Proof of Theorem 4*

We assume without loss of generality that all classifiers are distinct, as the presence of duplicate classifiers does not reduce the probability of error any more than distinct classifiers do.

1) Proof of Lemma 4: We expand the conditional entropy of Y given Z as

$$\begin{aligned} H(Y \mid Z) &= \sum_{k \in [K]} \pi(k) H(Y \mid Z = k) \\ &= \frac{1}{K} \sum_{k \in [K]} H(Y_1, \ldots, Y_m \mid Z = k) \\ &= \frac{1}{K} \sum_{k \in [K]} \sum_{i \in [m]} H(Y_i \mid Z = k) \end{aligned}$$

where the last equality follows from the conditional independence of the classifier predictions given Z. Next, note that for each $k \in [K]$ and $i \in [m]$ we have

$$H(Y_i \mid Z = k) = \begin{cases} \log R & \text{if } k \notin \mathcal{Y}_i \\ 0 & \text{if } k \in \mathcal{Y}_i \end{cases}$$

as Y_i is deterministic if $k \in \mathcal{Y}_i$, and otherwise equals a class chosen uniformly at random from the R classes in $\mathcal{Y}_i$. Let $N = \Big|\{(k,i) \in [K] \times [m] \,:\, k \notin \mathcal{Y}_i\}\Big|$, and note that $N = m(K-R)$ since for each $i \in [m]$ there are precisely $K - R$ classes that are not included in $\mathcal{Y}_i$. Continuing from before,

$$\begin{aligned} H(Y \mid Z) &= \frac{1}{K} \sum_{\substack{k \in [K],\, i \in [m] \\ k \notin \mathcal{Y}_i}} \log R \\ &= \frac{1}{K} \cdot N \cdot \log R \\ &= m \cdot \frac{(K-R)}{K} \cdot \log R. \end{aligned}$$

□

2) Main Proof: If $\mathcal{Y}$ denotes the set of possible values of Y, then the entropy of Y can be bounded as

$$H(Y) \leq \log |\mathcal{Y}| \leq \log(R^m) = m \log R.$$

From Lemma 4, $H(Y \mid Z) = m \cdot \frac{(K-R)}{K} \cdot \log R$. Thus, since the mutual information between Y and Z is given by $I(Y;Z) = H(Y) - H(Y \mid Z)$,

$$\begin{aligned} I(Y;Z) &\leq m \log R - m \cdot \frac{(K-R)}{K} \cdot \log R \\ &= m \cdot \frac{R}{K} \log R. \end{aligned}$$

Observe that $Z \to Y \to g(Y)$ forms a Markov chain. By the data processing inequality,

$$I(Y;Z) \geq I(Z;g(Y)) = H(Z) - H(Z \mid g(Y)).$$

Since Z is uniformly distributed over the K classes by assumption, we have $H(Z) = \log K$. If $H(P_e)$ denotes the binary entropy corresponding to $P_e = \mathbb{P}(g(Y) \neq Z)$, then $H(P_e) \leq \log(2)$. By Fano's inequality, combined with the assumption that $P_e \leq \epsilon$, we therefore have

$$\begin{aligned} H(Z \mid g(Y)) &\leq H(P_e) + P_e \log(K-1) \\ &\leq \log(2) + \epsilon \log K. \end{aligned}$$

Combining all of these inequalities yields

$$\begin{aligned} m \cdot \frac{R}{K} \log R &\geq H(Z) - H(Z \mid g(Y)) \\ &\geq \log K - (\log(2) + \epsilon \log K) \\ &= (1-\epsilon) \log K - \log(2) \end{aligned}$$

and dividing both sides of the inequality by $\frac{R}{K} \log R$ and taking the ceiling gives the final result. □

C. *Proof of Theorem 5*

The probability of error can be bounded as

$$P_e = \frac{1}{K}\sum_{k=1}^{K}\mathbb{P}(g(Y)\neq k\,|\,Z=k) \tag{5}$$

$$= \mathbb{P}(g(Y)\neq 1\,|\,Z=1) \tag{6}$$

$$\leq \mathbb{P}\Big(\exists j\in\{2,\ldots,K\}\;:\;\mathcal{L}(Y;j)\geq\mathcal{L}(Y;1)\,|\,Z=1\Big) \tag{7}$$

$$\leq \sum_{j=2}^{K}\mathbb{P}\Big(\mathcal{L}(Y;j)\geq\mathcal{L}(Y;1)\,|\,Z=1\Big) \tag{8}$$

$$\leq K\cdot\mathbb{P}\Big(\mathcal{L}(Y;2)\geq\mathcal{L}(Y;1)\,|\,Z=1\Big). \tag{9}$$

where (6) follows from the symmetry in the random classifier construction, and (8) uses the union bound. In (9), we again use the symmetry in the construction, and $\mathbb{P}$ represents the randomness in both the construction and in Y.

Let Y' be a random output drawn conditional on $Z=1$. Consider the event $E:\mathcal{L}(Y';2)\geq\mathcal{L}(Y';1)$, and further define the events $B_i:1\notin\mathcal{Y}_i,2\in\mathcal{Y}_i$ and $C_i:Y'_i=2$ for each $i\in[m]$. We claim that for any $i\in[m]$,

$$B_i\cap C_i^c\Rightarrow E^c,$$

which can be verified through the following argument. Fix $i\in[m]$, and note that the event $B_i\cap C_i^c$ means that both $1\notin\mathcal{Y}_i,2\in\mathcal{Y}_i$ and $Y'_i\neq 2$. Under our classifier model, we have $\mathbb{P}(Y_i=2\,|\,Z=2)=1$, or equivalently $\mathbb{P}(Y_i\neq 2\,|\,Z=2)=0$. It follows that $\mathcal{L}(Y';2)=0$, whereas $\mathcal{L}(Y';1)>0$ since Y' was drawn from $\mathcal{S}_1$, the output set of class 1, thus proving the claim.

As a consequence of the above claim, we have that

$$\bigcup_{i=1}^{m}(B_i\cap C_i^c)\subseteq E^c$$

and hence

$$E\subseteq\left(\bigcup_{i=1}^{m}(B_i\cap C_i^c)\right)^c$$

$$=\bigcap_{i=1}^{m}(B_i^c\cup C_i)$$

$$=\bigcap_{i=1}^{m}\Big(B_i^c\cup(C_i\cap B_i)\Big).$$

It follows that

$$\begin{aligned}
P_e &\le K \cdot \mathbb{P}(E) \\
&\le K \cdot \mathbb{P}\Big(\bigcap_{i=1}^{m} \Big(B_i^c \cup (C_i \cap B_i)\Big)\Big) \\
&= K \cdot \prod_{i=1}^{m} \mathbb{P}(B_i^c \cup (C_i \cap B_i)) \\
&= K \cdot \mathbb{P}(B_1^c \cup (C_1 \cap B_1))^m \\
&= K \cdot \Big(\mathbb{P}(B_1^c) + \mathbb{P}(C_1 \,|\, B_1)\mathbb{P}(B_1)\Big)^m \\
&= K \cdot \Big(1 - \mathbb{P}(B_1) + \mathbb{P}(C_1 \,|\, B_1)\mathbb{P}(B_1)\Big)^m.
\end{aligned}$$

It now remains to compute $\mathbb{P}(B_1)$ and $\mathbb{P}(C_1 \,|\, B_1)$. First, note that conditional on $B_1 : 1 \notin \mathcal{Y}_1, 2 \in \mathcal{Y}_1$, the probability of $C_1 : Y_1' = 2$ is exactly $1/R$ (recall that Y' is a random output conditional on $Z = 1$). Hence,

$$\mathbb{P}(C_1 \,|\, B_1) = \frac{1}{R}.$$

Secondly, under the random construction in which classifiers are selected independently and uniformly at random from the set of all size-R classifiers, the probability of B_1 is proportional to the number of ways to choose the remaining $(R-1)$ classes from the remaining $(K-2)$ total classes (since we are constraining the classifier to contain class 2 but not class 1). That is,

$$\mathbb{P}(B_1) = \frac{\binom{K-2}{R-1}}{\binom{K}{R}} = \frac{R(K-R)}{K(K-1)},$$

and we may now continue the previous bound as follows:

$$\begin{aligned}
P_e &\le K \cdot \Big(1 - \frac{R(K-R)}{K(K-1)} + \frac{K-R}{K(K-1)}\Big)^m \\
&= K \cdot \Big(1 - \frac{(K-R)(R-1)}{K(K-1)}\Big)^m \\
&\le K \cdot \exp\Big(-m \cdot \frac{(K-R)(R-1)}{K(K-1)}\Big).
\end{aligned}$$

To ensure that $P_e \le \epsilon$, we need

$$m \ge \frac{K(K-1)}{(K-R)(R-1)} \cdot \log\Big(\frac{K}{\epsilon}\Big)$$

and thus it suffices to have

$$m = \left\lceil \frac{K(K-1)}{(K-R)(R-1)} \cdot \log\Big(\frac{K}{\epsilon}\Big) \right\rceil.$$

□