Journal of Optimization Theory and Applications (2024) 203:1794–1819
https://doi.org/10.1007/s10957-024-02391-9

On Optimizing Ensemble Models using Column Generation

Vanya Aziz[1] · Ouyang Wu[1] · Ivo Nowak[1] · Eligius M. T. Hendrix[2] · Jan Kronqvist[3]

Received: 4 August 2023 / Accepted: 11 January 2024 / Published online: 22 February 2024

Abstract
In recent years, an interest appeared in integrating various optimization algorithms in machine learning. We study the potential of ensemble learning in classification tasks and how to efficiently decompose the underlying optimization problem. Ensemble learning has become popular for machine learning applications and it is particularly interesting from an optimization perspective due to its resemblance to column generation. The challenge for learning is not only to obtain a good fit for the training data set, but also good generalization, such that the classifier is generally applicable. Deep networks have the drawback that they require a lot of computational effort to get to an accurate classification. Ensemble learning can combine various weak learners, which individually require less computational time. We consider binary classification problems studying a three-phase algorithm. After initializing a set of base learners refined by a bootstrapping approach, base learners are generated using the solution of an linear programming (LP) master problem and then solving a machine learning sub-problem regarding a reduced data set, which can be viewed as a so-called pricing problem. We theoretically show that the algorithm computes an optimal ensemble model in

Communicated by Paula Amaral.

✉ Vanya Aziz
vanya.aziz@haw-hamburg.de

Ouyang Wu
Ouyang.Wu@haw-hamburg.de

Ivo Nowak
Ivo.Nowak@haw-hamburg.de

Eligius M. T. Hendrix
eligius@uma.es

Jan Kronqvist
jankr@kth.se

1 Maschinenbau und Produktion Department, HAW Hamburg, Hamburg, Germany

2 Universidad de Málaga, Málaga, Spain

3 KTH Royal Institute of Technology, Stockholm, Sweden

the convex hull of a given model space. The implementation of the algorithm is part of an ensemble learning framework called DECOLEARN. Numerical experiments with CIFAR-10 data set show that the base learners are diverse and that both the training and generalization error are reduced after several iterations.

Keywords Machine learning · Ensemble · Linear programming · Column generation · Pricing problem

1 Introduction

In recent years, an increasing interest has appeared among researchers in optimization to apply various "classic" optimization methods to machine learning. This is for instance visible in a weekly seminar called "Machine Learning NeEDS Mathematical Optimization" promoted by European project "NeEDS MSCA RISE 2018 (H2020 Project GA No 822214)". Recently, the authors have been interested in exploring the concept of ensemble learning, that is popular in the community of machine learning (ML) [8, 14, 21, 27], from an optimization perspective due to its similarity with column generation (CG) from the optimization community. Ensemble learning methods combine multiple individual models to enhance generalization performance of machine learning models, which have shown a remarkable potential. There are plenty of real-world applications of ensemble learning such as object detection, object recognition [1, 16, 17, 26], fault detection and diagnosis [11, 15, 25], semantic segmentation, edge detection [2, 10, 11], and for black box optimization [23].

It is known that deep learning networks may require an enormous computational effort to reach improvement in classification tasks. For example, state-of-the-art AI models can have more than 100 billion parameters [4]. From that point of view, combining easier or smaller learners may be an attractive approach. Ensemble learning has an interesting similarity with the concept of Column Generation (CG), see [7, 27]. The latter computes solution estimates by combining interesting partial solution candidates (options) in master optimization problems, which provide information to generate more interesting options (columns). Efficient CG algorithms for solving nonlinear optimization problems, like in [18], are based on fast generation of initial columns in a starting phase and solving low-dimensional (pricing) sub-problems for generating columns. Ensemble models typically combine simpler base learners to obtain a more accurate and complex model. [5] shows that the generalization error of an ensemble model strongly depends on the so-called diversity, see Sect. 2.4. Our interest is in the question whether we can generate base learners, similar to CG, based on a sub-set of the data and in that way promote diversity and general applicability of the ensemble model. At the same time, it reduces the computational burden of taking the complete training set into account to generate new base learners.

The main research question of this paper is how to efficiently decompose the optimization problem of training an ensemble model for classification problems and understanding the effect of dynamic data selection. One of the main goals of this work has been to develop a decomposition framework that splits the full optimization problem into smaller sub-problems with optimality and convergence guaranteed. This

paper shows that such a decomposition framework is possible for binary classification, and we hope this work brings us a step closer to being able to use global optimization techniques for determining globally optimal ensemble models with a guaranteed accuracy.

To show this, we design a specific ensemble learning algorithm which improves an initial model of unspecified complexity by generating and combining new base learners into an ensemble following a boosting strategy. The base learners and the initial model are combined into an ensemble by solving a linear programming (LP) master problem, which maximizes the soft margin of the ensemble classifier. The resulting dual solutions of the master problem are further used in sub-problems, which train new base learners to boost the previous ensemble. We consider several types of sub-problems for training base learners during sequential phases in the algorithm. To illustrate the efficiency of the ensemble algorithm, we measure the diversity and the generalization error on several test instances. The main contribution of this paper is summarized as follows:

- We develop and investigate a new method for (binary) classification tasks that has the following properties:
  - Enabling a dual decomposition that splits the classification problem into sub-problems considering small base learners and a sub-set of the data points. In this way, the learning task is decomposed into simpler sub-problems with fewer variables.
  - It forms a framework for decomposing certain classification problems that is globally convergent if sub-problems are solved to global optimality.
  - The data sub-sets defining the sub-problems are computed using sparse dual solutions of the so-called LPBoost master problem, see [7]. The choice of the number of data points in the set defines a trade-off between accuracy and computational complexity.
  - Sub-problems are defined as learning problems with either linear or nonlinear loss, instead of using a dual feasibility criterion for selecting base learners in the traditional LPBoost approach of [7].
  - The base learners generated by the method are diverse.
- We show the method can reliably improve an existing (ensemble) classifier.
- By analyzing the validation data, we found that the generalization error decreases as diversity increases, thus supporting the use of model diversity as a performance measure.
- We show that the algorithm computes an optimal ensemble model in the convex hull of a given model space. The results show that the accuracy of the ensemble model has the potential to reach an accuracy which is better than that of the base learners. One should keep in mind that the ensemble convex hull represents a richer model space in the sense that the used base learner model space is a proper sub-set of it. A generic theoretical result on the ensemble model space is presented.

The rest of this paper is structured as follows. Section 2 focuses on LPBoost-based ensemble learning and the mechanisms involved in the algorithm, namely data selection, diversity and ensemble learning. Section 3 describes the algorithm in a modular

fashion; first, it provides an overview over the general scheme and then describes the core features of the algorithm. Section 4 presents experiments with standard LPBoost and the new variant for two data sets with varying difficulties. Section 5 summarizes our findings.

2 Diverse Ensemble Learning and LPBoost

We first provide background on the problem formulation, LPBoost, and some fundamental theory which we build our algorithm upon. We first sketch the problem formulation and the ensemble learning concept in Sect. 2.1. Then, we describe the LP master problem of LPBoost and the so-called pricing problem in Sect. 2.2. We give an intuition for the potential of only considering a sub-set of data points in the generation of base learners by exploiting the sparsity in Sect. 2.3. Section 2.4 describes the role of model diversity based on considerations from literature.

2.1 Problem Formulation and Ensemble Learning (EL)

A general learning task is to learn a function $f : \mathbb{R}^\ell \mapsto \mathbb{R}^m$ to approximate an unknown mapping $\Psi : \mathcal{X} \mapsto \mathcal{Y}$ between the input space $\mathcal{X}$ and output space $\mathcal{Y}$. Typically, the sets $\mathcal{X}$ and $\mathcal{Y}$ are not known and we only have a finite set of matching observations $\left\{(\mathbf{x}_j, \mathbf{y}_j)\right\}_{j\in J} \subset \mathcal{X} \times \mathcal{Y}$, and J is its index set. We consider binary classification problems in this paper, that is, $\mathcal{Y} = \{-1, 1\}$. For one single arbitrary observation (x, y), y denotes the target of this observation with $y \in \{-1, 1\}$. The binary classifier $f : \mathbb{R}^\ell \mapsto \{-1, 1\}$ as $f(x) = \text{sign}(\hat{f}(x))$ is defined by the score function $\hat{f} : \mathbb{R}^\ell \mapsto \mathbb{R}$, $\hat{f} \in \mathcal{F}$. Given a model space $\mathcal{F}$, the learning task aims at obtaining a predictive score model $\hat{f}^* \in \mathcal{F}$ which minimizes the expectation of a loss function $\mathcal{L}$

$$\hat{f}^* = \text{argmin} \left\{ \mathbb{E}[\mathcal{L}(\hat{f}(x), y)] : \hat{f} \in \mathcal{F} \right\}, \tag{1}$$

where $\mathbb{E}$ denotes the expectation regarding a joint probability distribution $P(x, y)$ over $\mathcal{X} \times \mathcal{Y}$. As we are typically restricted to a finite training set $T = \left\{(\mathbf{x}_j, \mathbf{y}_j)\right\}_{j\in J}$, the learning task is usually approximated by

$$\hat{f}^* = \text{argmin} \left\{ \sum_{j\in J} \mathcal{L}(\hat{f}(x_j), y_j) : \hat{f} \in \mathcal{F} \right\}. \tag{2}$$

Throughout the paper, we assume that the model space $\mathcal{F}$ is limited to a predefined ML architecture, *e.g.,* a deep Neural Network (DNN) with a specific number of nodes and layers. In binary classification, a loss function could be for instance a linear classification error (20) or a nonlinear binary cross-entropy-based loss (19).

The basic idea of EL for binary classification is to construct a score function $\hat{f}$ by combining a collection of weak base learners (BL), [19]. An ensemble is typically called deep, if the BL are deep learning models [14].

Given a finite set of base learners $\mathcal{H} = \{h_i : \mathbb{R}^\ell \mapsto \mathbb{R} \mid i = 1, \ldots, n\}$, the ensemble model prediction is given by

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i h_i(\mathbf{x}), \tag{3}$$

where α_i are weights of the individual BL. In ensemble learning, the weights α are usually restricted to a standard simplex:

$$\Delta_n = \left\{ \alpha \in \mathbb{R}^n \;\middle|\; \sum_{i=1}^{n} \alpha_i = 1,\ \alpha_i \geq 0,\ i = 1, \ldots, n \right\}. \tag{4}$$

An important property and motivation of ensemble learning is that the final ensemble model is not restricted to the model space of the base learners, *i.e.,* the combination of base learners spans a larger model space. Thus, the base learners can be restricted to a far simpler model space than the final ensemble model. This property is further described in the following proposition:

Proposition 1 *Given a set of base learners $h_1, \ldots, h_n \in \mathcal{F}$, the ensemble model $\hat{f} := \sum_{i=1}^{n} \alpha_i h_i$ lies within the convex hull of $\mathcal{F}$, but it does not necessarily belong to the model space $\mathcal{F}$.*

Proof Let $\mathcal{F}$ be defined as the model space of neural networks with a single ReLU node, i.e., piecewise linear functions with one constant and one affine part. Consider a convex combination f of two base learners from $\mathcal{F}$. Then, f is an ensemble model that is a piecewise linear function with one constant part and two affine parts and hence not an element of $\mathcal{F}$. □

Considering a training set $T := \{(\mathbf{x}_j, y_j)\}_{j \in J}$, the learning task in ensemble learning is two-fold: learn the ensemble weights α and construct the set of base learners $\mathcal{H}$. To make the learning task tractable, the base learners h_i are typically restricted to a certain model structure, *e.g.,* tree models of specific structure. Consider for instance deep neural networks with a given architecture. The base learners h_i are thus parameterized by the weights and biases of the Deep Neural Network (DNN). To simplify the notation, we define

$$h_i(\mathbf{x}) := \varphi(\mathbf{W}_i, \mathbf{x}), \tag{5}$$

where φ denotes the input–output mapping of a DNN with weights and biases given by a parameter vector $\mathbf{W}_i \in \mathbb{R}^k$ of k parameters.. We are assuming that all the weights and biases are bounded, *i.e.,* $\|\mathbf{W}_i\|_1 \leq M < \infty$. The space of base learners is then reduced to the Hilbert space $\mathcal{F} = \{\varphi(\mathbf{W}, \mathbf{x}) \mid \mathbf{W} \in [-M,\ M]^k\}$. Furthermore, by limiting the number of BL in the ensemble model to n, the ensemble learning problem can be written as

$$\begin{aligned} \min_{\alpha, h_1, \ldots, h_n} \quad & \sum_{j \in J} \mathcal{L}\left(\sum_{i=1}^{n} \alpha_i h_i(\mathbf{x}_j),\, y_j \right) \\ \text{s.t.} \quad & \alpha \in \Delta_n, \\ & h_i \in \mathcal{F}, \quad i = 1, \ldots, n. \end{aligned} \tag{6}$$

The minimizer of (6) is an optimal ensemble model for the given training set T, loss function $\mathcal{L}$, base learner architecture, and number of base learners. By Proposition 1 , we know that the ensemble model given by the minimizer of problem (6) is within the convex hull of $\mathcal{F}$, denoted $\text{conv}(\mathcal{F})$. With an additional assumption on the number of base learners we get a more general optimality property for the ensemble model given by problem (6), as described in the following proposition.

Proposition 2 *Given that number of base learners in problem (6) is sufficiently large, then the resulting ensemble model will also be optimal within* $\text{conv}(\mathcal{F})$ *for the given objective function.*

Proof Follows from Carathéodory's theorem. □

The importance of Propositions 1 and 2 is that they show that the ensemble model obtained by solving (6) is optimal in a more general, and complex, model space than the model space of the base learners. This is one of the main motivations to design an algorithm that iteratively optimizes over the simpler base learners to solve problem (6) and obtain a strong ensemble model.

Each base learner h_i in problem (6) is parameterized by k parameters. The complete problem contains $k \times n + n$ variables. However, in practice, one might need to use a large number of auxiliary variables to represent the states of the activation functions for the base learners for each data point and thus greatly increasing the number of variables in the optimization problem. Ignoring the large number of variables, problem (6) is still non-trivial due to the non-convexity described in the following proposition.

Proposition 3 *Unless $\mathcal{F}$ is a singleton set, problem (6) is in general non-convex.*

Proof With $\mathcal{F}$ not restricted to a singleton set, the terms $\alpha_i h_i(\mathbf{x}_j)$ in the objective function results in bilinear terms as the function h_i are parameterized by the weights and biases. □

Considering that non-trivial classification problems typically requires DNNs with thousands to millions of parameters, problem (6) still remains well out of reach of today's general purpose deterministic global optimization solvers. To solve problem (6), we present an algorithm inspired by the LPBoost algorithm [7] that follows a column generation approach. By this decomposition approach, we are able to determine one base learner, *i.e.,* column, at a time and thus limit each sub-problem to a smaller sub-set of variables. The presented algorithm also allows us to only consider a sub-set of the training data when determining a new base learner, which further simplifies each sub-problem.

2.2 LPBoost and its LP Master Problem

To determine an optimal convex combination of a given finite set of fixed base learners, the LPBoost algorithm of [7] considers a LP optimization problem which maximizes the soft margin of the combined base learners in a boosting style. They consider a given set of base learners $\mathcal{H} = \{h_i | \, i = 1, \ldots, n\}$. The LP model determines an ensemble model $f(x) = \sum_{i=1}^{n} \alpha_i h_i(x)$ that maximizes the soft margin of training data set $\{X, Y\}$:

$$\begin{aligned} \max_{\alpha,\xi,\rho} \quad & \rho - \lambda \sum_{j \in J} \xi_j \\ \text{s.t.} \quad & y_j \sum_{i=1}^{n} \alpha_i h_i(\mathbf{x}_j) + \xi_j \geq \rho, \quad j \in J \\ & \alpha \in \Delta_n \\ & \xi_j \geq 0, \qquad\qquad j \in J \\ & \rho \in \mathbb{R} \end{aligned} \tag{7}$$

where $y_j \sum_{i=1}^{n} \alpha_i h_i(\mathbf{x}_j)$ is the margin of data point (x_j, y_j) for a finite set of base learners. Variable ξ_j is the margin slack variable of $(x_j, y_j) \in \mathcal{X} \times \{-1, 1\}$ and λ a positive penalty parameter, *e.g.*, $\lambda = 0.1$ which influences the size of the nonzero dual variables. A dual problem of (7) is given by

$$\begin{aligned} \min_{\mathbf{u},\beta} \quad & \beta \\ \text{s.t.} \quad & \sum_{j \in J} u_j y_j h(\mathbf{x}_j) \leq \beta, \quad h \in \mathcal{H} \\ & \sum_{j \in J} u_j = 1 \\ & 0 \leq u_j \leq \lambda \qquad\quad j \in J. \end{aligned} \tag{8}$$

In this formulation, u represents a dual solution of master problem (7), where u_j represents the so-called misclassification cost assigned to data point (x_j, y_j).

Following the column generation approach in [7], a new base learner $\hat{h}$ can be generated or selected by solving the following pricing problem, which in the machine learning context represents the so-called BL generation sub-problem

$$\hat{h} = \arg\max_{h \in \mathcal{N}} \sum_{j \in J} u_j y_j h(x_j), \tag{9}$$

where $\mathcal{N} \subset \mathcal{F}$ is a large finite set of base learners and u is the information of the master problem in terms of a dual solution of (7). Using column generation to solve problem (7) gives a computationally efficient framework for dealing with a potentially massive set of base learners $\mathcal{N}$ by only working with a smaller set of base learners $\mathcal{H}$. The

objective function value of (9) can be used as a termination criterion of the boosting method, see [7]. Note that the method in [7] is limited to a large set of predefined base learners. Our approach extends this framework by also considering the generation of new base learners. This is done by solving the less restrictive pricing problem, similar to the approach in [3]

$$\hat{h} = \arg\max_{h \in \mathcal{F}} \sum_{j \in J} u_j y_j h(x_j). \tag{10}$$

Hence, this method is more general and has the potential of producing better solutions since we are not limited to a finite set of initial base learners.

2.3 Sparse Dual Solution, Dynamic Data Selection and BL Generation

Numerical experiments show that the dual master problem (8) solution u is typically a sparse vector, with most of its components taking values of either 0 or λ. In linear programming, a sparse vector means that most of its entries are equal to 0. Let

$$R := \{(x_j, y_j) \in T : u_j > 0\} \tag{11}$$

be the set of data points with a nonzero dual value, which we refer to as active data points. Note that a dual value of $u_j = 0$ indicates that the associated data point is well-classified with a sufficient margin, as shown in [7]. Since $u \in \Delta_{|J|}$ and $u_j = \lambda$ if $u_j > 0$, the number of nonzero components of u is $|R| = \frac{1}{\lambda}$ for $\frac{1}{\lambda} \in \mathbb{N}$. The relative number of active data points is $\nu := \frac{1}{\lambda |T|}$, such that $|R| = \nu |T|$. Using ν as a parameter to control the number of active points, we can set $\lambda = \frac{1}{\nu T}$ in (8) to influence the sparsity of the dual solution that allows us to identify critical data points and discard the irrelevant data points.

For the non-active data points in $T \setminus R$, the objective function contribution coefficient $u_j = 0$ in BL generation sub-problem (10) is zero, such that the pricing problem (10) is equivalent to

$$\hat{h} = \arg\max_{h \in \mathcal{F}} \sum_{(x_j, y_j) \in R} u_j y_j h(x_j). \tag{12}$$

This means that we can limit the generation of new learners to the active data points in R, which tend to be misclassified or well-classified with insufficient margin. Numerical experiments showed that, in practice, it is more effective to use a nonlinear objective function in the BL generation problem

$$\hat{h} = \arg\max_{h \in \mathcal{F}} \sum_{(x, y) \in R} \mathcal{L}(h(x), y) \tag{13}$$

rather than the traditional linear objective in (12). Keep in mind, that in the numerical experiments, sub-problems are not necessarily solved to global optimality and the

nonlinear objective might work better interacting with the used deep learning tools. Note that since data set R is changing in each iteration, the related base learners tend to be more diverse which is an additional advantage of working with a smaller set of active points. Section 2.4 summarizes some results of [5], showing that the generalization error of an ensemble model is related to the diversity of its base learners.

2.4 Diversity Measurement for Ensemble

An interesting approach for analyzing an ensemble model is to compute the diversity of its base learners. [5] shows that the generalization error of an ensemble depends mainly on the diversity. We will also use the diversity concept for analyzing the investigated ensemble models. In the following, we shortly describe this approach. Consider an error function for binary classification based on $\text{margin}(f(x), y) = yf(x)$ defined as

$$\text{Err}(f(x), y) := \begin{cases} 1 & \text{if } \text{margin}(f(x), y) = -1 \\ 0.5 & \text{if } \text{margin}(f(x), y) = 0 \\ 0 & \text{if } \text{margin}(f(x), y) = 1. \end{cases}$$

Hence, $\text{Err}(f(x), y) = -\frac{1}{2}(yf(x) - 1)$. The generalization error of f is defined by

$$G := \mathbb{E}[\text{Err}(f(x), y)].$$

The diversity of f at $x \in \mathcal{X}$ is defined by the difference between ensemble error and the weighted average error of individual classifiers (base learners):

$$\begin{aligned} \text{div}(f(x), y) :=& \text{Err}(f(x), y) - \sum_{i \in I} \alpha_i \text{Err}(f_i(x), y) \\ =& \frac{1}{2}\text{margin}(f(x), y) - \frac{1}{2}\sum_{i \in I} \alpha_i \text{margin}(f_i(x), y). \end{aligned} \tag{14}$$

The relation between generalization error and diversity according to [5] is

$$G = \frac{1}{2}\left(1 - \mathbb{E}[\text{sign}(\gamma(x))]\right),$$

where $\gamma(x) := \text{sign}(\text{div}(f(x), x)) - 2\text{div}(f(x), x))$. This shows that increasing the diversity at points where $\gamma(x)$ is small will decrease G. In particular, the defined diversity measure and G varies in different ranges of diversity

- When $\text{div}(f(x), y)$ is negative, the ensemble classifier has misclassified instance x. Increasing diversity will improve the ensemble performance.
- When $\text{div}(f(x), y)$ is positive, diversity needs to be decreased to increase the generalization performance of the ensemble.

More details of derivations for diversity and generalization errors can be found in [5]. Note that the ensemble model in [5] combines the classifier output of base learners

$f_i(x) \in \{-1, 1\}$ instead of score function $h_i(x)$, which is different from the ensemble described in Sect. 2. We show in Sect. 4 experimentally that for base learners obtained from solving sub-problem (13), the generalization error is decreasing and the average diversity is increasing iteratively.

3 A Dynamic Data-Reduction Ensemble Learning Algorithm (DDA)

We present an ensemble learning algorithm, which iteratively generates base learners $h_i(x) \in [-1, 1]$. The algorithm defines low-dimensional BL generation sub-problems using dynamic data selection. The sub-problems can be solved using an arbitrary method. The initialization procedure to generate an initial set of base learners speeds up the calculations. The main algorithm solves a sequence of master problems (8) to determine convex combinations of base learners and sub-problems (either (12) or (13)) to determine new base learners. We show in Sect. 3.2 that if the BL generation sub-problems, i.e., pricing problems, are solved exactly, then the algorithm converges to an optimal solution of problem (6), which is equivalent to problem (16).

3.1 Algorithm

Algorithm 1 Main Algorithm, Dynamic Data-Reduction Ensemble Learning

Input: Training data $T = X \times Y \subset \mathcal{X} \times \{-1, 1\}$
Parameters: BL configuration and data reduction value $\nu \in (0, 1)$ defining $\lambda = \frac{1}{\nu|T|}$
Output: $f(\cdot)$

function DDA
  $h \leftarrow$ SOLVEINITIAL(T) `# initial phase`
  $\mathcal{H} \leftarrow \{h\}$, $R \leftarrow \emptyset$
  repeat `# generation phase (optional)`
    $\hat{R} = R$
    $S \leftarrow$ BOOTSTRAPDATA(h, T)
    $h \leftarrow$SOLVEBTSUBPROBLEM(S)
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{h\}$
    $(\hat{f}, u) \leftarrow$ SOLVEMPANDUPDATEENS($\mathcal{H}, T$)
    $R \leftarrow$ ACTIVEDATA(T, u)
  until $|R \cap \hat{R}| > \sigma|R|$
  repeat `# refinement phase`
    $R \leftarrow$ ACTIVEDATA(T, u)
    $h \leftarrow$SOLVEREFSUBPROBLEM(R)
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{h\}$
    $(\hat{f}, u) \leftarrow$ SOLVEMPANDUPDATEENS($\mathcal{H}, T$)
  until stopping criterion
  return sign($\hat{f}$)

Algorithm 1 describes the new ensemble learning algorithm. It consist of three phases, named initial, generation and refinement phase, respectively. The initial phase generates a base-line model h using procedure SOLVEINITIAL by fitting the training set, such that h initiates the base learner set $\mathcal{H}$. In the generation and refinement phases,

diverse base learners are generated and added to base learner set $\mathcal{H}$. After adding a new base learner, the primal and dual solution (α, u) of LP master problem (7) is used to update ensemble model $\hat{f}$ by calling function SOLVEMPANDUPDATEENS. Using the optional generation phase in combination with the refinement phase often leads to better end results than only using the refinement phase. The refinement phase also extends $\mathcal{H}$, but the base learners generated in this phase are fitted on a sub-set of active training data determined by u.

In order to accelerate the CG procedure, in the beginning, columns are generated using a start heuristic without solving the master problem, since a master problem defined by few columns may give a poor search direction, see [18]. We use a similar approach in our ensemble algorithm, which we call the generation phase. It is based on so-called error-based bootstrapping ([24]) as described in Algorithm 2. Error-based bootstrapping generates a new data set S of size $|S| = |T|$. Unlike conventional bootstrapping that forms data set S by random sampling from the original data set with replacement, error-based bootstrapping adds to the set of misclassified data points $M \subset T$ a uniform sample of $T \setminus M$ with replacement, such that it focuses on misclassified instances from a base learner h. This procedure is denoted by function BOOTSTRAPDATA(h, T) in Line 6 of Algorithm 1.

Algorithm 2 Generates error-based bootstrapping data set S of size $|S| = |T|$

```
1: function BootstrapData(h, T)
2:     M ← MissClassifiedData(h, T)
3:     S ← Uniform sample of |T| − |M| data points from T
4:     S ← S ∪ M
5:     return S
```

The misclassified set is obtained according to the predictions of h instead of preliminary ensemble $\hat{f}$. If instead the predictions of $\hat{f}$ are used, *i.e.,* BOOTSTRAPDATA$(\hat{f}, T)$ is called (see Algorithm 2), the generation phase might not terminate quickly and transition to the subsequent refinement phase takes longer. The set of misclassified data points is updated in each iteration. In Lines 7 and 8 of Algorithm 1, a new base learner $h_i(x)$ is fitted on S by SOLVEBTSUBPROBLEM(S) and added to set $\mathcal{H}$. SOLVEMPANDUPDATEENS($\mathcal{H}$, T) in Line 9 of Algorithm 1 constructs a new master problem (8) given the updated base learner inferences and true labels. The master problem is solved by an LP solver, and it returns the primal and dual solution. The ensemble score function $\hat{f}$ is obtained via the primal solution which contains the base learner weight vector.

The details of SOLVEMPANDUPDATEENS can be found in Algorithm 3. The sparse dual solution u is used to determine the set of active data (11). Function ACTIVEDATA(T, u) in Line 10 of Algorithm 1 selects active data from T fulfilling (11) and returns set R.

The generation phase stops if the difference between former active data set $\hat{R}$ and the current active set R is not significant. The stopping criterion parameter σ determines the minimum change in the active data set to continue the iterations in the generation phase. It is recommended to set its value to at least 0.2, which means the active set

Algorithm 3 Primal/dual solution of master problem and update ensemble

1: **function** SOLVEMPANDUPDATEENS($\mathcal{H}$, T)
2: $(\alpha, u) \leftarrow$ PRIMALDUALSOLVEMP($\mathcal{H}$, T, λ)
3: $\hat{f} = \sum_{h_i \in \mathcal{H}} \alpha_i h_i$
4: **return** $(\hat{f}, u)$

R in the final iteration has changed 20% in the last iteration. Note that a small value of σ can result in the generation of many base learners that do not contribute to the ensemble.

The refinement phase of Algorithm 1 adds base learners $h_i(x)$ to $\mathcal{H}$ solving the LP master problem based on a reduced set of data points based on the highest dual values. ACTIVEDATA() is called to determine the training set R used by SOLVEREFSUBPROBLEM() solving (13) or (12), depending on whether a linear or nonlinear objective is used, to generate base learners. This means that the generated base learners focus on a smaller training set in the refinement. Parameter ν is used in Algorithm 3 to calculate λ in (7) adjusting the sparsity of the dual solution to provide a limit on the number of considered data points. We discuss the choice of its value in Sect. 4. The termination criterion of the refinement phase in Algorithm 1 can be based on the dual feasibility criterion from the original LPBoost algorithm, see [7]. If the following condition is met, the base learner can not further improve the ensemble:

$$\sum_{j \in J} u_j y_j h_i(x_j) \leq \beta + \epsilon, \tag{15}$$

where ϵ is a small positive value. There are two main advantages of the column generation approach in Algorithm 3 for solving (6). First, it decomposes the problem into simpler sub-problems compared to simultaneously optimizing all base learners and their combination. Secondly, it allows us to only consider a small sub-set of active data points which results in easier sub-problems for determining new base learners, *i.e.,* easier pricing problems.

3.2 Convergence

We analyze the convergence properties of Algorithm 1 and focus on the soft margin loss function, even though the algorithm is not limited to this specific loss function. We show that Algorithm 1 computes an optimal solution of the extended master problem

$$\begin{aligned} \max_{f,\xi,\rho} \quad & \rho - \lambda \sum_{j \in J} \xi_j \\ \text{s.t.} \quad & y_j f(\mathbf{x}_j) + \xi_j \geq \rho, && j \in J \\ & \xi_j \geq 0, && j \in J \\ & \rho \in \mathbb{R}, \quad f \in \text{conv}(\mathcal{F}). \end{aligned} \tag{16}$$

Note that problem (16) is equivalent to problem (6) when using the soft margin loss function and allowing sufficiently many base learners. The proof requires a few intermediate results that are presented in Propositions 4, 5 and 6.

Proposition 4 *Problem (16) is convex.*

Proof Consider the feasible set of (16) as

$$\omega := \{(f, \xi, \rho) : y_j f(\mathbf{x}_j) + \xi_j \geq \rho,\ j \in J,\ \rho \in \mathbb{R},\ f \in \text{conv}(\mathcal{F})\}.$$

Let $g_1 = (f_1, \xi_1, \rho_1)$, $g_2 = (f_2, \xi_2, \rho_2)$ and $g = (f, \xi, \rho) := tg_1 + (1-t)g_2$, $t \in [0, 1]$. From $g_1, g_2 \in \omega$, we have

$$\begin{aligned} y_j f(\mathbf{x}_j) + \xi_j &= y_j(tf_1(\mathbf{x}_j) + (1-t)f_2(\mathbf{x}_j)) + t\xi_{1j} + (1-t)\xi_{2j} \\ &\geq t\rho_1 + (1-t)\rho_2 = \rho. \end{aligned}$$

Since $f = tf_1 + (1-t)f_2 \in \text{conv}(\mathcal{F})$, $g \in \omega$, which proves ω is convex. □

Proposition 5 *Problem (16) is equivalent to the semi-infinite dual-LP:*

$$\begin{aligned} \min_{\mathbf{u},\beta} \quad & \beta \\ s.t. \quad & \sum_{j\in J} u_j y_j h(\mathbf{x}_j) \leq \beta, \quad h \in \mathcal{F} \\ & \sum_{j\in J} u_j = 1 \\ & 0 \leq u_j \leq \lambda, \qquad\qquad j \in J. \end{aligned} \tag{17}$$

Proof Consider the Lagrangian of (16)

$$c(f, \rho, \xi, u) := \rho - \lambda \sum_{j\in J} \xi_j + \sum_{j\in J} u_j(y_j f(\mathbf{x}_j) + \xi_j - \rho).$$

Then,

$$\begin{aligned} c^*(u) &:= \max\left\{c(f, \rho, \xi, u) : (f, \xi, \rho) \in \text{conv}(\mathcal{F}) \times \mathbb{R}_+^{|J|} \times \mathbb{R}\right\} \\ &= \max_{\rho\in\mathbb{R}}(1 - \sum_{j\in J} u_j)\rho + \max_{\xi\in\mathbb{R}_+^{|J|}} \sum_{j\in J}(u_j - \lambda)\xi_j + \max_{f\in\text{conv}(\mathcal{F})} \sum_{j\in J} u_j y_j f(\mathbf{x}_j) \\ &= \max_{f\in\text{conv}(\mathcal{F})} \sum_{j\in J} u_j y_j f(\mathbf{x}_j) = \max_{h\in\mathcal{F}} \sum_{j\in J} u_j y_j h(\mathbf{x}_j). \end{aligned} \tag{18}$$

Since problem (16) is convex, strong duality applies, *i.e.,* the optimal value of (16) is equal to

$$c^* := \min\left\{c^*(u) : u \in \mathbb{R}_+^{|J|}\right\} = \min\left\{\beta : \beta \geq c^*(u),\ u \in \mathbb{R}_+^{|J|}\right\}.$$

For dual points with $c^*(u) < \infty$, we have $\sum_{j\in J} u_j = 1$ and $0 \le u_j \le \lambda,\ j \in J$. From (18) follows

$$c^* = \min\left\{\beta : \beta \ge \max_{h\in\mathcal{F}} \sum_{j\in J} u_j y_j h(\mathbf{x}_j),\ \sum_{j\in J} u_j = 1,\ 0 \le u_j \le \lambda,\ j \in J\right\}.$$

Hence, the optimal value of (17) is c^*. □

Proposition 6 *If the loop in the refinement phase of Algorithm 1 is not stopped, and pricing problem (10) is solved exactly, the primal-dual solution* $(\hat{f}, u)$ *converges toward a primal-dual solution of extended master problem (16).*

Proof Let U denote the set of dual points u computed by Algorithm 1. Since U is bounded, there exists a subsequence $\{u\} \subset U$ converging to a point $\hat{u}$, where u is a dual solution of (8). Let $\mathcal{H}$ be the set of base learners related to the subsequence. Then, $\hat{u}$ solves (8) regarding $\mathcal{H}$. Let $\hat{h}$ denote the solution of pricing problem (10) at $\hat{u}$. Since $\hat{u}$ is a limit point and by continuity of the dual function, we have

$$\max_{h\in\mathcal{H}} \sum_{j\in J} \hat{u}_j y_j h(\mathbf{x}_j) = \sum_{j\in J} \hat{u}_j y_j \hat{h}(\mathbf{x}_j) = \lim_{u\to\hat{u}} \max_{h\in\mathcal{F}} \sum_{j\in J} u_j y_j h(\mathbf{x}_j).$$

Hence, we can replace $\mathcal{H}$ by $\mathcal{F}$ in (8), which shows that $\hat{u}$ is a solution of (17). Since (17) is the dual of (16), the proposition follows. □

The theoretical results show that Algorithm 1 can generate an optimal solution of problem (6). Keep in mind that this requires the BL generation sub-problem to be solved exactly. Even if the dimension of a sub-problem is much smaller than of the original problem, it may still not be feasible to solve the problem exactly. The results show that the algorithm follows a solid decomposition technique and if the sub-problems are solved approximately then the algorithm becomes more heuristic.

4 Numerical Experiments

We study the behavior of Algorithm 1 on a set of binary classification problems (1). Algorithm 1 was implemented in Python, as a part of the general ensemble framework DECOLEARN[1] which contains learning methods to generate four types of initial models and base learners with any configuration in the generation and refinement phases: MLP, CNNs, CART-based decision trees (Classification and Regression Tree models) and XGBoost models [6, 9, 20]. The experiment focuses on base learners formed by convolutional neural networks (CNNs). The pricing sub-problems (12) or (13) in which a new base learner is determined, is solved using TensorFlow 2.3. This approach does not guarantee global optimality for the pricing sub-problems, but allows us to efficiently work with non-trivial base learners. Solving the pricing sub-problems to

[1] see https://github.com/ouyang-w-19/decolearn

guarantee global optimality in a reasonable time is still not feasible for CNN base learners of relevant size, and is not the focus of this paper. DECOLEARN also uses GUROBI 9.5.1 for solving the LP master problem. All computational experiments were carried out on a computer with i7-1165G7 4-Core 2.80 GHz CPU and 16GB RAM.

The test instances for binary classification were obtained from the widely used CIFAR-10 data set [13]. The data set contains 60,000 32×32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images, divided into five training batches and one test batch, each with 10,000 images. The test batch contains 1,000 randomly selected images from each class, while the training batches contain the remaining images in random order. However, some training batches may contain more images from one class than another, with exactly 5,000 images from each class in total. For our experiments on binary classification using CIFAR-10, we chose two pairs of labels with varying degrees of similarity. The label pair of categories 3 and 8 is considered easy to distinguish, while the label pair 3 and 2 requires more sophisticated learners to generate an ensemble that can score better than random.

For base learner generation with a nonlinear loss, we used CNNs with VGG architectures [22] in both the generation and refinement phase based on the binary cross-entropy:

$$\mathcal{L}(h(x), y) = y \log(Q(h(x)) + \epsilon) + (1 - y) \log(1 - Q(h(x)) + \epsilon), \qquad (19)$$

where the clipping is defined as $Q(y) := \max(\min(y, 1 - \epsilon), \epsilon)$. The tolerance ϵ is added for stability reasons and is a small positive number, e.g., 10^{-7}. If the base learners have an output range of $[0, 1]$, the output is re-scaled to $[-1, 1]$ or alternatively to $\mathbb{R}$. Note that (19) requires a different binary label type for base learner training, namely $y \in \{0, 1\}$. The complexity of the models were adjusted to the degree of similarity between two labels. The details of the algorithm parameters, model types and hyper-parameters are documented in Appendix A and Tables 8 and 9.

Evaluation of the algorithm entails measurable performance indicators. We will look at the ensemble binary accuracy, diversity and the solution of the master problem in each iteration. The diversity measurement is defined over one instance. To describe the overall diversity of an ensemble, we will use the average diversity over the training set defined by

$$\overline{\text{div}}(f) := \frac{1}{|T|} \sum_{j \in T} \text{div}(f(x_j), y_j).$$

We compare variants of the algorithm varying the types of sub-problems and settings. One of the variants solves sub-problems with active data only and another variant encompasses all training data when solving the sub-problem. For the data selection-based ensemble method, we use two variants. The linear variant (LL) applies the standard LPBoost with a sub-problem that minimizes the reduced cost (pricing problem) in the refinement phase with a linear expected loss function

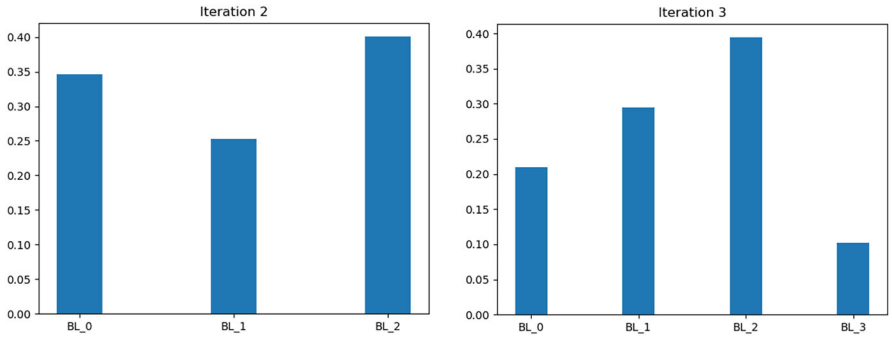


Fig. 1 Generated base learners with their master problem weight. The generation phase ends after 2 iterations. In the next iteration, the old base learners remain active together with a new active column

$$\max_{h\in\mathcal{F}} \sum_{(x_j,y_j)\in T} u_j y_j h(x_j). \tag{20}$$

We compare that to a nonlinear variant called NL which considers sub-problems on the active set using nonlinear loss function (19). We will first focus on the behavior of the variants of DDA in the generation phase and then on their behavior in the refinement phase.

4.1 Generation Phase

The generation phase is optional and serves the purpose of generating base learners through an error-based bootstrapping approach, which does not rely on any inputs from the master problem. In each iteration, the master problem is only solved to calculate an ensemble and its corresponding active set R, which provides stopping criterion values for this phase. Active columns in the master problem refer to the columns with nonzero weights of the corresponding base learner. The number of iterations depends on the success of the phase to generate new columns according to a termination criterion. In the best case, the generated columns remain active in the refinement phase. This is illustrated when running the DDA generation phase on an instance of CIFAR-10, where after 2 iterations the phase ends and all generated base learners stay active, see Fig. 1. The case in Fig. 2 illustrates a run over instance MNIST (a well known test problem) where four out of six BL generated in the generation phase have zero-value weight in the next phase. The idea is that the generation phase is effective if it provides many active columns in a short time.

We made experiments varying the stopping criterion of the generation phase. We compare a variant with a fixed number of 5 iterations, with a variant which uses the criterion on the relative progress of Line 11 in DDA, Algorithm 1. Table 1 presents the statistical results of 20 runs of DDA one iteration after the generation phase for both stopping criterion variants. The number of inactivate columns corresponds to the ensemble obtained from the generation phase. A relative criterion has a relatively smaller number of inactive columns than a fixed number of 5 iterations. The number of

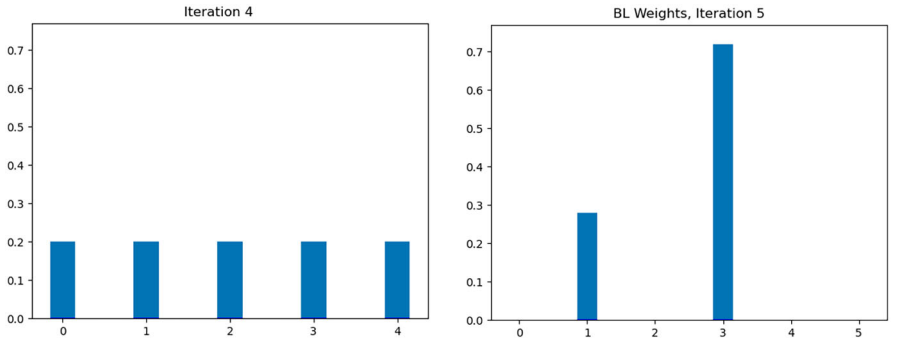


Fig. 2 Six base learners (corresponding weights on the y-axis) are generated in the generation phase. As shown by the second figure, most of the columns become inactive in the next iteration

Table 1 Effect of fixed versus relative stopping criterion on number of columns becoming inactive, number of iterations and misclassified data in the validation set

Stopping criterion	Fix 5 iterations	Relative progress
Inactivated col. (frac.)	0.49 ± 0.15	0.29 ± 0.23
Iterations	5	2.6 ± 0.66
nr. misclassified train	2084 ± 949	2271 ± 1064
nr. misclassified val	335 ± 184	649 ± 461

Table 2 DDA with or without generation phase on categories 3 and 2 of the CIFAR-10 data set

DDA	Without gen. phase	With gen. phase
Initial nr. misclassified train	4882 ± 303	
Initial nr. misclassified val	978 ± 54	
nr. misclassified train	3024 ± 1511	1607 ± 772
nr. misclassified val	723 ± 200	576 ± 34
Total time (s)	205.84 ± 177.15	412.22 ± 234.26

iterations for the relative criterion is 2.6 on average. The last row gives the number of misclassified instances in the validation set of the ensemble after the generation. This illustrates that with more iterations, the result generalizes better. The results reveal that proceeding with bootstrapping up to, on average, twice as many iterations generates an ensemble that generalizes better by a factor of roughly two.

To study the contribution of the generation phase to DDA, we run the algorithm with and without the phase for an instance of CIFAR-10 with binary labels (categories 3 and 2). Ten runs of the algorithm are performed, and Table 2 presents the obtained statistical results. We measured the classification quality for the final ensemble as the number of misclassified data points of the training set and of the validation set. The results show the average values over ten runs with the standard deviation. Table 2 indicates that the generation phase can improve the quality of the final ensemble, that

Table 3 Comparing LL and NL variant on category 3 and 8 instances of CIFAR-10, Experiment A

	LL variant	NL variant
nr. Runs	10	10
Initial nr. misclassified train	3283 ± 1342	
Initial nr. misclassified val	654 ± 267	
nr. misclassified train	1924 ± 767	840 ± 274
nr. misclassified val	389 ± 146	194 ± 49
Total time (s)	65.21 ± 40.61	51.53 ± 20.85

Table 4 Comparing LL and NL variant results on categories 3 and 2 instances of CIFAR-10, Experiment B

	LL variant	NL variant
nr. Runs	10	10
Initial nr. misclassified train	4661 ± 312	
Initial nr. misclassified val	933 ± 60	
nr. misclassified train	4248 ± 642	1607 ± 772
nr. misclassified val	861 ± 124	576 ± 34
Total time (s)	717.19 ± 717	412.22 ± 234.26

is, the number of misclassified data points (testing/validation set) decrease from 723 to 576 on average. On the other hand, the average computational time of DDA is doubled when adding the generation phase.

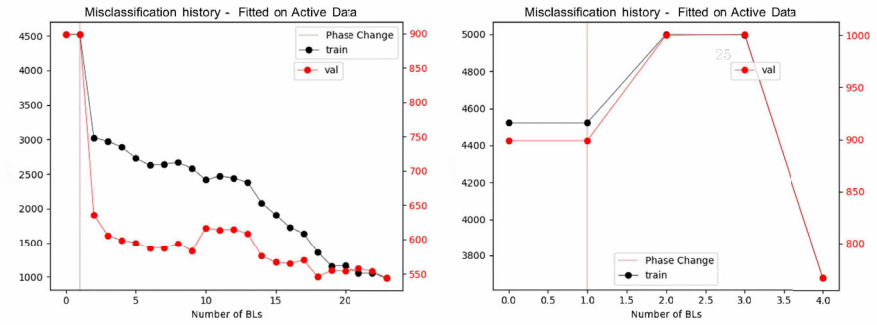
The next section will illustrate that a big improvement (lower) in misclassification score may occur immediately after the generation phase at a relatively low computational cost.

4.2 Refinement Phase

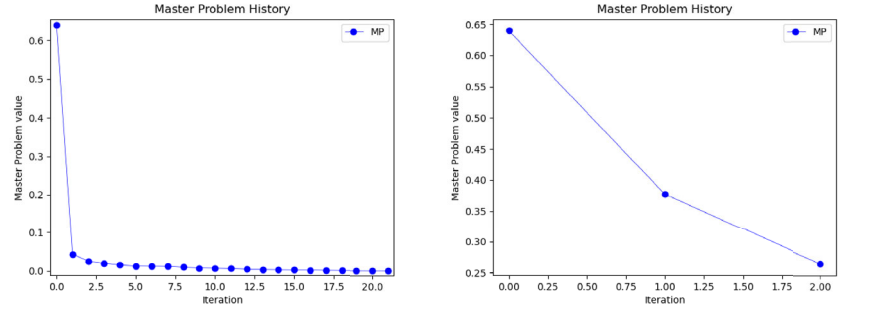
The refinement phase adds base learners to set $\mathcal{H}$. We used two data sets that vary in difficulty to compare the behavior for the variants LL and NL. Experiment A denotes the data set CIFAR-10 category pair 3 and 8 and is considered relatively easy. Experiment B enhances CIFAR-10 category pair 3 and 2 and is considered more challenging. We measure the misclassification over the validation and training set and the time consumed by the platform.

Tables 3 and 4 show the results of the two variants over the two experiments. In both cases, the nonlinear loss function variant appears to lead to less misclassification. The effectiveness over the validation set illustrates the generalization capability. The total computational time varies significantly, but seems reasonable for both variants.

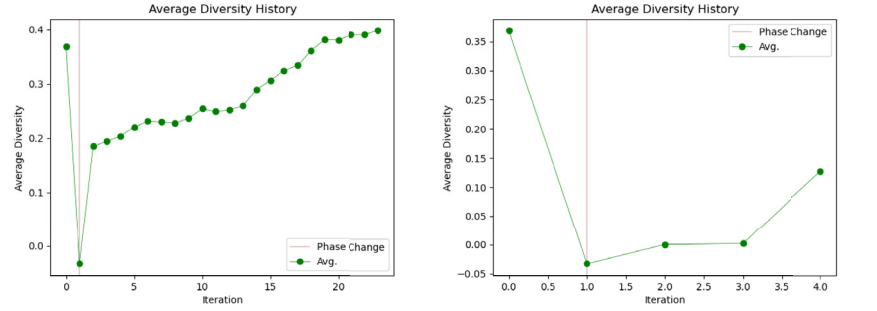
Figure 3 contains various graphs depicting different measures for the experiment. The x-axis gives the number of iterations (or number of BL) and the y-axis the misclassification score of training set and validation set for the LL and NL variant of



(a) Misclassification increasing iterations NL variant DDA (b) Misclassification increasing iterations LL variant DDA



(c) Master objective function value over refinement iterations for NL variant (d) Master objective function value over refinement iteration for LL variant

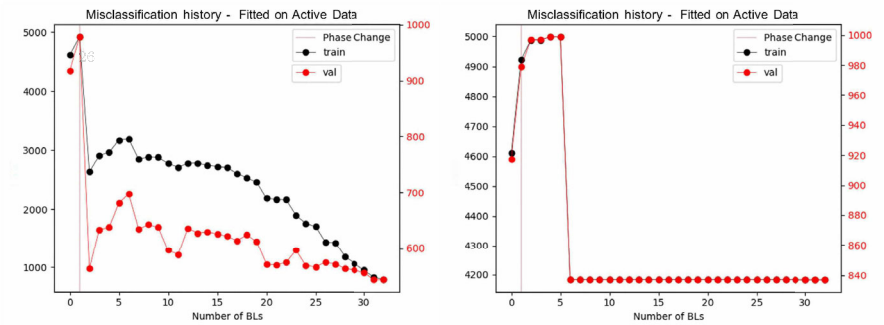


(e) Average diversity development for NL variant DDA (f) Average diversity development LL variant

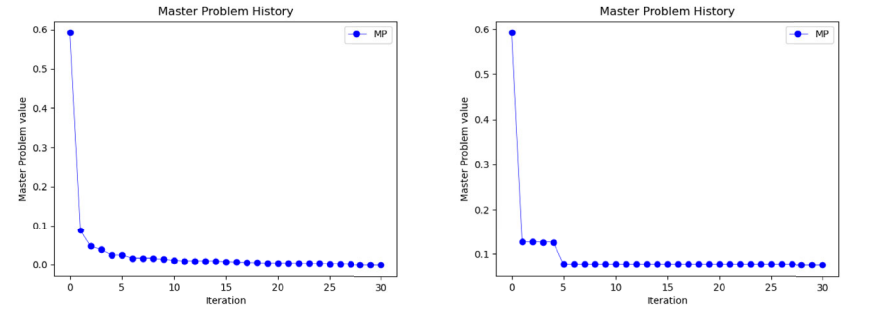
Fig. 3 Experiment B. Misclassification for increasing iterations (generated BL) for the LL and NL variants

the DDA algorithm. Note that the misclassification goes down fast when using the nonlinear loss function.

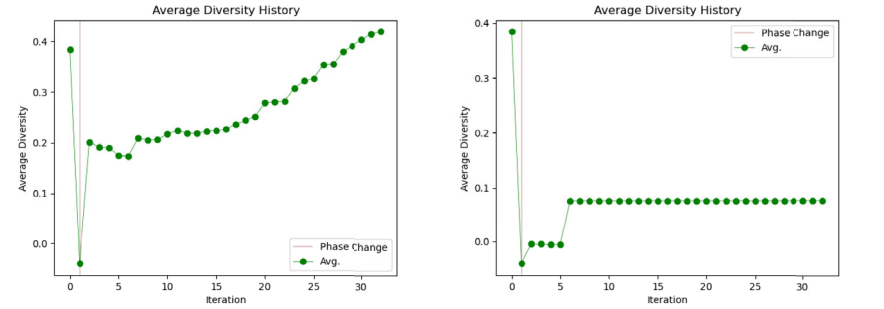
We now study the behavior of both variants of DDA using the data of Experiment B. Instead of using a stopping criterion in the refinement phase, we fix the number of iterations. By doing so, we can illustrate that the misclassification does not improve anymore for the LL variant after the sixth iteration. Using the nonlinear loss function, the misclassification keeps an improving trend when generating more base learners,



(a) Misclassification increasing iterations NL variant DDA (b) Misclassification increasing iterations LL variant DDA



(c) Master objective function value trend of the NL variant for refinement phase (d) Master objective function value trend of the LL variant for refinement phase



(e) Average diversity trend of NL variant (f) Average diversity trend of LL variant

Fig. 4 Experiment fixing 30 refinement phase iterations. Misclassification of data points for NL and LL variant

i.e., with increasing iterations. This is illustrated in Fig. 4. Interestingly, the average diversity has the same tendency as illustrated in Fig. 4e and f.

In our experiment, we observe the behavior of the algorithm when we vary the data reduction parameter ν, which influences the number of data points that are considered active, setting $\lambda = \frac{1}{\nu|T|}$ in (8). We use the data of Experiment B and use five runs with

Table 5 Results experiment B (NL variant) varying data reduction $\nu \in (0, 1)$

ν	Misclassified train	Misclassified val	Total time
0.1	2380 ± 908	625 ± 72	254.27 ± 227.60
0.4	2520 ± 862	592 ± 114	134.47 ± 77.12
0.5	2061 ± 486	511 ± 52	138.88 ± 93.65
0.6	2324 ± 163	510 ± 26	112.93 ± 63.15
0.7	2273 ± 132	502 ± 24	135.25 ± 49.40
0.8	3828 ± 670	787 ± 112	236.41 ± 227.02
1.0	4365 ± 442	874 ± 78	94.87 ± 29.49

the NL variant of the DDA algorithm. We measure the misclassification of the data and the total running time. The resulting values are reported in Table 5.

4.3 Comparison of Base Learner and Ensemble Accuracy

In our last series of experiments, we compare the highest base learner and ensemble accuracy, in order to measure the effectiveness of the ensemble model varying the quality of the base learners. Although a weak base learner is usually seen as a learner which performs slightly better than random (see [12]), in our experiments, we could use weak base learners that have an average validation accuracy of 72.47% over five runs and stronger base learners that have an average validation accuracy of 77.7% over five runs. The hyper-parameter configuration for the strong and weak base learners is listed in Table 10.

Comparing the results of the strong and weak base learners in Tables 6 and 7 one can observe that for the weak base learners, the difference between training and validation accuracy is relatively low (i.e., the base learners do not overfit). Consequently, the ensemble models have the same behavior.

5 Discussion and Conclusion

This paper presents and analyzes an algorithm that iteratively improves an ensemble of base learners based on a column generation approach. After an initial improvement based on error-based bootstrapping, the algorithm uses a data selection strategy which focuses on the active data points, *i.e.,* the points with nonzero dual value, which are sensitive regarding the margin (objective). The implementation of the algorithm is part of the ensemble framework DECOLEARN, and it allows various experimental setups to reliably improve an initial solution by adding base learners to an ensemble. All experiments were executed in a binary classification setting. The focus of the algorithm is on the refinement phase that performs linear programming on an iteratively growing set of score functions to maximize a soft margin (LPBoost). The LP constructs an ensemble from weighted score functions, populates the set of active data until an adjustable limit is reached and sets a stopping criterion for the algorithm. In numerical

Table 6 Highest base learner and ensemble accuracy with strong base learners (s. 10), data reduction $\nu = 0.5$

	Highest BL accuracy		Highest ensemble accuracy	
	Train	Validation	Train	Validation
Run 1	94.16	78.80	**96.80**	**82.00**
Run 2	94.33	78.20	**94.91**	**78.65**
Run 3	95.08	79.25	**95.47**	**79.45**
Run 4	95.25	78.65	**95.48**	78.65
Run 5	74.68	73.60	**75.05**	**77.40**

Highest base learner and ensemble accuracy written in bold numbers

Table 7 Highest base learner and ensemble accuracy with weaker base learners (s. 10), data reduction $\nu = 0.5$

	Highest BL accuracy		Highest ensemble accuracy	
	Train	Validation	Train	Validation
Run 1	71.44	71.6	**81.8**	**79.3**
Run 2	74.41	73.2	**83.37**	**78.65**
Run 3	73.38	73.35	**83.7**	**78.4**
Run 4	73.15	70.65	**84.08**	**79.85**
Run 5	75.25	73.55	**82.54**	**78.45**

Highest base learner and ensemble accuracy written in bold numbers

experiments, we investigated a variant with the traditional linear loss function and a nonlinear loss function based on cross-entropy.

Experiments show that the nonlinear variant which relies on solving nonlinear sub-problems defined by active data only is potentially more effective in reducing the generalization error than the variant that uses a linear loss function. Furthermore, experiments with this variant reveal a correlation between the generalization error and a broader ensemble diversity definition relying on base learners $h_i(x) \in [-1, 1]$, thus relaxing the original assumption set by [5]. We postulate that since the underlying distribution of the set of active data can significantly deviate from the original distribution, the investigated algorithm increases both accuracy and diversity as long as the set of active data, which by definition contains the data points most difficult to correctly predict, is significantly changing.

We summarize our findings by the following bullet points:

- We have presented a column generation based decomposition algorithm DDA for optimizing ensemble models that allows us to optimize one base learner at a time and only using a sub-set of the training data.
- We showed that algorithm DDA computes an optimal ensemble model in the convex hull of a given DNN model space.
- The results in Tables 6 and 7 show that the generated ensemble models have the potential to reach an accuracy which is significantly better than the accuracy of the individual base learners, demonstrating that the DNN model space used in the experiments is a proper sub-set of its convex hull.

- Using reduced data and non-linear sub-problems seems beneficial to the LPBoost approach.
- We pose the following hypothesis: Diversity emerges from fitting the model with active data set R. The size of R can be chosen deliberately small. However, a small active data set may deteriorate accuracy.
- Model diversity, as defined by [5], seems to be well-correlated to the generalization error.
- We found that the generation phase can improve the final ensemble results at higher computational cost.
- The implementation of the generation phase is efficient in a way that it generates effective BL as observed from the weight histograms. It is also recommended not to use a fixed iteration number as stopping criterion in the generation phase, otherwise generating ineffective BL is highly likely.

Our future work will focus on improving accuracy, *i.e.,* using a base learner generation strategy, which makes it possible to improve the model $\hat{f}$ locally on an active set R without compromising the performance on the non-active set $T \setminus R$. Moreover, methods to accelerate the generation of base learners can be used, *e.g.,* dimension reduction and decomposition methods. The results encourage investigating experiments in a broader sense, *i.e.,* extension to non-binary classification tasks or dynamic setting of the refinement phase parameters after each iteration.

Acknowledgements This work has been funded by grant 01IS19079 of the German Federal Ministry of Education and Research (BMBF), LFF FV 90 of the State Research Funding Hamburg (Landesforschungsförderung Hamburg), PID2021-123278OB-I00 from the Spanish Ministry of Science and Innovation financed by "ERDF A way of making Europe" and by MCIN/AEI/10.13039/501100011033 and by Digital Futures at KTH and C3.ai Digital Transformation Institute project "AI techniques for Power Systems Under Cyberattacks"

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availability The data of the constructed test instances can be found in a GitHub repository and is publicly available on https://github.com/ouyang-w-19/decolearn.

A Configuration of Algorithm Meta-Parameters and Model Hyper-Parameters

The settings of the algorithm meta-parameters in the numerical experiments are summarized in Table 8. The model type of the base learners obtained in both the generation and refinement phases is set as CNN. The threshold of active data is set as zero, that is, all instances with nonzero dual values are considered active. The meta-parameter

Table 8 Decolearn meta-parameters

Parameter name	Value
Nu	2e-1
Generation model type	CNN
Refinement model type	CNN
Combining mechanism	Averaging
Active data threshold ϕ	0.0
Min. generation phase change σ	2e-1

Table 9 Configuration of base learners for NL and LL comparison

Parameter name	Generation	Refinement
Architecture	VGG	VGG
Learning rate	1e-3	1e-3
Loss	BCE/LL	BCE/LL
Activation function	tanh	tanh
Last layer activation	tanh	tanh
Batch size	50	50
Episodes	5	8
Layers	(32, 32, 16)	(32, 32, 16)
Filter size	(2, 2)	(2, 2)
Max. pooling filter	(2, 2)	(2, 2)
FC layers	(8, 1)	(8, 1)

BCE: BinaryCross-entropy, L: Linear

ν of Algorithm 3 is taken as $\nu = 0.2$, while the meta-parameter σ of the termination condition for the generation phase in Algorithm 1 is set to $\sigma = 0.2$.

Base Learner Hyper-parameters for DDA The hyper-parameters and configurations for the base learners in the DDA algorithm are presented in Table 9. The configurations of base learners in the generation phase are slightly different from the ones in the refinement phase, that is, the number of episodes in the refinement phase is increased from five to eight. The NL variant uses binary cross-entropy, while the LL variant is based on a linear loss. Table 10 shows the base learner configuration in the DDA algorithm of base learners that significantly differ in their accuracy.

Table 10 Configuration of base learners

Parameter Name	Strong BL		Weak BL	
	Generation	Refinement	Generation	Refinement
Architecture	VGG	VGG	VGG	VGG
Learning rate	1e-3	1e-3	1e-3	1e-3
Loss	BCE	BCE	BCE	BCE
Activation function	tanh	tanh	tanh	tanh
Last layer activation	Sigmoid	Sigmoid	relu	relu
Batch size	25	25	25	25
Episodes	7	7	7	7
Layers	(64, 32, 32)	(64, 32, 32)	(64, 32, 32)	(64, 32, 32)
Filter size	(2, 2)	(2, 2)	(2, 2)	(2, 2)
Max. pooling filter	(2, 2)	(2, 2)	(2, 2)	(2, 2)
FC layers	(15, 1)	(15, 1)	(15, 1)	(15, 1)

BCE: Binary Cross-entropy

References

1. Aakerberg, A., Nasrollahi, K., Heder, T.: Improving a deep learning based rgb-d object recognition model by ensemble learning. In: 2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA), pp. 1–6. IEEE (2017)
2. Abdollahi, A., Pradhan, B., Alamri, A.M.: An ensemble architecture of deep convolutional SEGNET and UNET networks for building semantic segmentation from high-resolution aerial images. Geocarto International pp. 1–16 (2020)
3. Benítez-Peña, S., Carrizosa, E., Guerrero, V., Jiménez-Gamero, M.D., Martín-Barragán, B., Molero-Río, C., Ramírez-Cobo, P., Romero Morales, D., Sillero-Denamiel, M.R.: On sparse ensemble methods: an application to short-term predictions of the evolution of COVID-19. Eur. J. Oper. Res. **295**(2), 648–663 (2021)
4. Berrada, L., Zisserman, A., Kumar, M.P.: Deep Frank–Wolfe for neural network optimization. (2018). arXiv:1811.07591
5. Bian, Y., Chen, H.: When does diversity help generalization in classification ensembles? IEEE Trans. Cyber. **52**(9), 9059–9075 (2022)
6. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.: Classification and Regression Trees. Chapman and Hall/CRC, Boca Raton (1984)
7. Demiriz, A., Bennett, K.P., Shawe-Taylor, J.: Linear programming boosting via column generation. Mach. Learn. **46**(1), 225–254 (2002)
8. Dietterich, T.G.: Ensemble methods in machine learning. In: Multiple Classifier Systems, pp. 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)
9. Fukushima, K.: Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol. Cybern. **36**(4), 193–202 (1980)
10. Hijazi, M.H.A., Hwa, S.K.T., Bade, A., Yaakob, R., Jeffree, M.S.: Ensemble deep learning for tuberculosis detection using chest x-ray and canny edge detected images. IAES Int. J. AI **8**(4), 429 (2019)
11. Jiang, H., Qiu, X., Chen, J., Liu, X., Miao, X., Zhuang, S.: Insulator fault detection in aerial images based on ensemble learning with multi-level perception. IEEE Access **7**, 61797–61810 (2019)
12. Kearns, M.J., Vazirani, U.V.: An Introduction to Computational Learning Theory. MIT Press, Cambridge, MA, USA (1994)
13. Krizhevsky, A.: Learning Multiple Layers of Features from Tiny Images. http://api.semanticscholar.org/CorpusID:18268744 (2009)
14. Lakshminarayanan, B., Pritzel, A., Blundell, C.: Simple and scalable predictive uncertainty estimation using deep ensembles. Adv. Neural Inform. Process. Syst. **30** (2017)

15. Li, Y., Song, Y., Jia, L., Gao, S., Li, Q., Qiu, M.: Intelligent fault diagnosis by fusing domain adversarial training and maximum mean discrepancy via ensemble learning. IEEE Trans. Industr. Inf. **17**(4), 2833–2841 (2020)
16. Lin, Y.Y., Liu, T.L., Fuh, C.S.: Local ensemble kernel learning for object category recognition. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE (2007)
17. Malisiewicz, T., Gupta, A., Efros, A.A.: Ensemble of exemplar-svms for object detection and beyond. In: 2011 International Conference on Computer Vision, pp. 89–96. IEEE (2011)
18. Muts, P., Bruche, S., Nowak, I., Wu, O., Hendrix, E.M., Tsatsaronis, G.: A column generation algorithm for solving energy system planning problems. Optim. Eng. **24**, 317–351 (2023)
19. Rokach, L.: Ensemble-based classifiers. AI Rev. **33**, 1–39 (2010)
20. Ruck, D.W., Rogers, S., Kabrisky, M., Mills, J.: Target recognition: Conventional and neural network approaches. In: International 1989 Joint Conference on Neural Networks, pp. 608–vol. IEEE (1989)
21. Sagi, O., Rokach, L.: Ensemble learning: a survey. Wiley Interdisciplin. Rev.: Data Min. Knowled. Discov. **8**(4), e1249 (2018)
22. Simonyan, Z.: Very Deep Convolutional Networks for Large-Scale Image Recogniton. arXiv preprint, arXiv:1409.1556, (2014)
23. Thebelt, A., Kronqvist, J., Mistry, M., Lee, R.M., Sudermann-Merx, N., Misener, R.: Entmoot: a framework for optimization over ensemble tree models. Comput. Chem. Eng. **151**, 107343 (2021)
24. Tüysüzoğlu, G., Birant, D.: Enhanced bagging (Ebagging): a novel approach for ensemble learning. Int. Arab J. Inform. Technol. **17**(4), 515–528 (2020)
25. Wang, Y., Zhu, K., Sun, M., Deng, Y.: An ensemble learning approach for fault diagnosis in self-organizing heterogeneous networks. IEEE Access **7**, 125662–125675 (2019)
26. Xu, J., Wang, W., Wang, H., Guo, J.: Multi-model ensemble with rich spatial information for object detection. Pattern Recognit. **99**, 107098 (2020)
27. Zhou, Z.H.: Ensemble Methods: Foundations and Algorithms. CRC Press, Boca Raton (2012)