Machine Learning (2020) 109:1945–1986
https://doi.org/10.1007/s10994-020-05902-7

Fast greedy $\mathcal{C}$-bound minimization with guarantees

Baptiste Bauvin[1,3] · Cécile Capponi[3] · Jean-Francis Roy[2] · François Laviolette[3,4]

Received: 16 September 2019 / Revised: 21 July 2020 / Accepted: 11 August 2020 /
Published online: 23 September 2020

Abstract
The $\mathcal{C}$-bound is a tight bound on the true risk of a majority vote classifier that relies on the individual quality and pairwise disagreement of the voters and provides PAC-Bayesian generalization guarantees. Based on this bound, MinCq is a classification algorithm that returns a dense distribution on a finite set of voters by minimizing it. Introduced later and inspired by boosting, CqBoost uses a column generation approach to build a sparse $\mathcal{C}$-bound optimal distribution on a possibly infinite set of voters. However, both approaches have a high computational learning time because they minimize the $\mathcal{C}$-bound by solving a quadratic program. Yet, one advantage of CqBoost is its experimental ability to provide sparse solutions. In this work, we address the problem of accelerating the $\mathcal{C}$-bound minimization process while keeping the sparsity of the solution and without losing accuracy. We present CB-Boost, a computationally efficient classification algorithm relying on a greedy–boosting-based–$\mathcal{C}$-bound optimization. An in-depth analysis proves the optimality of the greedy minimization process and quantifies the decrease of the $\mathcal{C}$-bound operated by the algorithm. Generalization guarantees are then drawn based on already existing PAC-Bayesian theorems. In addition, we experimentally evaluate the relevance of CB-Boost in terms of the three main properties we expect about it: accuracy, sparsity, and computational efficiency compared to MinCq, CqBoost, Adaboost and other ensemble methods. As observed in these experiments, CB-Boost not only achieves results comparable to the state of the art, but also provides $\mathcal{C}$-bound sub-optimal weights with very few computational demand while keeping the sparsity property of CqBoost.

Editors: Ira Assent, Carlotta Domeniconi, Aristides Gionis, Eyke Hüllermeier.

✉ Baptiste Bauvin
baptiste.bauvin@lis-lab.fr
http://qarma.lis-lab.fr

Jean-Francis Roy
jfroy@coveo.com

François Laviolette
francois.laviolette@ift.ulaval.ca

[1] Aix Marseille University, Toulon University, CNRS, LIS (Qarma), Marseille, France

[2] Coveo Solutions Inc, Québec, QC, Canada

[3] Dép. d'informatique et de génie logiciel, Université Laval, Québec, QC, Canada

[4] Element AI, Montréal, QC, Canada

Keywords PAC-Bayes · Boosting · Ensemble methods · $\mathcal{C}$-bound · Greedy optimization

1 Introduction

Ensemble-based supervised classification consists in training a combination of many classifiers learnt from various algorithms and/or sub-samples of the initial dataset. Some of the most prominent ensemble learning frameworks include Boosting, with the seminal elegant Adaboost (Freund and Schapire 1997), which has been the inspiration of numerous other algorithms, Bagging (Breiman 1996) leading to the successful Random Forests (Breiman 2001), but also Multiple Kernel Learning (MKL) approaches (Sonnenburg et al. 2006; Lanckriet et al. 2004) or even the Set Covering Machines (Marchand and Taylor 2003). Most of these approaches are founded on theoretical aspects of PAC learning (Valiant 1984). Among them, the PAC-Bayesian theory studies the properties of the majority vote that is used to combine the classifiers (McAllester 1999) according to the distributions among them. Experimentally, valuable ad-hoc studies have been made over specific application domains in order to build relevant sets of classifiers. We address here the problem of learning one independently from the priors relevant to the application domain, together with a weighted schema that defines a majority vote over the members of that set of classifiers.

Introduced by McAllester (1999), the PAC-Bayesian theory provides some of the tightest Probably Approximately Correct (PAC) learning bounds. These bounds are often used for a better understanding of the learning capability of various algorithms (Seeger 2002; McAllester 2003; Langford and Shawe-Taylor 2003; Catoni 2007; Seldin et al. 2012; Dziugaite and Roy 2018). Based on the fact that PAC-Bayesian bounds gave rise to a powerful analysis of many algorithms' behavior, it has incited a research direction that consists in developing new (or new variants of) algorithms that simply are bound minimizers (Germain et al. 2009; Parrado-Hernández et al. 2012; Dziugaite and Roy 2018; Germain et al. 2015). In this paper, we revisit one of such algorithms, MinCq (Germain et al. 2015), which focuses on the minimization of the $\mathcal{C}$-bound and comes with PAC-Bayesian guarantees. The $\mathcal{C}$-bound, introduced in Lacasse et al. (2006), bounds the true risk of a weighted majority vote based on the averaged true risk of the voters, coupled with their averaged pairwise disagreement. According to the $\mathcal{C}$-bound, the quality of each individual voter can be compensated if the voting community tends to balance the individual errors by having plural opinions on "difficult" examples.

Although MinCq has state of the art performance on many tasks, it computes the output distribution on a set of voters through a quadratic program, which is not tractable for more than medium-sized datasets. To overcome this, CqBoost (Roy et al. 2016) has then been proposed. It iteratively builds a sparse majority vote from a possibly infinite set of classifiers, within a column generation setting. However, CqBoost's approach only partially tackles the computational challenge. In order to overcome this drawback, we propose CB-Boost, a greedy, boosting-based, $\mathcal{C}$-bound minimization algorithm designed to greatly reduce the computational cost of CqBoost and MinCq while maintaining the attractive peculiarities of the $\mathcal{C}$-bound on a finite set of hypothesis.

CB-Boost is somewhat similar to CqBoost, while closer to boosting in the sense that at each iteration, it selects a voter, finds its associated weight by minimizing an objective quantity (the $\mathcal{C}$-bound in the case of CB-Boost, and the exponential loss as for Adaboost) and adds it to the vote.

The main advantage of CB-Boost comes from the fact that at each iteration, it solves a $\mathcal{C}$-bound minimization problem by considering only one direction. Interestingly, it is possible to solve it analytically and with only a few light operations. Furthermore, we derive a guarantee that the $\mathcal{C}$-bound decreases throughout CB-Boost 's iterations.

This paper is organised as follows. Section 2 sets up basic notation and definitions, reviews the $\mathcal{C}$-bound and its PAC-Bayesian framework, and briefly introduces MinCq and CqBoost, two existing algorithms that aim at learning an ensemble of classifiers based on the minimization of the $\mathcal{C}$-bound. Section 3 introduces our new boosting-based algorithm named CB-Boost, which aims at keeping the benefits of these two algorithms, while reducing the disadvantages. Finally, Sect. 4 addresses the theoretical properties of CB-Boost, while Sect. 5 focuses on experiments that not only validate the theoretical aspects, but also shows that CB-Boost performs well empirically on major aspects.

2 Context

After setting up basic notations and definitions, the context of PAC-Bayesian learning is introduced through the $\mathcal{C}$-bound and two theorems, which are pivotal components of our contribution.

2.1 Basic notations and definitions

Let us consider a supervised bi-class learning classification task, where $\mathcal{X}$ is the input space and $\mathcal{Y} = \{-1, 1\}$ is the output space. The learning sample $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^m$ consists of m examples drawn *i.i.d* from a fixed, but unknown distribution D over $\mathcal{X} \times \mathcal{Y}$. Let $\mathcal{H} = \{h_1, \dots h_n\}$ be a set of n voters $h_s : \mathcal{X} \to \{-1, 1\}$, and $\text{Conv}(\mathcal{H})$ the convex hull of $\mathcal{H}$.

Definition 1 $\forall x \in \mathcal{X}$, the majority vote classifier (Bayes classifier) B_Q over a distribution Q on $\mathcal{H}$ is defined by

$$B_Q(x) \triangleq sg\left[\mathop{\mathbf{E}}_{h\sim Q} h(x)\right] \text{where } sg(a) = 1 \text{ if } a > 0 \text{ and } -1 \text{ otherwise.}$$

Definition 2 The true risk of the Bayes classifier B_Q over Q on $\mathcal{H}$, is defined as the expected zero one loss over D, a distribution on $\mathcal{X} \times \mathcal{Y}$:

$$R_D(B_Q) \triangleq \mathop{\mathbf{E}}_{(x,y)\sim D} I\left(\mathop{\mathbf{E}}_{h\sim Q} y \cdot h(x) < 0\right), \text{ with } I(a) = 1 \text{ if a true, 0 otherwise.}$$

Definition 3 The training error of the Bayes classifier B_Q over Q on $\mathcal{H}$, is defined as the empirical risk associated with the zero one loss on $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^m$.

$$R_{\mathcal{S}}(B_Q) = \frac{1}{m}\sum_{i=1}^{m} I\left(\mathop{\mathbf{E}}_{h\sim Q} y_i \cdot h(x_i) < 0\right).$$

Definition 4 The Kullback-Leibler (KL) divergence between distributions Q and P is defined as

$$KL(Q||P) = \mathop{\mathbf{E}}_{h\sim Q} \ln \frac{Q(h)}{P(h)}\,.$$

In the following study, let P denote the prior distribution on $\mathcal{H}$ that incorporates pre-existing knowledge about the task. And let Q denote the posterior distribution, which is an update of P after observing the task's data.

2.2 Previous work: the $\mathcal{C}$-Bound & PAC-Bayesian guarantees

Here, we state the main context of our work by presenting the $\mathcal{C}$-bound and its properties, as introduced in Lacasse et al. (2006). Let us first define one core concept: the margin of the majority vote.

Definition 5 Given an example $x \in \mathcal{X}$ and its label $y \in \mathcal{Y}$ drawn according to a distribution D, M_Q^D is the random variable that gives the margin of the majority vote B_Q, defined as $M_Q^D = y \mathop{\mathbf{E}}_{h\sim Q} h(x)\,.$

Given the margin's definition, the $\mathcal{C}$-bound is presented in Lacasse et al. (2006) as follows.

Definition 6 For any distribution Q on $\mathcal{H}$, for any distribution D on $\mathcal{X}\times\mathcal{Y}$, let $\mathscr{C}_Q^D$ be the $\mathcal{C}$-bound of B_Q over D, defined as

$$\mathscr{C}_Q^D = 1 - \frac{\left(\mu_1(M_Q^D)\right)^2}{\mu_2(M_Q^D)}\,,$$

with $\mu_1(M_Q^D)$ being the first moment of the margin

$$\mu_1(M_Q^D) = \mathop{\mathbf{E}}_{(x,y)\sim D} M_Q(x,y),$$

and $\mu_2(M_Q^D)$ being the second moment of the margin

$$\mu_2(M_Q^D) = \mathop{\mathbf{E}}_{(x,y)\sim D}\left[M_Q(x,y)^2\right]$$

where the last equality comes from the fact that $y \in \{-1, 1\}$, so $y^2 = 1$.

Definition 7 For any distribution Q on $\mathcal{H}$ and any $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^m$, let $\mathscr{C}_Q^{\mathcal{S}}$ be the **empirical** $\mathcal{C}$-bound of B_Q on $\mathcal{S}$, defined as

$$\mathscr{C}_Q^{\mathcal{S}} = 1 - \frac{1}{m}\frac{\left(\sum_{i=1}^m y_i \mathop{\mathbf{E}}_{h\sim Q} h(x_i)\right)^2}{\sum_{i=1}^m \left(y_i \mathop{\mathbf{E}}_{h\sim Q} h(x_i)\right)^2}$$

The following theorem, established and proven in Lacasse et al. (2006), shows that the $\mathcal{C}$-bound is an upper bound of the true risk of the majority vote classifier.

Theorem 1 *Lacasse et al. (2006) For any distribution Q on a set $\mathcal{H}$ of hypotheses, and for any distribution D on $\mathcal{X} \times \mathcal{Y}$, if $\mu_1(M_Q^D) > 0$, we have*

$$R_D(B_Q) \leq \mathscr{C}_Q^D .$$

From this result, we derive a corollary for the empirical risk, i.e., the training error that is used in Sect. 4:

Corollary 1 *For any distribution Q on a set $\mathcal{H}$ of hypotheses, and for $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^m$ a training sample, if $\frac{1}{m} \sum_{i=1}^{m} y_i \mathop{\mathbf{E}}_{h \sim Q} h(x_i) > 0$, we have*

$$R_{\mathcal{S}}(B_Q) \leq \mathscr{C}_Q^{\mathcal{S}} .$$

In terms of generalization guarantees, the PAC-Bayesian framework (McAllester 1999) provides a way to bound the true risk of B_Q, given the empirical $\mathcal{C}$-bound, and P and Q the prior and posterior distributions. The important following theorem, established by Roy et al. (2016) is used in Sect. 4.3; it gives an upper bound of the true risk of the majority vote, which depends on the first and second moments of the margin as introduced in Definitions 5 and 6, and on the Kullback-Leibler divergence between the prior and posterior distributions.

Theorem 2 *Roy et al. (2016) For any distribution D on $\mathcal{X} \times \mathcal{Y}$, for any set $\mathcal{H}$ of voters $h : \mathcal{X} \to \{-1, 1\}$, for any prior distribution P on $\mathcal{H}$ and any $\delta \in]0, 1]$ over the choice of the sample $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^m \sim D^m$, and for every posterior distribution Q on $\mathcal{H}$, we have, with a probability at least $1 - \delta$*

$$R_D(B_Q) \leq 1 - \frac{\left(max(0, \underline{\mu_1})\right)^2}{min(1, \overline{\mu_2})} , \text{ where:}$$

$$\underline{\mu_1} = \frac{1}{m} \sum_{i=1}^{m} y_i \mathop{\mathbf{E}}_{h \sim Q} h(x_i) - \sqrt{\frac{2}{m}\left[KL(Q||P) + \ln\left(\frac{2\sqrt{m}}{\delta/2}\right)\right]}$$

$$\overline{\mu_2} = \frac{1}{m} \sum_{i=1}^{m} \left(y_i \mathop{\mathbf{E}}_{h \sim Q} h(x_i)\right)^2 + \sqrt{\frac{2}{m}\left[2KL(Q||P) + \ln\left(\frac{2\sqrt{m}}{\delta/2}\right)\right]} .$$

2.3 Existing algorithms: MinCq & CqBoost

Let us focus on two algorithms that rely on the minimization of the empirical $\mathcal{C}$-bound in order to learn an accurate ensemble of classifiers. MinCq (Germain et al. 2015) finds the weights that minimize the empirical $\mathcal{C}$-bound on a given set of voters, and CqBoost (Roy et al. 2016), inversely, uses column generation and boosting to iteratively build a set of voters by minimizing the $\mathcal{C}$-bound.

MinCq[1] (Germain et al. 2015) The principle of MinCq is to create a majority vote over a finite set of voters, whose weights minimize the $\mathcal{C}$-bound. To avoid overfitting, it considers a restriction on the posterior distribution named quasi-uniformity (Germain et al. 2015), and adds an equality constraint on the first moment of the margin.

Beneficially, it is ensured to perform as well in train as in generalization, according to Corollary 1 and Theorem 2. The main drawback of MinCq is its computational training time: its algorithmic complexity is $\mathcal{O}(m \times n^2 + n^3)$, which prevents it from scaling up to large datasets.

CqBoost[2] (Roy et al. 2016) Like MinCq, CqBoost is an algorithm based on the minimization of the $\mathcal{C}$-bound. It was designed to accelerate MinCq and has proven to be a sparser $\mathcal{C}$-bound minimizer, hence enabling better interpretability and faster training.

CqBoost is based on a column generation process that iteratively builds the majority vote. It is similar to boosting in the way that, at each iteration t, the choice of the voter to add is made by greedily optimizing the edge criterion (Demiriz et al. 2002). Once a voter has been added to the current selected set, CqBoost finds the optimal weights by minimizing the $\mathcal{C}$-bound similarly to MinCq, solving a quadratic program.

The sparsity of CqBoost is explained by its ability to stop the iterative vote building early enough to avoid overfitting. Nevertheless, even though CqBoost is faster and sparser than MinCq, it is still not applicable to large datasets because of its algorithmic complexity is $\mathcal{O}(m \times T^2 + T^3)$, where T is the number of boosting iterations.

3 CB-Boost: A fast $\mathcal{C}$-Bound minimization algorithm

In this section, we present the mono-directional $\mathcal{C}$-bound minimization problem and its solution, which are central in CB-Boost. Then, we introduce CB-Boost 's pseudo-code in Algorithm 1.

The empirical $\mathcal{C}$-bound of a distribution $Q = \{\pi_1, \ldots, \pi_n\}$ of n weights over $\mathcal{H} = \{h_1, \ldots, h_n\}$ a set of n voters, on a learning sample $\mathcal{S}$ of m examples, is as follows.

$$\mathscr{C}_Q^{\mathcal{S}} = 1 - \frac{1}{m} \frac{\left(\sum_{i=1}^{m} y_i \sum_{s=1}^{n} \pi_s h_s(x_i) \right)^2}{\sum_{i=1}^{m} \left(y_i \sum_{s=1}^{n} \pi_s h_s(x_i) \right)^2}.$$

It is proven in Appendix D.1 that if we use positive weights $\{\alpha_1, \ldots, \alpha_n\} \in (\mathbb{R}^+)^n$ instead of a distribution, the empirical $\mathcal{C}$-bound is equivalent to using the distribution $Q = \{\frac{\alpha_1}{\sigma}, \ldots, \frac{\alpha_n}{\sigma}\}$, with σ being $\sum_{s=1}^{n} \alpha_s$. From here, we use the weights that do not sum to one in order to simplify the proofs.

[1] The pseudo-code of MinCq is given in Appendix A.1.
[2] CqBoost's pseudo-code is given in Appendix A.2.

3.1 Optimizing the $\mathcal{C}$-Bound in one direction

We outline some basic definitions to clarify the contributions of the following work : agreement ratio, margin and norm.

Definition 8 The agreement ratio between two voters $h, h' \in \mathcal{H}$ (or two combination of voters in $\text{Conv}(\mathcal{H})$) is defined as $\tau(h, h') \triangleq \frac{1}{m}\sum_{i=1}^{m} h(x_i)h'(x_i)$.

Definition 9 The empirical margin of a single voter $h \in \mathcal{H}$ (or combination of voters in $\text{Conv}(\mathcal{H})$) is $\gamma(h) \triangleq \frac{1}{m}\sum_{i=1}^{m} y_i h(x_i)$.

Definition 10 The squared and normalized L2-norm of $h \in \mathcal{H}$ (or combination of voters in $\text{Conv}(\mathcal{H})$), is defined as $\nu(h) \triangleq \frac{1}{m}\sum_{i=1}^{m} h(x_i)^2 = \frac{1}{m}\|h\|_2^2$.

Here, we consider the $\mathcal{C}$-bound optimization in a single direction, meaning that all weights, except one, are fixed. For readability reasons, we introduce $\forall i \in [m]$, $F_k(x_i) = \sum_{\substack{s=1 \\ s \neq k}}^{n} \alpha_s h_s(x_i)$ which denotes the majority vote built by all the fixed weights and their corresponding voters, and (α, h_k) the weight that varies during the optimization and its corresponding voter. We can thus rewrite the empirical $\mathcal{C}$-bound with respect to k, the varying direction, as

$$\mathscr{C}_k(\alpha) = 1 - \frac{1}{m}\frac{\left(\sum_{i=1}^{m} y_i\big(F_k(x_i) + \alpha h_k(x_i)\big)\right)^2}{\sum_{i=1}^{m}\big(y_i\big(F_k(x_i) + \alpha h_k(x_i)\big)\big)^2}.$$

Our goal here is to find the optimal α in terms of $\mathcal{C}$-bound, denoted by $\alpha_k^* = \arg\min_{\alpha \in \mathbb{R}^+} \mathscr{C}_k(\alpha)$. The following theorem is the central contribution of our work as it provides an analytical solution to this problem.

Theorem 3 *$\forall k \in [n]$ with the previously introduced notations, if $\gamma(h_k) > 0$ and $\gamma(F_k) > 0$, then*

$$\alpha_k^* = \arg\min_{\alpha \in \mathbb{R}^+} \mathscr{C}_k(\alpha) = \begin{cases} \frac{\gamma(h_k)\nu(F_k) - \gamma(F_k)\tau(F_k,h_k)}{\big(\gamma(F_k) - \gamma(h_k)\tau(F_k,h_k)\big)} \text{ if } \tau(F_k,h_k) < \frac{\gamma(F_k)}{\gamma(h_k)}, \\ 0 \text{ otherwise.} \end{cases}$$

The proof is provided in the Appendix, in Sect. D.1.

Theorem 3 states that in a specific direction, the $\mathcal{C}$-bound has a global minimum, provided three conditions. The first two ($\gamma(h_k) > 0$ and $\gamma(F_k) > 0$) are met trivially within our framework as h_k is a weak classifier[3] and F_k is a positive linear combination of weak classifiers. The third one $\big(\tau(F_k, h_k) < \frac{\gamma(F_k)}{\gamma(h_k)}\big)$ means that F_k and h_k are not supposed to be colinear, which in the next section, we will show is not restrictive.

[3] A weak classifier is a classifier that is slightly better than random classification.

This theoretical result is the main step in building a greedy $\mathcal{C}$-bound minimization algorithm. Moreover, as long as there is a direction k in which $\tau(F_k, h_k) < \frac{\gamma(F_k)}{\gamma(h_k)}$, the $\mathcal{C}$-bound can be optimized in this direction, and every other one in which $\tau(F_k, h_k) \geq \frac{\gamma(F_k)}{\gamma(h_k)}$ is a dead end.

In terms of complexity, the solution to the minimization problem is obtained in $\mathcal{O}(m)$ as $\gamma(h_k)$, $\nu\big(F_k\big)$, $\gamma(F_k)$, and $\tau(F_k, h_k)$ are sums over the m examples of the training set $\mathcal{S}$.

3.2 Optimally choosing the direction

In the previous subsection, we presented a theoretical result proving that, for a given direction, the $\mathcal{C}$-bound minimization problem has a unique solution. Here, we propose a way to optimally choose this direction and compare it to the main existing method.

Exhaustive search In our framework, $\mathcal{H}$ is finite and has a cardinality of n, implying that we have a finite number of available directions to choose from. As stated before, the minimum $\mathcal{C}$-bound in one direction is available in $\mathcal{O}(m)$. So by computing these minima in each direction, in $\mathcal{O}(n \times m)$, we are able to choose the optimal direction, in which the $\mathcal{C}$-bound decreases the most.

Comparison with gradient boosting In the gradient boosting framework (a), the optimization direction is chosen by gradient minimization. Coupled with an adequate method to choose the step size, it is a very efficient way of optimizing a loss function. However, thanks to our theoretical analysis, we know that at each iteration of CB-Boost, the best direction is chosen and the optimal step size is known analytically.

Nonetheless, we present a comparison between our exhaustive method and a gradient boosting version that we call GB-CB-Boost. We show in the experiments (Sects. 5.1 and 5.2) that it has no significant advantage and it is less stable than CB-Boost. The details about the gradient boosting variant are explained in Appendix B, and a toy example gives an intuition on the difference between the two processes in Appendix C.

3.3 Presenting CB-Boost

Armed with the theoretical and practical results presented in the previous subsections, we are now ready to present the overall view of CB-Boost, which optimizes the training error of the majority vote through the iterative minimization of the mono-directional $\mathcal{C}$-bound presented in Theorem 3.

Algorithm 1 CB-Boost

Require: T # the maximum number of iterations
Require: $\mathcal{H} = h_1, \ldots, h_n$ #Precomputed set of classifiers
1: $\alpha_1, \ldots, \alpha_n \leftarrow 0, \ldots, 0$
2: $I_1, \ldots, I_T \leftarrow 0, \ldots, 0$ # The list containing the indices of the chosen hypothesis
3: $I_1 \leftarrow \arg\max_{k \in [n]} \gamma(h_k)$ # Find the index of the voter of $\mathcal{H}$ with the highest margin
4: $\alpha_{I_1} \leftarrow 1$ # Initialize its weight with 1
5: **for** t in $2, \ldots, T$ **do**
6: $\quad \alpha_1^*, \ldots, \alpha_n^* \leftarrow 0, \ldots, 0$
7: $\quad$ **for** k in $[n] \backslash \{I_{t-1}\}$ **do**
8: $\quad\quad$ **if** $\tau(F_k, h_k) < \frac{\gamma(F_k)}{\gamma(h_k)}$ **then**
9: $\quad\quad\quad \alpha_k^* \leftarrow \frac{\gamma(h_k)\nu(F_k) - \gamma(F_k)\tau(F_k, h_k)}{(\gamma(F_k) - \gamma(h_k)\tau(F_k, h_k))}$ # Find the optimal weight in every direction
10: $\quad\quad$ **end if**
11: $\quad$ **end for**
12: $\quad I_t \leftarrow \arg\min_{k \in [n] \backslash \{I_{t-1}\}} \mathscr{C}_k(\alpha_k^*)$ # Find the best direction's index
13: $\quad \alpha_{I_t} \leftarrow \alpha_{z_t}^*$
14: **end for**
15: **return** $\alpha_1, \ldots, \alpha_n$

For the sake of clarity, we define $I_1, \ldots, I_T$ as a list that is initialized with zeros (Line 2), and that contains each of the chosen directions' indices (Updated in Lines 3 and 12). To initialize CB-Boost, we use $h_{I_1} \in \mathcal{H}$ the hypothesis with the best margin, we set its weight to 1, and all the others to zero (Lines 1, 3 and 4). This aims at accelerating the convergence by starting the vote building with the strongest available hypothesis.

Then, for each iteration t, we compute the $\mathcal{C}$-bound-optimal weights in every available direction, by solving multiple mono-directional optimization problems (Lines 7 to 11). The direction is then exhaustively chosen (Line 12).

After the initialization, the weights on $\mathcal{H}$ are a Dirac distribution with the best-margin hypothesis's weight being the only one non-zero, and at each iteration t, one more element of $\mathcal{H}$ will have a non-zero weight α_{I_t}.

One major advantage of CB-Boost when compared to MinCq and CqBoost is the simplicity of Line 9, where its predecessors solve quadratic programs. Indeed, the algorithmic complexity of CB-Boost only depends on the number of iterations T, the number of examples m, and the number of hypotheses n. As the mono-directional $\mathcal{C}$-bound optimization is solved in $\mathcal{O}(n \times m)$ CB-Boost 's complexity is $\mathcal{O}(n \times m \times T)$.

3.4 Remarks

On the $\mathcal{C}$-bound indirect example re-weighting To bring diversity in the majority vote, Adaboost (Freund and Schapire 1997) updates weights over the examples at each iteration, exponentially emphasizing the examples on which it previously failed.

In CB-Boost, by considering both the first and second moments of the margin, the $\mathcal{C}$-bound takes into account the individual performance of each voter and their disagreement. Therefore, minimizing the $\mathcal{C}$-bound requires to keep a trade-off between maximizing the vote's margin and internal disagreement. This is the reason why CB-Boost does not include any example weighting. Indeed, the mono-directional $\mathcal{C}$-bound minimization problem is equivalent to minimizing the following quantity

$$\left(\sum_{i=1}^{m} F_k(x_i)^2 + 2\alpha \sum_{i=1}^{m} F_k(x_i)h_k(x_i) + \alpha^2 m\right) \frac{1}{\left(\gamma(F_k) + \alpha\gamma(h_k)\right)^2}\,.$$

Intuitively, in this expression, $\sum_{i=1}^{m} F_k(x_i)h_k(x_i)$ is equivalent to $\tau(F_k, h_k)$, so it decreases as h and F disagreement increases. It encourages CB-Boost to choose directions that perform well on hard examples. Moreover, α^2 can be interpreted as a regularization term and $\frac{1}{\left(\gamma(F_k)+\alpha\gamma(h_k)\right)^2}$ encapsulates the quality of the vote.

On the difference between the majority votes Intuitively, the concession made in CB-Boost to accelerate CqBoost and MinCq is focused on the weights of the majority vote. Indeed, CqBoost returns the majority vote that exactly minimizes the $\mathcal{C}$-bound, for the considered set of voters where CB-Boost returns sub-optimal weights because they have been optimized greedily throughout the iterations. Nevertheless, the $\mathcal{C}$-bound computed during the training phase is not an approximation for the considered majority vote, which explains the theoretical results of the next section. Moreover, in Fig. 5 (page 19), we empirically show that the weight-by-weight optimization has similar accuracy than the quadratic programs of MinCq and CqBoost.

On the stopping criterion In Sect. 3.1, we stated that as long as there is still a direction in which $\tau(F_k, h_k) < \frac{\gamma(F_k)}{\gamma(h_k)}$, the $\mathcal{C}$-bound can be optimized by CB-Boost. However, this is a very loose stopping criterion. i In fact, as experimentally seen in Sect. 5, as in CqBoost, it is far more interesting to use a fixed number of iterations as an hyper-parameter of the algorithm, as the main improvements are made during the first iterations of the algorithm. This way of restricting the number of iterations helps to reach a sparse model.

4 Theoretical results on training and generalization aspects

4.1 Quantifying the empirical $\mathcal{C}$-Bound decrease

In this section, we quantify the decrease rate of the empirical $\mathcal{C}$-bound for each iteration of CB-Boost, depending on the previous one and the considered direction.

Property 1 *During iteration t of CB-Boost, if I_t is the chosen direction's index, h_{I_t} its corresponding voter, and $F_{I_t} = \sum_{\substack{s=1 \\ s \neq I_t}}^{n} \alpha_s h_s$ the majority vote of all the other directions, then the empirical $\mathcal{C}$-bound decreases exactly by*

$$\mathscr{S}_t = \frac{\left(\gamma(h_{I_t})\nu\left(F_{I_t}\right) - \gamma(F_{I_t})\tau(F_{I_t}, h_{I_t})\right)^2}{\nu\left(F_{I_t}\right)\left(\nu\left(F_{I_t}\right) - \tau(F_{I_t}, h_{I_t})^2\right)} > 0\,.$$

The proof is provided in the Appendix, in Sect. D.4.

4.2 Deriving the training error bound

Corollary 2 *The training error of the majority vote, built by CB-Boost at iteration $t > 2$ is bounded by*

$$1 - \frac{1}{m} \frac{\left(\sum_{i=1}^{m} y_i \left[h_{I_1}(x_i) + \alpha_{I_2} h_{I_2}(x_i) \right] \right)^2}{\sum_{i=1}^{m} \left(y_i \left[h_{I_1}(x_i) + \alpha_{I_2} h_{I_2}(x_i) \right] \right)^2} - \sum_{j=3}^{t} \mathscr{S}_j ,$$

with $\mathscr{S}_j$ being the quantity introduced in Property 1.

The proof is straightforward by combining Corollary 1 and Property 1.

This training error bound allows us to assess CB-Boost 's capacity to learn relevant models based on the available pool of voters.

4.3 Generalization guarantees

Theorem 2 presents a PAC-bound that gives generalization guarantees based on the empirical $\mathcal{C}$-bound. In order to apply it to CB-Boost 's output $F = \sum_{s=1}^{n} h_s \alpha_s$, we use $Q = \{ \frac{\alpha_1}{\sigma}, \ldots, \frac{\alpha_n}{\sigma} \}$ with $\sigma = \sum_{s=1}^{n} \alpha_s$. Note that only the T weights corresponding to the chosen directions are non-zero. So, according to Theorem 2, with probability $1 - \delta$, for any sample $\mathcal{S}$ drawn according to D,

$$R_D(F) \leq 1 - \frac{\left(max\left(0, \frac{1}{m} \sum_{i=1}^{m} y_i \mathop{\mathbf{E}}_{h \sim Q} h(x_i) - \sqrt{\frac{2}{m} \left[KL(Q||P) + \ln\left(\frac{2\sqrt{m}}{\delta/2} \right) \right]} \right) \right)^2}{min\left(1, \frac{1}{m} \sum_{i=1}^{m} \left(y_i \mathop{\mathbf{E}}_{h \sim Q} h(x_i) \right)^2 + \sqrt{\frac{2}{m} \left[2KL(Q||P) + \ln\left(\frac{2\sqrt{m}}{\delta/2} \right) \right]} \right)},$$

These guarantees are tighter when the empirical $\mathcal{C}$-bound of a majority vote is small, which is exactly what CB-Boost aims at returning. Moreover, as seen in Sect. 2.2, returning a majority vote with a small $KL(Q||P)$ is essential in order to have good generalization guarantees. In Roy et al. (2016), the authors established that if the number of voters is far lower than the number of examples, $n << m$, then minimizing $KL(Q||P)$ is negligible in comparison with minimizing the $\mathcal{C}$-bound of the majority vote.

If the case $n << m$ is not applicable, we need to characterize $KL(Q||P)$ intuitively. We use a uniform prior on $\mathcal{H}$, and as at each iteration of CB-Boost, one more weight of Q will be non-zero, $KL(Q||P)$ will increase as the posterior's number of non-zero weight augments. Moreover, we proved that the $\mathcal{C}$-bound of Q decreases over the iterations of CB-Boost. Thus, in order to keep the trade-off between $KL(Q||P)$ and the $\mathcal{C}$-bound for the bound of Theorem 2, it is relevant to use early-stopping by choosing a maximal number of iterations. Consequently, based on the generalization guarantees, we set the maximum number of iterations on CB-Boost as an hyper-parameter that can be chosen using hold-out data.

Comparison to non PAC-Bayesian generalization bounds In Cortes et al. (2014), a tight bound based on the Rademacher complexity is given for majority votes F of $\text{Conv}(\mathcal{H})$. This bound depends on $\hat{R}_{\mathcal{S},\rho}(F) = \underset{(x,y)\sim\mathcal{S}}{\mathbf{E}}[1_{yF(x)\leq\rho}]$ the training error of F's margin being lower than ρ, over the sample $\mathcal{S}$ drawn according to D and $\mathfrak{R}_m(\mathcal{H})$ the Rademacher complexity of $\mathcal{H}$, as

$$R_D(F) \leq \hat{R}_{\mathcal{S},\rho}(F) + \frac{4}{\rho}\sum_{t=1}^{T}\alpha_t\mathfrak{R}_m(\mathcal{H}) + \frac{2}{\rho}\sqrt{\frac{\ln n}{m}} + \sqrt{\left\lceil\frac{4}{\rho^2}\ln\left[\frac{\rho^2 m}{\ln n}\right]\right\rceil\frac{\ln n}{m} + \frac{\ln\frac{2}{\delta}}{2m}}\,.$$

As explained in the previous paragraph, the size of $KL(Q\|P)$ is either negligible or depends on the complexity of the distribution Q, and then can be reduced by early stopping. So, intuitively, both the expressions rely on $\sqrt{\frac{1}{m}}$ and are then fairly equivalent. The main difference is that the Rademacher-based bound relies on the training error of the majority vote and the Rademacher complexity of the hypothesis space $\mathcal{H}$ (in our case $\mathfrak{R}_m(\mathcal{H}) \leq \sqrt{\frac{2\ln(|\mathcal{H}|)}{m}}$, as our classifiers only outputs -1 or 1) whereas the $\mathcal{C}$-bound 's PAC-Bayesian bound relies on maximizing $\frac{\mu_1(M_Q^{\mathcal{S}})}{\mu_2(M_Q^{\mathcal{S}})}$, which CB-Boost is explicitly processing.

5 Experiments [4]

In order to experimentally study CB-Boost, we first compare it to CqBoost (Roy et al. 2016), MinCq (Germain et al. 2015) and GB-CB-Boost (Sect. 3.2), focusing on efficiency, sparsity and performance on chosen datasets featuring various properties. Then, we compare it to Adaboost (Freund and Schapire 1997) and other ensemble methods: Random Forest (Breiman 2001), Bagging (Breiman 1996), and Gradient Boosting (Friedman 2001).

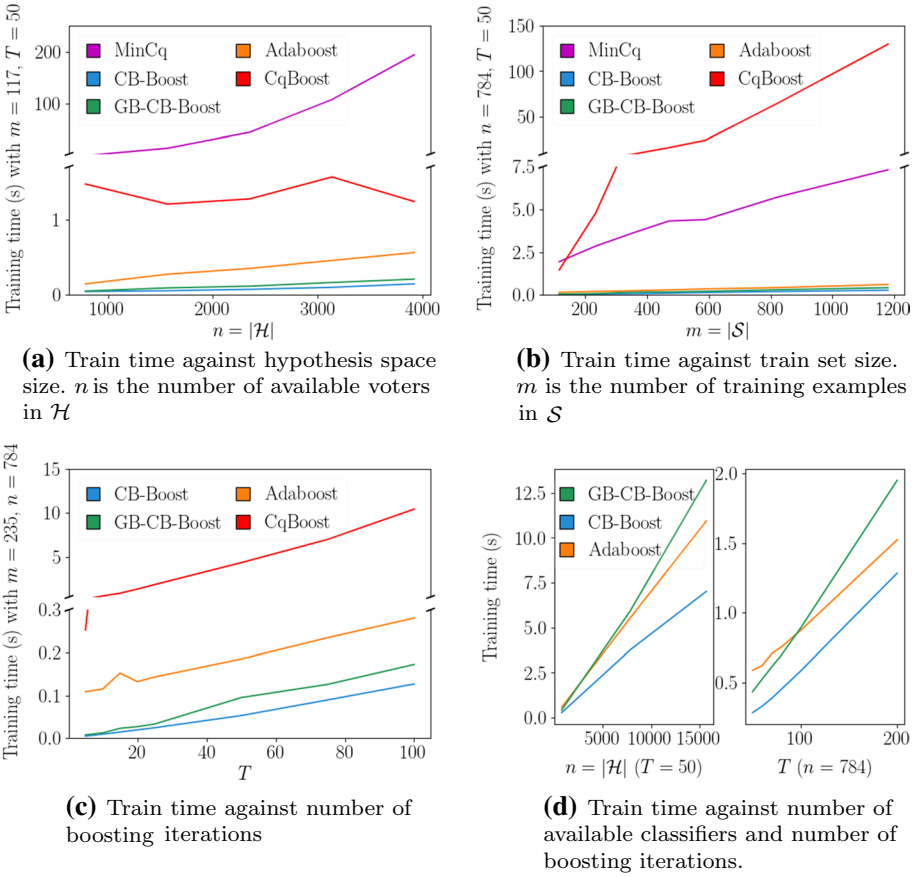
5.1 Computational time improvement

In order to highlight the main advantage of CB-Boost, we first analyze the computational time improvement obtained by greedily minimizing the $\mathcal{C}$-bound, comparing CB-Boost and GB-CB-Boost to CqBoost's and MinCq's quadratic programs. Moreover to compare them with a fast, broadly used boosting algorithm, we challenge CB-Boost 's computational efficiency with Adaboost's.

Protocol In this experiment, we want to compare MinCq, CqBoost , GB-CB-Boost, CB-Boost and Adaboost by varying three factors : the training set size, the hypothesis set size, and the number of boosting iterations, respectively denoted, m, n and T. To do so, we use MNist 4vs9 (LeCun and Cortes 2010) as it provides 11791 examples and 784 features. Moreover, to be as fair as possible with the quadratic programs, we allowed 8 threads for the solver. For comparative purposes, the only difference between GB-CB-Boost and CB-Boost 's implementations is the gradient computation, as they are based on the same code.

Results In Fig. 1a, we vary n, the number of available hypotheses, with constant m and T and, as expected, MinCq has a very long computational time. Moreover, even if CqBoost's

[4] The code used to realize each of the following experiments is available here: https://gitlab.lis-lab.fr/baptiste.bauvin/cb-boost-exps.



(a) Train time against hypothesis space size. n is the number of available voters in $\mathcal{H}$

(b) Train time against train set size. m is the number of training examples in $\mathcal{S}$

(c) Train time against number of boosting iterations

(d) Train time against number of available classifiers and number of boosting iterations.

Fig. 1 Efficiency of $\mathcal{C}$-bound algorithms and Adaboost on MNist 0v9. For **a**, **b**, **c**, we broke the ordinate axis, to highlight the difference between the fast algorithms while showing the most time consuming algorithms

boosting-like structure reduce its time consumption, it is still much longer than the three other algorithms.

In Fig. 1b, we vary m. It has a small effect on MinCq, but has a huge effect on CqBoost as it runs $T = 50$ iterations in which it solves one increasingly complex quadratic programs per iteration, whereas MinCq has only one. Here, the apparently small difference between the other algorithms is due to CqBoost's and MinCq's long duration.

Figure 1c only plots the iterative algorithms, and the duration difference between CqBoost and the greedy ones is clear even for this small learning set.

To sum up the results of these three sub-experiments, Adaboost, CB-Boost and GB-CB-Boost are far faster than the other $\mathcal{C}$-bound algorithms. However, Adaboost and GB-CB-Boost seem to be slightly more time consuming than CB-Boost. To compare them, we analyze Fig. 1d, showing that even though they are close, CB-Boost is constantly faster than the others. Moreover, it is visible that GB-CB-Boost 's gradient computation is more time consuming than CB-Boost 's exhaustive search.

To conclude, CB-Boost is a substantial acceleration of CqBoost and MinCq, and is faster than Adaboost and GB-CB-Boost, which means that it is able to scale up to much bigger datasets than the other $\mathcal{C}$-bound-based algorithms.

5.2 Performance comparison: $\mathcal{C}$-bound-based algorithms equivalence

Here, we compare CB-Boost 's performance in zero-one loss with the other $\mathcal{C}$-bound minimizers. Our goal is to measure the potential decrease in accuracy that greedy minimization would imply.

Protocol

- *Datasets* All the datasets used in this experiment are presented in Table 1, an in-depth presentation is made in Appendix E.
- *Classifiers* For each classifier, hyper-parameters were found with a randomized search with 30 draws, over a distribution that incorporates prior knowledge about the algorithm, but is independent from the dataset. They were validated by a 5-folds cross-validation and each experiment was run ten times, with the result in Table 2 being the mean and standard deviation over them. These results are not statistically significant, but multiplying the experiments helps avoiding an outlier split that could bias the results.

Results Table 2 shows that, for all datasets except Ionosphere and MNist-5v6, CB-Boost is performing at least as well as the other $\mathcal{C}$-bound algorithms, considering the standard deviation. It is quite clear that MinCq has the best zero-one loss on nearly all the datasets, as it uses all the available hypotheses. However, CB-Boost is competitive with both CqBoost and MinCq, and is even better on three datasets. On the more

Table 1 Datasets used in the different experiments

Source	Name	Size	Dim.	Train size (%)	Test size (%)
UCI Dua and Graff (2017)	**Australian**	690	14	345 (50)	345 (50)
	Balance	625	4	312 (50)	313 (50)
	Bupa	345	6	172 (50)	173 (50)
	Cylinder	540	35	270 (50)	270 (50)
	Hepatitis	155	19	77 (50)	78 (50)
	Ionosphere	351	34	175 (50)	176 (50)
	Pima	768	8	384 (50)	384 (50)
	Yeast	1484	8	742 (50)	742 (50)
MNIST LeCun and Cortes (2010)	**MNist-0v9**	11872	784	474 (4)	11398 (96)
	MNist-6v5	11339	784	453 (4)	10886 (96)
	MNist-5v3	11552	784	462 (4)	11090 (96)
	MNist-7v9	12214	784	488 (4)	11726 (96)
AwA Lampert et al. (2009)	**Awa-TvW**	1092	2000	546 (50)	546 (50)

UCI datasets are used as simple tasks, MNist datasets are intended to represent usual complex problems with numerous examples and medium dimensionality—we select usually confused binary tasks from it –, and AwA is supposed to represent a complex problem of high dimensionality. See Appendix E for more details

Table 2 Zero one loss on test

	CB-Boost	CqBoost	MinCq	GB-CB-Boost
Australian	**0.143 ± 0.009**	**0.144 ± 0.014**	**0.141 ± 0.008** *	**0.197 ± 0.109**
Bupa	**0.335 ± 0.033**	**0.332 ± 0.032** *	**0.350 ± 0.039**	**0.364 ± 0.037**
Cylinder	**0.270 ± 0.021**	**0.266 ± 0.027**	**0.261 ± 0.019** *	**0.285 ± 0.035**
Hepatitis	**0.382 ± 0.019**	**0.390 ± 0.053**	**0.342 ± 0.043** *	**0.35 ± 0.041**
Ionosphere	0.160 ± 0.014	**0.140 ± 0.021**	**0.132 ± 0.019** *	0.169 ± 0.01
Yeast	**0.299 ± 0.009**	**0.294 ± 0.018**	**0.289 ± 0.017** *	**0.297 ± 0.013**
Balance	**0.055 ± 0.016**	**0.053 ± 0.020**	**0.053 ± 0.014** *	0.071 ± 0.016
Pima	**0.235 ± 0.018** *	**0.242 ± 0.015**	0.250 ± 0.016	**0.24 ± 0.01**
MNist-0v9	**0.012 ± 0.001** *	0.015 ± 0.002	**0.012 ± 0.002**	**0.015 ± 0.004**
MNist-5v3	**0.083 ± 0.008**	**0.080 ± 0.009**	**0.076 ± 0.008** *	**0.081 ± 0.003**
MNist-5v6	0.041 ± 0.003	0.040 ± 0.002	**0.036 ± 0.003** *	**0.055 ± 0.038**
MNist-7v9	**0.073 ± 0.004**	**0.072 ± 0.005**	**0.071 ± 0.002** *	**0.091 ± 0.029**
Awa-TvW	**0.081 ± 0.001** *	0.093 ± 0.003	**0.09 ± 0.011**	0.091 ± 0.009

A result in bold means that, considering the standard deviation, the algorithm performed as well as the best mean. A starred result means that the classifier returned the best mean

complex Animals with Attributes, CB-Boost is the best of the $\mathcal{C}$-bound algorithms. As expected, GB-CB-Boost shows no significant improvement of the accuracy, and is even less stable on australian, MNist-7v9 and 5v6.

CB-Boost is competitive with both the quadratic $\mathcal{C}$-bound minimization algorithms 11 times out of 13, and has the best mean 3 times. Considering its shorter computational time, it is more efficient at extracting the information.

5.3 Sparsity conservation

Protocol We analyze here the convergence speed of the five previously studied algorithms with grid-searched optimal hyper-parameters, when needed (MinCq's margin hyper-parameter set to 0.05, and CqBoost's margin and error parameters, respectively set to 0.001 and 10^{-6}). The chosen dataset is MNIST 0v9 as it is more complex than UCI datasets but still small enough for CqBoost and MinCq to have reasonable computing time on it. We use the same hypotheses, training and testing sets as previously.

Results Figure 2 shows that, even if CB-Boost converges a bit less quickly than Adaboost and CqBoost on the training set, its performance on the test set is slightly better than all the other algorithms. Moreover, it is visible that CqBoost's and CB-Boost 's sparsity prevent them from the slight overfitting of MinCq. Furthermore, CqBoost seems to profit more from early-stopping than CB-Boost, even if their sparsity is clear as they have a better test performance than MinCq from the 30th voter. Finally, the experiments do not reveal any visible difference in sparsity between CB-Boost and GB-CB-Boost.

So, thanks to the closed form solution used in the calculation of CB-Boost, its computing time will always have an edge when compared to GB-CB-Boost. This, together with the fact that, as expected (and empirically shown), GB-CB-Boost is less stable, leads us to concentrate our analysis to CB-Boost for the remaining of the paper.

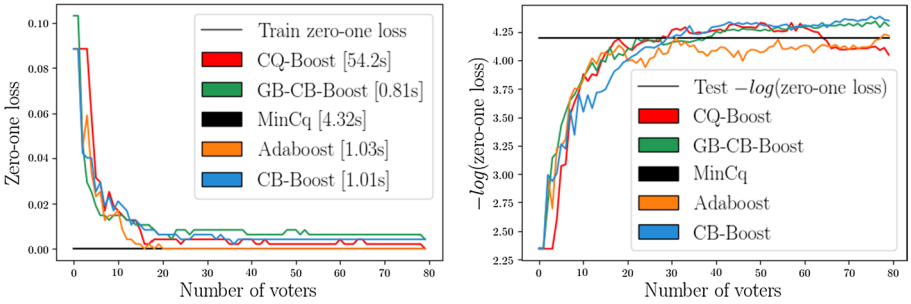


Fig. 2 Zero-one loss on train (left) and $-\log$(*zero-one loss*) on test (right) throughout the learning process on MNist 0v9

5.4 Performance comparison: noise robustness equivalence

In this section, we compare CB-Boost and Adaboost considering accuracy. The aim here is (1) to quantify how far CB-Boost is from Adaboost and (2) to evaluate their compared robustness to noise. Please note that the version of Adaboost used for these experiments is Adaboost.SAMME (Zhu et al. 2006), which is more robust to noise than the original.

Protocol We use the same framework as in Sect. 5.2 regarding datasets, train-test splits and hyper-parameter optimization. A Gaussian distribution is used to add noise to the data (see Appendix F.1 for a detailed protocol and an example). The MNist dataset requires a special pre-processing to reduce its contrast in order for the noise to have an effect on classification. See Appendix F.2 for details.

Results In Fig. 3, we show a view of the results with each matrix presenting the comparison between Adaboost and CB-Boost for each dataset (rows), and noise level

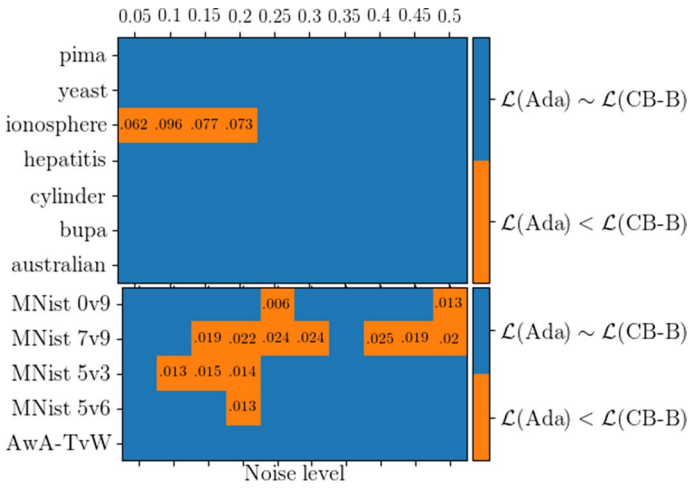


Fig. 3 Zero-one loss of CB-Boost and Adaboost. For each dataset (rows), we used several levels of noise (columns). An orange square means that Adaboost is better than CB-Boost in zero-one-loss ($\mathcal{L}$(Ada) < $\mathcal{L}$(CB-B)), and the difference between their scores is printed inside. A blue square means that they are equivalent, when considering their standard deviation

(columns). It shows that, except for the four first noise levels of ionosphere where the difference is noteworthy, no significant loss of accuracy is observed when using CB-Boost, as the maximum difference between the other scores is 2.5%. In Appendix F.3, the numerical results are provided.

5.5 Performance comparison: ensemble methods equivalence

We present the result of the performance comparison between CB-Boost and four other ensemble methods : Adaboost (Freund and Schapire 1997), Bagging (Breiman 1996), Random Forests (Breiman 2001), and Gradient Boosting (a) for which we use the same protocol as in Sect. 5.4. This experiment differs from Sect. 5.4 in the fact that we do not mean the results on each dataset but instead, in Fig. 4 we plot one dot for each train/test split on each dataset, for a matter of legibility.

Results In Fig. 4, the distance to the $x = y$ line represents the difference in zero-one loss. So, a dot under the line means that the ensemble method has lower loss and one over the line means it has higher loss. One can see that CB-Boost is similar to the state of the art for most of the datasets and train/test splits. The only perceptible tendency is that Bagging is frequently worse than CB-Boost, the other methods are equivalent to CB-Boost.

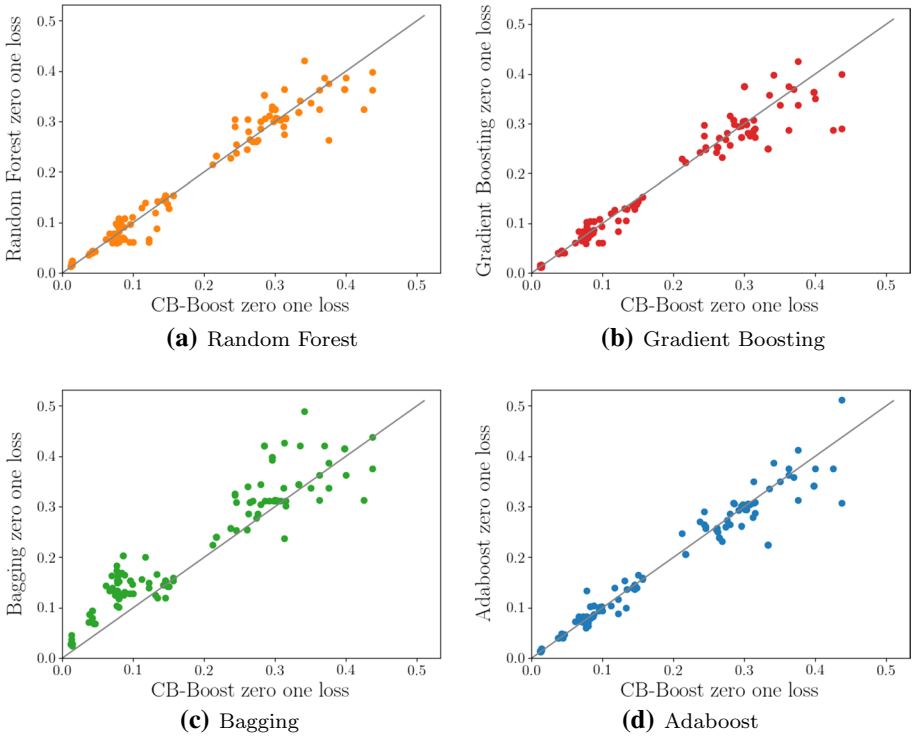


Fig. 4 Each ensemble method's zero-one loss against CB-Boost 's one on the test set. A dot represents one of the 10 splits of each dataset of Table 1

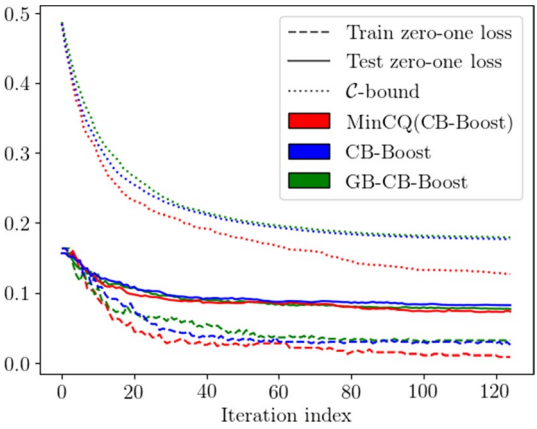


Fig. 5 Zero-one-loss and $\mathcal{C}$-bound for CB-Boost and the optimal $\mathcal{C}$-bound version MinCq(CB-Boost) on MNist 4v9. The dotted, dashed and plain lines represent respectively the $\mathcal{C}$-bound, training and testing error

5.6 $\mathcal{C}$-bound approximation characterization

Here, we aim at empirically analyzing the approximation made by greedily minimizing the $\mathcal{C}$-bound instead of using a quadratic program solver. To do so, at each iteration of CB-Boost, we run MinCq on the selected subset of voters to compute its optimal $\mathcal{C}$-bound.

Protocol We use the MNist 4v9 dataset with the same sets as previously and predefined hyper-parameters: CB-Boost 's maximum number of iterations is set to 200, and MinCq's margin parameter to 0.05.

Results Figure 5 shows that there is a slight, but noticeable difference between the $\mathcal{C}$-bound of CB-Boost 's majority vote and MinCq's one. However, this difference only has a small impact on CB-Boost 's performance for the first 30 iterations in train and 80 iterations in test. So even if the expected difference in $\mathcal{C}$-bound optimality is noteworthy, it does not impact the performance of CB-Boost. Finally, the small gap between both the $\mathcal{C}$-bounds empirically suggests that CB-Boost keeps the qualities of CqBoost and MinCq.

6 Conclusion

In this paper, we presented CB-Boost, a greedy $\mathcal{C}$-bound minimization algorithm. While maintaining its predecessors' sparsity and accuracy properties, it has much lighter computational demands. Its optimization process relies on a theoretical result allowing CB-Boost to efficiently minimize the $\mathcal{C}$-bound in one direction at a time. This algorithm keeps the training and generalization guarantees given by the $\mathcal{C}$-bound (Lacasse et al. 2006) and has the interesting property to allow a quantification of the decrease of its bound and training error.

Experimentally, the comparison of CB-Boost with relevant methods shows its real improvement in computational demand, without loss of accuracy. Furthermore, experiments shows that CB-Boost slightly improves the sparsity of the models, which is the main property of CqBoost. Finally, it is competitive with four state of the art ensemble methods with regards to performance, and with Adaboost in computational efficiency, sparsity and noise robustness.

In future work, we will analyze deeper theoretical properties of CB-Boost, focusing on its dual form, finding a stronger stopping criterion and adapting CB-Boost to infinite hypothesis spaces.

Acknowledgements This work has been supported by National Science and Engineering Research Council of Canada (NSERC) Discovery grant 262067 and by the French National Research Agency (grant ANR-15-CE23-0026). We warmly thank Robert Sadler and Sokol Koço for their proofreading.

A Appendix: algorithms

Here, we briefly introduce MinCq (Germain et al. 2015) and CqBoost (Roy et al. 2016). The notation is adapted to use our paper's instead of the original authors' to avoid any confusion.

- $\mathbf{M} \triangleq \begin{bmatrix} f_1(x_1) & f_2(x_1) & \dots & f_{2n}(x_1) \\ f_1(x_2) & f_2(x_2) & \dots & f_{2n}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_m) & f_2(x_m) & \dots & f_{2n}(x_m) \end{bmatrix}$, as the vote matrix, containing the vote of each weak classifier in $\mathcal{H}$ and its complementary $f_{n+j} = -f_j$, $j = 1..n$ on each example x_i, $i = 1..m$ of the training set $\mathcal{S}$, it is the matrix representing $\mathcal{H}$,
- $\mathbf{q}$ denotes the weight vector on the voters, corresponding to the weights α in CB-Boost,
- $\mathbf{1}_m$ denotes the unitary vector of size m,
- μ denotes the exact margin of the majority vote and $\tilde{\mu}$, its minimum margin,
- ω denotes a weight vector over the samples,
- ϵ denotes a small positive real.

A.1 MinCq

MinCq is simply a bound-minimizer, and therefore, its pseudo code is solving a quadratic program to output a weight vector on the hypothesis space, for details on the problem itself, see (Germain et al. 2015).

Algorithm 2 MinCq

1: **Solve**

$$\begin{aligned} \underset{\mathbf{q}}{\text{argmin}} \quad & \frac{1}{m}\mathbf{q}^\top \mathbf{M}^\top \mathbf{M}\mathbf{q} \\ \text{subject to} \quad & \frac{1}{m}\mathbf{y}^\top \mathbf{M}\mathbf{q} = \mu, \\ & [\mathbf{I}_n \mathbf{I}_n]\,\mathbf{q} = \mu, \quad \mathbf{q} \geq \mathbf{0}_{2n}. \end{aligned}$$

2: **return** $\mathbf{q}$

A.2 CqBoost

CqBoost is a bit more complex than MinCq, so we will analyze the main steps of the pseudocode :

- Lines 1 to 3 initialize the problem with a null weight vector over the voters, a uniform distribution on the examples.
- For each iteration,
  - Line 5 finds the voter with the best weighted edge, the edge being the dual of the margin, it allows to find the voter that has the best decision, given the weights of each example.
  - And Line 10 solves a $\mathcal{C}$-bound minimization problem to update the weights on the voters and examples.

Let us introduce the quadratic program that is solved at Line 10,

$$\begin{aligned} \underset{\mathbf{q},\rho}{\text{argmin}} \quad & \frac{1}{m}\rho^\top \rho \\ \text{subject to} \quad & \rho = \text{diag}(\mathbf{y})\tilde{\mathbf{M}}\mathbf{q}, \quad \frac{1}{m}\mathbf{1}_m^\top \rho \geq \tilde{\mu}, \\ & \mathbf{q} \geq \mathbf{0}_n, \quad \mathbf{1}_n^\top \mathbf{q} = 1. \end{aligned} \tag{1}$$

And its dual with ω, β, ν being Lagrange multipliers

$$\begin{aligned} \underset{\boldsymbol{\omega},\beta,\nu}{\text{argmin}} \quad & \frac{m}{4}\boldsymbol{\omega}^\top \boldsymbol{\omega} - \frac{\beta}{2}\mathbf{1}_m^\top \boldsymbol{\omega} + \frac{\beta^2}{4} - \beta\tilde{\mu} + \nu \\ \text{subject to} \quad & \tilde{\mathbf{M}}^\top \text{diag}(\mathbf{y})\boldsymbol{\omega} \leq \nu \mathbf{1}_n, \\ & \beta \geq 0. \end{aligned} \tag{2}$$

Algorithm 3 CqBoost

Require: T, number of iterations
1: $\mathbf{q} \leftarrow \mathbf{0}_n$
2: $\boldsymbol{\omega} \leftarrow \frac{1}{m}\mathbf{1}_m$
3: $\tilde{\mathbf{M}} \leftarrow$ empty matrix
4: **for** $t \leftarrow 1, \ldots, T$ **do**
5: $\quad i \leftarrow \text{argmax}_i \sum_{k=1}^m \omega_k y_k M_{ki}$
6: $\quad$ **if** $\sum_{k=1}^m \omega_k y_k M_{ki} \leq v + \epsilon$ **then**
7: $\quad\quad$ Leave loop
8: $\quad$ **end if**
9: $\quad$ Update $\tilde{\mathbf{M}}$ with $\mathbf{M}$'s i-th column
10: $\quad \mathbf{q}, \boldsymbol{\omega}, \nu \leftarrow$ solution of Eq. 1 or 2, considering $\tilde{\mathbf{M}}$(of size $m \times t$)
11: **end for**
12: **return** $\mathbf{q}$

Algorithm 4 Gradient boosting version of CB-Boost

Require: T # the maximum number of iterations
Require: $\mathcal{H} = h_1, \ldots, h_n$ #Precomputed set of classifiers
$\alpha_1, \ldots, \alpha_n \leftarrow 0, \ldots, 0$
$I_1, \ldots, I_T \leftarrow 0, \ldots, 0$ # The list containing the indices of the chosen hypothesis
$I_1 \leftarrow \arg\max_{k \in [n]} \gamma(h_k)$ # Find the index of the voter of $\mathcal{H}$ with the highest margin
$\alpha_{I_1} \leftarrow 1$ # Initialize its weight with 1
for t in $2, \ldots, T$ **do**
  $I_t \leftarrow \arg\max_{k=1..n} -\nabla\mathscr{C}(F_k) \cdot h_k$ # Find the best-gradient direction
  $\alpha_{I_t} \leftarrow \frac{\gamma(h_{I_t})\nu(F_{I_t}) - \gamma(F_{I_t})\tau(F_{I_t}, h_{I_t})}{(\gamma(F_{I_t}) - \gamma(h_{I_t})\tau(F_{I_t}, h_{I_t}))}$ # Find the optimal weight in the direction
end for
return $\alpha_1, \ldots, \alpha_n$

B Appendix: gradient boosting comparison

B.1 Getting a gradient boosting version of CB-Boost

To build the gradient boosting version of CB-Boost, we follow the method presented in Schapire and Freund (2012), with the $\mathcal{C}$-bound as the choice of loss function, defined for any $F \in \text{Conv}(\mathcal{H})$ as

$$\mathscr{C}(F) = 1 - \frac{1}{m} \frac{\left(\sum_{i=1}^{m} y_i F(x_i)\right)^2}{\sum_{i=1}^{m} \left(y_i F(x_i)\right)^2}.$$

The gradient $\nabla\mathscr{C}(F) = \left(\frac{\partial\mathscr{C}(F(x_1))}{\partial F(x_1)}, \ldots, \frac{\partial\mathscr{C}(F(x_i))}{\partial F(x_i)}, \ldots, \frac{\partial\mathscr{C}(F(x_m))}{\partial F(x_m)}\right)$ is given, $\forall i \in [1..m]$, by

$$\begin{aligned}
\frac{\partial \mathscr{C}\big(F(x_i)\big)}{\partial F(x_i)} &= \frac{\partial}{\partial F(x_i)}\left(1 - \frac{1}{m}\frac{\left(\sum_{j\neq i} y_j F(x_j) + F(x_i)y_i\right)^2}{\sum_{j\neq i}\big(y_j F(x_j)\big)^2 + (F(x_i)y_i)^2}\right) \\
&= \frac{1}{m}\frac{2y_i\left[\sum_{j\neq i} y_j F(x_j) + F(x_i)y_i\right]\left[\left(\sum_{j\neq i} y_j F(x_j)\right)F(x_i)y_i - \sum_{j\neq i}\big(y_j F(x_j)\big)^2\right]}{\left[\sum_{j\neq i}\big(y_j F(x_j)\big)^2 + (F(x_i)y_i)^2\right]^2} \\
&= \frac{2y_i\gamma(F)\big(\gamma(F)F(x_i)y_i - \nu(F)\big)}{\nu(F)^2}\,.
\end{aligned}$$

Given this result, at iteration t of the gradient boosting algorithms finds the following direction of optimization, with F_k and h_k defined as in Sect. 3.1:

$$\begin{aligned}
I_t &= \arg\max_{k=1..n} -\nabla\mathscr{C}(F_k)\cdot h_k \\
&= \arg\max_{k=1..n} -\sum_{i=1}^{m}\frac{\partial\mathscr{C}\big(F_k(x_i)\big)}{\partial F_k(x_i)}h_k(x_i)
\end{aligned}$$

Once this direction is found, the next goal is to find the best weight for the voter (or optimization step). Here, thanks to Theorem 3, the optimal weight is given by

$$\alpha^*_{I_t} = \frac{\gamma(h_{I_t})\nu\big(F_{I_t}\big) - \gamma(F_{I_t})\tau(F_{I_t}, h_{I_t})}{\big(\gamma(F_{I_t}) - \gamma(h_{I_t})\tau(F_{I_t}, h_{I_t})\big)}\,.$$

Armed with these results, we present GB-CB-Boost in Algorithm 4 as the gradient boosting variant of CB-Boost.

B.2 Difference between CB-Boost and GB-CB-Boost

The only difference between the two greedy algorithms is the choice of the optimization direction, made in step 12 of Algorithm 1:

$$I_t \leftarrow \arg\min_{k\in[n]\setminus\{I_{t-1}\}} \mathscr{C}_k(\alpha^*_k)$$

and step 4 of Algorithm 4:

$$I_t \leftarrow \arg\max_{k=1..n} -\nabla\mathscr{C}(F_k)\cdot h_k\,.$$

In CB-Boost, the direction is chosen as the one in which the $\mathcal{C}$-bound has the lowest minimum whereas in GB-CB-Boost, the choice is based on the maximum of the negative gradient.

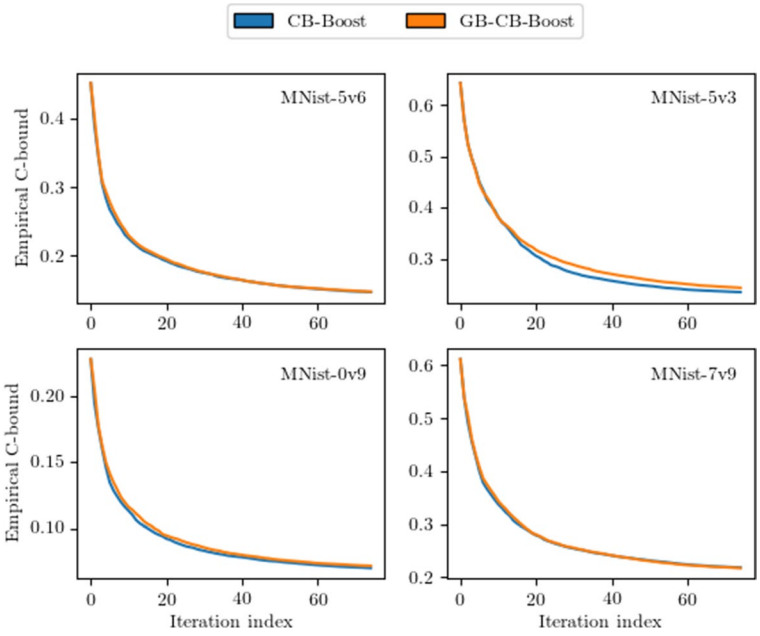


Fig. 6 The empirical (training) $\mathcal{C}$-bound for CB-Boost and GB-CB-Boost on the four MNist datasets presented in Table 1 over 75 boosting iterations

As seen in Sects. 5.1 and 5.2, it has an empirical impact on the computational efficiency and on the stability of the algorithm, as CB-Boost is faster and more stable than its gradient boosting counterpart. However, it does not have a significant impact on performance nor sparsity.

Concerning sheer $\mathcal{C}$-bound optimization, Fig. 6 shows that CB-Boost is as optimal as gradient boosting, and even slightly better, particularly on MNist-5v3. Therefore, it would be interesting to lead a more in-depth study on the differences between these variants. For example, in Appendix C, a toy example points a case where gradient boosting needs one more optimization step than our method to find the minimum of a toy loss.

C Appendix: intuitive understanding of direction choice on a toy example

Here, we present a toy example that is meant to give the reader an intuition on the difference between gradient boosting bound minimization, and CB-Boost 's $\mathcal{C}$-bound minimization process. To do so, we use a toy loss of two variables $\mathcal{L} : x, y \mapsto z$ and analyze how the two approaches tackle the optimization problem. In Fig. 7, we plot it in two dimensions, a 3D plot is available here.[5] This loss is convex and defined as

[5] https://pageperso.lis-lab.fr/baptiste.bauvin/toy_example.

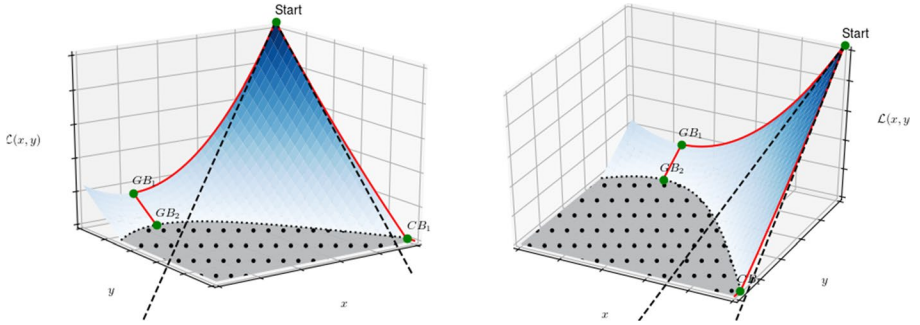


Fig. 7 Toy convex loss in two dimensions in blue, paths of the algorithms in red (GB for gradient boosting and CB for CB-Boost). The dotted surface is the minimum of the loss and dashed lines are the gradients directions

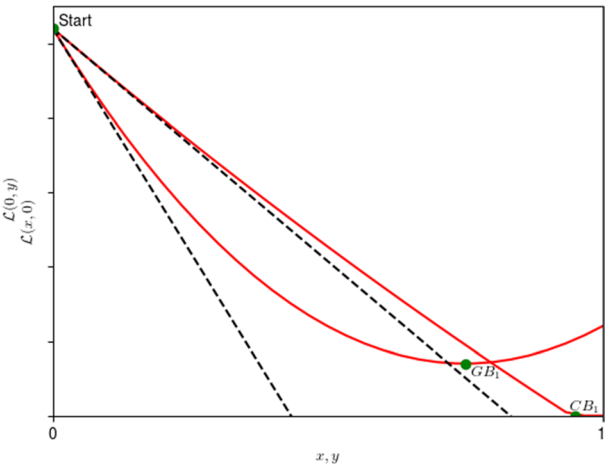


Fig. 8 2D Projection of Fig. 7 gradient directions, the dashed lines represent the gradient at the starting point, and the red line represents the mono-directional optimization path that is available for the algorithm. The green dots represent the optimal first step for gradient boosting (GB_1) and CB-Boost (CB_1)

$$\mathcal{L}(x, y) = \begin{cases} \big((x + c_0)(y + c_1)\big)^2 + (y + c_1)^2 - c_2 \textbf{ if } x^2 + 2xc_0 + c_0^2 + 1 > \frac{c_2}{(y+c_1)^2} \\ 0 \text{ otherwise,} \end{cases}$$

with c_0, c_1, c_2 constants. In Fig. 7, we can see that when starting at the same place on the loss function's surface (the "Start" point), Gradient Boosting (GB) will choose the direction where the gradient is the lowest (x), and even with a perfectly tuned step size, will have to use two iterations to reach the loss' minima, as in this direction, it is not attainable. However as CB-Boost 's strategy is to find the direction in which the attainable minimum is the lowest (y), regardless of the gradient, CB-Boost will find one minimum in just one iteration. In Fig. 8, we projected the gradient directions and the function in 2D, in order to have a clearer point of view.

This example is not meant to prove a theoretical nor empirical superiority but to point the difference between the two methods.

D Appendix: proofs

D.1 Proof of the equivalence between the distribution, and the weights version of the $\mathcal{C}$-bound

Let us consider a set of weights $\{\alpha_1, \dots, \alpha_n\}_n \in (\mathbb{R}^+)^n$ that do not sum to one, and the distribution version $Q' = \{\frac{\alpha_1}{\sigma}, \dots, \frac{\alpha_n}{\sigma}\}$ with $\sigma = \sum_{s=1} \alpha_s$.

$$
\begin{aligned}
\mathscr{C}^{\mathcal{S}}_{\{\alpha_1,\dots,\alpha_n\}} &= 1 - \frac{1}{m} \frac{\left(\sum_{i=1}^{m} y_i \sum_{s=1}^{n} \alpha_s h_s(x_i)\right)^2}{\sum_{i=1}^{m} \left(y_i \sum_{s=1}^{n} \alpha_s h_s(x_i)\right)^2} \\
&= 1 - \frac{1}{m} \frac{\frac{1}{\sigma^2}\left(\sum_{i=1}^{m} y_i \sum_{s=1}^{n} \alpha_s h_s(x_i)\right)^2}{\frac{1}{\sigma^2}\sum_{i=1}^{m} \left(y_i \sum_{s=1}^{n} \alpha_s h_s(x_i)\right)^2} \\
&= 1 - \frac{1}{m} \frac{\left(\sum_{i=1}^{m} y_i \sum_{s=1}^{n} \frac{\alpha_s}{\sigma} h_s(x_i)\right)^2}{\sum_{i=1}^{m} \left(y_i \sum_{s=1}^{n} \frac{\alpha_s}{\sigma} h_s(x_i)\right)^2} \\
&= \mathscr{C}^{\mathcal{S}}_{Q'}
\end{aligned}
$$

D.2 Proof of Theorem 3

The proof that follows is not technically complex; it mainly relies on second order polynomial analysis, but it is quite long. Therefore we propose in Sect. D.3 a graph version of it that is easier to read and provides the main steps and implications.

In this proof, we will use the same notations as in the theorem,

- $\forall i \in [m], F_k(x_i) = \sum_{\substack{s=1 \\ s \neq k}}^{n} \alpha_s h_s(x_i)$ is the fixed-weight majority vote,
- (α, h_k) is the variable weight and its corresponding voter.

So

$$
\mathscr{C}_k(\alpha) = 1 - \frac{1}{m} \frac{\left(\sum_{i=1}^{m} y_i \big(F_k(x_i) + \alpha h_k(x_i)\big)\right)^2}{\sum_{i=1}^{m} \big(F_k(x_i) + \alpha h_k(x_i)\big)^2} = 1 - \frac{A_2\alpha^2 + A_1\alpha + A_0}{B_2\alpha^2 + B_1\alpha + B_0}. \tag{3}
$$

With, $F_k(x_i) \in \mathbb{R}$ and $h_k(x_i) \in \{-1, 1\}$ $\forall x_i, i = 1 \dots m$ and

$$
\begin{aligned}
A_0 &= \gamma(F_k)^2 = \left(\frac{1}{m}\sum_{i=1}^{m} y_i F_k(x_i)\right)^2,\\
A_1 &= 2\gamma(h_k)\gamma(F_k),\\
A_2 &= \gamma(h_k)^2,\\
B_0 &= \nu\big(F_k\big) = \frac{1}{m}\sum_{i=1}^{m} F_k(x_i)^2,\\
B_1 &= 2\tau(F_k, h_k) = 2\frac{1}{m}\sum_{i=1}^{m} h_k(x_i)F_k(x_i),\\
B_2 &= 1.
\end{aligned}
\tag{4}
$$

Deriving the $\mathcal{C}$-bound with respect to α, we obtain

$$
\mathscr{C}'_k(\alpha) = \frac{C_2\alpha^2 + C_1\alpha + C_0}{(B_2\alpha^2 + B_1\alpha + B_0)^2},
\tag{5}
$$

as the third order terms of the numerator are simplified.

With

$$
\begin{aligned}
C_2 &= A_1B_2 - A_2B_1,\\
C_1 &= 2\big(A_0B_2 - A_2B_0\big),\\
C_0 &= A_0B_1 - A_1B_0.
\end{aligned}
\tag{6}
$$

So, with our notations

$$
\begin{aligned}
C_2 &= 2\gamma(h_k)\big(\gamma(F_k) - \gamma(h_k)\tau(F_k, h_k)\big),\\
C_1 &= 2\big(\gamma(F_k)^2 - \gamma(h_k)^2\nu\big(F_k\big)\big),\\
C_0 &= 2\gamma(F_k)\big(\gamma(F_k)\tau(F_k, h_k) - \gamma(h_k)\nu\big(F_k\big)\big).
\end{aligned}
\tag{7}
$$

For the sake of brevity, we define $\mathscr{P}(\alpha) = C_2\alpha^2 + C_1\alpha + C_0$.

Eventually, we will use only h_k that are weak classifiers, so $\gamma(h_k) > 0$. This implies that $\gamma(F_k) > 0$ because F_k is a linear combination, with only positive coefficients of weak classifiers.

D.2.1 Analysis of $B_2\alpha^2 + B_1\alpha + B_0$

Let us recall the definition of the denominator,

$$
B_2\alpha^2 + B_1\alpha + B_0 = \frac{1}{m}\sum_{i=1}^{m}\big(F_k(x_i) + \alpha h_k(x_i)\big)^2
$$

The only way to cancel this sum of squares is if $\forall i \in [m] F_k(x_i) = -\alpha h_k(x_i)$. Yet, we supposed that $\gamma(h_k) > 0$, so if $\forall i \in [m] F_k(x_i) = -\alpha h_k(x_i)$. So, either $h_k = -\frac{F_k}{\alpha}$ with $\alpha > 0$, which is absurd, as we supposed $\gamma(F_k) > 0$ and $\gamma(h_k) > 0$, or $h_k = -\frac{F_k}{\alpha}$ with $\alpha \leq 0$ which is absurd as we supposed $\alpha \geq 0$.

So our hypotheses lead to $B_2\alpha^2 + B_1\alpha + B_0 \neq 0, \forall\alpha \in \mathbb{R}$.

We will now analyse the behaviour of $B_2\alpha^2 + B_1\alpha + B_0$ without any knowledge or hypothesis on $\mathscr{C}_k(\alpha)$. We compute the discriminant

$$\begin{aligned}\Delta &= B_1^2 - 4B_2B_0 \\ &= 4\tau(F_k, h_k)^2 - 4\nu\big(F_k\big) \\ &= 4\big(\tau(F_k, h_k)^2 - \nu\big(F_k\big)\big).\end{aligned} \tag{8}$$

So as we proved earlier that $B_2\alpha^2 + B_1\alpha + B_0$ could not be cancelled on $\mathbb{R}$, we have

$$\tau(F_k, h_k)^2 - \nu\big(F_k\big) < 0. \tag{9}$$

D.2.2 Analysis of $\mathscr{C}_k'(\alpha)$ and $\mathscr{C}_k(\alpha)$

Even if, in CB-Boost, $\alpha \in [0, +\infty[$, we will analyse these function on $\mathbb{R}$.

If we look closer to $\mathscr{C}_k(\alpha)$, we can see that its limits in $\pm\infty$ is $1 - \frac{A_2}{B_2}$. Therefore, $\mathscr{C}_k(\alpha)$ has an asymptotic line in $\pm\infty$.

Moreover, thanks to (5) we know that the sign of $\mathscr{C}_k'(\alpha)$ only depends on the sign of $\mathscr{P}(\alpha)$.

Consequently, we will analyse the sign of $\mathscr{P}(\alpha)$ by exhaustion.

- If $C_2 > 0$ we have $\mathscr{P}(\alpha)$ is a positive parabola as represented in the following Fig. 9. Let's analyse the possibilities concerning its roots.
  - If $\mathscr{P}(\alpha)$ has no real roots, we get a table as represented in Fig. 10.
    This is impossible, because $\mathscr{C}_k(\alpha)$ is supposed to be strictly increasing and continuous.
    So $\mathscr{P}(\alpha)$ not having real roots is ABSURD.
  - If $\mathscr{P}(\alpha)$ has exactly one real root, then

$$\begin{aligned}&\Delta_{\mathscr{P}} = 0 \\ \Leftrightarrow &C_1^2 - 4C_2C_0 = 0 \\ \Leftrightarrow &4\big(\gamma(h_k)^2\nu\big(F_k\big) + \gamma(F_k)\big[\gamma(F_k) - 2\gamma(h_k)\tau(F_k, h_k)\big]\big)^2 = 0 \\ \Leftrightarrow &\gamma(h_k)^2\nu\big(F_k\big) + \gamma(F_k)^2 - 2\gamma(F_k)\gamma(h_k)\tau(F_k, h_k) = 0.\end{aligned} \tag{10}$$

    We can see this equality as a second order polynom roots problem, so it can be analysed through this polynom's discriminant,

$$\begin{aligned}&\Delta_{\mathscr{P}} = 0 \\ \Leftrightarrow &4\gamma(h_k)^2\tau(F_k, h_k)^2 - 4\gamma(h_k)^2\nu\big(F_k\big) \geq 0 \\ \Leftrightarrow &4\gamma(h_k)^2\big(\tau(F_k, h_k)^2 - \nu\big(F_k\big)\big) \geq 0.\end{aligned} \tag{11}$$

    However, we proved in Eq. (9) that it is ABSURD.
    So $\mathscr{P}(\alpha)$ can not have exactly one real root.
  - **Consequently, $\mathscr{P}(\alpha)$ has two real roots**:

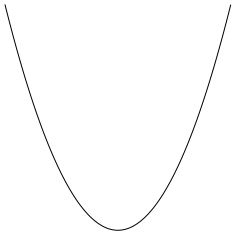


Fig. 9 The type of parabola that illustrates $\mathscr{P}(\alpha)$

α	$-\infty$	$+\infty$
$\mathscr{P}(\alpha)$	$+$	
$\mathscr{C}_k(\alpha)$	$1-\frac{A_2}{B_2}$ ↗	$1-\frac{A_2}{B_2}$

Fig. 10 Variation of $\mathscr{C}_k(\alpha)$ depending on the sign of $\mathscr{P}(\alpha)$ if it has no real roots

$$\alpha^- = \frac{-C_1 - \sqrt{\Delta_{\mathscr{P}}}}{2C_2},$$
$$\alpha^+ = \frac{-C_1 + \sqrt{\Delta_{\mathscr{P}}}}{2C_2}.$$

which leads to the table in Fig. 11. Thanks to the result on the asymptotic line, this implies $\mathscr{C}_k(\alpha)$ looking like Fig. 12 So α^+ is the global argminimum of $\mathscr{C}_k(\alpha)$. In the next section, we will prove that iy is admissible in CB-Boost 's framework.

- If $C_2 < 0$ we use similar methods, to prove that $\mathscr{P}(\alpha)$ has two real roots, obtaining Figs. 13 and 14. So, in this case α^- is the global argminimum.

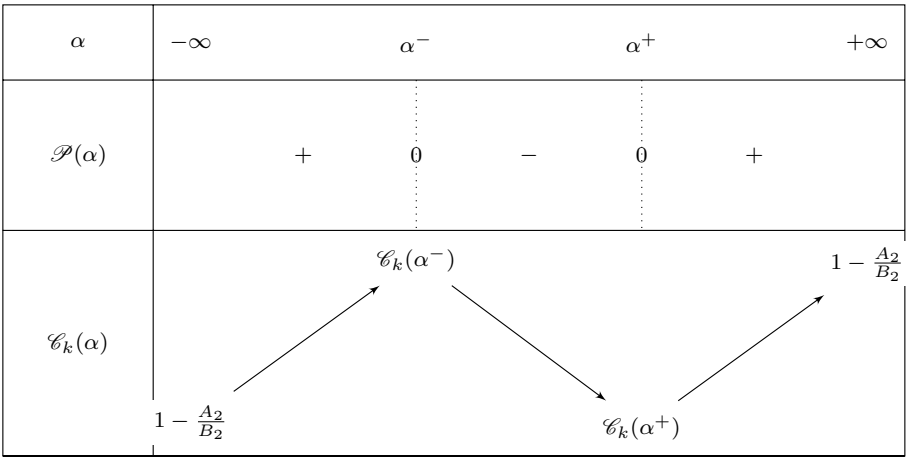


Fig. 11 Variation of $\mathscr{C}_k(\alpha)$ depending on the sign of $\mathscr{P}(\alpha)$ if it has two real roots and $C_2 > 0$

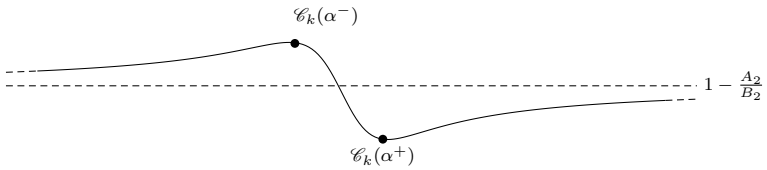


Fig. 12 Shape of $\mathscr{C}_k(\alpha)$, if $\mathscr{P}(\alpha)$ has two real roots and $C_2 > 0$

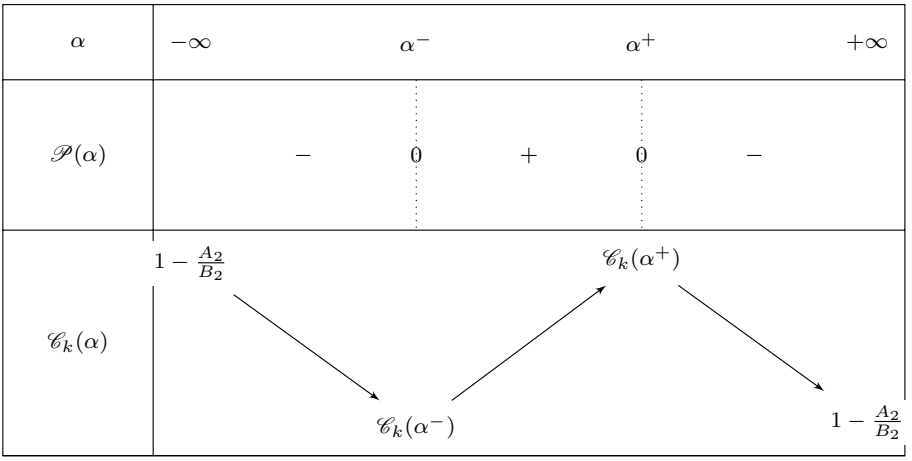


Fig. 13 Variation of $\mathscr{C}_k(\alpha)$ depending on the sign of $\mathscr{P}(\alpha)$ if it has two real roots and $C_2 < 0$

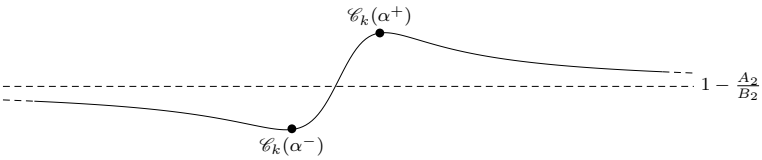


Fig. 14 Shape of $\mathscr{C}_k(\alpha)$, if $\mathscr{P}(\alpha)$ has two real roots and $C_2 < 0$

D.2.3 Possible argminima analysis

Preliminary results Following the previous analysis, we remind

$$\begin{aligned} \Delta_{\mathscr{P}} &= C_1^2 - 4C_0C_2, \\ \alpha^- &= \frac{-C_1 - \sqrt{\Delta_{\mathscr{P}}}}{2C_2}, \\ \alpha^+ &= \frac{-C_1 + \sqrt{\Delta_{\mathscr{P}}}}{2C_2}. \end{aligned} \tag{12}$$

Moreover, we will present two results before the proof by exhaustion.

– First of all, let us analyze $\mathscr{C}_k(0)$ with respect to the asymptotic line $1 - \frac{A_2}{B_2}$

$$\begin{aligned} &\mathscr{C}_k(0) < 1 - \frac{A_2}{B_2} \\ \Leftrightarrow &1 - \frac{A_0}{B_0} < 1 - \frac{A_2}{B_2} \\ \Leftrightarrow &\frac{A_2}{B_2} < \frac{A_0}{B_0} \\ \Leftrightarrow &A_2B_0 < A_0B_2 \,\#\text{ because } B_0 \text{ and } B_2 \text{ are positive} \\ \Leftrightarrow &C_1 < 0. \end{aligned}$$

So

$$\mathscr{C}_k(0) < 1 - \frac{A_2}{B_2} \Leftrightarrow C_1 < 0. \tag{13}$$

Let us note, that with the same method we can prove

$$\mathscr{C}_k(0) \geq 1 - \frac{A_2}{B_2} \Leftrightarrow C_1 \geq 0. \tag{14}$$

– Secondly, we will focus on $\Delta_{\mathscr{P}}$ and its square root.

$$\Delta_{\mathscr{P}} = \left[2\left(\gamma(h_k)^2\nu\left(F_k\right) + \gamma(F_k)\left[\gamma(F_k) - 2\gamma(h_k)\tau(F_k, h_k)\right]\right)\right]^2$$

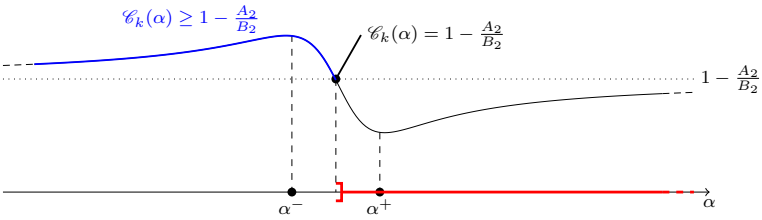


Fig. 15 Analysis of $\mathscr{C}_k(\alpha)$, if $\mathscr{P}(\alpha)$ has two real roots and $C_2 > 0$

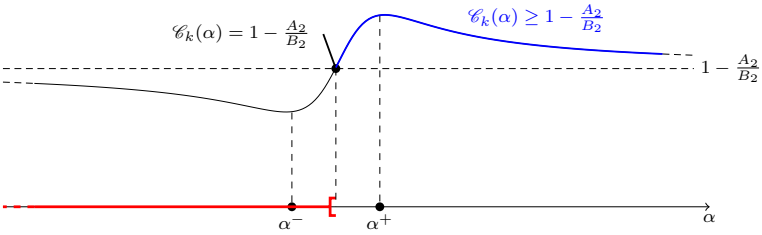


Fig. 16 Analysis of $\mathscr{C}_k(\alpha)$, if $\mathscr{P}(\alpha)$ has two real roots and $C_2 < 0$

In order to deduce $\sqrt{\Delta_{\mathscr{P}}}$, we have to know the sign of

$$\gamma(h_k)^2 \nu\big(F_k\big) + \gamma(F_k)\big[\gamma(F_k) - 2\gamma(h_k)\tau(F_k, h_k)\big].$$

It can be developed as a second order polynomial expression in $\gamma(h_k)$

$$\gamma(h_k)^2 \nu\big(F_k\big) - \gamma(h_k) 2\gamma(F_k)\tau(F, h) + \gamma(F_k)^2.$$

Its discriminant is

$$4\gamma(F_k)^2\tau(F_k, h_k)^2 - 4\nu\big(F_k\big)\gamma(F_k)^2 m = 4\gamma(F_k)^2\big(\tau(F_k, h_k)^2 - \nu\big(F_k\big)m\big),$$

which has the same sign as $\tau(F_k, h_k)^2 - \nu\big(F_k\big)$, which is negative, as seen earlier, in Eq. 9.

So, $\forall \gamma(h_k) \in \mathbb{R}$, $\gamma(h_k)^2 \nu\big(F_k\big) + \gamma(F_k)\big[\gamma(F_k) - 2\gamma(h_k)\tau(F_k, h_k)\big] > 0$ as its discriminant is negative and its evaluation in 0 is positive. So

$$\sqrt{\Delta_{\mathscr{P}}} = 2\big(\gamma(h_k)^2 \nu\big(F_k\big) + \gamma(F_k)\big[\gamma(F_k) - 2\gamma(h_k)\tau(F_k, h_k)\big]\big). \tag{15}$$

Let us now pursue with the proof by exhaustion.

If $C_2 > 0$

- **Let us suppose** $\mathscr{C}_k(0) \geq 1 - \frac{A_2}{B_2}$. So $\mathscr{C}_k(0)$ is over the asymptotic line, so, somewhere on the blue side of the function in Fig. 15. So, necessarily, the red part of the abscissa axis is positive. Consequently $\alpha^+ > 0$.
- **Now, let us suppose that** $\mathscr{C}_k(0) < 1 - \frac{A_2}{B_2}$, which is equivalent to $C_1 < 0$, thanks to Eq. 13 so $\alpha^+ = \frac{-C_1 + \sqrt{\Delta_{\mathscr{P}}}}{2C_2}$, with $C_2 > 0$, $-C_1 > 0$ and $\sqrt{\Delta_{\mathscr{P}}} > 0$ which implies $\alpha^+ > 0$.

As a conclusion for $C_2 > 0$,

$$C_2 > 0 \Rightarrow \alpha^+ > 0. \tag{16}$$

If $C_2 < 0$

– **Symmetrically, if we suppose** $\mathscr{C}_k(0) \geq 1 - \frac{A_2}{B_2}$ then, $\mathscr{C}_k(0)$ is on the blue side of the curve in Fig. 16, so the red side of the abscissa axis is negative. So, $\alpha^- < 0$.

Let us keep in mind that $\mathscr{C}_k(0) \geq 1 - \frac{A_2}{B_2} \Rightarrow \alpha^- < 0$. However, when we use Eq.14, we have $\mathscr{C}_k(0) \geq 1 - \frac{A_2}{B_2} \Leftrightarrow C_1 \geq 0$. So

$$\begin{aligned}
&\mathscr{C}_k(0) \geq 1 - \frac{A_2}{B_2} \\
\Leftrightarrow &C_1 \geq 0 \\
\Leftrightarrow &-C_1 \leq 0 \\
\Rightarrow &-C_1 - \sqrt{\Delta_{\mathscr{P}}} \leq 0 \\
\Rightarrow &\frac{-C_1 - \sqrt{\Delta_{\mathscr{P}}}}{2C_2} \geq 0 \,\#\text{ because } C_2 < 0 \\
\Rightarrow &\alpha^- \geq 0.
\end{aligned}$$

Consequently, if $C_2 < 0$, then $\mathscr{C}_k(0) \geq 1 - \frac{A_2}{B_2} \Rightarrow \alpha^- \geq 0$ and $\mathscr{C}_k(0) \geq 1 - \frac{A_2}{B_2} \Rightarrow \alpha^- < 0$, which is absurd. So $\mathscr{C}_k(0) \geq 1 - \frac{A_2}{B_2}$ is absurd.

– **Let us suppose that** $\mathscr{C}_k(0) < 1 - \frac{A_2}{B_2}$.

We also suppose that $\alpha^- \geq 0$, so

$$\begin{aligned}
&\alpha^- \geq 0 \\
\Rightarrow &\frac{-C_1 - \sqrt{\Delta_{\mathscr{P}}}}{2C_2} \geq 0 \\
\Rightarrow &-C_1 - \sqrt{\Delta_{\mathscr{P}}} \leq 0 \,\#\text{ because we supposed that } C_2 < 0 \\
\Rightarrow &C_1 \geq -\sqrt{\Delta_{\mathscr{P}}} \qquad (17) \\
\Rightarrow &2\big(\gamma(h_k)^2\nu(F_k) + \gamma(F_k)\big[\gamma(F_k)m - 2\gamma(h_k)\tau(F_k,h_k)\big]\big) \geq -2\big(m\gamma(F_k)^2 - \gamma(h_k)^2\nu(F_k)\big) \\
\Rightarrow &2\gamma(F_k)^2 m - 2\gamma(h_k)\tau(F_k,h_k)\gamma(F_k) \geq 0 \\
\Rightarrow &\gamma(F_k)m - \tau(F_k,h_k)\gamma(h_k) \geq 0 \\
\Rightarrow &C_2 \geq 0.
\end{aligned}$$

Which is absurd. So $\alpha^- < 0$. So if $\mathscr{C}_k(0) < 1 - \frac{A_2}{B_2}$ and $\alpha^- < 0$ then the lowest admissible $\mathcal{C}$-bound value in CB-Boost is for zero.

Conclusion So if $C_2 < 0$, the hypothesis does not add value in terms of $\mathcal{C}$-bound so the optimal choice is to weigh it with 0.

On the other hand, if in a direction $C_2 > 0$, α^+ is the global argminimum of $\mathscr{C}_k(\alpha)$.

On Fig. 17, we draw a graph of this proof.

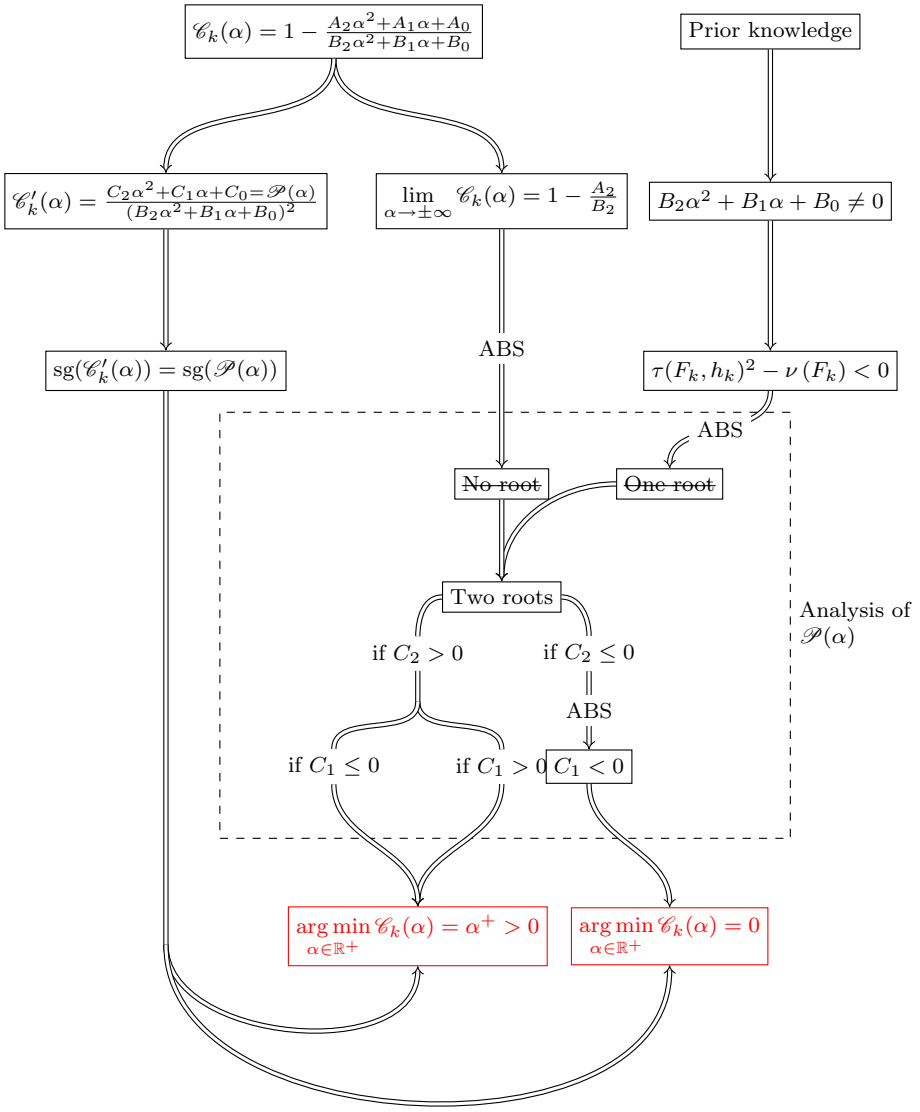


Fig. 17 This representation of the later proof shows the implications that lead to the result. The implications containing the "ABS" notation represent proofs by contradictions

D.3 Graph of the proof

D.4 Proof of Property 1

D.4.1 Proof of the quantification

Let us analyze the $\mathcal{C}$-bound of the vote at the beginning of iteration $t + 1$,

$$\mathscr{C}^F_{t+1} = 1 - \frac{\left(\sum_{i=1}^m [y_i(F_{I_t}(x_i) + \alpha_{I_t} h_{I_t}(x_i)]\right)^2}{\sum_{i=1}^m [y_i(F_{I_t}(x_i) + \alpha h_{I_t}(x_i)]^2} \frac{1}{m}.$$

And at the beginning of iteration t,

$$\mathscr{C}^F_t = 1 - \frac{\left(\sum_{i=1}^m [y_i F_{I_t}(x_i)]\right)^2}{\sum_{i=1}^m F_{I_t}(x_i)^2} \frac{1}{m}.$$

So the numerators and denominators are

$$N_{t+1} = \frac{1}{m^2}\left(\sum_{i=1}^m [y_i(F_{I_t}(x_i) + \alpha_{I_t} h_{I_t}(x_i))]\right)^2,$$
$$D_{t+1} = \frac{1}{m}\sum_{i=1}^m [F_{I_t}(x_i) + \alpha_{I_t} h_{I_t}(x_i)]^2,$$

so for iteration t

$$N_t = \frac{1}{m^2}\left(\sum_{i=1}^m [y_i F_{I_t}(x_i)]\right)^2,$$
$$D_t = \frac{1}{m}\sum_{i=1}^m [F_{I_t}(x_i)]^2.$$

So we obtain $\mathscr{C}^F_{t+1} = 1 - \frac{N_{t+1}}{D_{t+1}} = 1 - \frac{N_t + c}{D_t + c'}$. Yet, to be able to quantify the $\mathcal{C}$-bound 's decrease, we need to focus on $\mathscr{C}^F_{t+1} - \mathscr{C}^F_t = \frac{N_t}{D_t} - \frac{N_t + c}{D_t + c'} = \frac{N_t c' - D_t c}{(D_t + c')D_t}$.

Yet,

$$c = 2\alpha_{I_t}\gamma(F_{I_t})\gamma(h_{I_t}) + \alpha^2_{I_t}\gamma(h_{I_t})^2,$$
$$c' = 2\alpha_{I_t}\tau(F_{I_t}, h_{I_t}) + \alpha^2_{I_t},$$
$$N_t = \gamma(F_{I_t})^2,$$
$$D_t = \nu\left(F_{I_t}\right).$$

So

$$D_t c = \nu\left(F_{I_t}\right)\gamma(h_{I_t})\alpha_{I_t}\left(2\gamma(F_{I_t}) + \alpha_{I_t}\gamma(h_{I_t})\right),$$
$$N_t c' = \gamma(F_{I_t})^2\alpha_{I_t}\left(2\tau(F_{I_t}, h_{I_t}) + \alpha_{I_t}\right).$$

We subtract

$$N_t c' - D_t c = -\alpha_{I_t}\left(\nu\left(F_k\right)\alpha_{I_t}\gamma(h_{I_t})^2\nu\left(F_{I_t}\right) - \alpha_{I_t}\gamma(F_{I_t})^2\right.$$
$$\left.-2\gamma(F_{I_t})^2\tau(F_{I_t}, h_{I_t}) + 2\gamma(F_{I_t})\gamma(h_{I_t})\nu\left(F_{I_t}\right)\right).$$

So, combining with α_{I_t}'s expression found in Theorem 3,

$$N_t c' - D_t c = -\frac{\left(\gamma(h_{I_t})\nu\left(F_{I_t}\right) - \gamma(F_{I_t})\tau(F_{I_t}, h_{I_t})\right)^2\left(\gamma(F_{I_t})^2 - 2\gamma(F_{I_t})\gamma(h_{I_t})\tau(F_{I_t}, h_{I_t}) + \gamma(h_{I_t})^2\nu\left(F_{I_t}\right)\right)}{\left(\gamma(F_{I_t}) - \gamma(h_{I_t})\tau(F_{I_t}, h_{I_t})\right)^2}.$$

Similarly,

$$\begin{aligned}\left(D_t + c'\right)D_t &= \nu\left(F_{I_t}\right)\left(\alpha_{I_t}^2 + 2\alpha_{I_t}\tau(F_{I_t}, h_{I_t}) + \nu\left(F_{I_t}\right)\right)\\ &= \frac{\nu\left(F_{I_t}\right)\left(\nu\left(F_{I_t}\right) - \tau(F_{I_t}, h_{I_t})^2\right)\left(\gamma(F_{I_t})^2 - 2\gamma(F_{I_t})\gamma(h_{I_t})\tau(F_{I_t}, h_{I_t}) + \gamma(h_{I_t})^2\nu\left(F_{I_t}\right)\right)}{\left(\gamma(F_{I_t}) - \gamma(h_{I_t})\tau(F_{I_t}, h_{I_t})\right)^2}.\end{aligned}$$

So,

$$\begin{aligned}\frac{N_t c' - D_t c}{\left(D_t + c'\right)D_t} &= -\frac{\alpha_{I_t}\left(\alpha_{I_t}\gamma(h_{I_t})^2\nu\left(F_{I_t}\right) - \alpha_{I_t}\gamma(F_{I_t})^2 - 2\gamma(F_{I_t})^2\tau(F_{I_t}, h_{I_t}) + 2\gamma(F_{I_t})\gamma(h_{I_t})\nu\left(F_{I_t}\right)\right)}{\nu\left(F_{I_t}\right)\left(\alpha_{I_t}^2 + 2\alpha_{I_t}\tau(F_{I_t}, h_{I_t}) + \nu\left(F_{I_t}\right)\right)}\\ &= -\frac{\left(\gamma(h_{I_t})\nu\left(F_{I_t}\right) - \gamma(F_{I_t})\tau(F_{I_t}, h_{I_t})\right)^2}{\nu\left(F_{I_t}\right)\left(\nu\left(F_{I_t}\right) - \tau(F_{I_t}, h_{I_t})^2\right)}.\end{aligned}$$

Then

$$\begin{aligned}\mathscr{C}_t^F - \mathscr{C}_{t+1}^F &= \left(1 - \frac{N_t}{D_t}\right) - \left(1 - \frac{N_t + c}{D_t + c'}\right)\\ &= \left(\frac{N_t + c}{D_t + c'} - \frac{N_t}{D_t}\right)\\ &= \left(\frac{D_t c - N_t c'}{(D_t + c)D_t}\right)\\ &= \frac{\left(\gamma(h_{I_t})\nu\left(F_{I_t}\right) - \gamma(F_{I_t})\tau(F_{I_t}, h_{I_t})\right)^2}{\nu\left(F_{I_t}\right)\left(\nu\left(F_{I_t}\right) - \tau(F_{I_t}, h_{I_t})^2\right)}.\end{aligned}$$

In conclusion

$$\mathscr{C}_t^F - \mathscr{C}_{t+1}^F = \frac{\left(\gamma(h_{I_t})\nu\left(F_{I_t}\right) - \gamma(F_{I_t})\tau(F_{I_t}, h_{I_t})\right)^2}{\nu\left(F_{I_t}\right)\left(\nu\left(F_{I_t}\right) - \tau(F_{I_t}, h_{I_t})^2\right)}.$$

D.4.2 Proof of positiveness

We have established that

$$\mathscr{C}_t^F - \mathscr{C}_{t+1}^F = \frac{\left(\gamma(h_{I_t})\nu\left(F_{I_t}\right) - \gamma(F_{I_t})\tau(F_{I_t}, h_{I_t})\right)^2}{\nu\left(F_{I_t}\right)\left(\nu\left(F_{I_t}\right) - \tau(F_{I_t}, h_{I_t})^2\right)}.$$

So, $sg(\mathscr{S}_t) = sg(\nu\left(F_{I_t}\right) - \tau(F_{I_t}, h_{I_t})^2)$, yet in Appendix D.1, Eq. 9, we had the property that $\tau(F_{I_t}, h_{I_t})^2 - \nu\left(F_{I_t}\right) < 0$. So $\nu\left(F_{I_t}\right) - \tau(F_{I_t}, h_{I_t})^2 > 0$, so $\mathscr{S}_t > 0$.

E Appendix: datasets

E.1 UCI datasets

We used the following datasets, from the UCI repository (Dua and Graff 2017)[6] to test CB-Boost 's capacity to solve simple problems with few examples and/or a low dimension. We chose Australian (690×14), Balance (625×4), Bupa (345×6), Cylinder (540×35), Hepatitis (155×19), Ionosphere (351×34), Pima (768×8), Yeast (1484×8).

On every dataset, we generated 10 pairs of complementary decision stumps for each attribute to build our hypothesis space. For each experiment, used one half of the dataset to train our algorithms and the other half to test on unseen data.

E.2 MNist

To be able to analyse CB-Boost 's behaviour on larger data, we used M-Nist (LeCun and Cortes 2010)[7] in which we selected four difficult couple of classes : (0,9), (5,3), (5,6) and (7,9). As it has 784 attributes, we used 1 pair of complementary decision stumps for each attribute for the hypothesis space and 4% of the dataset ($\sim$ 471 ex.) to train and the remaining examples ($\sim$ 11000 ex.) to test.

E.3 Animal with attributes

Finally, to have a dataset with a large number of attributes, we used Animals with Attributes (AwA) (Lampert et al.2009)[8] by fusing two descriptors (surf-hist and cq-hist) generating 4688 attributes. As each one on its own is barely relevant, we learned 2000 decision trees of depth 3 as the hypothesis space. They where trained on randomly subsampled attributes (60% of the original dimension, with replacement). We selected two classes: tiger and wolf, that are very confused and provide some challenge to differentiate (we denote this dataset AwA-TvW). And one half of the dataset is used to train (546 ex.) and the other half to test (546ex.)

F Appendix: noise analysis

F.1 Data noising

In order to noise our datasets, we use a normal distribution, centred in 0 and of variable standard deviation. This distribution is then scaled with the attribute's range, added to the data and capped to the limits of the attribute.

[6] Available at https://archive.ics.uci.edu/ml/datasets.php.
[7] Available at http://yann.lecun.com/exdb/mnist/.
[8] Available at https://cvml.ist.ac.at/AwA/.

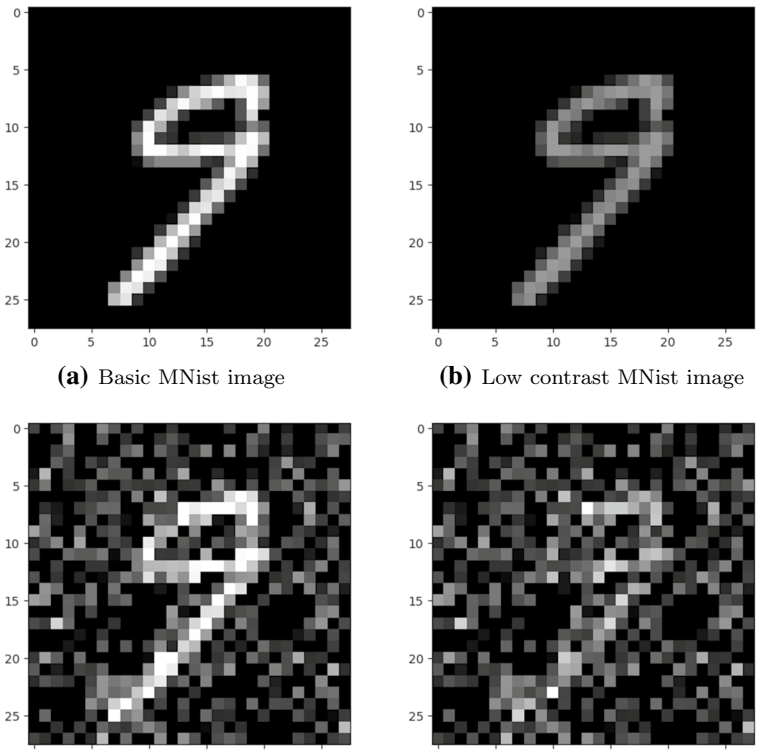
For example, for z a black and white pixel attribute, varying in [0, 255], to generate a noise level of 0.5, a 0-centered normal distribution with standard deviation of 0.5, $\mathcal{N}(0, 0.5)$ was used.

$$\text{Noisy } z = \begin{cases} z + 255 * \mathcal{N}(0, 0.5) & \text{if } z + 255 * \mathcal{N}(0, 0.5) \in [0, 255] \\ 255 & \text{if } z + 255 * \mathcal{N}(0, 0.5) > 255 \\ 0 & \text{if } z + 255 * \mathcal{N}(0, 0.5) < 0 \end{cases}$$

For each attribute (column) of each dataset, we used its upper and lower limit to generate an adequate noise.

F.2 MNist noising

The MNist dataset required a supplementary step in the noising process, at it is mostly comprised of black **or** white pixels. Indeed, this peculiarity made the previously introduced noising process ineffective.Thus, the contrast of each image in the dataset was halved in order for the noise to have a bigger impact on the performances of the studied algorithms. In Fig. 18, the basic, and low contrast images are shown with and without noise, to picture



(a) Basic MNist image **(b)** Low contrast MNist image

(c) Basic noise level 0.25 **(d)** Low contrast noise level 0.25

Fig. 18 Effect of contrast reduction on the noising process

the impact of the contrast decrease. It is clear that reducing the contrast leads to a more difficult classification task once the dataset has been through the noising process.

F.3 Numerical results

See Tables 3 and 4.

Table 3 Numerical results for noise analysis (dataset name : noise std) for the UCI benchmark

	CB-Boost	Adaboost
Australian:0.05	0.148 ± 0.02	0.135 ± 0.016
Australian:0.1	0.154 ± 0.014	0.148 ± 0.015
Australian:0.15	0.148 ± 0.027	0.136 ± 0.02
Australian:0.2	0.162 ± 0.018	0.142 ± 0.011
Australian:0.25	0.177 ± 0.021	0.16 ± 0.015
Australian:0.3	0.199 ± 0.017	0.178 ± 0.016
Australian:0.35	0.187 ± 0.014	0.173 ± 0.011
Australian:0.4	0.239 ± 0.016	0.225 ± 0.013
Australian:0.45	0.245 ± 0.014	0.235 ± 0.019
Australian:0.5	0.245 ± 0.02	0.237 ± 0.018
Bupa:0.05	0.43 ± 0.027	0.416 ± 0.035
Bupa:0.1	0.429 ± 0.054	0.424 ± 0.039
Bupa:0.15	0.46 ± 0.031	0.436 ± 0.026
Bupa:0.2	0.457 ± 0.028	0.477 ± 0.034
Bupa:0.25	0.426 ± 0.03	0.442 ± 0.023
Bupa:0.3	0.455 ± 0.023	0.453 ± 0.032
Bupa:0.35	0.454 ± 0.032	0.445 ± 0.02
Bupa:0.4	0.466 ± 0.038	0.459 ± 0.027
Bupa:0.45	0.48 ± 0.035	0.466 ± 0.023
Bupa:0.5	0.463 ± 0.043	0.452 ± 0.017
Cylinder:0.05	0.303 ± 0.02	0.323 ± 0.021
Cylinder:0.1	0.313 ± 0.024	0.333 ± 0.015
Cylinder:0.15	0.317 ± 0.035	0.341 ± 0.025
Cylinder:0.2	0.363 ± 0.03	0.368 ± 0.03
Cylinder:0.25	0.358 ± 0.018	0.372 ± 0.023
Cylinder:0.3	0.374 ± 0.034	0.4 ± 0.028
Cylinder:0.35	0.405 ± 0.024	0.406 ± 0.03
Cylinder:0.4	0.381 ± 0.021	0.397 ± 0.028
Cylinder:0.45	0.4 ± 0.03	0.407 ± 0.019
Cylinder:0.5	0.409 ± 0.027	0.418 ± 0.029

Table 3 (continued)

	CB-Boost	Adaboost
Hepatitis:0.05	0.385 ± 0.02	0.383 ± 0.039
Hepatitis:0.1	0.38 ± 0.057	0.379 ± 0.055
Hepatitis:0.15	0.365 ± 0.045	0.341 ± 0.036
Hepatitis:0.2	0.395 ± 0.046	0.41 ± 0.04
Hepatitis:0.25	0.378 ± 0.055	0.378 ± 0.065
Hepatitis:0.3	0.398 ± 0.05	0.361 ± 0.044
Hepatitis:0.35	0.404 ± 0.058	0.405 ± 0.074
Hepatitis:0.4	0.416 ± 0.052	0.439 ± 0.034
Hepatitis:0.45	0.431 ± 0.038	0.436 ± 0.059
Hepatitis:0.5	0.443 ± 0.034	0.441 ± 0.05
Ionosphere:0.05	0.097 ± 0.027	0.159 ± 0.006
Ionosphere:0.1	0.099 ± 0.024	0.194 ± 0.028
Ionosphere:0.15	0.135 ± 0.031	0.212 ± 0.025
Ionosphere:0.2	0.188 ± 0.013	0.261 ± 0.031
Ionosphere:0.25	0.198 ± 0.041	0.261 ± 0.021
Ionosphere:0.3	0.212 ± 0.029	0.232 ± 0.019
Ionosphere:0.35	0.256 ± 0.021	0.287 ± 0.027
Ionosphere:0.4	0.27 ± 0.022	0.297 ± 0.034
Ionosphere:0.45	0.292 ± 0.025	0.304 ± 0.024
Ionosphere:0.5	0.318 ± 0.023	0.31 ± 0.031
Pima:0.05	0.255 ± 0.015	0.248 ± 0.011
Pima:0.1	0.29 ± 0.013	0.274 ± 0.012
Pima:0.15	0.319 ± 0.017	0.318 ± 0.013
Pima:0.2	0.326 ± 0.015	0.318 ± 0.019
Pima:0.25	0.346 ± 0.019	0.334 ± 0.017
Pima:0.3	0.32 ± 0.016	0.31 ± 0.012
Pima:0.35	0.373 ± 0.019	0.367 ± 0.012
Pima:0.4	0.374 ± 0.02	0.368 ± 0.017
Pima:0.45	0.371 ± 0.021	0.356 ± 0.012
Pima:0.5	0.374 ± 0.027	0.367 ± 0.013
Yeast:0.05	0.303 ± 0.007	0.306 ± 0.008
Yeast:0.1	0.316 ± 0.009	0.317 ± 0.01
Yeast:0.15	0.324 ± 0.013	0.318 ± 0.01
Yeast:0.2	0.317 ± 0.01	0.319 ± 0.01
Yeast:0.25	0.321 ± 0.008	0.315 ± 0.005
Yeast:0.3	0.313 ± 0.003	0.316 ± 0.002
Yeast:0.35	0.315 ± 0.006	0.318 ± 0.006
Yeast:0.4	0.319 ± 0.01	0.325 ± 0.008
Yeast:0.45	0.314 ± 0.003	0.321 ± 0.007
Yeast:0.5	0.315 ± 0.006	0.318 ± 0.006

Table 4 Numerical results for noise analysis (dataset name : noise std) for the bigger datasets

	CB-Boost	Adaboost
AwA-TvW:0.05	0.085 ± 0.007	0.089 ± 0.012
AwA-TvW:0.1	0.084 ± 0.006	0.084 ± 0.008
AwA-TvW:0.15	0.086 ± 0.006	0.087 ± 0.011
AwA-TvW:0.2	0.089 ± 0.009	0.093 ± 0.008
AwA-TvW:0.25	0.097 ± 0.01	0.099 ± 0.009
AwA-TvW:0.3	0.093 ± 0.007	0.101 ± 0.011
AwA-TvW:0.35	0.097 ± 0.008	0.107 ± 0.01
AwA-TvW:0.4	0.105 ± 0.008	0.104 ± 0.008
AwA-TvW:0.45	0.101 ± 0.009	0.113 ± 0.015
AwA-TvW:0.5	0.106 ± 0.008	0.115 ± 0.008
MNist 0v9:0.05	0.014 ± 0.001	0.013 ± 0.002
MNist 0v9:0.1	0.018 ± 0.002	0.016 ± 0.001
MNist 0v9:0.15	0.025 ± 0.002	0.022 ± 0.002
MNist 0v9:0.2	0.035 ± 0.006	0.028 ± 0.002
MNist 0v9:0.25	0.04 ± 0.004	0.033 ± 0.002
MNist 0v9:0.3	0.046 ± 0.003	0.041 ± 0.002
MNist 0v9:0.35	0.059 ± 0.004	0.055 ± 0.003
MNist 0v9:0.4	0.072 ± 0.006	0.067 ± 0.004
MNist 0v9:0.45	0.094 ± 0.005	0.083 ± 0.006
MNist 0v9:0.5	0.111 ± 0.006	0.098 ± 0.005
MNist 5v3:0.05	0.082 ± 0.005	0.075 ± 0.003
MNist 5v3:0.1	0.101 ± 0.005	0.089 ± 0.005
MNist 5v3:0.15	0.133 ± 0.008	0.118 ± 0.007
MNist 5v3:0.2	0.159 ± 0.006	0.145 ± 0.007
MNist 5v3:0.25	0.174 ± 0.006	0.166 ± 0.006
MNist 5v3:0.3	0.187 ± 0.006	0.188 ± 0.009
MNist 5v3:0.35	0.21 ± 0.007	0.21 ± 0.006
MNist 5v3:0.4	0.234 ± 0.007	0.236 ± 0.01
MNist 5v3:0.45	0.261 ± 0.008	0.255 ± 0.005
MNist 5v3:0.5	0.285 ± 0.008	0.281 ± 0.009
MNist 5v6:0.05	0.043 ± 0.002	0.041 ± 0.003
MNist 5v6:0.1	0.053 ± 0.004	0.048 ± 0.002
MNist 5v6:0.15	0.068 ± 0.006	0.061 ± 0.006
MNist 5v6:0.2	0.089 ± 0.008	0.077 ± 0.004
MNist 5v6:0.25	0.099 ± 0.008	0.089 ± 0.006
MNist 5v6:0.3	0.111 ± 0.006	0.106 ± 0.006
MNist 5v6:0.35	0.133 ± 0.004	0.124 ± 0.008
MNist 5v6:0.4	0.154 ± 0.007	0.143 ± 0.006
MNist 5v6:0.45	0.176 ± 0.006	0.167 ± 0.009
MNist 5v6:0.5	0.2 ± 0.007	0.19 ± 0.009
MNist 7v9:0.05	0.079 ± 0.004	0.074 ± 0.005
MNist 7v9:0.1	0.1 ± 0.009	0.09 ± 0.006
MNist 7v9:0.15	0.128 ± 0.004	0.109 ± 0.006
MNist 7v9:0.2	0.156 ± 0.006	0.135 ± 0.007
MNist 7v9:0.25	0.175 ± 0.005	0.151 ± 0.005
MNist 7v9:0.3	0.196 ± 0.005	0.172 ± 0.006

Table 4 (continued)

	CB-Boost	Adaboost
MNist 7v9:0.35	0.221 ± 0.012	0.201 ± 0.008
MNist 7v9:0.4	0.246 ± 0.009	0.22 ± 0.007
MNist 7v9:0.45	0.265 ± 0.007	0.246 ± 0.006
MNist 7v9:0.5	0.287 ± 0.006	0.267 ± 0.006

References

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Catoni, O. (2007). PAC-Bayesian supervised classification: the thermodynamics of statistical learning. arXiv preprint arXiv:0712.0248.

Cortes, C., Mohri, M., & Syed, U. (2014). Deep boosting. In: *Proceedings of the thirty-first international conference on machine learning* (ICML) (2014).

Demiriz, A., Bennett, K. P., & Shawe-Taylor, J. (2002). Linear programming boosting via column generation. *Machine Learning*, *46*(1), 225–254.

Dua, D., & Graff, C. (2017). UCI machine learning repository.

Dziugaite, G.K., & Roy, D.M. (2018). Data-dependent PAC-Bayes priors via differential privacy. In: *Advances in neural information processing systems*, pp. 8440–8450.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*(5), 1189–1232.

Germain, P., Lacasse, A., Laviolette, F., & Marchand, M. (2009). PAC-Bayesian learning of linear classifiers. In: *Proceedings of the 26th ICML*, pp. 353–360. ACM.

Germain, P., Lacasse, A., Laviolette, F., Marchand, M., & Roy, J. F. (2015). Risk bounds for the majority vote: From a PAC-Bayesian analysis to a learning algorithm. *Journal of Machine Learning Research*, *16*(1), 787–860.

Lacasse, A., Laviolette, F., Marchand, M., Germain, P., & Usunier, N. (2006). PAC-Bayes bounds for the risk of the majority vote and the variance of the Gibbs classifier. In: B. Schölkopf, J.C. Platt, T. Hoffman (Eds.) *Advances in neural information processing systems* 19, pp. 769–776. MIT Press.

Lampert, C.H., Nickisch, H., & Harmeling, S. (2009). Learning to detect unseen object classes by between-class attribute transfer. In: *2009 IEEE Conference on computer vision and pattern recognition*, pp. 951–958. IEEE.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E., & Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, *5*, 27–72.

Langford, J., & Shawe-Taylor, J. (2003). PAC-Bayes & margins. In: *Advances in neural information processing systems*, pp. 439–446.

LeCun, Y., & Cortes, C. (2010). MNIST handwritten digit database.

Marchand, M., & Taylor, J. S. (2003). The set covering machine. *Journal of Machine Learning Research*, *3*, 723–746.

McAllester, D. A. (1999). Some PAC-Bayesian theorems. *Machine Learning*, *37*(3), 355–363.

McAllester, D. A. (2003). PAC-Bayesian stochastic model selection. *Machine Learning*, *51*(1), 5–21.

Parrado-Hernández, E., Ambroladze, A., Shawe-Taylor, J., & Sun, S. (2012). PAC-Bayes bounds with data dependent priors. *Journal of Machine Learning Research*, *13*, 3507–3531.

Roy, J.F., Marchand, M., & Laviolette, F. (2016) A column generation bound minimization approach with PAC-Bayesian generalization guarantees. In: A. Gretton, C.C. Robert (eds.) *Proceedings of the 19th international conference on artificial intelligence and statistics, proceedings of machine learning research*, vol. 51, pp. 1241–1249. PMLR, Cadiz, Spain. http://proceedings.mlr.press/v51/roy16.html.

Schapire, R. E., & Freund, Y. (2012). *Boosting: foundations and algorithms*. Cambridge: The MIT Press.

Seeger, M. (2002). PAC-Bayesian generalisation error bounds for gaussian process classification. *Journal of Machine Learning Research*, *3*, 233–269.

Seldin, Y., Cesa-Bianchi, N., Auer, P., Laviolette, F., & Shawe-Taylor, J. (2012). PAC-Bayes-bernstein inequality for martingales and its application to multiarmed bandits. *Proceedings of the Workshop on Online Trading of Exploration and Exploitation*, *2*, 98–111.

Sonnenburg, S., Rätsch, G., Schäfer, C., & Schölkopf, B. (2006). Large scale multiple kernel learning. *Journal of Machine Learning Research*, *7*, 1531–1565.
Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, *27*(11), 1134–1142.
Zhu, J., Rosset, S., Zou, H., & Hastie, T. (2006). Multi-class adaboost. *Statistics and its Interface*,. https://doi.org/10.4310/SII.2009.v2.n3.a8.