A Column Generation Bound Minimization Approach with PAC-Bayesian Generalization Guarantees

Jean-Francis Roy jean-francis.roy@ift.ulaval.ca
Mario Marchand mario.marchand@ift.ulaval.ca
François Laviolette francois.laviolette@ift.ulaval.ca
Département d'informatique et de génie logiciel, Université Laval, Québec, Canada

Abstract

The $\mathcal{C}$-bound, introduced in Lacasse et al. [2006], gives a tight upper bound on the risk of the majority vote classifier. Laviolette et al. [2011] designed a learning algorithm named MinCq that outputs a dense distribution on a finite set of base classifiers by minimizing the $\mathcal{C}$-bound, together with a PAC-Bayesian generalization guarantee. In this work, we design a column generation algorithm that we call CqBoost, that optimizes the $\mathcal{C}$-bound and outputs a sparse distribution on a possibly infinite set of voters. We also propose a PAC-Bayesian bound for CqBoost that holds for finite and two cases of continuous sets of base classifiers. Finally, we compare the accuracy and the sparsity of CqBoost with MinCq and other state-of-the-art boosting algorithms.

1 INTRODUCTION

Many state-of-the-art binary classification learning algorithms, such as Bagging [Breiman, 1996], Boosting [Schapire and Singer, 1999], and Random Forests [Breiman, 2001], output prediction functions that can be seen as a majority vote of "simple" classifiers. Majority votes are also central in the Bayesian approach (see Gelman et al. for an introductory text). Moreover, classifiers produced by kernel methods, such as the Support Vector Machine (SVM) [Cortes and Vapnik, 1995], can also be viewed as majority votes. Indeed, to classify an example x, the SVM classifier computes $\operatorname{sgn}\left(\sum_{i=1}^{|S|} \alpha_i\, y_i\, k(x_i, x)\right)$. Hence, as for standard binary majority votes, if the total weight of each $\alpha_i\, y_i\, k(x_i, x)$ that votes positive is larger than the total weight for the negative choice, the classifier will output a $+1$ label (and a -1 label in the opposite case).

Appearing in Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51. Copyright 2016 by the authors.

Most bounds on majority votes take into account the margin of the majority vote on an example (x, y), that is the difference between the total vote weight that has been given to the winning class minus the weight given to the alternative class. As an example, PAC-Bayesian bounds on majority vote classifiers are related to a stochastic classifier, called the *Gibbs* classifier, which is, up to a linear transformation, equivalent to the first statistical moment of the margin when each (x, y) is drawn independently from the same distribution [Laviolette et al., 2011]. Unfortunately, in many ensemble methods, the voters are weak and no majority vote can achieve a large margin. Lacasse et al. [2006] proposed a tighter relation between the risk of the majority vote that takes into account both the first and the second moments of the margin: the $\mathcal{C}$-bound. This bound corroborates classical knowledge on the behavior of majority votes: it is not only how good are the voters but also how they are correlated in their voting. With this idea in mind, many boosting algorithms such as MDBoost [Shen and Li, 2010], minimize an objective function that controls the mean and variance of the margin distribution. However, these approaches are not based on the direct minimization of a risk bound.

The $\mathcal{C}$-bound of Lacasse et al. [2006] has inspired a learning algorithm that *directly* minimize a PAC-Bayesian generalization bound, named MinCq [Laviolette et al., 2011], that has also been extended to consider *a priori* constraints [Bellet et al., 2014], to *late classifier fusion* [Morvant et al., 2014] and to *domain adaptation* [Morvant, 2015]. This algorithm turns out to be a quadratic program (QP) that produces *dense* weight distributions over a *finite* set of voters. In many applications of machine learning, sparse solutions are preferred as the learned classification rules are easier to interpret, and the resulting classifier is computationally faster. In this work, we design CqBoost, a boosting learning algorithm based on a relaxed formulation of MinCq, but that produces sparse solutions and can consider infinite sets of voters. It makes use of Lagrange duality and column generation techniques, and is based on a PAC-Bayesian generalization bound that holds for finite and two cases of continuous sets of classifiers. Despite the fact that CqBoost is designed to minimize a different generalization bound than other column generation techniques, the resulting algorithm turns out to share similarities with other algorithms, such as LPBoost [Demiriz et al., 2002], CG-Boost [Bi et al., 2004] and MDBoost [Shen and Li, 2010]. Empirical experiments show that CqBoost is state of the art when compared with these algorithms, with AdaBoost [Schapire and Singer, 1999] and with MinCq, while outputting solutions that are considerably sparser than MinCq.

This paper is organized as follows. Section 2 reviews the $\mathcal{C}$-bound and the MinCq algorithm, and presents generalized definitions of the margin that consider two cases of continuous sets of voters. Section 3 presents a generalization guarantee for CqBoost by providing a specialized PAC-Bayesian bound. Section 4 explains how to construct a meaningful Lagrange dual of MinCq. Section 5 presents the resulting column generation algorithm. Section 6 discusses the relations with previous work. Section 7 shows empirical results on benchmark data, and we conclude in Section 8.

2 THE $\mathcal{C}$-BOUND AND MINCQ

Consider a supervised learning classification task, where $\mathcal{X}$ is the input space and $\mathcal{Y} = \{-1, +1\}$ is the output space. The learning sample $S = \{(x_i, y_i)\}_{i=1}^m$ consists of m examples drawn *i.i.d.* from a fixed but unknown distribution D over $\mathcal{X}\times\mathcal{Y}$. Let $\mathcal{H}$ be a (possibly continuous) set of real-valued voters $h : \mathcal{X} \to [-1, 1]$. For any *posterior* distribution Q on $\mathcal{H}$, the *majority vote classifier* B_Q is defined by

$$B_Q(x) \triangleq \operatorname{sgn}\left[\mathop{\mathbf{E}}_{h\sim Q} h(x)\right],$$

where $\operatorname{sgn}(a)$ returns 1 if $a > 0$ and -1 otherwise.

Given a data-generating distribution D and a sample S, the objective of the PAC-Bayesian approach is to find the posterior distribution Q on $\mathcal{H}$ that minimizes the *true* risk of the Q-weighted majority vote $B_Q(\cdot)$, given by

$$R_D(B_Q) \triangleq \mathop{\mathbf{E}}_{(x,y)\sim D} I\left(B_Q(x) \neq y\right),$$

where $I(a) = 1$ if predicate a is true and 0 otherwise.

It is well know that minimizing $R_D(B_Q)$ is NP-hard. To get around this problem, one solution is to consider the $\mathcal{C}$-bound as a surrogate, that is a tight upper bound on $R_D(B_Q)$. This quantity depends on the first two statistical moments of the *margin* of the majority vote B_Q, defined below.

Definition 1. *Given a set $\mathcal{H}$ of voters $h : \mathcal{X} \to [-1, 1]$ and a distribution Q on $\mathcal{H}$, the margin $M_Q(x, y)$ on example (x, y) is defined by*

$$M_Q(x, y) \triangleq y \mathop{\mathbf{E}}_{h\sim Q} h(x).$$

Given a distribution D' on $\mathcal{X}\times\mathcal{Y}$, the first and second statistical moments of M_Q are given, respectively, by

$$\mu_1(M_Q^{D'}) \triangleq \mathop{\mathbf{E}}_{(x,y)\sim D'} M_Q(x, y), \quad \textit{and}$$

$$\mu_2(M_Q^{D'}) \triangleq \mathop{\mathbf{E}}_{(x,y)\sim D'} \Big(M_Q(x, y)\Big)^2.$$

According to the definition of the margin, $B_Q(\cdot)$ correctly classifies an example (x, y) when its margin is strictly positive, hence $R_{D'}(B_Q) = \mathbf{Pr}_{(x,y)\sim D'}\left(M_Q(x, y) \leq 0\right)$. This equality allows us to prove the following theorem.

Theorem 1 (The $\mathcal{C}$-bound of Laviolette et al. [2011])**.** *For any distribution Q on a set of real-valued functions $\mathcal{H}$, and for any distribution D' on $\mathcal{X}\times\mathcal{Y}$, if $\mu_1(M_Q^D) > 0$, then we have:*

$$R_{D'}(B_Q) \leq 1 - \frac{\left(\mu_1(M_Q^{D'})\right)^2}{\mu_2(M_Q^{D'})}.$$

Proof. The Cantelli-Chebyshev inequality states that for any random variable Z and any $a > 0$, we have that $\mathbf{Pr}\left(Z \leq \mathbf{E}[Z] - a\right) \leq \frac{\mathbf{Var}\, Z}{\mathbf{Var}\, Z + a^2}$. We obtain the result by applying this inequality with $Z = M_Q(x, y)$ and $a = \mu_1(M_Q^{D'})$. $\square$

The minimization of the empirical counterpart of the $\mathcal{C}$-bound is a natural solution for learning a distribution Q that leads to a Q-weighted majority vote $B_Q(\cdot)$ with a low generalization error. This strategy is justified by a PAC-Bayesian generalization bound over the $\mathcal{C}$-bound, and has given the MinCq algorithm [Laviolette et al., 2011]. Given a training set S of m examples, MinCq minimizes the second moment of the margin $\mu_2(M_Q^S)$ with an equality constraint on the first moment of the margin $\mu_1(M_Q^S) = \mu$, where μ is a hyperparameter of the algorithm that gives an explicit control on the first moment of the margin. MinCq considers *all* voters from $\mathcal{H}$ in its optimization problem. As the approach presented in this paper will allow us to consider larger (possibly *infinite*) sets of voters, let us present three possible choices of sets $\mathcal{H}$.

When $\mathcal{H}$ denotes the simpler case of a finite set of voters, q_i denotes the probability mass of distribution

Q on voter $h_i \in \mathcal{H}$. We also consider two cases of continuous sets $\mathcal{H}$. One case consists of the set of linear functions $x \mapsto \langle \mathbf{w}, \boldsymbol{\phi}(x)\rangle \in \mathbb{R}$ where $\boldsymbol{\phi}$ denotes some high-dimensional feature map, $\mathbf{w}$ is an arbitrary weight vector in the space of feature vectors, and $\langle , \rangle$ denotes the inner product in this space. The other case is when $\mathcal{H}$ denotes the set of linear classifiers $x \mapsto \mathrm{sgn}(\langle \mathbf{w}, \boldsymbol{\phi}(x)\rangle) \in \{-1, +1\}$. In both of these cases the feature map $\boldsymbol{\phi}$ is fixed and $\mathcal{H}$ denotes the set of weight vectors in the feature space. In these cases, the learner uses a distribution Q which consists of a mixture of n isotropic Gaussians[1]. Hence, for all $\mathbf{v} \in \mathcal{H}$,

$$Q(\mathbf{v}) \;=\; \sum_{i=1}^{n} q_i \left(\frac{1}{\sqrt{2\pi}}\right)^{N} \exp\left(-\frac{1}{2}\|\mathbf{v} - \mathbf{w}_i\|^2\right), \quad (1)$$

where q_i denotes the weight assigned to the Gaussian centered on $\mathbf{w}_i$, N denotes the dimension[2] of $\mathbf{v}$, and $\|\cdot\|$ denotes the Euclidean norm. In the proposed column generation approach, each weight vector $\mathbf{w}_i$ can be chosen in $\mathcal{H}$ such as to optimize a criterion (to be defined). Once $\mathbf{w}_i$ is chosen, $q_i, q_{i-1}, \ldots, q_1$ will be found by solving a quadratic program.

The margin $M_Q(x, y)$ of the Gaussian mixture Q on the set of linear functions is given by

$$\begin{aligned} M_Q(x,y) \;&=\; \int_{\mathcal{H}} Q(\mathbf{v})\, y \langle \mathbf{v}, \boldsymbol{\phi}(x)\rangle \, d\mathbf{v} \\ &=\; \sum_{i=1}^{p} q_i \, y \, \langle \mathbf{w}_i, \boldsymbol{\phi}(x)\rangle \,, \end{aligned}$$

whereas, for the case of linear classifiers, we find

$$\begin{aligned} M_Q(x,y) \;&=\; \int_{\mathcal{H}} Q(\mathbf{v})\, \mathrm{sgn}(y\langle \mathbf{v}, \boldsymbol{\phi}(x)\rangle) \, d\mathbf{v} \\ &=\; \sum_{i=1}^{p} q_i \, F(y\langle \mathbf{w}_i, \boldsymbol{\phi}(x)\rangle) \,, \end{aligned}$$

where $F : \mathbb{R} \to [-1, +1]$ is a monotonously increasing "sigmoidal" type function which is very similar to the tanh() function, and which is obtained from a Gaussian cumulative:

$$F(y\langle \mathbf{w}_i, \boldsymbol{\phi}(x)\rangle) \;\triangleq\; -1 + \sqrt{\frac{2}{\pi}} \int_{-\infty}^{y\langle \mathbf{w}_i, \boldsymbol{\phi}(x)\rangle} e^{-\frac{1}{2}z^2} \; dz \,.$$

To write down the quadratic program that needs to be solved at each step t (once $\mathbf{w}_t$ has been chosen for the cases when $\mathcal{H}$ is continuous), let $\mathbf{H}$ be the *classification matrix* of m rows and t columns, where m is the size of the training set, t is the amount of voters in the set $\mathcal{H}$ considered at step t, and where $H_{ki} = h_i(x_k)$. Each column represents a voter of $\mathcal{H}$, and each line represents an example. Also, let $\mathbf{y}$ be the vector containing the m labels and $\mathbf{q} \triangleq (q_1, \ldots, q_t)$ where q_i is the probability weight on voter h_i defined above in all three cases for $\mathcal{H}$. The empirical first and second moments of the margin can be written as

$$\begin{aligned} \mu_1(M_Q^S) \;&=\; \frac{1}{m}\sum_{k=1}^{m} M_Q(x_k, y_k) \;&=\; \frac{1}{m}\mathbf{y}^\top \mathbf{H}\,\mathbf{q}\,, \\ \mu_2(M_Q^S) \;&=\; \frac{1}{m}\sum_{k=1}^{m} \big(M_Q(x_k, y_k)\big)^2 \;&=\; \frac{1}{m}\mathbf{q}^\top \mathbf{H}^\top \mathbf{H}\,\mathbf{q}\,, \end{aligned}$$

where the last equality comes from the fact that $y_k^2 = 1$ for all $k \in \{1, \ldots, m\}$.

Together with additional restrictions described below, MinCq solves the following quadratic program.

$$\begin{gathered} \text{Solve:} \;\; \operatorname*{argmin}_{\mathbf{q}} \;\; \frac{1}{m}\mathbf{q}^\top \mathbf{H}^\top \mathbf{H}\,\mathbf{q} \\ \text{subject to:} \qquad\qquad\qquad\qquad (2) \\ \frac{1}{m}\mathbf{y}^\top \mathbf{H}\,\mathbf{q} \;\le\; \mu\,, \quad \mathbf{q} \;\succeq\; \mathbf{0}\,, \quad \mathbf{1}^\top \mathbf{q} \;=\; 1\,, \end{gathered}$$

where $\mathbf{H}$ is the $m \times n$ classification matrix that considers all n voters from a finite set $\mathcal{H}$, $\mathbf{0}$ is a vector of n zeros, and $\mathbf{1}$ is a vector of n ones. To solve this problem, one must first compute $\mathbf{H}^\top\mathbf{H}$ in $\mathcal{O}(m \times n^2)$, and then solve the quadratic program in $\mathcal{O}(n^3)$. Consequently, MinCq is not suitable when the number n of considered voters is large.

The version of MinCq proposed in Laviolette et al. [2011] (and Germain et al. [2015]) is slightly more restrictive by requiring the set $\mathcal{H}$ to be *self-complemented* (meaning that for each voter $h \in \mathcal{H}$ we also have its complement $-h \in \mathcal{H}$) together with another restriction on Q named the *quasi-uniformity*, which introduces a L_∞ regularization on Q. These restrictions on Q enable the authors to replace the inequality constraint on $\mu_1(M_Q^S)$ with an equality constraint, and allow to obtain a tight PAC-Bayesian bound that does not rely on the Kullback-Leibler (KL) divergence, defined in Section 3.

In this paper, we remove these restrictions and relax the primal formulation of the problem to obtain an interesting dual problem in Section 4. Removing these constraints also has the effect of allowing the resulting column generation algorithm of Section 5 to output sparser solutions.[3] The next section shows that we

[1]The choice of Gaussians for infinite sets of voters is common in PAC-Bayesian literature, as it allows to obtain exact analytical expressions for the majority vote, the margin and KL($Q\|P$). See Langford and Shawe-Taylor [2002] and Germain et al. [2009] for related literature.

[2]For infinite dimensional feature spaces, we can replace each Gaussian distribution by a Gaussian process.

[3]Note that we chose to use column generation in this paper, but this relaxed formulation (and associated generalization bound results) could be used to derive other types of algorithms as well.

can still obtain PAC-Bayesian guarantees at the price of introducing a KL-divergence term which is upper-bounded by a small value.

3 GENERALIZATION GUARANTEES

PAC-Bayesian theorems [McAllester, 1999] bound the *true* risk of the majority vote classifier from an empirical estimate computed from a sample S of m examples (drawn *i.i.d.* from an unknown distribution D), a *prior* distribution P on $\mathcal{H}$ (chosen before seeing S), and a *posterior* distribution Q (chosen after seeing S). These bounds generally rely on the Kullback-Leibler divergence between Q and P, defined below.

Definition 2. *The Kullback-Leibler divergence between distributions Q and P is defined by*[4]

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

Typical PAC-Bayesian theorems [McAllester, 2003, Seeger, 2003, Catoni, 2007, Germain et al., 2009] indirectly bound the risk of the majority vote classifier through a bound on the so-called *Gibbs risk*, which has a linear relationship with $\mu_1(M_Q^D)$. The bound on the majority vote risk is then given by twice the bound on the Gibbs risk. However, by taking into account the second moment of the margin, the $\mathcal{C}$-bound of Theorem 1 provides a tighter relation for the risk of the majority vote classifier. Lacasse et al. [2006], Laviolette et al. [2011] and Germain et al. [2015] have developed PAC-Bayesian bounds that relate the real value of the $\mathcal{C}$-bound and its empirical estimate. The MinCq algorithm directly minimizes such a generalization bound under the restrictions that the set of voters is finite and that the weight distribution is quasi-uniform. In the following, we present a PAC-Bayesian bound on the risk of the majority vote that depends on the first and second moments of the margin, and applies to the cases when $\mathcal{H}$ is finite and to the two cases described above where the set of $\mathcal{H}$ voters is continuous and the posterior distribution Q is given by Equation (1).

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

$$R_D(B_Q) \ \le\ 1-\frac{\max\Big(0,\ \big(\underline{\mu_1}\big)^2\Big)}{\min\big(1,\ \overline{\mu_2}\big)}\,,$$

where

$$\underline{\mu_1} \ \triangleq\ \mu_1(M_Q^S)-\sqrt{\tfrac{2}{m}\Big[\mathrm{KL}(Q\|P)+\ln\tfrac{2\sqrt{m}}{\delta/2}\Big]}\,,$$

$$\overline{\mu_2} \ \triangleq\ \mu_2(M_Q^S)+\sqrt{\tfrac{2}{m}\Big[2\mathrm{KL}(Q\|P)+\ln\tfrac{2\sqrt{m}}{\delta/2}\Big]}\,.$$

Proof. The result is an application of the definition of the $\mathcal{C}$-bound of Theorem 1, by lower-bounding $\mu_1(M_Q^D)$ and upper-bounding $\mu_2(M_Q^D)$. These bounds and their respective proof are provided in Appendix A, as Theorems 3 and 4. The proof of Theorem 3 uses common techniques of the PAC-Bayesian literature. The proof of Theorem 4 is a bit trickier, and make use of ideas discussed in Germain et al. [2015]. □

Hence, the upper bound on the risk of the majority vote depends on a lower bound of the first moment of the margin and on an upper bound of the second moment. Both of these bounds depend on $\mathrm{KL}(Q\|P)$ which, therefore, should be much smaller than m in order to obtain a good guarantee on the risk of the majority vote. In the case where $\mathcal{H}$ just consists of a set of n voters and the prior distribution P is uniform, we have the following upper-bound for the KL divergence:

$$\begin{aligned}\mathrm{KL}(Q\|P) \ &=\ \sum_{i=1}^n Q(h_i)\ln\frac{Q(h_i)}{1/n}\\ &\le\ \sum_{i=1}^n Q(h_i)\ln\frac{1}{1/n} \ =\ \ln n\,.\end{aligned}$$

Since we normally have $n\ll m$, finding the distribution Q over $\mathcal{H}$ that minimizes the upper bound on $R_D(B_Q)$ given by Theorem 2 amounts at solving the optimization problem of Equation (2).

In the cases where $\mathcal{H}$ is continuous and when Q is given by Equation (1), let us denote the isotropic unit-variance Gaussian centered on $\mathbf{w}_i$ by $G_{\mathbf{w}_i}$. For the prior P, let us use $G_{\mathbf{0}}$. We then have

$$\begin{aligned}\mathrm{KL}(Q\|P) \ &=\ \int_{\mathcal{H}}\sum_{i=1}^n q_i G_{\mathbf{w}_i}(\mathbf{v})\ln\frac{\sum_{i=1}^n q_i G_{\mathbf{w}_i}(\mathbf{v})}{G_{\mathbf{0}}(\mathbf{v})}d\mathbf{v}\\ &\le\ \int_{\mathcal{H}}\sum_{i=1}^n q_i G_{\mathbf{w}_i}(\mathbf{v})\ln\frac{G_{\mathbf{w}_i}(\mathbf{v})}{G_{\mathbf{0}}(\mathbf{v})}d\mathbf{v}\\ &=\ \frac{1}{2}\sum_{i=1}^n q_i\|\mathbf{w}_i\|^2\,,\end{aligned}$$

where the inequality comes from Jensen's inequality applied the the convex function $x\ln x$, and where the last equality is a standard result for Gaussian distributions. Since the q_is sums to one, we will have $\mathrm{KL}(Q\|P)\ll m$ whenever the Euclidean norm of each

[4]When P and Q are distributions on a discrete set $\mathcal{H}$, $P(h)$ and $Q(h)$ denote probability masses at h. When $\mathcal{H}$ is a continuous set, $P(h)$ and $Q(h)$ denote probability densities at h.

voter $\mathbf{w}_i$ is upper-bounded by a constant much smaller than m. This is the case, for example, for the set of linear functions $x \mapsto \langle \mathbf{w}, \boldsymbol{\phi}(x)\rangle$ where $\mathbf{w} = \boldsymbol{\phi}(x')$ for some arbitrary point $x' \in \mathcal{X}$ and where $\|\mathbf{w}\|^2 = \langle \boldsymbol{\phi}(x'), \boldsymbol{\phi}(x')\rangle \triangleq k(x', x') = 1$ for a normalized kernel.

The next section presents a new algorithm for solving the problem of Equation (2) that follows from a Lagrangian duality formulation, leading to CqBoost. As we will see, the criterion for incorporating the most promising voter at step t in the quadratic program, will be the one which maximizes a quantity which is often called the *edge*.

4 A MEANINGFUL LAGRANGE DUAL

In this section, we use Lagrange duality techniques to transform the optimization problem of Equation (2) to a dual optimization problem that will be used in Section 5 to develop a column generation algorithm. The reader can refer to Boyd and Vandenberghe [2004] for more information about the techniques that are used in this section.

Directly applying the method of Lagrangian multipliers to a primal optimization problem does not always yield a useful or interesting dual problem. But introducing new variables and equality constraints can sometimes help [Boyd and Vandenberghe, 2004, Section 5.7.1]. As in Shen and Li [2010], Shen et al. [2013], we first introduce a new vector of variables $\boldsymbol{\gamma}$ representing the margin on each example, and its associated equality constraints forcing each γ_k to be equal to $y_k \sum_{i=1}^{n} q_i H_{ki}$. By adding these variables in the primal optimization problem, we obtain a meaningful dual formulation. The new primal problem then becomes

$$\begin{aligned}
&\text{Solve:} \quad \underset{\mathbf{q},\boldsymbol{\gamma}}{\text{argmin}} \ \frac{1}{m}\boldsymbol{\gamma}^\top\boldsymbol{\gamma} \\
&\text{subject to:} \\
&\qquad \boldsymbol{\gamma} \ = \ \text{diag}(\mathbf{y})\mathbf{H}\,\mathbf{q}\,, \quad \frac{1}{m}\mathbf{1}^\top\gamma \ \le \ \mu\,, \\
&\qquad \mathbf{q} \ \succeq \ \mathbf{0}, \qquad \text{and} \qquad \mathbf{1}^\top\mathbf{q} \ = \ 1\,.
\end{aligned} \tag{3}$$

Note that the objective function of Equation (3) is equivalent to the objective function of Equation (2), as $\text{diag}(\mathbf{y})^\top\text{diag}(\mathbf{y}) = \mathbf{I}$. Then, by adding a weighted sum of the constraints of the primal to its objective function, we obtain the following *Lagrangian*:

$$\begin{aligned}
&\Lambda(\mathbf{q}, \boldsymbol{\gamma}, \boldsymbol{\alpha}, \beta, \boldsymbol{\xi}, \nu) \\
&\quad \triangleq \ \frac{1}{m}\boldsymbol{\gamma}^\top\boldsymbol{\gamma} + \boldsymbol{\alpha}^\top\left(\boldsymbol{\gamma} - \text{diag}(\mathbf{y})\mathbf{H}\,\mathbf{q}\right) \\
&\qquad + \beta\left(\frac{1}{m}\mathbf{1}^\top\gamma - \mu\right) - \boldsymbol{\xi}^\top\mathbf{q} + \nu\left(\mathbf{1}^\top\mathbf{q} - 1\right)\,,
\end{aligned} \tag{4}$$

where $\boldsymbol{\alpha}$, β, $\boldsymbol{\xi}$ and ν are *Lagrange multipliers*. The multipliers β and those in $\boldsymbol{\xi}$ are nonnegative as they are related to inequality constraints. Now, the *Lagrangian dual function* is obtained by finding the vectors $\mathbf{q}^\star$ and $\boldsymbol{\gamma}^\star$ minimizing the Lagrangian. The stationarity condition of *Karush-Kuhn-Tucker* (KKT) conditions indicates that this solution is attained when the partial derivatives of the Lagrangian with respect to vectors $\mathbf{q}$ and $\boldsymbol{\gamma}$ are null. Therefore, we need

$$\begin{aligned}
\mathbf{H}^\top\text{diag}(\mathbf{y})\,\boldsymbol{\alpha} \ &= \ -\boldsymbol{\xi} + \nu\mathbf{1}\,, \quad \text{and} \\
\boldsymbol{\gamma}^\star \ &= \ -\frac{m}{2}\boldsymbol{\alpha} - \frac{\beta}{2}\mathbf{1}\,.
\end{aligned} \tag{5}$$

Substituting these expressions for $\boldsymbol{\xi}$ and $\boldsymbol{\gamma}$ in the Lagrangian yields the following dual formulation.[5]

$$\begin{aligned}
&\text{Solve:} \ \underset{\boldsymbol{\alpha},\beta,\boldsymbol{\xi},\nu}{\text{argmax}} \ -\frac{m}{4}\boldsymbol{\alpha}^\top\boldsymbol{\alpha} - \frac{\beta}{2}\mathbf{1}^\top\boldsymbol{\alpha} - \frac{\beta^2}{4} - \beta\mu - \nu \\
&\quad \text{s.t.:} \ \mathbf{H}^\top\text{diag}(\mathbf{y})\,\boldsymbol{\alpha} = -\boldsymbol{\xi} + \nu\mathbf{1}\,, \quad \beta \ge 0\,, \quad \boldsymbol{\xi} \succeq \mathbf{0}\,.
\end{aligned}$$

The non-negative $\boldsymbol{\xi}$ vector being absent from the objective function, its only role is to affect the constraint on the $\boldsymbol{\alpha}$ vector. Thus, we can rewrite the dual problem the following minimization problem:

$$\begin{aligned}
&\text{Solve:} \ \underset{\boldsymbol{\alpha},\beta,\nu}{\text{argmin}} \ \frac{m}{4}\boldsymbol{\alpha}^\top\boldsymbol{\alpha} + \frac{\beta}{2}\mathbf{1}^\top\boldsymbol{\alpha} + \frac{\beta^2}{4} + \beta\mu + \nu \\
&\quad \text{s.t.:} \ \mathbf{H}^\top\text{diag}(\mathbf{y})\,\boldsymbol{\alpha} \ \preceq \ \nu\mathbf{1}\,, \quad \beta \ \ge \ 0\,.
\end{aligned} \tag{6}$$

This dual formulation highlights a variable β that controls the trade-off between the quadratic and linear parts of the objective function. The term $\mathbf{H}^\top\text{diag}(\mathbf{y})\,\boldsymbol{\alpha}$ of Equation (6) can be seen as a "score" given to each voter h, and corresponds to the weighted sum of correctly classified examples minus the weighed sum of incorrectly classified examples (when the voters are classifiers). This measure is often called the *edge* of a voter [Demiriz et al., 2002, Shen et al., 2013], and can be used to guide the choice of the next base voter to be added in the majority vote. Starting from the primal optimization problem of Equation (3) which minimizes the generalization bound of Theorem 2, we are thus able to recover the same criterion as in many column generation techniques [Demiriz et al., 2002, Bi et al., 2004, Shen and Li, 2010, Shen et al., 2013]. In the next section, we develop a CG algorithm using this insight.

5 A CG ALGORITHM THAT MINIMIZES THE $\mathcal{C}$-BOUND

The column generation approach [Nash and Sofer, 1996] has been used to create tractable boosting algorithms using linear programming [Demiriz et al., 2002]

[5]Calculation details of the last two steps can be found in Appendix B.

Algorithm 1 CqBoost

```
– Let $\mathbf{q}$ be a vector of $n$ zeros, let $\boldsymbol{\alpha}$ be a vector of $m$ values equal to $\frac{1}{m}$ and let et $\hat{\mathbf{H}}$ be an empty matrix.
loop
    – Select the column $i$ violating the most Eq. (6) constraint (i.e., the voter with the steepest edge).
    – Break if column $i$ does not violate dual constraint: if $\sum_{k=1}^{m} \alpha_k y_k H_{ki} \ \leq \ \nu + \epsilon$ then break.
    – Add the $i$-th column of $\mathbf{H}$ to matrix $\hat{\mathbf{H}}$.
    – Update $\mathbf{q}$, $\boldsymbol{\alpha}$ and $\nu$ by solving either the primal or dual optimization problem of Equations (3) or (6),
      using matrix $\hat{\mathbf{H}}$ (of size $m{\times}t$, where $t$ is the number of columns generated so far).
return q
```

and quadratic programming [Bi et al., 2004, Shen and Li, 2010]. More generally, it can be used to solve any program based on a convex objective function and regularization term [Shen et al., 2013]. The general idea is to restrict the original optimization problem by considering only a subset of the possible voters, which are columns of the classification matrix associated to the problem. As the ignored columns of the restricted primal problem represent ignored constraints in the dual problem, solving the former corresponds to solving a relaxation of the latter. CG techniques iteratively select (or generate, when considering infinite sets of voters $\mathcal{H}$) a new column to be added the problem, and solve either the primal or dual problem using this subset. The algorithm stops when no more columns violate the dual constraint (up to the desired accuracy ϵ), as optimality is attained.[6] The value of ϵ can be tuned, and can act as a stopping criterion for the algorithm.

Algorithm 1 shows the pseudo-code of CqBoost. To simplify the notation and ease the empirical comparison with other algorithms, we restrict the problem to finite sets of voters and we following Demiriz et al. [2002], Bi et al. [2004] and Shen and Li [2010], we formulate the algorithm as if all the hypotheses had already been generated, that is, we consider that the classification matrix $\mathbf{H}$ is computed a priori.[7] At each iteration, CqBoost computes $\hat{\mathbf{H}}^\top\hat{\mathbf{H}}$ in $\mathcal{O}(m{\times}t^2)$ and solves a QP in $\mathcal{O}(t^3)$, where t is only the number of columns generated so far, as opposed with MinCq that is not iterative but considers all n voters.

Algorithm 1 is initialized in the degenerate case where no column have been chosen yet. The vector $\mathbf{q}$ is therefore initialized to $\mathbf{0}$. Having no prior knowledge on the weights that should be given to each example in the dual formulation of the problem, we initialize the vector $\boldsymbol{\alpha}$ to a uniform distribution, as in Demiriz et al. [2002], Bi et al. [2004] and Shen and Li [2010], respectively for LPBoost, CG-Boost and MDBoost.

[6]At each iteration, any column that violates the dual constraint can be selected (or generated). However, choosing one that maximally violates the constraint may lead to even faster convergence [Demiriz et al., 2002].

[7]The CG technique presented in this paper is not restricted to finite sets of voters $\mathcal{H}$. Using the two continuous sets presented in Section 2 is also theoretically grounded by the generalization bound of Section 3. Doing so opens many questions that we intend to tackle as future work.

6 RELATIONS WITH OTHER APPROACHES

In this section, we discuss the similarities and differences of three related learning algorithms, namely LPBoost [Demiriz et al., 2002], CG-Boost [Bi et al., 2004] and MDBoost [Shen and Li, 2010].

LPBoost is a column generation technique based on a linear programming algorithm with L_1-norm regularization. The primal and dual problems are :

$$\begin{array}{lll} \textbf{Primal:} & \text{Solve:} & \underset{\mathbf{q},\boldsymbol{\xi}}{\text{argmin}} \ \ \mathbf{1}^\top\mathbf{q} + C\ \mathbf{1}^\top\boldsymbol{\xi} \\ & \text{subject to:} & \text{diag}(\mathbf{y})\mathbf{H}\,\mathbf{q} \ \preceq \ \mathbf{1} - \boldsymbol{\xi}, \\ & & \mathbf{q} \ \succeq \ \mathbf{0}, \quad \boldsymbol{\xi} \ \succeq \ \mathbf{0}, \end{array}$$

$$\begin{array}{lll} \textbf{Dual:} & \text{Solve:} & \underset{\boldsymbol{\alpha}}{\text{argmax}} \ \ \mathbf{1}^\top\boldsymbol{\alpha} \\ & \text{subject to:} & \mathbf{H}^\top\text{diag}(\mathbf{y})\,\boldsymbol{\alpha} \preceq \mathbf{1}, \\ & & \mathbf{0} \ \preceq \ \boldsymbol{\alpha} \preceq \ C\,\mathbf{1}, \end{array}$$

where $\mathbf{q}$ is a weight vector of n elements, $\boldsymbol{\xi}$ is a vector of m slack variables related to the margin on each example, where $C > 0$ is a regularization parameter, and $\boldsymbol{\alpha}$ is a weight vector of m elements.

We first observe that the dual of LPBoost is similar to the dual of CqBoost (Equation (6)). In both cases, the $\boldsymbol{\alpha}$ vector represents weights on the examples, and the inequality constraint that guides the CG algorithm into choosing the next column is very similar. Also note that LPBoost considers a L_1-norm regularization on $\mathbf{q}$, which will directly penalize dense solutions.

CG-Boost is a family of column generation algorithms that generalize LPBoost to L_2-regularized objective functions. The main difference with LPBoost is that the weights $\mathbf{q}$ are regularized with a L_2 norm, transforming the original linear program into a quadratic program. Note that the primal objective function of CqBoost (Equation (3)) is also quadratic. The primal and dual optimization problems of the main result are as follows:

$$\textbf{Primal:}\quad \text{Solve:} \quad \underset{\mathbf{q},\boldsymbol{\xi}}{\operatorname{argmin}} \ \frac{1}{2}\mathbf{q}^\top\mathbf{q} + C\ \mathbf{1}^\top\boldsymbol{\xi}$$
$$\text{subject to:} \quad \operatorname{diag}(\mathbf{y})\mathbf{H}\,\mathbf{q} + \boldsymbol{\xi} \ \preceq\ \mathbf{1},$$
$$\mathbf{q}\ \succeq\ \mathbf{0}, \quad \boldsymbol{\xi}\ \succeq\ \mathbf{0},$$

$$\textbf{Dual:}\quad \text{Solve:} \quad \underset{\boldsymbol{\alpha}}{\operatorname{argmax}}\ \underset{\mathbf{q}}{\operatorname{argmin}}\ \mathbf{1}^\top\boldsymbol{\alpha} - \frac{1}{2}\mathbf{q}^\top\mathbf{q}$$
$$\text{subject to:} \quad \mathbf{H}^\top\operatorname{diag}(\mathbf{y})\,\boldsymbol{\alpha}\ \preceq\ \mathbf{q},$$
$$\mathbf{0}\ \preceq\ \boldsymbol{\alpha} \preceq\ C\,\mathbf{1},$$

where $\mathbf{q}$, $\boldsymbol{\xi}$ and $\boldsymbol{\alpha}$ have the same meaning as in the LP-Boost, but where the primal variables vector $\mathbf{q}$ remains in the dual. Nevertheless, CG-Boost's dual has a similar inequality constraint that yields the same heuristic as LPBoost and CqBoost for choosing the column to add at each iteration.

MDBoost starts with the same goals in mind than Cq-Boost: its optimization function directly deals with the margin distribution. Its objective function minimizes the variance of the margin, while maximizing its first moment. A hyperparameter D controls the trade-off between these two quantities. When written using the first and second moments of the margin, the objective function of MDBoost corresponds to minimizing

$$\frac{Dm}{2(m-1)}\left(\mu_2(M_Q^S) - \left(\mu_1(M_Q^S)\right)^2\right) - \mu_1(M_Q^S)\,.$$

It is noteworthy to point out that the strategy of minimizing the second moment of the margin while maximizing the first moment is indirectly justified by the $\mathcal{C}$-bound. The trade-off between the first two moments is, however, not the same as CqBoost, whose objective function is directly inspired by this upper bound on the risk of the majority vote. When comparing both objective functions, we see that MDBoost will have a tendency to choose margin distributions with a higher first moment, whereas CqBoost provides a direct control over the first moment. The primal and dual problems of MDBoost are as follows:

$$\textbf{Primal:}\quad \text{Solve:} \quad \underset{\mathbf{q},\boldsymbol{\gamma}}{\operatorname{argmin}}\ \frac{D}{2}\,\boldsymbol{\gamma}^\top\mathbf{A}\boldsymbol{\gamma}\ -\ \mathbf{1}^\top\boldsymbol{\gamma}$$
$$\text{subject to:} \quad \boldsymbol{\gamma}\ =\ \operatorname{diag}(\mathbf{y})\mathbf{H}\,\mathbf{q}\,,$$
$$\mathbf{q}\ \succeq\ \mathbf{0}, \quad \mathbf{1}^\top\mathbf{q}\ =\ 1\,,$$

$$\textbf{Dual:}\quad \text{Solve:}\ \underset{\boldsymbol{\alpha},\nu}{\operatorname{argmin}}\frac{1}{2}\,(\boldsymbol{\alpha}-\mathbf{1})^\top\mathbf{A}^\dagger\,(\boldsymbol{\alpha}-\mathbf{1}) + D\nu$$
$$\text{subject to:} \quad \mathbf{H}^\top\operatorname{diag}(\mathbf{y})\,\boldsymbol{\alpha}\ \preceq\ \nu\mathbf{1}\,,$$

where $\mathbf{A}$ is a $m{\times}m$ matrix, where $A_{ii} = 1$ for $i \in \{1,\ldots,m\}$, $A_{ij} = \frac{-1}{m-1}$ for $i,j \in \{1,\ldots,m\}$ and $i \neq j$, and where $\mathbf{A}^\dagger$ is its pseudo-inverse. We once again notice that the strategy for selecting the next column in the resulting CG technique is the same as all aforementioned algorithms. Also, to ease the comparison with CqBoost, the primal and dual problems above correspond to a version of MDBoost that is stated if terms of the *normalized* margin.

Finally, note that Shen et al. [2013] have developed a general column generation framework named CGBoost (not to be confused with CG-Boost of Bi et al. [2004]) from whom one may recover many column generation algorithms variants, by carefully choosing a loss function and a regularizer, and we might be able to recover the optimization problem of CqBoost within this framework. However, as we have shown, CqBoost is motivated by a PAC-Bayesian upper bound on the risk of the majority vote classifier. It is currently unknown if the general form proposed by Shen et al. [2013] can be motivated by a risk bound.

7 EMPIRICAL EXPERIMENTS

We now compare how CqBoost performs in terms of accuracy and sparsity, first against related algorithms MDBoost, LPBoost and CG-Boost, but also against AdaBoost [Schapire and Singer, 1999] and the original version of MinCq as proposed in Laviolette et al. [2011]. We consider *decision stumps* (one-level decision trees) as base voters. For each attribute, 10 pairs of decision stumps (with the same decision threshold but inverse decisions) are generated.

We run all algorithms on binary classification datasets of the UCI Machine Learning Repository [Lichman, 2013], normalized using a hyperbolic tangent function. For each dataset, half of the examples are randomly chosen to be in the training set S of at most 500 examples, and the remaining testing examples T are used to evaluate the performance of the algorithms. The hyperparameter value of each algorithm has been selected by 5-folds cross-validation on the training set, among 15 values on a logarithmic scale. The value of hyperparameter μ of CqBoost and MinCq is selected among values between 10^{-2} and $10^{-0.5}$. The value of hyperparameter D of MDBoost is chosen in values between 10^0 and 10^2. The value of hyperparameter C of CG-Boost and LPBoost is selected in values between 10^{-3} and 10^3. The number of iterations of AdaBoost is selected among values between 10^2 and 10^6. The stopping criterion additive constant ϵ of all column generation algorithms has been set to 10^{-6}. All linear and quadratic programs have been solved using the CVXOPT solver [Dahl and Vandenberghe, 2007].

	CqBoost		MDBoost		LPBoost		CG-Boost		AdaBoost		MinCq	
Dataset	Risk	Cols.	Risk	Cols.	Risk	Cols.	Risk	Cols.	Risk	Cols.	Risk	Cols.
australian	**0.151**	40	0.159	63	0.171	**39***	0.200	140	0.191	42	0.145*	280
balance	0.073	**16***	0.073	**16***	**0.029**	**16***	**0.029**	40	0.026*	16*	0.029	80
breast	0.037	46	0.037	32	0.037	**20***	**0.034***	90	0.046	30	0.037	180
bupa	**0.289***	31	**0.289***	**23***	0.312	39	0.324	60	0.289*	38	0.329	120
car	0.140	**16**	0.140	17	**0.138**	**16**	**0.138**	66	0.130*	13*	0.141	120
cmc	**0.300***	21	0.302	20	0.311	**18***	0.303	90	0.318	34	0.305	180
credit	0.133	41	**0.125***	25	0.133	**1***	0.130	150	0.165	34	0.128	300
cylinder	0.330	**18***	0.311	19	0.278	52	**0.256***	353	0.274	90	0.300	700
ecoli	**0.054***	27	0.060	**22***	0.095	25	0.083	70	0.095	22*	0.065	140
glass	0.215	45	**0.206***	55	0.308	**43**	0.224	90	0.215	33*	0.271	180
heart	**0.185**	26	0.200	21	0.193	**14***	0.200	130	0.215	36	0.170*	260
hepatitis	**0.143***	43	0.208	**9***	0.156	17	0.195	190	0.182	26	0.169	380
horse	**0.174***	43	**0.174***	40	0.207	**2***	0.179	260	0.185	44	0.223	520
ionosphere	**0.091***	121	0.097	83	0.114	**63**	**0.091***	340	0.114	52*	0.109	680
letter:ab	**0.009**	61	**0.009**	47	0.011	**41***	0.012	160	0.010	55	0.005*	320
monks	**0.231***	**10***	0.236	15	**0.231***	11	**0.231***	117	0.255	15	0.236	120
optdigits	0.084	174	0.084	150	0.083	**87***	**0.080***	640	0.084	132	0.090	1280
pima	**0.237***	**26***	0.250	35	0.279	50	0.250	80	0.273	47	0.242	160
titanic	**0.222**	6	**0.222**	8	**0.222**	**5***	**0.222**	50	0.222	7	0.212*	60
vote	**0.051**	33	**0.051**	20	**0.051**	**1***	**0.051**	155	0.046*	19	0.051	320
wine	**0.067**	49	**0.067**	54	**0.067**	**29***	0.079	130	0.045*	31	0.056	260
yeast	0.291	19	0.292	**15***	0.299	27	**0.289***	80	0.300	38	0.303	160
zoo	**0.039***	20	**0.039***	54	0.118	**9***	0.137	157	0.118	10	0.039*	320

Table 1: Performance and sparsity comparison of CqBoost, MDBoost, LPBoost, CG-Boost, AdaBoost and MinCq, using decision stumps as weak classifiers. A bold value indicates that the risk (or number of chosen columns) is the lowest among the column generation algorithms. A star indicates that the risk is the lowest among all six algorithms.

Table 1 show for each algorithm the testing risks and the number of non-zero weights in the output majority vote (associated to the chosen columns). In terms of accuracy, we observe that CqBoost is very competitive with all other column generation algorithms, winning (or tying) 15 times on 23 datasets. The runner-up is CG-Boost with 10 wins, closely followed by MDBoost with 9 wins, and finally LPBoost with only 6 wins. When comparing all algorithms together, CqBoost is also the algorithm that collects the highest number of wins. However note that using the *sign test* [Mendenhall, 1983], the only statistically significant results are that CqBoost outperforms LPBoost (p-value of 0.02) and that MDBoost outperforms AdaBoost (p-value of 0.04). Moreover comparing CqBoost only with MDBoost, CqBoost shows a slight advantage 10 wins and 3 losses, with a corresponding sign test p-value of 0.13. We conjecture that the direct control over the first moment of the margin provided by CqBoost is advantageous when using voters as weak as decision stumps.

In terms of sparsity, we observe that CqBoost reaches its goal of outputting significantly sparser solutions than MinCq, while keeping a similar performance. We also observe that, as expected, LPBoost is the algorithm that produces the sparsest solutions. Its accuracy is, however, significantly worse than CqBoost according to the sign test. Hence, CqBoost is an accurate algorithm that also gives sparse solutions, as wanted.

We also compared all above-mentioned algorithms using *Radial Basis Function* (RBF) kernels as voters, along with the Support Vector Machine (SVM) algorithm [Cortes and Vapnik, 1995]. More details and empirical results can be found in Appendix C of the supplementary material.

8 CONCLUSION

In this paper, we designed a column generation ensemble method based on MinCq [Laviolette et al., 2011, Germain et al., 2015] that outputs much sparser ensembles and can consider larger (or even infinite) sets of voters. The algorithm is motivated by a generalization bound that holds for finite and two cases of continuous sets of base classifiers. Empirically, CqBoost performs as well as the original algorithm, and slightly outperforms other column generation algorithms.

In future work, we will investigate how CqBoost compares with other boosting algorithms when using infinite sets of classifiers. We will also extend the algorithm to take into account informative prior information, as was done for MinCq by Bellet et al. [2014]. Finally, we will investigate the effect of ϵ as a stopping criterion to control the sparsity of the ensemble.

Acknowledgements

This work has been supported by National Science and Engineering Research Council (NSERC) Discovery grants 262067 and 0122405.

References

Aurélien Bellet, Amaury Habrard, Emilie Morvant, and Marc Sebban. Learning a priori constrained weighted majority votes. *Machine Learning*, 97(1-2):129–154, 2014.

Jinbo Bi, Tong Zhang, and Kristin P. Bennett. Column-generation boosting methods for mixture of kernels. *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04*, page 521, 2004.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, New York, NY, USA, mar 2004. ISBN 0521833787.

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

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

Olivier Catoni. *PAC-Bayesian supervised classification: the thermodynamics of statistical learning.* Monograph series of the Institute of Mathematical Statistics, 2007.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

Joachim Dahl and Lieven Vandenberghe. CVXOPT, 2007. `http://mloss.org/software/view/34/`.

Ayhan Demiriz, Kristin P Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1-3):225–254, 2002.

Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis, Third Edition.* Chapman & Hall/CRC Texts in Statistical Science. ISBN 9781439840955.

Pascal Germain, Alexandre Lacasse, François Laviolette, and Mario Marchand. PAC-Bayesian learning of linear classifiers. In *ICML*, page 45, 2009.

Pascal Germain, Alexandre Lacoste, François Laviolette, Mario Marchand, and Sara Shanian. A PAC-Bayes sample-compression approach to kernel methods. In *ICML*, pages 297–304, 2011.

Pascal Germain, Alexandre Lacasse, Francois Laviolette, Mario Marchand, and Jean-Francis Roy. Risk bounds for the majority vote: From a PAC-Bayesian analysis to a learning algorithm. *Journal of Machine Learning Research*, 16:787–860, 2015.

Alexandre Lacasse, François Laviolette, Mario Marchand, Pascal Germain, and Nicolas Usunier. PAC-Bayes bounds for the risk of the majority vote and the variance of the Gibbs classifier. In *NIPS*, pages 769–776, 2006.

John Langford and John Shawe-Taylor. PAC-Bayes & margins. In *NIPS*, pages 423–430, 2002.

François Laviolette, Mario Marchand, and Jean-Francis Roy. From PAC-Bayes bounds to quadratic programs for majority votes. In *ICML*, pages 649–656, 2011.

M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Andreas Maurer. A note on the PAC-Bayesian theorem. *CoRR*, cs.LG/0411099, 2004.

D. McAllester. Simplified PAC-Bayesian margin bounds. In *Learning Theory and Kernel Machines*, pages 203–215. Springer, 2003.

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

David McAllester. A PAC-Bayesian tutorial with a dropout bound. *CoRR*, abs/1307.2118, 2013.

W. Mendenhall. Nonparametric statistics. *Introduction to Probability and Statistics*, 604, 1983.

Emilie Morvant. Domain adaptation of weighted majority votes via perturbed variation-based self-labeling. *Pattern Recognition Letters*, 51:37–43, 2015.

Emilie Morvant, Amaury Habrard, and Stéphane Ayache. Majority vote of diverse classifiers for late fusion. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 153–162. Springer, 2014.

Stephen Nash and Ariela Sofer. *Linear and Nonlinear Programming.* McGraw-Hill series in industrial engineering and management science. McGraw-Hill, 1996. ISBN 9780070460652.

Robert E. Schapire and Yoram Singer. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

Matthias Seeger. PAC-Bayesian generalisation error bounds for gaussian process classification. *The Journal of Machine Learning Research*, 3:233–269, 2003.

Yevgeny Seldin and Naftali Tishby. PAC-Bayesian analysis of co-clustering and beyond. *Journal of Machine Learning Research*, 11:3595–3646, 2010.

Chunhua Shen and Hanxi Li. Boosting through optimization of margin distributions. *Neural Networks, IEEE Transactions on*, 21(4):659–666, April 2010.

Chunhua Shen, Hanxi Li, and Anton van den Hengel. Fully corrective boosting with arbitrary loss and regularization. *Neural networks : the official journal of the International Neural Network Society*, 48:44–58, dec 2013. ISSN 1879-2782.