CqBoost : A Column Generation Method for Minimizing the C-Bound

François Laviolette, Mario Marchand, Jean-Francis Roy
Département d'informatique et de génie logiciel, Université Laval
Québec (QC), Canada
firstname.lastname@ift.ulaval.ca

Abstract

The $\mathcal{C}$-bound, introduced in Lacasse et al. [1], gives a tight upper bound of a majority vote classifier. Laviolette et al. [2] designed a learning algorithm named MinCq that outputs a dense distribution on a set of base classifiers by minimizing the $\mathcal{C}$-bound, together with a PAC-Bayesian generalization guarantee. In this work, we use optimization techniques to design a column generation algorithm that optimizes the $\mathcal{C}$-bound and outputs particularly sparse solutions.

1 Introduction

Many state-of-the-art binary classification learning algorithms, such as Bagging [3], Boosting [4], and Random Forests [5], 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. [6] for an introductory text). Moreover, classifiers produced by kernel methods, such as the Support Vector Machine [7], 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).

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 [2]. Unfortunately, in most ensemble methods, the voters are weak and no majority vote can achieve a large margin. Lacasse et al. [1] 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 sheds a new light on the behavior of majority votes : it is not only how good are the voters but also how they are correlated in their voting. Namely, this has inspired a new learning algorithm, named MinCq [2], whose performance is state-of-the-art.

MinCq turns out to be a quadratic program (QP) that minimizes the $\mathcal{C}$-bound in a way that produces dense weight distributions. In many applications of machine learning, sparse solutions are preferred as the learned classification rules are easier to explain, and the resulting classifier is computationally faster. In this work, we tackle this issue by using optimization techniques to design CqBoost, a learning algorithm that produces much sparser solutions.

This paper is organized as follows. Section 2 briefly reviews the $\mathcal{C}$-bound and the MinCq learning algorithm. Section 3 presents a Lagrange dual for MinCq. Section 4 presents the resulting column generation algorithm together with empirical results. Finally, we conclude in Section 5.

2 The $\mathcal{C}$-bound and MinCq

Let $\mathcal{X}$ be the input space and let $\mathcal{Y} = \{-1, +1\}$ be 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 set of real-valued voters from $\mathcal{X}$ to $\mathcal{Y}$. Given a prior distribution P over $\mathcal{H}$ and given a sample S, the objective of the PAC-Bayesian approach is to find the posterior distribution Q on $\mathcal{H}$ which minimizes the true risk of the Q-weighted majority vote $B_Q(\cdot)$ given by

$$R_D(B_Q) \triangleq \mathop{\mathbf{E}}_{(x,y)\sim D} I\left(B_Q(x) \neq y\right), \quad \text{where} \quad B_Q(x) \triangleq \operatorname{sgn}\left[\mathop{\mathbf{E}}_{h\sim Q} h(x)\right]$$

and 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 make use of the $\mathcal{C}$-bound which is a tight upper-bound on $R_D(B_Q)$. This bound is based on the notion of margin of $B_Q(\cdot)$ defined as follows.

Definition 1. *Given a set $\mathcal{H}$ of voters 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 $\mathcal{X} \times \mathcal{Y}$, we then consider the first and second statistical moments of M_Q, given, respectively, by $\mu_1(M_Q^D) \triangleq \mathbf{E}_{(x,y)\sim D} M_Q(x, y)$ and $\mu_2(M_Q^D) \triangleq \mathbf{E}_{(x,y)\sim D} M_Q^2(x, y)$.

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 to prove the following theorem.

Theorem 1 (The $\mathcal{C}$-bound of Laviolette et al. [2])**.** *For every distribution Q on a set of real-valued functions $\mathcal{H}$, and for every distribution P 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 any random variable Z and any $a > 0$, we have that $\mathbf{Pr}\left(Z \leq \mathbf{E}\left[Z\right] - a\right) \;\leq\; \frac{\mathbf{Var}\, Z}{\mathbf{Var}\, Z + a^2}$. We obtain the result by applying this inequality with $Z = M_Q(x, y)$ and with $a = \mu_1(M_Q^D)$, and by using the definition of the variance. □

Note that 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 error. This strategy is justified by a PAC-Bayesian generalization bound over the $\mathcal{C}$-bound, and has given the MinCq algorithm. Let $\mathbf{H}$ be the *classification matrix* of m rows and n columns, where $H_{ki} = h_i(x_k)$. Each column represents a classifier of $\mathcal{H}$, and each line represents an example. Let $\mathbf{y}$ be the (column) vector containing the labels. MinCq solves the following quadratic program.[1]

$$\begin{aligned} \text{Solve :} \quad & \operatorname*{argmin}_{\mathbf{q}} \; \frac{1}{m}\mathbf{q}^\top \mathbf{H}^\top \mathbf{H}\, \mathbf{q} \\ \text{subject to :} \quad & \frac{1}{m}\mathbf{y}^\top \mathbf{H}\, \mathbf{q} \;=\; \mu\,, \quad \text{and} \quad \mathbf{q} \;\succeq\; \mathbf{0}\,, \end{aligned} \tag{1}$$

where μ is an hyperparameter controlling the value of the first moment of the margin, $\mathbf{q}$ is a vector of n variables representing the weights associated to the voters, and $\mathbf{0}$ is a vector containing zeros.

[1] The version of MinCq in Laviolette et al. [2] is slightly more restrictive by requiring the set $\mathcal{H}$ to be *auto-complemented* (meaning that each voter for each h, $\mathcal{H}$ must also contains its complement $-h$) together with another restriction on Q, in order to obtain a tighter PAC-Bayesian bound. We report here a relaxed version to ease the forthcoming analysis. This version also has a PAC-Bayesian generalization guarantee with the price of an additional Kullback-Leibler term in the bound.

3 A Meaningful Lagrange Dual

In this section, we use optimization techniques to transform the optimization problem of Equation (1) to a dual optimization problem that will be used in Section 4 to develop a column generation algorithm. See Boyd et al. [8] for more information about the techniques that are used here.

As in [9, 10], we first introduce a new vector of variables $\boldsymbol{\gamma}$ representing the margin on each example. By adding these variables in the primal optimization problem, we obtain a meaningful dual formulation. The new primal problem then becomes

$$\begin{aligned} \text{Solve :} \quad & \underset{\mathbf{q},\boldsymbol{\gamma}}{\operatorname{argmin}} \ \frac{1}{m}\boldsymbol{\gamma}^\top\boldsymbol{\gamma} \\ \text{subject to :} \quad & \boldsymbol{\gamma} \ = \ \operatorname{diag}(\mathbf{y})\mathbf{H}\,\mathbf{q}\,, \quad \frac{1}{m}\mathbf{1}^\top\boldsymbol{\gamma} = \ \mu\,, \quad \text{and} \quad \mathbf{q} \ \succeq \ \mathbf{0}\,. \end{aligned} \tag{2}$$

Note that the objective function of Equation (2) is equivalent to the objective function of Equation (1), as $\operatorname{diag}(\mathbf{y})^\top\operatorname{diag}(\mathbf{y}) \ = \ \mathbf{I}$. Then, by adding a weighted sum of the constraints to the objective function, we obtain the *Lagrangian* of the optimization problem of Equation (2):

$$\Lambda(\mathbf{q},\boldsymbol{\gamma},\boldsymbol{\alpha},\beta,\boldsymbol{\xi}) \triangleq \frac{1}{m}\boldsymbol{\gamma}^\top\boldsymbol{\gamma} + \boldsymbol{\alpha}^\top\left(\boldsymbol{\gamma} - \operatorname{diag}(\mathbf{y})\mathbf{H}\,\mathbf{q}\right) + \beta\left(\frac{1}{m}\mathbf{1}^\top\boldsymbol{\gamma} - \mu\right) - \boldsymbol{\xi}^\top\mathbf{q}\,,$$

where $\boldsymbol{\alpha}$, β and $\boldsymbol{\xi}$ are *Lagrange multipliers*, and the multipliers in $\boldsymbol{\xi}$ are nonnegative as they are related to inequality constraints. Now, the *Lagrangian dual function* is obtained by finding the vectors $\mathbf{q}$ and $\boldsymbol{\gamma}$ minimizing the Lagrangian. The stationarity condition 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

$$\mathbf{H}^\top\operatorname{diag}(\mathbf{y})\,\boldsymbol{\alpha} \ = \ -\boldsymbol{\xi} \qquad \text{and} \qquad \boldsymbol{\gamma} = -\frac{m}{2}\boldsymbol{\alpha} - \frac{\beta}{2}\mathbf{1}\,.$$

Substituting these constraints in the Lagrangian, by realizing that $\boldsymbol{\xi}$ is a vector of *slack variables*, and with tedious (but straightforward) calculations, we obtain the following dual problem :

$$\begin{aligned} \text{Solve :} \quad & \underset{\boldsymbol{\alpha},\beta}{\operatorname{argmax}} \ -\frac{m}{4}\boldsymbol{\alpha}^\top\boldsymbol{\alpha} - \frac{\beta}{2}\mathbf{1}^\top\boldsymbol{\alpha} - \frac{\beta^2}{4} - \beta\mu \\ \text{subject to :} \quad & \mathbf{H}^\top\operatorname{diag}(\mathbf{y})\,\boldsymbol{\alpha} \ \preceq \ \mathbf{0}\,. \end{aligned} \tag{3}$$

This dual formulation highlights a new hyperparameter β, which controls the trade-off between the quadratic and linear parts of the objective function. Moreover, we recover the same constraint $\mathbf{H}^\top\operatorname{diag}(\mathbf{y})\,\boldsymbol{\alpha} \preceq \mathbf{0}$ as in LPBoost [11], CG-Boost of [12], and CGBoost of [10]. This constraint 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. This measure is often called the *edge* of a classifier [10]. In all three aforementioned column generation (CG) techniques, the algorithm uses this constraint to guide the choice of the next base voter to be added in the majority vote. In the next section, we develop a CG algorithm for MinCq using this insight.

4 A Column Generation Algorithm

The column generation approach [13] has been used to create tractable boosting algorithms using linear programming [11], quadratic programming [12, 9], and in a more general setting allowing any convex objective function and regularization term [10]. The general idea of column generation is to restrict the original optimization problem by considering only a subset of the base classifiers, which are columns of the classification matrix associated to the problem. The restricted optimization problem is called the *restricted master problem*. In an appropriate dual formulation of the original problem, the ignored columns of the primal problem represent ignored constraints in the dual problem. Solving the restricted master problem therefore corresponds to solving a relaxation of the dual problem. CG techniques iteratively select (or generate) a new column to be added the the problem, and solve the restricted master problem. The algorithm stops when no more columns violate the dual constraint (up to an additive user-specified constant ϵ), as optimality is attained. The value of ϵ can be tuned to add a stopping criterion effect.

Algorithm 1 CqBoost

– Let $t = 0$, let $\mathbf{q}$ be a vector of n zeros, and let $\boldsymbol{\alpha}$ be a vector of m elements equal to $\frac{1}{m}$.
– Let $\hat{\mathbf{H}}$ be an empty matrix of m rows.
loop
  – Select the column most violating the constraint of Eq. (3) : $j \leftarrow \text{argmax}_j \sum_{i=1}^{m} \alpha_i y_i H_{ij}$.
  – Break if column j does not violate dual constraint : **if** $\sum_{i=1}^{m} \alpha_i y_i H_{ij} \leq 0+\epsilon$ **then break**.
  – Add column j to matrix $\hat{\mathbf{H}}$.
  – Update $\mathbf{q}$ and $\boldsymbol{\alpha}$ with primal and dual solutions of QP of Equation (2).
return $\mathbf{q}$

In Algorithm 1, we consider that the classification matrix $\mathbf{H}$ is computed a priori. When the number n of classifiers is large (or even infinite), we want to avoid computing this matrix. Many solutions exist, namely by using a stratified CG process [12]. When $\mathbf{H}$ is not computed a priori, the result of a CG algorithm will be an approximation of the optimal solution but it can scale to higher n.

Table 1 compares MinCq and CqBoost on several UCI binary classification datasets [14]. Each dataset is randomly split into a training set S and a testing set T. For both algorithms, we use RBF kernel functions $h_i(\cdot) = y_i\, k(x_i, \cdot)$ as base classifiers, where the width σ is set to the mean squared distance between pairs of training examples. The hyperparameter μ of MinCq is selected by cross-validation among 10 values between 10^{-4} and 10^{-2}, and the same value is then used for CqBoost when solving the QP of Equation (2). The stopping criterion additive constant ϵ of CqBoost has been set to 10^{-6}. For most datasets, MinCq slightly outperforms CqBoost. However, CqBoost provides majority votes of base classifiers that are much sparser compared to MinCq.

Dataset			MinCq			CqBoost	
Name	$\lvert S\rvert$	$\lvert T\rvert$	$R_T(B_Q)$	Non-zero Weights	μ	$R_T(B_Q)$	Non-zero Weights
australian	345	345	0.1449	279	0.0008	**0.1420**	**11**
balance	313	312	0.0577	221	0.0001	**0.0513**	**3**
breast	350	349	**0.0372**	266	0.0001	0.0401	**2**
bupa	173	172	**0.2907**	127	0.0002	**0.2907**	**8**
car	864	864	**0.2836**	644	0.0008	0.2870	**9**
cmc	737	736	**0.2921**	513	0.0002	0.3003	**8**
credit	345	345	0.1362	269	0.0005	**0.1333**	**7**
cylinder	270	270	0.3074	209	0.0002	**0.3037**	**4**
ecoli	168	168	**0.0893**	111	0.0022	0.0952	**12**
flags	97	97	**0.3505**	75	0.0001	0.3918	**4**
glass	107	107	**0.2617**	78	0.0002	0.2804	**10**
heart	135	135	**0.1630**	110	0.0060	**0.1630**	**12**
hepatitis	78	77	0.2078	54	0.0100	**0.1688**	**33**
horse	184	184	0.1848	151	0.0013	**0.1793**	**9**
ionosphere	176	175	**0.1257**	159	0.0001	0.1714	**3**
letter_AB	778	777	**0.0180**	572	0.0008	0.0257	**6**
mnist_08	1208	1208	**0.0480**	1166	0.0002	0.0662	**2**
monks	216	216	**0.3426**	164	0.0003	0.3657	**5**
optdigits	1912	1911	**0.1423**	1725	0.0003	0.1978	**4**
pageblock	2737	2736	**0.0559**	2002	0.0003	0.0625	**6**
pendigits	3747	3747	**0.0806**	3037	0.0003	0.1044	**4**
pima	384	384	0.2630	275	0.0100	**0.2500**	**24**
segment	1155	1155	**0.0684**	1027	0.0002	0.1091	**6**
spambase	2301	2300	**0.0804**	1636	0.0002	0.1109	**3**
tictactoe	479	479	0.3507	305	0.0002	**0.3424**	**5**
titanic	1101	1100	**0.2245**	750	0.0001	0.2345	**5**
vote	218	217	**0.0599**	159	0.0003	0.0737	**3**
wine	89	89	**0.0225**	80	0.0001	0.1124	**2**
yeast	742	742	0.3005	475	0.0003	**0.2898**	**11**
zoo	51	50	0.0400	37	0.0003	**0.0000**	**8**

Table 1: Performance and sparsity comparison of MinCq and CqBoost.

5 Conclusion and Outlooks

In this paper, we designed a column generation ensemble method based on MinCq [2] that provides much sparser (and easier to interpret) ensembles. In future work, we will extend this new CG technique to obtain an algorithm that scales to larger sets of examples and base voters.

References

[1] 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.

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

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

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

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

[6] A. Gelman, J.B. Carlin, H.S. Stern, and D.B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, 2004.

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

[8] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, mar 2004.

[9] Chunhua Shen and Hanxi Li. Boosting through optimization of margin distributions. 21:1–9, 2010.

[10] 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.

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

[12] 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.

[13] S.G. Nash and A. Sofer. *Linear and Nonlinear Programming*. McGraw-Hill series in industrial engineering and management science. McGraw-Hill, 1996.

[14] C.L. Blake and C.J. Merz. *UCI Repository of machine learning databases*. Department of Information and Computer Science, Irvine, CA: University of California, http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998.