Fully Corrective Boosting with Arbitrary Loss and Regularization

Chunhua Shen[a,1,*], Hanxi Li[b], Anton van den Hengel[a]

[a] *School of Computer Science, The University of Adelaide, Adelaide, SA 5005, Australia*
[b] *NICTA, Canberra Research Laboratory, ACT 0200, Australia*

Abstract

We propose a general framework for analyzing and developing fully corrective boosting-based classifiers. The framework accepts any convex objective function, and allows any convex (for example, ℓ_p-norm, $p \geq 1$) regularization term. By placing the wide variety of existing fully corrective boosting-based classifiers on a common footing, and considering the primal and dual problems together, the framework allows direct comparison between apparently disparate methods. By solving the primal rather than the dual the framework is capable of generating efficient fully-corrective boosting algorithms without recourse to sophisticated convex optimization processes. We show that a range of additional boosting-based algorithms can be incorporated into the framework despite not being fully corrective. Finally, we provide an empirical analysis of the performance of a variety of the most significant boosting-based classifiers on a few machine learning benchmark datasets.

Keywords:
Boosting, ensemble learning, convex optimization, column generation.

---

1. Introduction

Boosting has become one of the best known methods for building highly accurate classifiers and regressors from a set of weak learners [1]. As a result, significant research effort has been applied to both extending and understanding boosting (see [2, 3, 4, 5, 6] amongst many others). Totally corrective boosting, which aims to achieve classification efficiency without sacrificing effectiveness, has particularly given rise to a wide variety of competing analyses and methods. We present here a framework which not only allows the consolidation and comparison of this important work, but which also generalises the underlying approach. More importantly, however, the framework also enables direct and immediate derivation of a classifier implementation for the wide variety of cost functions and regularization terms it accepts.

Much effort has been devoted to analyzing AdaBoost [2] and other boosting algorithms [3, 4, 5] due to their great success in both classification and regression when applied to a wide variety of computer vision and machine learning tasks (see [7, 8] for example). Both theoretical and experimental results have shown that boosting algorithms have impressive generalization performance.

Researchers have been trying to interpret the success of boosting from a variety of different perspectives. Early work focused on developing theories in the framework of probably approximately correct (PAC) learning [9] or the large margin principle [2]. Friedman *et al.* [10] developed a statistical perspective that views AdaBoost as a gradient-based stage-wise optimization method in a functional space, minimizing the exponential loss function $l(y, F) = \exp(-yF)$. AnyBoost [11, 12] generalizes this concept in the sense that AnyBoost can optimize a broader family of loss functions. Hereafter, we use the term "AnyBoost" to refer to

[*] To whom all correspondence should be addressed (email: chunhua.shen@adelaide.edu.au).
[1] This work is in part supported by Australian Research Council Future Fellowship FT120100969.
[2] NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications, and the Digital Economy, and the Australian Research Council through the ICT Center of Excellence Program.

all gradient based boosting methods, as their theoretical essentials are almost identical. For example, within the AnyBoost framework, one can optimize the binomial log-likelihood loss $l(y, F) = \log(1 + \exp(yF))$, which penalizes a mis-classified point with less penalty than the exponential loss, in the hope that it might be more robust to outliers. In [11], a non-convex loss function was used in order to achieve a better margin distribution, which can be translated into a smaller test error rate. Shen and Li [13] explicitly derived a Lagrange dual of ℓ_1 regularized boosting for a variety of common loss functions. The relationship between these dual formulations and the soft-margin LPBoost [3] was established in [13].

Rosset *et al*. [14] observed that asymptotically stage-wise boosting converges to a ℓ_1 regularized solution. They deliberately set the coefficient of the weak classifier to a very small value (ε-boost) such that the boosting method converges extremely slowly. The slow rate of convergence plays the role of ℓ_1 regularization as we will discuss in detail later. It is not new to impose regularization other than ℓ_1 in boosting. In [11], ℓ_2 norm regularized boosting has been considered and gradient based boosting is used to perform the optimization. In [15], Duchi and Singer introduced a family of coordinate-descent methods for optimizing the upper-bounds of mixed-norm regularized boosting based on gradient boosting [11]. In contrast to conventional gradient-based boosting, the authors there also prune selected features that are not informative, thus sharing conceptual similarities with the FloatBoost of Li and Zhang [16] and Zhang's forward-backward sparse learning [17]. Duchi and Singer mainly focused on learning with structural sparsity in the context of multi-class and/or multi-task applications. Both works are largely inspired by gradient AnyBoost and no analysis of Lagrange duality was performed. Most of the boosting algorithms in the literature can thus be seen as building upon the gradient-based boosting of AnyBoost.

AnyBoost [11] is a seminal work in the sense that it enables one to design boosting algorithms for optimizing a given cost function. It uses coordinate descent, however, and is therefore not fully (totally) corrective. Totally corrective boosting algorithms, like LPBoost [3], TotalBoost [18] and those proposed in [13], update the coefficients of all previously selected weak learners at each iteration. The fully corrective boosting algorithms thus require significantly fewer training iterations to achieve convergence [13] and result in smaller, and therefore more efficient, ensemble classifiers. Although it is not discussed in [11], the slow convergence rate of AnyBoost is critical to its success, as it plays the role of the ℓ_1 norm regularization parameter [14]. As an illustration of the importance of AnyBoost's slow convergence to its success consider the case where the training data are separable, which leaves AdaBoost's objective function $\sum_i \exp(-\sum_j y_i w_j h_j(\boldsymbol{x}_i))$ not well defined. In fact the objective function can always be rendered arbitrarily close to zero by multiplying $\boldsymbol{w}$ by a large enough positive factor.

In contrast to AnyBoost, we here explicitly put boosting learning into the regularized empirical risk minimization framework and use convex optimization tools to analyze its characteristics. We will see as a result that the only difference between boosting and kernel learning is the optimization procedure. If all weak learners were known a-priori, there would be no essential difference between boosting and kernel methods. The most important aspect of the work presented here, however, is that we propose a general and fully corrective boosting learning framework that can be used to minimize regularized risk with arbitrary convex loss functions and arbitrary convex regularization terms in the form of problem (1).

The main contributions of this work are as follows.

1. We propose a general framework that can accommodate *arbitrary* convex regularization terms other than ℓ_1 norm. By explicitly deriving the Lagrange dual formulations, we demonstrate that fuuly-corrective boosting based on column generation can be designed to facilitate boosting. In particular, we focus on analyzing the ℓ_1, ℓ_2, and ℓ_∞ norm regularization.
2. We generalize the fully corrective ℓ_1 regularized boosting algorithms in [13] to *arbitrary* convex loss functions. We also show that a few variants of boosting algorithms in the literature can be interpreted in the proposed framework.
3. By introducing the concept of the nonnegatively-clipped edge of a weak classifier, we show the connection between ℓ_1, ℓ_2, and ℓ_∞ norm regularized boosting. The Lagrange dual formulations can then be interpreted as part of the presented unifying framework.
4. We show that the Fenchel conjugate of a convex loss function in the primal regularizes the dual variable (the training samples' importance weights). We thus generalize the results in [13], where only

the exponential loss, logistic loss and generalized hinge loss are considered. We thus show that the Fenchel conjugate of an arbitrary convex loss penalizes the divergence of the sample weights.

Moreover, we observe that fully-corrective boosting's primal problems are much simpler than their counterpart dual problems. So at each iteration of a column generation based boosting algorithm, it is much faster to solve the primal problem. In the proposed CGBoost, generally we do not require sophisticated convex solvers and only gradient descent methods like L-BFGS-B [19] are needed. Previous totally-corrective boosting algorithms [3, 13, 18] all solve the dual problems using convex optimization solvers. Besides the primal problems' much simpler structures, in most cases, the dual problems have many more variables than their corresponding primal problems.

The remainder of the paper is organized as follows. Before present the main results, we introduce the basic idea of boosting and Fenchel conjugate in Section 2. In Section 3, we extend the results in [13] to arbitrary convex loss functions. A new fully corrective boosting is proposed to minimize the ℓ_1 regularized risk. Connections of the proposed algorithm to some previous boosting algorithms are discussed. In Section 4, we generalize the fully corrective boosting to arbitrary convex regularization. We also briefly discuss boosting for regression. We present experimental results in Section 5 and conclude the paper in the last section.

2. Preliminaries

2.1. Notation

We introduce some notation that will be used before we proceed.

Let $\{(\boldsymbol{x}_i, y_i)\} \in \mathbb{R}^d \times \{-1, +1\}$, $i = 1 \cdots m$, be a set of m training examples. We denote $\mathcal{H}$ a set of weak classifiers; the size of $\mathcal{H}$ can be infinite. Each $h_j(\cdot) \in \mathcal{H}$, $j = 1 \cdots n$, is a function that maps $\boldsymbol{x}$ to $[-1, +1]$.[3] We denote H a matrix of size $m \times n$, where its (i, j) entry $H_{ij} = h_j(\boldsymbol{x}_i)$; that is H_{ij} is the label/confidence predicted by weak classifier $h_j(\cdot)$ on the training datum $\boldsymbol{x}_i$. So each column $H_{:j}$ of the matrix H consists of the output of weak classifier $h_j(\cdot)$ on the whole training data; while each row $H_{i:}$ contains the outputs of all weak classifiers on the training datum $\boldsymbol{x}_i$.

The edge of a weak classifier is defined as $d = \sum_{i=1}^m u_i y_i h(\boldsymbol{x}_i)$ where u_i is the weight associated with training example $\boldsymbol{x}_i$. The edge is an affine transformation of the weighted error for the case when $h(\cdot)$ takes discrete outputs $\{-1, +1\}$ and $\boldsymbol{u}$ is normalized (*e.g.*, in AdaBoost). The weighted error of $h(\cdot)$ is $\varpi = \frac{1}{2} - \frac{1}{2}d$. An edge can be used to measure the quality of a weak classifier: a larger d means a better $h(\cdot)$. The symbol $\mathrm{diag}(\boldsymbol{y}) \in \mathbb{R}^{m \times m}$ is a diagonal matrix with its (i, i) entry being label y_i. Column vectors are denoted by bold letters (*e.g.*, $\boldsymbol{x}, \boldsymbol{d}$). We denote by $A^\dagger$ the Moore-Penrose pseudo-inverse of the matrix A when A is not strictly positive definite.

2.2. Boosting

In statistics or signal processing, when we face ill-posed problems, regularization is needed to enforce stability of the solution. Regularization usually improves the conditioning of the problem. Literature on this subject is immense. In statistical learning, in particular, supervised learning, one learns a function that best describes the relation between input $\boldsymbol{x}$ and output y. Statistical learning theory tells us that often a regularization term is needed for a learning machine in order to trade off the training error and generalization capability [20].

Concretely, we solve the following problem for training a classifier or a regressor:

$$\inf_{F \in \mathcal{F}} \sum_{i=1}^m l(F(\boldsymbol{x}_i), y_i) + \vartheta \Omega(F(\cdot)). \tag{1}$$

Here $l(\cdot)$ is a data-fitting loss function, which corresponds to the empirical risk measure, and $\Omega(\cdot)$ is a regularization function. For example, $\Omega(\cdot)$ can be the Tikhonov regularization (ridge regression) for obtaining

[3]Later, we will discuss the general case that $h(\cdot)$ could be any real value.

a stable solution of ill-posed problems [21]. The parameter $\vartheta \geq 0$ balances these two terms. $\mathcal{F}$ is the functional space in which the classification function $F(\cdot)$ resides. Clearly, without the regularization term, if $\mathcal{F}$ is very large, it can easily lead to over-fitting and the minimizers can be nonsense. The regularized formulation considers the trade-off between the quality of the approximation over the training data and the complexity of the approximating function [20]. Often simplicity is manifested as sparsity in the solution vector—or some transformation of it. Typically, ℓ_p norm functions can be used for regularization such as ℓ_1 norm in Lasso [22], ℓ_2 norm in ridge regression, and RKHS regularization in kernel methods[4]. ℓ_1 norm regularization may create sparse answers and better approximations in relevant cases. ℓ_1 norm regularization methods have recently gained much attention in compressed sensing [23] and machine learning due to the induced sparsity and being easy-to-optimize as a surrogate of the non-convex ℓ_0 pseudo-norm [24, 25].

For boosting algorithms, $F(\cdot)$ takes the form

$$F(\boldsymbol{x}) = \sum_{i=1}^{n} w_i h_i(\boldsymbol{x}), \tag{2}$$

with $\boldsymbol{w} \succcurlyeq 0$. This nonnegativeness constraint can always be enforced because one can flip the sign of the weak classifier $h(\cdot)$.

It has been shown that some boosting algorithms can be viewed as ℓ_1 norm regularized model fitting [14]. We can rewrite the learning problem into

$$\inf_{F \in \mathcal{F}} \sum_{i=1}^{m} l(\gamma_i) + \vartheta \mathbf{1}^\top \boldsymbol{w} \tag{3}$$

where γ_i is the unnormalized margin: $\gamma_i = y_i F(\boldsymbol{x}_i) = y_i H_{i:} \boldsymbol{w}$. Our analysis relies on the concept of Fenchel duality.

Definition 2.1. *(Fenchel duality) Let $f : \mathbb{R}^n \to \mathbb{R}$. The function $f^* : \mathbb{R}^n \to \mathbb{R}$, defined as*

$$f^*(\boldsymbol{u}) = \sup_{\boldsymbol{x} \in \operatorname{dom} f} \boldsymbol{u}^\top \boldsymbol{x} - f(\boldsymbol{x}), \tag{4}$$

is called the Fenchel duality of the function $f(\cdot)$. The domain of the conjugate function consists of $\boldsymbol{u} \in \mathbb{R}^n$ for which the supremum is finite.

$f^*(\cdot)$ is always a convex function because it is the point-wise supremum of a family of affine functions of $\boldsymbol{u}$, even if $f(\cdot)$ is non-convex [26]. If $f(\cdot)$ is convex and closed, then $f^{**} = f$. For a point-wise loss function, $l(\boldsymbol{\gamma}) = \sum_{i=1}^{m} l(\gamma_i)$, the Fenchel duality of the sum is the sum of the Fenchel dualities:

$$\begin{aligned} l^*(\boldsymbol{u}) &= \sup_{\boldsymbol{\gamma}} \left\{ \boldsymbol{u}^\top \boldsymbol{\gamma} - \sum_{i=1}^{m} l(\gamma_i) \right\} = \sum_{i=1}^{m} \sup_{\gamma_i} \{ u_i \gamma_i - l(\gamma_i) \} \\ &= \sum_{i=1}^{m} l^*(u_i). \end{aligned}$$

Clearly, the shape of $f^*(u)$ is determined by $f(x)$ and *vice versa*. We consider functions of Legendre type [27] in this work. That means, the gradient $f'(\cdot)$ is defined on the domain of $f(\cdot)$ and is an isomorphism between the domains of $f(\cdot)$ and $f^*(\cdot)$. If $f(\cdot)$ admits a strict supporting line at x with slope u, then $f^*(\cdot)$ admits a tangent supporting line at u with slope $f^{*\prime}(u) = x$.

[4]Due to the representer theorem, RKHS regularization is special: typically, the optimal solution in an infinite-dimensional functional space can be found by solving a finite dimensional minimization problem.

Table 1: Loss functions and their derivatives.

Name	Loss $l(F,y)$	Derivative $l'(F,y)$
Exponential	$\exp(-yF)$	$-y\exp(-yF)$
Logistic	$\log(1+\exp(-yF))$	$-y/(1+\exp(yF))$
Hinge	$\max(0,-yF)$	0 if $yF \geq 0$; $-y$ otherwise
Squared hinge	$0.5[\max(0,-yF)]^2$	0 if $yF \geq 0$; F otherwise
MadaBoost loss [28]	$\exp(-yF)$ if $yF \geq 0$, otherwise $1-yF$	$-y\exp(-yF)$ if $yF \geq 0$; $-y$ otherwise
Least square	$0.5(y-F)^2$	$F-y$
ℓ_1 norm	$\lvert y-F\rvert$	$\mathrm{sgn}(F-y)$
Huber's loss	$0.5(y-F)^2$ if $\lvert y-F\rvert < 1$, otherwise $\lvert y-F\rvert - 0.5$	$F-y$ if $\lvert y-F\rvert < 1$; $\mathrm{sgn}(F-y)$ otherwise
Poisson regression	$\exp(F)-yF$	$\exp(F)-y$
Quantile regression	$\max(\tau(F-y),(1-\tau)(y-F))$	τ if $F>y$; $\tau-1$ otherwise
ε-insensitive regression	$\max(0,\lvert y-F\rvert-\varepsilon)$	0 if $\lvert y-F\rvert \leq \varepsilon$; $\mathrm{sgn}(F-y)$ otherwise

3. ℓ_1 Norm Regularized CGBoost

The general ℓ_1 regularized optimization problem we want to solve is

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{\gamma}} \; & \sum_{i=1}^{m} l(\gamma_i) + \vartheta \cdot \mathbf{1}^\top \boldsymbol{w} \\ \text{s.t.:} \; & \gamma_i = y_i H_{i:}\boldsymbol{w} \, (\forall i = 1 \cdots m), \; \boldsymbol{w} \succcurlyeq 0. \end{aligned} \tag{5}$$

We now derive its Lagrange dual problem. Although the variable of interest is $\boldsymbol{w}$, we keep the auxiliary variable $\boldsymbol{\gamma}$ in order to derive a meaningful dual. The Lagrangian is

$$\begin{aligned} L &= \sum_{i=1}^{m} l(\gamma_i) + \vartheta \mathbf{1}^\top \boldsymbol{w} - \boldsymbol{u}^\top(\boldsymbol{\gamma} - \mathrm{diag}(\boldsymbol{y})H\boldsymbol{w}) - \boldsymbol{p}^\top \boldsymbol{w} \\ &= \left(\vartheta \mathbf{1}^\top + \boldsymbol{u}^\top \mathrm{diag}(\boldsymbol{y})H - \boldsymbol{p}^\top\right)\boldsymbol{w} - \left(\boldsymbol{u}^\top\boldsymbol{\gamma} - \sum_{i=1}^{m} l(\gamma_i)\right), \end{aligned}$$

with $\boldsymbol{p} \succcurlyeq 0$. To find its infimum over the primal variables $\boldsymbol{w}$ and γ, we must have

$$\vartheta \mathbf{1}^\top + \boldsymbol{u}^\top \mathrm{diag}(\boldsymbol{y})H - \boldsymbol{p}^\top = \mathbf{0},$$

which leads to

$$\boldsymbol{u}^\top \mathrm{diag}(\boldsymbol{y})H \succcurlyeq -v\mathbf{1}^\top; \tag{6}$$

and

$$\inf_{\boldsymbol{w},\gamma} L = -\sup_{\gamma} \boldsymbol{u}^\top\boldsymbol{\gamma} - \sum_{i=1}^{m} l(\gamma_i) = -\sum_{i=1}^{n} l^*(u_i).$$

Therefore, the dual problem is

$$\min_{\boldsymbol{u}} \sum_{i=1}^{m} l^*(u_i), \quad \text{s.t.:} (6). \tag{7}$$

We can reverse the sign of $\boldsymbol{u}$ and rewrite (7) into its equivalent form

$$\min_{\boldsymbol{u}} \sum_{i=1}^{m} l^*(-u_i), \tag{8a}$$

$$\text{s.t.:} \quad \boldsymbol{u}^\top \operatorname{diag}(\boldsymbol{y}) H \preccurlyeq \vartheta \mathbf{1}^\top. \tag{8b}$$

From the Karush-Kuhn-Tucker (KKT) conditions, between the primal (5) and the dual (8) the relationship

$$u_i = -l'(\gamma_i), \forall i, \tag{9}$$

holds at optimality. This means, the weight u_i associated with each sample is the negative gradient of the loss at γ_i. This can be easily obtained by setting the first derivative of L w.r.t. γ_i to zeros. Under the assumption that both the primal and dual problems are feasible and the Slater's condition satisfies, strong duality holds between (5) and (8), which means that their solutions coincide such that one can obtain both solutions by solving either of them for many convex problems.

If we know all the weak classifiers, *i.e.*, the matrix H can be computed a priori, the original problem (5) can be easily solved (at least in theory) because it is an optimization problem with simple nonnegativeness constraints. In practice, however, we usually cannot compute all the weak classifiers since the size of the weak classifier set $\mathcal{H}$ could be prohibitively large or even infinite. In convex optimization, column generation (CG) is a technique that can be used to attack this difficulty. The crucial insight behind CG is: for a linear program, the number of non-zero variables of the optimal solution is equal to the number of constraints, hence although the number of possible variables may be large, we only need a small subset of these in the optimal solution. For a general convex problem, CG can still be used to obtain an approximate solution. It works by only considering a small subset of the entire variable set. Once it is solved, we ask the question "Are there any other variables that can be included to improve the solution?". So we must be able to solve the subproblem: given a set of dual values, one either identifies a variable that has a favorable reduced cost, or indicates that such a variable does not exist. In essence, CG finds the variables with negative reduced costs without explicitly enumerating all variables.

We now only consider a small subset of the variables in the primal; *i.e.*, only a subset of $\boldsymbol{w}$ is used. The problem solved using this subset is usually termed restricted master problem (RMP). Because the primal variables correspond to the dual constraints, solving RMP is equivalent to solving a relaxed version of the dual problem. With a finite $\boldsymbol{w}$, the set of constraints in the dual (8) are finite, and we can solve (8) that satisfies all the existing constraints. If we can prove that among all the constraints that we have not added to the dual problem, no single constraint is violated, then we can conclude that solving the restricted problem is equivalent to solving the original problem. Otherwise, there exists at least one constraint that is violated. The violated constraints correspond to variables in primal that are not in RMP. Adding these variables to RMP leads to a new RMP that needs to be re-optimized.

The general algorithm to solve our boosting optimization problem using CG—hence the name CGBoost—is summarized in Algorithm 5. A few comments on CGBoost are:

1. Practically, we set the stopping criterion as $\sum_{i=1}^m u_i y_i h'(\boldsymbol{x}_i) \leq \vartheta + \varepsilon$ where ε is a small user-specified constant;
2. The core part of CGBoost is the update of $\boldsymbol{u}$ (Line 5). Standard CG in convex optimization typically solves the dual problem. In our case, the dual problem (8) is a convex program that has m variables and n constraints. The primal problem (5) has n variables and n simple constraints (the equality constraints are only for deriving the dual and can be put back to the cost function). In boosting, often we have more training examples than final weak classifiers. That is, $m \geq n$. Moreover, n increases by one at each iteration. At the beginning, only small-scale problems are involved in the primal (5). As we will show, the simple constraints in (5) are also much easier to cope with. Quasic-Newton algorithms like L-BFGS-B [19] can be used to solve (5). In contrast, usually sophisticated primal-dual interior-point based algorithms are needed for solving the convex problem (8). All in all, (5) is easier to solve. Given $\boldsymbol{w}$, we can calculate $\boldsymbol{u}$ via the optimality condition (9). We do not need to know the Fenchel conjugate of the loss $l^*(\cdot)$ explicitly. We list some popular classification and regression loss functions and their first derivatives in Table 1.

Algorithm 5 terminates after a finite number of iterations at a global optimum. Theorem 3.1 guarantees the convergence of Algorithm 5. Generally, the CG method's convergence follows by standard CG algorithms in convex optimization. We include this theorem for self-completeness.

Algorithm 1 ℓ_1 norm regularized CGBoost for classification.

```
    Input: Training data $\{(\boldsymbol{x}_i, y_i)\}, i = 1 \cdots m$; a convergence threshold $\varepsilon > 0$.
1   Initialization: $\boldsymbol{w} = \boldsymbol{0}, \boldsymbol{u} = \frac{1}{m}\boldsymbol{1}$.
    while true do
2       – Receive a weak classifier that most violates the dual constraint:
              $\hat{h}(\cdot) = \operatorname*{argmax}_{h(\cdot)} \sum_{i=1}^{m} u_i y_i h(\boldsymbol{x}_i)$;
3       – Check for the stopping criterion:
        if $\sum_{i=1}^{m} u_i y_i \hat{h}(\boldsymbol{x}_i) \le \vartheta + \varepsilon$, then break;
4       – Add $\hat{h}(\cdot)$ into the primal problem that corresponds to a new variable;[a]
5       – Obtain $\boldsymbol{w}$ by solving the primal (5) and also using (9) to update the dual variable $\boldsymbol{u}$.
    Output: Output a convex combination of the weak classifiers.
```

[a]We can also add $\hat{h}(\cdot)$ to the dual problem as a new constraint. Then one solves the dual problem in the next step.

Theorem 3.1. *Assume that we can exactly solve the subproblem* $\hat{h}(\cdot) = \operatorname*{argmax}_{h(\cdot)} \sum_{i=1}^{m} u_i y_i h(\boldsymbol{x}_i)$ *at each iteration, then Algorithm 5 either halts on the round that the stopping criterion is met* $\sum_{i=1}^{m} u_i y_i \hat{h}(\boldsymbol{x}_i) \le \vartheta + \varepsilon$, *up to the desired accuracy* ε, *or converges to some finite value.*

The convergence follows the general column generation based technique in convex optimization, although the convergence rate is not known.

From the KKT condition (9) we can derive some interesting results.

Result 3.1. *At each iteration of LPBoost [3], a sample* $\boldsymbol{x}_i$ *that has a negative margin* γ_i (i.e., *miss-classified) will have a nonzero weight* u_i*; those samples that have positive margins (correctly classified) will all have zero weights, and they are not considered in the next iteration.*

We now show that AdaBoost and other boosting algorithms, for the appropriate choice of the regularization parameter ϑ, loss function and optimization strategy are just specific cases of CGBoost in Algorithm 5.

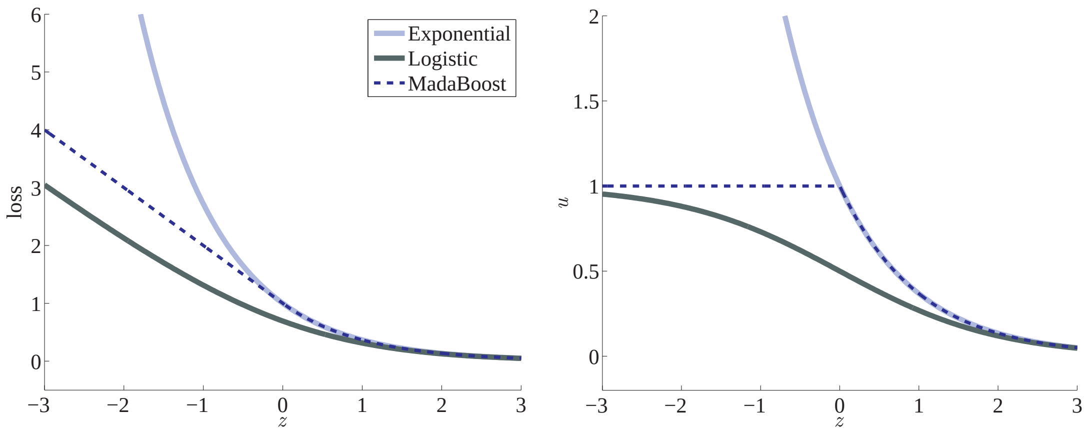

Figure 1: Some loss functions (first) and their first derivatives (second). Here the cost function of MadaBoost is defined as $l(z) = \exp(-z)$ if $z > 0$; $1 - z$ otherwise.

Theorem 3.2. *AdaBoost$_\rho$ [29] minimizes the regularized AdaBoost's cost function with the regularization parameter $\vartheta = \rho$ via coordinate-descent.*

AdaBoost is a special case of CGBoost with a very small regularization parameter ϑ (ϑ approaching zero), and a coordinate-descent optimization strategy to minimize the primal problem (Step (3) of CGBoost).

Proof: To prove that AdaBoost is indeed CGBoost with loss $l(y, F) = \exp(-yF)$, let us examine each step of CGBoost. Clearly, Step (1) of CGBoost is identical with AdaBoost. For the stopping criterion, when $\vartheta \to 0$, it is easy to verify that both algorithms stop at iteration $t + 1$ when

$$\varpi_+ - \varpi_- < 0;$$

with

$$\varpi_+ = \sum_{i: y_i h_{t+1}(\boldsymbol{x}_i) > 0} u_i^t,$$
$$\varpi_- = \sum_{i: y_i h_{t+1}(\boldsymbol{x}_i) < 0} u_i^t;$$

i.e.[5], the weighted error is larger than the weighted accuracy. Here we assume a discrete AdaBoost. It is straightforward to extend to real-valued $h(\cdot) \in [-1, +1]$ as discussed in [30].

If we adopt coordinate-descent to optimize the primal at Step (3), at iteration $t + 1$, we keep $w_1, \cdots, w_t$ fixed. Given the chosen $h_{t+1}(\cdot)$ we want to find w_{t+1} that minimizes

$$\begin{aligned} C_{\exp} &= \sum_{i=1}^m \exp(-\gamma_i^t) \cdot \exp(-y_i w_{t+1} h_{t+1}(\boldsymbol{x}_i)) + \vartheta \mathbf{1}^\top \boldsymbol{w} \\ &= \sum_{i=1}^m u_i^t \exp(-y_i w_{t+1} h_{t+1}(\boldsymbol{x}_i)) + \vartheta w_{t+1}, \\ &= \varpi_+ \exp(-w_{t+1}) + \varpi_- \exp(w_{t+1}) + \vartheta w_{t+1}, \end{aligned} \tag{10}$$

subject to $w_{t+1} > 0$. We have dropped the terms that are irrelevant to the variable w_{t+1}. Here we have used the fact from (9) that

$$u_i = \exp(-\gamma_i), \, \forall i, \tag{11}$$

To minimize $C_{\exp}$, set its first derivative to zero:

$$-\varpi_+ \exp(-w_{t+1}) + \varpi_- \exp(w_{t+1}) + \vartheta = 0. \tag{12}$$

A closed-form solution for w_{t+1} is:

$$w_{t+1} = \log\left(\sqrt{\frac{\varpi_+}{\varpi_-} + \frac{\vartheta^2}{4\varpi_-^2}} - \frac{\vartheta}{2\varpi_-}\right). \tag{13}$$

When ϑ is negligible, we have a solution for w_{t+1}:

$$w_{t+1} = \frac{1}{2} \log \frac{\varpi_+}{\varpi_-}, \tag{14}$$

which is consistent with AdaBoost. The rule for updating $\boldsymbol{u}$ can be trivially seen from (11).

Note that in the above analysis, we do not need to normalize $\boldsymbol{u}$. This is different from AdaBoost. Actually if we replace the entire loss $\sum_{i=1}^m \exp(-\gamma_i)$ with its logarithmic version $\log(\sum_{i=1}^m \exp(-\gamma_i))$; *i.e.*, we minimize $\log(\sum_{i=1}^m \exp(-\gamma_i)) + \vartheta \mathbf{1}^\top \boldsymbol{w}$, we have

$$u_i = \frac{\exp(-\gamma_i)}{\sum_{i=1}^m \exp(-\gamma_i)}, \forall i,$$

[5]Hereafter, subscript t indexes the iteration of CGBoost.

which results in $\mathbf{1}^\top \boldsymbol{u} = 1$. The cost (10) becomes $C_{\exp} = \log(\varpi_+ \exp(-w_{t+1}) + \varpi_- \exp(w_{t+1})) + \vartheta w_{t+1}$, where we have dropped $\log(\sum_{i=1}^m \exp(-\gamma_i^t))$ that is independent of w_{t+1}. It is easy to see that

$$w_{t+1} = \frac{1}{2}\left(\log\frac{\varpi_+}{\varpi_-} - \log\frac{1+\vartheta}{1-\vartheta}\right) \tag{15}$$

minimizes the new log-sum-exp cost function. This is the rule used in AdaBoost$_\rho$ [29]. Clearly when ϑ is small, (14) and (15) coincide. So, by simply replacing the update rule in AdaBoost with (15), we get an explicitly ℓ_1-norm regularized AdaBoost. □

We now know that AdaBoost is indeed a ℓ_1-norm regularized algorithm [14]. It is mysterious that AdaBoost does not have any parameter to tune and it works so well on many datasets. We have shown that AdaBoost simply sets the regularization parameter ϑ to a very small value. Note that one cannot set the regularization parameter ϑ to zero. Without this regularization term, the problem (1) is ill-posed. In the case of the AdaBoost and logistic boosting losses, on separable data, one can always make the first term of (1) approach zero by multiplying an arbitrarily large positive factor to $\boldsymbol{w}$.

We conjecture that a carefully-selected ϑ would yield better performance, especially on noisy datasets. This may partially explain why AdaBoost over-fits on noisy datasets. However, early stopping for AdaBoost eliminates over-fitting to some extent [31].

For the normalized version (log-sum-exp loss), we have a simpler closed-form update rule and no computation overhead is introduced compared with the standard AdaBoost. From (15), ϑ must be less than 1; hence $0 < \vartheta < 1$. As long as the selected weak classifier $h_{t+1}(\cdot)$ does not satisfy the stopping criterion, *i.e.*, $\varpi_+ - \varpi_- > \vartheta$, w_{t+1} calculated with (15) must be positive. Clearly a larger ϑ makes CGBoost converge faster.

Strategies such as *shrinkage* [10] and *bounded step-size* [31] have been proposed to preventing over-fitting. It is well known that these methods are other forms of regularization. In AdaBoost, shrinkage corresponds to replacing the w_{t+1} with $\eta' w_{t+1}$ where $0 < \eta' < 1$; while bounded step-size caps w_{t+1} by $\min\{w_{t+1}, \eta''\}$ where η'' is a small value. Both of these two methods decrease the step-size for producing better generalization performance. Starting from the general regularized statistical learning machine (1), we are able to show the regularized AdaBoost takes the form of (15) for updating the step-size. The fundamental idea is consistent with the two previous heuristics. Arc-Gv [32], proposed for producing larger minimum margins, modifies AdaBoost's updating rule as

$$w_{t+1} = \frac{1}{2}\left(\log\frac{\varpi_+}{\varpi_-} - \log\frac{1+\gamma_t'}{1-\gamma_t'}\right), \tag{16}$$

where γ_t' is the normalized minimum margin over all training samples of the combined classifier up to iteration t: $\gamma_t' = \min_i\{y_i \sum_{j=1}^t w_j h_j(\boldsymbol{x}_i)/\sum_{j=1}^t w_j\}$. Comparing (15) and (16), we have the following corollary.

Corollary 3.1. *Arc-Gv [32] is a regularized version of AdaBoost with an adaptive regularization parameter, which is the normalized minimum margin over all training examples.*

From the viewpoint of regularization theory, there is no particular reason that we should relate the regularization parameter ϑ to the minimum margin. Arc-Gv's purpose is to maximize the minimum margin to the extreme, which has been shown not beneficial for the final performance [33].

We can also design a fully-corrective AdaBoost easily according to the CGBoost framework. As described in Algorithm 5, we can either optimize the dual or primal. With the log-sum-exp loss, the dual problem of AdaBoost is

$$\min_{\boldsymbol{u}} \sum_{i=1}^m u_i \log u_i, \quad \text{s.t.:} \text{(8b) and } \mathbf{1}^\top \boldsymbol{u} = 1, \boldsymbol{u} \succcurlyeq \mathbf{0}, \tag{17}$$

which is an entropy maximization problem. This is a general constrained convex program. It can be solved using primal-dual interior point algorithms like [34]. Alternatively we can also solve it in the primal. We use L-BFGS-B [19] to solve the primal problem. L-BFGS-B is faster and more scalable. TotalBoost [18] takes the same form as (17) except that the parameter ϑ is adaptively set to the minimum edge over all weak classifiers generated up to the current iteration. It is clear now that TotalBoost is also a regularized AdaBoost:

Corollary 3.2. *AdaBoost$^*_\vartheta$ [29] and TotalBoost [18] minimize the regularized versions of AdaBoost's loss with an adaptive regularization parameter, which is the minimum edge over all weak classifiers (up to a numerical accuracy ϑ).*

AdaBoost$^*_\vartheta$ employs an coordinate descent optimization strategy while TotalBoost optimizes the cost function fully-correctively.

Again it remains unclear whether ϑ should be related to the minimum edge and how it is translated into the final generalization performance, although it is very clear that the minimum margin is efficiently maximized in AdaBoost$^*_\vartheta$ and TotalBoost.

TotalBoost [18] failed to discuss the primal-dual relationship of AdaBoost's cost function and (17), and an LPBoost is solved to obtain the final strong classifier after iteratively solving (17). In other words, in TotalBoost, the dual problem (17) is only used to generate weak classifiers.

As a conclusion of this section, we highlight that ℓ_1 norm regularized CGBoost in Algorithm 5 is consistent with AnyBoost of [11].

Theorem 3.3. *In Algorithm 5, if we set the regularization parameter $\vartheta = 0$, and solve the primal problem using coordinate descent (Line* 5 *of Algorithm 5),* i.e., *keep $w_1, \dots, w_j$ fixed at iteration $j+1$, then Algorithm 5 is the same as AnyBoost of Mason* et al. *[11]. Therefore, AnyBoost can be seen as a special of Algorithm 5.*

Proof: The proof shares similarities with the proof of Theorem 3.2. It is straightforward to establish this connection. □

Mason *et al.*'s AnyBoost is not immediately applicable to regression while our CGBoost can be used for regression without any modification. Also, for CGBoost, we may add multiple weak classifiers into the boosting optimization problem at each iteration to accelerate the convergence. However, AnyBoost can only include one weak learn at each iteration. Moreover, it is possible for CGBoost to work with some constraints. The optimization problem of AnyBoost is an unconstrained problem and cannot have additional constraints. We will discuss these issues in the following context.

4. A More General Formulation

Let us consider a more general formulation of the optimization problem (5):

$$\min_{\boldsymbol{w},\boldsymbol{\gamma}} \sum_{i=1}^m l(\gamma_i) \quad \text{s.t.:}\, Q\boldsymbol{w} \le \boldsymbol{r}, \gamma_i = y_i H_{i:}\boldsymbol{w}\, (\forall i = 1\dots m), \boldsymbol{w} \succcurlyeq 0, \tag{18}$$

where $Q \in \mathbb{R}^{p\times n}$ and $\boldsymbol{r} \in \mathbb{R}^p$ encode the regularization term and prior information when available. it is trivial to show (18) covers (5) as a special case. If we take $Q = \mathbf{1}^\top \in \mathbb{R}^{1\times n}$ and write the first constraint as $\mathbf{1}^\top \boldsymbol{w} \le r$. We know that for a certain ϑ, one can always find a r such that the solution of (5) also solves (18).

The Lagrange dual of (18) is

$$\min_{\boldsymbol{u},\boldsymbol{s}} \sum_{i=1}^m l^*(-u_i) + \boldsymbol{r}^\top \boldsymbol{s} \tag{19a}$$

$$\text{s.t.:}\, \boldsymbol{u}^\top \operatorname{diag}(\boldsymbol{y}) H \preccurlyeq \boldsymbol{s}^\top Q, \tag{19b}$$

$$\boldsymbol{s} \succcurlyeq \mathbf{0}. \tag{19c}$$

Here the dual variables are $\boldsymbol{u} \in \mathbb{R}^m$ and $\boldsymbol{s} \in \mathbb{R}^p$.

The optimality condition (9) also holds.

ℓ_∞ norm regularization is also a special case of the above formulation. ℓ_∞-norm regularization has been used in kernel classifiers [35]. If we let $Q = \mathbf{I} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{r} = r\mathbf{1}$, this is $\|\boldsymbol{w}\|_\infty \leq r$.

Let us have a close look at the ℓ_∞-norm regularized boosting. ℓ_∞-norm regularized boosting can be written as

$$\min_{\boldsymbol{w}, \boldsymbol{\gamma}} \sum_{i=1}^{m} l(\gamma_i) \tag{20a}$$

$$\text{s.t.}: \mathbf{0} \preccurlyeq \boldsymbol{w} \preccurlyeq r\mathbf{1}, \;\; \gamma_i = y_i H_{i:} \boldsymbol{w} \, (\forall i = 1 \ldots m). \tag{20b}$$

The Lagrangian is

$$\begin{aligned} L &= \sum_{i=1}^{m} l(\gamma_i) + \boldsymbol{s}^\top (\boldsymbol{w} - r\mathbf{1}) - \boldsymbol{q}^\top \boldsymbol{w} - \boldsymbol{u}^\top (\boldsymbol{\gamma} - \operatorname{diag}(\boldsymbol{y}) H \boldsymbol{w}) \\ &= \left(\boldsymbol{s}^\top + \boldsymbol{u}^\top \operatorname{diag}(\boldsymbol{y}) H - \boldsymbol{q}^\top \right) \boldsymbol{w} - \left(\boldsymbol{u}^\top \boldsymbol{\gamma} - \sum_{i=1}^{m} l(\gamma_i) \right) \\ &\quad - r\mathbf{1}^\top \boldsymbol{s}, \end{aligned}$$

with $\boldsymbol{q} \succcurlyeq 0$ and $\boldsymbol{s} \succcurlyeq 0$.

Therefore, its corresponding Lagrange dual is

$$\min_{\boldsymbol{u}, \boldsymbol{s}} \sum_{i=1}^{m} l^*(-u_i) + r\mathbf{1}^\top \boldsymbol{s} \tag{21a}$$

$$\text{s.t.}: \boldsymbol{u}^\top \operatorname{diag}(\boldsymbol{y}) H \preccurlyeq \boldsymbol{s}^\top, \tag{21b}$$

$$\boldsymbol{s} \succcurlyeq \mathbf{0}. \tag{21c}$$

Note that here we have reversed the sign of $\boldsymbol{u}$ too. Essentially the above dual problem can be converted into the following unconstrained problem

$$\min_{\boldsymbol{u}} \sum_{i=1}^{m} l^*(-u_i) + r \sum_{j=1}^{n} \left[\sum_{i=1}^{m} u_i y_i H_{ij} \right]_+, \tag{22}$$

where $[z]_+ = \max(0, z)$ is the hinge loss. That means, if the edge of a weak classifier is non-positive, it does not have any impact on the optimization problem. The first term can be seen as a regularization term that makes the sample weight $\boldsymbol{u}$ uniform and the second term encourages the edge of a weak classifier to become non-positive. Let us define a symbol, the nonnegatively-clipped edge of weak classifier j,

$$d_j^+ = \left[\sum_{i=1}^{m} u_i y_i H_{ij} \right]_+ \tag{23}$$

for convenience. As we will see, with the notation of this nonnegatively-clipped edge, we are able to unify the Lagrange dual formulations of ℓ_p ($p = 1, 2, \infty$) regularized boosting. So (22) is

$$\min_{\boldsymbol{u}} \sum_{i=1}^{m} l^*(-u_i) + r \left\| \boldsymbol{d}^+ \right\|_1, \tag{24}$$

with $\boldsymbol{d}^+ = [d_1^+, \cdots, d_j^+, \cdots, d_n^+]^\top$. To establish the connection with the ℓ_1 regularized boosting, we have the following result:

Proposition 4.1. *The Lagrange dual of ℓ_1-norm regularized boosting can be equivalently written as the following unconstrained optimization*

$$\min_{\boldsymbol{u}} \sum_{i=1}^{m} l^*(-u_i) + r \left\| \boldsymbol{d}^+ \right\|_\infty . \tag{25}$$

Proof: Let us start from rewriting the primal problem (5) of the ℓ_1 norm regularized boosting. Clearly we can also write the regularization as an explicit constraint

$$\min_{\boldsymbol{w},\boldsymbol{\gamma}} \sum_{i=1}^{m} l(\gamma_i), \text{ s.t.:} \|\boldsymbol{w}\|_1 \leq r, \boldsymbol{w} \succcurlyeq \boldsymbol{0}, \gamma_i = y_i H_{i:}\boldsymbol{w}, \forall i. \tag{26}$$

Given the regularization constant ϑ in (5), one can always find a r such that (5) and (26) have the same solution. It is easy to see that the optimal $\boldsymbol{w}^\star$ is always located at the boundary of the feasibility set: $\|\boldsymbol{w}^\star\|_1 = r$ [13]. We use the inequality constraint here. The Lagrange dual of (26) is

$$\min_{\boldsymbol{u},s} \sum_{i=1}^{m} l^*(-u_i) + rs, \text{ s.t.: } \boldsymbol{u}^\top \operatorname{diag}(\boldsymbol{y}) H \preccurlyeq s\boldsymbol{1}^\top, s \geq 0. \tag{27}$$

Here the dual variables are $\boldsymbol{u}$ and $s \geq 0$. From the first constraint, it is clear that

$$s = \max_{j=1\dots n} \left\{ \textstyle\sum_{i=1}^m u_i y_i H_{ij} \right\} = \max_{j=1\dots n} \{d_j\},$$

if the largest edge $\max_j \{d_j\} \geq 0$. Here $d_j = \sum_{i=1}^m u_i y_i H_{ij}$ is the edge of weak classifier j. Otherwise $s = 0$ must hold because of the constraint $s \geq 0$. Hence, using the concept of clipped edges (23), we have

$$s = \max_j \left\{ d_j^+ \right\} = \|\boldsymbol{d}^+\|_\infty .$$

Now we can eliminate s and rewrite the above problem into (25). □

Comparing (24) and (25), the only difference is the norm employed in the second term. This is not a surprising result if one is aware of ℓ_1 norm and ℓ_∞ norm being dual to each other. We also know that the concept of *margin* that is associated with a sample and the concept of *edge* associated with a weak classifier are dual to each other in boosting.

From the KKT conditions for the ℓ_∞-norm regularized boosting, we have the following equalities at optimality:

$$u_i = -l'(\gamma_i), \forall i = 1 \cdots m, \tag{28}$$

which is the same as (9). From the complementary conditions, we also have

$$\boldsymbol{s}^\top (\boldsymbol{w} - r\boldsymbol{1}) = 0;$$
$$\boldsymbol{q}^\top \boldsymbol{w} = 0.$$

Therefore, if $w_j \neq r$, then $s_j = 0$ must hold. If $w_j = r$, then $q_j = 0$; hence $s_j - \sum_{i=1}^m u_i y_i H_{ij} = q_j = 0$. In summary, we can obtain both the dual variables $\boldsymbol{u}$ and $\boldsymbol{s}$ from the primal variables $\boldsymbol{w}$ using the following relationships

$$s_j = \begin{cases} 0 & \text{if } w_j < r, \\ \sum_{i=1}^m u_i y_i H_{ij} & \text{if } w_j = r; \end{cases} \tag{29}$$

and (28).

Although the Lagrange dual problems of ℓ_1-norm and ℓ_∞-norm regularized boosting can both be rewritten into unconstrained problems, Neither of them is not differentiable, hence still difficult to solve. That is why we solve the primal problems (5) and (20) instead (as long as the loss function $l(\cdot)$ is convex and differentiable), and using the optimality conditions to recover the dual variables from the primal variables.

However, we know that the ℓ_2 norm is dual to itself and it is differentiable, unlike the ℓ_1 and ℓ_∞ norms. There is hope that the Lagrange dual problem of ℓ_2 norm regularized boosting can be written as an unconstrained differentiable problem so that it is much easier to solve. We discuss this case in the next section.

4.1. Arbitrary Regularization

Let us consider the following very general case. Now we not only assume a general loss function, we also assume that the regularization term is any popular regularization function. Concretely, we have the following form,

$$\min_{\boldsymbol{w}} \sum_{i=1}^{m} l(y_i H_{i:}\boldsymbol{w}) + \vartheta \cdot \Omega(\boldsymbol{w}) \text{ s.t.: } \boldsymbol{w} \succcurlyeq \mathbf{0}. \tag{30}$$

We rewrite (30) into the following equivalent form by introducing another auxiliary variable η:

$$\begin{aligned} &\min_{\boldsymbol{w},\boldsymbol{\gamma},\boldsymbol{\eta}} \sum_{i=1}^{m} l(\gamma_i) + \vartheta \cdot \Omega(\boldsymbol{\eta}) \\ &\text{s.t.: } \gamma_i = y_i H_{i:}\boldsymbol{w}, \forall i = 1 \dots n, \; \boldsymbol{\eta} = \boldsymbol{w}, \; \boldsymbol{w} \succcurlyeq \mathbf{0}. \end{aligned} \tag{31}$$

The Lagrangian is

$$\begin{aligned} L = &\sum_{i=1}^{m} l(\gamma_i) + \vartheta\Omega(\boldsymbol{\eta}) - \boldsymbol{u}^\top(\gamma - \operatorname{diag}(\boldsymbol{y})H\boldsymbol{w}) \\ &- \boldsymbol{s}^\top(\vartheta\boldsymbol{\eta} - \vartheta\boldsymbol{w}) - \boldsymbol{p}^\top\boldsymbol{w}, \end{aligned} \tag{32}$$

with $\boldsymbol{p} \succcurlyeq \mathbf{0}$.

The Lagrange dual is

$$\begin{aligned} &\max_{\boldsymbol{u},\boldsymbol{s}} \; -\sum_{i=1}^{m} l^*(u_i) - \vartheta \cdot \Omega^*(\boldsymbol{s}) \\ &\text{s.t.: } \vartheta\boldsymbol{s}^\top + \boldsymbol{u}^\top \operatorname{diag}(\boldsymbol{y})H \succcurlyeq \mathbf{0}. \end{aligned} \tag{33}$$

It is important to introduce the auxiliary variable $\boldsymbol{\eta}$; otherwise we are not able to arrive at this meaningful dual formulation. As before, we reverse the sign of $\boldsymbol{u}$, and we obtain

$$\begin{aligned} &\min_{\boldsymbol{u},\boldsymbol{s}} \sum_{i=1}^{n} l^*(-u_i) + \vartheta \cdot \Omega^*(\boldsymbol{s}) \\ &\text{s.t.: } \boldsymbol{u}^\top \operatorname{diag}(\boldsymbol{y})H \preccurlyeq \vartheta\boldsymbol{s}^\top. \end{aligned} \tag{34}$$

Next we discuss a special case, namely, ℓ_2 regularization. In the case of ℓ_2 regularization, we set $\Omega(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$, and the Fenchel conjugate $\Omega^*(\boldsymbol{s}) = \frac{1}{2}\|\boldsymbol{s}\|_2^2$. So the primal problem is

$$\min_{\boldsymbol{w}} \sum_{i=1}^{m} l(\gamma_i) + \tfrac{1}{2}\vartheta\|\boldsymbol{w}\|_2^2, \text{ s.t.: } \gamma_i = y_i H_{i:}\boldsymbol{w}, \; \boldsymbol{w} \succcurlyeq \mathbf{0}. \tag{35}$$

The Lagrange dual can be written into an unconstrained problem again:

$$\min_{\boldsymbol{u}} \sum_{i=1}^{m} l^*(-u_i) + r\left\|\boldsymbol{d}^+\right\|_2^2, \tag{36}$$

with $r = 0.5/\vartheta$.

Table 2: The primal and dual problems of ℓ_p ($p = 1, 2, \infty$) norm regularized boosting algorithms. Samples' margins $\boldsymbol{\gamma}$ and weak classifiers' clipped edges $\boldsymbol{d}^+$ are dual to each other. ℓ_p regularization in primal corresponds to ℓ_q regularization in dual with $1/p + 1/q = 1$. Note that $\boldsymbol{\gamma}$ is a function of $\boldsymbol{w}$ and $\boldsymbol{d}^+$ is a function of $\boldsymbol{u}$.

	Primal	Dual
ℓ_1	$\min \sum_{i=1}^m l(\gamma_i) + \vartheta \, \|\boldsymbol{w}\|_1$	$\min \sum_{i=1}^m l^*(-u_i) + r \|\boldsymbol{d}^+\|_\infty$
ℓ_2	$\min \sum_{i=1}^m l(\gamma_i) + \vartheta \, \|\boldsymbol{w}\|_2^2$	$\min \sum_{i=1}^m l^*(-u_i) + r \|\boldsymbol{d}^+\|_2^2$
ℓ_∞	$\min \sum_{i=1}^m l(\gamma_i) + \vartheta \, \|\boldsymbol{w}\|_\infty$	$\min \sum_{i=1}^m l^*(-u_i) + r \|\boldsymbol{d}^+\|_1$
	$l(\boldsymbol{\gamma})$: loss in primal	$\|\boldsymbol{d}^+\|_q$: loss in dual
	$\|\boldsymbol{w}\|_p$: regularization in primal	$l^*(\boldsymbol{u})$: regularization in dual

Result 4.1. *The cost function in* (36) *is differentiable everywhere. Hence gradient descent methods like L-BFGS can be used.*

This result follows the fact that the squared hinge loss is differentiable. So we have answered the conjecture in the last section: we indeed obtain a convex, differentiable unconstrained dual problem for the ℓ_2 norm regularized boosting problem. Note that the Fenchel conjugate $l^*(\cdot)$ may have extra constraints on its variable $\boldsymbol{u}$. For example, in the case of exponential loss, $\boldsymbol{u}$ has nonnegative-ness constraints, which still can be solved by L-BFGS-B. In practice, it is still better to solve the primal problem because the size of the primal problem is usually smaller than the size of the dual problem ($n < m$).

Table 2 summarizes our results. Note that it may be possible to extend the analysis to the case of a more general ℓ_p, in general it is not of interest for $p \notin \{1, 2, \infty\}$ in the machine learning community. Moreover, when $p \notin \{1, 2, \infty\}$ the optimization problem becomes much more difficult.

The RKHS regularization term is $\Omega(\boldsymbol{w}) = \boldsymbol{w}^\top K \boldsymbol{w}$ where K is the kernel matrix, usually being strictly positive definite. We can easily show its Lagrange dual by noticing the dual of $\boldsymbol{w}^\top K \boldsymbol{w}$ being $\boldsymbol{w}^\top K^{-1} \boldsymbol{w}$.

4.2. How the Fenchel Conjugate of the Primal Loss Regularizes the Dual Variable

Firstly, we have the following theorem.

Theorem 4.1. *The dual variable $\boldsymbol{u}$ in classification is always a probability distribution up to a normalization factor. This normalization factor has no influence on the CGBoost algorithms.*

Proof: In classification, the relationship between the margin in the primal and the sample weight in the dual is $u_i = -l'(\gamma_i), \forall i$. This equation holds for all the three ℓ_1, ℓ_2 and ℓ_∞ regularization cases.

We know that typically the classification loss function is convex and monotonically decreasing. Therefore, $u_i = -l'(\gamma_i)$ *must be nonnegative*. This is guaranteed as long as we infer $\boldsymbol{u}$ from the primal. So we do not need explicit constraints to make the sample weights nonnegative if working in the primal.

For some loss functions, *e.g.*, AdaBoost's log-sum-exp function, $\boldsymbol{u}$ is normalized $\|\boldsymbol{u}\|_1 = 1$. $\boldsymbol{u}$ is a probability distribution on the samples. However, the normalization of $\boldsymbol{u}$ does not have any impact on the algorithm in our framework. It does not affect the selection of weak classifiers or the update of $\boldsymbol{u}$ in the iteration. □

Let us have a close look at the loss function $l(\gamma)$ and its role in the Lagrange dual. From Table 2, we know that $l^*(-u)$ works as a regularization term. Shen and Li discussed the cases when $l(\cdot)$ is exponential loss, logistic loss and generalized hinge loss, $l^*(\cdot)$ is Shannon entropy, binary entropy and Tsallis entropy, respectively [13]. All of them make the dual variable $\boldsymbol{u}$ *uniform*. In LPBoost that employs the non-differentiable hinge loss, the Fenchel conjugate is an indicator function that caps the dual variable $\boldsymbol{u}$ so that $\boldsymbol{u}$ is confined in a box. $\boldsymbol{u}$ is uniformed in a hard way. The hinge loss is an exception in that it is *not strongly convex* and non-differentiable. It is important because we can view hinge loss as the extreme of many loss functions, *e.g.*, the logistic loss. Theorem 4.2 generalizes the theoretical results of [13].

Theorem 4.2. *Let us assume that $l(\gamma)$ is strictly convex and differentiable everywhere in $(-\infty, +\infty)$. The Fenchel conjugate $\sum_{i=1}^m l^*(-u_i)$ penalizes the divergence of $\boldsymbol{u}$;* i.e., *$\sum_i l^*(-u_i)$ encourages $\boldsymbol{u}$ become uniform.*

Proof: Because $l(\gamma)$ is strictly convex and differentiable everywhere, we know that the first derivative $l'(\gamma)$ is continuous and monotonically increasing for increasing γ. In this case, the Fenchel duality has the following analytic expression:

$$l^{*\prime}(-u_i) = \gamma_i, \;\forall i. \tag{37}$$

Here $l^{*\prime}(\cdot)$ is the first derivative of $l^*(\cdot)$. Because Theorem 4.1 holds for all the three cases considered, $\boldsymbol{u}$ take values in $(0, \kappa)$ with $\kappa > 0$ (κ could be $+\infty$). So $l^*(-u_i)$ is defined in the domain of $(-\kappa, 0)$. Equations $u_i = -l'(\gamma_i)$ and (37) hold at the same time for a pair of $\{u_i, \gamma_i\}$, $\forall i$. When $\gamma \to +\infty$, $u \to 0$ from left side. With (37),

$$l^{*\prime}(u \to 0) \to +\infty. \tag{38}$$

When $\gamma \to -\infty$, $u \to -\kappa$, so

$$l^{*\prime}(u \to -\kappa) \to -\infty. \tag{39}$$

Therefore, $l^{*\prime}(-u^\diamond) = 0$ for a certain $0 < -u^\diamond < \kappa$. In other words, $l^*(-u)$ must be "$\cup$-shaped" in $0 < -u < \kappa$.

$l^*(-u)$ is convex and has a unique minimum at $0 < -u^\diamond < \kappa$. Clearly, $\sum_{i=1}^m l^*(-u_i)$ penalizes those u_i's that deviate from $u^\diamond$. □

In the above theorem, we have assumed that $l(\cdot)$ has no non-differentiable points, it should not be difficult to extend it to the case that $l(\cdot)$ has non-differentiable points using the definition of Fenchel conjugate.

It has been shown in [13] that minimizing the exponential loss function results in minimizing the divergence of margins too. The authors theoretically proved that AdaBoost (also its fully corrective version) approximately maximizes the unnormalized average margin and at the same time minimizes the variance of the margin distribution under the assumption that the margin follows a Gaussian distribution. They have proved this result by analyzing the *primal* optimization problem. Now with Theorem 4.2, we can show this result from the *dual* problem. With $u = -l'(\gamma)$, minimizing the divergence of $\boldsymbol{u}$ also minimizes the divergence of $l'(\boldsymbol{\gamma})$. But generally it cannot be translated into minimizing the divergence of $\boldsymbol{\gamma}$ *unless* $l'(\cdot)$ *is strictly monotonic*. When the exponential loss is used, $u = -l'(\gamma) = \exp(-\gamma)$, in which $l'(\cdot)$ is indeed strictly monotonic. See Fig. 1 for a demonstration of this relationship. For the logistic loss, this conclusion also holds. For those loss functions whose first derivatives are truncated from above, *e.g.*, the MadaBoost loss, this conclusion applies only approximately. For instance, in the case of the MadaBoost loss, as long as the margin γ_i is positive, the corresponding u_i equals a constant.

As mentioned before, the hinge loss is an extreme case. The regularization term $\sum_i l^*(-u_i)$ in the dual of LPBoost is a *hard* indicator function. Essentially they are a set of box constraints on $\boldsymbol{u}$; The first derivative of the hinge loss is a non-continuous step function. Unlike the exponential or logistic loss, to minimize the divergence of $\boldsymbol{u}$ does not effectively minimize the divergence of the margins. If a small margin divergence does contribute to a better generalization capability, the hinge loss would not be an ideal choice for boosting.

Note that the optimization strategy of boosting is entirely different from support vector machines (SVMs). In SVMs, the hypotheses/dictionary for building the final classifier is fixed. In boosting, the weak hypotheses could be infinitely large.

4.3. Confidence-rated Predictions

The derivations of the last sections do not depend on the condition that the output of the weak classifier (the matrix H) must be discrete $\{-1, +1\}$. Therefore, as in the case of LPBoost [3], the proposed methods can use a weak learner belonging to a finite set of confidence-related functions. The outputs of the weak learner can be any real values.

Indeed, it is not difficult to show that the same framework can be applied to learn a mixture of kernels as in [36]. It is also possible to include an offset in the learned strong classifier, in which case the optimization problem is only slightly different. However, this topic is beyond the scope of this paper.

4.4. CGBoost for Regression

In this section, we extend the presented framework to regression problems. The presented framework can be easily extended to regression. The difference is the concept of margin $\gamma = yF(\boldsymbol{x})$. In classification, one tries to push this margin as large as possible. In regression, instead one tries to minimize the distance of predicted response $F(\boldsymbol{x})$ and the given response y, *i.e.*, min $l(F(\boldsymbol{x}) - y)$. Here $l(\cdot)$ is usually a convex loss function that penalizes the deviation between $F(\boldsymbol{x})$ and y. Let us consider the arbitrary regularization. We rewrite the problem in (31):

$$\begin{aligned}&\min_{\boldsymbol{w},\boldsymbol{\gamma},\boldsymbol{\eta}} \sum_{i=1}^{m} l(\gamma_i) + \vartheta \cdot \Omega(\boldsymbol{\eta}) \\ &\text{s.t.: } \gamma_i = y_i - H_{i:}\boldsymbol{w}\,(\forall i = 1 \ldots n), \boldsymbol{\eta} = \boldsymbol{w},\, \boldsymbol{w} \succcurlyeq \mathbf{0}.\end{aligned} \tag{40}$$

Compared with (31), the only difference is the first constraint, *i.e.*, the definition of margins. Using the same technique, we arrive at the corresponding Lagrange dual:

$$\min_{\boldsymbol{u},\boldsymbol{s}} \sum_{i=1}^{n} l^*(u_i) + \vartheta\Omega^*(\boldsymbol{s}) - \boldsymbol{y}^\top \boldsymbol{u}, \text{ s.t.: } \boldsymbol{u}^\top H \preccurlyeq \vartheta \boldsymbol{s}^\top. \tag{41}$$

The KKT condition is

$$u_i = l'(\gamma_i),\ \forall i = 1 \ldots m. \tag{42}$$

We list a few special cases in the Appendix.

The dual Lagrange multiplier $\boldsymbol{u}$ here could be negative from (42) because the regression loss function must not be monotonic. The derivative of the loss function could be negative or positive. Therefore, in regression, the dual variable $\boldsymbol{u}$ cannot be viewed as a sample weight that measures the importance of the training sample. Generally, the regression loss function $l(\cdot)$ is symmetric, hence the absolute value of $|\boldsymbol{u}|$ might be seen as the weight associated with the training samples. This is different from the classification case as we have shown in the last section.

5. Experiments

In this section, we run some experiments to verify the performance and efficiency of CGBoost with various loss functions and regularization terms. More specifically, exponential loss ($l(y, F) = \exp(-yF)$) and hinge loss ($\max(0, -yF)$) are used as loss function candidates, while the regularization term is either ℓ_1, ℓ_2, or ℓ_∞ norm. Consequently, there are 6 combinations of the loss function plus the regularization term. In order to control the complexity of weak classifiers, we have used decision stumps. CGBoost based on all the loss-regularizer pairs are implemented. AdaBoost is also compared as the baseline. A 5-fold cross-validation procedure is used for each CGBoost to tune the regularization parameter. For fair comparison, the stopping iteration for AdaBoost is also cross-validated.

The first experiment is carried out on the 13 UCI datasets obtained from [37, 38]. Each dataset is randomly split into two groups. 60% of data samples are used for training and validation and the remaining ones are used for test[6]. L-BFGS-B is used to solve the primal optimization problem with the exponential loss, while Mosek [34] is used to solve the ones with hinge loss, which are not differentiable. The convergence threshold ε is 10^{-5} for the exponential loss case and 10^{-3} for the ones with hinge loss because Mosek is much slower than L-BFGS-B. All the experiments are repeated 40 times and both the mean and standard deviations are reported. Both test and training errors of boosting algorithms are reported in Table 8. In the case that an algorithm converges earlier than any pre-selected iterations, we simply copy the converged results to this iteration and the latter ones.

[6]For the datasets ringnorm, twonorm and waveform, only 10% of data samples are selected for training and validation due to the extreme large amount of samples.

Table 3: Training and test errors of linear SVM over 40 runs, with the regularization parameter C being selected using 5-fold cross validation from $\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}\}$. The errors and standard deviations are shown in percentage (%).

Error	banana	b-cancer	diabetis	f-solar	german	heart	image	ringnorm	splice	thyroid	titanic	twonorm	waveform
Train	9.3±0.4	17.6±4.7	16.2±2.1	21.9±5.3	13.4±3.4	7.9±3.0	0.9±0.5	0.8±0.6	0.2±0.5	1.0±1.3	15.9±7.0	1.3±0.6	4.0±2.5
Test	9.6±0.6	26.8±3.5	24.3±2.1	36.9±5.3	25.0±2.1	17.1±2.9	3.5±0.6	1.7±0.2	9.6±0.8	4.8±1.9	21.8±11.7	2.4±0.2	10.4±1.0

Table 4: Results of the Wilcoxon Signed-Ranks Test (WSRT) [39]. The test is processed pairwisely among the compared boosting methods. The block where "Better" takes place indicates that the algorithm corresponding to its row is better than the algorithm corresponding to its column. "No" suggests that the row algorithm is not better than the column algorithm. The inequality in the parenthesis is the comparison between the Wilcoxon statistic (l.h.s.) and the critical value (r.h.s.). The critical value depends on the number of data sets yielding different performances. Hence it is not fixed. The hypothesis is rejected When the statistic is larger than the critical value.

	AdaBoost	**CG exp, ℓ_1**	**CG exp, ℓ_2**	**CG exp, ℓ_∞**	**CG hinge, ℓ_1**	**CG hinge, ℓ_2**	**CG hinge, ℓ_∞**
AdaBoost	–	No ($27 < 61$)	No ($51 < 53$)	No ($19 < 70$)	No ($21 < 70$)	No ($17 < 70$)	No ($23 < 70$)
CG exp, ℓ_1	No ($51 < 61$)	–	No ($62 > 61$)	No ($37 < 70$)	No ($47 < 70$)	No ($43 < 70$)	No ($45 < 70$)
CG exp, ℓ_2	No ($15 < 53$)	No ($16 < 61$)	–	No ($15 < 70$)	No ($20 < 70$)	No ($16 < 70$)	No ($45 < 70$)
CG exp, ℓ_∞	**Better** ($72 > 70$)	No ($54 < 70$)	**Better** ($76 > 70$)	–	No ($59 < 70$)	No ($62 < 70$)	No ($67 < 70$)
CG hinge, ℓ_1	No ($70 \leq 70$)	No ($44 < 70$)	No ($71 > 70$)	No ($32 < 70$)	–	No ($16 < 45$)	No ($47 < 61$)
CG hinge, ℓ_2	**Better** ($74 > 70$)	No ($48 < 70$)	**Better** ($75 > 70$)	No ($29 < 70$)	No ($39 < 45$)	–	No ($56 > 61$)
CG hinge, ℓ_∞	No ($68 < 70$)	No ($46 < 70$)	No ($62 < 70$)	No ($24 < 70$)	No ($31 < 61$)	No ($22 < 61$)	–

As demonstrated in the table, the first observation is that all the methods perform similarly. In terms of the *test error*, standard AdaBoost only wins once, which is almost the worst performance among all the methods. Considering that cross-validation is also performed for AdaBoost, we may draw the conclusion that in general, CGBoost could be slightly better if the regularization is carefully selected.

Here, we also run the linear SVM on the same datasets for comparison. The training and test results are reported in Table 3. Compared with the results of boosting in Table 8, as we can see, boosting and SVM have achieved comparable overall classification accuracy.

In order to verify the performance of CGBoost statistically, we implement the Wilcoxon signed-rank test on the experimental results. The Wilcoxon signed-ranks test (WSRT) [39] is a non-parametric alternative of the paired t-test, which can rank the difference in performance of two classifiers for each data set. In this paper, the WSRT test is used for pairwise comparison of all the boosting algorithms in terms of empirical test error. The null-hypothesis declares that an algorithm is not better than the comparisons in terms of performance. Thus, it is a one-tail test. We set conventional confidence level to be 95% and the rejection region is $\{w \in \mathbb{R} \mid w > 70\}$, considering the number of datasets with different performance is 13.[7] The output of WSRT is illustrated in Table 4.

We can see that statistically (at a confidence level of 95%), 1) CGBoost with exponential loss and ℓ_∞, and CGBoost with hinge and ℓ_2 are *marginally* superior to AdaBoost and CGBoost with exponential and ℓ_2; This hypothesis test also suggests that AdaBoost is never significantly better than any form of CGBoost.

We have also performed another statistical test, namely, the Bonferroni-Dunn test [39]. The result shows that no algorithm statistically outperforms the other one. The comparison results are in Table 5. Note that the Bonferroni-Dunn test is a post-hoc manner to verify whether a classifier over-performs the others under the circumstance of multiple comparison [39]. It is not a pair-wise comparison. In general, on these standard benchmark datasets, we would expect most methods perform similarly. We have also plotted boxplots of different methods on a few datasets. Boxplots present an easy-to-interpret graphical representation of the experiment results [40]. Figure 2 shows the plots on a few datasets. We see that there is no significant difference between different methods.

The computational complexity of AdaBoost is trivial due to the closed-form solution at each iteration. In contrast, many fully-corrective boosting methods [3, 13, 18, 41] are computationally demanding because

[7]Here the critical value is not fixed since it depends on the number of datasets with different performance of two algorithms.

Table 5: Results of the Bonferroni-Dunn Test (BDT) [39]. Each algorithms is compared with other 6 boosting methods. The block where "Better" takes place indicates that the algorithm corresponding to its row is better than the algorithm corresponding to its column. "No" means the row algorithm cannot be considered better than the column algorithm. The inequality in the parenthesis is the comparison between the Bonferroni statistic (l.h.s.) and the critical value (r.h.s.). The critical value depends on the number of comparing classifiers, thus it is fixed (here it is 2.693). The hypothesis is rejected when the statistic is larger than the critical value.

	AdaBoost	**CG exp, ℓ_1**	**CG exp, ℓ_2**	**CG exp, ℓ_∞**	**CG hinge, ℓ_1**	**CG hinge, ℓ_2**	**CG hinge, ℓ_∞**
AdaBoost	–	No ($-1.725 < 2.693$)	No ($-0.317 < 2.693$)	No ($-2.496 < 2.693$)	No ($-1.225 < 2.693$)	No ($-1.861 < 2.693$)	No ($-1.044 < 2.693$)
CG exp, ℓ_1	No ($1.725 < 2.693$)	–	No ($1.407 < 2.693$)	No ($-0.772 < 2.693$)	No ($0.499 < 2.693$)	No ($-0.136 < 2.693$)	No ($0.680 < 2.693$)
CG exp, ℓ_2	No ($0.317 < 2.693$)	No ($-1.407 < 2.693$)	–	No ($-2.178 < 2.693$)	No ($-0.908 < 2.693$)	No ($-1.542 < 2.693$)	No ($-0.726 < 2.693$)
CG exp, ℓ_∞	No ($2.496 < 2.693$)	No ($0.771 < 2.693$)	No ($2.178 < 2.693$)	–	No ($1.271 < 2.693$)	No ($0.635 < 2.693$)	No ($1.452 < 2.693$)
CG hinge, ℓ_1	No ($1.225 < 2.693$)	No ($-0.499 < 2.693$)	No ($0.908 < 2.693$)	No ($-1.271 < 2.693$)	–	No ($-0.635 < 2.693$)	No ($0.181 < 2.693$)
CG hinge, ℓ_2	No ($1.861 < 2.693$)	No ($0.136 < 2.693$)	No ($1.542 < 2.693$)	No ($-0.635 < 2.693$)	No ($0.635 < 2.693$)	–	No ($0.817 < 2.693$)
CG hinge, ℓ_∞	No ($1.044 < 2.693$)	No ($-0.680 < 2.693$)	No ($0.726 < 2.693$)	No ($-1.452 < 2.693$)	No ($-0.181 < 2.693$)	No ($0.817 < 2.693$)	–

Table 6: Test and training errors of AdaBoost, CGBoost algorithms with three types of loss function (exponential loss, logistic loss and MadaBoost loss) and ℓ_1-norm regularization. The errors are shown in percentage. All the algorithms are run 8 times. In all the cases, the logistic or MadaBoost loss outperforms the exponential loss, as expected.

Data	Method	Train 10	Train 50	Train 100	Train 500	Train 1000	Test 10	Test 50	Test 100	Test 500	Test 1000
image	AdaBoost	28.8 ± 1.8	22.9 ± 0.5	21.6 ± 0.8	20.8 ± 1.2	20.8 ± 1.2	19.2 ± 3.2	9.8 ± 1.6	9.6 ± 1.5	10.0 ± 1.5	10.0 ± 1.5
	CG, exp	25.6 ± 0.5	$\mathbf{20.5 \pm 0.2}$	$\mathbf{17.2 \pm 1.1}$	$\mathbf{8.9 \pm 3.8}$	$\mathbf{8.9 \pm 3.8}$	12.9 ± 1.1	13.6 ± 2.5	15.8 ± 3.6	19.6 ± 2.2	19.6 ± 2.2
	CG, logit	25.1 ± 0.6	20.9 ± 0.4	20.0 ± 0.5	19.7 ± 0.7	19.7 ± 0.7	12.2 ± 0.9	$\mathbf{8.3 \pm 0.6}$	$\mathbf{8.2 \pm 1.3}$	$\mathbf{8.9 \pm 1.8}$	$\mathbf{8.9 \pm 1.8}$
	CG, Mada	$\mathbf{24.2 \pm 0.7}$	20.9 ± 0.8	21.4 ± 1.9	21.2 ± 2.1	21.2 ± 2.1	$\mathbf{10.7 \pm 2.0}$	8.5 ± 1.2	9.4 ± 1.7	10.7 ± 1.7	10.7 ± 1.7
ringnorm	AdaBoost	29.5 ± 1.7	19.6 ± 0.9	17.4 ± 1.5	13.5 ± 5.0	13.0 ± 5.7	27.8 ± 1.0	17.4 ± 1.1	17.3 ± 1.1	17.8 ± 1.2	17.8 ± 1.2
	CG, exp	27.3 ± 0.8	17.7 ± 0.8	$\mathbf{10.2 \pm 3.0}$	$\mathbf{4.3 \pm 3.5}$	$\mathbf{4.3 \pm 3.5}$	$\mathbf{23.4 \pm 0.8}$	19.9 ± 1.3	21.8 ± 2.2	21.6 ± 1.1	21.6 ± 1.1
	CG, logit	26.9 ± 1.1	17.8 ± 1.0	16.1 ± 1.1	15.7 ± 1.7	15.7 ± 1.7	23.9 ± 1.5	15.2 ± 1.1	14.7 ± 1.3	16.2 ± 1.9	16.2 ± 1.9
	CG, Mada	$\mathbf{26.4 \pm 1.2}$	$\mathbf{17.1 \pm 1.8}$	15.6 ± 3.1	14.7 ± 3.4	14.7 ± 3.4	23.6 ± 1.4	$\mathbf{14.6 \pm 1.4}$	$\mathbf{14.3 \pm 2.3}$	$\mathbf{14.6 \pm 3.3}$	$\mathbf{14.6 \pm 3.3}$
twonorm	AdaBoost	24.5 ± 1.2	20.8 ± 1.8	20.1 ± 2.7	19.7 ± 3.3	19.7 ± 3.3	16.2 ± 0.6	13.2 ± 0.7	13.6 ± 1.0	13.7 ± 1.2	13.7 ± 1.2
	CG, exp	23.6 ± 1.2	$\mathbf{18.7 \pm 1.2}$	$\mathbf{15.3 \pm 3.7}$	$\mathbf{11.6 \pm 4.6}$	$\mathbf{11.6 \pm 4.6}$	15.0 ± 0.4	13.5 ± 3.9	14.5 ± 4.6	22.9 ± 4.6	22.9 ± 4.6
	CG, logit	$\mathbf{22.9 \pm 1.1}$	19.7 ± 1.0	19.5 ± 0.8	19.5 ± 0.8	19.5 ± 0.8	14.8 ± 0.6	7.6 ± 0.5	7.6 ± 0.5	$\mathbf{7.8 \pm 0.6}$	$\mathbf{7.8 \pm 0.6}$
	CG, Mada	23.0 ± 1.1	18.8 ± 0.5	18.8 ± 0.5	18.6 ± 0.5	18.6 ± 0.5	$\mathbf{14.4 \pm 0.4}$	$\mathbf{6.9 \pm 0.5}$	$\mathbf{6.8 \pm 0.5}$	7.9 ± 1.4	7.9 ± 1.4
waveform	AdaBoost	26.3 ± 1.6	23.8 ± 4.2	23.8 ± 4.2	23.8 ± 4.2	23.8 ± 4.2	20.1 ± 1.6	19.5 ± 1.3	19.5 ± 1.3	19.5 ± 1.3	19.5 ± 1.3
	CG, exp	26.0 ± 1.4	22.9 ± 3.9	22.5 ± 5.7	$\mathbf{19.7 \pm 6.1}$	$\mathbf{19.7 \pm 6.1}$	17.3 ± 1.0	17.7 ± 2.5	18.0 ± 2.5	26.0 ± 5.7	26.0 ± 5.7
	CG, logit	25.0 ± 1.1	23.2 ± 1.7	22.8 ± 1.9	23.0 ± 1.9	23.0 ± 1.9	$\mathbf{16.2 \pm 1.3}$	$\mathbf{15.2 \pm 0.9}$	$\mathbf{15.2 \pm 0.8}$	$\mathbf{15.3 \pm 0.9}$	$\mathbf{15.3 \pm 0.9}$
	CG, Mada	$\mathbf{24.4 \pm 1.7}$	$\mathbf{22.3 \pm 4.0}$	$\mathbf{22.0 \pm 3.9}$	22.7 ± 3.6	22.7 ± 3.6	16.3 ± 1.0	15.6 ± 0.7	15.8 ± 0.9	18.2 ± 2.7	18.2 ± 2.7

complicated convex problems are usually involved. By explicitly establishing the primal and dual problems, we can solve the primal that has some special structure to exploit. We use L-BFGS-B to solve the primal. Compared with conventional fully-corrective algorithms that use standard convex solvers, ours is much faster. Fig. 3 illustrates the time consumption of training for AdaBoost, CGBoost (exponential loss with ℓ_1) that solves the primal with L-BFGS-B and that solves the dual with Mosek, respectively. Note that with this loss-regularization combination, CGBoost solves the essentially same problem as AdaBoost. We can clearly see the advantage of solving the primal.

The second experiment is to evaluate the classification performance of different loss functions on *noisy* data. The exponential loss is generally sensitive to noise owing to the "over-penalty" for the training samples with negative margin (see the loss function shape in Fig. 1). In contrast, logistic loss and MadaBoost loss are supposed to be much more robust on noisy data. This experiment is performed on several artificial noisy datasets to confirm this assumption. The noise is generated by randomly flipping the labels of a subset of the original dataset. Our test is carried out with AdaBoost and ℓ_1 norm regularized CGBoost with logistic loss and MadaBoost loss on 4 noisy datasets originated from image, ringnorm, twonorm and waveform respectively. 20% of the labels are flipped. Experimental results are displayed in Table 6. We can see that after 1000 rounds, AdaBoost always ranks first in terms of the training error. In terms of test error, The exponential loss is almost always inferior to the MadaBoost loss and logistic loss. The MadaBoost loss and Logistic loss show their superiority on classification with the noisy data over other algorithms. Standard AdaBoost over-fits on all the noisy datasets progressively.

One of the appealing properties of the proposed fully corrective CGBoost is that at each iteration, more

Table 7: Two weak classifiers are added at each step of CGBoost with exponential loss and ℓ_1 or ℓ_2 regularization. Each case is run 5 times. Training and test error in percentage (%) are reported at iteration 25, 50, 250 and 500. The parameter setting for cross validation is the same as the first experiment.

Data	Method	Train 25	Train 50	Train 250	Train 500	Test 25	Test 50	Test 250	Test 500
banana	CG, ℓ_1	24.1 ± 0.3	22.9 ± 0.4	19.5 ± 0.7	19.5 ± 0.7	27.4 ± 1.0	28.4 ± 0.9	30.1 ± 1.8	30.1 ± 1.8
	CG, ℓ_2	24.2 ± 0.5	22.7 ± 0.6	18.6 ± 1.5	18.6 ± 1.5	28.3 ± 1.3	29.3 ± 1.2	29.6 ± 0.8	29.6 ± 0.8
b-cancer	CG, ℓ_1	21.9 ± 2.0	22.4 ± 2.0	23.0 ± 2.3	23.0 ± 2.3	26.1 ± 2.0	25.5 ± 1.9	25.0 ± 2.1	25.0 ± 2.1
	CG, ℓ_2	22.2 ± 1.1	24.2 ± 6.2	23.4 ± 6.4	23.4 ± 6.4	25.7 ± 1.6	29.1 ± 3.9	30.3 ± 4.1	30.3 ± 4.1
diabetis	CG, ℓ_1	15.3 ± 1.3	12.9 ± 2.2	12.2 ± 1.8	12.2 ± 1.8	23.4 ± 1.2	24.6 ± 1.9	26.0 ± 1.9	26.0 ± 1.9
	CG, ℓ_2	15.3 ± 2.4	12.1 ± 6.7	17.0 ± 15.8	17.0 ± 15.8	24.6 ± 3.5	27.9 ± 4.4	32.8 ± 7.8	32.8 ± 7.8
f-solar	CG, ℓ_1	29.8 ± 1.2	29.8 ± 1.2	29.1 ± 1.6	29.1 ± 1.6	36.9 ± 6.1	37.6 ± 5.8	37.9 ± 4.6	37.9 ± 4.6
	CG, ℓ_2	29.5 ± 1.6	29.1 ± 1.6	30.0 ± 3.5	30.0 ± 3.5	39.0 ± 5.1	35.2 ± 2.1	33.8 ± 3.6	33.8 ± 3.6
german	CG, ℓ_1	18.5 ± 0.2	17.7 ± 0.8	16.0 ± 1.0	16.0 ± 1.0	25.9 ± 1.4	25.8 ± 1.7	27.5 ± 1.9	27.5 ± 1.9
	CG, ℓ_2	18.4 ± 0.3	15.7 ± 0.5	12.2 ± 1.4	12.2 ± 1.4	25.6 ± 1.0	26.6 ± 1.6	27.8 ± 1.6	27.8 ± 1.6
heart	CG, ℓ_1	4.9 ± 5.8	4.8 ± 5.9	5.4 ± 6.7	5.4 ± 6.7	19.6 ± 2.7	19.8 ± 3.1	20.2 ± 2.8	20.2 ± 2.8
	CG, ℓ_2	5.7 ± 4.2	3.6 ± 3.8	0.7 ± 1.5	0.7 ± 1.5	20.2 ± 2.3	21.9 ± 2.9	21.5 ± 3.2	21.5 ± 3.2
image	CG, ℓ_1	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	3.8 ± 0.5	3.0 ± 0.5	2.9 ± 0.5	2.9 ± 0.5
	CG, ℓ_2	0.8 ± 0.7	0.5 ± 0.5	0.3 ± 0.5	0.3 ± 0.5	3.8 ± 0.5	3.4 ± 0.5	3.1 ± 0.7	3.1 ± 0.7
ringnorm	CG, ℓ_1	0.4 ± 0.4	0.1 ± 0.1	0.0 ± 0.0	0.0 ± 0.0	8.2 ± 1.1	6.2 ± 0.6	5.9 ± 0.6	5.9 ± 0.6
	CG, ℓ_2	0.4 ± 0.5	0.1 ± 0.1	0.0 ± 0.0	0.0 ± 0.0	9.5 ± 1.2	7.2 ± 0.8	5.8 ± 0.2	5.8 ± 0.2
splice	CG, ℓ_1	7.2 ± 0.4	6.3 ± 0.3	6.3 ± 0.3	6.3 ± 0.3	8.8 ± 0.7	8.1 ± 0.4	8.3 ± 0.1	8.3 ± 0.1
	CG, ℓ_2	7.2 ± 0.5	6.2 ± 0.3	5.9 ± 0.5	5.9 ± 0.5	8.5 ± 0.7	8.6 ± 0.4	9.4 ± 0.3	9.4 ± 0.3
thyroid	CG, ℓ_1	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	7.4 ± 2.4	7.7 ± 2.6	7.4 ± 2.4	7.4 ± 2.4
	CG, ℓ_2	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	7.7 ± 2.0	7.4 ± 2.7	7.2 ± 3.1	7.2 ± 3.1
titanic	CG, ℓ_1	12.9 ± 8.3	12.9 ± 8.3	12.9 ± 8.3	12.9 ± 8.3	24.0 ± 10.2	24.0 ± 10.2	24.0 ± 10.2	24.0 ± 10.2
	CG, ℓ_2	12.9 ± 8.3	12.9 ± 8.3	12.9 ± 8.3	12.9 ± 8.3	24.0 ± 10.2	24.0 ± 10.2	24.0 ± 10.2	24.0 ± 10.2
twonorm	CG, ℓ_1	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	4.6 ± 0.2	4.0 ± 0.2	3.9 ± 0.2	3.9 ± 0.2
	CG, ℓ_2	0.0 ± 0.1	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	4.4 ± 0.3	3.8 ± 0.1	3.8 ± 0.2	3.8 ± 0.2
waveform	CG, ℓ_1	1.1 ± 1.2	0.6 ± 0.8	0.9 ± 1.0	0.9 ± 1.0	13.5 ± 1.1	12.9 ± 0.8	13.2 ± 0.7	13.2 ± 0.7
	CG, ℓ_2	2.2 ± 2.1	1.0 ± 1.7	1.7 ± 3.4	1.7 ± 3.4	12.9 ± 0.4	12.7 ± 0.4	13.6 ± 1.6	13.6 ± 1.6

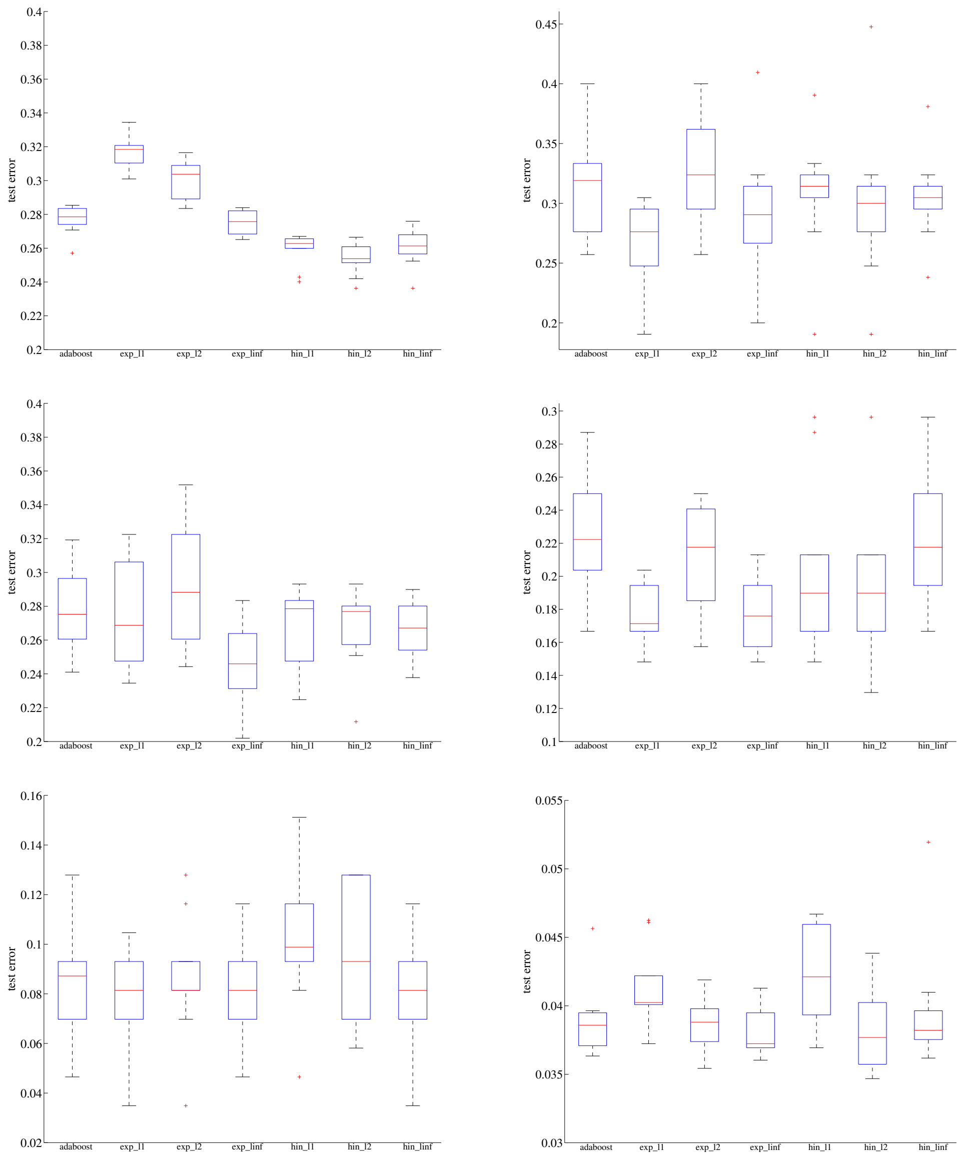

Figure 2: Box plots of test errors with 1000 weak learners for different methods on the datasets of "banana", "breast cancer", "diabetis", "heart", "thyroid", and "twonorm".

than one weak learners can be added into the optimization problem, as long as each weak learner corresponds to a violated constraint in the dual problem. In contrast, stage-wise boosting can only include one weak learner at each iteration. Adding multiple weak learners at each iteration can improve the convergence rate in general, as demonstrated in the following experiment. Here we add two weak learner at each iteration of CGBoost. Decision stump is used as the weak classifier, again, because of its simplicity. The strategy for choosing the two weak classifiers at each iteration is as follows. We train the decision stump with minimum weighted error at each dimension and the top two decision stumps are added into the primal problem. That is, the first weak classifier is actually selected same as in the standard CGBoost. The experiment setup is the same as the first experiment that we have run. Table 7 reports the results (the exponential loss with ℓ_1 or ℓ_2 regularization). Compared with the results of the standard CGBoost in Table 8, the test errors are comparable. One some datasets, the new strategy is even slightly better. Only on the dataset "titanic", the

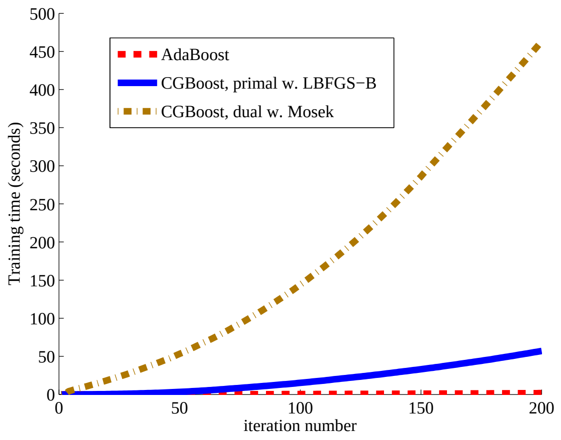

Figure 3: Cumulative training time needed for AdaBoost, CGBoost (exponential loss and ℓ_1 norm regularization) that solves the primal using LBFGS-B and that solves the dual using Mosek. We can see that using LBFGS-B to solve the primal is much faster. Here we have used 80% of the banana dataset for training. A standard desktop PC is used here.

new strategy is slightly worse. We can draw the conclusion that, in CGBoost, adding multiple weak learners at each iteration may produce comparable test accuracy with much faster convergence. For applications like boosting based face detection [8], to reduce half of the training time could be desirable.

6. Conclusion and Discussion

In this work, we have presented an abstract fully-corrective boosting framework (CGBoost) that can be used to minimize a broad range of regularized risk. An objective in the form of an arbitrary convex loss function plus an arbitrary convex regularization term can be minimized using the boosting technique developed here. We have also shown that a few existing boosting algorithms can be interpreted within our framework.

Like the seminal work of AnyBoost [11], which has been extensively used to design new stage-wise boosting methods, the proposed CGBoost framework may inspire new fully-corrective boosting algorithms. For example, Shen *et al.* has designed FisherBoost for asymmetric learning in a cascade classifier [42]. Compared with stage-wise boosting, fully-corrective boosting is more flexible in the sense that domain knowledge in the form of additional constraints may be taken into consideration. The stage-wise AnyBoost cannot deal with constraints. Another flexibility of fully-corrective boosting is that multiple weak learners can be added into the strong learner. As long as the added weak learners violate the current solution, it is guaranteed that the primal cost will be reduced. In contrast, stage-wise boosting can only include one weak classifier per iteration. Our preliminary experiments on training a face detector show that the training time can be reduced when two weak classifiers are added at each iteration, without compromising the detection performance.

We believe that the proposed CGBoost framework will help to understand how boosting works in general.

References

[1] R. Meir, G. Rätsch, An introduction to boosting and leveraging, Advanced lectures on machine learning, Springer-Verlag, pp. 118–183.

[2] R. E. Schapire, Theoretical views of boosting and applications, in: Proc. Int. Conf. Algorithmic Learn. Theory, Springer-Verlag, London, UK, 1999, pp. 13–25.

[3] A. Demiriz, K. Bennett, J. Shawe-Taylor, Linear programming boosting via column generation, Mach. Learn. 46 (2002) 225–254.

[4] C. Shen, H. Li, Boosting through optimization of margin distributions, IEEE Trans. Neural Netw. 21 (2010) 659–666.

[5] N. Garcia-Pedrajas, Constructing ensembles of classifiers by means of weighted instance selection, IEEE Trans. Neural Netw. 20 (2009) 258–277.
[6] T. Kanamori, Deformation of log-likelihood loss function for multiclass boosting, Neural Networks 23 (2010) 843–864.
[7] J. Lu, K. N. Plataniotis, A. N. Venetsanopoulos, S. Z. Li, Ensemble-based discriminant learning with boosting for face recognition, IEEE Trans. Neural Netw. 17 (2006) 166–178.
[8] P. Viola, M. J. Jones, Robust real-time face detection, Int. J. Comp. Vis. 57 (2004) 137–154.
[9] L. G. Valiant, A theory of the learnable, Commun. ACM 27 (1984) 1134–1142.
[10] J. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors), Ann. Statist. 28 (2000) 337–407.
[11] L. Mason, J. Baxter, P. Bartlett, M. Frean, Functional gradient techniques for combining hypotheses, Advances in Large Margin Classifiers, MIT press, pp. 221–247.
[12] J. H. Friedman, Greedy function approximation: A gradient boosting machine, Ann. Statist. 29 (2001) 1189–1232.
[13] C. Shen, H. Li, On the dual formulation of boosting algorithms, IEEE Trans. Pattern Anal. Mach. Intell. 32 (2010) 2216–2231. 25 Feb. 2010. IEEE computer Society Digital Library. http://doi.ieeecomputersociety.org/10.1109/TPAMI.2010.47.
[14] S. Rosset, J. Zhu, T. Hastie, Boosting as a regularized path to a maximum margin classifier, J. Mach. Learn. Res. 5 (2004) 941–973.
[15] J. Duchi, Y. Singer, Boosting with structural sparsity, in: Proc. Int. Conf. Mach. Learn., ACM, Montreal, Quebec, Canada, 2009, pp. 297–304.
[16] S. Z. Li, Z. Zhang, FloatBoost learning and statistical face detection, IEEE Trans. Pattern Anal. Mach. Intell. 26 (2004) 1112–1123.
[17] T. Zhang, Adaptive forward-backward greedy algorithm for sparse learning with linear models, in: Proc. Adv. Neural Inf. Process. Syst., Vancouver, Canada.
[18] M. K. Warmuth, J. Liao, G. Rätsch, Totally corrective boosting algorithms that maximize the margin, in: Proc. Int. Conf. Mach. Learn., Pittsburgh, Pennsylvania, pp. 1001–1008.
[19] C. Zhu, R. H. Byrd, J. Nocedal, L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization, ACM Trans. Math. Softw. 23 (1997) 550–560.
[20] V. Vapnik, The nature of statistical learning theory, Statistics for Engineering and Information Science, Springer Verlag, Berlin, 2000.
[21] A. N. Tikhonov, V. Y. Arsenin, Solution of Ill-posed Problems, Winston & Sons, 1977.
[22] R. Tibshirani, Regression shrinkage and selection via the Lasso, J. Royal Stat. Soc. Ser. B (Methodological) 58 (1996) 267–288.
[23] E. J. Candès, M. B. Wakin, An introduction to compressive sampling, IEEE Signal Process. Mag. 25 (2008) 21–30.
[24] A. Y. Ng, Feature selection, ℓ_1 vs. ℓ_2 regularization, and rotational invariance, in: Proc. Int. Conf. Mach. Learn., Banff, Alberta, Canada, pp. 78–83.
[25] Y. Xi, Z. Xiang, P. J. Ramadge, R. Schapire, Speed and sparsity of regularized boosting, in: Proc. Int. Workshop Artificial Intell. & Statistics, Florida, US, pp. 615–622.
[26] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[27] R. T. Rockafella, Convex Analysis, Princeton University Press, 1997.
[28] C. Domingo, O. Watanabe, MadaBoost: A modification of AdaBoost, in: Proc. Annual Conf. Learn. Theory, Morgan Kaufmann, 2000, pp. 180–189.
[29] G. Rätsch, M. K. Warmuth, Efficient margin maximization with boosting, J. Mach. Learn. Res. 6 (2005) 2131–2152.
[30] R. E. Schapire, Y. Freund, Improved boosting algorithms using confidence-rated predictions, Mach. Learn. 37 (1999) 297–336.
[31] T. Zhang, B. Yu, Boosting with early stopping: Convergence and consistency, Ann. Statist. 33 (2005) 1538–1579.
[32] L. Breiman, Prediction games and arcing algorithms, Neural Comp. 11 (1999) 1493–1517.
[33] L. Reyzin, R. E. Schapire, How boosting the margin can also boost classifier complexity, in: Proc. Int. Conf. Mach. Learn., Pittsburgh, Pennsylvania, USA.
[34] MOSEK ApS, The MOSEK optimization toolbox for matlab manual, version 5.0, revision 93, 2008. http://www.mosek.com/.
[35] H. Zou, M. Yuan, The f_∞-norm support vector machine, Statistica Sinica 18 (2008) 379–398.
[36] J. Bi, T. Zhang, K. P. Bennett, Column-generation boosting methods for mixture of kernels, in: Proc. ACM Int. Conf. Knowledge Discovery & Data Mining, ACM, Seattle, WA, USA, 2004, pp. 521–526.
[37] G. Rätsch, T. Onoda, K.-R. Müller, Soft margins for AdaBoost, Mach. Learn. 42 (2001) 287–320. Data sets are available at http://theoval.cmp.uea.ac.uk/~gcc/matlab/index.shtml.
[38] D. Newman, S. Hettich, C. Blake, C. Merz, UCI repository of machine learning databases, 1998. http://archive.ics.uci.edu/ml/.
[39] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[40] J. M. Chambers, W. S. Cleveland, P. A. Tukey, B. Kleiner, Graphical Methods for Data Analysis, Wadsworth & Brooks, 1983.
[41] M. K. Warmuth, K. Glocer, S. V. Vishwanathan, Entropy regularized lpboost, in: Proc. Int. Conf. Algorithmic Learn. Theory, Budapest, Hungary, pp. 256–271.
[42] C. Shen, P. Wang, H. Li, LACBoost and FisherBoost: Optimally building cascade classifiers, in: Proc. Eur. Conf. Comp. Vis., volume 2, LNCS 6312, Crete Island, Greece, pp. 608–621.

Appendix

ℓ_p-regularized CGBoost Regression

We list the special cases of CGBoost for regularization (primal (40) and dual (41)).

The ℓ_1 regularized primal can be written as

$$\min_{\boldsymbol{w},\boldsymbol{\gamma}} \sum_{i=1}^{m} l(\gamma_i), \text{ s.t.: } \|\boldsymbol{w}\|_1 \le r, \boldsymbol{w} \succcurlyeq \boldsymbol{0}, \gamma_i = y_i - H_{i:}\boldsymbol{w}\,(\forall i). \tag{43}$$

Its corresponding dual is

$$\min_{\boldsymbol{u},s} \sum_{i=1}^{m} l^*(u_i) - \boldsymbol{y}^\top \boldsymbol{u} + rs, \text{ s.t.: } \boldsymbol{u}^\top H \preccurlyeq s\boldsymbol{1}^\top, s \ge 0. \tag{44}$$

Similar to (23), we define

$$d_j^+ = \left[\sum_{i=1}^{m} u_i H_{ij} \right]_+ . \tag{45}$$

The dual (44) can be written into

$$\min_{\boldsymbol{u}} \textstyle\sum_{i=1}^{m} l^*(u_i) - \boldsymbol{y}^\top \boldsymbol{u} + r\|\boldsymbol{d}^+\|_\infty .$$

The ℓ_∞ regularized primal writes

$$\min_{\boldsymbol{w},\boldsymbol{\gamma}} \sum_{i=1}^{m} l(\gamma_i), \text{ s.t.: } \|\boldsymbol{w}\|_\infty \le r, \boldsymbol{w} \succcurlyeq \boldsymbol{0}, \gamma_i = y_i - H_{i:}\boldsymbol{w}\,(\forall i). \tag{46}$$

Its dual is

$$\min_{\boldsymbol{u}} \textstyle\sum_{i=1}^{m} l^*(u_i) - \boldsymbol{y}^\top \boldsymbol{u} + r\|\boldsymbol{d}^+\|_1 .$$

The ℓ_2 regularized primal writes

$$\min_{\boldsymbol{w},\boldsymbol{\gamma}} \sum_{i=1}^{m} l(\gamma_i) + \tfrac{1}{2}\vartheta\|\boldsymbol{w}\|_2^2, \text{ s.t.: } \boldsymbol{w} \succcurlyeq \boldsymbol{0}, \gamma_i = y_i - H_{i:}\boldsymbol{w}\,(\forall i). \tag{47}$$

The dual is

$$\min_{\boldsymbol{u}} \textstyle\sum_{i=1}^{m} l^*(u_i) - \boldsymbol{y}^\top \boldsymbol{u} + \frac{1}{2\vartheta}\|\boldsymbol{d}^+\|_2^2 .$$

Table 8: Training and test errors of AdaBoost, CGBoost with two types of loss functions (exponential and hinge) and three types of regularization (ℓ_1 norm, ℓ_2 norm and ℓ_∞ norm). The errors are shown in percentage (%). All the methods are run 40 times.

	Method	Train 50	100	500	1000	Test 50	100	500	1000
banana	AdaBoost	27.0±1.5	26.1±0.8	24.6±0.7	23.9±0.7	29.0±1.2	28.4±1.0	27.8±0.8	27.7±0.8
	CG exp, ℓ_1	25.2±0.6	24.0±0.5	**17.9**±2.4	**17.9**±2.4	27.9±1.1	28.4±1.2	31.8±0.9	31.8±0.9
	CG exp, ℓ_2	24.4±0.8	23.1±0.5	18.7±0.7	18.7±0.7	28.1±0.9	28.7±0.7	30.1±1.0	30.1±1.0
	CG exp, ℓ_∞	27.8±1.0	28.1±1.2	24.5±0.9	24.5±0.9	29.7±1.1	29.8±1.3	27.5±0.7	27.5±0.7
	CG hin, ℓ_1	**22.9**±0.8	**22.6**±1.2	22.4±1.5	22.4±1.5	**25.8**±1.2	**25.8**±0.9	25.9±0.9	25.9±0.9
	CG hin, ℓ_2	27.0±1.0	24.2±1.3	23.0±1.0	23.0±1.0	28.3±1.5	26.3±1.5	**25.4**±0.9	**25.4**±0.9
	CG hin, ℓ_∞	33.9±7.0	33.0±8.1	21.9±2.3	21.9±2.3	35.5±6.8	35.0±7.5	26.0±1.0	26.0±1.0
b-cancer	AdaBoost	20.9±2.9	**20.4**±2.8	**20.2**±2.5	**20.3**±2.7	29.3±4.2	30.1±4.4	31.2±4.5	31.6±4.0
	CG exp, ℓ_1	22.2±3.1	22.3±2.9	22.3±3.2	22.3±3.2	**26.7**±3.2	**26.9**±3.3	**26.7**±3.3	**26.7**±3.3
	CG exp, ℓ_2	**20.8**±3.1	20.8±2.7	21.0±3.4	21.0±3.4	29.9±5.0	30.4±5.0	32.4±4.3	32.4±4.3
	CG exp, ℓ_∞	25.2±5.0	22.5±2.3	21.3±2.7	21.3±2.7	30.8±6.9	27.7±5.0	29.2±5.2	29.2±5.2
	CG hin, ℓ_1	22.5±3.6	22.5±3.6	22.5±3.6	22.5±3.6	30.7±4.8	30.7±4.8	30.7±4.8	30.7±4.8
	CG hin, ℓ_2	24.8±2.5	25.8±5.0	24.6±5.8	24.6±5.8	29.1±3.9	30.0±6.2	30.0±6.2	30.0±6.2
	CG hin, ℓ_∞	25.0±1.5	24.8±1.6	22.3±3.9	22.3±3.9	29.4±3.2	29.5±2.6	30.5±3.4	30.5±3.4
diabetis	AdaBoost	17.7±1.6	15.4±1.2	9.4±0.7	**4.8**±1.2	25.8±1.9	25.9±2.1	26.4±2.6	27.8±2.3
	CG exp, ℓ_1	17.3±1.8	17.5±1.8	17.0±1.8	17.0±1.8	**25.0**±2.5	**25.3**±2.5	27.7±3.0	27.7±3.0
	CG exp, ℓ_2	**15.2**±2.2	**11.3**±5.6	**8.2**±6.8	8.2±6.8	26.5±1.6	27.4±2.8	29.0±3.3	29.0±3.3
	CG exp, ℓ_∞	24.0±2.1	22.8±1.2	19.2±1.5	19.2±1.5	27.1±2.0	26.4±3.3	**24.5**±2.2	**24.5**±2.2
	CG hin, ℓ_1	19.5±4.0	19.1±4.6	19.1±4.6	19.1±4.6	26.5±2.2	26.7±2.3	26.7±2.3	26.7±2.3
	CG hin, ℓ_2	23.4±1.9	21.8±2.5	18.3±3.8	18.3±3.8	27.1±2.4	26.1±2.9	26.7±2.2	26.7±2.2
	CG hin, ℓ_∞	23.9±1.6	23.9±1.6	17.6±4.0	17.6±4.0	27.1±2.4	27.1±2.4	26.6±1.6	26.6±1.6
f-solar	AdaBoost	30.6±4.8	29.8±5.8	30.2±6.0	30.5±5.5	34.0±4.3	35.3±5.3	35.0±5.5	35.0±5.8
	CG exp, ℓ_1	30.0±5.3	29.9±5.4	30.1±5.6	30.1±5.6	34.8±5.3	34.7±5.5	35.5±4.9	35.5±4.9
	CG exp, ℓ_2	30.0±5.6	29.5±5.8	30.4±6.5	30.4±6.5	36.1±5.9	36.1±6.1	35.9±6.7	35.9±6.7
	CG exp, ℓ_∞	33.8±7.6	30.5±6.1	30.0±5.9	30.0±5.9	37.9±5.3	34.4±5.3	34.2±4.7	34.2±4.7
	CG hin, ℓ_1	**28.4**±5.7	**28.4**±5.7	28.4±5.7	28.4±5.7	**33.8**±5.3	**33.8**±5.3	**33.8**±5.3	**33.8**±5.3
	CG hin, ℓ_2	29.9±6.2	29.2±5.9	**27.4**±5.0	**27.4**±5.0	35.5±5.8	34.8±7.1	34.9±5.7	34.9±5.7
	CG hin, ℓ_∞	36.5±5.4	34.9±6.8	**27.4**±6.0	**27.4**±6.0	41.2±7.6	41.1±6.2	34.6±5.0	34.6±5.0
german	AdaBoost	20.2±1.1	19.0±1.1	15.6±1.1	13.9±1.7	**25.0**±1.6	**24.5**±2.1	26.2±2.3	27.1±2.0
	CG exp, ℓ_1	19.8±1.7	19.5±1.8	19.0±2.3	19.0±2.3	**25.0**±2.0	25.2±1.7	26.3±1.9	26.3±1.9
	CG exp, ℓ_2	18.9±1.5	15.9±2.3	**11.1**±4.2	**11.1**±4.2	26.3±2.0	27.2±2.1	28.0±2.1	28.0±2.1
	CG exp, ℓ_∞	26.2±1.7	24.2±2.1	19.7±1.6	19.7±1.6	27.8±2.5	26.0±2.1	**24.2**±1.9	**24.2**±1.9
	CG hin, ℓ_1	**18.8**±2.0	**18.4**±2.5	18.4±2.5	18.4±2.5	25.7±1.9	25.7±1.7	25.7±1.7	25.7±1.7
	CG hin, ℓ_2	24.3±3.0	21.3±2.5	17.4±1.8	17.4±1.8	27.5±2.2	26.1±2.4	25.7±2.2	25.7±2.2
	CG hin, ℓ_∞	28.1±1.1	28.1±1.1	17.8±1.3	17.8±1.3	30.0±2.0	30.0±2.0	25.6±2.4	25.6±2.4
heart	AdaBoost	**5.7**±1.1	**2.0**±1.2	**0.0**±0.0	**0.0**±0.0	20.6±2.6	21.2±2.9	21.7±3.2	22.3±3.6
	CG exp, ℓ_1	11.4±1.6	11.5±1.6	11.4±1.5	11.4±1.5	**18.1**±2.2	**17.8**±2.4	**17.7**±1.8	**17.7**±1.8
	CG exp, ℓ_2	7.2±4.6	5.4±4.0	2.2±2.3	2.2±2.3	19.4±3.3	20.1±3.0	21.2±3.1	21.2±3.1
	CG exp, ℓ_∞	16.0±3.7	13.8±2.2	11.2±1.6	11.2±1.6	20.4±5.5	18.2±3.1	17.8±2.3	17.8±2.3
	CG hin, ℓ_1	11.7±4.4	11.9±4.2	11.9±4.2	11.9±4.2	20.2±4.8	20.3±4.9	20.3±4.9	20.3±4.9
	CG hin, ℓ_2	16.4±4.3	15.1±4.9	11.2±4.3	11.2±4.3	21.6±5.5	20.6±4.8	19.3±4.3	19.3±4.3
	CG hin, ℓ_∞	20.9±5.3	16.2±5.8	7.2±5.3	7.2±5.3	26.5±2.0	23.8±3.6	22.0±3.6	22.0±3.6
image	AdaBoost	4.1±0.6	2.3±0.3	**0.0**±0.0	**0.0**±0.0	5.8±1.0	4.3±0.6	**2.9**±0.6	**2.9**±0.7
	CG exp, ℓ_1	**0.0**±0.1	**0.0**±0.0	**0.0**±0.0	**0.0**±0.0	4.2±0.5	**3.2**±0.7	3.0±0.8	3.0±0.8
	CG exp, ℓ_2	0.5±0.7	0.2±0.3	**0.0**±0.0	**0.0**±0.0	**4.1**±0.8	**3.2**±0.7	**2.9**±0.7	**2.9**±0.7
	CG exp, ℓ_∞	5.9±2.3	2.4±1.5	0.1±0.2	0.1±0.2	7.6±2.5	5.0±1.5	3.0±0.7	3.0±0.7
	CG hin, ℓ_1	1.9±0.5	1.2±0.6	1.0±0.6	1.0±0.6	**4.1**±1.0	3.5±1.1	3.3±0.5	3.3±0.5
	CG hin, ℓ_2	2.7±1.2	1.7±0.6	0.8±0.6	0.8±0.6	**4.1**±1.5	3.3±0.9	3.3±0.5	3.3±0.5
	CG hin, ℓ_∞	8.3±2.1	3.6±1.4	**0.0**±0.0	**0.0**±0.0	9.6±1.7	5.1±1.1	3.7±0.7	3.7±0.7
ringnorm	AdaBoost	4.6±0.5	1.6±0.5	**0.0**±0.0	**0.0**±0.0	10.5±0.7	7.4±0.6	**5.4**±0.3	**5.4**±0.4
	CG exp, ℓ_1	**0.5**±0.6	**0.1**±0.4	0.1±0.3	0.1±0.3	11.6±1.8	7.0±0.8	5.7±0.7	5.7±0.7
	CG exp, ℓ_2	0.9±0.9	**0.1**±0.1	**0.0**±0.0	**0.0**±0.0	10.4±1.9	7.8±1.6	5.7±0.5	5.7±0.5
	CG exp, ℓ_∞	9.3±3.2	4.7±2.0	0.3±0.4	0.3±0.4	14.6±3.1	10.2±2.1	6.1±0.5	6.1±0.5
	CG hin, ℓ_1	2.4±0.5	1.4±0.4	1.0±0.5	1.0±0.5	**7.4**±0.4	**5.9**±0.5	5.8±0.4	5.8±0.4
	CG hin, ℓ_2	6.1±1.6	3.4±0.9	1.8±0.5	1.8±0.5	11.3±1.5	7.9±1.0	6.0±0.3	6.0±0.3
	CG hin, ℓ_∞	26.1±11.5	24.5±13.8	3.2±2.1	3.2±2.1	30.5±10.8	29.0±13.1	8.9±1.1	8.9±1.1
splice	AdaBoost	7.4±0.2	6.5±0.5	5.6±0.3	5.7±0.3	9.3±0.7	9.0±0.9	9.7±0.6	10.1±0.6
	CG exp, ℓ_1	6.7±0.4	6.2±0.3	6.2±0.2	6.2±0.2	9.0±1.5	8.7±1.0	8.7±0.8	8.7±0.8
	CG exp, ℓ_2	6.6±0.2	5.8±0.3	5.7±0.3	5.7±0.3	9.4±1.1	9.8±0.8	10.1±0.6	10.1±0.6
	CG exp, ℓ_∞	14.4±0.1	10.8±0.4	6.7±0.2	6.7±0.2	14.7±0.4	11.8±0.4	8.5±0.6	8.5±0.6

	CG hin, ℓ_1	**6.0**±0.3	**5.4**±0.3	**5.4**±0.3	**5.4**±0.3	**8.8**±0.5	**8.5**±0.4	**8.3**±0.3	**8.3**±0.3
	CG hin, ℓ_2	7.5±0.3	6.2±0.2	**5.4**±0.2	**5.4**±0.2	9.2±0.7	8.6±0.8	**8.3**±0.4	**8.3**±0.4
	CG hin, ℓ_∞	22.9±0.9	22.8±0.8	6.2±0.2	6.2±0.2	23.2±1.4	23.1±1.5	8.5±0.6	8.5±0.6
thyroid	AdaBoost	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	**7.8**±2.3	**7.6**±2.2	**8.1**±2.3	8.3±2.4
	CG exp, ℓ_1	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	8.1±1.9	8.1±1.9	**8.1**±1.9	**8.1**±1.9
	CG exp, ℓ_2	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	8.4±2.6	8.3±2.4	8.5±2.4	8.5±2.4
	CG exp, ℓ_∞	2.3±2.4	1.0±1.0	**0.0±0.0**	**0.0±0.0**	9.2±2.9	8.4±2.3	8.4±2.1	8.4±2.1
	CG hin, ℓ_1	1.3±1.7	1.3±1.7	1.3±1.7	1.3±1.7	10.1±2.7	10.1±2.7	10.1±2.7	10.1±2.7
	CG hin, ℓ_2	2.6±4.4	1.0±1.3	0.7±1.3	0.7±1.3	10.3±4.9	9.2±3.3	9.3±2.6	9.3±2.6
	CG hin, ℓ_∞	2.6±4.8	1.6±4.9	0.2±0.5	0.2±0.5	10.2±5.3	9.7±5.3	7.9±2.2	7.9±2.2
titanic	AdaBoost	19.3±9.1	19.3±9.1	19.3±9.1	19.3±9.1	**19.0**±9.4	**19.0**±9.4	**19.0**±9.4	**19.0**±9.4
	CG exp, ℓ_1	19.3±9.1	19.3±9.1	19.3±9.1	19.3±9.1	**19.0**±9.4	**19.0**±9.4	**19.0**±9.4	**19.0**±9.4
	CG exp, ℓ_2	19.3±9.1	19.3±9.1	19.3±9.1	19.3±9.1	**19.0**±9.4	**19.0**±9.4	**19.0**±9.4	**19.0**±9.4
	CG exp, ℓ_∞	17.9±8.0	17.9±8.0	18.6±8.0	18.6±8.0	20.0±11.0	21.0±11.4	21.0±11.4	21.0±11.4
	CG hin, ℓ_1	**17.1**±7.3	**17.1**±7.3	**17.1**±7.3	**17.1**±7.3	**19.0**±12.2	**19.0**±12.2	**19.0**±12.2	**19.0**±12.2
	CG hin, ℓ_2	17.9±8.0	17.9±8.0	17.9±8.0	17.9±8.0	**19.0**±12.2	**19.0**±12.2	**19.0**±12.2	**19.0**±12.2
	CG hin, ℓ_∞	**17.1**±7.3	**17.1**±7.3	**17.1**±7.3	**17.1**±7.3	**19.0**±12.2	**19.0**±12.2	**19.0**±12.2	**19.0**±12.2
twonorm	AdaBoost	0.3±0.4	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	4.5±0.2	4.0±0.3	**3.8**±0.2	3.9±0.3
	CG exp, ℓ_1	**0.1**±0.1	**0.0±0.0**	**0.0**±0.1	**0.0**±0.1	5.0±0.3	4.2±0.3	4.1±0.3	4.1±0.3
	CG exp, ℓ_2	**0.1**±0.3	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	**4.2**±0.2	**3.8**±0.2	3.9±0.2	3.9±0.2
	CG exp, ℓ_∞	1.3±1.3	0.4±0.7	**0.0±0.0**	**0.0±0.0**	5.0±0.7	4.2±0.3	**3.8**±0.2	**3.8**±0.2
	CG hin, ℓ_1	0.9±0.6	0.5±0.5	0.5±0.6	0.5±0.6	4.7±0.4	4.2±0.3	4.2±0.3	4.2±0.3
	CG hin, ℓ_2	1.3±0.7	0.6±0.6	0.3±0.4	0.3±0.4	4.4±0.1	**3.8**±0.2	**3.8**±0.3	**3.8**±0.3
	CG hin, ℓ_∞	12.7±13.9	11.2±13.7	0.5±0.6	0.5±0.6	15.9±13.6	15.2±13.2	4.0±0.4	4.0±0.4
waveform	AdaBoost	3.7±0.9	**0.7**±0.5	**0.0**±0.0	**0.0**±0.0	13.2±0.6	13.1±0.7	13.4±0.8	13.7±0.8
	CG exp, ℓ_1	**2.0**±1.1	1.6±1.0	1.7±1.1	1.7±1.1	13.3±1.4	**12.7**±0.8	**12.7**±0.8	**12.7**±0.8
	CG exp, ℓ_2	2.2±2.2	1.1±1.6	0.2±0.3	0.2±0.3	**13.1**±0.8	12.9±0.5	13.1±0.6	13.1±0.6
	CG exp, ℓ_∞	10.2±3.8	7.8±3.6	2.2±2.5	2.2±2.5	14.8±1.6	13.9±1.2	12.9±0.8	12.9±0.8
	CG hin, ℓ_1	3.9±1.5	3.0±1.5	2.9±1.7	2.9±1.7	12.8±0.8	13.1±1.0	13.1±1.0	13.1±1.0
	CG hin, ℓ_2	9.5±6.2	6.2±4.3	3.3±2.6	3.3±2.6	15.8±4.0	13.7±1.5	12.9±0.8	12.9±0.8
	CG hin, ℓ_∞	14.9±6.5	13.0±7.5	2.8±1.9	2.8±1.9	19.3±4.3	19.2±3.4	13.2±1.1	13.2±1.1