Journal of Machine Learning Research 20 (2019) 1-68 Submitted 3/17; Published 9/19

Multiclass Boosting: Margins, Codewords, Losses, and Algorithms

Mohammad Saberian SABERIAN@UCSD.EDU
Statistical Visual Computing Laboratory,
University of California, San Diego
La Jolla, CA 92039, USA

Nuno Vasconcelos NUNO@UCSD.EDU
Statistical Visual Computing Laboratory,
University of California, San Diego
La Jolla, CA 92039, USA

Editor: Koby Crammer

Abstract

The problem of multiclass boosting is considered. A new formulation is presented, combining multi-dimensional predictors, multi-dimensional real-valued codewords, and proper multiclass margin loss functions. This leads to a number of contributions, such as maximum capacity codeword sets, a family of proper and margin enforcing losses, denoted as $\gamma - \phi$ losses, and two new multiclass boosting algorithms. These are descent procedures on the functional space spanned by a set of weak learners. The first, CD-MCBoost, is a coordinate descent procedure that updates one predictor component at a time. The second, GD-MCBoost, a gradient descent procedure that updates all components jointly. Both MCBoost algorithms are defined with respect to a $\gamma - \phi$ loss and can reduce to classical boosting procedures (such as AdaBoost and LogitBoost) for binary problems. Beyond the algorithms themselves, the proposed formulation enables a unified treatment of many previous multiclass boosting algorithms. This is used to show that the latter implement different combinations of optimization strategy, codewords, weak learners, and loss function, highlighting some of their deficiencies. It is shown that no previous method matches the support of MCBoost for real codewords of maximum capacity, a proper margin-enforcing loss function, and any family of multidimensional predictors and weak learners. Experimental results confirm the superiority of MCBoost, showing that the two proposed MCBoost algorithms outperform comparable prior methods on a number of datasets.

Keywords: Boosting, Multiclass Boosting, Multiclass Classification, Margin Maximization, Loss Function.

1. Introduction

Boosting is a popular approach for classifier design. It is a simple and effective procedure to learn strong decision rules by combination of weak learners. However, most boosting algorithms were designed primarily for binary classification. In many cases, the extension to M-ary problems (of $M > 2$) is not straightforward. There are many ways to interpret boosting (Schapire and Freund, 2012) and these have been used to justify different multiclass extensions. In this work, we consider the view of boosting as a method for empirical risk minimization, using some optimization procedure (usually gradient descent) on the functional space spanned by a set of weak learners (Friedman et al., 1998; Mason et al., 2000). Under this view, the effectiveness of a boosting algorithm is determined by the underlying choices of optimization method, weak learner pool, and risk function. The latter can be further decomposed into a choice of a loss function and an encoding of class labels. All of these are fairly well understood in the binary setting. In some cases, such as the selection of codeword labels, there is virtually universal agreement in the literature, where most algorithms use -1 as the label of "negative" examples and $+1$ for "positives". On others, there is wide but not universal agreement. For example, while it can be shown that most boosting algorithms implement some form of functional gradient descent (Mason et al., 2000; Zhang, 2004; Buja et al., 2006; Masnadi-Shirazi and Vasconcelos, 2008), there have also been proposals to use second order optimization methods, based on Taylor series expansions (Saberian et al., 2010) and Newton's method (Friedman et al., 1998). The remaining aspects are less consensual. Most algorithms differ in terms of the loss function that defines the risk for which they are optimal. Several loss properties, such as encouraging large margins (Vapnik, 1998) and allowing the recovery of class probabilities (Zhang, 2004; Buja et al., 2006; Mease and Wyner, 2008; Masnadi-Shirazi and Vasconcelos, 2008; Reid and Williamson, 2010) have been identified as important, and shown to hold for a large family of functions (Masnadi-Shirazi and Vasconcelos, 2015). Some members of this family, such as the exponential loss of AdaBoost (Freund and Schapire, 1997), the logit loss of LogitBoost (Friedman et al., 1998), the hinge loss of support vector machines (Vapnik, 1998) or the Savage loss of SavageBoost (Masnadi-Shirazi and Vasconcelos, 2008), have been widely used in practice. Finally, weak learners tend to vary with the application. Popular choices include weak classifiers, such as the decision stumps or shallow decision trees commonly used in computer vision (Viola and Jones, 2004; Dollar et al., 2012), and regression models (Friedman et al., 1998).

To date, there have been no comprehensive efforts to extend this understanding to the M-ary case. While boosting components such as the optimization strategy generalize in a straightforward manner, others do not. In particular, many label encodings and loss functions are possible. There is limited understanding of what constitutes a good encoding or when a loss function is proper (allows the recovery of class conditional probabilities). As we will see later on, some popular choices for these components are quite suboptimal. In fact, for multiclass boosting, it is not even easy to guarantee that a set of multiclass weak learners can be boosted. For binary classification, it is well known this is possible as long as a weak learner with "less than 50% error" can be found in all bosting iterations. Since this only requires a weak learner marginaly better than random guessing, the condition can be met trivially. However, for M-ary classification, the accuracy of a random classifier is only $\frac{100}{M}\%$. In this case, straightforward extensions of binary boosting algorithms that require multiclass "weak" learners with "less than 50% error", such as the well known AdaBoost.M1 algorithm (Freund and Schapire, 1996), are too difficult to implement. In practice, they tend to stop prematurely and fail to produce strong ensemble decision rules. Due to all this, many of the existing approaches to multiclass boosting involve some form of reduction of the M-class problem to a collection of binary problems. While the most popular are the well known "one vs all" (Nilsson, 1965) and "all pairs" (Hastie and Tibshirani, 1998) classification architectures, many boosting algorithms based on this type of representation have been proposed. A smaller number of true multiclass methods, such as AdaBoost.M1(Freund and Schapire, 1996; Eibl and Schapire, 2005), SAMME (Zhu et al., 2009), and AdaBoost.MM (Mukherjee and Schapire, 2013) have also been proposed. These boost multiclass weak learners, usually decision trees. The limitations of these methods are discussed in Section 2.

In this work, we study multiclass boosting with the goal of an integrated understanding of the roles of the optimization strategy, label codewords, weak learners, and multiclass risk. This leads to a new formulation of the problem based on 1) multi-dimensional predictors, 2) multi-dimensional real valued codewords, and 3) proper multiclass margin loss functions. We start by studying the role of the label encoding, showing that the selected codewords impose an upper bound on the maximum margin achievable by any predictor. This is denoted the margin capacity bound. We then define a family of losses, denoted $\gamma - \phi$ losses, which extend the classical margin losses used by binary boosting algorithms. These losses connect the classification margin to a set of dot-products between a multidimensional predictor and a codeword set, enabling the formulation of multiclass boosting as a margin maximization problem in multidimensional functional space. This objective is formulated through an empirical risk that combines a $\gamma - \phi$ loss and a margin capacity codeword set. Two algorithms are then derived to solve this optimization. The first, denoted CD-MCBoost, implements a functional coordinate descent procedure. CD-MCBoost supports any type of weak learners, updating one component of the predictor per boosting iteration. This method has some similarities to binary reduction procedures but 1) uses real-valued codewords and 2) learns all predictor components jointly. The second, denoted GD-MCBoost, implements functional gradient descent, in a space of multidimensional weak learners, updating all predictor components simultaneously. Both MCBoost algorithms reduce to classical boosting algorithms (such as AdaBoost or LogitBoost) for binary problems, depending on the choice of $\gamma - \phi$ loss. They are also shown to exhibit classical boosting properties, such as seeking the weak learner of maximum margin on a reweighted training sample at each iteration and well defined boostability conditions. These properties are, however, shown to hold more generally, as would be expected of the multiclass setting. With respect to weights, MCBoost emphasizes not only the most difficult examples but also the most difficult classes, at each boosting iteration. With regards to boostability, MCBoost is shown to boost any set of weak learners with better than multiclass chance performance, i.e. less than $M/100\%$ error.

In the remainder of the paper, we consider the design of good codeword sets and $\gamma - \phi$ loss functions for MCBoost algorithms. We start by considering the design of a set of codewords of maximum capacity, for any predictor dimension d. We derive necessary and sufficient conditions for a codeword set to achieve this bound and show that such a codeword set is guaranteed to exist whenever $d > M - 1$. A procedure to generate the associated codewords is then presented. We next consider the case of low-dimensional predictors, with $d \leq M - 1$. We show that, while there are no guarantees of meeting the capacity bound in this case, it is possible to generate codeword sets that are optimal in a related sense, the max-min codeword distance. A procedure to generate such codeword sets is again provided. We next consider the interplay between codewords and risk, by studying several properties of $\gamma - \phi$ losses. In particular, we derive conditions under which these losses are margin enforcing and show that they are proper under mild conditions on the codeword set. These conditions are shown to hold for all codeword sets that achieve the margin capacity bound. By relating $\gamma - \phi$ losses to the binary margin losses of classical boosting algorithms, we then derive extensions of the latter to the multiclass setting. This leads to natural multiclass extensions of algorithms like AdaBoost (Freund and Schapire, 1997), LogitBoost (Friedman et al., 1998), and SavageBoost (Masnadi-Shirazi and Vasconcelos, 2008). The theoretical framework now introduced is finally used to place previous multiclass boosting algorithms in a common footing. This exercise shows that all algorithms implement different combinations of optimization strategy, codewords, weak learners, and loss function. It also highlights some of the deficiencies of these algorithms, explaining why they fail under some settings. In particular, it is shown that no previous method matches the support of MCBoost for real codewords of maximum capacity, a proper margin-enforcing loss function, multidimensional predictors and any class of weak learners. Experimental results confirm the superiority of MCBoost, showing that the two proposed MCBoost algorithms outperform comparable prior methods on a number of datasets.

The paper is organized as follows. In section 2, we briefly review prior work on multiclass boosting. The foundations of the proposed formulation are introduced in Section 3, where we propose a set of multiclass definitions for class labels, predictor, and margin. The MCBoost algorithms are derived in Section 4, where we also discuss properties such as weighting mechanisms and weak learners. The problem of optimal codeword design is then discussed in Section 5, where we introduce the notions of margin capacity and derive necessary and sufficient conditions for maximum capacity and max-min distance codeword sets. Section 6 is devoted to weak learners, analyzing issues such as boostability or the role of classification vs. real valued learners. Section 7 discusses various properties of interest for $\gamma - \phi$ losses, and introduces a number of losses that meet these properties, leading to the multiclass extensions of various classical boosting algorithms. These are compared to previous multiclass boosting algorithms in Section 8, where existing methods are studied in light of the proposed boosting framework. Finally, experimental results are discussed in Section 9 and some conclusions drawn in Section 10.

2. Related work

In this section, we briefly review previous work in multiclass boosting.

2.1. Origins

The problem of multiclass classification has attracted significant attention since the early days of machine learning. The first, and still popular, method for designing an M-class classifier is to learn M binary classifiers, each separating one class form the remaining (Nilsson, 1965). At the test time, a prediction score is computed by each binary classifier and the class of highest score is selected. This is known as “one-vs-all” (OVA) classification. Later, (Sejnowski and Rosenberg, 1987) extended the idea by assigning a binary string of length l to each class and learning l binary classifiers to predict the bits of those strings. While this is similar to OVA, the class selected at test time is that whose string is closest, in the Hamming distance sense, to that recovered by the binary classifiers. This was, to the best of our knowledge, the first attempt to represent classes by multidimensional codewords (binary strings in this case). (Dietterich and Bakiri, 1995) improved this idea by introducing an error correcting output code (ECOC). This consists of using techniques from the error correction literature to design a codeword set resistant to errors. It made it possible to recover the true class even if a few of the binary classifiers produced erroneous bits. (Hastie and Tibshirani, 1998) suggested an alternative approach, by designing $\frac{M(M-1)}{2}$ classifiers to discriminate between all pairs of classes. At test time, a class was selected through a vote among these classifiers. (Allwein et al., 2001) unified all these binary classification methods, showing that they all reduce the multiclass problem into binary sub-problems, for a specific choice of coding matrix.

2.2. Reductions to binary

Upon the introduction of AdaBoost (Freund and Schapire, 1997), there was a substantial effort to extend this simple and effective binary classification algorithm to multiclass problems. Like the methods above, many of these approaches are based on a reduction to a collection of binary classification problems. These can be grouped into three main sub-classes.

The first subclass follows the ECOC approach of (Dietterich and Bakiri, 1995). (Schapire, 1997) extended this approach by combining it with a pseudo-loss previously introduced to derive AdaBoost.M2 (Freund and Schapire, 1996). The combination of the two components, through the AdaBoost.OC algorithm, enabled the joint learning of the binary sub-classifiers. This work also proposed using binary random codes and codes designed to have the best error correction performance, using "max-cut" algorithms. A number of algorithms, including AdaBoost.ECC (Guruswami and Sahai, 1999; Allwein et al., 2001), AdaBoost.SECC (Sun et al., 2005), AdaBoost.ERP (Li), AdaBoost.SIP (Zhang et al., 2009) and HingeBoost (Gao and Koller, 2011), were then proposed to generalize or improve the combination of boosting and error correction. For example, (Guruswami and Sahai, 1999) modified the pseudo-loss of AdaBoost.OC and proposed AdaBoost.ECC, which was shown to have better generalization guarantees and performance. (Allwein et al., 2001) studied the impact of the codeword distance function, proposing a loss-based distance to replace the Hamming distance, during both training and classification. (Sun et al., 2005) connected these methods to the margin framework, showing that AdaBoost.OC and AdaBoost.ECC were in fact maximizing multiclass definitions of the margin. The problem of finding the optimal binary coding matrix was studied in (Li; Zhang et al., 2009; Gao and Koller, 2011), which proposed optimization methods to find a good set of codes for each boosting iteration. However, (Crammer and Singer, 2002b) showed that the problem of determining the optimal binary coding matrix is NP-hard, and suggested the use of real-valued codes. In general, the performance of these algorithms depends on two factors: 1) the error correction performance of the coding matrix and the weighting algorithms used to train the binary classifiers. In practice, the optimization of these two factors often requires extensive computation, for both training and classification. This is not surprising since, as pointed out by (Crammer and Singer, 2002b), the optimization problem is NP-hard.

The second subclass of binary reductions originated with AdaBoost.M2 (Freund and Schapire, 1997, 1996). This algorithm couples a class identifier $c \in \mathcal{C}$ with the example $x \in \mathcal{X}$, learning a real valued predictor $f : \mathcal{X} \times \mathcal{C} \rightarrow \mathbb{R}$ in the product space of examples and class labels. This predictor produces a score for each example x and class c . It is learned by minimizing a pseudo-loss function defined over all pairs of examples and incorrect labels, so as to penalize classification errors. At the test time, f is evaluated for all pairs of example x and classes c. The class of largest score is finally selected. (Schapire and Singer, 1999) used this approach to extend AdaBoost.M2 into AdaBoost.MR, which addressed multi-label problems. The procedure was also improved by the introduction of AdaBoost.MH, whose weights are updated using the Hamming loss. More recently, (Kuznetsov et al., 2014) introduced new upper bounds and improved algorithms for this class of approaches. As shown by (Friedman et al., 1998), coupling examples with class labels effectively converts the multiclass problem into a binary problem. This is, however, an extremely non-linear problem, since very similar examples, namely all (x, c) of common x, have very different scores (depending on c). In result, these binary reductions require weak learners of high discriminant power, usually deep decision trees. This makes the boosting algorithm prone to over-fitting.

The final subclass of binary reductions originated with the multiclass LogitBoost method of (Friedman et al., 1998). This is based on the statistical view of Boosting as gradient descent in functional space and learns additive logistic regression models for each class, using binary LogitBoost. While these regressors are binary predictors, the key difference to OVA is that they are learned jointly, and must add up to zero. (Huang et al., 2007) extended this framework to a multiclass version of GentleBoost, proposing the GAMBLE algorithm. This was further extended for any Fisher consistent loss function by the AdaBoost.ML algorithm of (Zou et al., 2008). However, it has been shown that scores of the binary regression classifiers do not represent true class probabilities (Mease and Wyner, 2008).

2.3. Boosting multiclass base learners

In contrast to the large literature in binary reductions of the multiclass boosting problem, there have been relatively few attempts to produce true multiclass boosting algorithms. The earliest such attempt is AdaBoost.M1 (Freund and Schapire, 1997, 1996), which directly extends AdaBoost to the multiclass setting. However, by maintaining the AdaBoost boostability requirement of "less than 50% error weak learners," this method has the limitations discussed in Section 1. A few attempts have been made to relax these weak learner requirements, such as the works of (Eibl and Schapire, 2005) and (Mukherjee and Schapire, 2013). The latter provided the most extensive treatment, introducing a game theoretic analysis of the necessary conditions for boostable base learners, proposing a less strict base learner selection criterion. However, the optimal criteria for selection of base learners is still an open problem. Alternatively, (Zhu et al., 2009) devoted more attention to the loss function, attacking the problem through the definition of an exponential loss for multiclass classification in multi-dimensional functional space. They then proposed a gradient descent procedure for this optimization, which was denoted as SAMME boosting. While this has some similarity to the framework now introduced, we show in Section 8 that the exponential loss function is not margin enforcing. Hence SAMME is not a maximum margin algorithm.

3. Multiclass boosting

In this work, we seek multiclass boosting algorithms that do not rely on binary reductions. We start by reviewing the fundamental ideas behind the classical use of boosting for the design of *binary* classifiers, and then extend these ideas to the multiclass setting.

3.1. Binary classification

A binary classifier, $F(x)$, implements a *decision rule* that maps examples $x \in \mathcal{X}$ to classes $c \in \{1, 2\}$. The classifier is optimal when this decision rule minimizes some *classification risk*. A classical risk is the probability of classification error, which is minimized by the Bayes decision rule

$$F(x) = \arg\min_{c\in\{1,2\}} P_{C|X}(c|x). \tag{1}$$

This rule is not easy to implement, due to the difficulty of estimating the probabilities $P_{C|X}(c|x)$. Large margin methods, such as boosting, avoid this difficulty by adopting alternative risks. They implement the classifier as

$$F(x) = \begin{cases} 1 & \text{if} \quad f^*(x) < 0 \\ 2 & \text{if} \quad f^*(x) > 0. \end{cases} \tag{2}$$

where $f^*(x) : \mathcal{X} \to \mathbb{R}$ is the *continuous valued predictor*

$$f^*(x) = \arg\min_f R_L(f) \tag{3}$$

that minimizes the risk

$$R_L(f) = E_{X,C}\{L[y^c, f(x)]\} \tag{4}$$

defined by a *loss function* $L[.,.]$ and a set of *class labels* y^c, where y^c is the label of class $c \in \{1, 2\}$. The loss $L[.,.]$ is Bayes consistent if the minimization of (4) results in the Bayes decision rule, i.e. (1) and (2) are equivalent.

To learn the optimal classifier, the risk of (4) is estimated by the empirical risk

$$\overline{R}_L(f) = \frac{1}{n}\sum_{i=1}^{n} L[y^{c_i}, f(x_i)] \tag{5}$$

over a training sample $\mathcal{D} = \{(x_i, c_i)\}_{i=1}^n$. Large margin methods use the *labels* $y^1 = -1$ and $y^2 = 1$ and a Bayes consistent loss function that only depends on the *classification margin* $y^c f(x)$, i.e.

$$L[y^c, f(x)] = L[y^c f(x)]. \tag{6}$$

This guarantees that the classifier has good generalization for finite training samples (Vapnik, 1998). Boosting learns the optimal predictor $f^*(x) : \mathcal{X} \to \mathbb{R}$ as the solution of

$$\begin{cases} \min_{f(x)} & \overline{R}_L(f) \\ s.t & f(x) \in span(\mathcal{H}) \end{cases} \tag{7}$$

where $\overline{R}_L(f)$ is the empirical risk of (5), and $\mathcal{H} = \{h_1(x), \ldots, h_r(x)\}$, a set of weak learners $h_i(x) : \mathcal{X} \to \mathbb{R}$. The optimization is carried out by gradient descent in the functional space $span(\mathcal{H})$ of linear combinations of $h_i(x)$ (Friedman et al., 1998; Mason et al., 2000; Saberian et al., 2010). The extension of binary boosting to the multiclass setting requires multiclass definitions of class labels, predictor, margin, decision rule, loss function and risk minimization procedure.

3.2. Multiclass extensions

The definition of the classification labels as $y^c = \pm 1$ plays a significant role in the binary formulation. One of the difficulties of the multiclass extension is that these labels do not have an obvious generalization. For M-ary classification, $c \in \{1, \ldots, M\}$, each class c must be mapped into a distinct class label $y^c \in \mathcal{Y} = \{y^1, \ldots, y^M\}$. This label can be thought of as a *codeword* that identifies the class. In the binary case, the predictor is a real valued function, i.e. $f(x) \in \mathbb{R}$, and the codewords ± 1 are the two directions on the line. To generalize these concepts to the multiclass setting, we introduce a multi-dimensional predictor $f(x) \in \mathbb{R}^d$ and codewords y^k which are directions in this space

$$y^c \in \mathbb{R}^d, \quad \|y^c\| = 1. \tag{8}$$

Given a multi-dimensional predictor and and a set of multi-dimensional codewords[1], the multiclass margin is defined as follows.

Definition 1 *Let $y^k \in \mathbb{R}^d, k \in \{1, \dots, M\}$, be the set of codewords of an M-ary classifier of predictor $f : \mathcal{X} \to \mathbb{R}^d$. The margin of example x with respect to class k is*

$$\begin{aligned} \mathcal{M}(y^k, f(x)) &= \min_{l \neq k} \left[u^k - u^l \right] && (9) \\ &= u^k - \max_{l \neq k} u^l, && (10) \end{aligned}$$

where

$$u^k = \frac{1}{2} \left\langle y^k, f(x) \right\rangle, \tag{11}$$

is the component of $f(x)$ along codeword y^k and $< ., . >$ is the Euclidean dot-product. The quantity

$$\left[u^k - u^l \right] = \frac{1}{2} \left\langle y^k - y^l, f(x) \right\rangle \tag{12}$$

is the l^{th} margin component of $f(x)$ with respect to class k.

This definition is closely related to previous definitions of multiclass margin in the literature. For example, it generalizes that of (Guermeur, 2007), where the codewords y^k are restricted to binary vectors in the canonical basis of $\mathbb{R}^d$, and is a special case of that of (Allwein et al., 2001), where the dot products $\langle y^k, f(x) \rangle$ are replaced by a generic function of f, x, and k. Furthermore, when $M = 2$ and $y^1 = -y^2 = 1$,

$$\mathcal{M}(y^k, f(x)) = \frac{1}{2}[y^k f(x) - \max_{l \neq k} y^l f(x)] = \frac{1}{2}[y^k f(x) + y^k f(x)] = y^k f(x), \tag{13}$$

and (9) is identical to the classic definition of binary margin. Similarly to the binary case, it is possible to define the margin of a predictor for a particular dataset, as follows.

Definition 2 *The margin of a predictor $f(.)$ with respect to a set of codewords $\mathcal{Y} = \{y^1, \dots, y^M\}$ and examples $\mathcal{D} = \{(x_i, c_i)\}_{i=1}^n$ is*

$$\mathcal{M}_p(\mathcal{D}, \mathcal{Y}, f) = \min_{(x_i, y^{c_i}) \in \mathcal{D}} \mathcal{M}(y^{c_i}, f(x_i)). \tag{14}$$

This can be seen as a measure of the distance between the classification boundary and the point closest to it.

To extent the binary decision rule of (2) to the multiclass case, we start by noting that, for a binary classifier with $y^1 = 1$ and $y^2 = -1$, it can be written as

$$F(x) = \arg \max_{k = \{1,2\}} y^k f^*(x), \tag{15}$$

i.e. the classifier simply chooses the class of largest margin for example x. This has the following straightforward extension.

1. At this point, we assume that a good set of codeword exists and is known. In Section 5 we will discuss how the selection of codewords affects learning performance and procedures for determining the optimal ones.

Definition 3 *Consider an M-ary classification problem with codewords $y^k \in \mathbb{R}^d, k \in \{1, \ldots, M\}$. A maximum margin classifier of predictor $f : \mathcal{X} \to \mathbb{R}^d$ implements the decision rule*

$$F(x) = \arg \max_{k \in \{1,\ldots,M\}} \mathcal{M}(y^k, f(x)). \tag{16}$$

The following result shows that this is equivalent to selecting the class along whose codeword $f(x)$ has the largest component.

Lemma 1 *The decision rule of the maximum margin classifier of (16) is equivalent to*

$$F(x) = \arg \max_{k \in \{1,\ldots,M\}} \left\langle y^k, f(x) \right\rangle. \tag{17}$$

Proof See Appendix A.1. ■

A corollary of this result is that, as is the case for binary classification, an example x of class c is correctly classified by the max margin classifier if and if and only if the example margin of x with respect to class c is positive.

Corollary 1 *Let c be the class of example x and $f(x)$ the predictor of a maximum margin classifier $F(x)$. Then $F(x) = c$ if and only if*

$$\mathcal{M}(y^c, f(x)) > 0. \tag{18}$$

Proof See Appendix A.2 ■

Finally, a maximum margin classifier classifies all examples in a dataset $\mathcal{D}$ correctly if and only if its predictor margin with respect to $\mathcal{D}$ is positive.

Corollary 2 *Let $f(.)$ be the predictor of a maximum margin classifier $F(x)$ and $\mathcal{D}$ a set of examples (x_i, c_i). $f(.)$ classifies all $x_i \in \mathcal{D}$ correctly if and only if*

$$\mathcal{M}_p(\mathcal{D}, f, \mathcal{Y}) > 0. \tag{19}$$

Proof See Appendix A.3. ■

These corollaries extend the equivalent properties of binary large margin predictors to the multiclass case.

4. Multiclass Boosting Algorithms

In this section we introduce two multiclass boosting algorithms. Both are gradient descent procedures for the minimization of a multiclass empirical risk in the functional space of linear combinations of weak learners and can be seen as generalizations of GradientBoost (Mason et al., 2000). We start by extending the definitions of risk and margin loss to the multiclass setting.

Table 1: Binary losses and corresponding γ and ϕ functions.

Name	$\xi(v)$	$\gamma(v)$	$\phi(v)$
Exponential	$\exp(-v)$	v	$\exp(-v)$
Logistic	$\log(1+e^{-2v})$	$\log(1+v)$	$\exp(-2v)$
Savage	$\frac{1}{(1+e^{2v})^2}$	$\left(\frac{v}{1+v}\right)^2$	$\exp(-2v)$

4.1. $\gamma - \phi$ losses

Given a set of codewords $\mathcal{Y}$, the optimal multiclass predictor $f^*(x)$ minimizes the classification risk

$$R_{L_M}(f) \quad = \quad E_{X,C}\{L_M[y^c, f(x)]\}, \tag{20}$$

where c is the class of example x, y^c its codeword and $L_M[y^c, f(x)]$ the loss of prediction $f(x)$. For classifier design, this is approximated by the empirical estimate

$$\overline{R}_{L_M}(f) \quad = \quad \frac{1}{n}\sum_{i=1}^{n} L_M[y^{c_i}, f(x_i)], \tag{21}$$

derived from a training sample $\mathcal{D} = (x_i, c_i)_{i=1}^n$. When the minimization of (21) encourages predictors for which the margin of (14) is large, $L_M[.,.]$ is denoted a *margin loss*. This property guarantees that the optimal predictor has good generalization beyond the training set.

For binary classification, margin losses are monotonically decreasing functions of the margin $y^c f$. The natural multiclass extension would be to consider decreasing functions of the margin, now defined in (9), i.e.

$$L_M[y^c, f(x)] = \chi\left(\min_{l\neq c}(u^c - u^l)\right), \tag{22}$$

with u^k as in (11), for some monotonically decreasing function χ. This, however, is a non-differentiable function of the predictor f. We avoid this difficulty by considering the set of $\gamma - \phi$ losses.

Definition 4 *Let $\mathcal{Y} = \{y^1, \ldots, y^M\} \in \mathbb{R}^d$ be a set of codewords, $f(x) : \mathcal{X} \to \mathbb{R}^d$ a predictor. A $\gamma - \phi$ loss is defined as*

$$L_M^{\gamma-\phi}[y^c, f(x)] = \gamma\left(\sum_{l=1,l\neq c}^{M} \phi\left(u^c - u^l\right)\right), \tag{23}$$

where $\phi : \mathbb{R} \to \mathbb{R}^+$ and $\gamma : \mathbb{R}^+ \to \mathbb{R}^+$ are strictly positive and u^j is defined in (11).

We leave a theoretical discussion of the properties of $\gamma - \phi$ losses to Section 7. For now, we simply point out that this set includes a large family of losses. In fact, for binary classification with the classic labels $y^1 = -y^2 = -1$, $u^1 = -u^2 = -\frac{1}{2}f(x)$ and $L_M^{\gamma-\phi}[y^c, f(x)]$ reduces to

$$\begin{aligned} L_2^{\gamma-\phi}[y^c, f(x)] &= \gamma\left(\sum_{k=1|k\neq c}^{2} \phi\left[u^c - u^k\right]\right) = \gamma\left(\phi\left[\frac{(y^c - (-y^c))f}{2}\right]\right) \\ &= \gamma\left(\phi(y^c f)\right) = \xi(y^c f), \end{aligned} \tag{24}$$

Algorithm 1 CD-MCBoost and GD-MCBoost

```
Input: Number of classes $M$, dimension $d$, codeword set $\mathcal{Y} = \{y^1, \dots, y^M\} \in \mathbb{R}^d$, boosting
iterations $N$ and dataset $\mathcal{D} = \{(x_i, c_i)\}_{i=1}^n$ of examples $x_i$ and class labels $c_i \in \{1, \dots, M\}$.
Initialization: set $t = 0$, and $f^t = 0 \in \mathbb{R}^d$

CD-MCBoost
  while $t < N$ do
    Compute $w_i$ with (29)
    for $j = 1$ to $d$ do
      Find $\hat{g}_j(x)$, $\hat{\alpha}_j$ using (39) and (40)
    end for
    Set $j^* = \arg\min_j \overline{R}[f^t(x) + \hat{\alpha}_j \hat{g}(x)\mathbf{1}_j]$
    Update $f^{t+1}(x) = f^t(x) + \hat{\alpha}_{j^*} \hat{g}_{j^*}(x)\mathbf{1}_j$
    $t = t + 1$
  end while

GD-MCBoost
  while $t < N$ do
    Compute $w_i$ with (29)
    Find $g^*(x)$, $\alpha^*$ using (32) and (33)
    Update $f^{t+1}(x) = f^t(x) + \alpha^* g^*(x)$
    $t = t + 1$
  end while

Output: decision rule: $F(x) = \arg\max_{y^k} \langle f^N(x), y^k \rangle$
```

where $\xi = \gamma \circ \phi$ is a composite function. Table 1 shows that the exponential loss of AdaBoost (Freund and Schapire, 1997), the logistic loss of LogitBoost (Friedman et al., 1998), and the Savage loss of (Masnadi-Shirazi and Vasconcelos, 2008) can all be interpreted as $\gamma - \phi$ losses with different choices of γ and ϕ. It should be noted that these decompositions are not unique. In all cases, ξ could be equally decomposed into $\gamma(v) = v$ and $\phi = \xi$. The key property is that ξ is a monotonically decreasing function. In Section 7 we show that $\gamma - \phi$ losses are margin enforcing whenever γ is strictly increasing and ϕ is strictly decreasing. This implies that $\xi = \gamma \circ \phi$ is decreasing.

4.2. Gradient descent

The first boosting algorithm is a gradient descent procedure to seek the optimal predictor $f^*(x) = [f_1^*(x), \dots, f_d^*(x)]$ of the optimization problem

$$\begin{cases} \min_{f(x)} & \overline{R}_{L_M^{\gamma-\phi}}[f(x)] \\ s.t & f(x) \in span(\overline{\mathcal{H}}), \end{cases} \tag{25}$$

where $\overline{\mathcal{H}} = \{\overline{h}_1(x), \dots, \overline{h}_r(x)\}$ is a set of multivariate weak learners,

$$\overline{h}_i(x) : \mathcal{X} \to \mathbb{R}^d. \tag{26}$$

Let $f^t(x) = [f_1^t(x), \dots, f_d^t(x)]$ be the predictor available after t boosting iterations. At iteration $t+1$, a step is given along the direction $g(x) \in \overline{\mathcal{H}}$ of largest decrease of the risk $\overline{R}_{L_M^{\gamma-\phi}}[f(x)]$. This is determined by the directional derivative of $\overline{R}_{L_M^{\gamma-\phi}}[f(x)]$ along the functional $g : \mathcal{X} \to \mathbb{R}^d$, at point $f(x) = f^t(x)$ (Frigyik et al., 2008),

$$\delta\overline{R}_{L_M^{\gamma-\phi}}[f^t; g] \quad = \quad \left. \frac{\partial \overline{R}_{L_M^{\gamma-\phi}}[f^t + \epsilon g]}{\partial \epsilon} \right|_{\epsilon=0}. \tag{27}$$

As shown in Appendix B.1,

$$-\delta\overline{R}_{L_M^{\gamma-\phi}}[f^t;g] = \frac{1}{2n}\sum_{i=1}^{n} w_i \left\langle g(x_i), y^{c_i} - \sum_{k\neq c_i} y^k \tau_k(x_i,c_i)\right\rangle, \tag{28}$$

where

$$w_i = -\gamma'\left(\sum_{k\neq c_i}\phi\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle\right]\right)\left(\sum_{k\neq c_i}\phi'\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle\right]\right), \tag{29}$$

and

$$\tau_k(x,c) = \frac{\phi'\left[\frac{1}{2}\left\langle f^t(x), y^c - y^k\right\rangle\right]}{\sum_{k\neq c}\phi'\left[\frac{1}{2}\left\langle f^t(x), y^c-y^k\right\rangle\right]}. \tag{30}$$

The direction of steepest descent is the weak learner

$$\begin{aligned} g^*(x) &= \arg\min_{g\in\mathcal{H}} \delta\overline{R}[f^t(x);g(x)] & (31)\\ &= \arg\max_{g\in\mathcal{H}} \sum_{i=1}^{n} w_i \left\langle g(x_i), y^{c_i} - \sum_{k\neq c_i} y^k\tau_k(x_i,c_i)\right\rangle, & (32)\end{aligned}$$

and the optimal step size along this direction

$$\alpha^* = \arg\min_{\alpha\in\mathbb{R}} \overline{R}_{L_M^{\gamma-\phi}}[f^t(x)+\alpha g^*(x)]. \tag{33}$$

Note that α^* may not have a closed form and a line search might be required. The predictor is finally updated according to

$$f^{t+1}(x) = f^t(x) + \alpha^* g^*(x). \tag{34}$$

This procedure is summarized in Algorithm 1-right, and denoted Gradient Descent Multiclass Boosting (GD-MCBoost).

4.3. Coordinate descent

Alternatively, (21) can be minimized by learning a linear combination of scalar functions. In this case, the optimization problem is

$$\left\{\begin{array}{ll} \min_{f_1(x),\ldots,f_d(x)} & \overline{R}_{L_M^{\gamma-\phi}}([f_1(x),\ldots,,f_d(x)]) \\ s.t & f_j(x)\in span(\mathcal{H}) \quad \forall j=1,\ldots,d, \end{array}\right. \tag{35}$$

where $\mathcal{H} = \{h_1(x),\ldots,h_r(x)\}$ is a set of *scalar* weak learners $h_i(x):\mathcal{X}\to\mathbb{R}$. Let $f^t(x)$ be the predictor available after t boosting iterations. At iteration $t+1$ a single component $f_j(x)$ of $f(x)$ is updated with a step in the direction of the scalar functional g that most decreases the risk $\overline{R}_{L_M^{\gamma-\phi}}[f_1^t,\ldots,f_j^t+\alpha_j^* g,\ldots,f_d^t]$. For this, we consider the functional derivative of $\overline{R}_{L_M^{\gamma-\phi}}[f(x)]$

along the direction of the functional $g : \mathcal{X} \to \mathbb{R}$, at point $f(x) = f^t(x)$, with respect to the j^{th} component $f_j(x)$ of $f(x)$

$$\delta \overline{R}_{L_M^{\gamma-\phi}}[f^t; j, g] = \left. \frac{\partial \overline{R}_{L_M^{\gamma-\phi}}[f^t + \epsilon g \mathbf{1}_j]}{\partial \epsilon} \right|_{\epsilon=0}, \tag{36}$$

where $\mathbf{1}_j \in \mathbb{R}^d$ is a vector whose j^{th} element is one and the remaining zero, i.e. $f^t + \epsilon g \mathbf{1}_j = [f_1^t, \ldots, f_j^t + \epsilon g, \ldots, f_d^t]$. As shown in Appendix B.2,

$$-\delta \overline{R}_{L_M^{\gamma-\phi}}[f^t; j, g] \quad = \quad -\frac{1}{2} \sum_{i=1}^{n} w_i g(x_i) \left\langle \mathbf{1}_j, y^{c_i} - \sum_{k \neq c_i} y^k \tau_k(x_i, c_i) \right\rangle, \tag{37}$$

where w_i is as in (29) and $\tau_k(x, c)$ as in (30). The direction of largest descent is then

$$g^*(x) \quad = \quad \arg\min_{g \in \mathcal{H}} \delta \overline{R}_{L_M^{\gamma-\phi}}[f^t; j, g] \tag{38}$$

$$\quad = \quad \arg\max_{g \in \mathcal{H}} \sum_{i=1}^{n} w_i g(x_i) \left\langle \mathbf{1}_j, y^{c_i} - \sum_{k \neq c_i} y^k \tau_k(x_i, c_i) \right\rangle, \tag{39}$$

and the optimal step size along this direction

$$\alpha^* = \arg\min_{\alpha \in \mathbb{R}} \overline{R}[f^t(x) + \alpha g^*(x) \mathbf{1}_j]. \tag{40}$$

Again, this step size may not have a closed form and a line search might be required. The predictor is finally updated with

$$f^{t+1} = f^t(x) + \alpha^* g^*(x) \mathbf{1}_j = [f_1^t, \ldots, f_j^t + \alpha^* g^*, \ldots, f_d^t]. \tag{41}$$

This procedure is summarized in Algorithm 1-left and denoted Coordinate Descent Multiclass Boosting (CD-MCBoost). In each iteration of CD-MCBoost, the best weak learner update is found for each of the d coordinates and the coordinate whose update results in the largest reduction of the risk is then chosen. As can be seen from Algorithms 1, MC-Boost algorithms share the simplicity of implementation (a few lines of code) of binary boosting. The main differences between the two are 1) the use of multi-dimensional predictor and codewords, and 2) the weak learner selection rule. We leave an analysis of the role of codewords to Section 5 and consider the weak learner selection rule next.

4.4. Weak learner selection and boosting weights

The update equations of the two MCBoost algorithms are a natural generalization of the update equations of binary boosting. From (32) and (39), the update equation can, in both cases, be written as

$$g^*(x) = \arg\max_{p} \frac{1}{n} \sum_{i=1}^{n} w_i \hat{\mathcal{M}}_{f^t}(y^{c_i}, g(x_i)) \tag{42}$$

where

$$\hat{\mathcal{M}}_{f^t}(y^c, g(x)) = \frac{1}{2} \left\langle g(x), y^c - \sum_{k \neq c} \tau_k^t(x, c) y^k \right\rangle. \tag{43}$$

The only difference is that, while for GD-MCBoost $g(x)$ is a generic vector $g(x) \in \mathbb{R}^d$, for CD-MCBoost it is a vector of the form $g(x)\mathbf{1}_j$ with $g(x) \in \mathbb{R}$. The two algorithms are thus conceptually equivalent. We next show that they are conceptually equivalent to binary boosting algorithms in the sense that they seek, at each iteration, the weak learner of largest margin on a reweighted training sample. The only difference is that MCBoost implements two weighting mechanisms. The first one, implemented through the weights w_i, emphasizes difficult examples. The second one, implemented through $\hat{\mathcal{M}}_{f_t}$, emphasizes difficult classes.

When the $\gamma - \phi$ loss is margin enforcing, the first weighting mechanism of MCBoost assigns to each example a weight inversely proportional to how well the currently predictor classifies the example. This can be seen by rewriting (29) as

$$w_i \;=\; -\gamma'\left(\sum_{k\neq c_i}\phi(v^k)\right)\left(\sum_{k\neq c_i}\phi'(v^k)\right)\Bigg|_{v^k=\frac{1}{2}\left\langle f^t(x_i),y^{c_i}-y^k\right\rangle} \tag{44}$$

and using Theorem 5 (see Section 7), which states that a $\gamma - \phi$ loss is margin enforcing when $\xi(v) = \gamma \circ \phi(v)$ is a decreasing function. Since $\xi(v) \geq 0$, by definition of $\gamma - \phi$ loss, the derivative of $\xi(v)$ must approach zero for large positive values of v, i.e. $\gamma'(\phi(v))\phi'(v) \to 0$ for large positive v. Hence, w_i is a decreasing function of the margin components $< f^t(x_i), y^{c_i} - y^k >$ and close to zero when the smallest component is positive and large. Since, from (9), this implies a large margin, examples of larger margin under the current predictor $f^t(x)$ will receive smaller weights than examples of smaller margin. As is common in boosting, this focuses the learning resources in the examples that are poorly classified by $f^t(x)$. Hence, the first weighting mechanism is the multiclass generalization of the weighting mechanism of binary boosting.

The second weighting mechanism appears on $\hat{\mathcal{M}}_{f_t}(y^c, g(x))$. By rewriting this quantity as

$$\hat{\mathcal{M}}_{f_t}(y^c, g(x)) = \frac{1}{2}\sum_{k\neq c}\tau_k^t(x,c)\left[\left\langle g(x), y^c - y^k\right\rangle\right] \tag{45}$$

and comparing to (9), it can be seen that $\hat{\mathcal{M}}_{f_t}(y^c, g(x))$ is an estimate of the margin $\mathcal{M}(y^c, g(x))$. Rather than taking the minimum of the margin components $\langle g(x), y^c - y^k\rangle$ over the classes $k \neq c$, this estimate is a weighted average of these components, assigning weight $\tau_k^t(x,c)$ to class k. Hence, $\tau_k^t(x,c)$ is a weighting mechanism that favors some classes over others. To interpret this mechanism it is useful to consider the case where $\phi(v) = e^{-v}$ and (30) reduces to

$$\tau_k^t(x,c) \;=\; \frac{e^{\left[-\frac{1}{2}\left\langle f^t(x),y^c-y^k\right\rangle\right]}}{\sum_{k\neq c} e^{-\left[\frac{1}{2}\left\langle f^t(x),y^c-y^k\right\rangle\right]}} = \frac{e^{\left[\frac{1}{2}\left\langle f^t(x),y^k\right\rangle\right]}}{\sum_{k\neq c} e^{\left[\frac{1}{2}\left\langle f^t(x),y^k\right\rangle\right]}}, \tag{46}$$

i.e. the soft-min of the margin components (and the soft-max of codeword projections). More generally, when $\phi'(v)$ is any decreasing function, $\tau_k^t(x,c)$ is a generalized soft-min operator. It assigns larger weights to classes of smaller margin component and smaller weights to classes of larger margin component. The choice of ϕ function controls the softness of the weight assignment. For example, if $\phi(v) = e^{-\alpha v}$ with $\alpha > 0$, the softness of the assignments is controlled by the choice of α. For large α the weights are harder, i.e. closer to one for the smallest margin component and zero for all others. For smaller α the weights are more uniform. Hence, for this choice of ϕ function, the soft-min of (45) approaches the min of (9) as $\alpha \to \infty$. More generally, up to the approximation of the min by the soft-min operator,

$$\hat{\mathcal{M}}_{f_t}(y^c, g(x)) = \frac{1}{2}\sum_{k\neq c} \tau_k^t(x,c)\left[\left\langle g(x), y^c - y^k \right\rangle\right] \tag{47}$$

$$\approx \frac{1}{2}\langle g(x), y^c - \overline{y}\rangle \tag{48}$$

where

$$\overline{y} = \arg\min_{y_j \neq y^c} \langle f_t(x), y^c - y^j\rangle = \arg\max_{y_j\neq y^c} \langle f_t(x), y^j\rangle. \tag{49}$$

This is the multiclass margin of weak learner $g(x)$ under the alternative margin definition $\hat{\mathcal{M}}_{f_t}(y^c, g(x))$. Comparing to the original definition of (9), which can be written as

$$\mathcal{M}(y^c, g(x)) = \frac{1}{2}\left\langle g(x), y^c - \overline{\overline{y}}\right\rangle \qquad \text{where} \qquad \overline{\overline{y}} = \arg\min_{y_j\neq y^c}\langle g(x), y^c - y_j\rangle, \tag{50}$$

$\hat{\mathcal{M}}_{f^t}(y^c, g(x))$ restricts the margin of $g(x)$ to the worst case codeword $\overline{y}$ for the current predictor $f^t(x)$. The strength of this restriction is determined by the soft-min operator. If $< f^t(x), y^c - \overline{y} >$ is much smaller than $< f^t(x), y^c - y_j >, y^j \neq \overline{y}$, $\tau_k^t(x,c)$ closely approximates the minimum operator. Otherwise, the remaining codewords also contribute to (45). In summary, $\tau_k^t(x,c)$ is a set of class weights that emphasizes classes of small margin for $f^t(x)$. In addition, $\hat{\mathcal{M}}_{f^t}(y^c, g(x))$ is an estimate of the margin of $g(x)$ under the restriction to the most difficult classes for the current predictor $f^t(x)$. When combined with (29) this leads to a weighting mechanism that emphasizes difficult examples (through the weights w) and difficult classes (through the margin $\hat{\mathcal{M}}_{f^t}$).

4.5. GD vs. CD-MCBoost

Since (29) and (30) only depend on the current predictor $f^t(x)$ and not on the weak learner $g(x)$, the computation of weights is identical for GD-MCBoost and CD-MCBoost. The only difference thus resides on the margin estimate of (45). While GD-MCBoost simply chooses the vector weak learner $g(x)$ with (42)-(45), CD-MCBoost uses

$$g^*(x) = \arg\max_{g\in\mathcal{H}} \max_j \frac{1}{n}\sum_{i=1}^n w_i \hat{\mathcal{M}}_{f^t}(y^{c_i}, g(x_i), j). \tag{51}$$

In this case, the approximation of (48) is

$$\hat{\mathcal{M}}_{f^t}(y^c, g(x), j) \approx \frac{g(x)}{2}\left[\langle \mathbf{1}_j, y^c\rangle - \langle \mathbf{1}_j, \overline{y}\rangle\right] \tag{52}$$

$$= g(x)\frac{y_j^c - \overline{y}_j}{2}, \tag{53}$$

and $\mathcal{M}_{f^t}(y^c, g(x), j)$ is *a measure of the margin of $g(x)$ under the j^{th} coordinate of the codewords that determine the margin of the current predictor $f^t(x)$.* Note that, for binary codewords,

$$\mathcal{M}_{f^t}(y^{c_i}, g(x_i), j) \approx \begin{cases} 0, & y_j^{c_i} = \overline{y}_j \\ y_j^{c_i} g(x_i) & y_j^{c_i} \neq \overline{y}_j. \end{cases} \tag{54}$$

This is the standard definition of margin for a scalar weak learner $g(x)$, but discards the points for which $\overline{y}_j = y_j^{c_i}$. In this way, the boosting algorithms modulates the emphasis on poorly classified points in a coordinate-by-coordinate manner.

5. Optimal codewords

So far, we have assumed the existence of a good codeword set $\mathcal{Y} = \{y^1, \ldots, y^M\}$. In this section we study the impact of the codeword set on the performance of MCBoost algorithms.

5.1. The role of codewords

In both MCBoost algorithms, the predictor is initialized with $f(x) = 0 \quad \forall x$, i.e. all examples are mapped to the origin. By minimizing (21), MCBoost then learns a predictor $f(x)$ of maximum margin. From (10), the margin of example x_i is maximal when the projection of $f(x_i)$ is maximal along correct class codeword, y^{c_i}, and minimal along the remaining codewords. If the margin of x_i is positive, it can be increased by increasing the magnitude of the projection of $f(x_i)$ along y^{c_i}. Hence, both MCBoost algorithms seek a predictor $f(x)$ that, for all i, $f(x_i)$ 1) is as aligned as possible with y^{c_i} and 2) has the largest possible magnitude along this direction. Hence, starting from the origin, both MCBoost algorithms *push all example predictions outward, in the direction of the corresponding class codewords*. The effectiveness of this mechanism depends on the structure of the codeword set. For example, if two classes were to share a codeword, it would be impossible to distinguish them with (16) or (17). This implies that some codeword sets result in larger margins for the decision rule of (17) than others.

Intuitively, the margin capacity will increase if the codewords are better separated. This suggests that better performance should be possible for larger values of the codeword dimension d. However, since increasing d increases the complexity of the decision rule, there is a trade-off between margin capacity and implementation complexity. For a fixed d, it remains to determine how to best separate the codewords, so as to maximize the margin capacity, and whether a solution to this problem, i.e. an optimal codeword set, exists.

5.2. Optimal codeword sets

To find the optimal set of codewords, we start by noting that there are several factors that impact the margin, (14), of the learned predictor by MCBoost. The first is the complexity and separability of the underlying problem, i.e., if the problem is not separable, there will never be a classifier with 100% accuracy and the margin of (14) will always be negative. The second is the set of weak learners, i.e. if the set of weak learners is limited then the space of their linear combinations won't be rich enough to have a good classifier. Finally as mentioned above the set of codewords can also impact the margin.

To focus on the impact of the codewords, we assume that 1) the problem is separable and, 2) our weak learners are rich enough and there exists a linear combination of them, f^*, that can separate training examples with 100% accuracy. Under these assumptions, according to (14), the margin of f^* is the positive and we can arbitrarily increase it by increasing norm of f^*. Therefore we can achieve arbitrarily large margin for any codewords set. To resolve this problem we assume that norm of f^* is bounded, e.g., $\|f^*\| = 1$ [2].

Definition 5 *A predictor $f(x)$ is normalized if $||f(x)|| = 1, \forall x$. $\mathcal{F}$ is the set of normalized predictors.*

2. Note that this normalization constraint is only required for finding the optimal set of codewords and is not required in MCBoost algorithm.

Finally note that (9) is invariant to the addition of a constant to all codewords. Also if the margin is positive you can arbitrarily increase it by increasing the codewords norms. To remove these issues, we complement the unit norm constraint of (8) with the constraint that the codewords be centered, i.e., add up to zero.

Definition 6 *A set of vectors $\mathcal{Y} = \{y^1, \ldots, y^M\} \in \mathbb{R}^d$, is denoted a centered (M, d) codeword set if*

$$\textstyle\sum_{k=1}^{M} y^k = 0, \quad \|y^k\| = 1 \quad \forall k = 1, \ldots, M. \tag{55}$$

The set of all centered (M, d) codeword sets is denoted $\mathcal{S}(M, d)$.

In the rest of this section, we assume that codeword sets are always centered, simply referring to a centered (M, d) codeword set as "an (M, d) codeword set,".

Definition 7 *Consider an M-ary classification problem with a codeword set $\mathcal{Y} \in \mathcal{S}(M, d)$. The margin capacity of $\mathcal{Y}$ is*

$$\mathcal{C}[\mathcal{Y}] \quad = \quad \min_{k=1,\ldots,M} \mathcal{M}(y^k, \xi^k), \tag{56}$$

where

$$\xi^k = \arg \max_{||v||=1} \mathcal{M}(y^k, v). \tag{57}$$

is the predictor direction of largest margin for class k.

The margin capacity $\mathcal{C}[\mathcal{Y}]$ can then be interpreted as the maximum margin achievable by any predictor in $\mathcal{F}$ using codewords $\mathcal{Y}$ on any dataset $\mathcal{D}$. Note, from (14), that it is the margin, with respect to $\mathcal{Y}$ and $\mathcal{D}$, of a predictor which maps all examples from class k into the direction ξ^k of largest margin. A large capacity implies that the codeword set is such that a large margin can be achieved for all classes. A small capacity implies that there is at least one class for which the largest achievable margin is small. We define the optimal codeword set as that of largest capacity.

Definition 8 *$\mathcal{Y}^* \in \mathcal{S}(M, d)$ is a codeword set of maximum capacity if*

$$\mathcal{Y}^* = \arg \max_{\mathcal{Y} \in \mathcal{S}(M,d)} \mathcal{C}[\mathcal{Y}]. \tag{58}$$

To formalize this discussion, we start by deriving an upper bound on the margin achievable along any direction of largest margin.

Theorem 1 *Let $\mathcal{Y} \in \mathcal{S}(M, d)$ be a codeword set with directions of largest margin ξ^k, $k \in \{1, \ldots, M\}$ as defined in (57). Then, $\forall k$*

$$\mathcal{M}(y^k, \xi^k) \leq \frac{M}{2(M-1)}, \tag{59}$$

Proof See Appendix C.1. ■

An immediate consequence of (56) and (59) is that, for any codeword set $\mathcal{Y} \in \mathcal{S}(M, d)$,

$$\mathcal{C}[\mathcal{Y}] \leq \frac{M}{2(M-1)}. \tag{60}$$

The right hand side of this inequality is denoted the capacity bound of $\mathcal{S}(M, d)$. The following result characterizes the codeword sets that achieve this bound.

Theorem 2 *A codeword set $\mathcal{Y} \in \mathcal{S}(M, d)$ meets the capacity bound of (60) with equality if and only if its directions of largest margin are $\xi^k = y^k, \forall k$ and*

$$\left\langle y^k, y^l \right\rangle = -\frac{1}{M-1} \quad \forall k, l, k \neq l. \tag{61}$$

Proof See Appendix C.2. ■

The theorem shows that a codeword set meets the capacity bound if and only if it has the directions of largest margin as codewords. We next derive the conditions under which such a set of codewords exists.

Theorem 3 *$\mathcal{S}(M, d)$ contains a set of codewords $\mathcal{Y}^c(M, d)$ that meets the capacity bound if and only if $d \geq M - 1$. In this case, the codewords in $\mathcal{Y}^c(M, d)$ are the vertexes of a regular simplex in $\mathbb{R}^d$.*

Proof See Appendix C.3. ■

The proof of the theorem is constructive, providing a procedure to determine the codeword set $\mathcal{Y}^c(M, d)$. The main results of this section are summarized in the following corollary, which is a straightforward consequence of Theorems 1 and 3.

Corollary 3 *Let $\mathcal{Y}^*$ be a codeword set of maximum capacity in $\mathcal{S}(M, d)$. If $d \geq M - 1$, the codewords $(y*)^k$ of $\mathcal{Y}^*$ are the vertexes of a regular simplex in $\mathbb{R}^d$. In this case, the directions of largest margin are $\xi^k = (y^*)^k$,*

$$\left\langle (y^*)^k, (y^*)^l \right\rangle = -\frac{1}{M-1}, \ \forall k, l \neq k \tag{62}$$

and

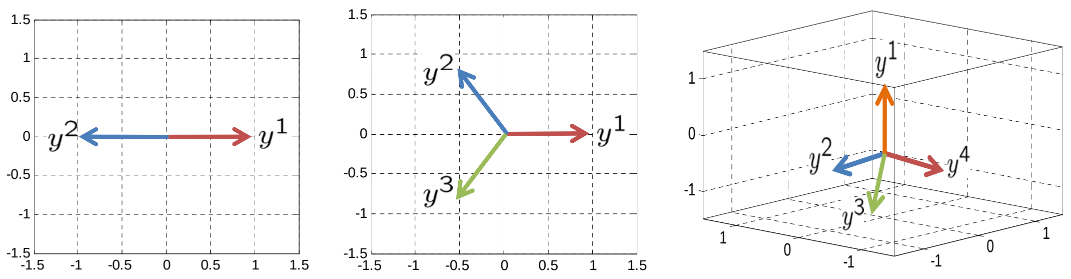
$$\mathcal{C}[\mathcal{Y}^*] = \frac{M}{2(M-1)}. \tag{63}$$

In summary, the capacity bound of (60) is met whenever $d \geq M - 1$. Since this bound only depends on the number of classes M, not in the dimension d, and the decision rule of (17) has linear complexity in d, there is usually no benefit in adopting codewords of dimension larger than $M - 1$. Hence, when the classification problem has no further constraints, it is natural to rely on codeword sets of dimension

$$d = M - 1. \tag{64}$$

Figure 1 presents the optimal codeword sets for various M. Note that in the binary case, $M = 2$, the optimal codewords are the classical $\{+1, -1\}$ labels[3].

3. A set of Matlab scripts that determine the optimal codeword set for any M is available at http://www.svcl.ucsd.edu/publications/conference/2014/icml/ICML_2014_guess_averse_code_data.zip



$(M = 2, d = 1)$ $(M = 3, d = 2)$ $(M = 4, d = 3)$

Figure 1: Codewords of maximal capacity for different values of M.

5.3. Low-dimensional predictors

When $d < M - 1$, there is no guarantee that a codeword set of maximum capacity will achieve the capacity bound. Nevertheless, the choice of $d < M - 1$ can be appealing for applications where it is critical to use low-dimensional predictors. In this case, from (9)-(11), (56), (57), and (58), the search for the codeword set of maximum capacity requires the solution of

$$\mathcal{Y}^* = \arg \max_{\mathcal{Y} \in \mathcal{S}(M,d)} \min_{k=1,\dots,M} \max_{||v||=1} \min_{l \neq k} \left[\left\langle y^k, v \right\rangle - \left\langle y^l, v \right\rangle \right]. \tag{65}$$

This is a non-trivial optimization for which, to the best of our knowledge, there are no efficient algorithms. Hence, it is of interest to consider alternative optimality criteria. One possibility is the max-min codeword distance criterion.

Definition 9 *Let $\mathcal{Y}$ be a codeword set in $\mathcal{S}(M, d)$. The minimum distance of $\mathcal{Y}$ is*

$$d_{min}[\mathcal{Y}] = \min_{k, l \neq k} ||y^k - y^l||^2. \tag{66}$$

$\mathcal{Y}^*$ is a max-min distance codeword set in $\mathcal{S}(M, d)$ if

$$\mathcal{Y}^* = \arg \max_{\mathcal{Y} \in \mathcal{S}(M,d)} d_{min}[\mathcal{Y}]. \tag{67}$$

Max-min distance codeword sets have various appealing properties. First, the maximization of $d_{min}[\mathcal{Y}]$ is intuitive. From (17), the decision rule of the maximum margin classifier is based on the projections $\langle f, y^k \rangle$ of the predictor f along the codewords. If the codewords are similar, the same will hold for the projections, resulting in lower margins. Second, the problem of (67) is equivalent to determining the maximum diameter of M equal circles placed on the surface of the unit sphere without overlap. This is known as the *Tammes* problem (Tammes, 1930). While it does not have closed-form solution, or even a unique solution (any rotation of a valid solution is a valid solution), its numerical solution is much simpler than that of (65). Finally, as shown in the following result, there is a close relationship between the capacity and the minimum distance of any codeword set in $\mathcal{S}(M, d)$.

Lemma 2 *Let $\mathcal{Y}$ be a codeword set in $\mathcal{S}(M, d)$. Then*

$$\frac{1}{4} d_{min}[\mathcal{Y}] \leq \mathcal{C}[\mathcal{Y}] < \frac{1}{2} d_{min}[\mathcal{Y}] \tag{68}$$

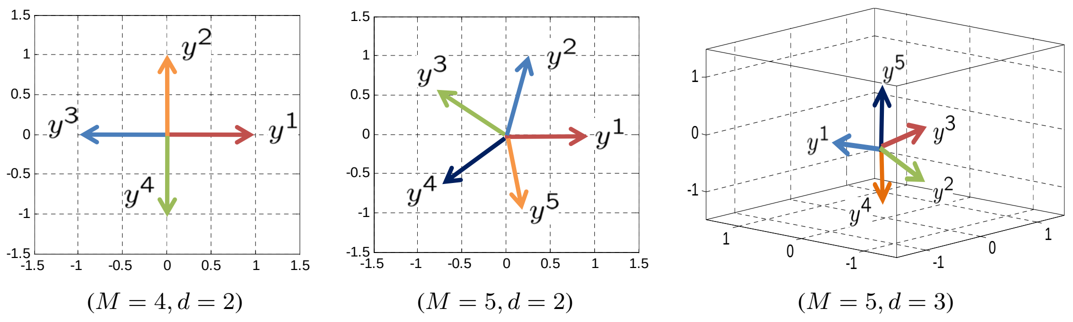


Figure 2: max-min codeword sets for different values of d and M.

and

$$d_{min}[\mathcal{Y}] \leq \frac{2M}{M-1}. \tag{69}$$

Proof See Appendix D.1. ■

The following theorem uses this result to show that the optimal codeword sets under the optimality criteria of Definitions 8 and 9 are identical when $d > M - 1$.

Theorem 4 *Let $\mathcal{Y}^*$ be a codeword set in $\mathcal{S}(M,d)$. If $d \geq M-1$ then $\mathcal{Y}^*$ is a codeword set of maximum capacity, i.e.*

$$\mathcal{C}[\mathcal{Y}^*] = \frac{M}{2(M-1)}, \tag{70}$$

if and only if $\mathcal{Y}^*$ is a codeword set of max-min distance, i.e.

$$d_{min}[\mathcal{Y}^*] = \frac{2M}{M-1}. \tag{71}$$

Proof See Appendix D.2. ■

In summary, in the regime of $d \geq M-1$, the vertexes of a regular simplex in $\mathbb{R}^d$ are both a maximum capacity and a max-min distance codeword sets of $\mathcal{S}(M,d)$. These codeword sets achieve both the capacity and max-min distance bounds of $\mathcal{S}(M,d)$. The main difference between the two optimality criteria is the difficulty of finding an optimal solution for $d < M-1$. While the optimization problem of (65) is difficult, there are many algorithms for the solution of (67). Our implementation uses a solver based on the barrier method (Nocedal and Wright, 1999). Figure 2 presents max-min codewords sets for different values of M and d.

5.4. Complexity vs. capacity

In general, the complexity of an M-ary predictor increases with the number M of classes. For example, the "one vs all" architecture requires the evaluation of one predictor per class, i.e. has complexity $O(M)$, while the "all pairs" architecture has complexity $O(M^2)$. While the decision rule $F(x)$ of (17) always has complexity $O(M)$, this consists of computing M d-dimensional dot-products and can be usually be performed efficiently for large values of M, as long as d is small.

Hence, for MCBoost, predictor complexity is mostly a function of the predictor dimension d, not the number of classes. In fact, MCBoost can solve any M-ary classification problem with a predictor of dimension as low as $d = 2$. Obviously, this carries a penalty, by limiting the margin capacity of the codeword set. While, as discussed in Theorem 3, the margin capacity can be achieved for any dimension $d \geq M - 1$, smaller dimensions typically lead to reduced classification accuracy. Hence, the dimension d can be seen as a parameter that controls the trade-off between predictor complexity and accuracy. Note that complexity does not need to refer simply to the number of computations. For $d < M$, the act of learning the predictor $f(x)$ can be seen as a form of discriminant dimensionality reduction. This could be of interest for applications such as hashing (Datar et al., 2004; Torralba et al., 2008; Kulis and Darrell, 2009; Gong et al., 2012; Liu et al., 2012), where it is important to represent information from many classes by short codes. In this case, MCBoost can directly learn a discriminant *representation* of low complexity.

6. Weak Learners

In this section, we investigate the role of weak learners in MCBoost. We consider the questions of which weak learner families can be boosted, the role of classification versus real-valued learners, and how these connect to interpretations of boosting as voting rules. While we limit the discussion to GD-MCBoost, for brevity, all considerations extend to CD-MCBoost.

6.1. Boostability

The selection of weak learner family for a boosting algorithm is usually guided by a boostability condition. A weak learner family is boostable if it guarantees zero training error after a sufficient number of iterations of the boosting algorithm. For binary classification, this holds if the family contains a weak learner with better than random performance, i.e. error rate smaller than 50%, for any weight distribution over training examples. Existing multiclass boosting algorithms are usually designed for specific classes of weak learners, e.g regression or decision stumps, and require this boostability condition. However, while trivial for binary classification, the "less than chance error" condition can be non-trivial to prove for multiclass problems, where chance level error is $M^{-1} \times 100\%$. For example, (Mukherjee and Schapire, 2013) used a fairly sophisticated game theoretic analysis to derive boostability conditions in terms of the error rate of a set of weak learners.

Unlike most of these algorithms, MCBoost has a fairly straightforward boostability condition for any convex $\gamma - \phi$ loss. It suffices that, at any iteration t, there exists a weak learner $h \in \mathcal{H}$ for which the directional derivative of (28) satisfies

$$-\delta \overline{R}_{L_M^{\gamma-\phi}}[f^t; h] > \delta, \tag{72}$$

for some fixed $\delta > 0$. In this case, it is always possible to reduce the classification risk by adding h to the weak learner ensemble and, by convexity, the algorithm will converge to the minimum training error, which is zero for separable data. In summary, a set of weak learners $\mathcal{H}$ is boostable by MCBoost if, under any weight distribution over training examples, it contains a member h for which (72) holds with $\delta > 0$.

Similarly to binary boosting, this condition can be expressed in terms of the performance of a set of random weak learners. Let $\rho(x)$ be a weak learner with random output, i.e.

$$P(\rho(x) = y^k) = \frac{1}{M} \quad \forall x, k = 1, \ldots, M. \tag{73}$$

It follows that, for any given x_i, $\epsilon_i = \rho(x_i)$ is a sample from a random vector ϵ uniformly distributed over the codeword set $\{y^1, \dots, y^M\}$. The derivative of (28)

$$-\delta \overline{R}_{L_M^{\gamma-\phi}}[f^t;\rho] \;=\; \frac{1}{2n}\sum_{i=1}^{n} w_i \left\langle \epsilon_i, y^{c_i} - \sum_{k\neq c_i} y^k \tau_k(x_i,c_i) \right\rangle \tag{74}$$

has expected value

$$E_\epsilon\left\{-\delta \overline{R}_{L_M^{\gamma-\phi}}[f^t;\rho]\right\} \;=\; \frac{1}{2n}\sum_{i=1}^{n} w_i \left\langle E_\epsilon\left\{\epsilon_i\right\}, y^{c_i} - \sum_{k\neq c_i} y^k \tau_k(x_i,c_i) \right\rangle, \tag{75}$$

Since

$$E_\epsilon\left\{\epsilon_i\right\} = \sum_{k=1}^{M} \frac{y^k}{M} = 0, \tag{76}$$

where the second equality follows from (55), it follows that

$$E_\epsilon\left\{-\delta \overline{R}_{L_M^{\gamma-\phi}}[f^t;\rho]\right\} = 0. \tag{77}$$

Hence, the boostability condition of (72) can be written as

$$-\delta \overline{R}_{L_M^{\gamma-\phi}}[f^t;h] > E_\epsilon\left\{-\delta \overline{R}_{L_M^{\gamma-\phi}}[f^t;\rho]\right\} + \delta, \tag{78}$$

with $\delta > 0$. A weak learner family is boostable if, for any weight distribution, it contains a weak learner h that beats the random learner ρ. Hence, as is the case for binary boosting, MCBoost can boost any set of any weak learners h whose performance is slightly better than random, i.e. has "less than $1/M \times 100\%$ error," for any weight distribution. Note that this result is valid for any set $\mathcal{H}$ of weak learners. In fact, the weak learners do not even have to be classifiers. CD-Boost can be applied to any set of real-valued functions $h : \mathcal{X} \to \mathbb{R}$ and GD-MCBoost to any set of multi-dimensional functions $\overline{h} : \mathcal{X} \to \mathbb{R}^d$.

6.2. Classification weak learners and voting rules

Obviously, MCBoost can be implemented with classification weak learners. In this case, the algorithm can be presented as a voting rule, as in the classical presentation of AdaBoost. In appendix E.1 we derive the GD-MCBoost equations for the case where the weak learner space $\overline{\mathcal{H}}$ is a space of multiclass classifiers, i.e. $h : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{Y} = \{y^1, \dots, y^M\}$ is the codeword set of the classification problem. We show that, when this is the case, the learned predictor has the form

$$f(x) \;=\; \sum_k y^k \sum_{t|g_t(x)=y^k} \alpha_t \tag{79}$$

if $\mathcal{Y}$ is a codeword set of maximum capacity in $\mathcal{S}(M,d)$ and

$$f_k(x) \;=\; \sum_{t|g_t(x)=\mathbf{1}_k} \alpha_t. \tag{80}$$

if $\mathcal{Y}$ is a set of canonical codewords $\mathbf{1}_j$. In either case, denoting $I(g(x) = y^k)\alpha$ as a vote of strength α from weak learner $g(x)$ for class k, $f(x)$ can be seen as a weighted average of the class codewords, where each class is weighted by the strength of its weak learner votes.

It is also shown that, although these predictors have different codeword projections, namely

$$\left\langle f(x), y^j \right\rangle = \sum_{t|g_t(x)=y^j} \alpha_t - \frac{1}{M-1} \sum_{k \neq j} \sum_{t|g_t(x)=y^k} \alpha_t, \tag{81}$$

for the maximum capacity codeword set and

$$\left\langle f(x), y^j \right\rangle = \sum_{t|g_t(x)=\mathbf{1}_j} \alpha_t \tag{82}$$

for canonical codewords, the margin components are similar up to a rescaling, namely

$$\left\langle f(x), y^c - y^k \right\rangle = \frac{M}{M-1} \left(\sum_{t|g_t(x)=y^c} \alpha_t - \sum_{t|g_t(x)=y^k} \alpha_t \right), \tag{83}$$

for a maximum capacity set and

$$\left\langle f(x), y^c - y^k \right\rangle = \sum_{t|g_t(x)=\mathbf{1}_c} \alpha_t - \sum_{t|g_t(x)=\mathbf{1}_k} \alpha_t \tag{84}$$

for a canonical set. In summary, for maximum capacity codewords, the projection of $f(x)$ along codeword y^j is the difference between the strength of the weak learner votes for class j and the average vote strength for the remaining classes. Hence, the projection is large if and only if the class receives votes of strength well above average. On the other hand, for canonical codewords, the projection reduces to the strength of the weak learner votes for class j. The margin components are always larger under the maximum capacity codeword set, but the difference vanishes for large M. Nevertheless, due to the similar form of the margin components, the decision rule of (17) is the same in the two cases and consists of choosing the class of strongest vote

$$F(x) = \arg\max_k \sum_{t|g_t(x)=y^k} \alpha_t. \tag{85}$$

The margin condition of (18) thus holds if and only if this is the true class y^c.

With regards to learning, it is shown in appendix E.1 that the steepest descent direction of (32) is also

$$g^*(x) = \arg\min_{g \in \mathcal{H}} \sum_{i|g(x_i) \neq y^{c_i}} w_i \left(1 + \sum_{j \neq c_i} \tau_j(x_i, c_i) I(g(x_i) = y^j) \right), \tag{86}$$

in the two cases. The weak learner selection rule reduces to selecting the classifier $g(x)$ of lowest error rate on a dataset where each point x_i is reweighted proportionally to how poorly it is classified by the current predictor $f^t(x)$ (which is captured by w_i) and the agreement between the (incorrect) class to which it is assigned by $g(x)$ and $f^t(x)$ (captured by $\sum_{j \neq c_i} \tau_j(x_i, c_i) I(g(x_i) = y^j)$). Note that, even though the weak learner selection rule is identical for the two codeword sets, the algorithms are not the same, since the weights w_i and $\tau_j(x_i, c_i)$ are determined by the margin components, according to (29)-(30),

6.3. Classification vs. real valued learners

Finally, we consider the question of whether anything is lost by using a set of classification weak learners. Consider the implementation of MCBoost in a space $\mathcal{H}$ of continuous weak learners. Let the direction of steepest descent of the risk at a given iteration be $h(x) \in \mathcal{H}$. Then, it is possible to define a classification weak learner

$$g(x) = y^k, \qquad k = \arg\max_j < h(x), y^j > . \tag{87}$$

It follows that

$$\begin{aligned} \int \langle g(x), h(x) \rangle \, dx &= \int \left\langle y^{\arg\max_j < h(x), y^j >}, h(x) \right\rangle dx & (88) \\ &= \int \arg\max_j \left\langle h(x), y^j \right\rangle dx. & (89) \end{aligned}$$

Since $\arg\max_j \left\langle h(x), y^j \right\rangle \geq \left\langle \frac{y^j + y^l}{2}, y^j \right\rangle, \forall l \neq j$, and

$$\left\langle \frac{y^j + y^l}{2}, y^j \right\rangle = \begin{cases} \frac{1}{2} & \text{if } \mathcal{Y} \text{ is a canonical codeword set} \\ \frac{1}{2} - \frac{1}{2(M-1)} & \text{if } \mathcal{Y} \text{ is a maximum capacity codeword set,} \end{cases} \tag{90}$$

the inequality $\arg\max_j \left\langle h(x), y^j \right\rangle > 0$ holds for both classes of codewords, whenever $M > 2$. Hence,

$$\int \langle g(x), h(x) \rangle \, dx > 0 \tag{91}$$

and $g(x)$ is a descent direction, albeit not the steepest descent direction, in $\mathcal{H}$. It follows that the family of weak learners $g(x)$ is boostable. In summary, for either canonical or maximum capacity codeword sets, whenever there is a boostable set $\mathcal{H}$ of generic weak learners, there is also a boostable family of classification weak learners, given by (87). The only penalty in adopting the latter is that, because MCBoost is no longer a steepest descent procedure in $\mathcal{H}$, its convergence can be slower.

7. Loss functions

So far, we have considered a generic $\gamma - \phi$ loss function. In this section, we derive conditions under which $\gamma - \phi$ losses belong to some families with desirable properties for classification, such as margin or proper losses. We then design specific $\gamma - \phi$ losses functions in these families and use them to implement different MCBoost algorithms.

7.1. Margin losses

An important question is under which conditions $\gamma - \phi$ losses are margin enforcing, i.e. when the minimization of the empirical risk of a $\gamma - \phi$ loss encourages predictors of large margin $\mathcal{M}_p(\mathcal{D}, f, \mathcal{Y})$. The following result shows that this holds whenever γ is strictly increasing and ϕ is strictly positive and decreasing.

Theorem 5 *Consider a training set $\mathcal{D}$, a codeword set $\mathcal{Y} = \{y^1, \dots, y^M\}$, a predictor $f(x)$. Let $\overline{R}_{L_M^{\gamma-\phi}}(f)$ be the empirical risk of (21) for a $\gamma - \phi$ loss as in (23), such that γ is strictly increasing and ϕ is strictly positive and decreasing. Then*

$$\overline{R}_{L_M^{\gamma-\phi}}(f) \quad > \quad \frac{1}{n}\gamma\left(\phi[\mathcal{M}_p(\mathcal{D}, f, \mathcal{Y})]\right), \tag{92}$$

where $\mathcal{M}_p(\mathcal{D}, f, \mathcal{Y})$ is the margin of predictor f - as defined in (14).

Proof See Appendix F.1. ■

Since γ is strictly increasing and ϕ is strictly decreasing, $\xi = \gamma \circ \phi$ is strictly decreasing. It follows from (92) that the minimization of $\overline{R}_{L_M^{\gamma-\phi}}(f)$ forces large margins and the $\gamma - \phi$ loss is margin enforcing. Note that the minimization of a decreasing ξ is similar to the definition of margin loss for binary classification, where $\xi = \gamma \circ \phi$ is always decreasing. The main difference is that the components γ and ϕ now have different roles. While ϕ operates on each margin component, γ is applied to the aggregate of all the components. Hence, it no longer suffices to specify ξ, but γ and ϕ have to be specified explicitly.

7.2. Proper losses

A loss $L_M[.,.]$ is said to be *proper* if it is possible to recover the class posterior probabilities $P_{C|X}(c|x), \quad C = 1, \dots, M$, for all x from any predictor $f^*(x)$ optimal under that loss. This property is important for applications that require a confidence score for the classification. It is natural to ask under which conditions are $\gamma - \phi$ losses proper. For simplicity, we adopt the notation

$$\eta_k(x) = P_{C|X}(k|x), \tag{93}$$

for the posterior class probabilities and represent a codeword set $\mathcal{Y} = \{y^1, \dots, y^M\} \in \mathbb{R}^d$ by a code matrix $\mathbf{Y} \in \mathbb{R}^{d \times M}$, whose columns are the codewords in $\mathcal{Y}$, i.e.

$$\mathbf{Y} = [y^1, \dots, y^M]. \tag{94}$$

We also resort to the well known result that, to minimize (20), it suffices to determine the predictor $f^*(x)$ of minimum conditional risk

$$R_{L_M^{\gamma-\phi}}(f|x) \quad = \quad E_{C|X}\{L_M^{\gamma-\phi}[y^c, f(x)]|x\}, \tag{95}$$

for all x. The following theorem characterizes the set of minimizers of the conditional risk of a $\gamma - \phi$ loss.

Theorem 6 *Let $\mathcal{Y} = \{y^1, \dots, y^M\} \in \mathbb{R}^d$ be a codeword set, $f(x) : \mathcal{X} \to \mathbb{R}^d$ a predictor, $\eta_k(x)$ the posterior probabilities of (93), and $R_{L_M^{\gamma-\phi}}(f|x)$ the conditional risk of (95) when $L_M^{\gamma-\phi}[.,.]$ is a $\gamma - \phi$ loss with differentiable γ, ϕ. In this case any minimizer, f^*, of (95) is a solution of*

$$\mathbf{Y}\mathbf{Q}_{f^*}^{\phi}\mathbf{\Gamma}_{f^*}^{\gamma-\phi}\eta = 0, \tag{96}$$

where $\mathbf{Y} \in \mathbb{R}^{d \times M}$ is the code matrix of (94), $\eta = [\eta_1, \dots, \eta_M]^T$ the vector of posterior probabilities, $\mathbf{Q}_f^{\phi}, \mathbf{\Gamma}_f^{\gamma-\phi} \in \mathbb{R}^{M \times M}$ with

$$\mathbf{Q}_f^{\phi}(k, l) = \begin{cases} -\phi'(u^l - u^k) & k \neq l \\ \sum_{j \neq k} \phi'(u^k - u^j) & k = l, \end{cases} \tag{97}$$

$$\mathbf{\Gamma}_f^{\gamma-\phi}(k,l) = \left\{ \begin{array}{ll} \gamma'\left(\sum_{l\neq k}\phi(u^k-u^l)\right) & k=l \\ 0 & k\neq l, \end{array}\right. \tag{98}$$

u^j as defined in (11) and we have omitted the dependency of all terms on x for notational simplicity.

Proof See Appendix G.1. ■

Since (96) holds for any minimizer f^* of (95), the posterior probability vector η can be uniquely recovered from f^* only when the null space of $\mathbf{Y}\mathbf{Q}_{f^*}^{\phi}\mathbf{\Gamma}_{f^*}^{\gamma-\phi}$ contains a single probability vector. The following lemma provides sufficient conditions for this to holds. In the following results, we use $|\mathcal{A}|$ to denote the dimensionality of a space $\mathcal{A}$, and $Rank(\mathbf{A}), Null(\mathbf{A})$, and $Range(\mathbf{A})$ to denote the rank, null space, and column space of matrix $\mathbf{A}$, respectively.

Lemma 3 *Let $\mathcal{Y} = \{y^1,\ldots,y^M\} \in \mathbb{R}^d$ be a codeword set, $\mathbf{Y}$ the corresponding matrix of (94) and $\mathbf{Q}_f^{\phi}$, $\mathbf{\Gamma}_f^{\gamma-\phi}$ as defined in (97), (98) respectively. The following statements hold.*

1. *If $\gamma : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ is strictly monotonic then*
$$Null(\mathbf{Y}\mathbf{Q}_f^{\phi}\mathbf{\Gamma}_f^{\gamma-\phi}) = Null(\mathbf{Y}\mathbf{Q}_f^{\phi}) \quad \textit{and} \quad Null(\mathbf{Q}_f^{\phi}\mathbf{\Gamma}_f^{\gamma-\phi}) = Null(\mathbf{Q}_f^{\phi}).$$
2. *If $Rank(\mathbf{Y}) \leq M-2$ then $|Null(\mathbf{Y}\mathbf{Q}_f^{\phi})| \geq 2$*
3. *If $Rank(\mathbf{Y}) \geq M-1$ then $Null(\mathbf{Y}\mathbf{Q}_f^{\phi}) = Null(\mathbf{Q}_f^{\phi})$.*
4. *If $\phi : \mathbb{R} \rightarrow \mathbb{R}^+$ is strictly monotonic then $|Null(\mathbf{Q}_f^{\phi})| = 1$.*

Proof See Appendix G.2. ■

The lemma shows that, if $Rank(\mathbf{Y}) \geq M-1$ and ϕ, γ are strictly monotonic, the set of vectors η for which (96) holds is a one-dimensional space. It remains to verify if this space contains a probability vector, i.e. a vector of non-negative entries that sum to one. The following lemma shows that this is indeed the case.

Lemma 4 *Let $\eta = [\eta_1,\ldots,\eta_M]^T$ and $\mathbf{Q}_f^{\phi}, \mathbf{\Gamma}_f^{\gamma-\phi}$ be as defined in (97) and (98), respectively. If $\mathbf{Q}_f^{\phi}\mathbf{\Gamma}_f^{\gamma-\phi}\eta = 0$ then*

$$\eta_k = \frac{\sum_{l=1}^M \eta_l \pi_l \phi'(u^l-u^k)}{\pi_k \sum_{l=1}^M \phi'(u^k-u^l)} \quad \forall k, \tag{99}$$

where

$$\pi_j = \gamma'\left(\sum_{l\neq j}\phi(u^j-u^l)\right) \ \forall j. \tag{100}$$

Furthermore, if $\gamma : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ and $\phi : \mathbb{R} \rightarrow \mathbb{R}^+$ are strictly monotonic and $\eta \neq 0$ then all η_k have the same sign.

Proof See Appendix G.3. ■

Together, Lemmas 3 and 4 state that, for any code matrix $\mathbf{Y}$ such that $Rank(\mathbf{Y}) \geq M-1$ and any pair of strictly monotonic functions ϕ, γ, the space of η that satisfy (96) is one dimensional and composed by vectors that satisfy (99). Since, for any vector $\eta \neq \mathbf{0}$ in this subspace, all η_k have the same sign, $\sum_k \eta_k \neq 0$ and thus $\bar{\eta} = \frac{\eta}{\sum_k \eta_k}$ is a probability vector. Under these conditions, the class posterior probabilities of (96) can be recovered from the optimal predictor f^* as follows.

Theorem 7 *Let $\mathcal{Y} = \{y^1, \ldots, y^M\} \in \mathbb{R}^d$ be a codeword set of matrix $\mathbf{Y}$, $L_M^{\gamma-\phi}[.,.]$ a $\gamma - \phi$ loss, $R_{L_M^{\gamma-\phi}}(f|x)$ the conditional risk of (95), and η the vector of the posterior probabilities of (93). If $Rank(\mathbf{Y}) \geq M-1$ and (ϕ, γ) is a pair of strictly monotonic differentiable functions, then η can be recovered from any minimizer, f^*, of $R_{L_M^{\gamma-\phi}}(f|x)$ with*

$$\eta = \left(\mathbf{\Omega}_f^{\gamma-\phi}\right)^{-1} \mathbf{e}_1, \tag{101}$$

where $\mathbf{e}_1 = [1, 0, \ldots, 0]^T \in \mathbb{R}^M$ and

$$\mathbf{\Omega}_f^{\gamma-\phi}(k,l) = \begin{cases} 1 & k=1 \\ -\pi_l \phi'(u^l - u^k) & k \neq l \\ \pi_k \sum_{j \neq k} \phi'(u^k - u^j) & k = l, \end{cases} \tag{102}$$

with u^k, π_k as defined in (11) and (100), respectively.

Proof See Appendix G.4 ■

Theorems 3 and 4 show that, for an M-ary classification problem, a codeword space of dimension $M-1$ is the smallest to contain a set of codewords that achieve either the margin capacity or maximum distance bounds. In this case, the optimal codeword set consists of the M vertexes of the regular simplex in $\mathbb{R}^{M-1}$ and can be obtained with the procedure of (Coxeter, 1973). The associated matrix $\mathbf{Y}$ has rank $M-1$ and the codewords are the directions of largest margin for each of the M classes. Theorem 7 shows that the combination of the $M-1$ dimensional simplex codewords with strictly monotonic γ, ϕ functions is a sufficient condition for a proper $\gamma - \phi$ loss. In this case, the posterior probability vector η can be recovered from any minimizer f^* of the risk, using (101).

7.3. Relations to binary classification

A third question of interest is how the results above generalize previously known properties of binary losses. For binary classification, where $M = 2$ and $y^1 = -y^2 = 1$, it follows from (102) that

$$\begin{aligned} \mathbf{\Omega}_f^{\gamma-\phi} &= \begin{bmatrix} 1 & 1 \\ -\phi'(u^1 - u^2)\gamma'(\phi(u^1-u^2)) & \phi'(u^2-u^1)\gamma'(\phi(u^2-u^1)), \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 \\ -\xi'(f) & \xi'(-f), \end{bmatrix}, \end{aligned} \tag{103}$$

Table 2: Loss, link, and inverse link, of various boosting algorithms.

Algorithm	$\xi(v)$	$\zeta_{\gamma-\phi}(\eta)$	$\zeta^{-1}_{\gamma-\phi}(v)$
AdaBoost (Freund and Schapire, 1997)	$\exp(-v)$	$\frac{1}{2}\log\frac{\eta}{1-\eta}$	$\frac{e^{2v}}{1+e^{2v}}$
LogitBoost (Friedman et al., 1998)	$\log(1+e^{-2v})$	$\frac{1}{2}\log\frac{\eta}{1-\eta}$	$\frac{e^{2v}}{1+e^{2v}}$
SavageBoost (Masnadi-Shirazi and Vasconcelos, 2008)	$\frac{1}{(1+e^{2v})^2}$	$\frac{1}{2}\log\frac{\eta}{1-\eta}$	$\frac{e^{2v}}{1+e^{2v}}$

where $f = u^1 - u^2$ and $\xi = \gamma \circ \phi$. Using (103), the solution of (101) is then

$$\eta_1 = \frac{\xi'(-f)}{\xi'(f) + \xi'(-f)} \tag{104}$$

$$\eta_2 = \frac{\xi'(f)}{\xi'(f) + \xi'(-f)}, \tag{105}$$

where we omitted the dependence on x. Defining $\eta = \eta_1$, the two equations can be rewritten as

$$\eta\xi'(f) = (1-\eta)\xi'(-f). \tag{106}$$

This is a popular equation in the binary classification literature, where it is known as a sufficient condition for f to minimize the classification risk associated with the binary margin loss $\xi(.)$ (Zhang, 2004; Reid and Williamson, 2010). It is usually possible to solve (106) for the optimal f. This is given by

$$f = \zeta_{\gamma-\phi}(\eta), \tag{107}$$

for some function $\zeta_{\gamma-\phi}$, which is denoted the link function of the loss ξ. The link function plays an important role in the recovery of the posterior probabilities η_k because, for a proper loss ϕ, $\zeta_{\gamma-\phi}$ is invertible and

$$\eta = \zeta^{-1}_{\gamma-\phi}(f). \tag{108}$$

Hence, probabilities $\eta(x)$ can be recovered by simply feeding the classifier predictions $f(x)$ through the inverse of the link. Table 2 lists the loss, link, and inverse link of the margin losses that underlie various popular boosting algorithms.

For M-ary classification, (99) generalizes the inverse link relationship of (108). The main difficulty of the M-ary extension is that the inverse link does not always have closed-form. In fact, while the exact decomposition of the loss ξ into the components γ and ϕ does not affect the link for binary classification, this is not the case for M-ary classification. Consider, for example, the exponential and logistic losses of Table 1. For all these losses, $\phi(v) = e^{\alpha v}$ with $\alpha \in \mathbb{R}^-$. Consider next a multiclass $\gamma - \phi$ loss whose $\phi(.)$ function is in this family. From (99)

$$\begin{aligned} \eta_k &= \frac{\sum_{l=1}^M \eta_l \pi_l \alpha e^{\alpha(u^l - u^k)}}{\pi_k \sum_{l=1}^M \alpha e^{\alpha(u^k - u^l)}} \\ &= \frac{e^{-\alpha u^k} \sum_{l=1}^M \eta_l \pi_l e^{\alpha u^l}}{\pi_k e^{\alpha u^k} \sum_{l=1}^M e^{-\alpha u^l}} \\ &\propto e^{-2\alpha u^k - \log \pi_k}, \end{aligned} \tag{109}$$

Table 3: Expressions of γ, ϕ, η_k, and π_k for multiclass generalizations of the binary $\gamma - \phi$ losses of Table 1. In all cases, $p_k = \frac{e^{2u^k}}{\sum_i e^{2u^i}}$.

name	multiclass loss	$\gamma(v)$	$\phi(v)$	η_k	π_k
Exponential	$\sum_{l \neq c} e^{-(u^c - u^l)}$	v	e^{-v}	p_k	1
Logistic	$\log\left(1 + \sum_{l \neq c} e^{-2(u^c - u^l)}\right)$	$\log(1+v)$	e^{-2v}	p_k	p_k
Savage	$\left(1 - \frac{1}{1+\sum_{l \neq c} e^{-2(u^c-u^l)}}\right)^2$	$\left(\frac{v}{1+v}\right)^2$	e^{-2v}	$\frac{1}{1+\sum_{j \neq k} \frac{1-p_k}{1-p_j}}$	$2p_k^2(1-p_k)$

where, from (100),

$$\pi_k = \gamma' \left(e^{\alpha u^k} \sum_{l=1}^{M} e^{-\alpha u^l} - 1 \right). \tag{110}$$

Since η is a probability vector, it follows that

$$\eta_k = \frac{e^{-2\alpha u^k - \log \pi_k}}{\sum_i e^{-2\alpha u^i - \log \pi_i}}. \tag{111}$$

Hence, the inverse link of any $\gamma - \phi$ loss of exponential ϕ function is the softmax mapping $\rho : \mathbb{R}^M \to [0,1]^M$,

$$\rho_k(v) = \frac{e^{v_k}}{\sum_i e^{v_i}}, \tag{112}$$

of the class scores u^i rescaled according to

$$\begin{aligned} \eta &= \rho(-2\alpha u - \kappa(u)) && (113)\\ u &= \frac{1}{2}\left[< y^1, f >, \ldots, < y^M, f >\right]^T && (114)\\ \kappa(u) &= \left[\log \pi_1(u), \ldots, \log \pi_M(u)\right]^T. && (115) \end{aligned}$$

In summary, whenever the recovery of the posterior probability vector η is of interest, the choice of $\phi(v) = e^{\alpha v}$ is desirable, as it guarantees a closed form-solution for the inverse link. The choice of $\gamma(.)$ function simply determines the rescaling $\kappa(u)$ of the scores, through (110). This suggests a very natural generalization of the binary losses of Table 1, which is given in Table 3. Note that, in all cases, $\gamma \circ \phi$ is a decreasing function. Hence, by Theorem 5, all losses are margin enforcing. The table also presents the values of π_k and η_k for all losses. Since in all cases the probabilities are

$$\eta_k = \frac{e^{2u^k}}{\sum_i e^{2u^i}}, \tag{116}$$

i.e. directly proportional to the class scores u^k, the decision rule of (17) is identical to the Bayes decision rule (1).

Table 4: Expressions of the MCBoost weights of (29) and (30) for the multiclass losses of of Table 3.

name	multiclass loss	w_i	$\tau_k(x_i, c_i)$
Exponential	$\sum_{l \neq c} e^{-(u^c - u^l)}$	$\sum_{k \neq c_i} e^{-\frac{1}{2}\left\langle f^t(x_i), y^{c_i} - y^k \right\rangle}$	$\frac{e^{-\frac{1}{2}\left\langle f^t(x_i), y^{c_i} - y^k \right\rangle}}{\sum_{k \neq c_i} e^{-\frac{1}{2}\langle f^t(x_i), y^{c_i} - y^k \rangle}}$
Logistic	$\log\left(1 + \sum_{l \neq c} e^{-2(u^c - u^l)}\right)$	$\frac{2\sum_{k \neq c_i} e^{-\left\langle f^t(x_i), y^{c_i} - y^k \right\rangle}}{1 + \sum_{k \neq c_i} e^{-\langle f^t(x_i), y^{c_i} - y^k \rangle}}$	$\frac{e^{-\left\langle f^t(x_i), y^{c_i} - y^k \right\rangle}}{\sum_{k \neq c_i} e^{-\langle f^t(x_i), y^{c_i} - y^k \rangle}}$
Savage	$\left(1 - \frac{1}{1 + \sum_{l \neq c} e^{-2(u^c - u^l)}}\right)^2$	$\frac{4\left(\sum_{k \neq c_i} e^{-\left\langle f^t(x_i), y^{c_i} - y^k \right\rangle}\right)^2}{\left(1 + \sum_{k \neq c_i} e^{-\langle f^t(x_i), y^{c_i} - y^k \rangle}\right)^3}$	$\frac{e^{-\left\langle f^t(x_i), y^{c_i} - y^k \right\rangle}}{\sum_{k \neq c_i} e^{-\langle f^t(x_i), y^{c_i} - y^k \rangle}}$

7.4. Multiclass boosting algorithms

A final important question is how the discussion above can be used to derive multiclass boosting algorithms that generalize the ones available in the literature for binary classification. In Section 4.4, we saw that the update equations of the two MCBoost algorithms are given by (42)-(44) and (30). In the binary setting ($M = 2$) with $y^1 = 1, y^2 = -1$, (30) reduces to $\tau_k^t(x, c) = 1$ for $k \neq c$ and

$$\hat{\mathcal{M}}_{f^t}(y^c, g(x)) \quad = \quad \frac{1}{2}\left[g(x)y^c - g(x)(-y^c)\right] = y^c g(x), \tag{117}$$

reduces to the standard definition of margin. Furthermore, from (44)

$$w_i \quad = \quad -\gamma'\left(\phi(y^{c_i} f^t(x_i))\right)\left(\phi'(y^{c_i} f^t(x_i))\right) = -\xi'(y^{c_i} f^t(x_i)), \tag{118}$$

where $\xi = \gamma \circ \phi$ is a decreasing function. This leads to the classical weak learner selection rule

$$g^*(x) \quad = \quad \arg\max_{g \in \overline{\mathcal{H}}} \frac{1}{n} \sum_{i=1}^{n} w_i \mathcal{M}_{f^t}(y^{c_i}, g(x_i)) \tag{119}$$

$$\quad = \quad \arg\max_{g \in \overline{\mathcal{H}}} \frac{1}{n} \sum_{i=1}^{n} y^{c_i} w_i g(x_i), \tag{120}$$

of binary boosting algorithms. Hence, when $M = 2$, both GD-MCBoost and CD-MCBoost reduce to binary boosting with loss $\xi(v) = \gamma(\phi(v))$. It follows that, by using the losses of Table 2, MCBoost can implement all popular binary boosting algorithms, including AdaBoost (Freund and Schapire, 1997), LogitBoost (Friedman et al., 1998) and SavageBoost (Masnadi-Shirazi and Vasconcelos, 2008). Furthermore, because these are special cases of the losses of Table 3, MCBoost algorithms derived with these losses can be seen as multiclass extensions of AdaBoost and LogitBoost. Table 4 summarizes the expressions of the weights w_i and $\tau_k(x_i, c_i)$ that define these algorithms.

8. Comparison to previous learning algorithms

In this section we compare MCBoost learning algorithms in the literature. We start by comparing MCBoost with the current multiclass boosting algorithms and then compare it with the kernel-SVM.

Table 5: Comparison of multiclass boosting algorithms.

Algorithm	Optimization	Codewords	Weak Learners	Loss
OVA	CD	canonical	any	L_{ova} (225)
AdaBoost.M1	GD	canonical	classification	$L_{Ada.M1}$ (238)
SAMME	GD	max-capacity	classification	L_{SAMME} (240)
AdaBoost.MH	GD	canonical	product	L_{ova} (225)
AdaBoost.ECC	CD	binary	any	$\gamma(v) = v, \phi(v) = e^{-2v}$
LogitBoost	CD	canonical	regression	Logistic $\gamma - \phi$
CD-MCBoost	CD	max-capacity	any	any $\gamma - \phi$
AdaBoost.M2/MR	GD	canonical	product	Exponential $\gamma - \phi$
AdaBoost.MM	GD	canonical	classification	$\gamma(v) = v, \phi(v) = e^{-2v}$
GD-MCBoost	GD	max-capacity	any	any $\gamma - \phi$

8.1. Comparison to boosting algorithms

Many of the current multiclass boosting algorithms are special instances of MCBoost using different, and sometimes sub-optimal, codewords, weak learners, loss functions, or optimization strategy (CD v. GD). We discuss these relationships in detail in Appendix H, where we show that these multiclass boosting algorithms can be cast as special cases of, or approximations to, MCBoost. The differences between them are summarized in Table 5. To simplify comparisons, the table is organized in three sections. The top section contains algorithms that do not use $\gamma - \phi$ losses. These are used by the algorithms in the middle and bottom sections, which differ in optimization strategy. It can be seen that all algorithms are either gradient descent (GD) or coordinate descent (CD) procedures. They also differ in the codeword set used to label the different classes. While some use a maximum capacity codeword set, most rely on canonical codewords, or codeword choices (ECOC) of margin capacity below the bound of (59). With regards to weak learners, most algorithms adopt classification learners $h : \mathcal{X} \to \mathcal{Y} = \{y^1, \dots, y^M\}$, but there also have been proposals for linear regressors (multiclass LogitBoost), or the product learners $h : \mathcal{X} \times \{1, \dots, M\} \to \mathbb{R}$ of AdaBoost.M2, AdaBoost.MR, and AdaBoost.MH. Finally, while most algorithms use a special form of the $\gamma - \phi$ loss of (23), a few (OVA, AdaBoost.MH) rely on the OVA loss of (225), while others (Ada.M1,SAMME) minimize the exponential losses of (238) and (240).

As discussed in the previous sections, many of the configurations in the table can result in sub-optimal performance. In the bottom two sections, it is worth noting the codewords of AdaBoost.ECC, which do not offer guarantees in terms of margin capacity, and the product weak learners of AdaBoost.M2 and AdaBoost.MR, which can lead to slow convergence. These problems can be compounded by the choice of loss function. In fact, a consistent experimental observation is that all algorithms derived from losses outside of the $\gamma - \phi$ family (top section of the table) tend to produce sub-optimal classifiers. In Section H.1.1, we show that the combination of the L_{ova} loss and canonical codewords, used by OVA and AdaBoost.MH, encourages the independent learning of coordinate classifiers. This frequently leads to uncalibrated scores across classes and poor classification performance. On the other hand, the losses

$$L^{\alpha}[y^c, f(x)] = e^{-\alpha\langle y^c, f(x)\rangle}, \tag{121}$$

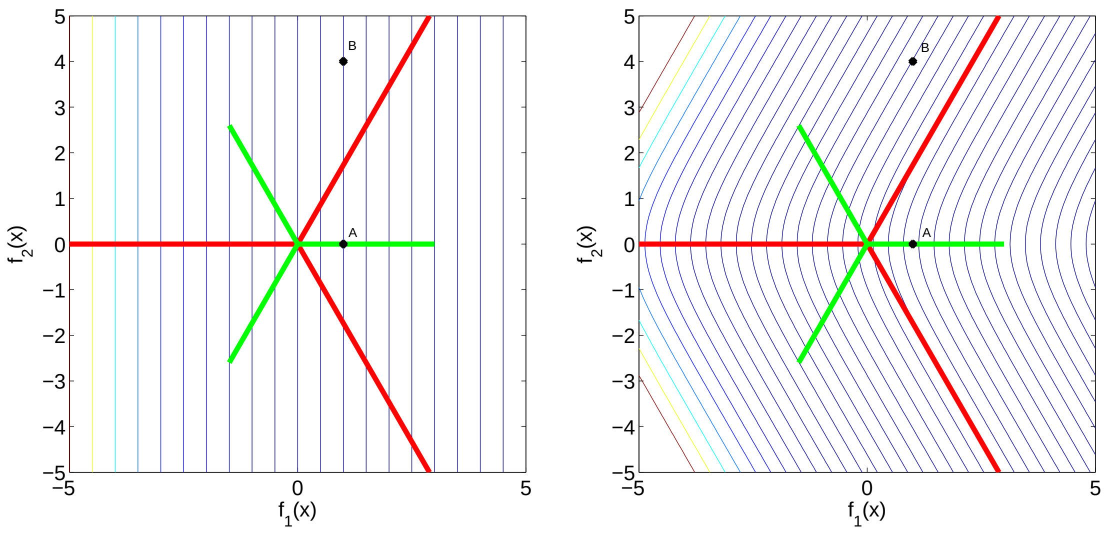


Figure 3: Loss functions $L^\alpha[y^1, f(x)]$ for a classification problem with $M = 3$, $d = 2$, and the max-capacity codewords of Figure 1 (center), where $y^1 = (0, 1)$. Left: exponential loss of (125). Right: $\gamma - \phi$ loss of (126). In both plots $\alpha = 1$, codewords are shown in green and class boundaries in red.

used by AdaBoost.M1 ($\alpha = 1$) and SAMME ($\alpha = 1/M$) are vaguely similar to a $\gamma - \alpha$ loss with $\gamma(v) = v$ and $\phi(v) = e^{-\alpha v}$

$$L^{\gamma-\phi}[y^c, f(x)] = \sum_{l \neq c} e^{-\alpha\left(\langle y^c, f(x)\rangle - \langle y^l, f(x)\rangle\right)}. \tag{122}$$

However, these losses do not consider the pair-wise difference between class scores. Hence, while it has been shown that, asymptotically, the minimizer of the risk defined by these losses implements Bayes rule (Zhu et al., 2009), these losses do not encourage large values of the margin of (9).

Consider, for example, the case where $M = 3$, $d = 2$, and the codeword set is the max-capacity set shown in the center of Figure 1, i.e.

$$y^1 = (1, 0) \quad y^2 = (-1/2, \sqrt{3}/2) \quad y^3 = (-1/2, -\sqrt{3}/2). \tag{123}$$

In this case, as shown in Figure 3, the class boundaries are the lines of direction

$$l^{12} = (1/2, \sqrt{3}/2) \quad l^{13} = (1/2, -\sqrt{3}/2) \quad l^{23} = (-1, 0), \tag{124}$$

where l^{ij} is the direction of the boundary between classes i and j. The loss associated with y^1 is then

$$L^\alpha[y^1, f(x)] = e^{-\alpha f_1(x)} \tag{125}$$

for the exponential loss and

$$L^{\gamma-\phi}[y^1, f(x)] = e^{-\alpha\left\langle d^{12}, f(x)\right\rangle} + e^{-\alpha\left\langle d^{13}, f(x)\right\rangle} \tag{126}$$

with

$$d^{12} = y^1 - y^2 = (3/2, -\sqrt{3}/2) \quad d^{13} = y^1 - y^3 = (3/2, \sqrt{3}/2). \tag{127}$$

The contour plots of these functions, shown in Figure 3, illustrate the limitations of the exponential loss. While this loss has contours orthogonal to y^1, it does not take into account the remaining codewords. In result, its contours cross the class boundaries. This implies that many classification errors (e.g. the prediction output marked as 'B') receive the *same* penalty as correct decisions (prediction marked as 'A'), which is clearly sub-optimal for classification. More generally, the loss functions of (121) produce poor classifiers whenever codewords are not aligned with the coordinate axes. This predicts poor performance for methods like SAMME. The empirical studies of (Mukherjee and Schapire, 2013) and Section 9 confirm this prediction. On the other hand, by aligning the loss contours with the decision boundaries, the $\gamma - \phi$ loss eliminates this problem. Note how it assigns a larger penalty to 'B' than 'A'.

These observations support the claim that the design of a multiclass boosting algorithm requires the careful *joint* selection of optimization strategy, codewords, weak learners, and loss function. For example, AdaBoost.M1 combines the loss of (121) and (axis-aligned) canonical codewords. However, the poor choice of step size along the gradient direction, see (236), leads to a boostability condition that demands weak learners with "better than 50%" error. Since this is unfeasible for most problems, the algorithm frequently stops prematurely. SAMME corrects this problem, but through the introduction of a set of max-capacity codewords that are no-longer axis-aligned. The mismatch between the loss of (121) and these codewords in turn compromises it performance. MCBoost eliminates these problems, supporting any max-capacity codeword set and any family of weak learners, as long as a $\gamma - \phi$ loss is used. In fact, MCBoost even establishes a connection between boosting and other machine learning algorithms, such as support vector machines (SVMs).

8.2. MCBoost and the SVM

Given a training set $\mathcal{D} = \{(x_i, c_i)\}_{i=1}^n$ kernel support vector machines (K-SVMs) first map training examples x_i into a reproducing kernel Hilbert space (RKHS), using the mapping $\mathbf{\Phi}(x) = \mathbf{K}(x, .)$ induced by the kernel $\mathbf{K}(., .)$. The linear classifier that maximizes the margin in this space is then learned, so as to separate examples of different classes (Vapnik, 1998). As is the case for boosting, multiclass K-SVMs can be learned in multiple ways, including the reduction to several binary sub-problems, as in the 'one vs all', 'all pairs' or 'ECOC' methods (Scholkopf et al., 1995; Blanz et al., 1996; Allwein et al., 2001). These are usually sub-optimal. In this work, we consider direct multiclass methods (Vapnik, 1998; Weston and Watkins, 1999; Hsu and Lin, 2002; Crammer and Singer, 2002a).

Under the direct formulation, the multiclass K-SVM solves an M-ary classification problem by learning the M linear discriminants $\mathbf{w}_l^*$, $l = 1, \dots, M$ that maximize a risk based on the multiclass margin

$$\mathcal{M}_{K-SVM}(x_i, \mathbf{w}_{c_i}) = \frac{1}{2}\left[\langle \Phi(x_i), \mathbf{w}_{c_i}\rangle - \max_{l \neq c_i} \langle \Phi(x_i), \mathbf{w}_l\rangle\right], \qquad (128)$$

so as to implement the decision rule

$$F_{SVM}(x) = \arg\max_{l=1,\dots,M} \langle \Phi(x), \mathbf{w}_l^*\rangle . \qquad (129)$$

The comparison of (128) with (9) and (129) with (17) reveals a parallel with MCBoost, where

- the kernel mapping $\Phi(x)$ of the SVM plays the role of the predictor $f(x)$ of boosting,

- the linear discriminants $\mathbf{w}_l$ of the SVM are equivalent to the class codewords y^l of boosting.

This parallel can be taken even further, by considering the SVM optimization. While several formulations have been proposed to maximize (128) (Vapnik, 1998; Weston and Watkins, 1999; Hsu and Lin, 2002; Crammer and Singer, 2002a), the most popular (Vapnik, 1998) is to find the optimal linear discriminants $\mathbf{w}_l^*$ by solving

$$\begin{cases} \min_{\mathbf{w}_l,\ldots,\mathbf{w}_M} & \sum_{l=1}^M \|\mathbf{w}_l\|_2^2 + C\sum_i \xi_i \\ s.t & \langle\Phi(x_i),\mathbf{w}_{c_i}\rangle - \langle\Phi(x_i),\mathbf{w}_l\rangle \geq 1-\xi_i \\ & \forall(x_i,c_i)\in\mathcal{D}, l\neq c_i, \\ & \xi_i \geq 0 \quad \forall i. \end{cases} \tag{130}$$

Rewriting the constraints as

$$\xi_i \geq \max[0, (1-\langle\Phi(x_i),\mathbf{w}_{c_i}\rangle - \max_{l\neq c_i}\langle\Phi(x_i),\mathbf{w}_l\rangle)],$$

and using the fact that the objective function is monotonically increasing in ξ_i, this is identical to solving the problem

$$\min_{\mathbf{w}_l,\ldots,\mathbf{w}_M} \quad \overline{R}_{svm} + \lambda\sum_{l=1}^M \|\mathbf{w}_l\|_2^2, \tag{131}$$

where

$$\overline{R}_{svm} = \sum_i \left\lfloor \langle\Phi(x_i),\mathbf{w}_{c_i}\rangle - \max_{l\neq c_i}\langle\Phi(x_i),\mathbf{w}_l\rangle \right\rfloor_+, \tag{132}$$

$\lfloor x\rfloor_+ = \max(0, 1-x)$ is the hinge loss, and $\lambda = 1/C$. Hence, the SVM minimizes the risk $\overline{R}_{svm}$ subject to a regularization constraint on $\sum_l \|\mathbf{w}_l\|_2^2$. Note that, under the parallel above, $\overline{R}_{svm}$ is the risk $\overline{R}_{L_M}(\mathbf{w})$ of (21) using the margin loss $L_M[\mathbf{w}_c, \Phi(x)]$ of (22), and the hinge loss as χ function, i.e. $\chi(v) = \lfloor v\rfloor_+$. It follows that the parallel is exact, i.e. the methods are mathematically identical, up to the replacement of $L_M[.,.]$ by the $\gamma-\phi$ loss of (23).

In summary, boosting and SVMs optimize very similar costs, but with different optimization strategies. In SVM learning, examples are first mapped into the RKHS, using a *pre-defined* transformation based on the kernel. The SVM then finds a set of linear discriminants that maximizes the margin of examples in this RKHS. On the other hand, boosting pre-defines the linear discriminants as the codewords y^c. It then finds a mapping of the examples, the predictor $f(x)$, that maximizes the margin with respect to these codewords. Both the SVM discriminants and the boosting predictor are found by solving optimization problems that are equivalent, up to slight differences in loss function ($L_M[.,.]$ vs. $\gamma-\phi$ loss) and regularization. While the SVM uses an explicit regularization constraint on the norm of the discriminants $\mathbf{w}_l$, boosting uses two implicit constraints on the complexity of the predictor $f(x)$: it 1) restricts $f(x)$ to a linear combination of *weak* learners and 2) limits the number of weak learners in this combination to the number of boosting iterations.

While this shows that there is a great degree of similarity between the two approaches, there is also a significant difference. This follows from the fact that it is much easier to design a good set of codewords than a good kernel. As we have shown in Section 5, under the boosting strategy it is possible to first design a set of codewords of maximum margin capacity and then find the predictor that maximizes the margin with respect to these codewords. This is unlike the SVM strategy, where it is quite difficult to determine an optimal kernel. In practice, the SVM kernel is usually chosen by cross-validation within a relatively small set of functions. In summary, while the SVM exploits a

Table 6: Characteristics of the used UCI datasets

#Id	Data Name	#Training	#Testing	#Attributes	# Classes
1	*Landsat Satellite*	$4,435$	$2,000$	36	6
2	*Letter*	$16,000$	$4,000$	16	26
3	*Pen Digit*	$7,494$	$3,498$	16	10
4	*Poker*	$25,010$	$1,000$	10	10
5	*Optical Digit*	$3,823$	$1,797$	64	10
6	*Shuttle*	$43,500$	$14,500$	9	7
7	*Isolet*	$6,238$	$1,559$	617	26
8	*Vehicle*	692	154	18	4
9	*Vertebral*	239	71	6	3
10	*Image Segmentation*	210	$2,100$	19	7
11	*Ecoli*	258	78	7	8
12	*Breast Tissue*	81	25	9	6

single tool for margin maximization (finding the optimal linear discriminants), boosting decouples the margin maximization into a sequence of two optimizations 1) *determination of a codeword set* (linear discriminants) of maximum margin capacity and 2) *learning of an example mapping* (predictor) of maximum margin with respect to these codewords.

9. Evaluation

Several experiments were conducted to evaluate the performance of MCBoost algorithms. Unless otherwise noted, these were implemented with $d = M - 1$, a max capacity codeword set composed by the vertexes of a regular simplex in $\mathbb{R}^{M-1}$, and the exponential $\gamma - \phi$ loss of Table 3 ($\gamma(v) = v$ and $\phi(v) = e^{-v}$)[4].

9.1. Data

The first experiment was based on a synthetic dataset, for which the optimal decision rule is known. This is a three class problem, with two-dimensional Gaussian classes of means

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ -1 \end{bmatrix} \tag{133}$$

and covariances

$$\begin{bmatrix} 1 & 0.5 \\ 0.5 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 0.3 \\ 0.3 & 1 \end{bmatrix}, \begin{bmatrix} 0.4 & 0.1 \\ 0.1 & 0.8 \end{bmatrix} \tag{134}$$

respectively. Training and test sets of $1,000$ examples each were randomly sampled and the Bayes rule computed in closed form (Duda et al., 2001). This rule achieved an error rate of 11.67% in the training and 11.13% in the test set, suggesting a Bayes error of about 11%. The remaining experiments were based on the twelve UCI datasets of Table 6. For these, we used the training/test set split provided by the dataset, whenever possible. If a split was unavailable, 20% of the examples were randomly selected for testing.

4. A Matlab implementation of CD-MCBoost and GD-MCBoost is available from http://www.svcl.ucsd.edu/publications/conference/2014/icml/ICML_2014_guess_averse_code_data.zip. A C++ implementation of GD-MCBoost for deep convolutional neural networks (Moghimi et al., 2016) integrated with the CAFFE library (Jia et al., 2014) is available from https://github.com/mmoghimi/BoostCNN.

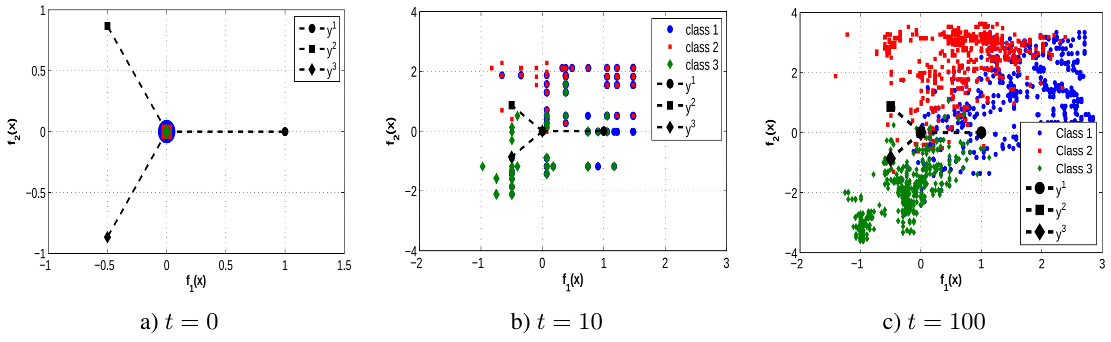


Figure 4: Classifier predictions $f^t(x_i)$ of CD-MCBoost, on the test set, after $t = 0, 10, 100$ boosting iterations.

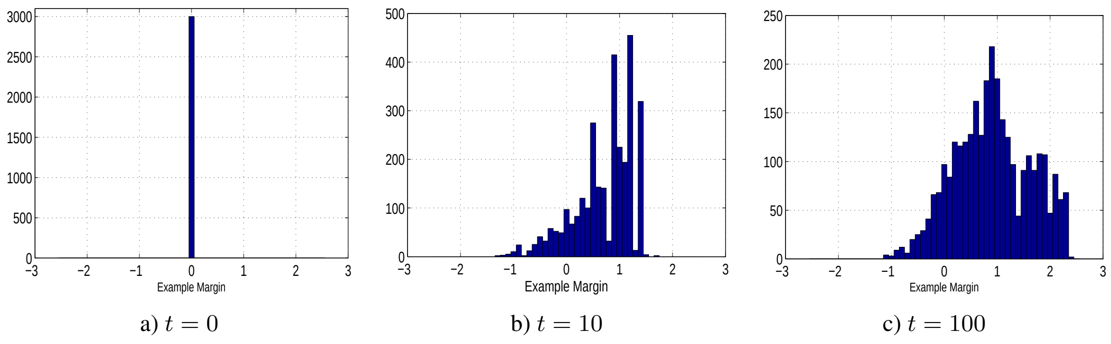


Figure 5: Histogram of the example margins $\mathcal{M}(y^{c_i}, f^t(x_i))$ using CD-MCBoost prediction, on the test set, after $t = 0, 10, 100$ boosting iterations.

9.2. Synthetic data

We start with the synthetic dataset. A classifier was learned with CD-MCBoost, using decision stumps as weak learners. Figures 4 and 5 show the predictions[5] $f^t(x_i)$ and histograms of the associated example margins $\mathcal{M}(y^{c_i}, f^t(x_i))$ for all test examples, after $t = 0, 10$, and 100 iterations, respectively. While the predictor initially maps all examples to the origin, $f^0(x_i) = [0, 0]^T \, \forall x_i$, CD-MCBoost produces predictions that are gradually more aligned with the class codewords, shown as dashed lines in Figure 4. Note how the example predictions, whose color reflects their class label, are pushed out, filling up the space that surrounds the codewords of the corresponding classes. This alignment of $f^t(x_i)$ with y^{c_i} guarantees that $f^t(x_i)$ will have a larger dot product with y^{c_i} than with the remaining codewords, leading to the correct classification of x_i by the decision rule of (17). Similarly, since $f^0(x_i) = [0, 0]^T \, \forall x_i$, all examples have zero margin in the initial iterations. However, as shown by the histograms of Figure 5, the margins $\mathcal{M}(y^{c_i}, f^t(x_i))$ increase with the boosting iteration t. Note that, as the example predictions expand in Figure 4, both the percentage of examples with positive margin and the strength of their margins increase in Figure 5. This implies that, in addition to a decrease in error rate, the decision rule generalizes better. Finally, the test set error rate was 11.30% after 100 iterations, which is very close to the estimates of the Bayes error rate of 11%.

5. We emphasize the fact that these are plots of $f^t(x) \in \mathbb{R}^2$, not $x \in \mathbb{R}^2$.

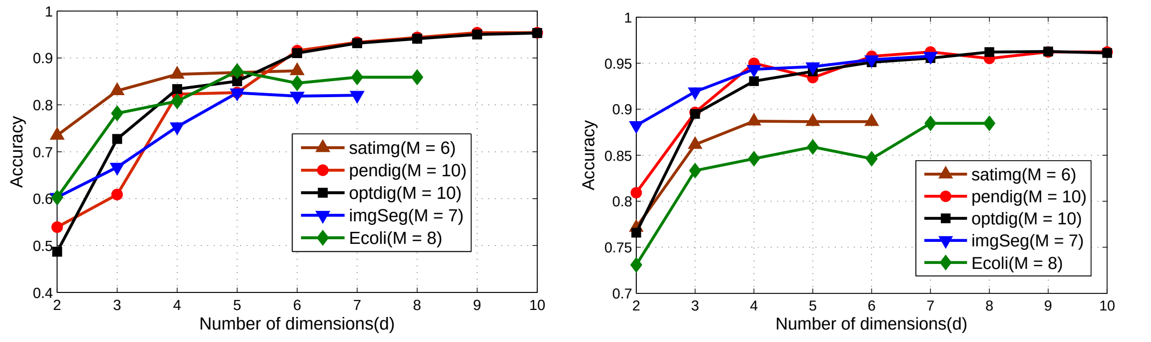


Figure 6: Effect of the dimension of codewords on the accuracy of the trained classifiers. Left: CD-MCBoost with decision stumps. Right: GD-MCBoost with trees of depth 2.

9.3. Effect of codeword dimension, d

The theoretical analysis of Section 5 suggests that there is no benefit to using a predictor dimensionality d larger than $M-1$, where M is the number of classes. Since larger dimensionality increases computation and over-fitting potential, this implies that $d = M-1$ is the best selection for the predictor dimension. In this section, we report on an empirical study of the effect of this parameter on the performance of MCBoost.

We considered 5 UCI datasets, *Landsat, Pen Digit, Optical Digit, Image Segmentation, and Ecoli,* with $M \in [6, 10]$ classes. For each dataset, we learned classifiers with predictors of dimensionality d ranging from 2 to M. A max-min codeword set was used for each value of d. Note that these codewords are identical to the optimal set for $d = M-1$. All classifiers were learned with 200 iterations of MCBoost. We considered both CD-MCBoost and GD-MCBoost, using decision stumps and trees of depth 2 as weak learners, respectively. Figure 6 shows the accuracy of the learned classifiers as a function of the dimension d, for each of the five datasets.

A number of conclusions can be drawn from the figure. First, increasing d usually leads to a better classifier, for both CD-MCBoost and GD-MCBoost. This is not surprising, since for larger d it is possible to increase the minimum mutual distance d_{min} between codewords. Since, from Lemma 2, this yields a larger margin capacity, MCBoost can produce predictors of larger margin. This explains the performance improvement. Second, since increasing d increases classification complexity, the curves of Figure 6 also depict the trade-off between accuracy and complexity of the MCBoost classifiers. Third, all curves saturate as D approaches the value of $M-1$, confirming the theoretical prediction for the lack of benefit of higher dimensions. This follows from the fact that the margin capacity does not increase beyond $d = M-1$. Finally, it is interesting that most curves saturate well below this point. Note, for example, that the GD-MCBoost performance saturates at a dimension between 50% and 70% of the number M of classes, for most datasets. This implies that it is possible to perform “dimensionality reduction on the output space,” without a noticeable cost in classification accuracy. In this sense, the predictor $f(x)$ could be interpreted as an extractor of low dimensional features without loss of discriminant power. This could be interesting for applications where classification has to be performed under complexity constraints.

Table 7: Classification accuracy (%) of coordinate descent boosting algorithms with decision stump weak learners. CD-MCBoost implemented with the exponential $\gamma - \phi$ loss.

Data set id	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12
AdaBoost-OVA	88.0	83.3	95.0	50.9	**95.0**	79.2	**95.3**	68.2	83.1	67.8	85.9	32.0
AdaBoost-ECC	**88.2**	77.8	94.9	**51.6**	94.8	79.2	93.6	69.5	83.1	**82.7**	84.6	**44.0**
CD-MCBoost	87.0	**84.0**	**95.4**	51.5	**95.0**	79.2	95.0	**71.4**	83.1	81.4	**85.9**	**44.0**

Table 8: Classification accuracy (%) of coordinate descent boosting algorithms with regression weak learners. CD-MCBoost implemented with the logistic $\gamma - \phi$ loss.

Data set id	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12
LogitBoost	84.4	77.5	92.9	50.8	89.6	94.7	**96.3**	81.2	87.3	**93.0**	85.9	**76.0**
CD-MCBoost	**84.5**	77.5	92.9	50.8	**95.1**	**96.7**	95.6	81.2	87.3	92.3	**87.2**	72.0

9.4. Comparison to other multiclass Boosting methods

We finish with an experimental comparison of MCBoost to previous boosting algorithms, which complements the theoretical analysis of Section 8.

9.4.1. CD-MCBOOST

We started by comparing the coordinate descent (CD) methods of Table 5, namely OVA, AdaBoost.ECC (Guruswami and Sahai, 1999) and multiclass LogitBoost (Friedman et al., 1998), to CD-MCBoost. The first experiment compared CD-MCBoost to OVA and AdaBoost.ECC, using decision stumps as weak learners. As shown in Table 7, CD-MCBoost had better accuracy than OVA in six and equal accuracy in four of the twelve datasets. Compared to AdaBoost.ECC, it had better performance in six and same performance in three of the twelve datasets. Comparing all methods, CD-MCBoost produced the most accurate classifier for six of twelve datasets, the next best method was AdaBoost-ECC with 4 wins and AdaBoost-OVA had the worst performance (1 win). These results are explained by the sub-optimal codewords of AdaBoost.ECC and the loss function of OVA, which encourages independent learning of coordinate predictors.

We next compared CD-MCBoost to multiclass LogitBoost. To guarantee a fair comparison, CD-MCBoost was implemented with regression weak learners and the logistic $\gamma - \phi$ loss of Table 3. As shown in Table 8, CD-MCBoost had better accuracy in four datasets, LogitBoost was better in three and the results were identical in five datasets. The similar performance of the two classifiers is not surprising since, as discussed in Section H.1.3, they are conceptually equivalent. The only difference between LogitBoost and this implementation of CD-MCBoost is in the use of different codewords. However, since LogitBoost uses the canonical basis in $\mathbb{R}^M$ and CD-MCBoost the vertexes of a simplex in $\mathbb{R}^{M-1}$, the theory predicts identical margin capacity for the two sets. These results confirm the lack of benefit in increasing d beyond $M - 1$.

Table 9: Classification accuracy (%) of gradient descent boosting algorithms with depth 2 tree weak learners. GD-MCBoost implemented with the exponential $\gamma - \phi$ loss.

Data set id	$\#1$	$\#2$	$\#3$	$\#4$	$\#5$	$\#6$	$\#7$	$\#8$	$\#9$	$\#10$	$\#11$	$\#12$
Ada.M1	72.9	–	–	–	–	96.4	–	59.1	**88.7**	–	73.1	60.0
SAMME	82.0	53.3	90.4	52.0	91.8	99.7	85.7	76.6	84.5	93.1	87.2	**80.0**
AdaBoost.MM	89.0	64.4	95.2	59.2	95.4	99.8	91.2	81.2	87.3	94.9	84.6	68.0
GD-MCBoost	**89.1**	**85.2**	**96.5**	**60.8**	**96.8**	**99.9**	**95.2**	**81.8**	87.3	**95.3**	**89.7**	76.0

9.4.2. GD-MCBOOST

The final set of experiments compared the performance of previous gradient descent methods to GD-MCBoost. In this experiment, we considered AdaBoost.M1 (Freund and Schapire, 1996), AdaBoost.MM (Mukherjee and Schapire, 2013) and AdaBoost.SAMME (Zhu et al., 2009), using decision trees of depth 2 as weak learners. These weak learners were designed with a greedy procedure, so as to 1) minimize the weighted error rate of (235) for AdaBoost.M1 (Freund and Schapire, 1996) and AdaBoost.SAMME (Zhu et al., 2009), 2) minimize the classification cost of (266) for AdaBoost.MM (Mukherjee and Schapire, 2013), and 3) maximize (32) for GD-MCBoost.

Several conclusions can be drawn from the results, presented in Table 9. First, AdaBoost.M1 was not able to boost the weak learners used in this experiment for half of the datasets. This is due to the "better than 50% error" boostability condition, which is too stringent for trees of depth 2. Second, compared to SAMME, GD-MCBoost achieved superior performance in eleven of the twelve datasets. The improvements were quite significant in some cases e.g. from 53% to 85% in dataset $\#2$ or from 85% to 95% in dataset $\#7$. The inferior performance of SAMME is explained by the use of the loss of (121), as discussed in Section 8.1. Third, compared to AdaBoost.MM, GD-MCBoost achieved higher accuracy in eleven datasets and the same performance in the remaining one. Again, the improvements were sometimes substantial, e.g. from 64% to 85% in dataset $\#2$. This is not surprising since, as discussed in Section H.2.5, that AdaBoost.MM is a sub-optimal special case of GD-MCBoost. Finally, when compared to all methods, GD-MCBoost achieved the highest accuracy in ten of the twelve datasets. Among the remaining methods, AdaBoost.MM had better performance, followed by AdaBoost-SAMME. AdaBoost.M1 had the worst results. It should be noted that the results of Tables 7, 8 and 9 are not directly comparable, since the classifiers are based on different types of weak learners and have different complexities.

10. Conclusion

In this work, we studied the problem of multiclass boosting with the goal of an integrated understanding of the roles of the optimization strategy, label codewords, weak learners, and multiclass risk. This motivated a new formulation of the problem based on multi-dimensional predictors, multi-dimensional real valued codewords, and proper multiclass margin loss functions. This formulation led to a number of interesting results, such as maximum capacity codeword sets, proper and margin enforcing $\gamma - \phi$ losses, and two new multiclass boosting algorithms, CD-MCBoost and GD-MCBoost, which differ in optimization strategy. CD-MCBoost implements a functional coordinate descent procedure and updates one predictor component at a time, GD-MCBoost a functional gradient descent in a space of multidimensional weak learners, updating all predictor components simultaneously. Both MCBoost algorithms reduce to classical boosting algorithms (such as AdaBoost or LogitBoost) for binary problems, depending on the choice of $\gamma - \phi$ loss and have all the classical boosting properties, such as seeking the weak learner of maximum margin on a reweighted training sample at each iteration and well defined boostability conditions. Beyond the algorithms themselves, the proposed formulation enables a unified treatment of many previous multiclass boosting algorithms. This was used to show that all algorithms implement different combinations of optimization strategy, codewords, weak learners, and loss function, highlighting some of the deficiencies of these algorithms and explaining why they fail under some settings. In particular, it was shown that no previous method matches the support of MCBoost for real codewords of maximum capacity, a proper margin-enforcing loss function, and any family of multidimensional predictors and weak learners. Experimental results confirm the superiority of MCBoost, showing that the two proposed MCBoost algorithms outperform comparable prior methods on a number of datasets.

Beyond algorithms, a number of insights were shown both theoretically and experimentally. First, the dimension d of the predictor $f(x)$ seems to play a central role in large margin multiclass classification. For M classes, any dimension larger $M - 1$ supports a codeword set of maximum capacity. This enables the decomposition of the classifier design problem into two sub-problems: the design of a maximal capacity codeword set, for which an exact algorithm is available, and the design of the best predictor for this codeword set. The latter can be accomplished with the MCBoost algorithms now introduced or some other risk minimization procedure. Second, a predictor of dimension lower than $M - 1$ will, in principle, entail some loss of classification accuracy. It is still, nevertheless, possible to find the best codeword set in a sense related to the margin capacity. This can again be done with an exact algorithm. Interestingly, our experimental results have shown that it is possible to achieve performance identical to that of the margin capacity predictors for dimensions substantially smaller than $M - 1$. The use of a low dimensional predictor can be seen as a form of dimensionality reduction. Unlike classical approaches, which operate on the space of observations x, this is done directly in the output space. Such an interpretation of low-dimensional predictors could be of interest for applications with complexity constraints, for example. Third, for a predictor of dimension $M - 1$ or larger, it is fairly simple to guarantee that a $\gamma - \phi$ loss is margin enforcing and proper. It suffices to guarantee that γ is strictly increasing and ϕ is strictly positive and decreasing, and that the codeword set is of maximum margin capacity. In this case, the posterior class probabilities can be recovered from the outputs of the optimal predictor by a simple mapping, usually a softmax. Since these constraints are quite weak, it is not difficult to define new proper multiclass margin losses, tailored for particular applications. This was exemplified by the derivation of multiclass extensions of popular binary losses, such as the exponential, logistic, or Savage. The resulting MCBoost algorithms are multiclass generalizations of AdaBoost, LogitBoost, and SavageBoost, respectively. Finally, the discussion above highlights the fact that the design of a multiclass boosting algorithm requires the careful *joint* selection of optimization strategy, codewords, weak learners, and loss function. The combination of MCBoost, margin capacity codewords, a proper margin enforcing $\gamma - \phi$ loss, and a family of weak learners with better than random performance is a good solution to all these problems.

References

E. Allwein, R. Schapire, and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal Machine Learning Research*, pages 113–141, 2001.

O. Beijbom, M. Saberian, D. Kriegman, and N. Vasconcelos. Guess-averse loss functions for cost-sensitive multiclass boosting. In *Proceedings of International Conference on Machine Learning*, 2014.

V. Blanz, B. Scholkopf, H. Bultho, C. Burges, V. Vapnik, and T. Vetter. Comparison of view-based object recognition algorithms using realistic 3d models. In *Artificial Neural Networks ICANN*, pages 251–256. Springer, 1996.

A. Buja, W. Stuetzle, and Y. Shen. Loss functions for binary class probability estimation and classification: Structure and applications. 2006.

H. Coxeter. *Regular Polytopes*. Dover Publications, 1973.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2002a.

K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47:201–233, May 2002b.

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SOCG*, pages 253–262, 2004.

T. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761, 2012.

R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley, New York, 2. edition, 2001.

G. Eibl and R. Schapire. Multiclass boosting for weak classifiers. In *Journal of Machine Learning Research*, pages 6–189, 2005.

Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proceedings of International Conference on Machine Learning*, pages 148–156, 1996.

Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Comp. and Sys. Science*, 1997.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28, 1998.

B. Frigyik, S. Srivastava, and M. Gupta. An introduction to functional derivatives. *Technical Report(University of Washington)*, 2008.

T. Gao and D. Koller. Multiclass boosting with hinge loss based on output coding. In *Proceedings of International Conference on Machine Learning*, 2011.

Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (99):1–15, 2012.

Y. Guermeur. Vc theory of large margin multi-category classifiers. *Journal of Machine Learning Research*, 8:2551–2594, December 2007.

V. Guruswami and A. Sahai. Multiclass learning, boosting, and error-correcting codes. In *Proceedings of Annual Conference on Computational Learning Theory*, pages 145–155, 1999.

T. Hastie and R. Tibshirani. Classification by pairwise coupling. In *Neural Information Processing Systems*, 1998.

Roger A. Horn and Charles R. Johnson, editors. *Matrix Analysis*. Cambridge University Press, New York, NY, USA, 1986. ISBN 0-521-30586-1.

C. Hsu and C. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

J. Huang, S. Ertekin, Y. Song, H. Zha, and C. Giles. Efficient multiclass boosting classification with active learning. In *SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 2007.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In *Neural Information Processing Systems*, volume 22, pages 1042–1050. 2009.

Vitaly Kuznetsov, Mehryar Mohri, and Umar Syed. Multi-class deep boosting. In *Advances in Neural Information Processing Systems 27*, pages 2501–2509. 2014.

L. Li. Multiclass boosting with repartitioning. In *Proceedings of International Conference on Machine Learning*, pages 569–576.

Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2074–2081, 2012.

H. Masnadi-Shirazi and N. Vasconcelos. On the design of loss functions for classification: theory, robustness to outliers, and savageboost. In *Neural Information Processing Systems*, pages 1049–1056, 2008.

H. Masnadi-Shirazi and N. Vasconcelos. A view of margin losses as regularizers of probability estimates. *Jounal of Machine Learning Research*, 16:2751–2795, December 2015.

L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In *Neural Information Processing Systems*, 2000.

D. Mease and A. Wyner. Evidence contrary to the statistical view of boosting. *Journal of Machine Learning Research*, 9:131–156, June 2008.

Mohammad Moghimi, Mohammad Saberian, Jian Yang, Li-Jia Li, Nuno Vasconcelos, and Serge Belongie. Boosted convolutional neural networks. In *British Machine Vision Conference*, 2016.

I. Mukherjee and R. Schapire. A theory of multiclass boosting. *Journal of Machine Learning Research*, 14, Feb 2013.

Nils Nilsson. *Learning machines*. McGraw-Hill, New York, 1965.

J. Nocedal and S. Wright. *Numerical Optimization*. Springer Verlag, New York, 1999.

M.D. Reid and R.C. Williamson. Composite binary losses. *Journal of Machine Learning Research*, 11, September 2010.

M. Saberian, H. Masnadi-Shirazi, and N. Vasconcelos. Taylorboost: First and second order boosting algorithms with explicit margin control. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

R. Schapire. Using output codes to boost multiclass learning problems. In *Proceedings of International Conference on Machine Learning*, pages 313–321, 1997.

R. Schapire and Y. Freund. *Boosting: Foundations and Algorithms*. MIT Press, 2012.

R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37:297–336, December 1999.

B. Scholkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In *Proceedings, First International Conference on Knowledge Discovery and Data Mining, Menlo Park*, pages 252–257. AAAI Press, 1995.

T. Sejnowski and C. Rosenberg. Parallel networks that learn to pronounce english text. *Journal of Complex Systems*, 1(1):145–168, 1987.

Y. Sun, S. Todorovic, J. Li, and D. Wu. Unifying the error-correcting and output-code adaboost within the margin framework. In *Proceedings of International Conference on Machine Learning*, pages 872–879, 2005.

R. Tammes. On the origin of number and arrangement of places of exit on the surface of pallen grains. *Rec. Trav. Bot. Neerl.*, 27:1–84, 1930.

Antonio Torralba, Robert Fergus, and Yair Weiss. Small codes and large image databases for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

V. Vapnik. *Statistical Learning Theory*. John Wiley Sons Inc, 1998.

Paul A. Viola and Michael J. Jones. Robust real-time face detection. *International Journal Computer Vision*, 57:137–154, 2004.

J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of European Symposium On Artificial Neural Networks*, pages 219–224, 1999.

B. Zhang, G. Ye, Y. Wang, J. Xu, and G. Herman. Finding shareable informative patterns and optimal coding matrix for multiclass boosting. In *Proceedings of International Conference on Computer Vision*, pages 56 –63, 2009.

Tong Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32:56–85, 2004.

J. Zhu, H. Zou, S. Rosset, and T. Hastie. Multi-class adaboost. *Statistics and Its Interface*, 2:349–3660, 2009.

H. Zou, J. Zhu, and T. Hastie. New multicategory boosting algorithms based on multicategory fisher-consistent losses. *Annals of Statistics*, 2, 2008.

Appendix A. Decision rule

A.1. Proof of Lemma 1

Consider a prediction $f(x)$. Defining

$$k^* = \arg \max_{k \in \{1,\ldots,M\}} \left\langle y^k, f(x) \right\rangle, \tag{135}$$

it follows that

$$\mathcal{M}(y^{k^*}, f(x)) \geq 0 \geq \mathcal{M}(y^l, f(x)) \quad \forall\, l \neq k^*, \tag{136}$$

and thus

$$k^* = \arg \max_{k \in \{1,\ldots,M\}} \mathcal{M}(y^k, f(x)). \tag{137}$$

To prove the converse we start by noting that, by definition of margin in (10), for any $f(x)$ there is at least one codeword with non-negative margin. Hence, if

$$k^* = \arg \max_{k \in \{1,\ldots,M\}} \mathcal{M}(y^k, f(x)), \tag{138}$$

it follows from (10) that

$$\left\langle y^{k^*}, f(x) \right\rangle \geq \max_{l \neq k^*} \left\langle y^l, f(x) \right\rangle, \tag{139}$$

from which

$$k^* = \arg \max_{k \in \{1,\ldots,M\}} \left\langle y^k, f(x) \right\rangle. \tag{140}$$

A.2. Proof of Corollary 1

If $F(x) = c$, it follows from Lemma 1 that

$$c = \arg \max_{k \in \{1,\ldots,M\}} \left\langle y^k, f(x_i) \right\rangle, \tag{141}$$

and (18) follows from (10). If $\mathcal{M}(y^c, f(x)) > 0$, it follows from (10) that

$$\langle y^c, f(x) \rangle > \max_{k \neq c} \left\langle y^k, f(x) \right\rangle. \tag{142}$$

and, from (17), $F(x) = c$.

A.3. Proof of Corollary 2

If $\mathcal{M}_p(\mathcal{D}, f, \mathcal{Y}) > 0$, it follows from (14) that

$$\mathcal{M}(y^{c_i}, f(x_i)) > 0 \quad \forall x_i \in \mathcal{D}, \tag{143}$$

and, from Corollary 1, all examples are classified correctly. Conversely, if all examples are classified correctly, then (143) follows from Corollary 1, and

$$\mathcal{M}_p(\mathcal{D}, f, \mathcal{Y}) = \min_{(x_i, y^{c_i}) \in \mathcal{D}} \mathcal{M}(y^{c_i}, f(x_i)) > 0. \tag{144}$$

Appendix B. Algorithms

B.1. Derivation of GD-MCBoost

For the problem of (25), the gradient expression follows from the application of (27) to (21).

$$\begin{aligned}
-\delta\overline{R}[f^t;g] &= -\frac{1}{n}\frac{\partial}{\partial\epsilon}\sum_{i=1}^{n} L_M^{\gamma-\phi}[y^{c_i}, f^t(x_i)+\epsilon g(x_i)]\Big|_{\epsilon=0} \\
&= -\frac{1}{n}\sum_{i=1}^{n}\frac{\partial L_M^{\gamma-\phi}[y^{c_i}, f^t(x_i)+\epsilon g(x_i)]}{\partial\epsilon}\Big|_{\epsilon=0}
\end{aligned}$$

For the $\gamma-\phi$ loss of (23) this leads to

$$\begin{aligned}
-\delta\overline{R}[f^t;g] &= -\frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial\epsilon}\gamma\left(\sum_{k\neq c_i}\phi\left[\frac{1}{2}\left\langle f^t(x_i)+\epsilon g(x_i), y^{c_i}-y^k\right\rangle\right]\right)\Bigg|_{\epsilon=0} \\
&= -\frac{1}{n}\sum_{i=1}^{n}\gamma'\left(\sum_{k\neq c_i}\phi\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle\right]\right)\times \\
&\times\left(\sum_{k\neq c_i}\frac{\partial}{\partial\epsilon}\phi\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle+\frac{\epsilon}{2}\left\langle g(x_i), y^{c_i}-y^k\right\rangle\right]\right)\Bigg|_{\epsilon=0} \\
&= -\frac{1}{n}\sum_{i=1}^{n}\gamma'\left(\sum_{k\neq c_i}\phi\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle\right]\right)\times \\
&\times\left(\sum_{k\neq c_i}\frac{1}{2}\left\langle g(x_i), y^{c_i}-y^k\right\rangle\phi'\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle\right]\right) \\
&= -\frac{1}{n}\sum_{i=1}^{n}\gamma'\left(\sum_{k\neq c_i}\phi\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle\right]\right)\left(\sum_{k\neq c_i}\phi'\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle\right]\right)\times \\
&\times\left(\sum_{k\neq c_i}\frac{1}{2}\left\langle g(x_i), y^{c_i}-y^k\right\rangle\frac{\phi'\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle\right]}{\sum_{k\neq c_i}\phi'\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle\right]}\right).
\end{aligned}$$

Using (29) and (30), this can finally be written as

$$\begin{aligned}
-\delta\overline{R}[f^t;g] &= \frac{1}{n}\sum_{i=1}^{n} w_i\left(\sum_{k\neq c_i}\frac{1}{2}\left\langle g(x_i), y^{c_i}-y^k\right\rangle\tau_k(x_i,c_i)\right) \\
&= \frac{1}{2n}\sum_{i=1}^{n} w_i\left\langle g(x_i), \sum_{k\neq c_i}(y^{c_i}-y^k)\tau_k(x_i,c_i)\right\rangle \\
&= \frac{1}{2n}\sum_{i=1}^{n} w_i\left\langle g(x_i), y^{c_i}-\sum_{k\neq c_i}y^k\tau_k(x_i,c_i)\right\rangle.
\end{aligned}$$

B.2. Derivation of CD-MCBoost

For the problem of (35) the gradient expression follows from the application of (27) to (21),

$$\begin{aligned}
-\delta\overline{R}[f^t; j, g] &= -\frac{1}{n}\frac{\partial}{\partial\epsilon}\sum_{i=1}^{n} L_M^{\gamma-\phi}[y^{c_i}, f^t(x_i)+\epsilon g(x_i)\mathbf{1}_j]\Bigg|_{\epsilon=0} \\
&= -\frac{1}{n}\sum_{i=1}^{n}\frac{\partial L_M^{\gamma-\phi}[y^{c_i}, f^t(x_i)+\epsilon g(x_i)\mathbf{1}_j]}{\partial\epsilon}\Bigg|_{\epsilon=0}
\end{aligned}$$

For the $\gamma-\phi$ loss of (23) this leads to

$$\begin{aligned}
-\delta\overline{R}[f^t; j, g] &= -\frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial\epsilon}\gamma\left(\sum_{k\neq c_i}\phi\left[\frac{1}{2}\left\langle f^t(x_i)+\epsilon g(x_i)\mathbf{1}_j, y^{c_i}-y^k\right\rangle\right]\right)\Bigg|_{\epsilon=0} \\
&= -\frac{1}{n}\sum_{i=1}^{n}\gamma'\left(\sum_{k\neq c_i}\phi\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle\right]\right)\times \\
&\quad\left(\sum_{k\neq c_i}\frac{\partial}{\partial\epsilon}\phi\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle+\frac{1}{2}\epsilon g(x_i)\left\langle \mathbf{1}_j, y^{c_i}-y^k\right\rangle\right]\right)\Bigg|_{\epsilon=0} \\
&= -\frac{1}{n}\sum_{i=1}^{n}\gamma'\left(\sum_{k\neq c_i}\phi\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle\right]\right)\times \\
&\quad\left(\sum_{k\neq c_i}\frac{1}{2}g(x_i)\left\langle \mathbf{1}_j, y^{c_i}-y^k\right\rangle\phi'\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle\right]\right) \\
&= -\frac{1}{2n}\sum_{i=1}^{n}\gamma'\left(\sum_{k\neq c_i}\phi\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle\right]\right)\left(\sum_{k\neq c_i}\phi'\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle\right]\right) \\
&\times\quad\left(\sum_{k\neq c_i}g(x_i)\left\langle \mathbf{1}_j, y^{c_i}-y^k\right\rangle\frac{\phi'\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle\right]}{\sum_{k\neq c_i}\phi'\left[\frac{1}{2}\left\langle f^t(x_i), y^{c_i}-y^k\right\rangle\right]}\right).
\end{aligned}$$

Using (29) and (30), this can finally be written as

$$\begin{aligned}
-\delta\overline{R}[f^t; j, g] &= \frac{1}{2n}\sum_{i=1}^{n}w_i\left(\sum_{k\neq c_i}g(x_i)\left\langle \mathbf{1}_j, y^{c_i}-y^k\right\rangle\tau_k(x_i,c_i)\right) \\
&= \frac{1}{2n}\sum_{i=1}^{n}w_i g(x_i)\left\langle \mathbf{1}_j, \sum_{k\neq c_i}(y^{c_i}-y^k)\tau_k(x_i,c_i)\right\rangle \\
&= \frac{1}{2n}\sum_{i=1}^{n}w_i g(x_i)\left\langle \mathbf{1}_j, y^{c_i}-\sum_{k\neq c_i}y^k\tau_k(x_i,c_i)\right\rangle.
\end{aligned}$$

Appendix C. Maximum capacity codewords

C.1. Proof of Theorem 1

Consider any direction v such that $||v|| = 1$. Using (9), (55) and the fact that the minimum cannot be larger than the average

$$
\begin{aligned}
\mathcal{M}(y^k, v) = \min_{l \neq k} \frac{1}{2} \left\langle y^k - y^l, v \right\rangle &\leq \frac{1}{2(M-1)} \sum_{l \neq k} \left\langle y^k - y^l, v \right\rangle &(145)\\
&= \frac{1}{2(M-1)} \left\langle \sum_{l \neq k} (y^k - y^l), v \right\rangle \\
&= \frac{1}{2(M-1)} \left\langle (M-1) y^k - \sum_{l \neq k} y^l, v \right\rangle \\
&= \frac{1}{2(M-1)} \left\langle M y^k - \sum_{l=1}^{M} y^l, v \right\rangle \\
&= \frac{M}{2(M-1)} \left\langle y^k, v \right\rangle \\
&\leq \frac{M}{2(M-1)}. &(146)
\end{aligned}
$$

C.2. Proof of Theorem 2

By definition of margin capacity

$$
\mathcal{C}[\mathcal{Y}] = \min_{k=1,\ldots,M} \mathcal{M}(y^k, \xi^k), \tag{147}
$$

where ξ^k is the direction of largest capacity for class k. As shown in the proof of Theorem 1,

$$
\begin{aligned}
\mathcal{M}(y^k, \xi^k) &= \min_{l \neq k} \frac{1}{2} \left\langle y^k - y^l, \xi^k \right\rangle &(148)\\
&\leq \frac{1}{2(M-1)} \sum_{l \neq k} \left\langle y^k - y^l, \xi^k \right\rangle &(149)\\
&= \frac{M}{2(M-1)} \left\langle y^k, \xi^k \right\rangle. &(150)
\end{aligned}
$$

The margin component $\mathcal{M}(y^k, \xi^k)$ is maximum when equality holds. Since the minimum is equal to the average if and only if all elements are equal, this is the case if and only if

$$
\frac{1}{2} \left\langle y^k - y^l, \xi^k \right\rangle = \frac{M}{2(M-1)} \left\langle y^k, \xi^k \right\rangle \quad \forall l \neq k. \tag{151}
$$

Hence the maximum value of the margin component is

$$
\mathcal{M}(y^k, \xi^k) = \frac{M}{2(M-1)} \left\langle y^k, \xi^k \right\rangle \leq \frac{M}{2(M-1)} \tag{152}
$$

with equality if and only if

$$\left\langle y^k, \xi^k \right\rangle = 1. \tag{153}$$

Since both y^k and ξ^k are unit vectors, this holds if and only if

$$y^k = \xi^k. \tag{154}$$

It follows that the margin capacity of (147) meets the capacity bound of (60) if and only if, for all k, the directions of largest margin are $\xi^k = y^k$ and, from (151),

$$\frac{1}{2}\left\langle y^k - y^l, y^k \right\rangle = \frac{M}{2(M-1)} \quad \forall l \neq k \tag{155}$$

$$\frac{1}{2}\left(1 - \left\langle y^l, y^k \right\rangle\right) = \frac{M}{2(M-1)} \quad \forall l \neq k, \tag{156}$$

$$\left\langle y^l, y^k \right\rangle = -\frac{1}{M-1}, \forall l \neq k. \tag{157}$$

C.3. Proof of Theorem 3

Assume that $\mathcal{Y}^c(M,d)$ is a set of codewords that achieves the capacity bound of (60). Then, from Theorem 2 and Definition 6, $\mathcal{Y}^c(M,d)$ is a set of M centered, unit norm, d-dimensional vectors y^k such that

$$\left\langle y^k, y^l \right\rangle = -\frac{1}{M-1}, \ \forall k, l \neq k. \tag{158}$$

The proof is by construction and uses a known method for the design of regular simplexes (Coxeter, 1973). Let y^k be the codewords in $\mathcal{Y}^c(M,d)$. Without loss of generality we can set $y^1 = [1, 0, \dots, 0]^T \in \mathbb{R}^d$. From (158) it follows that

$$y_1^k = -\frac{1}{M-1} \quad \forall k > 1, \tag{159}$$

where y_i^k is the i^{th} coordinate of vector y^k. Defining

$$\bar{y}^k = \frac{M-1}{\sqrt{M(M-2)}}[y_2^{k+1}, \dots, y_d^{k+1}] \in \mathbb{R}^{d-1} \quad k = 1, \dots, M-1, \tag{160}$$

it follows that $\sum_{k=1}^{M-1} \bar{y}^k = 0$,

$$\begin{aligned} \|\bar{y}^k\|^2 &= \frac{(M-1)^2}{M(M-2)}[\|y^{k+1}\|^2 - [y_1^{k+1}]^2] \\ &= \frac{(M-1)^2}{M(M-2)}\left[1 - \frac{1}{(M-1)^2}\right] \\ &= 1, \quad \forall k \end{aligned} \tag{161}$$

and

$$\begin{aligned} \left\langle \bar{y}^k, \bar{y}^l \right\rangle &= \frac{(M-1)^2}{M(M-2)}\left[\left\langle y^{k+1}, y^{l+1} \right\rangle - y_1^{k+1} y_1^{l+1}\right] \\ &= \frac{(M-1)^2}{M(M-2)}\left[-\frac{1}{M-1} - \frac{1}{(M-1)^2}\right] \\ &= -\frac{(M-1)^2}{M(M-2)}\frac{M}{(M-1)^2} = -\frac{1}{M-2}, \quad \forall k, l \neq k. \end{aligned} \tag{162}$$

Hence, the codewords $\bar{y}^k$ are the elements of $\mathcal{Y}^c(M-1, d-1)$, the codeword set that meets the capacity bound of $\mathcal{S}(M-1, d-1)$. In summary, the application of the simplex design procedure of (160) to a codeword set $\mathcal{Y}^c(M, d)$ produces a codeword set $\mathcal{Y}^c(M-1, d-1)$.

If $d \leq M-2$, the procedure can be applied $d-1$ times, to produce a codeword set $\mathcal{Y}^c(M-d+1, 1)$. This is a set of $M-d+1$ centered, unit norm, scalars that satisfy (158). It follows from the unit norm constraint that $\bar{y}^k \in \{+1, -1\}$ and thus, for $k \neq l$, $\langle \bar{y}^k, \bar{y}^l \rangle = -1$. This, however, contradicts (158), since $-\frac{1}{M-d+1-1} = -\frac{1}{M-d} \geq -\frac{1}{2}$. Hence, $\mathcal{S}(M, d)$ contains no set of codewords that meets the capacity bound, when $d \leq M-2$.

If $d \geq M-1$, the procedure can be applied $M-2$ times, to produce a codeword set $\mathcal{Y}^c(2, d-M+2)$. This is a set of 2 centered, unit norm, $(d-M+2)$-dimensional vectors that satisfy (158), i.e. $\langle \bar{y}^1, \bar{y}^2 \rangle = -1$. Since $\bar{y}^1 = [1, 0, \ldots, 0]$ and $\bar{y}^2 = [-1, 0, \ldots, 0]$ in $\mathbb{R}^{d-M+2}$ satisfy these conditions, there exists a sequence $\mathcal{Y}^c(2, d-M+2), \ldots, \mathcal{Y}^c(M, d)$ of codeword sets that meet the capacity bounds of $\mathcal{S}(2, d-M+2), \ldots, \mathcal{S}(M, d)$, respectively. Since the procedure used to design this sequence is the regular simplex design procedure of (160), the codewords in $\mathcal{Y}^c(2, d-M+2), \ldots, \mathcal{Y}^c(M, d)$ form a regular simplex in $\mathbb{R}^{d-M+2}, \ldots, \mathbb{R}^d$, respectively.

Appendix D. Low dimensional predictors

D.1. Proof of Lemma 2

We start with (68). The left inequality follows from (56), (57) and (9), since

$$\begin{aligned}
\mathcal{C}[\mathcal{Y}] &= \min_{k=1,\ldots,M} \max_{||v||=1} \mathcal{M}(y^k, v) \\
&\geq \min_{k=1,\ldots,M} \mathcal{M}(y^k, y^k) \\
&= \frac{1}{2} \min_{k=1,\ldots,M} \min_{l \neq k} \left[\|y^k\| - \left\langle y^k, y^l \right\rangle \right] \\
&= \frac{1}{4} \min_{k=1,\ldots,M} \min_{l \neq k} \left[2\|y^k\| - 2\left\langle y^k, y^l \right\rangle \right] \\
&= \frac{1}{4} \min_{k=1,\ldots,M} \min_{l \neq k} \left[\|y^k\| + \|y^l\| - 2\left\langle y^k, y^l \right\rangle \right] \\
&= \frac{1}{4} \min_{k, l \neq k} \|y^k - y^l\|^2. \qquad (163)
\end{aligned}$$

The right inequality follows from (9) since, for any v such that $||v|| = 1$,

$$\begin{aligned}
\mathcal{M}(y^k, v) &= \frac{1}{2} \min_{l \neq k} \left\langle y^k - y^l, v \right\rangle \\
&\leq \frac{1}{2} \min_{l \neq k} \|y^k - y^l\|^2, \qquad (164)
\end{aligned}$$

and thus

$$\begin{aligned}
\mathcal{C}[\mathcal{Y}] &= \min_{k=1,\ldots,M} \max_{||v||=1} \mathcal{M}(y^k, v) \\
&\leq \min_{k=1,\ldots,M} \max_{||v||=1} \frac{1}{2} \min_{l \neq k} \|y^k - y^l\|^2 \\
&= \frac{1}{2} \min_{k, l \neq k} \|y^k - y^l\|^2. \qquad (165)
\end{aligned}$$

Finally, (69) follows from the left inequality of (68) and (60).

D.2. Proof of Theorem 4

Let $\mathcal{Y}^*$ be a codeword set of maximum capacity. From (66) and (62)

$$\begin{aligned} d_{min}[\mathcal{Y}^*] &= \min_{k,l\neq k} \|(y^*)^k - (y^*)^l\|^2 \\ &= \min_{k,l\neq k} \left[2 - 2\left\langle (y^*)^k, (y^*)^l \right\rangle\right] \\ &= \min_{k,l\neq k} \left[2 + \frac{2}{M-1}\right] \\ &= \frac{2M}{M-1} \end{aligned} \tag{166}$$

and $\mathcal{Y}^*$ meets the bound of (69). Hence, it is a max-min distance codeword set. Let $\mathcal{Y}^*$ be a codeword set of max-min distance, i.e.

$$d_{min}[\mathcal{Y}^*] = \frac{2M}{M-1}. \tag{167}$$

From (68) it follows that

$$\mathcal{C}[\mathcal{Y}^*] \geq \frac{M}{2(M-1)}, \tag{168}$$

and $\mathcal{Y}^*$ meets the capacity bound of (60). Hence, it is a codeword set of maximum capacity.

Appendix E. Properties

In this appendix, we derive several properties of MCBoost algorithms.

E.1. Classification weak learners

We start by deriving the specialization of the algorithm to classification weak learners. While we consider only GD-MCBoost, a similar derivation is possible for CD-MCBoost. We assume a weak learner space $\overline{\mathcal{H}}$ of multiclass classifiers, i.e. $h : \mathcal{X} \rightarrow \mathcal{Y}$ where $\mathcal{Y} = \{y^1, \ldots, y^M\}$ is the codeword set of the classification problem.

E.1.1. MAXIMUM CAPACITY CODEWORDS

Assume that $\mathcal{Y}$ is a codeword set of maximum capacity in $\mathcal{S}(M, d)$. The predictor learned by MCBoost is then

$$\begin{aligned} f(x) &= \sum_t \alpha_t g_t(x) \\ &= \sum_t \alpha_t \sum_k I(g_t(x) = y^k) y^k \\ &= \sum_k y^k \sum_{t|g_t(x)=y^k} \alpha_t \\ &= \sum_k \zeta_k y_k \end{aligned} \tag{169}$$

where

$$\zeta_k = \sum_{t|g_t(x)=y^k} \alpha_t. \tag{170}$$

Since

$$\begin{aligned}\left\langle f(x), y^j \right\rangle &= \sum_k \zeta_k < y_k, y_j > \\ &= \left(\zeta_j < y_j, y_j > + \sum_{k\neq j} \zeta_k < y_k, y_j >\right) \\ &= \left(\zeta_j - \frac{1}{M-1}\sum_{k\neq j}\zeta_k\right) \\ &= \left(\sum_{t|g_t(x)=y^j} \alpha_t - \frac{1}{M-1}\sum_{k\neq j}\sum_{t|g_t(x)=y^k}\alpha_t\right),\end{aligned} \tag{171}$$

where we have used the codeword properties of (55) and (61), and

$$\begin{aligned}\left\langle f(x), y^j \right\rangle - \left\langle f(x), y^k \right\rangle &= \zeta_j - \frac{1}{M-1}\sum_{l\neq j}\zeta_l - \zeta_k + \frac{1}{M-1}\sum_{l\neq k}\zeta_l \\ &= \zeta_j - \frac{1}{M-1}\zeta_k - \zeta_k + \frac{1}{M-1}\zeta_j \\ &= \frac{M}{M-1}(\zeta_j - \zeta_k).\end{aligned} \tag{172}$$

$f(x)$ has margin components

$$\begin{aligned}\left\langle f(x), y^c - y^k \right\rangle &= \frac{M}{M-1}\left(\zeta_c - \zeta_k\right) \\ &= \frac{M}{M-1}\left(\sum_{t|g_t(x)=y^c}\alpha_t - \sum_{t|g_t(x)=y^k}\alpha_t\right).\end{aligned} \tag{173}$$

With regards to learning, the steepest descent direction of (32) can be written as

$$g^*(x) = \arg\max_{g\in\overline{\mathcal{H}}} \sum_{j=1}^{M}\sum_{i|g(x_i)=y^j} w_i \left\langle y^j, y^{c_i} - \sum_{k\neq c_i} y^k \tau_k(x_i, c_i)\right\rangle. \tag{174}$$

There are two possibilities for the dot-products $\left\langle y^j, y^{c_i} - \sum_{k\neq c_i} y^k \tau_k(x_i, c_i)\right\rangle$:

1. $y^j = y^{c_i}$: in this case

$$\begin{aligned}\left\langle y^j, y^{c_i} - \sum_{k\neq c_i} y^k \tau_k(x_i, c_i)\right\rangle &= ||y^{c_i}||^2 - \sum_{k\neq c_i}\left\langle y^{c_i}, y^k\right\rangle \tau_k(x_i, c_i) \\ &= 1 + \frac{1}{M-1}\sum_{k\neq c_i}\tau_k(x_i, c_i) \\ &= 1 + \frac{1}{M-1} = \frac{M}{M-1}\end{aligned} \tag{175}$$

where we have also used (30).

2. $y^j \neq y^{c_i}$: in this case

$$\begin{aligned}\left\langle y^j, y^{c_i} - \sum_{k\neq c_i} y^k \tau_k(x_i,c_i)\right\rangle &= \langle y^j, y^{c_i}\rangle - ||y^j||^2\tau_j(x_i,c_i) - \sum_{k\neq c_i,j}\left\langle y^j, y^k\right\rangle \tau_k(x_i,c_i) \\ &= -\frac{1}{M-1} - \tau_j(x_i,c_i) + \frac{1}{M-1}\sum_{k\neq c_i,j}\tau_k(x_i,c_i) \\ &= -\frac{1}{M-1} - \tau_j(x_i,c_i) + \frac{1}{M-1}\left(1-\tau_j(x_i,c_i)\right) \\ &= -\frac{M}{M-1}\tau_j(x_i,c_i), \qquad (176)\end{aligned}$$

where we have also used (30).

It follows that

$$\begin{aligned} g^*(x) &= \arg\max_{g\in\overline{\mathcal{H}}} \sum_{i|g(x_i)=y^{c_i}} w_i - \sum_{j\neq c_i}\sum_{i|g(x_i)=y^j} w_i\tau_j(x_i,c_i) \\ &= \arg\max_{g\in\overline{\mathcal{H}}} \sum_i w_i - \sum_{i|g(x_i)\neq y^{c_i}} w_i - \sum_{j\neq c_i}\sum_{i|g(x_i)=y^j} w_i\tau_j(x_i,c_i) \\ &= \arg\min_{g\in\overline{\mathcal{H}}} \sum_{j\neq c_i}\sum_{i|g(x_i)=y^j} w_i(1+\tau_j(x_i,c_i)) \\ &= \arg\min_{g\in\overline{\mathcal{H}}} \sum_{i|g(x_i)\neq y^{c_i}} w_i\left(1+\sum_{j\neq c_i}\tau_j(x_i,c_i)I(g(x_i)=y^j)\right). \qquad (177)\end{aligned}$$

E.1.2. CANONICAL CODEWORDS

We next consider the case where $\mathcal{Y}$ is the of set canonical codewords $y^j = \mathbf{1}_j$. In this case, the predictor learned by MCBoost is

$$\begin{aligned} f(x) &= \sum_t \alpha_t g_t(x) \\ &= \sum_t \alpha_t \sum_k I(g_t(x)=\mathbf{1}_k)\mathbf{1}_k \\ &= \sum_k \mathbf{1}_k \sum_{t|g_t(x)=\mathbf{1}_k}\alpha_t. \qquad (178)\end{aligned}$$

which can be written as

$$f_k(x) = \sum_{t|g_t(x)=\mathbf{1}_k}\alpha_t. \qquad (179)$$

It follows that

$$\left\langle f(x), y^j\right\rangle = \langle f(x), \mathbf{1}_j\rangle = f_j(x) = \sum_{t|g_t(x)=\mathbf{1}_j}\alpha_t. \qquad (180)$$

and

$$\left\langle f(x), y^c - y^k \right\rangle = f_c(x) - f_k(x) = \sum_{t|g_t(x)=\mathbf{1}_c} \alpha_t - \sum_{t|g_t(x)=\mathbf{1}_k} \alpha_t. \tag{181}$$

With regards to learning, the steepest descent direction of (32) can be written as

$$g^*(x) = \arg\max_{g\in\overline{\mathcal{H}}} \sum_{j=1}^{M} \sum_{i|g(x_i)=y^j} w_i \left\langle y^j, y^{c_i} - \sum_{k\neq c_i} y^k \tau_k(x_i, c_i) \right\rangle. \tag{182}$$

and there are two possibilities for the dot-products $\left\langle y^j, y^{c_i} - \sum_{k\neq c_i} y^k \tau_k(x_i, c_i) \right\rangle$:

1. $y^j = y^{c_i}$: in this case

$$\left\langle y^j, y^{c_i} - \sum_{k\neq c_i} y^k \tau_k(x_i, c_i) \right\rangle = ||y^{c_i}||^2 - \sum_{k\neq c_i} \left\langle y^{c_i}, y^k \right\rangle \tau_k(x_i, c_i) = 1. \tag{183}$$

2. $y^j \neq y^{c_i}$: in this case

$$\begin{aligned}\left\langle y^j, y^{c_i} - \sum_{k\neq c_i} y^k \tau_k(x_i, c_i) \right\rangle &= \langle y^j, y^{c_i} \rangle - ||y^j||^2 \tau_j(x_i, c_i) - \sum_{k\neq c_i, j} \left\langle y^j, y^k \right\rangle \tau_k(x_i, c_i) \\ &= -\tau_j(x_i, c_i). \end{aligned} \tag{184}$$

It follows that

$$\begin{aligned} g^*(x) &= \arg\max_{g\in\overline{\mathcal{H}}} \sum_{i|g(x_i)=y^{c_i}} w_i - \sum_{j\neq c_i} \sum_{i|g(x_i)=y^j} w_i \tau_j(x_i, c_i) \\ &= \arg\max_{g\in\overline{\mathcal{H}}} \sum_i w_i - \sum_{i|g(x_i)\neq y^{c_i}} w_i - \sum_{j\neq c_i} \sum_{i|g(x_i)=y^j} w_i \tau_j(x_i, c_i) \\ &= \arg\min_{g\in\overline{\mathcal{H}}} \sum_{j\neq c_i} \sum_{i|g(x_i)=y^j} w_i (1 + \tau_j(x_i, c_i)) \\ &= \arg\min_{g\in\overline{\mathcal{H}}} \sum_{i|g(x_i)\neq y^{c_i}} w_i \left(1 + \sum_{j\neq c_i} \tau_j(x_i, c_i) I(g(x_i) = y^j) \right) \end{aligned} \tag{185}$$

E.2. Exponential loss

We next consider the algorithm derived from the loss function

$$L_M[\bar{y}^{c_i}, f(x_i)] = e^{-\langle \bar{y}^{c_i}, \bar{f}(x_i) \rangle}. \tag{186}$$

E.2.1. GD-MCBOOST

Using this loss in (21), it follows that

$$\begin{aligned}
-\delta\overline{R}[f^t;g] &= -\frac{1}{n}\sum_{i=1}^{n}\left.\frac{\partial L_M[y^{c_i}, f^t(x_i)+\epsilon g(x_i)]}{\partial\epsilon}\right|_{\epsilon=0} \\
&= -\frac{1}{n}\sum_{i=1}^{n}\left.\frac{\partial}{\partial\epsilon}e^{-\left\langle f^t(x_i)+\epsilon g(x_i), y^{c_i}\right\rangle}\right|_{\epsilon=0} \\
&= \frac{1}{n}\sum_{i=1}^{n} e^{-\left\langle f^t(x_i), y^{c_i}\right\rangle}\left\langle g(x_i), y^{c_i}\right\rangle \\
&= \frac{1}{n}\sum_{i=1}^{n} w_i \left\langle g(x_i), y^{c_i}\right\rangle, \qquad (187)
\end{aligned}$$

with

$$w_i = e^{-\left\langle f^t(x_i), y^{c_i}\right\rangle}. \qquad (188)$$

Hence, the gradient descent direction is

$$g_t(x) = \arg\max_{g\in\overline{\mathcal{H}}}, \sum_{i=1}^{n} w_i \left\langle g(x_i), y^{c_i}\right\rangle, \qquad (189)$$

and the descent update for the predictor

$$f^{t+1}(x) = f^t(x) + \alpha_t g_t(x), \qquad (190)$$

for a suitably chosen step size α^t.

E.2.2. MAXIMUM CAPACITY CODEWORDS

Assume that $g_t(x) \in \mathcal{Y}$, where $\mathcal{Y}$ is a codeword set of maximum capacity in $\mathcal{S}(M,d)$. Then, as shown in Section E.1.1, the predictor learned by MCBoost is

$$f(x) = \sum_t \alpha_t g_t(x) = \sum_k y^k \sum_{t|g_t(x)=y^k} \alpha_t, \qquad (191)$$

and has codeword projections

$$\left\langle f(x), y^j\right\rangle = \sum_{t|g_t(x)=y^j} \alpha_t - \frac{1}{M-1}\sum_{k\neq j}\sum_{t|g_t(x)=y^k} \alpha_t. \qquad (192)$$

Hence, the weights of (188) are

$$w_i = e^{-\sum_t \alpha_t I(g_t(x)=y^{c_i}) - \frac{1}{M-1}\sum_{k\neq c_i}\sum_t \alpha_t I(g_t(x)=y^k)}. \qquad (193)$$

(189) can be written as

$$\begin{aligned}
g_t(x) &= \arg\max_{g\in\overline{\mathcal{H}}} \sum_{j=1}^{M} \sum_{i=1|g_t(x_i)=y^j}^{n} w_i \left\langle y^j, y^{c_i}\right\rangle \\
&= \arg\max_{g\in\overline{\mathcal{H}}} \left\{ \sum_{i|g_t(x_i)=y^{c_i}} w_i ||y^{c_i}||^2 + \sum_{j\neq c_i} \sum_{i|g_t(x_i)=y^j} w_i \left\langle y^j, y^{c_i}\right\rangle \right\} \\
&= \arg\max_{g\in\overline{\mathcal{H}}} \left\{ \sum_{i|g_t(x_i)=y^{c_i}} w_i - \frac{1}{M-1} \sum_{i|g_t(x_i)\neq y^{c_i}} w_i \right\} \\
&= \arg\max_{g\in\overline{\mathcal{H}}} \left\{ \sum_i w_i - \sum_{i|g_t(x_i)\neq y^{c_i}} w_i - \frac{1}{M-1} \sum_{i|g_t(x_i)\neq y^{c_i}} w_i \right\} \\
&= \arg\min_{g\in\overline{\mathcal{H}}} \sum_{i|g_t(x_i)\neq y^{c_i}} w_i \\
&= \arg\max_{g\in\overline{\mathcal{H}}} \sum_{i|g_t(x_i)=y^{c_i}} w_i. \qquad (194)
\end{aligned}$$

E.2.3. CANONICAL CODEWORDS

Assume that $\mathcal{Y}$ is the of set canonical codewords $y^j = \mathbf{1}_j$. In this case, as shown in Section E.1.2, the predictor learned by boosting has components

$$f_j(x) = \sum_{t|g_t(x)=\mathbf{1}_j} \alpha_t = \sum_t \alpha_t I(g_t(x) = \mathbf{1}_j), \qquad (195)$$

and codeword projections

$$\left\langle f(x), y^j \right\rangle = f_j(x) = \sum_t \alpha_t I(g_t(x) = \mathbf{1}_j). \qquad (196)$$

Hence, the weights of (188) are

$$w_i = e^{-\sum_t \alpha_t I(g_t(x)=\mathbf{1}_{c_i})}. \qquad (197)$$

(189) can be written as

$$\begin{aligned}
g_t(x) &= \arg\max_{g\in\overline{\mathcal{H}}} \sum_{j=1}^{M} \sum_{i=1|g_t(x_i)=y^j}^{n} w_i \left\langle \mathbf{1}_j, \mathbf{1}_{c_i}\right\rangle \\
&= \arg\max_{g\in\overline{\mathcal{H}}} \sum_{i=1|g_t(x_i)=\mathbf{1}_{c_i}}^{n} w_i \\
&= \arg\max_{g\in\overline{\mathcal{H}}} \sum_{i=1}^{n} w_i I(g_t(x_i) = \mathbf{1}_{c_i}). \qquad (198)
\end{aligned}$$

Appendix F. Margin maximization

F.1. Proof of Theorem 5

Defining $l^* = \arg\min_{l \neq c} u^c - u^l$, (23) can be written as

$$L_M^{\gamma-\phi}[y^c, f(x)] \quad = \quad \gamma \left[\phi\left(u^c - u^{l^*}\right) + \sum_{l \neq c, l^*} \phi\left(u^c - u^l\right) \right], \tag{199}$$

and it follows from (9) that

$$L_M^{\gamma-\phi}[y^c, f(x)] \quad = \quad \gamma \left[\phi[\mathcal{M}(y^c, f(x))] \left(1 + \sum_{l \neq c, l^*} \frac{\phi(u^c - u^l)}{\phi[\mathcal{M}(y^c, f(x))]} \right) \right]. \tag{200}$$

Since γ is strictly increasing and $\phi(.)$ is strictly positive

$$\frac{\phi(u^c - u^l)}{\phi[\mathcal{M}(y^c, f(x))]} > 0, \, \forall c, l, x. \tag{201}$$

and

$$\phi[\mathcal{M}(y^c, f(x))] \left(1 + \sum_{l \neq c, l^*} \frac{\phi(u^c - u^l)}{\phi[\mathcal{M}(y^c, f(x))]} \right) > \phi[\mathcal{M}(y^c, f(x))], \, \forall x \tag{202}$$

from which

$$L_M^{\gamma-\phi}[y^c, f(x)] \quad > \quad \gamma\left(\phi[\mathcal{M}(y^c, f(x))]\right) \qquad \forall x. \tag{203}$$

Combining (21), (14) and the fact that $\gamma(.)$ is strictly positive

$$\begin{aligned}
\overline{R}_{L_M^{\gamma-\phi}}(f) \quad &= \quad \frac{1}{n} \sum_{i=1}^n L_M^{\gamma-\phi}[y^{c_i}, f(x_i)] \\
&> \quad \frac{1}{n} \sum_{i=1}^n \gamma\left(\phi[\mathcal{M}(y^{c_i}, f(x_i))]\right) \\
&= \quad \frac{1}{n} \gamma\left(\phi[\mathcal{M}_p(\mathcal{D}, f, \mathcal{Y})]\right) \left[1 + \sum_{i \neq i^*} \frac{\gamma\left(\phi[\mathcal{M}(y^{c_i}, f(x_i))]\right)}{\gamma\left(\phi[\mathcal{M}_p(\mathcal{D}, f, \mathcal{Y})]\right)} \right] \\
&> \quad \frac{1}{n} \gamma\left(\phi[\mathcal{M}_p(\mathcal{D}, f, \mathcal{Y})]\right)
\end{aligned} \tag{204}$$

where $i^* = \arg\min_i \mathcal{M}(y^{c_i}, f(x_i))$.

Appendix G. Proper $\gamma - \phi$ losses

G.1. Proof of Theorem 6

From (95),

$$\begin{aligned} R_{L_M^{\gamma-\phi}}(f|x) &= \sum_{k=1}^{M} \eta_k(x) L_M^{\gamma-\phi}[y^k, f(x)] \\ &= \sum_{k=1}^{M} \eta_k(x) \gamma \left(\sum_{l=1, l\neq k}^{M} \phi(u^k(x) - u^l(x)) \right). \end{aligned} \tag{205}$$

The derivative of $R_{L_M^{\gamma-\phi}}(f|x)$ with respect to $f(x)$ is

$$\begin{aligned} \frac{\partial R_{L_M^{\gamma-\phi}}(f|x)}{\partial f(x)} &= \frac{\partial}{\partial f(x)} \sum_{k=1}^{M} \eta_k \gamma \left(\sum_{l=1, l\neq k}^{M} \phi(u^k - u^l) \right) \\ &= \frac{1}{2} \sum_{k=1}^{M} \left\{ \eta_k \gamma' \left(\sum_{l\neq k} \phi(u^k - u^l) \right) \sum_{l\neq k} \phi'(u^k - u^l)[y^k - y^l] \right\}. \end{aligned} \tag{206}$$

Defining

$$\pi_k = \gamma' \left(\sum_{l\neq k} \phi(u^k - u^l) \right), \tag{207}$$

results in

$$\begin{aligned} \frac{\partial R_{L_M^{\gamma-\phi}}(f|x)}{\partial f(x)} &= \frac{1}{2} \sum_{k=1}^{M} \left\{ \eta_k \pi_k \sum_{l\neq k} \phi'(u^k - u^l)[y^k - y^l] \right\} \\ &= \frac{1}{2} \sum_{l,k|k\neq l} \eta_k \pi_k \phi'(u^k - u^l)[y^k - y^l] \\ &= \frac{1}{2} \sum_{l,k|k\neq l} y^k \eta_k \pi_k \phi'(u^k - u^l) - \frac{1}{2} \sum_{l,k|k\neq l} y^l \eta_k \pi_k \phi'(u^k - u^l) \\ &= \frac{1}{2} \sum_{j,k|k\neq j} y^k \eta_k \pi_k \phi'(u^k - u^j) - \frac{1}{2} \sum_{l,k|k\neq l} y^k \eta_l \pi_l \phi'(u^l - u^k) \\ &= \frac{1}{2} \sum_{k=1}^{M} y^k \eta_k \pi_k \sum_{j\neq k} \phi'(u^k - u^j) - \frac{1}{2} \sum_{k=1}^{M} y^k \sum_{l\neq k} \eta_l \pi_l \phi'(u^l - u^k) \\ &= \frac{1}{2} \sum_{k=1}^{M} y^k \left[\eta_k \pi_k \sum_{j\neq k} \phi'(u^k - u^j) - \sum_{l\neq k} \eta_l \pi_l \phi'(u^l - u^k) \right] \\ &= \frac{1}{2} \mathbf{Y} \mathbf{Q}_f^{\phi} \Gamma_f^{\gamma-\phi} \eta. \end{aligned} \tag{208}$$

where $\mathbf{Y}$ is the code matrix of (94), $\mathbf{Q}_f^{\phi}$ as defined in (97) and $\Gamma_f^{\gamma-\phi}$ as defined in (98). Hence, (96) holds for any minimizer of $R_{L_M^{\gamma-\phi}}(f|x)$.

G.2. Proof of Lemma 3

1. If γ is strictly monotonic then, from (98), $\mathbf{\Gamma}_f^{\gamma-\phi}$ is full rank and the statement follows.

2. Let $Rank(\mathbf{Y}) < M - 1$ and consider two possibilities.

   (a) The set $Null(\mathbf{Y}) \bigcap Range(\mathbf{Q}_f^\phi)$ is empty. In this case, since $Null(\mathbf{Y}) \bigcup Range(\mathbf{Q}_f^\phi) \subset \mathbb{R}^M$ and

$$\begin{aligned} M &\geq |Null(\mathbf{Y})| + |Range(\mathbf{Q}_f^\phi)| \\ &= M - Rank(\mathbf{Y}) + |Range(\mathbf{Q}_f^\phi)| \\ &\geq 2 + |Range(\mathbf{Q}_f^\phi)|. \end{aligned} \tag{209}$$

   Since $\mathbf{Q}_f^\phi \in \mathbb{R}^{M\times M}$, it follows that $|Null(\mathbf{Q}_f^\phi)| \geq 2$. Since $Null(\mathbf{Q}_f^\phi) \subset Null(\mathbf{Y}\mathbf{Q}_f^\phi)$, it follows that $|Null(\mathbf{Y}\mathbf{Q}_f^\phi)| \geq 2$.

   (b) The set $Null(\mathbf{Y}) \bigcap Range(\mathbf{Q}_f^\phi)$ is non-empty. Hence, there is at least one vector $v_1 \in Null(\mathbf{Y}) \bigcap Range(\mathbf{Q}_f^\phi)$. Since $v_1 \in Range(\mathbf{Q}_f^\phi)$, there exists a vector v_2 such that $\mathbf{Q}_f^\phi v_2 = v_1$, i.e. $v_2 \notin Null(\mathbf{Q}_f^\phi)$. Since $v_1 \in Null(\mathbf{Y})$, it follows that $\mathbf{Y}\mathbf{Q}_f^\phi v_2 = 0$ and $v_2 \in Null(\mathbf{Y}\mathbf{Q}_f^\phi)$. On the other hand, it follows from (97) that the rows of $\mathbf{Q}_f^\phi$ sum to zero and

$$|Null(\mathbf{Q}_f^\phi)| \geq 1. \tag{210}$$

   Hence, there is at least a vector $v_0 \neq 0$ such that $\mathbf{Q}_f^\phi v_0 = 0$. It follows that $v_0 \in Null(\mathbf{Y}\mathbf{Q}_f^\phi)$. In summary, there is a vector $v_0 \in Null(\mathbf{Q}_f^\phi)$ and a vector $v_2 \notin Null(\mathbf{Q}_f^\phi)$ such that $v_0, v_2 \in Null(\mathbf{Y}\mathbf{Q}_f^\phi)$. Hence, $|Null(\mathbf{Y}\mathbf{Q}_f^\phi)| \geq 2$.

   By combination of the two possibilities it follows that $|Null(\mathbf{Y}\mathbf{Q}_f^\phi)| \geq 2$.

3. Since $\mathbf{Y} \in \mathbb{R}^{d\times M}$, $Rank(\mathbf{Y}) \leq M$. If $Rank(\mathbf{Y}) = M$, then the null space of $\mathbf{Y}$ contains only the origin. It follows that a vector is in the null space of $\mathbf{Y}\mathbf{Q}_f^\phi$ if and only if it is also on the null space of $\mathbf{Q}_f^\phi$. Assume that $Rank(\mathbf{Y}) = M - 1$. Since $\mathbf{Y} \in \mathbb{R}^{d\times M}$, $|Null(\mathbf{Y})| = 1$. Since the codewords are centered, as in (55), $\mathbf{Y}\mathbf{1} = 0$ and $Null(\mathbf{Y}) = Range(\mathbf{1})$, i.e. the null space of $\mathbf{Y}$ is spanned by the vector $\mathbf{1}$. Hence, for any $\eta \in Null(\mathbf{Y}\mathbf{Q}_f^\phi)$ there is a scalar λ such that

$$\mathbf{Q}_f^\phi \eta = \lambda \mathbf{1}. \tag{211}$$

   Denoting by r_Q^i the i^{th} row of $\mathbf{Q}_f^\phi$, it follows that $< r_Q^i, \eta >= \lambda$ and $< \sum_{i=1}^M r_Q^i, \eta >= M\lambda$. Since, from (97), $\sum_{i=1}^M r_Q^i = 0$ it follows that $\lambda = 0$. Hence, $\eta \in Null(\mathbf{Q}_f^\phi)$ and $Null(\mathbf{Y}\mathbf{Q}_f^\phi) \subset Null(\mathbf{Q}_f^\phi)$. The statement follows from the fact that $Null(\mathbf{Q}_f^\phi) \subset Null(\mathbf{Y}\mathbf{Q}_f^\phi)$ always holds.

4. Let $\overline{\mathbf{Q}}_f^\phi \in \mathbb{R}^{M-1\times M-1}$ be the matrix obtained by eliminating the first row and column of $\mathbf{Q}_f^\phi$. From (97) and strict monotonicity of ϕ

$$\begin{aligned} |\mathbf{Q}_f^\phi(k,k)| &= \left|\sum_{j\neq k}\phi'(u^k-u^j)\right| \\ &= \sum_{j\neq k}|\phi'(u^k-u^j)|, \end{aligned} \tag{212}$$

and

$$\begin{aligned} |\overline{\mathbf{Q}}_f^\phi(k,k)| &= \sum_{j\neq k+1}|\phi'(u^{k+1}-u^j)| \\ &= \sum_{j\neq k+1}|\mathbf{Q}_f^\phi(j,k+1)| \\ &= |\mathbf{Q}_f^\phi(1,k+1)| + \sum_{j\neq k}|\overline{\mathbf{Q}}_f^\phi(j,k)| \\ &> \sum_{j\neq k}|\overline{\mathbf{Q}}_f^\phi(j,k)|. \end{aligned} \tag{213}$$

Therefore $\overline{\mathbf{Q}}_f^\phi$ is strictly diagonally dominant and thus non-singular (Horn and Johnson, 1986). It follows that rows $r_Q^i, i=2,\ldots,M$ of $\mathbf{Q}_f^\phi$ are linearly independent and $Rank(\mathbf{Q}_f^\phi) \geq M-1$. Since, from (97),

$$r_Q^1 = -\sum_{i=2}^{M} r_Q^i, \tag{214}$$

it follows that $Rank(\mathbf{Q}_f^\phi) = M-1$. Hence, $|Null(\mathbf{Q}_f^\phi)| = 1$.

G.3. Proof of Lemma 4

From (97), (98) and (100), $\mathbf{Q}_f^\phi\Gamma_f^{\gamma-\phi}\eta = 0$ is equivalent to

$$\begin{aligned} 0 &= \eta_k\pi_k\sum_{j\neq k}\phi'(u^k-u^j) - \sum_{l\neq k}\eta_l\pi_l\phi'(u^l-u^k) \\ &= \eta_k\pi_k\sum_{j\neq k}\phi'(u^k-u^j) + \eta_k\pi_k\phi'(u^k-u^k) \\ &- \eta_k\pi_k\phi'(u^k-u^k) - \sum_{l\neq k}\eta_l\pi_l\phi'(u^l-u^k) \\ &= \eta_k\pi_k\sum_{j=1}^{M}\phi'(u^k-u^j) - \sum_{l=1}^{M}\eta_l\pi_l\phi'(u^l-u^k), \end{aligned} \tag{215}$$

and (99) follows. The proof that all η_k have the same sign is by contradiction. Assume the η_j have different signs and let $\mathcal{N},\mathcal{P}$ be sets of positive and negative indices, i.e. $\eta_j < 0\ \forall j \in \mathcal{N}$, $\eta_j \geq 0\ \forall j \in \mathcal{P}$. Define

$$\zeta_{k,l} = \pi_k\phi'(u^k-u^l), \tag{216}$$

with π_k as in (100). Since ϕ, γ are strictly monotonic, all $\zeta_{k,l}$'s are non-zero and have the same sign. Without loss of generality, we assume they are positive. From (99),

$$\eta_k \sum_{l=1}^{M} \zeta_{k,l} - \sum_{l=1}^{M} \eta_l \zeta_{l,k} = 0. \tag{217}$$

Adding over all $k \in \mathcal{N}$,

$$\begin{aligned}
0 &= \sum_{k\in\mathcal{N}} \left\{ \eta_k \sum_{l=1}^{M} \zeta_{k,l} - \sum_{l=1}^{M} \eta_l \zeta_{l,k} \right\} \\
&= \sum_{k\in\mathcal{N}} \eta_k \left\{ \sum_{l\in\mathcal{N}} \zeta_{k,l} + \sum_{l\in\mathcal{P}} \zeta_{k,l} \right\} - \sum_{k\in\mathcal{N}} \left\{ \sum_{l\in\mathcal{N}} \eta_l \zeta_{l,k} + \sum_{l\in\mathcal{P}} \eta_l \zeta_{l,k} \right\} \\
&= \sum_{k\in\mathcal{N}, l\in\mathcal{N}} \eta_k \zeta_{k,l} + \sum_{k\in\mathcal{N}, l\in\mathcal{P}} \eta_k \zeta_{k,l} - \sum_{k\in\mathcal{N}, l\in\mathcal{N}} \eta_l \zeta_{l,k} - \sum_{k\in\mathcal{N}, l\in\mathcal{P}} \eta_l \zeta_{l,k} \\
&= \sum_{k\in\mathcal{N}, l\in\mathcal{P}} \eta_k \zeta_{k,l} - \sum_{k\in\mathcal{N}, l\in\mathcal{P}} \eta_l \zeta_{l,k}
\end{aligned} \tag{218}$$

and

$$\sum_{k\in\mathcal{N}, l\in\mathcal{P}} \eta_k \zeta_{k,l} = \sum_{k\in\mathcal{P}, l\in\mathcal{N}} \eta_k \zeta_{k,l}. \tag{219}$$

Since by definition of the sets $\mathcal{N}, \mathcal{P}$, the two sides of this equation have opposite signs, we have a contradiction unless one of the sets $\mathcal{N}, \mathcal{P}$ is empty.

G.4. Proof of theorem 7

From Lemmas 3 and 4, since $Rank(\mathbf{Y}) \geq M-1$ and (ϕ, γ) strictly monotonic, the solution of (96) is identical to that of $\mathbf{Q}_f^{\phi}\mathbf{\Gamma}_f^{\gamma-\phi}\eta = 0$ and, up to a normalization constant, a probability vector. Defining $\mathbf{\Psi}_f^{\gamma-\phi} = \mathbf{Q}_f^{\phi}\mathbf{\Gamma}_f^{\gamma-\phi}$, it follows that the system of equations

$$\left\{ \begin{array}{l} \mathbf{\Psi}_f^{\gamma-\phi}\eta = 0 \\ \mathbf{1}^T\eta = 1, \end{array} \right. \tag{220}$$

has a unique solution. Furthermore, it follows from the monotonicity of γ that $Rank(\mathbf{\Gamma}_f^{\gamma-\phi}) = M$ and from (97) that $Rank(\mathbf{Q}_f^{\phi}) = M-1$. Hence, $Rank(\mathbf{\Psi}_f^{\gamma-\phi}) = M-1$, i.e. the first row of $\mathbf{\Psi}_f^{\gamma-\phi}$ is a linear combination of the other rows and can be removed from (220). The system of equations of (220) can thus be written as

$$\mathbf{\Omega}_f^{\gamma-\phi}\eta = \mathbf{e}_1, \tag{221}$$

where

$$\mathbf{\Omega}_f^{\gamma-\phi}(k,l) = \left\{ \begin{array}{ll} 1 & k = 1, \forall l \\ \mathbf{\Psi}_f^{\gamma-\phi}(k,l) & \text{otherwise.} \end{array} \right. \tag{222}$$

Since (220) has a unique solution, $\mathbf{\Omega}_f^{\gamma-\phi}$ is invertible and $\eta = \left(\mathbf{\Omega}_f^{\gamma-\phi}\right)^{-1}\mathbf{e}_1$.

Appendix H. MCBoost vs previous multiclass boosting algorithms

In this appendix we compare MCBoost to multiclass learning algorithms in the literature. We start by showing that many of the multiclass boosting algorithms are special instances of MCBoost using different, and sometimes sub-optimal, codewords, weak learners, loss functions, or optimization procedures.

H.1. Coordinate descent algorithms

Several previous learning algorithms can be seen as coordinate descent optimization procedures and are thus closely related to CD-MCBoost.

H.1.1. ONE-VS-ALL CLASSIFIERS

Multiclass classifiers are frequently implemented with the one-vs-all architecture (OVA), where M predictors $\bar{f}_k(x)$ are learned independently. Predictor $\bar{f}_k(x)$ discriminates between class k and all other classes. For predictors learned with AdaBoost, $\bar{f}_k(x)$ is learned by minimizing the risk

$$\overline{R}_k = \frac{1}{n}\sum_{i=1}^{n} e^{-\bar{z}_i^k \bar{f}_k(x_i)}, \tag{223}$$

where

$$\bar{z}_i^k = \begin{cases} 1 & \text{if } c_i = k \\ -1 & \text{otherwise.} \end{cases} \tag{224}$$

The M OVA predictors are combined into a multiclass predictor $\bar{f}(x) = [\bar{f}_1(x), \dots, \bar{f}_M(x)] \in \mathbb{R}^M$, which is used to implement the decision rule $\arg\max_k \bar{f}_k(x)$.

Since this predictor is learned in a coordinate-wise manner, OVA is closest to CD-MCBoost. In fact, the individual risks of (223) can be combined into

$$\begin{aligned} \overline{R} &= \sum_k \overline{R}_k = \frac{1}{n}\sum_{i=1}^{n}\sum_k e^{-\bar{z}_i^k \bar{f}_k(x_i)} \\ &= \frac{1}{n}\sum_{i=1}^{n}\left\{ e^{-\bar{f}_{c_i}(x_i)} + \sum_{k\neq c_i} e^{\bar{f}_k(x_i)} \right\}, \end{aligned}$$

which is the risk of (21) for the loss

$$L_{ova}[y^{c_i}, \bar{f}(x_i)] = e^{-\langle \bar{y}^{c_i}, \bar{f}(x_i)\rangle} + \sum_{k\neq c_i} e^{\langle \bar{y}^k, \bar{f}(x_i)\rangle}, \tag{225}$$

under the choice of canonical codewords $\bar{y}^j = \mathbf{1}_j \in \mathbb{R}^M$. Hence, the fundamental differences between OVA and CD-MCBoost are the use, by the former, of 1) canonical codewords and 2) the loss of (225). This combination is sub-optimal for two reasons. First, the minimization of the resulting risk decouples into M *independent* problems for learning M coordinate classifiers. This tends to produce weaker decision rules than the joint learning of all coordinate predictors. Second, the loss of (225) is not guess-averse (Beijbom et al., 2014). This can easily lead to sub-optimal classifiers.

H.1.2. ADABOOST.ECC

AdaBoost.ECC (Guruswami and Sahai, 1999) is a multiclass boosting algorithm based on the the error correcting output coding (ECOC) strategy of (Dietterich and Bakiri, 1995). It relies on binary-valued codewords and the $\gamma - \phi$ loss of (23) with $\phi(v) = e^{-2v}$ and $\gamma(v) = v$. It is initialized with an empty codeword set. At each iteration t, 1) each codeword is augmented with a new bit and 2) a corresponding predictor dimension $\bar{f}_t$ is learned. After T iterations, the algorithm learns a set of T-dimensional binary codewords $\bar{y}^j \in \{+1, -1\}^T$ and a T-dimensional predictor $\bar{f}(x) = [\bar{f}_1(x), \ldots, \bar{f}_T(x)] \in \mathbb{R}^T$. The decision rule is

$$\bar{F}(x) \quad \equiv \quad \arg \max_{j \in \{1,\ldots,M\}} \left\langle \bar{f}(x), \bar{y}^j \right\rangle. \tag{226}$$

Since the multi-dimensional predictor is learned in a coordinate-wise manner, AdaBoost.ECC is closest to CD-MCBoost. The main difference is in the definition of the codewords. While CD-MCBoost uses real-valued codewords selected a priori so as to maximize the margin capacity of (56), AdaBoost.ECC codewords are generated on-the-fly, by either random selection or the solution of a "max-cut" problem. This is sub-optimal for two reasons. First, the limitation to binary codewords can lead to codeword sets of margin capacity below the bound of (59). This limits the margin maximizing ability of the learning algorithm and degrades its generalization. Second, the codeword length increases at each iteration of the algorithm. Since this increases predictor dimensionality, it can easily lead to high computational costs and over-fitting. Finally, the procedures used to learn the codewords do not guarantee a codeword set optimal for classification. These limitations extend to a number of other methods based on the ECOC strategy (Schapire, 1997; Li; Zhang et al., 2009; Gao and Koller, 2011).

H.1.3. MULTICLASS LOGITBOOST

Multiclass LogitBoost (Friedman et al., 1998) is a multiclass boosting method that learns an M-dimensional predictor $\bar{f}(x) = [\bar{f}_1(x), \ldots, \bar{f}_M(x)] \in \mathbb{R}^M$ where each coordinate is a linear regressor

$$\bar{f}_j(x) = ax + b, \tag{227}$$

for some $a, b \in \mathbb{R}$. It is assumed that

$$\sum_{k=1}^{M} \bar{f}_k(x) = 0. \tag{228}$$

The predictor is learned by minimizing the negative log likelihood,

$$L_{Logit}[c_i, \bar{f}(x_i)] \quad = \quad -\log \left[\bar{P}_{C,X}(c_i|x_i)\right], \tag{229}$$

with probabilities defined as

$$\bar{P}_{C,X}(c_i|x_i) = \frac{e^{\bar{f}_{c_i}(x_i)}}{\sum_{j=1}^{M} e^{\bar{f}_j(x_i)}}. \tag{230}$$

Each iteration of the algorithm computes the best regression update for all coordinates, using Newton's method. These updates are then centered to satisfy (228) and added to the multi-dimensional predictor.

To compare this to MCBoost note that, from (229)-(230),

$$\begin{aligned} L_{Logit}[c_i, \bar{f}(x_i)] &= -\log\left[\frac{e^{\bar{f}_{c_i}(x_i)}}{\sum_{j=1}^M e^{\bar{f}_j(x_i)}}\right] \\ &= \log\left[\frac{\sum_{j=1}^M e^{\bar{f}_j(x_i)}}{e^{\bar{f}_{c_i}(x_i)}}\right] \\ &= \log\left[1+\sum_{j\neq c_i} e^{\bar{f}_j(x_i)-\bar{f}_{c_i}(x_i)}\right] \\ &= \log\left[1+\sum_{j\neq c_i} e^{-\left[\langle \bar{y}^{c_i},\bar{f}(x_i)\rangle - \langle \bar{y}^j,\bar{f}(x_i)\rangle\right]}\right], \end{aligned} \quad (231)$$

with $\bar{y}^j = \mathbf{1}_j \in \mathbb{R}^M$. Since this is the special case of (23) with $\phi(v) = e^{-2v}$ and $\gamma(v) = \log(1+v)$, multiclass LogitBoost is an implementation of CD-MCBoost with the logistic $\gamma - \phi$ loss, canonical codewords, and regression weak learners. The only difference is that LogitBoost relies on the Newton method for finding the best update per iteration while CD-MCBoost uses gradient descent updates. Although it is possible to use Newton updates in MCBoost (Saberian et al., 2010), this is beyond the scope of the current manuscript. Finally, note that the analysis above holds for a number of algorithms inspired by multiclass LogitBoost (Huang et al., 2007; Zou et al., 2008).

H.2. Gradient descent algorithms

Several previous learning algorithms can be seen as gradient descent optimization procedures and are thus closely related to GD-MCBoost.

H.2.1. ADABOOST.M1

AdaBoost.M1 is the first multiclass Boosting algorithm introduced by (Freund and Schapire, 1996). It relies on multiclass weak learners $\hat{g}(x) : \mathcal{X} \to \{1, \dots, M\}$ and the decision rule

$$\bar{F}(x) = \arg\max_{j\in\{1,\dots,M\}} \hat{f}_j(x_i), \quad (232)$$

where

$$\hat{f}_j(x_i) = \sum_t \alpha_t I(\hat{g}_t(x_i) = j), \quad (233)$$

and $I(.)$ is the indicator function. AdaBoost.M1 is initialized with a uniform weight distribution over the training sample, e.g. $w_i = 1$. At iteration t, it selects the weak learner $\hat{g}_t$ of lowest weighted error rate

$$\begin{aligned} e_t^* &= \min_g \sum_i w_i[1 - I(g(x_i) = c_i)] \quad (234) \\ &= \max_g \sum_i w_i I(g(x_i) = c_i). \quad (235) \end{aligned}$$

The algorithm stops if $e_t^* > 50\%$, otherwise the weak learner is added to the ensemble with

$$\alpha_t = \log\left(\frac{1-e_t^*}{e_t^*}\right), \tag{236}$$

and the weights are updated according to

$$w_i = w_i \times e^{-\alpha_t I(\hat{g}_t(x_i)=c_i)} = e^{-\sum_t \alpha_t I(\hat{g}_t(x_i)=c_i)}. \tag{237}$$

In Appendix E.2.3 we show that (233), (235), and (237) are the equations of a GD-MCBoost algorithm for the minimization of the risk defined by the loss

$$L_{Ada.M1}[\bar{y}^{c_i}, \bar{f}(x_i)] = e^{-\langle \bar{y}^{c_i}, \bar{f}(x_i)\rangle} \tag{238}$$

with $\bar{y}^j = \mathbf{1}_j \in \mathbb{R}^M$ and $\bar{f} = [\hat{f}_1, \ldots, \hat{f}_M]$ where $\hat{f}_j$ is given by (233). Hence, AdaBoost.M1 is a descent algorithm using the loss of (238), canonical codewords, and classification weak learners. While this has some similarity with GD-MCBoost, there are important differences. First, (238) is not a $\gamma - \phi$ loss. Second, the constraint $\alpha_t > 0$ requires the existence of a weak learner with error rate smaller than 50% for any weight distribution. As discussed above, this is usually too stringent for problems with large M. In practice, AdaBoost.M1 frequently terminates too early, due to the impossibility of finding such a weak learner (Zhu et al., 2009).

H.2.2. SAMME

SAMME (Zhu et al., 2009) is a multiclass Boosting algorithm explicitly introduced to address this problem. It learns an M-dimensional predictor $\bar{f} = [\bar{f}_1, \ldots, \bar{f}_M] \in \mathbb{R}^M$ such that $\sum_{k=1}^M \bar{f}_k(x) = 0$, using codewords

$$\bar{y}^j = \frac{M\mathbf{1}_j - \mathbf{1}}{M-1} = \left[\frac{-1}{M-1}, \frac{-1}{M-1}, \ldots, 1, \frac{-1}{M-1}, \frac{-1}{M-1}\right] \in \mathbb{R}^M, \tag{239}$$

and multiclass classification weak learners $\hat{g}(x) : \mathcal{X} \to \{\bar{y}^1 \ldots \bar{y}^M\}$. The boosting algorithm is derived explicitly to minimize the loss

$$L_{SAMME}[\bar{y}^{c_i}, f(x_i)] = e^{-\frac{1}{M}\langle \bar{y}^{c_i}, \bar{f}(x_i)\rangle}. \tag{240}$$

using updates of the form of (33) and a decision rule

$$\begin{aligned}\bar{F}(x) &= \arg\max_{j\in\{1,\ldots,M\}} \bar{f}_j(x) \qquad (241)\\ &\equiv \arg\max_{j\in\{1,\ldots,M\}} \langle \bar{y}^j, \bar{f}(x)\rangle \qquad (242)\end{aligned}$$

with $\bar{y}^j = \mathbf{1}_j \in \mathbb{R}^M$. Appendix E.2.2 shows that this leads to the weak learner selection rule of (235), making the updates of SAMME nearly identical to those of AdaBoost.M1. The only difference is the step size α_t. (Zhu et al., 2009) have shown that the optimal step is

$$\alpha_t = \log\left(\frac{1-e_t^*}{e_t^*}\right) + \log(M-1). \tag{243}$$

When compared to (236), the addition of the term $\log(M-1)$ enables the algorithm to continue running when $e_t^* < 50\%$. This substantially loosens the boostability condition, enabling the use of much weaker weak learners. Comparing to GD-MCBoost, SAMME is a gradient descent procedure, uses a maximum capacity codeword set, and classification weak learners. There are, however, two main differences. The first is the use of the exponential loss of (240) instead of the $\phi - \gamma$ loss of (23). These losses are compared in Section 8.1. The second is the use of different codeword sets for learning the predictor (maximum capacity) and computing the decision rule (canonical). This inconsistency is likely to degrade classification accuracy.

H.2.3. ADABOOST.M2 AND ADABOOST.MR

AdaBoost.MR (Schapire and Singer, 1999) is a multiclass boosting algorithm that supports multiple labels per example. Its single-label version is AdaBoost.M2 (Freund and Schapire, 1996). Like SAMME, it was introduced to increase the boostability of AdaBoost.M1. The extension is, however, of a different nature. The idea behind AdaBoost.MR and AdaBoost.M2 is to define a distribution over mislabels and a pseudo-loss of weak learner h_t with respect to this distribution. The weak learners are defined on the product space of input and class labels

$$g_t : \mathcal{X} \times \{1, \ldots, M\} \rightarrow \mathbb{R} \tag{244}$$

and denoted *product weak learners*. The goal is to learn a decision rule

$$\bar{F}(x) = \arg \max_{j \in \{1,\ldots,M\}} \sum_t \alpha_t \hat{g}_t(x, j). \tag{245}$$

AdaBoost.MR defines a *weight per class* for each training example, $w(x_i, k) \quad k = 1, \ldots, M$. This can be seen as a weight distribution over classes and is initially uniform, i.e. $w(x_i, k) = 1, \forall k$. At each iteration, the weak learner $\hat{g}_t(.,.)$ of lowest pseudo loss

$$e_t = \frac{1}{2} \sum_{(i,y) \in B} w(x_i, y)(1 - g(x_i, y_i) + g(x_i, y)), \tag{246}$$

where $B = \{(i, y) | y \neq y_i\}$ is the set of mislabels, is selected and added to the current predictor with coefficient

$$\alpha_t = \log\left(\frac{1 - e_t}{e_t}\right). \tag{247}$$

The weights are updated according to

$$w(x_i, k) = w(x_i, k) \times e^{-\frac{\alpha_t}{2}[\hat{g}_t(x_i,y_i) - \hat{g}_t(x_i,k)]}. \tag{248}$$

To compare AdaBoost.M2 to MCBoost, we start by defining the class-specific predictors

$$\hat{f}_j = \sum_t \alpha_t \hat{g}_t(x, j), \tag{249}$$

the multi-dimensional predictor $\bar{f}(x) = [\hat{f}(x_i, 1), \ldots, \hat{f}(x_i, M)]$ and the multi-dimensional weak learner $g(x) = [g(x, 1), \ldots, g(x, M)]$. The decision rule of (245) is then equivalent to

$$\bar{F}(x) = \arg \max_{j \in \{1,\ldots,M\}} \left\langle \bar{y}^j, \bar{f}(x) \right\rangle, \tag{250}$$

with canonical codewords $\bar{y}^j = \mathbf{1}_j \in \mathbb{R}^M$ and the weights of (248) reduce to

$$w(x_i, j) = e^{-\frac{1}{2}\left[\langle \bar{y}^{c_i}, \bar{f}(x_i)\rangle - \langle \bar{y}^j, \bar{f}(x_i)\rangle\right]}. \tag{251}$$

The pseudo-loss of (246) can also be written as

$$\begin{aligned} e_t &= \frac{1}{2}\sum_i \sum_{j\neq c_i} w(x_i, j)(1 - \langle g(x_i), \bar{y}^{c_i}\rangle + \langle g(x_i), \bar{y}^j\rangle) \\ &= \frac{1}{2}\sum_i w_i \left(1 - \sum_{j\neq c_i} \frac{w(x_i, j)}{w_i} \langle g(x_i), \bar{y}^{c_i} - \bar{y}^j\rangle\right) \\ &= \frac{1}{2}\sum_i w_i \left(1 - \left\langle g(x_i), \bar{y}^{c_i} - \sum_{j\neq c_i} \bar{y}^j \frac{w(x_i, j)}{w_i}\right\rangle\right), \end{aligned} \tag{252}$$

with

$$w_i = \sum_{j\neq c_i} w(x_i, j). \tag{253}$$

Hence, the weak learner selected by AdaBoost.M2 is

$$\hat{g}_t \propto \arg\max_g \frac{1}{2}\sum_i w_i \left\langle g(x_i), \bar{y}^{c_i} - \sum_{j\neq c_i} \bar{y}^j \frac{w(x_i, j)}{w_i}\right\rangle. \tag{254}$$

Since the combination of (251), (253), and (254) is the special case of (29)-(32) with $\phi(v) = e^{-v}$ and $\gamma(v) = v$, AdaBoost.M2 is conceptually equivalent to GD-MCBoost with the exponential $\gamma - \phi$ loss of Table 3, canonical codewords, and the product weak learners of (244). In practice, however, the algorithms can behave very differently, due to the replacement of the MCBoost weak learners of (26) by the product learners of (244). Note that the latter augment training examples with their class identities, to form a new set of training examples $\bar{x} = [x, j]\ \forall j$. This is problematic because, besides increasing the size of the of training set M-fold, it creates many examples that differ by only one coordinate. These examples can be difficult to discriminate with the simple functions typically used to implement weak learners, e.g. decision stumps. These problems make AdaBoost.MR computationally intensive and very slow to converge. The same problem arises for AdaBoost.MH (Schapire and Singer, 1999) which also uses the weak learners of (244).

H.2.4. ADABOOST.MH

AdaBoost.MH is very similar to AdaBoost.MR. The only difference is the weight update stage, where (248) is replaced by

$$w(x_i, k) = w(x_i, k) \times e^{-\alpha_t z_{c_i,k}\hat{g}_t(x_i,k)}, \tag{255}$$

with $z_{c_i,k}$ as defined in (224). Using the procedure above, it can be shown that AdaBoost.MH minimizes the one-vs-all loss. This leads to a combination of the problems associated with the weak learners of (244) and the loss function of (225), discussed in Section H.1.1.

H.2.5. ADABOOST.MM

(Mukherjee and Schapire, 2013) proposed a framework for the characterization of multiclass boostability conditions and used it to motivate the AdaBoost.MM algorithm. This a is multiclass boosting method that uses classification weak learners $\hat{g}(x) : \mathcal{X} \to \{1, 2, \ldots, M\}$ to learn a predictor $\hat{f}(x) = \sum_t \alpha_t \hat{g}_t(x)$. The decision rule is

$$\bar{F}(x) = \arg \max_{j \in \{1,\ldots,M\}} \hat{f}(x_i, j), \tag{256}$$

where

$$\hat{f}(x_i, j) = \sum_t \alpha_t I(\hat{g}_t(x_i) = j), \tag{257}$$

and $I(.)$ is the indicator function. Each round of AdaBoost.MM computes the cost of assigning example i to class j according to

$$C_{i,j} = \begin{cases} e^{\hat{f}(x_i,j) - \hat{f}(x_i,c_i)} & \text{if } j \neq c_i \\ -\sum_{l \neq c_i} e^{\hat{f}(x_i,l) - \hat{f}(x_i,c_i)} & \text{if } j = c_i \end{cases}, \tag{258}$$

and selects the weak learner of lowest cost

$$g^* = \arg \min_{\hat{g}} \sum_i C_{i,\hat{g}(x_i)}. \tag{259}$$

AdaBoost.MM can be related to MCBoost by considering multidimensional weak learners of canonical output, i.e.

$$\bar{g}(x_i) \quad = \quad \mathbf{1}_{\hat{g}(x_i)}, \tag{260}$$

and a predictor

$$\bar{f}(x) \quad = \quad \sum_t \alpha_t \bar{g}_t(x). \tag{261}$$

Under these definitions, (257) can be written as

$$\hat{f}(x_i, j) = \left\langle \bar{f}(x_i), \bar{y}^j \right\rangle, \tag{262}$$

where

$$\bar{y}^j \quad = \quad \mathbf{1}_j \in \mathbb{R}^M, \tag{263}$$

is the canonical codeword set and the decision rule of (256) is equivalent to

$$\bar{F}(x) = \arg \max_{j \in \{1,\ldots,M\}} \left\langle \bar{y}^j, \bar{f}(x) \right\rangle. \tag{264}$$

Similarly, the costs of (258) can be written as

$$C_{i,j} = \begin{cases} e^{\left\langle \bar{f}(x_i), \bar{y}^j \right\rangle - \left\langle \bar{f}(x_i), \bar{y}^{c_i} \right\rangle} & \text{if } j \neq c_i \\ -\sum_{l \neq c_i} e^{\left\langle \bar{f}(x_i), \bar{y}^l \right\rangle - \left\langle \bar{f}(x_i), \bar{y}^{c_i} \right\rangle} & \text{if } j = c_i \end{cases}, \tag{265}$$

and the weak learner selection rule as

$$g^* = \arg\min_{\hat{g}} \sum_i \langle C_i, \bar{g}(x_i) \rangle \,, \tag{266}$$

where $C_i \in \mathbb{R}^M$ is the vector of components $C_{i,j}$. Using (263) and (265), this can be written as

$$\begin{aligned} C_i &= \sum_{j \neq c_i} \bar{y}^j e^{\langle \bar{y}^j, \bar{f}(x_i) \rangle - \langle \bar{y}^{c_i}, \bar{f}(x_i) \rangle} - \bar{y}^{c_i} \sum_{j \neq c_i} e^{\langle \bar{y}^j, \bar{f}(x_i) \rangle - \langle \bar{y}^{c_i}, \bar{f}(x_i) \rangle} \\ &= \sum_{j \neq c_i} (\bar{y}^j - \bar{y}^{c_i}) e^{-[\langle \bar{y}^{c_i}, \bar{f}(x_i) \rangle - \langle \bar{y}^j, \bar{f}(x_i) \rangle]}. \end{aligned} \tag{267}$$

It follows from (266) that (259) is equivalent to

$$\begin{aligned} g^* &= \arg\min_{\hat{g}} \sum_i \sum_{j \neq c_i} \langle \bar{g}(x_i), \bar{y}^j - \bar{y}^{c_i} \rangle \, e^{-[\langle \bar{y}^{c_i}, \bar{f}(x_i) \rangle - \langle \bar{y}^j, \bar{f}(x_i) \rangle]} \\ &= \arg\max_{\hat{g}} \sum_i w_i \left\langle \bar{g}(x_i), \bar{y}^{c_i} - \sum_{j \neq c_i} \bar{y}^j \frac{w(x_i, j)}{w_i} \right\rangle, \end{aligned} \tag{268}$$

with

$$\begin{aligned} w(x_i, j) &= e^{-[\langle \bar{y}^{c_i}, \bar{f}(x_i) \rangle - \langle \bar{y}^j, \bar{f}(x_i) \rangle]} \\ w_i &= \sum_{j \neq c_i} w(x_i, j), \end{aligned} \tag{269}$$

which is equivalent to the GD-MCBoost weak learner selection rule of (29) for $\phi(v) = e^{-2v}$ and $\gamma(v) = v$. Hence, AdaBoost.MM is an implementation of GD-MCBoost with the canonical codeword set $\mathbf{1}_j$, classification weak learners, and a $\gamma - \phi$ loss with $\phi(v) = e^{-2v}$ and $\gamma(v) = v$. The only difference is that each iteration of GD-MCBoost finds the optimal step size through a line search. On the other hand, AdaBoost.MM relies on heuristics to determine the step size. This can be sub-optimal.