IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 29, 2020, accepted November 12, 2020, date of publication November 17, 2020,
date of current version December 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3038490

User Clustering for MIMO NOMA via Classifier Chains and Gradient-Boosting Decision Trees

CHAOUKI BEN ISSAID[1], (Associate Member, IEEE), CARLES ANTÓN-HARO[2], (Senior Member, IEEE), XAVIER MESTRE[2], AND MOHAMED-SLIM ALOUINI[3], (Fellow, IEEE)

[1]Centre for Wireless Communications (CWC), University of Oulu, 90570 Oulu, Finland
[2]Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/iCERCA), Parc Mediterrani Tecnologia (PMT), 08860 Castelldefels, Spain
[3]Computer, Electrical and Mathematical Science and Engineering (CEMSE) Division, King Abdullah University of Science and Technology (KAUST), Thuwal 23955, Saudi Arabia

Corresponding author: Carles Antón-Haro (carles.anton@cttc.es)

This work was supported in part by the Spanish Government through the Aprendizaje Estadístico e Inferencia para Sistemas de Comunicación de Alta Dimensionalidad (ARISTIDES) project under Grant RTI2018-099722-B-I00 and in part by the King Abdullah University of Science and Technology (KAUST).

ABSTRACT In this article, we propose a data-driven approach to group users in a Non-Orthogonal Multiple Access (NOMA) MIMO setting. Specifically, we formulate user clustering as a multi-label classification problem and solve it by coupling a Classifier Chain (CC) with a Gradient Boosting Decision Tree (GBDT), namely, the LightGBM algorithm. The performance of the proposed CC-LightGBM scheme is assessed via numerical simulations. For benchmarking, we consider two classical adaptation learning schemes: Multi-Label k-Nearest Neighbours (ML-KNN) and Multi-Label Twin Support Vector Machines (ML-TSVM); as well as other naive approaches. Besides, we also compare the computational complexity of the proposed scheme with those of the aforementioned benchmarks.

INDEX TERMS NOMA, multi-label classification, classifier chains, gradient-boosting decision trees, user clustering.

I. INTRODUCTION

Non-orthogonal multiple access (NOMA) [1] has been intensively investigated in the context of (beyond) 5G wireless networks. NOMA makes it possible to serve more than one user in each resource block (RB), e.g., a time slot, subcarrier, spreading code, or space. Consequently, NOMA exhibits a higher spectral efficiency than *orthogonal* multiple access techniques. This is particularly relevant for scenarios with a massive number of connections requiring sporadic/low data-rate transmissions: allocating one entire RB to each connection would be largely inefficient here. Besides, NOMA architectures can be easily combined with multi-antenna (MIMO) techniques.

One of the main challenges with NOMA is how to group the users sharing the same RB efficiently. Most of the literature on NOMA has traditionally considered the case of two users per resource block, because it appears complicated to be able to successfully separate more than two codewords under imperfect channel state information conditions. In a multi-antenna setting, this translates into two users per degree of freedom, understood as RB per antenna. Equivalently, for the MIMO scenario considered in this work, we propose to group users into *two subsets* (clusters). To decode users, we use a Linear Minimum Mean Square Error (LMMSE) receiver in combination with Successive Interference Cancellation (SIC) [2]. The challenge here is that the number of clustering solutions (CS) grows exponentially in the number of users. This precludes the use of an exhaustive search to identify the optimal CS in terms of sum-rate. We also want to depart from greedy/heuristic approaches attempting to e.g., minimize inter-cluster interference [3] or maximize channel gain disparity [4] in order to accomplish the task in a computationally-affordable manner. Instead, we design a clustering strategy based on data-driven methodologies (see e.g., [5], [6]) which generally offer a good trade-off between performance and computational complexity. Specifically, we model user grouping as a *multi-label* (ML) classification problem where binary labels indicate the cluster to which each user belongs. Admittedly, the so-called label power-set method allows to transform a ML classification problem into multi-class classification one by combining entire label sets into a single atomic label. However, the number of such atomic labels increases exponentially in the

The associate editor coordinating the review of this manuscript and approving it for publication was Christoph Studer.

number of original labels (the curse of dimensionality problem), which translates into unaffordable complexity. To avoid that, a number of specific multi-label classification techniques have been developed. Those techniques can be categorized [7] into (*i*) *transformation-learning* methods, which decompose the multi-label problem into several single-label ones (binary relevance method, BM), or transform it into a label ranking problem (e.g., via calibrated label ranking); and (*ii*) *adaptation learning* methods, which modify single-label algorithms so that they can directly process multi-label data: multi-label k Nearest Neighbours (ML-kNN), Decision Trees (ML-DT), or Twin Support Vector Machines (ML-TSVM). Even if BM is widely used in the literature, it disregards pair-wise label correlations (e.g., the fact that two specific labels frequently/ seldom co-occur). Neglecting such side information has a negative impact on the performance of the individual classifiers. To circumvent that, Classifier Chains (CC) [8], [9] can be used to link the individual classifiers along a probability chain.

Contribution: In this work, we model user clustering as a multi-label classification problem. Notably, this application area is radically different from that of text categorization (in multiple simultaneous topics) where multi-label learning techniques originated [10]. Further, ML classification allows to overcome the scalability problem found in our previous user clustering work [11] which was based on the power label set method. We solve the problem by coupling a *boosted* decision tree (DT) for each single-label binary classifier (i.e., BM approach within transformation learning) with a classifier chain, to account for label correlation which emerges from a pre-sorting of users. As for the boosting strategy, we adopt a *gradient*-based (vs. bagging) approach [12], which is able to enhance the limited generalization capability of the (low-complexity) DTs used as base learners. To minimize the computational complexity of the resulting Gradient Boosting Decision Tree (GBDT), we implement it via Microsoft's LightGBM algorithm [13]. The overall approach will be referred to in the sequel as a Classifier Chain-based LightGBM (CC-LightGBM). To the best of authors' knowledge, the combination of CC with LightGBM is novel and, further, it has never been used to solve a user clustering problem for MIMO NOMA. Besides, we assess the performance of CC-LightGBM via numerical simulations and use two classical adaptation learning algorithms from the literature, ML-kNN [14] and ML-TSVM [15], as benchmarks.

II. SIGNAL AND SYSTEM MODEL

Consider the uplink of a multi-user SIMO system where one Base Station (BS) equipped with N_{BS} antennas serves K single-antenna users. The received signal at the BS reads[1]

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}, \tag{1}$$

where $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_K] \in \mathbb{C}^{N_{\mathrm{BS}} \times K}$ is the channel matrix with $\mathbf{h}_k \in \mathbb{C}^{N_{\mathrm{BS}}}$ standing for the (column) channel vector associated to the k^{th} user; vector $\mathbf{s} \in \mathbb{C}^K$ accounts for the transmit signal, with its elements fulfilling $\mathbb{E}\left[|s_k|^2\right] = 1; \forall k$, that is, transmit power is identical for all nodes and no power control mechanisms are in place. Finally, $\mathbf{n} \in \mathbb{C}^{N_{\mathrm{BS}}}$ denotes zero-mean i.i.d. additive white Gaussian noise of variance σ^2, namely, $\mathbf{n} \sim \mathcal{CN}(0, \sigma^2 \mathbf{I}_{N_{\mathrm{BS}}})$. Nodes operate at a central frequency f_c and are uniformly distributed in the served cell. We also assume that full Channel State Information (CSI) is available at the BS. We adopt a geometric channel model with L_p scattering paths that is widely used in the literature (see e.g., [16]). Hence, for each column in the channel matrix $\mathbf{H}$ we have that

$$\mathbf{h}_k = \frac{1}{\rho_k} \sum_{l=1}^{L_p} \alpha_{k,l} \mathbf{a}(\theta_{k,l}), \tag{2}$$

with ρ_k accounting for the path-loss *and* shadow-fading associated to the k-th node. Consequently, we have that $\rho_k = \sqrt{(1 + d_k^{\eta})/10^{\frac{\beta_k}{10}}}$, where d_k is the node-to-BS distance, η denotes the path-loss exponent (typically, $\eta \in [2, 6]$); and $\beta_k \sim \mathcal{N}(0, \sigma_\beta^2)$ is the spatially-uncorrelated shadow-fading coefficient, with σ_β^2 typically ranging from 6 dB (free-space propagation) to 10 dB (indoor environments) [17]. Further, the coefficient $\alpha_{k,l}$ of (2) is the complex gain of the l-th path, with $\mathbb{E}\left[|\alpha_{k,l}|^2\right] = 1$; and $\theta_{k,l} \sim \mathcal{N}(\bar{\theta}_k, \sigma_\theta^2)$ denotes the angle-of-arrival (AoA) of the l-th path of node k, with $\bar{\theta}_k$ associated to the actual location of such node. Vector $\mathbf{a}(\theta_{k,l}) \in \mathbb{C}^{N_{\mathrm{BS}} \times 1}$ accounts for the antenna array response. For a uniform linear array, it reads

$$\mathbf{a}\left(\theta_{k,l}\right) = \left[1, e^{-j\frac{2\pi}{\lambda} d \sin(\theta_{k,l})}, \ldots, e^{-j(N_{\mathrm{BS}}-1)\frac{2\pi}{\lambda} d \sin(\theta_{k,l})}\right]^T,$$

where λ is the signal wavelength, and d is the distance between antenna elements.

A. MULTIPLE-ACCESS AND DECODING STRATEGIES

Let $\mathcal{K}$ denote the set of K active users in the system. We partition $\mathcal{K}$ into *two* disjoint subsets (clusters) $\mathcal{K}_1$ and $\mathcal{K}_2$ of cardinalities K_1 and K_2, and such that $K = K_1 + K_2$. The subset $\mathcal{K}_1$ is decoded first. After detection, the received signal associated to those users is reconstructed and its contribution removed from $\mathbf{y}$ (i.e., via Successive Interference Cancellation, SIC [2]). After this interference cancellation step, the nodes in the $\mathcal{K}_2$ subset are finally decoded. For a Linear Minimum Mean Square Error (LMMSE) receiver, the optimal beamformers $\mathbf{w}_k^{(1,2)} \in \mathbb{C}^{N_{\mathrm{BS}}}$ associated to an arbitrary node k in $\mathcal{K}_1$ or $\mathcal{K}_2$ read, respectively (see Section 8.3.3 in [18]),

$$\mathbf{w}_k^{(1)} = (\mathbf{H}\mathbf{H}^H + \sigma^2 \mathbf{I}_{N_{\mathrm{BS}}})^{-1} \mathbf{h}_k, \tag{3}$$

$$\mathbf{w}_k^{(2)} = (\mathbf{H}_{(\mathcal{K}_1)} \mathbf{H}_{(\mathcal{K}_1)}^H + \sigma^2 \mathbf{I}_{N_{\mathrm{BS}}})^{-1} \mathbf{h}_k, \tag{4}$$

where matrix $\mathbf{H}_{(\mathcal{K}_1)}$ contains all the columns of $\mathbf{H}$ except for those corresponding to nodes from subset $\mathcal{K}_1$. Based on (3) and (4), one can easily prove that the instantaneous SINR for

[1] Of course, the same signal model is valid in a multi-user MIMO setting transmitting according to a purely spatial multiplexing technique, where K would be the total number of transmit antennas.

an arbitrary node in the first/second subsets read, respectively (see further [18]):

$$\gamma_k^{(1)} = \mathbf{h}_n^H (\mathbf{H}_{(k)}\mathbf{H}_{(k)}^H + \sigma^2 \mathbf{I}_{N_{\text{BS}}})^{-1}\mathbf{h}_k, \tag{5}$$

$$\gamma_k^{(2)} = \mathbf{h}_n^H (\mathbf{H}_{(\mathcal{K}_1\cup\{k\})}\mathbf{H}_{(\mathcal{K}_1\cup\{k\})}^H + \sigma^2 \mathbf{I}_{N_{\text{BS}}})^{-1}\mathbf{h}_k. \tag{6}$$

Finally, the instantaneous sum-rate R can be expressed as

$$R = \sum_{k\in\mathcal{K}} \log_2(1 + \bar{l}_k\gamma_k^{(1)} + l_k\gamma_k^{(2)}), \tag{7}$$

where $l_k \in \{0, 1\}$ is an indicator variable such that $l_k = 0$ if node k belongs to subset $\mathcal{K}_1$, and 1 otherwise; and $\bar{l}_k \triangleq 1 - l_k$ denotes the opposite of l_k.

B. PRE-SORTING OF USERS

Prior to detection, we assume that users are sorted in such a way that the *spatial* correlation for consecutive users (i.e., users with successive indices after pre-sorting) is high. The rationale behind is as follows: when the spatial correlation is high, it becomes harder for the MIMO system to separate those users. Hence, consecutive users are likely to be assigned to different clusters (i.e., if $l_k = 0$ then $l_{k+1} = 1$). This strategy introduces additional constraints in the clustering process that, as we discuss in Section V ahead, improve the performance of the proposed data-driven approaches. For single-scattering LOS scenarios (i.e., $L_p = 1$), this can be readily accomplished by (*i*) estimating their respective AoAs (e.g., by means of spectral estimation methods such as minimum variance or MUSIC [19]); and (*ii*) sorting them in ascending order of their AoAs. For the more general case $L_p > 1$, one can first identify the *two* users out of the K active users exhibiting the highest correlation by checking the alignment of the vector channel responses (2) as in [6]; and then subsequently include the user with the highest alignment with the last one (that is, in a greedy manner).

III. FORMULATION AS A MULTI-LABEL CLASSIFICATION PROBLEM

Our goal is thus to define a partition of the set $\mathcal{K}$ of K nodes into two disjoint subsets $\mathcal{K}_1$ and $\mathcal{K}_2$. such that the sum-rate of a MIMO-NOMA system based on a LMMSE receiver with SIC is maximized. In the sequel, we will refer to each of those possible node partitions as a *Clustering Solution* (CS). More formally, the optimization problem can be formulated as follows:

$$\begin{aligned}\max_{\{l_1,\dots l_K\}} \quad & \sum_{k\in\mathcal{K}} \log_2(1 + \bar{l}_k\gamma_k^{(1)} + l_k\gamma_k^{(2)}) \\ \text{s.t. } & l_k \in \{0, 1\} \quad \text{for } k = 1 \dots K \end{aligned} \tag{8}$$

The task of selecting the optimal CS can be modeled as a supervised learning problem. Specifically it can be cast into a multi-label *binary* classification task where:

- The input is a vector $\mathbf{t} = \left[t_1, \dots, t_{N_\text{f}}\right]^T \in \mathbb{R}^{N_\text{f}\times 1}$ of *features* formed by stacking the real and imaginary parts of the entries in the channel matrix $\mathbf{H}$, with $N_\text{f} = 2KN_{\text{BS}}$.
- The output is a $K \times 1$ binary vector $\mathbf{l} = [l_1, \dots, l_K]^T$ of *labels* where the k^{th} element indicates the cluster the k^{th} user belongs to.

A. DESCRIPTION AND GENERATION OF THE TRAINING DATASET

The training dataset comprises a total of N_{tr} examples stacked in a training matrix $\mathbf{T} = \left[\mathbf{t}_1, \dots, \mathbf{t}_{N_{\text{tr}}}\right] \in \mathbb{R}^{N_\text{f}\times N_{\text{tr}}}$ as an input, where the subscript denotes the example index. And, as an output, a binary matrix $\mathbf{L} = \left[\mathbf{l}_1, \dots, \mathbf{l}_{N_{\text{tr}}}\right] \in \mathbb{N}^{K\times N_{\text{tr}}}$ which gathers the corresponding labels for each example. The generation of the training dataset is given in Algorithm 1.

Algorithm 1 Generation of the Training dataset

```
1: Inputs: $K, N_{\text{tr}}, \eta, \sigma_\beta, \sigma_\theta$.
2: Generation of the channel matrices $\mathbf{H}$ in (1) for each of
   the $N_{\text{tr}}$ realizations of the system scenario (node locations
   and $\{d_k\}_{k=1}^K$, $\{\bar{\theta}_k\}_{k=1}^K$, path-loss, shadow-fading).
3: Computation of the corresponding sum-rate as per (7) for
   each of the $N_{\text{CS}} = 2^K$ clustering solutions.
4: Let the label of each example be the set of $K$ binary vari-
   ables $\mathbf{l}_i$, $i = 1, \dots, N_{\text{CS}}$ associated to the CS yielding
   the highest sum-rate, which is determined via exhaustive
   search.
```

To avoid significant bias in the training, features are normalized prior to their use by the learning scheme, namely, $t_{ij} \leftarrow \left(t_{ij} - \mathbb{E}_i[t_{ij}]\right) / \left(\max_i \left[t_{ij}\right] - \min_i \left[t_{ij}\right]\right)$, with $\mathbb{E}_i[\cdot]$ denoting the row-wise empirical average in matrix $\mathbf{T}$.

B. ANALYSIS OF LABEL CORRELATIONS

Figure 1 below depicts the pair-wise Pearson correlation coefficients between the labels of a training dataset generated according to the above procedure (for the computations, the binary labels were mapped onto the $\{-1, 1\}$ set). As discussed in Section II-B, users are pre-sorted in an increasing order of their angles of arrival ($L_p = 1$ case). Off-diagonal elements in Fig. 1 (left) confirm that, for the $K = 6$ scenario, label correlation is quite high for consecutive users (the ones which are spatially closer), as conjectured in Section II. The fact that correlation takes negative values indicates that in roughly 45% of the cases, consecutive users are assigned to different clusters. On the contrary, label correlation decreases rapidly for non-consecutive ones. In Fig. 1 (right), we observe a stronger correlation for non-consecutive users since, in a scenario with $K = 9$ users, they are spatially closer.

Such empirical observations substantiate the need for introducing specific mechanisms to exploit correlation in the aforementioned single-label classification problems, as we discuss next.

IV. CC-LightGBM: CLASSIFIER CHAINS WITH LIGHT GRADIENT BOOSTING MACHINE

In this section, we provide an overview of the two algorithms that will be combined to solve our multi-label classification problem efficiently. On the one hand, a Classifier Chain (CC)

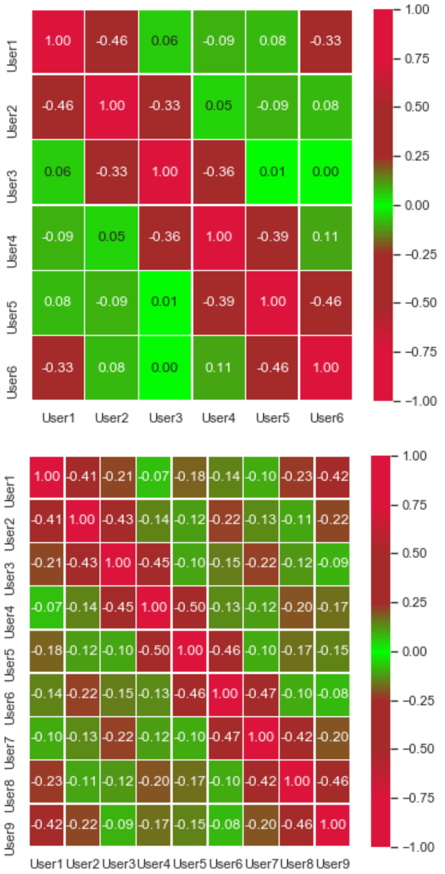


FIGURE 1. Matrix with pair-wise label correlation coefficients for a system with $K = 6$ (top) and $K = 9$ (bottom) users sorted according to increasing angles-of-arrival ($N_{\mathrm{BS}} = 4$ antennas).

will be adopted to exploit label correlation. On the other, Gradient Boosted Decision Trees (GBDT) and, more specifically, the LightGBM algorithm will be used to build the K single-label binary classifiers.

A. CLASSIFIER CHAINS (CC)

The Classifier Chain model (CC) involves K binary classifiers $\{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$ linked along a chain. Each classifier $\mathcal{C}_k$ is responsible for learning and predicting the binary association of label l_k given the original feature space $\mathbf{t}$ described in Section III (i.e., entries in the channel matrix), augmented by all prior binary relevance predictions in the chain, namely, $l_1, \ldots, l_{k-1}$, for $k \leq 2$. The *augmented* feature space $\mathbf{x}_k$ can thus be defined as

$$\mathbf{x}_k = \begin{cases} \mathbf{t}, & k = 1 \\ [\mathbf{t}^T, l_1, \ldots, l_{k-1}]^T, & k = 2, \ldots, K. \end{cases} \tag{9}$$

This augmented feature space will be used by each classifier $\mathcal{C}_k$ in the chain in order to compute

$$\hat{p}(\mathbf{x}_k) = \Pr(l_k = 1 | \mathbf{x}_k), \tag{10}$$

namely, the conditional probability of $l_k = 1$ given $\mathbf{x}_k$. Using this construction, label information is propagated among classifiers and, hence, their correlation is explicitly taken into account. Even if, on average, $K/2$ inputs are added to each individual classifier, in general this is a small number in comparison with the original number of features ($N_{\mathrm{f}} = 2KN_{\mathrm{BS}}$). It is important to note that the K single-label classification problems are solved *sequentially* and, thus, the specific ordering of the chain may have a remarkable impact on performance. *Ensemble* Chain Classifiers [8] allow to perform an averaging over orderings and training data subsets. However, for our scenario with pre-sorted users the *baseline* CC yields excellent performance results (see Section V ahead), which allows to avoid the additional computational complexity that ensemble methods entail.

B. GRADIENT BOOSTING DECISION TREES (GBDT)

Here, we describe how to build each of the aforementioned K single-label binary classifiers $\mathcal{C}_k$. Decision trees (DT) are known to be powerful tools for classification and regression tasks. The main advantages are their low complexity once trained/constructed, and their understandability from a human viewpoint (a sequence of split decisions). To construct a DT, the root node is first divided (split) into two children nodes as a function of a given feature in the augmented feature space $\mathbf{x}_k$ of (9) (e.g., $x_{k,4}$ in Fig. 2, namely, the fourth feature in $\mathbf{x}_k$), and a splitting point (equal to 10, in this example) to be identified, see next paragraph. The entire training dataset, which is initially associated to the root node, is split accordingly: all records with e.g., $x_{k,4} < 10$ go to the left node, otherwise to the right node. The process is iterated with the children nodes. If a node does not split into further nodes, then it is called a leaf node (grayed nodes in Fig. 2).

To grow a tree, the main tasks are thus: (*i*) to identify the optimal feature to split a node; (*ii*) to determine the splitting points, in particular for continuous features; and, in leaf-wise (vs. level-wise) strategies, (*iii*) to decide which node to split next. Tasks (*i*) and (*ii*) typically entail an exhaustive search over features/splitting points for each node. As for (*iii*), the goal when splitting a parent node is to increase the homogeneity of the records in the children nodes and, ultimately, in the leaves. This means that, in classification problems, records in children nodes should belong to fewer classes; and for regression ones, they should have similar output values. Typical homogeneity measures include Gini gain (i.e., reduction of Gini Impurity Index) or the Information Gain (namely, decrease of entropy) between the parent and children nodes, for classification; or decrease of variance/MSE-related measures, for regression (see e.g., [20]). Those measures help determine which node should be split next, in general in a greedy manner (e.g, the node resulting in the largest information gain). Node splitting stops when the number of tree levels reaches a pre-defined maximum value, the number of records in a node is below some threshold, or the records in a node are homogenous enough (e.g., low impurity index). Leaf nodes, each accounting for a disjoint region $\mathcal{R}_j$ of the feature space, determine the output of the decision tree for any new example in the test dataset.

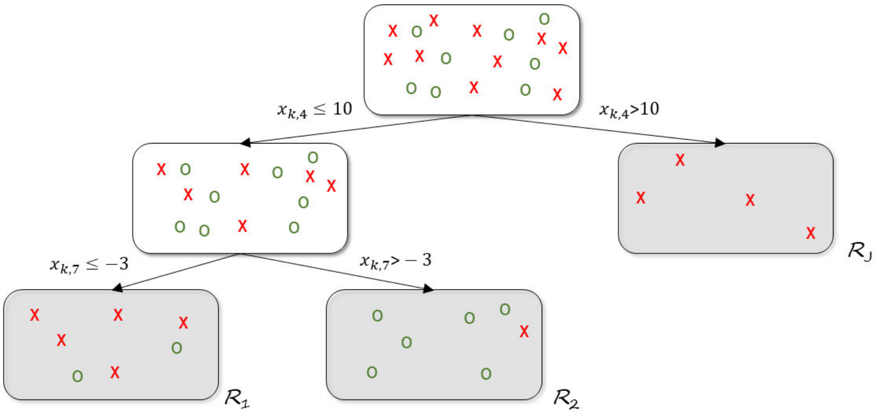


FIGURE 2. Sample decision tree for binary classification with $J = 3$ leaf nodes each associated to a disjoint region.

For *classification*, this is accomplished by taking the class with the highest probability in that particular leaf/region (see Fig. 2). For *regression*, on the contrary, the output associated to each leaf is given by some average (e.g., mean, median) of the output values of the examples in the training dataset.

Boosting is a method of converting an ensemble of *weak* (or base) learners such as DTs into a strong learner. Subsequent trees help classify observations that are not well classified by the previous ones. And the prediction of the final ensemble model is the weighted sum of the predictions made by the DTs. In boosting, each new tree is a fit on a modified version of the original data set. In Gradient Boosting Decision Trees (GBDT), this modified version follows from the gradient of the loss function for the ensemble model.

To maximize binary classification accuracy, one needs to minimize the cross-entropy loss (also referred to as log-loss or deviance in the literature) for all the examples in the training dataset, namely,

$$\begin{aligned}\mathcal{L}_k &= \sum_{i=1}^{N_{\text{tr}}} \mathcal{L}_k\left(l_{k,i}, \hat{p}(\mathbf{x}_{k,i})\right) \\ &= -\sum_{i=1}^{N_{\text{tr}}} l_{k,i} \cdot \log\left(\hat{p}(\mathbf{x}_{k,i})\right) + (1 - l_{k,i}) \cdot \log\left(1 - \hat{p}(\mathbf{x}_{k,i})\right)\end{aligned} \tag{11}$$

with $\mathbf{x}_{k,i}$ denoting the i-th example of the augmented feature space defined in (9); $l_{k,i}$ accounting for the k-th label associated to the i-th example, as defined in Section IV-A; and $\hat{p}(\mathbf{x}_{k,i})$ standing for the a posteriori probability in (10). For brevity, in the sequel we will omit the user/classifier index k in the loss function and its parameters. A common approach (see Fig. 3) is to build an ensemble model and *train* it to compute the log-likelihood ratio, namely,

$$F(\mathbf{x}) = \log\left(\frac{\Pr(l = 1|\mathbf{x})}{\Pr(l = 0|\mathbf{x})}\right) = \log\left(\frac{\hat{p}(\mathbf{x})}{1 - \hat{p}(\mathbf{x})}\right), \tag{12}$$

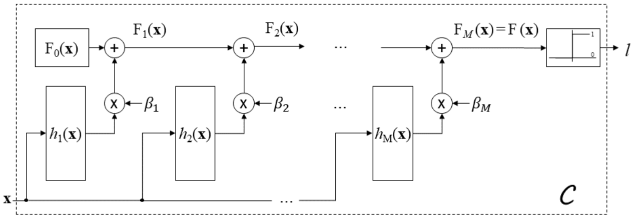


FIGURE 3. Gradient-boosting architecture for binary classification with M base learners.

such that the corresponding label can, in turn, be estimated as $\hat{l} = \mathrm{u}(F(\mathbf{x}))$, with $\mathrm{u}(\cdot)$ standing for the Heaviside step function. A boosting strategy approximates $F(\mathbf{x})$ by an additive expansion of the form

$$F(\mathbf{x}) = \sum_{m=0}^{M} \beta_m h_m(\mathbf{x}), \tag{13}$$

where β_m denotes the m-th expansion coefficient; and function $h_m(\mathbf{x}) = h(\mathbf{x}; \mathbf{a}_m)$ is the corresponding base learner, with $\mathbf{a}_m$ denoting its parameter set (optimal splitting features, splitting points, average output value for each leaf, etc; see Section IV-B). Notice that, despite that the resulting GBDT solves a *classification* problem, the base learners $h_m(x)$ turn out to be *regression* trees with real-valued outputs whose summation yields[2] $F(\mathbf{x})$. The coefficients β_m and the parameters $\mathbf{a}_m$ are jointly fit to the training data in a forward stage-wise manner, as we explain next. Starting from an initial guess $F_0(\mathbf{x})$, for $m = 1, \ldots, M$ we thus have

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \beta_m h_m(\mathbf{x}), \tag{14}$$

as Fig. 3 illustrates. For an arbitrary differentiable loss function $\mathcal{L}(l, F)$, this problem can be approximately solved with a two-step procedure [12]. First, function $h_m(\mathbf{x})$ is fit to the current pseudo-residuals given by the gradient of the preceding

[2] *Classification* trees, each providing a prediction of $\hat{p}$ cannot be added up to get a useful quantity.

loss function $F_{m-1}(\mathbf{x})$, namely,

$$\begin{aligned} r_{m,i} &= -\left[\frac{\partial \mathcal{L}(l_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i))}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})} \\ &= l_i \cdot \frac{e^{-F_{m-1}(\mathbf{x}_i)}}{1+e^{-F_{m-1}(\mathbf{x}_i)}} - (1-l_i)\frac{e^{F_{m-1}(\mathbf{x}_i)}}{1+e^{F_{m-1}(\mathbf{x}_i)}}, \quad (15) \end{aligned}$$

with i denoting the training example index. And where, in the second equality, we have used the change of variables $\hat{p}(\mathbf{x}) = \frac{1}{1+e^{-F(\mathbf{x})}}$ resulting from equation (12). This step is followed by a single parameter optimization for the step size β_m based on the general criterion $\mathcal{L}$ defined in (11). With regression trees as base learners, as it follows from Section IV-B, recursion (14) can be *approximately* (and efficiently) computed region-wise [12] as

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \sum_{j=1}^{J} \nu_{m,j} \mathbb{1}[\mathbf{x} \in \mathcal{R}_{m,j}], \quad (16)$$

where $\mathbb{1}[\cdot]$ is the indicator function, $\mathcal{R}_{m,j}$ denotes the j-th disjoint region of the m-the tree (base learner); and $\nu_{m,j}$, the output value associated to such disjoint region (leaf), is given by

$$\nu_{m,j} = \frac{\sum_{\mathbf{x}_i \in \mathcal{R}_{m,j}} r_{m,i}}{\sum_{\mathbf{x}_i \in \mathcal{R}_{m,j}} (l_i - r_{m,i})(l_i - r_{m,i} - 1)}. \quad (17)$$

As for the initial guess $F_0(\mathbf{x})$, we let it be

$$F_0(\mathbf{x}) = \arg\min_{\nu} \sum_{i=1}^{N_{\text{tr}}} \mathcal{L}(l_i, \nu) = \log\left(\frac{\bar{l}}{1-\bar{l}}\right), \quad (18)$$

where $\bar{l} \triangleq \frac{1}{N_{\text{tr}}} \sum_{i=1}^{N_{\text{tr}}} l_i$ can be interpreted as the *a priori* equivalent to probability $\hat{p}(\mathbf{x})$ in (12) since, clearly, $\bar{l} = \Pr(l = 1)$.

C. LightGBM: A SCALABLE GBDT

One major difficulty with GBDTs is their limited scalability when the feature dimension is high and/or the training dataset is large. This stems from the fact that, for *each* feature, the regression tree in each base learner needs to scan *every* example of the dataset to estimate the information gain of all possible split points. This, of course, is computationally intensive. To alleviate this, LightGBM [13] introduces two novel techniques, namely, (*i*) Gradient-based One-Side Sampling, by which only examples in the dataset with larger gradients (and just a fraction of those with smaller gradients) are used for the information gain computations needed to grow the tree; and (*ii*) Exclusive Feature Bundling, which reduces the number of effective features by bundling together those which rarely take non-zero values (i.e. are mutually exclusive). Besides, LightGBM consumes very few memory resources compared to other baseline classifiers in the literature [13].

V. COMPUTER SIMULATION RESULTS

In this section, we assess the performance of the proposed CC-LightGBM scheme for a system scenario with one multi-antenna Base Station and K single-antenna terminals/users (uplink). The main parameters used for computer simulations can be found in Table 1. For LightGBM, we used Microsoft's implementation as a Python library (https://lightgbm.readthedocs.io). Performance is measured in terms of (*i*) Sum-Rate Loss (SRL), that is, the loss (in percentage) with respect to the sum-rate achievable by the optimal clustering solution which is found via exhaustive search; and (*ii*) Hamming Loss (HL) for multi-label problems, which can be defined as [9, Eq. (5)]

TABLE 1. Main Parameters Used in Computer Simulations.

Parameter	Value
Size of the test dataset, N_{tst}	2000
Number of BS antennas, N_{BS}	4
Number of scattering paths, L_p	1
Distance between antenna elements, d	0.5λ
Standard deviation of the noise, σ	1
Standard deviation of the Lognormal fading exponent, σ_α	7
Path-loss exponent, β	3.66

$$\text{HL} \triangleq \frac{1}{N_{\text{tst}} K} \sum_{i=1}^{N_{\text{tst}}} \sum_{k=1}^{K} \mathbb{1}\left[l_{k,i} \neq \hat{l}_{k,i}\right]. \quad (19)$$

The HL allows to evaluate, for each example in the test dataset, the fraction of wrongly classified labels (users) to the total number of labels[3]. Notice that the minimization of the cross-entropy loss, the score function adopted in Section IV-B for each individual classifier, along with the CC framework results into the minimization of the HL.

A. PERFORMANCE OF CC-LightGBM

First, in Fig. 4 we analyze the performance of the CC-LightGBM as a function of the number of the base learners (M) used. This is accomplished for a range of learning rates[4] $\lambda_r \in \{0.1, 0.05, 0.01\}$, while keeping other hyperparameters to their default values. For an increasing M, the residuals to which subsequent learners are fit become smaller. Consequently, the resulting regression tree is able to generate increasingly complex decision regions. As a result, the HL (i.e., the percentage of misclassified users) depicted in Fig. 4b decreases and, hence, so does the SRL in Fig. 4a. Interestingly enough, the SRL values are lower than those of the HL (e.g., for $M = 10^5$, HL equals 0.13 whereas SRL is close to 1%). This is due to the fact that, even if one user/few users have not been assigned to the right cluster[5], the sum-rate for the *wrong* clustering solution can still be quite high. The reason is two-fold: (*i*) the LMMSE receiver can accommodate one extra user in a cluster at the expense of some SINR penalty; and (*ii*) we have assumed that users in the first subset can always be reliably decoded irrespectively

[3] This different from the so-called 0/1 loss, where a prediction is deemed to be correct if and only if all its label predictions are correct.

[4] After computing the optimal step size β_m in (14), it is further shrunk (multiplied) by a λ_r factor since, empirically, it was found that small values ($\lambda_r \leq 0.1$) lead to much smaller generalization error [12]

[5] Notice that for HL $= 0.13$ and $K = 10$, roughly one user per example is misclassified, on average.

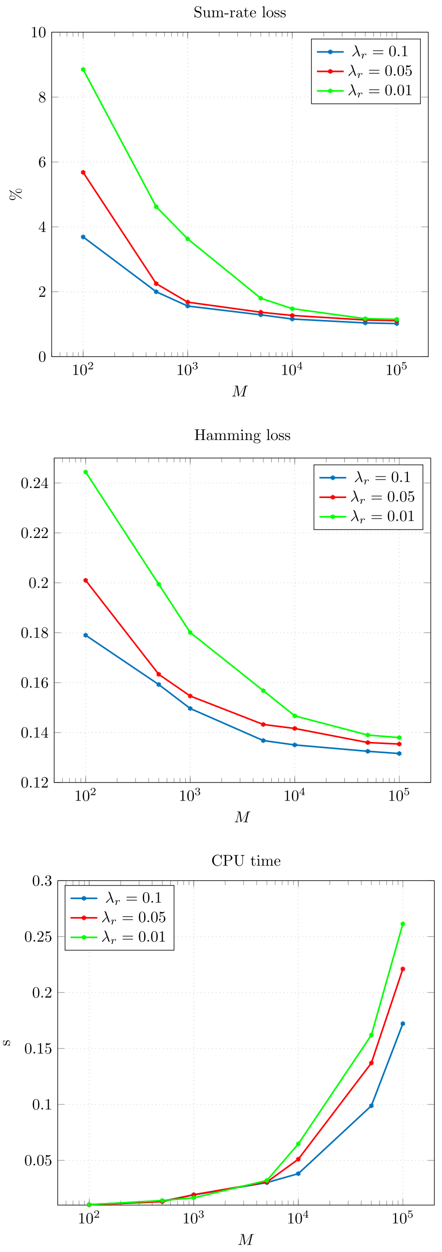


FIGURE 4. Performance of CC-LightGBM in terms of: (a) Sum-rate loss; (b) Hamming loss; and (c) CPU time on a 1.8 GHz Intel Core i5 CPU with a 16 GB 1.6 GHz DDR3 RAM; as a function of the number of base learners M ($K = 10$ users, $N_{\text{tr}} = 10^5$).

of their SINR (i.e., no error propagation among subsets). Further, Fig. 4a reveals that the SRL reaches a floor, which depends on the size of the training dataset as we discuss in the next paragraph, when the number of classifiers is in the range of 10^4 to 10^5. However, from Fig. 4c, the CPU time rapidly increases beyond $M = 10^4$. Consequently, for this setting it is advisable to use a number of base learners in the range of $M \in [10^4, 10^5]$. Finally, we observe that lower values of the learning rate λ_r result in a slower decrease of both the HL and the SRL.

Complementarily, Fig. 5 depicts the SRL and HL as a function of the number of examples in the training dataset N_{tr}.

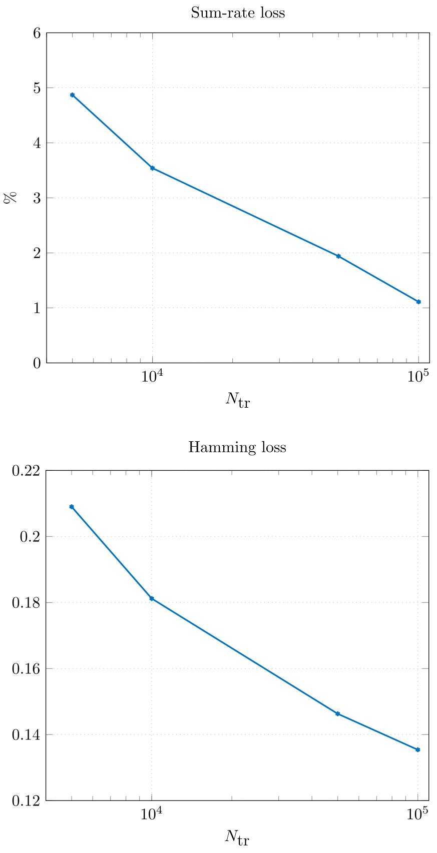


FIGURE 5. Performance of CC-LightGBM in terms of: (a) Sum-rate loss; and (b) Hamming loss as function of the training size N_{tr} ($K = 10$ users, $M = 10^5$ base learners).

As expected, larger training datasets result in lower losses. Specifically, the SRL decreases from 4.9% to roughly 1.1% when the number of examples increases from $5 \cdot 10^3$ to roughly 10^5. And, accordingly, the HL decreases from 0.21 to 0.135.

B. ALGORITHM BENCHMARKING

Next, we benchmark the proposed scheme with other approaches specifically tailored to *multi-label* classification problems, namely,

- **Multi-label k Nearest Neighbours** (ML-KNN) [14]: As its single-label counterpart, ML-KNN first identifies the k nearest neighbours $\mathcal{N}_{\mathbf{x}}$ in the training set for a given (new) test example $\mathbf{x}$. Then, it counts how many times *each* label (user) can be found in $\mathcal{N}_{\mathbf{x}}$. Finally, a decision is made on a per-label basis. Specifically, ML-KNN lets the l-th label in the test example be equal to 1 if its *a posteriori* probability given the number of occurrences of such label in $\mathcal{N}_{\mathbf{x}}$ is larger than that of being equal to 0 (i.e., Maximum A Posteriori principle).The a posteriori probabilities can be estimated from the training set (based on counting) and the Bayes rule.
- **Multi-label Twin Support Vector Machine** (ML-TSVM) [15]: To capture the multi-label information in the data, ML-TSVM builds a *set* of K proximal

hyperplanes (i.e., as many as labels/users) such that each hyperplane is as close as possible to the instances in the training set *with* the l^{th} label, and as far as possible from the others. The multiple non-parallel hyperplanes can be found by solving a set of quadratic programming problems. For simulations, we use a Gaussian (radial-based) kernel function. The optimal hyperparameters, namely, the variance of the Gaussian kernel ψ and the regularization parameter $c \in \mathbb{R}^+$ were found via grid search in the $[2^{-6}, 2^6]$ range.

We also consider three additional naive strategies as benchmarks: (*i*) the so-called '1010 strategy', a low-complexity approach in line with our discussions in Section II-B which assigns consecutive users to clusters in an alternating manner (i.e., alternatively, to cluster 1 or 0); (*ii*) 'random' user clustering; and (*iii*) 'CC-LightGBM wo/sort', that is, CC-LightGBM *without* user pre-sorting.

Figure 6 depicts the SRL as a function of the number of active users. The proposed CC-LightGBM scheme clearly outperforms all the benchmarks and naive strategies for the whole range of K values. Moreover, the gap is wider when the number of users is high. For $K = 12$ in particular, CC-LightGBM exhibits a SRL as low as 2%, whereas the loss for ML-KNN with an optimal number of neighbours is roughly 12% (one order of magnitude higher) or 30.3% for ML-TSVM (15 times higher). Such a large gain stems from (*i*) the gradient-boosting mechanism embedded in CC-LightGBM, that enhances accuracy in particular for large M; and (*ii*) the fact that the classifier chain explicitly models and leverages on inter-label (user) correlation, whereas this is simply ignored by ML-kNN and ML-TSVM. Moreover, the gain with respect to random user clustering is tremendous. For $K = 6$ users, the sum-rate degradation for random clustering is 20%; and for $K = 12$ users, it reaches 40%. The explanation for this behaviour can be found in Fig. 7 below. Even if the cost function in the optimization problem (8) is multi-modal in all cases, the sum-rate difference between maxima and minima is larger for increasing K (and so is loss when performing a random selection). All the above, of course, justifies the adoption of data-driven approaches to solve the user clustering problem.

Again in Fig. 6, the gap between the CC-LightGBM curves with and without user pre-sorting evidences the gain associated to the CC mechanism itself. Without pre-sorting, labels/users are totally uncorrelated. Consequently, the estimates from previously classified users provide no side information to be exploited by the CC. Specifically, for CC-LightGBM wo/sort and $K = 12$ users, its SRL is roughly 9%, this meaning that the 7% extra gain should be attributed to the CC mechanism.

Finally, the behaviour exhibited by the lazy '0101' strategy deserves some explanations. For a reduced number of users, its performance is comparable to that of the ML-TSVM or ML-kNN approaches. For large K, on the contrary, the 0101 strategy performs as poorly as random clustering. Figure 7 provides some insights into such behaviour. Whereas for $K = 6$ users the 0101..01 clustering solution or its symmetric solution (see vertical dashed lines in the plots) often provide the global maximum of the sum-rate various realizations, this is no longer the case for $K = 12$ users. Hence, in this case, there is no point in getting 'locked' to a given solution. This extent is further confirmed in the histogram of Fig. 8 where the peaks associated to the 0101..01 and 1010..10 clustering solutions for $K = 6$ (top) do not exist for $K = 12$ (bottom).

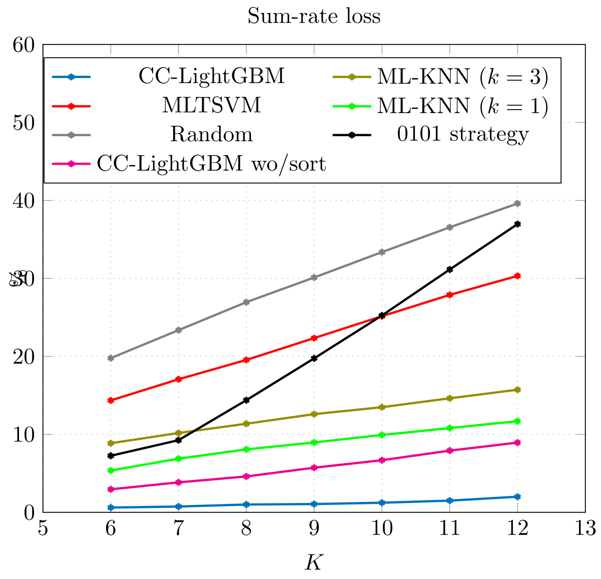


FIGURE 6. Sum-rate loss performance of the various approaches for a varying number of users.

C. COMPUTATIONAL COMPLEXITY ANALYSIS

In Table 2, we report on the computational complexities (for both the training and test phases) of the CC-LightGBM scheme and those of the various benchmarks. For ML-kNN in particular, the corresponding training and test complexities can be found in [7], with k denoting the number of nearest neighbours (a hyper-parameter) and $N_f = 2\,KN_{BS}$ accounting for the number of features in the input vector (see definition in Section III).

TABLE 2. Computational Complexities of CC-LightGBM and Several Benchmarks.

Algorithm	Training Complexity	Test Complexity
ML-kNN	$\mathcal{O}(N_f N_{tr}^2 + KkN_{tr})$	$\mathcal{O}(N_f N_{tst} + Kk)$
MLTSVM	$\mathcal{O}(KSN_{tr})$	$\mathcal{O}(KN_f N_{tst})$
CC-LightGBM	$\mathcal{O}(KBM\log_2(L)N_{tr})$	$\mathcal{O}(KM\log_2(L)N_{tst})$
Random	$--$	$\mathcal{O}(N_{tst})$
0101 strategy	$--$	$\mathcal{O}(N_{tst})$

As for MLTSVM, in the training phase it needs to solve K sub-quadratic programming problems (one for each label) by using the successive over-relaxation (SOR) solver. The complexity of the SOR solver is on the order of N_{tr} per iteration. If S denotes the number of iterations of SOR, then

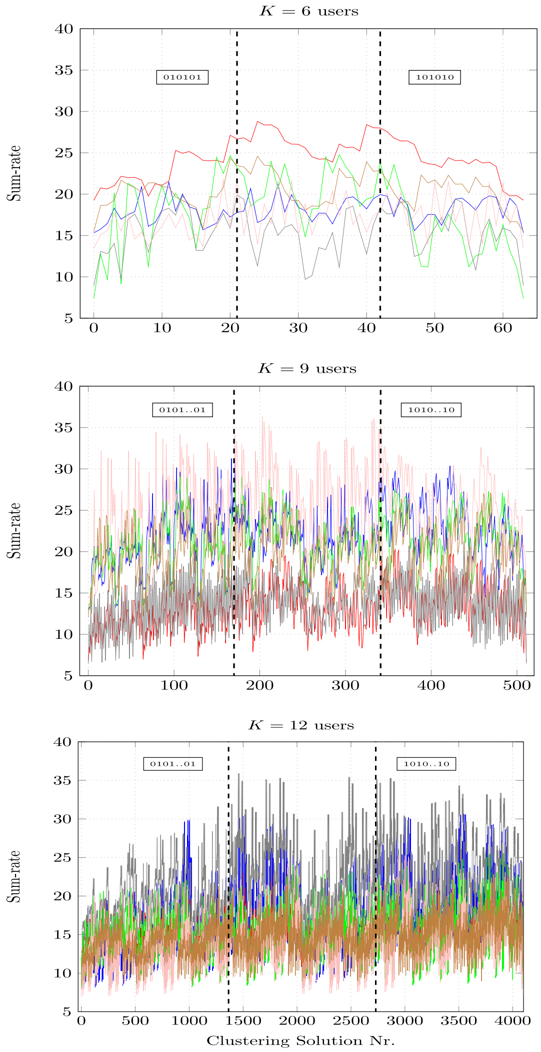


FIGURE 7. Sum-rate associated to each clustering solution for several examples/realizations (colors) and varying number of users: $K = 6$ (top), $K = 9$ (middle) and $K = 12$ (bottom).

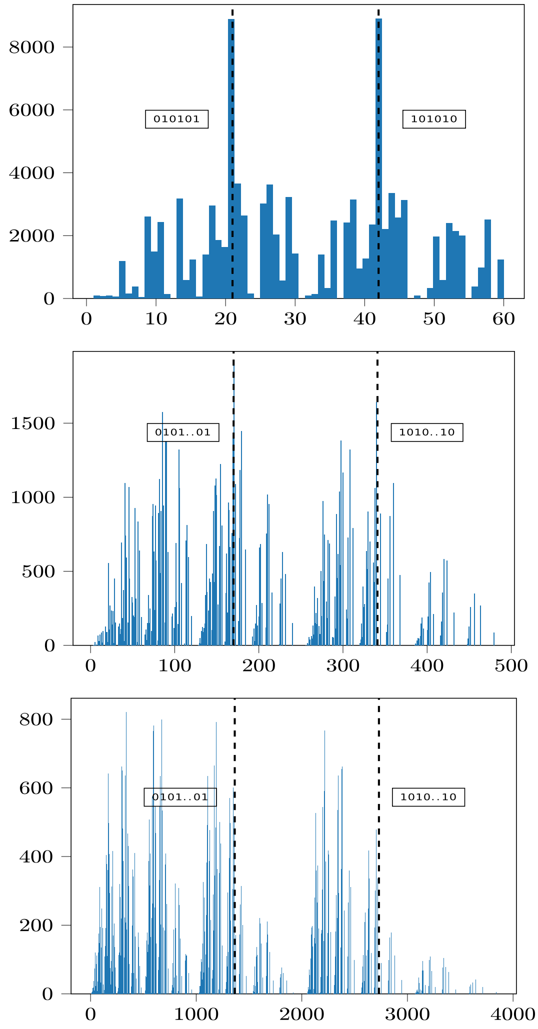


FIGURE 8. Histogram of optimal clustering solutions for a varying number of users: $K = 6$ (top), $K = 9$ (middle) and $K = 12$ (bottom).

the complexity of solving the k-th subproblem is $\mathcal{O}(SN_{\mathrm{tr}})$. Therefore, the total training complexity of MLTSVM reads $\mathcal{O}(KSN_{\mathrm{tr}})$. The test complexity reduces to computing the distance of each N_{f}-dimensional instance to each constructed hyperplane. Therefore, the test complexity of MLTSVM is $\mathcal{O}(KN_{\mathrm{f}}N_{\mathrm{tst}})$.

Next, let f_{train} and f_{test} denote the training and test complexities of LightGBM, respectively. According to [7], Classifier Chains have a computational complexity of $\mathcal{O}(K \cdot f_{\mathrm{train}})$ for training and $\mathcal{O}(K \cdot f_{\mathrm{test}})$ for testing. In those expressions, $B \ll (N_{\mathrm{f}} + K)$ stands for the number of bundles used by the Exclusive Feature Bundling mechanism of LightGBM that, as discussed in Section IV-B, is in charge of re-grouping mutually exclusive features of the $(N_{\mathrm{f}} + K)$-dimensional augmented feature space. Next, we need to determine the complexities of LightGBM in the training and test stages. Let L denote the number of leaf nodes, then the depth of the tree is on the order of $\log_2(L)$ levels. In the training phase, all the N_{tr} examples need to be assigned to one of the leaves and, to that aim, the optimal splitting variables (out of the B bundles) and splitting points must be found via exhaustive search. Hence, the complexity of constructing one tree is $\mathcal{O}(N_{\mathrm{tr}}B\log_2(L))$. Considering M trees, we have that $f_{\mathrm{train}} = \mathcal{O}(N_{\mathrm{tr}}BM\log_2(L))$ and, hence, the training complexity of CC-LightGBM yields $\mathcal{O}(KBM\log_2(L)N_{\mathrm{tr}})$. In the test phase, one just needs to determine the target leaf node for each example in the dataset. This, on average, entails $\log_2(L)$ comparisons against the optimal splitting variables and points identified in the training phase. Consequently, the test complexity is much lower. Specifically, we have that for a scenario with M weak learners, the computational complexity reads $\mathcal{O}(KM\log_2(L)N_{\mathrm{tst}})$. As for the two naïve approaches (random and '0101' strategy), they only have a test complexity and it can be easily shown to be $\mathcal{O}(N_{\mathrm{tst}})$. Finally, since the complexity of pre-sorting the users is negligible compared to training and test complexities, CC-LightGBM and CC-LightGBM without sorting have the same complexity.

We start by comparing the corresponding training complexities. To that aim, it should be noticed from [21] that the number of SOR iterations needed in MLTSVM is typically on the order of N_{tr}. Consequently, the training complexity of MLTSVM can be approximated by $\mathcal{O}(KN_{\mathrm{tr}}^2)$. This means that, unlike the ML-kNN and MLTSVM approaches, the training complexity of CC-LightGBM grows linearly in (not with the square of) the size of the training dataset. Note also that, even if in Fig. 6 the number of trees was relatively high ($M = 10^4$), it could be substantially reduced with a moderate performance penalty. Specifically, for $M = 100$ (and $K = 10$, $\lambda_r = 0.1$) the SRL in Fig. 4 is still below 4%. In other words, for $M = 100$ CC-LightGBM would still outperform the other two approaches. As for the test complexity, ML-KNN exhibits the lowest one, followed by MLTSVM and CC-LightGBM. The complexity of MLTSVM vs. CC-LightGBM will ultimately depend on the specific values of N_{f} and $M\log_2(L)$ (see expressions in Table 2). Nonetheless, for the parameter set we used in Section V-B (i.e., $N_{\mathrm{f}} = 80$ for $K = 10$, $N_{\mathrm{BS}} = 4$; and $M = 10^4$), the CPU time that CC-LightGBM takes is substantially lower than that of MLTSVM. This is mostly attributed to the fact that the basic operation performed by the decision trees of CC-LightGBM (comparison against splitting thresholds) is less computationally intensive than the computation of distances to hyperplanes which are required by MLTSVM. Last, the complexities exhibited by the naïve approaches are, unsurprisingly, the lowest ones. On the one hand, no training is involved. On the other, the prediction cost per test example is negligible and the overall cost scales linearly in the size of the test dataset.

VI. CONCLUSION

In this article, we have formulated user clustering in a MIMO NOMA setting as a multi-label classification problem. To solve it, we have adopted a transformation learning (data-driven) approach to avoid the exponential complexity of exhaustive search methods. Specifically, we have used a Gradient-Boosting Decision Tree (LightGBM) for each single-label classifier *coupled* with a Classifier Chain (CC) to account for label correlation stemming from user pre-sorting. Computer simulation results reveal that, for a number of base learners in the $10^4 - 10^5$ range, the sum-rate and Hamming losses of LightGBM can be minimized while avoiding a rapid increase of CPU time. The proposed CC-LightGBM scheme clearly outperforms all the benchmarks and naive strategies for the whole range of the number of active users. For $K = 12$ users in particular, its sum-rate loss is one order of magnitude/15 times lower than that of the ML-kNN/ML-TSVM algorithms, respectively. Moreover, the gain with respect to random user clustering (and the lazy '1010' strategy for large K) is tremendous. With respect to the case without user pre-sorting, the CC mechanism achieves a 7% extra gain. From the computational complexity analysis, we learnt that the training complexity of CC-LightGBM grows linearly in the size of the training dataset. As for the test phase, its complexity is potentially higher than that of ML-KNN and MLTSVM, yet this ultimately depend on the parameter set. However, the actual CPU time is also affected by the fact that the basic operation in CC-LightGBM is less computationally intensive than those of the benchamrks.

ACKNOWLEDGMENT

This work was done when Chaouki Ben Issaid was a Ph.D. student at KAUST and a research intern at CTTC.

REFERENCES

[1] Y. Saito, Y. Kishiyama, A. Benjebbour, T. Nakamura, A. Li, and K. Higuchi, "Non-orthogonal multiple access (NOMA) for cellular future radio access," in *Proc. IEEE 77th Veh. Technol. Conf. (VTC Spring)*, Jun. 2013, pp. 1–5.

[2] S. Verdu, *Multiuser Detection*. Cambridge, U.K.: Cambridge Univ. Press, 1998.

[3] X. Hu, C. Zhong, X. Chen, W. Xu, and Z. Zhang, "Cluster grouping and power control for angle-domain mmWave MIMO NOMA systems," *IEEE J. Sel. Topics Signal Process.*, vol. 13, no. 5, pp. 1167–1180, Sep. 2019.

[4] A. Celik, M.-C. Tsai, R. M. Radaydeh, F. S. Al-Qahtani, and M.-S. Alouini, "Distributed user clustering and resource allocation for imperfect NOMA in heterogeneous networks," *IEEE Trans. Commun.*, vol. 67, no. 10, pp. 7211–7227, Oct. 2019.

[5] N. Ye, X. Li, H. Yu, A. Wang, W. Liu, and X. Hou, "Deep learning aided grant-free NOMA toward reliable low-latency access in tactile Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 15, no. 5, pp. 2995–3005, May 2019.

[6] J. Cui, Z. Ding, P. Fan, and N. Al-Dhahir, "Unsupervised machine learning-based user clustering in millimeter-wave-NOMA systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 11, pp. 7425–7440, Nov. 2018.

[7] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1819–1837, Aug. 2014.

[8] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Berlin, Germany: Springer, 2009, pp. 254–269.

[9] J. Read and J. Hollmén, "Multi-label classification using labels as hidden nodes," 2015, *arXiv:1503.09022*. [Online]. Available: http://arxiv.org/abs/1503.09022

[10] R. E. Schapire and Y. Singer, "BoosTexter: A boosting-based system for text categorization," *Mach. Learn.*, vol. 39, nos. 2–3, pp. 135–168, May 2000.

[11] A. Dejonghe, C. Antón-Haro, X. Mestre, L. Cardoso, and C. Goursaud, "Deep learning-based user clustering for MIMO-NOMA networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (IEEE WCNC)*, Nanjing, China, Mar. 2021.

[12] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001.

[13] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3146–3154.

[14] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognit.*, vol. 40, no. 7, pp. 2038–2048, Jul. 2007.

[15] W.-J. Chen, Y.-H. Shao, C.-N. Li, and N.-Y. Deng, "MLTSVM: A novel twin support vector machine to multi-label learning," *Pattern Recognit.*, vol. 52, pp. 61–74, Apr. 2016.

[16] S. Park, A. Ali, N. Gonzalez-Prelcic, and R. W. Heath, Jr., "Spatial channel covariance estimation for the hybrid architecture at a base station: A tensor-decomposition-based approach," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Nov. 2018, pp. 1008–1012.

[17] B. Sklar, "Rayleigh fading channels in mobile digital communication systems. I. Characterization," *IEEE Commun. Mag.*, vol. 35, no. 7, pp. 90–100, Jul. 1997.

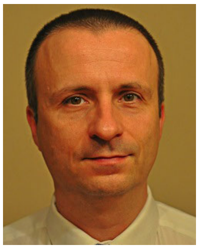
[18] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[19] H. L. Van Trees, *Optimum Array Processing: Part IV of Detection, Estimation, and Modulation Theory*. Hoboken, NJ, USA: Wiley, 2002.

[20] H. Shi, "Best-first decision tree learning," Ph.D. dissertation, Univ. Waikato, Hamilton, New Zealand, 2007.

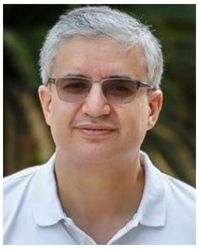
[21] Z. Qi, Y. Tian, and Y. Shi, "Successive overrelaxation for Laplacian support vector machine," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 4, pp. 674–683, Apr. 2015.

CHAOUKI BEN ISSAID (Associate Member, IEEE) received the Diplôme d'Ingénieur in economics and financial engineering from the Ecole Polytechnique de Tunisie (EPT), in 2013, and the master's degree in applied mathematics and computational science (AMCS) and the Ph.D. degree in statistics from KAUST, in 2015 and 2019, respectively. He is currently a Postdoctoral Fellow with the Centre for Wireless Communications, University of Oulu. His current research interest includes communication-efficient distributed machine learning.



CARLES ANTÓN-HARO (Senior Member, IEEE) received the Ph.D. degree in telecommunications engineering from the Technical University of Catalonia (UPC). In 1999, he joined Ericsson, Spain, where he participated in rollout projects of 2G and 3G mobile networks. He is currently a Senior Researcher and the Director of R&D Programs and also with the Centre Tecnòlogic de Telecomunicacions de Catalunya (CTTC). He has published more than 110 technical articles in IEEE journals and conferences (2 best paper awards) and books/book chapters. He has supervised 6 Ph.D. Theses (1 in progress). As a Senior Researcher, his research interests include signal processing and estimation theory for communications, this including machine learning, sensor and IoT networks, M2M communications, array signal processing, MIMO, energy harvesting, and smart grids. In recent years, he has been actively involved in the organization of major conferences such as the IEEE Wireless Communications and Networking Conference 2018 as a General Chair, the Workshop on Integrating Communications, Control, and Computing Technologies for Smart Grid @ ICC17 (TPC Chair), or the European Signal Processing Conference 2011 as a General Vice-Chair. He is an Associate Editor of the *EURASIP Journal on Wireless Communications and Networks* (JWCN).

XAVIER MESTRE received the M.S. and Ph.D. degrees in electrical engineering from the Technical University of Catalonia (UPC), in 1997 and 2002, respectively, and the Licenciate degree in mathematics in 2011. From January 1998 to December 2002, he was with the Communications Signal Processing Group, UPC, where he was also a Research Assistant. In January 2003, he joined the Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), where he is currently a Senior Research Associate in radio communications. During this time, he has actively participated in multiple European projects and several contracts with the local industry. He is also the Head of the Advanced Signal and Information Processing Department. He was an Elected Member of the IEEE Sensor Array and Multichannel Signal Processing Technical Committee. During the pursuit of his Ph.D., he was recipient of a 1998–2001 Ph.D. scholarship (granted by the Catalan Government) and was awarded the 2002 Rosina Ribalta second prize for the best doctoral thesis project within areas of information technologies and communications by the Epson Iberica Foundation. From 2007 to 2011 and since 2015, he has been an Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING.



MOHAMED-SLIM ALOUINI (Fellow, IEEE) was born in Tunis, Tunisia. He received the Ph.D. degree in electrical engineering from the California Institute of Technology (Caltech), Pasadena, CA, USA, in 1998. He served as a Faculty Member for the University of Minnesota, Minneapolis, MN, USA, and also for Texas A&M University at Qatar, Education City, Doha, Qatar. In 2009, he joined as a Professor of electrical engineering with the King Abdullah University of Science and Technology (KAUST), Thuwal, Makkah Province, Saudi Arabia. His current research interest includes the modeling, design, and performance analysis of wireless communication systems.