Game theory and optimization in boosting

Indraneel Mukherjee

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of
Computer Science
Advisor: Robert E. Schapire

November 2011

Abstract

Boosting is a central technique of machine learning, the branch of artificial intelligence concerned with designing computer programs that can build increasingly better models of reality as they are presented with more data. The theory of boosting is based on the observation that combining several models with low predictive power can often lead to a significant boost in the accuracy of the combined meta-model. This approach, introduced about twenty years ago, has been a prolific area of research, and has proved immensely successful in practice. However, despite extensive work, many basic questions about boosting remain unanswered. In this thesis, we increase our understanding of three such theoretical aspects of boosting.

In Chapter 2 we study the convergence properties of the most well known boosting algorithm, AdaBoost. Rate bounds for this important algorithm are known for only special situations that rarely hold in practice. Our work guarantees fast rates hold under all situatons, and the bounds we provide are optimal. Apart from being important for practitioners, this bound also has implications for the statistical properties of AdaBoost.

Like AdaBoost, most boosting algorithms are used for classification tasks, where the object is to create a model that can categorize relevant input data into one of a finite number of different classes. The most commonly studied setting is binary classification, when there are only two possible classes, although the tasks arising in practice are almost always multiclass in nature. In Chapter 3 we provide a broad and general framework for studying boosting for multiclass classification. Using this approach, we are able to identify for the first time the minimum assumptions under which boosting the accuracy is possible in the multiclass setting. Such theory existed previously for boosting for binary classification, but straightforward extensions of that to the multiclass setting lead to assumptions that are either too strong or too weak for boosting to be effectively possible. We also design boosting algorithms using these minimal assumptions, which work in more general situations than previous algorithms that assumed too much.

In the final chapter, we study the problem of learning from expert advice which is closely related to boosting. The goal is to extract useful advice from the opinions of a group of experts even when there is no consensus among the experts themselves. Although algorithms for this task enjoying excellent guarantees have existed in the past, these were only approximately optimal, and exactly optimal strategies were known only when the experts gave binary "yes/no" opinions. Our work derives exactly optimal strategies when the experts provide probabilistic opinions, which can be more nuanced than deterministic ones. In terms of boosting, this provides the optimal way of combining individual models that attach confidence rating to their predictions indicating predictive quality.

Acknowledgements

Getting to know my advisor, Rob Schapire, has been a privilege, and working with him a fantastic learning experience. A brilliant researcher, a wonderful human being and a very pragmatic person at the same time, Rob is the best advisor I could ever hope for. I thank him with all my heart, and will truly miss him when I leave.

I thank my collaborators Cynthia Rudin and David Blei, who has also been a great mentor. I am grateful to my advisor, Yoav Freund and David Blei for taking the time to read and comment on my thesis, and to Moses Charikar and Philippe Rigollet for providing valuable feedback as committee members. Many thanks also to Jacob Abernethy, Yoav Freund, Nikhil Srivastava, Aditya Bhaskara and Matus Telgarsky for many helpful discussions on the research carried out in this thesis, and to the machine learning group at Princeton for providing a supportive as well as stimulating environment for carrying out research. The research in this thesis was supported by NSF grants IIS-0325500, IIS-1016029 and IIS-1053407, for which I am grateful.

Outside of work, I have spent some splendid time with some splendid people during my five years at Princeton. Thank you for your terrific company!

Finally, to my parents and my sister, thank you for always being there for me.

To my parents

Contents

List of Figures

Chapter 1

Overview

This thesis falls into the broad area of theoretical computer science, which concerns the mathematical study of objects, typically algorithms, of interest to computer scientists. In particular, we study various aspects of a certain methodology, known as boosting, which is a very important technique in the branch of artificial intelligence known as machine learning. In the process, we also shed light on topics in neighboring research areas such as optimization and game theory that might have significance even outside of machine learning.

The goal of artificial intelligence is to build machines capable of mimicking intelligent human behavior. Such machines would carry out a variety of very complex tasks in a constantly changing real environment such as see, run, read and joke, among other things, none of which can be performed by following a single static collection of rules designed by human experts. As Schapire [43] wrote twenty years ago, "what is needed is the development of systems that learn, computers that can program themselves". Machine learning is an approach to creating such computer programs, that learn the necessary rules *on the fly* through some form of trial and error based on their past failures and successes, and thereby improve at their tasks as they do more of it. This single philosophy has led to a breakthrough and given rise to a large number of practical and semi-intelligent pieces of software for diverse tasks such as recommending music, detecting spam, answering web search queries, understanding human speech and recognizing objects from their images.

The variety of techniques used to create such programs is enormous, but it is safe to say almost all of them proceed by first building a real life model based on past data, and then make predictions based on this inferred model. The models for some of these tasks are very complex, and learning them all at once is next to impossible. Various techniques have been created to learn such models incrementally. Boosting is one such methodology that has given rise to a number of very practically successful machine learning algorithms that also enjoy strong theoretical guarantees.

Boosting incrementally combines several models with low predictive power into a single highly predictive meta-model. The strength of boosting comes from its flexibility to combine any kinds of models, including highly non-linear ones making sophisticated predictions necessary for difficult tasks. At the same time boosted models do not suffer from being overly complex, a pitfall for many machine-built models, which has severe consequences for prediction. Finally, unlike many other non-linear model building procedures, boosting is based on rigorous theory which is partially responsible for both its wide adoption and its ubiquitous success.

Since its conception more than twenty years ago, thousands of articles have been published on both its empirical and theoretical properties. While great progress has been made in understanding boosting, many basic questions about this intriguing and potent approach have eluded researchers. This thesis makes progress on understanding three such questions on the theoretical aspects of boosting, which we briefly describe next.

In Chapter 2 we study the convergence properties of arguably the most popular boosting algorithm, AdaBoost. Despite extensive theoretical and empirical study, basic properties of this algorithm are not yet understood. We address one such property, namely the rate at which AdaBoost builds up its predictive power, and provide much faster guarantees than were previously known to hold in general.

In Chapter 3, we study the theory of boosting for multiclass classification problems. In a classification problem, we build a model for predicting one of a finite number of fixed outcomes. Such problems were the first and are perhaps the most common examples of machine learning applications. Classifying emails into spam or valid, images of handwritten letters into the corresponding alphabets, a credit card transaction as legitimate or fraudulent, etc. are some examples of the variety of situations in which such models come handy. Boosting was originally designed for binary classification problems (when there are only two outcomes for any scenario, e.g., the spam filtering problem), and the corresponding theory is well understood. However, the more general case of multiclass classification is far more natural for practical applications, and despite extensive work, basic understanding of multiclass boosting was still lacking. Here we rethink multiclass boosting by casting it into a novel framework in which we can finally find the answers to some of these basic questions. In particular we show how to combine the simplest models in the best possible way, which leads to multiclass boosting algorithms that require the least assumptions and therefore are applicable most generally.

In Chapter 4 we study a problem closely related to boosting — that of learning from expert advice. Boosting can be viewed as combining the predictions or "advice" of each of its constituent models, or "experts", into one final model. In general, one may consider what is the best way of combining the opinions of several experts when there is no simple consensus among them. Here we show how to gather such opinions in an optimal way, when the experts differ considerably among themselves, and further when each expert is not even very sure of her own opinion but provides only guesses. For boosting, this leads to algorithms for optimally combining models that make random predictions.

Boosting. All the problems considered in this thesis are classification tasks, where the goal is to come up with a classifier based on past data that achieves low prediction error on unseen future data. While building a highly accurate classifier is a very difficult task, it is not hard to generate simple rules of thumb that can predict with moderate accuracy. A classic example often cited in the literature is that of filtering spam emails. A simple rule of thumb for this problem could be to classify as spam if the body of the message contained the word "viagra". Other rules could correspond to phrases such as "special offer", "free investment" , etc. The procedure for finding such rules of thumbs are known as "weak learning algorithms". A boosting algorithm typically repeatedly calls such a procedure with different weighted subsets of past data, also known as training data. Each time it is called, the weak learning algorithm returns a new rule of thumb, aka weak classifier or weak hypothesis, that best fits the data it was provided in that iteration. After many such calls, the boosting algorithm aggregates these weak classifiers into a single predictor.

The main choices left up to the boosting algorithm are how to select the training data in each round, and how to combine the weak classifiers at the end. Typically, in any iteration, those parts of the data that have been most poorly fit by the weak classifiers till that point are used to form the training set. Intuitively, the weak learning algorithm is forced to focus on the "hardest" parts of the data in each round. To combine the weak classifiers, simply taking a (weighted) majority vote of their predictions suffices. Behind this simplicity lies wide applicability; while combining the classifiers, no assumptions are made about their internal form or structure, and hence any kind of classification functions can be used.

Schapire [43] originally showed the remarkable result that with a properly designed boosting algorithm, the resulting combined predictor could have much higher accuracy than the individual weak classifiers constituting it. Many simplifications and improvements were subsequently made to the original boosting algorithm, and one of the most popular that emerged was the AdaBoost algorithm by Freund and Schapire [23].

The convergence rate of AdaBoost. AdaBoost has been named in 2006 as one of the "top 10" algorithms in data mining [54] and has performed favorably with respect to other popular machine learning algorithms in empirical comparisons [10]. AdaBoost can be viewed as an optimization problem that iteratively minimizes a certain loss function derived from the training data. Knowing the rate at which this loss function is driven down is a basic question about this very important algorithm that is highly relevant to theoreticians and practitioners alike. A number of convergence guarantees have been provided till date, but each of them hold under restrictive assumptions that are almost never true in practice. Freund and Schapire [23] gave an exponentially fast rate of convergence under the assumption that perfect accuracy was achievable. Compactness of the optimization space is a common assumption in traditional optimization literature which does not hold for machine learning datasets, and therefore many of the classical results from this neighboring area cannot be used either. Variants of AdaBoost have been shown to converge fast, but such variants typically cannot match AdaBoost's performance in practice. The only rate for AdaBoost known to hold in general is excruciatingly slow, and of very limited practical or theoretical use. Therefore, despite the importance of this problem and the large number of prior attempts, the state of affairs was rather poor, and Schapire [45] announced a monetary reward for settling this problem.

In Chapter 2 we settle this conjecture. We study two rates of convergence of AdaBoost: the first with respect to an arbitrary solution, and the second with respect to the optimal solution. In the first case, we achieve a rate that depends only on the approximation parameter and some notion of complexity of the reference solution. Further, we show the dependence on both parameters is polynomial, which was the statement of the conjecture. In the second case, we show a rate of convergence that has optimal dependence on the approximation parameter. The two results require different techniques, and do not follow from each other. Unlike previous work, our rates do not require any assumption, such as weak learnability, or a compact space of solutions. Additionally, we carefully study the constants in our bounds, and provide estimates in terms of easily measurable parameters. Finally, we construct many lower bound instances showing that most our rates are nearly tight, although for some of the bounds there is scope for improvement. Our work therefore exhaustively answers most of the questions regarding the convergence rates of AdaBoost, and uses novel techniques to do so.

A theory of multiclass boosting. AdaBoost is an excellent algorithm for binary classification, but most real life classification tasks require categorizing into more than two classes. The theory of boosting for multiclass classification is surprisingly less understood than that for binary classification, and in Chapter 3 we address this gap in our knowledge.

One reason boosting is popular is because it places minimal demands on the weak classifiers. For the case of binary classification, a weak learning algorithm that predicts just better than random guessing in each iteration is sufficient for boosting to be possible. However, straightforward extensions of this approach to the multiclass setting do not work. Requiring better than random guessing on different weighted subsets of the training data turns out to be too weak for multiclass boosting to be possible. Worse still, changing this threshold to any value does not work; it leads to either too weak conditions, or ones that are too stringent which simple boostable weak classifiers may fail to satisfy. Knowing the correct assumptions to make about the weak classifier will affect both the applicability, as well as performance of the boosting algorithm. With too strong conditions, only complex weak classifiers may be combined, and this may lead to a final model that is too complex. Such models may be *overfitted* to past data and not very useful at predicting on unforeseen data, since, according to the Occam's razor principle, the simplest model explaining data has the best performance. On the other hand, if the weak classifiers are too weak, no amount of boosting can lead to high accuracy.

We create a broad and general framework for studying multiclass boosting in which we can finally identify the correct or minimal assumptions to make about the weak classifiers. Further, we design boosting algorithms that rely on only these minimal assumptions and which drive down error as efficiently as possible. Additionally, with our understanding we are able to characterize the assumptions implicitly used in most prior work, and it turns out most of them were either too strong or too weak. So, in a certain sense, our algorithm is more generally applicable than previously existing ones. Finally, we report some preliminary experiments to demonstrate the effectiveness of our theory.

Learning from expert advice. In the final chapter, we study a problem closely related to that of boosting, that of learning to predict as well as the best in a group of experts. The goal is to be able to extract the most useful expert opinion, even when the experts differ considerably among themselves. The setup of the problem is best explained through an example. On each day, a bunch of experts predict an outcome for a common event, say, whether or not it will rain. Based on the expert opinions, our algorithm makes its own prediction, whose correctness is revealed at the end of the day, when we know whether or not it rained that day. The goal of our algorithm is to not make too many more mistakes than the best expert over a long period of time.

The problem of learning with expert advice has a long history, and many variants of it have been studied. For instance, to justify the term "expert", various performance requirements have been placed on them. Further, different versions, based on the nature of the outcomes predicted by the experts, and the structure on the space of experts, have been and continue to be explored. The exponential-weights algorithm by Littlestone and Warmuth [28] was one of the original expert learning strategies that works for most variants, and achieves excellent guarantees. However, for many situations, this algorithm is only approximately optimal, and tighter bounds are desirable.

The simplest such situation is where the experts make binary "yes/no" predictions, as in the weather predicting example above. A common requirement on the experts is that there be at least one expert which does not make more than some given finite number of mistakes. The task of the learning algorithm then is to make as few mistakes overall as possible. In this situation, the Binomial Weights algorithm of Cesa-Bianchi et al [11] achieves the optimal guarantees (when the number of experts is sufficiently large), and is based on the sophisticated analysis for a related problem developed by Spencer [51].

Our goal in this work was to achieve similar tightly optimal results in the case where experts make *probabilistic* predictions, i.e., they guess "yes" or "no" with different probabilities. Such experts are harder to learn from, but due to their extra power of randomness, they also typically provide more nuanced advice than deterministic binary experts who have to make a hard choice between one of the two outcomes. In terms of boosting for binary classification, this leads to optimal strategies for combining *confidence-rated* weak classifiers, that provide, along with their prediction on each example, their *confidence*, encoded by a real number, in their own prediction. Prior work [47] has shown that the use of such confidence-rated weak classifiers can lead to dramatic speedups on practical datasets.

Technically, analyzing the optimal strategy for learning from experts making probabilistic predictions turns out to be much harder than learning from their deterministic counterparts. We employ the powerful drifting games framework of Schapire [44] for designing and analyzing our algorithm, and extend Spencer's technique for showing the lower bounds. A novel technical contribution is the analysis of certain potential functions that arise frequently while studying boosting, and showing that they do not become chaotic but remain piecewise convex.

The results in Chapters 2, 3 and 4 have appeared in [35], [34] and [33], respectively. A preliminary version of the results in Chapter 4 have also appeared in Algorithmic Learning Theory, 19th International Conference, 2008.

In summary, we make progress in theoretically understanding three aspects of boosting. Along the way, we create new notions and techniques which might have broader significance beyond boosting, and be of independent interest. Although we try to exhaustively answer the questions we tackle, interesting questions remain, which we hope will be addressed in the future. Finally, boosting is a beautiful and powerful technique that holds many more mysteries, and I hope that the contributions of this thesis will play their small part in helping unravel some of them.

Chapter 2

The Rate of Convergence of AdaBoost

The AdaBoost algorithm of Freund and Schapire [23] was designed to combine many "weak" hypotheses that perform slightly better than random guessing into a "strong" hypothesis that has very low error. Despite extensive theoretical and empirical study, basic properties of AdaBoost's convergence are not fully understood. In this work, we focus on one of those properties, namely, to find convergence rates that hold in the absence of any simplifying assumptions. Such assumptions, relied upon in much of the preceding work, make it easier to prove a fast convergence rate for AdaBoost, but often do not hold in the cases where AdaBoost is commonly applied.

AdaBoost can be viewed as a coordinate descent (or functional gradient descent) algorithm that iteratively minimizes an objective function $L : \mathbb{R}^n \to \mathbb{R}$ called the *exponential loss* [9, 16, 24, 25, 32, 36, 38, 47]. Given m labeled training examples $(x_1, y_1), \ldots, (x_m, y_m)$, where the x_i's are in some domain $\mathcal{X}$ and $y_i \in \{-1, +1\}$, and a finite (but typically very large) space of weak hypotheses $\mathcal{H} = \{\hbar_1, \ldots, \hbar_N\}$, where each $\hbar_j : \mathcal{X} \to \{-1, +1\}$, the exponential loss is defined as

$$L(\boldsymbol{\lambda}) \triangleq \frac{1}{m} \sum_{i=1}^{m} \exp\left(- \sum_{j=1}^{N} \lambda_j y_i \hbar_j(x_i) \right)$$

where $\boldsymbol{\lambda} = \langle \lambda_1, \ldots, \lambda_N \rangle$ is a vector of weights or parameters. In each iteration, a coordinate descent algorithm moves some distance along some coordinate direction λ_j. For AdaBoost, the coordinate directions correspond to the individual weak hypotheses. Thus, on each round, AdaBoost chooses some weak hypothesis and step length, and adds these to the current weighted combination of weak hypotheses, which is equivalent to updating a single weight. The direction and step length are so chosen that the resulting vector $\boldsymbol{\lambda}^t$ in iteration t yields a lower value of the exponential loss than in the previous iteration, $L(\boldsymbol{\lambda}^t) < L(\boldsymbol{\lambda}^{t-1})$. This repeats until it reaches a minimizer if one exists. It was shown by Collins et al [13], and later by Zhang and Yu [56], that AdaBoost asymptotically converges to the minimum possible exponential loss. That is,

$$\lim_{t\to\infty} L(\boldsymbol{\lambda}^t) = \inf_{\boldsymbol{\lambda}\in\mathbb{R}^N} L(\boldsymbol{\lambda}).$$

However, that work did not address a convergence rate to the minimizer of the exponential loss.

Our work specifically addresses a recent conjecture of Schapire [45] stating that there exists a positive constant c and a polynomial poly() such that for all training sets and all finite sets of weak hypotheses, and for all $B > 0$,

$$L(\boldsymbol{\lambda}^t) \leq \min_{\boldsymbol{\lambda}:\|\boldsymbol{\lambda}\|_1\leq B} L(\boldsymbol{\lambda}) + \frac{\mathrm{poly}(\log N, m, B)}{t^c}. \tag{2.1}$$

In other words, the exponential loss of AdaBoost will be at most ε more than that of any other parameter vector $\boldsymbol{\lambda}$ of ℓ_1-norm bounded by B in a number of rounds that is bounded by a polynomial in $\log N$, m, B and $1/\varepsilon$. (We require $\log N$ rather than N since the number of weak hypotheses will typically be extremely large.) Along with an upper bound that is polynomial in these parameters, we also provide lower bound constructions showing some polynomial dependence on B and $1/\varepsilon$ is necessary. Without any additional assumptions on the exponential loss L, and without altering AdaBoost's minimization algorithm for L, the best known convergence rate of AdaBoost prior to this work that we are aware of is that of Bickel et al [7] who prove a bound on the rate of the form $O(1/\sqrt{\log t})$.

We provide also a convergence rate of AdaBoost to the minimum value of the exponential loss. Namely, within C/ϵ iterations, AdaBoost achieves a value of the exponential loss that is at most ϵ more than the best possible value, where C depends on the dataset. This convergence rate is different from the one discussed above in that it has better dependence on ϵ (in fact the dependence is optimal, as we show), and does not depend on the best solution within a ball of size B. However, this second convergence rate cannot be used to prove (2.1) since in certain worst case situations, we show the constant C may be larger than 2^m (although usually it will be much smaller).

Within the proof of the second convergence rate, we provide a lemma (called the *decomposition lemma*) that shows that the training set can be split into two sets of examples: the "finite margin set," and the "zero loss set." Examples in the finite margin set always make a positive contribution to the exponential loss, and they never lie too far from the decision boundary. Examples in the zero loss set do not have these properties. If we consider the exponential loss where the sum is only over the finite margin set (rather than over all training examples), it is minimized by a finite $\boldsymbol{\lambda}$. The fact that the training set can be decomposed into these two classes is the key step in proving the second convergence rate.

This problem of determining the rate of convergence is relevant in the proof of the consistency of AdaBoost given by Bartlett and Traskin [4], where it has a direct impact on the rate at which AdaBoost converges to the Bayes optimal classifier (under suitable assumptions). It may also be relevant to practitioners who wish to have a guarantee on the exponential loss value at iteration t (although, in general, minimization of the exponential loss need not be perfectly correlated with test accuracy).

There have been several works that make additional assumptions on the exponential loss in order to attain a better bound on the rate, but those assumptions are not true in general, and cases are known where each of these assumptions are violated. For instance, better bounds are proved by Rätsch et al [39] using results from Luo and Tseng [31], but these appear to require that the exponential loss be minimized by a finite $\boldsymbol{\lambda}$, and also depend on quantities that are not easily measured. There are many cases where L does not have a finite minimizer; in fact, one such case is provided by Schapire [45]. Shalev-Shwartz and Singer [50] have proved bounds for a variant of AdaBoost. Zhang and Yu [56] also have given rates of convergence, but their technique requires a bound on the change in the size of $\boldsymbol{\lambda}^t$ at each iteration that does not necessarily hold for AdaBoost. Many classic results are known on the convergence of iterative algorithms generally [see for instance 30, 8]; however, these typically start by assuming that the minimum is attained at some finite point in the (usually compact) space of interest, assumptions that do not generally hold in our setting. When the weak learning assumption holds, there is a parameter $\gamma > 0$ that governs the improvement of the exponential loss at each iteration. Freund and Schapire [23] and Schapire and Singer [47] showed that the exponential loss is at most $e^{-2t\gamma^2}$ after t rounds, so AdaBoost rapidly converges to the minimum possible loss under this assumption.

In Section 2.1 we summarize the coordinate descent view of AdaBoost. Section 2.2 contains the proof of the conjecture, with associated lower bounds proved in Section 2.2.3. Section 2.3 provides the C/ϵ convergence rate. The proof of the decomposition lemma is given in Section 2.3.2.

2.1 Coordinate Descent View of AdaBoost

From the examples $(x_1, y_1), \ldots, (x_m, y_m)$ and hypotheses $\mathcal{H} = \{\hbar_1, \ldots, \hbar_N\}$, AdaBoost iteratively computes the function $F : \mathcal{X} \to \mathbb{R}$, where $\text{sign}(F(x))$ can be used as a classifier for a new instance x. The function F is a linear combination of the hypotheses. At each iteration t, AdaBoost chooses one of the weak hypotheses h_t from the set $\mathcal{H}$, and adjusts its coefficient by a specified value α_t. Then F is constructed after T iterations as: $F(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$. Figure 2.1 shows the AdaBoost algorithm [23].

Since each h_t is equal to $\hbar_{j_t}$ for some j_t, F can also be written $F(x) = \sum_{j=1}^{N} \lambda_j \hbar_j(x)$ for a vector of values $\boldsymbol{\lambda} = \langle \lambda_1, \ldots \lambda_N \rangle$ (such vectors will sometimes also be referred to as *combinations*, since they represent combinations of weak hypotheses). In different notation, we can write AdaBoost as a coordinate descent algorithm on vector $\boldsymbol{\lambda}$. We define the *feature matrix* $\mathbf{M}$ elementwise by $M_{ij} = y_i \hbar_j(x_i)$, so that this matrix contains all of the inputs to AdaBoost (the training examples and hypotheses). Then

```
Given:$(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$
    set $\mathcal{H} = \{\hbar_1, \ldots, \hbar_N\}$ of weak hypotheses $\hbar_j : \mathcal{X} \to \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:
  • Train weak learner using distribution $D_t$; that is, find weak hypothesis $h_t \in \mathcal{H}$
    whose correlation $r_t \triangleq \mathbb{E}_{i \sim D_t}[y_i h_t(x_i)]$ has maximum magnitude $|r_t|$.
  • Choose $\alpha_t = \frac{1}{2} \ln \{(1 + r_t) / (1 - r_t)\}$.
  • Update, for $i = 1, \ldots, m$: $D_{t+1}(i) = D_t(i) \exp(-\alpha_t y_i h_t(x_i))/Z_t$
    where $Z_t$ is a normalization factor (chosen so that $D_{t+1}$ will be a distribution).
Output the final hypothesis: $F(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$.
```

Figure 2.1: The boosting algorithm AdaBoost.

the exponential loss can be written more compactly as:

$$L(\boldsymbol{\lambda}) = \frac{1}{m} \sum_{i=1}^{m} e^{-(\mathbf{M}\boldsymbol{\lambda})_i}$$

where $(\mathbf{M}\boldsymbol{\lambda})_i$, the i^{th} coordinate of the vector $\mathbf{M}\boldsymbol{\lambda}$, is the (unnormalized) *margin* achieved by vector $\boldsymbol{\lambda}$ on training example i.

Coordinate descent algorithms choose a coordinate at each iteration where the directional derivative is the steepest, and choose a step that maximally decreases the objective along that coordinate. To perform coordinate descent on the exponential loss, we determine the coordinate j_t at iteration t as follows, where $\mathbf{e}_j$ is a vector that is 1 in the j^{th} position and 0 elsewhere:

$$j_t \in \operatorname*{argmax}_j \left| \left(- \frac{dL(\boldsymbol{\lambda}^{t-1} + \alpha \mathbf{e}_j)}{d\alpha} \bigg|_{\alpha=0} \right) \right| = \operatorname*{argmax}_j \frac{1}{m} \left| \sum_{i=1}^{m} e^{-(\mathbf{M}\boldsymbol{\lambda}^{t-1})_i} M_{ij} \right|. \quad (2.2)$$

We can show that this is equivalent to the weak learning step of AdaBoost. Unraveling the recursion in Figure 2.1 for AdaBoost's weight vector D_t, we can see that $D_t(i)$ is proportional to

$$\exp\left(- \sum_{t'<t} \alpha_{t'} y_i h_{t'}(x_i) \right).$$

The term in the exponent can also be rewritten in terms of the vector $\boldsymbol{\lambda}^t$, where λ_j^t is the sum of α_t's where hypothesis $\hbar_j$ was chosen: $\sum_{t'<t} \alpha_{t'} \mathbf{1}_{[\hbar_j = h_{t'}]} = \lambda_{t-1,j}$. The term in the exponent is:

$$\sum_{t'<t} \alpha_{t'} y_i h_{t'}(x_i) = \sum_j \sum_{t'<t} \alpha_{t'} \mathbf{1}_{[\hbar_j = h_{t'}]} y_i \hbar_j(x_i) = \sum_j \lambda_j^{t-1} M_{ij} = (\mathbf{M}\boldsymbol{\lambda}^{t-1})_i,$$

where $(\cdot)_i$ denotes the ith component of a vector. This means $D_t(i)$ is proportional to $e^{-(\mathbf{M}\boldsymbol{\lambda}^{t-1})_i}$. Eq. (2.2) can now be rewritten as

$$j_t \in \operatorname*{argmax}_j \left| \sum_i D_t(i) M_{ij} \right| = \operatorname*{argmax}_j \left| \mathbb{E}_{i \sim D_t}\left[M_{ij}\right] \right| = \operatorname*{argmax}_j \left| \mathbb{E}_{i \sim D_t}\left[y_i h_j(x_i)\right] \right|,$$

which is exactly the way AdaBoost chooses a weak hypothesis in each round (see Figure 2.1). The correlation $\sum_i D_t(i) M_{ij_t}$ will be denoted by r_t and its absolute value $|r_t|$ denoted by δ_t. The quantity δ_t is commonly called the *edge* for round t. The distance α_t to travel along direction j_t is found for coordinate descent via a linesearch [see for instance 32]:

$$0 = -\frac{dL(\boldsymbol{\lambda}_t + \alpha_t \mathbf{e}_{j_t})}{d\alpha_t} = \sum_i e^{-\left(M(\boldsymbol{\lambda}_t + \alpha_t \mathbf{e}_{j_t})\right)_i} M_{ij_t}$$

and dividing both sides by the normalization factor,

$$0 = \sum_{i:M_{ij}=1} D_t(i) e^{-\alpha_t} - \sum_{i:M_{ij}=-1} D_t(i) e^{\alpha_t} = (1+r_t)e^{-\alpha_t} - (1-r_t)e^{\alpha_t} \implies \alpha_t = \frac{1}{2}\ln\left(\frac{1+r_t}{1-r_t}\right),$$

just as in Figure 2.1. Thus, AdaBoost is equivalent to coordinate descent on $L(\boldsymbol{\lambda})$. With this choice of step length, it can be shown [23] that the exponential loss drops by an amount depending on the edge:

$$\begin{aligned} L(\boldsymbol{\lambda}_t) &= L\left(\boldsymbol{\lambda}_{t-1} + \alpha_t \mathbf{e}_{\mathbf{j_t}}\right) = \left(\sum_{i:M_{ij}=1} D_t(i) e^{-\alpha_t} + \sum_{i:M_{ij}=-1} D_t(i) e^{\alpha_t} \right) L(\boldsymbol{\lambda}_{t-1}) \\ &= \left((1+r_t)e^{-\alpha_t} + (1-r_t)e^{\alpha_t}\right) L(\boldsymbol{\lambda}_{t-1}) = \left(2\sqrt{(1+r_t)(1-r_t)}\right) L(\boldsymbol{\lambda}_t) \\ &= \left(\sqrt{1-r_t^2}\right) L(\boldsymbol{\lambda}_{t-1}) = \left(\sqrt{1-\delta_t^2}\right) L(\boldsymbol{\lambda}_{t-1}). \end{aligned}$$

Our rate bounds also hold when the weak-hypotheses are confidence-rated, that is, giving real-valued predictions in $[-1,+1]$, so that $h : \mathcal{X} \to [-1,+1]$. In that case, the criterion for picking a weak hypothesis in each round remains the same, that is, at round t, an $\hbar_{j_t}$ maximizing the absolute correlation $j_t \in \operatorname{argmax}_j \left| \sum_{i=1}^m e^{-(\mathbf{M}\boldsymbol{\lambda}^{t-1})_i} M_{ij} \right|$, is chosen, where M_{ij} may now be non-integral. An exact analytical line search is no longer possible, but if the step size is chosen in the same way,

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1+r_t}{1-r_t}\right), \tag{2.3}$$

then Freund and Schapire [23] and Schapire and Singer [47] show that a similar drop in the loss is still guaranteed:

$$L(\boldsymbol{\lambda}^t) \le L(\boldsymbol{\lambda}^{t-1})\sqrt{1-\delta_t^2}. \tag{2.4}$$

With confidence rated hypotheses, other implementations may choose the step size in a different way. However, in this chapter, by “AdaBoost” we will always mean the version in [23, 47] which chooses step sizes as in (2.3), and enjoys the loss guarantee as in (2.4). That said, all our proofs work more generally, and are robust to numerical inaccuracies in the implementation. In other words, even if the previous conditions are violated by a small amount, similar bounds continue to hold, although we leave out explicit proofs of this fact to simplify the presentation.

2.2 First convergence rate: Convergence to any target loss

In this section, we bound the number of rounds of AdaBoost required to get within ε of the loss attained by a parameter vector $\boldsymbol{\lambda}^*$ as a function of ε and the ℓ_1-norm $\|\boldsymbol{\lambda}^*\|_1$. The vector $\boldsymbol{\lambda}^*$ serves as a reference based on which we define the target loss $L(\boldsymbol{\lambda}^*)$, and we will show that its ℓ_1-norm measures the difficulty of attaining the target loss in a specific sense. We prove a bound polynomial in $1/\varepsilon$, $\|\boldsymbol{\lambda}^*\|_1$ and the number of examples m, showing (2.1) holds, thereby resolving affirmatively the open problem posed in [45]. Later in the section we provide lower bounds showing how a polynomial dependence on both parameters is necessary.

2.2.1 Upper Bound

The main result of this section is the following rate upper bound.

Theorem 2.1. *For any $\boldsymbol{\lambda}^* \in \mathbb{R}^N$, AdaBoost achieves loss at most $L(\boldsymbol{\lambda}^*) + \varepsilon$ in at most $13\|\boldsymbol{\lambda}^*\|_1^6\varepsilon^{-5}$ rounds.*

The high level idea behind the proof of the theorem is as follows. To show a fast rate, we require a large edge in each round, as indicated by (2.4). A large edge is guaranteed if the size of the current solution of AdaBoost is small. Therefore AdaBoost makes good progress if the size of its solution does not grow too fast. On the other hand, the increase in size of its solution is given by the step length, which in turn is proportional to the edge achieved in that round. Therefore, if the solution size grows fast, the loss also drops fast. Either way the algorithm makes good progress. In the rest of the section we make these ideas concrete through a sequence of lemmas.

We provide some more notation. Throughout, $\boldsymbol{\lambda}^*$ is fixed, and its ℓ_1-norm is denoted by B [matching the notation in 45]. One key parameter is the suboptimality R_t of AdaBoost's solution measured via the logarithm of the exponential loss:

$$R_t \triangleq \ln L(\boldsymbol{\lambda}^t) - \ln L(\boldsymbol{\lambda}^*).$$

Another key parameter is the ℓ_1-distance S_t of AdaBoost's solution from the closest combination that achieves the target loss:

$$S_t \triangleq \inf_{\boldsymbol{\lambda}} \left\{ \|\boldsymbol{\lambda} - \boldsymbol{\lambda}^t\|_1 : L(\boldsymbol{\lambda}) \le L(\boldsymbol{\lambda}^*) \right\}.$$

We will also be interested in how they change as captured by

$$\Delta R_t \triangleq R_{t-1} - R_t, \qquad \Delta S_t \triangleq S_t - S_{t-1}.$$

Notice that ΔR_t is always non-negative since AdaBoost decreases the loss, and hence the suboptimality, in each round. Let T_0 be the bound on the number of rounds in Theorem 2.1. We assume without loss of generality that $R_0, \ldots, R_{T_0}$ and $S_0, \ldots, S_{T_0}$ are all strictly positive, since otherwise the theorem holds trivially. Also, in the rest of the section, we restrict our attention entirely to the first T_0 rounds of boosting. We first show that a poly(B, ε^{-1}) rate of convergence follows if the edge is always polynomially large compared to the suboptimality.

Lemma 2.2. *If for some constants c_1, c_2, where $c_2 > 1/2$, the edge satisfies $\delta_t \geq B^{-c_1} R_{t-1}^{c_2}$ in each round t, then AdaBoost achieves at most $L(\boldsymbol{\lambda}^*) + \varepsilon$ loss after $2B^{2c_1}(\varepsilon \ln 2)^{1-2c_2}$ rounds.*

Proof. From the definition of R_t and (2.4) we have

$$\Delta R_t = \ln L(\boldsymbol{\lambda}^{t-1}) - \ln L(\boldsymbol{\lambda}^t) \geq -\frac{1}{2}\ln(1 - \delta_t^2). \tag{2.5}$$

Combining the above with the inequality $e^x \geq 1 + x$, and the assumption on the edge

$$\Delta R_t \geq -\frac{1}{2}\ln(1-\delta_t^2) \geq \frac{1}{2}\delta_t^2 \geq \frac{1}{2}B^{-2c_1}R_{t-1}^{2c_2}.$$

Let $T = \lceil 2B^{2c_1}(\varepsilon \ln 2)^{1-2c_2} \rceil$ be the bound on the number of rounds in the lemma. If any of $R_0, \ldots, R_T$ is negative, then by monotonicity $R_T < 0$ and we are done. Otherwise, they are all non-negative. Then, applying Lemma 2.32 from the Appendix to the sequence $R_0, \ldots, R_T$, and using $c_2 > 1/2$ we get

$$R_T^{1-2c_2} \geq R_0^{1-2c_2} + c_2 B^{-2c_1} T > (1/2) B^{-2c_1} T \geq (\varepsilon \ln 2)^{1-2c_2} \implies R_T < \varepsilon \ln 2.$$

If either ε or $L(\boldsymbol{\lambda}^*)$ is greater than 1, then the lemma follows since $L(\boldsymbol{\lambda}^T) \leq L(\boldsymbol{\lambda}^0) = 1 < L(\boldsymbol{\lambda}^*) + \varepsilon$. Otherwise,

$$L(\boldsymbol{\lambda}^T) < L(\boldsymbol{\lambda}^*) e^{\varepsilon \ln 2} \leq L(\boldsymbol{\lambda}^*)(1 + \varepsilon) \leq L(\boldsymbol{\lambda}^*) + \varepsilon,$$

where the second inequality uses $e^x \leq 1 + (1/\ln 2)x$ for $x \in [0, \ln 2]$. $\square$

We next show that large edges are achieved provided S_t is small compared to R_t.

Lemma 2.3. *In each round t, the edge satisfies $\delta_t \geq R_{t-1}/S_{t-1}$.*

Proof. For any combination $\boldsymbol{\lambda}$, define $D_{\boldsymbol{\lambda}}$ as the distribution on examples $\{1, \ldots, m\}$ that puts weight proportional to the loss $D_{\boldsymbol{\lambda}}(i) = e^{-(\mathbf{M}\boldsymbol{\lambda})_i}/(mL(\boldsymbol{\lambda}))$. Choose any $\boldsymbol{\lambda}$ suffering at most the target loss $L(\boldsymbol{\lambda}) \leq L(\boldsymbol{\lambda}^*)$. By non-negativity of relative entropy we get

$$\begin{aligned} 0 \ \le\ \mathtt{RE}(D_{\boldsymbol{\lambda}^{t-1}} \parallel D_{\boldsymbol{\lambda}}) &= \sum_{i=1}^{m} D_{\boldsymbol{\lambda}^{t-1}} \ln\left(\frac{\frac{1}{m}e^{-(\mathbf{M}\boldsymbol{\lambda}^{t-1})_i}/L(\boldsymbol{\lambda}^{t-1})}{\frac{1}{m}e^{-(\mathbf{M}\boldsymbol{\lambda})_i}/L(\boldsymbol{\lambda})}\right) \\ &= -R_{t-1} + \sum_{i=1}^{m} D_{\boldsymbol{\lambda}^{t-1}}(i)\left(\mathbf{M}\boldsymbol{\lambda} - \mathbf{M}\boldsymbol{\lambda}^{t-1}\right)_i . \end{aligned} \tag{2.6}$$

Note that $D_{\boldsymbol{\lambda}^{t-1}}$ is the distribution D_t that AdaBoost creates in round t. The above summation can be rewritten as

$$\begin{aligned} \sum_{i=1}^{m} D_{\boldsymbol{\lambda}^{t-1}}(i) \sum_{j=1}^{N}\left(\lambda_j - \lambda_j^{t-1}\right) M_{ij} &= \sum_{j=1}^{N}\left(\lambda_j - \lambda_j^{t-1}\right)\sum_{i=1}^{m} D_t(i)M_{ij} \\ &\le \left(\sum_{j=1}^{N}\left|\lambda_j - \lambda_j^{t-1}\right|\right)\max_j\left|\sum_{i=1}^{m} D_t(i)M_{ij}\right| \\ &= \delta_t\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^{t-1}\|_1 . \end{aligned} \tag{2.7}$$

Since the previous holds for any $\boldsymbol{\lambda}$ suffering less than the target loss, the last expression is at most $\delta_t S_{t-1}$. Combining this with (2.7) completes the proof. □

To complete the proof of Theorem 2.1, we show S_t is small compared to R_t in rounds $t \le T_0$ (during which we have assumed S_t, R_t are all positive). In fact we prove:

Lemma 2.4. *For any $t \le T_0$, $S_t \le B^3 R_t^{-2}$.*

This, along with Lemmas 2.2 and 2.3, immediately proves Theorem 2.1. The bound on S_t in Lemma 2.4 can be proved if we can first show S_t grows slowly compared to the rate at which the suboptimality R_t falls. Intuitively this holds since growth in S_t is caused by a large step, which in turn will drive down the suboptimality. In fact we can prove the following.

Lemma 2.5. *In any round $t \le T_0$, we have $\frac{2\Delta R_t}{R_{t-1}} \ge \frac{\Delta S_t}{S_{t-1}}$.*

Proof. Firstly, it follows from the definition of S_t that $\Delta S_t \le \|\boldsymbol{\lambda}^t - \boldsymbol{\lambda}^{t-1}\|_1 = |\alpha_t|$. Next, using (2.5) and (2.3) we may write $\Delta R_t \ge \Upsilon(\delta_t)\,|\alpha_t|$, where the function Υ has been defined in [37] as

$$\Upsilon(x) = \frac{-\ln(1-x^2)}{\ln\left(\frac{1+x}{1-x}\right)}.$$

It is known [37, 41] that $\Upsilon(x) \ge x/2$ for $x \in [0,1]$. Combining and using Lemma 2.3,

$$\Delta R_t \ge \delta_t \Delta S_t/2 \ge R_{t-1}\left(\Delta S_t/2S_{t-1}\right).$$

Rearranging completes the proof. □

Using this we may prove Lemma 2.4.

Proof. We first show $S_0 \le B^3 R_0^{-2}$. Note, $S_0 \le \|\boldsymbol{\lambda}^* - \boldsymbol{\lambda}^0\|_1 = B$, and by definition the quantity $R_0 = -\ln\left(\frac{1}{m}\sum_i e^{-(\mathbf{M}\boldsymbol{\lambda}^*)_i}\right)$. The quantity $(\mathbf{M}\boldsymbol{\lambda}^*)_i$ is the inner product of row i of matrix $\mathbf{M}$ with the vector $\boldsymbol{\lambda}^*$. Since the entries of $\mathbf{M}$ lie in $[-1,+1]$, this is at most $\|\boldsymbol{\lambda}^*\|_1 = B$. Therefore $R_0 \le -\ln\left(\frac{1}{m}\sum_i e^{-B}\right) = B$, which is what we needed.

To complete the proof, we show that $R_t^2 S_t$ is non-increasing. It suffices to show for any t the inequality $R_t^2 S_t \le R_{t-1}^2 S_{t-1}$. This holds by the following chain:

$$\begin{aligned} R_t^2 S_t &= (R_{t-1} - \Delta R_t)^2 (S_{t-1} + \Delta S_t) = R_{t-1}^2 S_{t-1} \left(1 - \frac{\Delta R_t}{R_{t-1}}\right)^2 \left(1 + \frac{\Delta S_t}{S_{t-1}}\right) \\ &\le R_{t-1}^2 S_{t-1} \exp\left(-\frac{2\Delta R_t}{R_{t-1}} + \frac{\Delta S_t}{S_{t-1}}\right) \le R_{t-1}^2 S_{t-1}, \end{aligned}$$

where the first inequality follows from $e^x \ge 1+x$, and the second one from Lemma 2.5. □

This completes the proof of Theorem 2.1. Although our bound provides a rate polynomial in B, ε^{-1} as desired by the conjecture in [45], the exponents are rather large, and (we believe) not tight. One possible source of slack is the bound on S_t in Lemma 2.4. Qualitatively, the distance S_t to some solution having target loss should decrease with rounds, whereas Lemma 2.4 only says it does not increase too fast. Improving this will directly lead to a faster convergence rate. In particular, showing that S_t never decreases would imply a B^2/ε rate of convergence. Whether or not the monotonicity of S_t holds, we believe that the obtained rate bound is probably true, and state it as a conjecture.

Conjecture 2.6. *For any $\boldsymbol{\lambda}^*$ and $\varepsilon > 0$, AdaBoost converges to within $L(\boldsymbol{\lambda}^*) + \varepsilon$ loss in $O(B^2/\varepsilon)$ rounds, where the order notation hides only absolute constants.*

As evidence supporting the conjecture, we show in the next section how a minor modification to AdaBoost can achieve the above rate.

2.2.2 Faster rates for a variant

In this section we introduce a new algorithm, AdaBoost.S , which will enjoy the much faster rate of convergence mentioned in Conjecture 2.6. AdaBoost.S is the same as AdaBoost, except that at the end of each round, the current combination of weak hypotheses is *scaled back*, that is, multiplied by a scalar in $[0,1]$ if doing so will reduce the exponential loss further. The code is largely the same as in Section 2.1, maintaining a combination $\boldsymbol{\lambda}^{t-1}$ of weak hypotheses, and greedily choosing α_t and $\hbar_{j_t}$ on each round to form a new combination $\tilde{\boldsymbol{\lambda}}^t = \boldsymbol{\lambda}^{t-1} + \alpha_t \hbar_{j_t}$. However, after creating the new combination $\tilde{\boldsymbol{\lambda}}^t$, the result is multiplied by the value s_t in $[0,1]$ that causes the greatest decrease in the exponential loss: $s_t = \operatorname{argmin}_s L(s\tilde{\boldsymbol{\lambda}}^t)$, and $\boldsymbol{\lambda}^t = s_t \tilde{\boldsymbol{\lambda}}^t$. Since $L(s\tilde{\boldsymbol{\lambda}}^t)$, as a function of s, is convex, its minimum on $[0,1]$ can be found easily, for instance, using a simple binary search. The new distribution D_{t+1} on the examples is constructed using $\boldsymbol{\lambda}^t$ as before; the weight $D_{t+1}(i)$ on example i is proportional to its exponential loss $D_{t+1}(i) \propto e^{-(\mathbf{M}\boldsymbol{\lambda}^t)_i}$. With this modification we may prove the following:

Theorem 2.7. *For any $\boldsymbol{\lambda}^*, \varepsilon > 0$, AdaBoost.*S *achieves at most $L(\boldsymbol{\lambda}^*) + \varepsilon$ loss within $3\|\boldsymbol{\lambda}^*\|_1^2/\varepsilon$ rounds.*

The proof is similar to that in the previous section. Reusing the same notation, note that proof of Lemma 2.2 continues to hold (with very minor modifications to that are straightforward). Next we can exploit the changes in AdaBoost.S to show an improved version of Lemma 2.3. Intuitively, scaling back has the effect of preventing the weights on the weak hypotheses from becoming "too large", and we may show

Lemma 2.8. *In each round t, the edge satisfies $\delta_t \geq R_{t-1}/B$.*

Proof. We will reuse parts of the proof of Lemma 2.3. Setting $\boldsymbol{\lambda} = \boldsymbol{\lambda}^*$ in (2.6) we may write

$$R_t \leq \sum_{i=1}^m D_{\boldsymbol{\lambda}^{t-1}}(i)\left(\mathbf{M}\boldsymbol{\lambda}^*\right)_i + \sum_{i=1}^m -D_{\boldsymbol{\lambda}^{t-1}}(i)\left(\mathbf{M}\boldsymbol{\lambda}^{t-1}\right)_i.$$

The first summation can be upper bounded as in (2.7) by $\delta_t\|\boldsymbol{\lambda}^*\| = \delta_t B$. We will next show that the second summation is non-positive, which will complete the proof. The scaling step was added just so that this last fact would be true.

If we define $G : [0,1] \to \mathbb{R}$ to be $G(s) = L\left(s\tilde{\boldsymbol{\lambda}}^t\right) = \sum_i e^{-(\mathbf{M}\tilde{\boldsymbol{\lambda}}^t)_i}$, then observe that the scaled derivative $G'(s)/G(s)$ is exactly equal to the second summation. Since $G(s) \geq 0$, it suffices to show the derivative $G'(s) \leq 0$ at the optimum value of s, denoted by s^*. Since G is a strictly convex function ($\forall s : G''(s) > 0$), it is either strictly increasing or strictly decreasing throughout $[0,1]$, or it has a local minimum. In the case when it is strictly decreasing throughout, then $G'(s) \leq 0$ everywhere, whereas if G has a local minimum, then $G'(s) = 0$ at s^*. We finish the proof by showing that G cannot be strictly increasing throughout $[0,1]$. If it were, we would have $L(\tilde{\boldsymbol{\lambda}}^t) = G(1) > G(0) = 1$, an impossibility since the loss decreases through rounds. □

Lemmas 2.2 and 2.8 together now imply Theorem 2.7, where we used that $2\ln 2 < 3$.

In experiments we ran, the scaling back never occurs. For such datasets, AdaBoost and AdaBoost.S are identical. We believe that even for contrived examples, the rescaling could happen only a few times, implying that both AdaBoost and AdaBoost.S would enjoy the convergence rates of Theorem 2.7. In the next section, we construct rate lower bound examples to show that this is nearly the best rate one can hope to show.

2.2.3 Lower-bounds

Here we show that the dependence of the rate in Theorem 2.1 on the norm $\|\boldsymbol{\lambda}^*\|_1$ of the solution achieving target accuracy is necessary for a wide class of datasets. The arguments in this section are not tailored to AdaBoost, but hold more generally for any coordinate descent algorithm, and can be readily generalized to any loss function L' of the form $L'(\boldsymbol{\lambda}) = (1/m)\sum_i \phi(\mathbf{M}\boldsymbol{\lambda})$, where $\phi : \mathbb{R} \to \mathbb{R}$ is any non-decreasing function. The first lemma connects the size of a reference solution to the required number of rounds of boosting, and shows that for a wide variety of datasets the convergence rate to a target loss can be lower bounded by the ℓ_1-norm of the smallest solution achieving that loss.

Lemma 2.9. *Suppose the feature matrix* $\mathbf{M}$ *corresponding to a dataset has two rows with* $\{-1, +1\}$ *entries which are complements of each other, i.e., there are two examples on which any hypothesis gets one wrong and one correct prediction. Then the number of rounds required to achieve a target loss* L^* *is at least* $\inf\{\|\boldsymbol{\lambda}\|_1 : L(\boldsymbol{\lambda}) \le L^*\}/(2\ln m)$.

Proof. We first show that the two examples corresponding to the complementary rows in $\mathbf{M}$ both satisfy a certain margin boundedness property. Since each hypothesis predicts oppositely on these, in any round t their margins will be of equal magnitude and opposite sign. Unless both margins lie in $[-\ln m, \ln m]$, one of them will be smaller than $-\ln m$. But then the exponential loss $L(\boldsymbol{\lambda}^t) = (1/m)\sum_j e^{-(\mathbf{M}\boldsymbol{\lambda}^t)_j}$ in that round will exceed 1, a contradiction since the losses are non-increasing through rounds, and the loss at the start was 1. Thus, assigning one of these examples the index i, we have the absolute margin $|(\mathbf{M}\boldsymbol{\lambda}^t)_i|$ is bounded by $\ln m$ in any round t. Letting $\mathbf{M}(i)$ denote the ith row of $\mathbf{M}$, the step length α_t in round t therefore satisfies

$$|\alpha_t| = |M_{ij_t}\alpha_t| = |\langle \mathbf{M}(i), \alpha_t \mathbf{e}_{j_t}\rangle| = \left|(\mathbf{M}\boldsymbol{\lambda}^t)_i - (\mathbf{M}\boldsymbol{\lambda}^{t-1})_i\right| \le \left|(\mathbf{M}\boldsymbol{\lambda}^t)_i\right| + \left|(\mathbf{M}\boldsymbol{\lambda}^{t-1})_i\right| \le 2\ln m,$$

and the statement of the lemma directly follows. □

When the weak hypotheses are *abstaining* [47], it can make a definitive prediction that the label is -1 or $+1$, or it can "abstain" by predicting zero. No other levels of confidence are allowed, and the resulting feature matrix has entries in $\{-1, 0, +1\}$. The next theorem constructs a feature matrix satisfying the properties of Lemma 2.9 and where additionally the smallest size of a solution achieving $L^* + \varepsilon$ loss is at least $\Omega(2^m)\ln(1/\varepsilon)$, for some fixed L^* and every $\varepsilon > 0$.

Theorem 2.10. *Consider the following matrix* $\mathbf{M}$ *with* m *rows (or examples) labeled* $0, \ldots, m-1$ *and* $m-1$ *columns labeled* $1, \ldots, m-1$ *(assume* $m \ge 3$*). The square submatrix ignoring row zero is an upper triangular matrix, with* 1*'s on the diagonal,* -1*'s above the diagonal, and* 0 *below the diagonal. Therefore row 1 is* $(+1, -1, -1, \ldots, -1)$*. Row 0 is defined to be just the complement of row 1. Then, for any* $\varepsilon > 0$*, a loss of* $2/m + \varepsilon$ *is achievable on this dataset, but with large norms*

$$\inf\{\|\boldsymbol{\lambda}\|_1 : L(\boldsymbol{\lambda}) \le 2/m + \varepsilon\} \ge (2^{m-2} - 1)\ln(1/(3\varepsilon)).$$

Therefore, by Lemma 2.9, the minimum number of rounds required for reaching loss at most $2/m + \varepsilon$ *is at least* $\left(\frac{2^{m-2}-1}{2\ln m}\right)\ln(1/(3\varepsilon))$.

A picture of the matrix constructed in the above lemma for $m = 5$ is shown in Figure 2.2. Theorem 2.10 shows that when ε is a small constant (say $\varepsilon = 0.01$), and

$$\begin{pmatrix} - & + & + & + & + \\ + & - & - & - & - \\ 0 & + & - & - & - \\ 0 & 0 & + & - & - \\ 0 & 0 & 0 & + & - \\ 0 & 0 & 0 & 0 & + \end{pmatrix}$$

Figure 2.2: The matrix used in Theorem 2.10 when $m = 5$.

$\boldsymbol{\lambda}^*$ is some vector with loss $L^* + \varepsilon/2$, AdaBoost takes at least $\Omega(2^m/\ln m)$ steps to get within $\varepsilon/2$ of the loss achieved by $\boldsymbol{\lambda}^*$, that is, to within $L^* + \varepsilon$ loss. Since m and ε are independent quantities, this shows that a polynomial dependence on the norm of the reference solution is unavoidable, and this norm might be exponential in the number of training examples in the worst case.

Corollary 2.11. *Consider feature matrices containing only* $\{-1, 0, +1\}$ *entries. If, for some constants* c *and* β*, the bound in Theorem 2.1 can be replaced by* $O\left(\|\boldsymbol{\lambda}^*\|_1^c \varepsilon^{-\beta}\right)$ *for all such matrices, then* $c \geq 1$*. Further, for such matrices, the bound* $\mathrm{poly}(1/\varepsilon, \|\boldsymbol{\lambda}^*\|_1)$ *in Theorem 2.1 cannot be replaced by* $\mathrm{poly}(1/\varepsilon, m, N)$*.*

We now prove Theorem 2.10.
Proof of Theorem 2.10. We first lower bound the norm of solutions achieving loss at most $2/m + \varepsilon$. Observe that since rows 0 and 1 are complementary, any solution's loss on just examples 0 and 1 will add up to at least $2/m$. Therefore, to get within $2/m+\varepsilon$, the margins on examples $2, \ldots, m-1$ should be at least $\ln\left((m-2)/(m\varepsilon)\right) \geq \ln(1/(3\varepsilon))$ (for $m \geq 3$). Now, the feature matrix is designed so that the margins due to a combination $\boldsymbol{\lambda}$ satisfy the following recursive relationships:

$$\begin{aligned}(M\boldsymbol{\lambda})_{m-1} &= \lambda_{m-1}, \\ (M\boldsymbol{\lambda})_i &= \lambda_i - (\lambda_{i+1} + \ldots + \lambda_{m-1}), \text{ for } 1 \leq i \leq m-2.\end{aligned}$$

Therefore, the margin on example $m-1$ is at least $\ln(1/(3\varepsilon))$ implies $\lambda_{m-1} \geq \ln(1/(3\varepsilon))$. Similarly, $\lambda_{m-2} \geq \ln(1/(3\varepsilon)) + \lambda_{m-1} \geq 2\ln(1/(3\varepsilon))$. Continuing this way,

$$\lambda_i \geq \ln\left(\frac{1}{3\varepsilon}\right) + \lambda_{i+1} + \ldots + \lambda_{m-1} \geq \ln\left(\frac{1}{3\varepsilon}\right)\left\{1 + 2^{(m-1)-(i+1)} + \ldots + 2^0\right\} = \ln\left(\frac{1}{3\varepsilon}\right)2^{m-1-i},$$

for $i = m-1, \ldots, 2$. Hence $\|\boldsymbol{\lambda}\|_1 \geq \ln(1/(3\varepsilon))(1 + 2 + \ldots + 2^{m-3}) = (2^{m-2} - 1)\ln(1/(3\varepsilon))$.

We end by showing that a loss of at most $2/m + \varepsilon$ is achievable. The above argument implies that if $\lambda_i = 2^{m-1-i}$ for $i = 2, \ldots, m-1$, then examples $2, \ldots, m-1$ attain margin exactly 1. If we choose $\lambda_1 = \lambda_2 + \ldots + \lambda_{m-1} = 2^{m-3} + \ldots + 1 = 2^{m-2} - 1$, then the recursive relationship implies a zero margin on example 1 (and hence example

$$\begin{pmatrix} -1 & +1 \\ +1 & -1 \\ -1+\nu & +1 \\ +1 & -1+\nu \end{pmatrix}$$

Figure 2.3: A picture of the matrix used in Theorem 2.12.

0). Therefore the combination $\ln(1/\varepsilon)(2^{m-2}-1, 2^{m-3}, 2^{m-4}, \ldots, 1)$ achieves a loss $(2+(m-2)\varepsilon)/m \le 2/m + \varepsilon$, for any $\varepsilon > 0$. □

We finally show that if the weak hypotheses are confidence-rated with arbitrary levels of confidence, so that the feature matrix is allowed to have non-integral entries in $[-1, +1]$, then the minimum norm of a solution achieving a fixed accuracy can be arbitrarily large. Our constructions will satisfy the requirements of Lemma 2.9, so that the norm lower bound translates into a rate lower bound.

Theorem 2.12. *Let $\nu > 0$ be an arbitrary number, and let $\mathbf{M}$ be the (possibly) non-integral matrix with 4 examples and 2 weak hypotheses shown in Figure 2.3. Then for any $\varepsilon > 0$, a loss of $1/2 + \varepsilon$ is achievable on this dataset, but with large norms*

$$\inf\{\|\boldsymbol{\lambda}\|_1 : L(\boldsymbol{\lambda}) \le 1/2 + \varepsilon\} \ge 2\ln(1/(2\varepsilon))\nu^{-1}.$$

Therefore, by Lemma 2.9, the number of rounds required to achieve loss at most $1/2+\varepsilon$ is at least $\ln(1/(2\varepsilon))\nu^{-1}/\ln(m)$.

Proof. We first show a loss of $1/2+\varepsilon$ is achievable. Observe that the vector $\boldsymbol{\lambda} = (c, c)$, with $c = \nu^{-1}\ln(1/(2\varepsilon))$, achieves margins $0, 0, \ln(1/(2\varepsilon)), \ln(1/(2\varepsilon))$ on examples $1, 2, 3, 4$, respectively. Therefore $\boldsymbol{\lambda}$ achieves loss $1/2 + \varepsilon$. We next show a lower bound on the norm of a solution achieving this loss. Observe that since the first two rows are complementary, the loss due to just the first two examples is at least $1/2$. Therefore, any solution $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)$ achieving at most $1/2 + \varepsilon$ loss overall must achieve a margin of at least $\ln(1/(2\varepsilon))$ on both the third and fourth examples. By inspecting the two columns, this implies

$$\begin{aligned} \lambda_1 - \lambda_2 + \lambda_2\nu &\ge \ln(1/(2\varepsilon)) \\ \lambda_2 - \lambda_1 + \lambda_1\nu &\ge \ln(1/(2\varepsilon)). \end{aligned}$$

Adding the two equations we find

$$\nu(\lambda_1 + \lambda_2) \ge 2\ln(1/(2\varepsilon)) \implies \lambda_1 + \lambda_2 \ge 2\nu^{-1}\ln(1/(2\varepsilon)).$$

By the triangle inequality, $\|\boldsymbol{\lambda}\|_1 \ge \lambda_1 + \lambda_2$, and the lemma follows. □

Note that if $\nu = 0$, then the optimal solution is found in zero rounds of boosting and has optimal loss 1. However, even the tiniest perturbation $\nu > 0$ causes the optimal loss to fall to $1/2$, and causes the rate of convergence to increase drastically.

In fact, by Theorem 2.12, the number of rounds required to achieve any fixed loss below 1 grows as $\Omega(1/\nu)$, which is arbitrarily large when ν is infinitesimal. We may conclude that with non-integral feature matrices, the dependence of the rate on the norm of a reference solution is absolutely necessary.

Corollary 2.13. *When using confidence rated weak-hypotheses with arbitrary confidence levels, the bound* $\mathrm{poly}(1/\varepsilon, \|\boldsymbol{\lambda}^*\|_1)$ *in Theorem 2.1 cannot be replaced by any function of purely m, N and ε alone.*

The construction in Figure 2.3 can be generalized to produce datasets with any number of examples that suffer the same poor rate of convergence as the one in Theorem 2.12. We discussed the smallest such construction, since we feel that it best highlights the drastic effect non-integrality can have on the rate.

In this section we saw how the norm of the reference solution is an important parameter for bounding the convergence rate. In the next section we investigate the optimal dependence of the rate on the parameter ε and show that $\Omega(1/\varepsilon)$ rounds are necessary in the worst case.

2.3 Second convergence rate: Convergence to optimal loss

In the previous section, our rate bound depended on both the approximation parameter ε, as well as the size of the smallest solution achieving the target loss. For many datasets, the optimal target loss $\inf_{\boldsymbol{\lambda}} L(\boldsymbol{\lambda})$ cannot be realized by any finite solution. In such cases, if we want to bound the number of rounds needed to achieve within ε of the optimal loss, the only way to use Theorem 2.1 is to first decompose the accuracy parameter ε into two parts $\varepsilon = \varepsilon_1 + \varepsilon_2$, find some finite solution $\boldsymbol{\lambda}^*$ achieving within ε_1 of the optimal loss, and then use the bound $\mathrm{poly}(1/\varepsilon_2, \|\boldsymbol{\lambda}^*\|_1)$ to achieve at most $L(\boldsymbol{\lambda}^*) + \varepsilon_2 = \inf_{\boldsymbol{\lambda}} L(\boldsymbol{\lambda}) + \varepsilon$ loss. However, this introduces implicit dependence on ε through $\|\boldsymbol{\lambda}^*\|_1$ which may not be immediately clear. In this section, we show bounds of the form C/ε, where the constant C depends only on the feature matrix $\mathbf{M}$, and not on ε. Additionally, we show that this dependence on ε is optimal in Lemma 2.31 of the Appendix, where $\Omega(1/\varepsilon)$ rounds are shown to be necessary for converging to within ε of the optimal loss on a certain dataset. Finally, we note that the lower bounds in the previous section indicate that C can be $\Omega(2^m)$ in the worst case for integer matrices (although it will typically be much smaller), and hence this bound, though stronger than that of Theorem 2.1 with respect to ε, cannot be used to prove the conjecture in [45], since the constant is not polynomial in the number of examples m.

2.3.1 Upper Bound

The main result of this section is the following rate upper bound. A similar approach to solving this problem was taken independently by Telgarsky [52].

Theorem 2.14. *AdaBoost reaches within ε of the optimal loss in at most C/ε rounds, where C only depends on the feature matrix.*

Our techniques build upon earlier work on the rate of convergence of AdaBoost, which have mainly considered two particular cases. In the first case, the *weak learning assumption* holds, that is, the edge in each round is at least some fixed constant. In this situation, Freund and Schapire [23] and Schapire and Singer [47] show that the optimal loss is zero, that no solution with finite size can achieve this loss, but AdaBoost achieves at most ε loss within $O(\ln(1/\varepsilon))$ rounds. In the second case some finite combination of the weak classifiers achieves the optimal loss, and Rätsch et al [39], using results from Luo and Tseng [31], show that AdaBoost achieves within ε of the optimal loss again within $O(\ln(1/\varepsilon))$ rounds.

Here we consider the most general situation, where the weak learning assumption may fail to hold, and yet no finite solution may achieve the optimal loss. The dataset used in Lemma 2.31 and shown in Figure 2.4 exemplifies this situation. Our main technical contribution shows that the examples in any dataset can be partitioned into a *zero-loss set* and *finite-margin set*, such that a certain form of the weak learning assumption holds within the zero-loss set, while the optimal loss considering only the finite-margin set can be obtained by some finite solution. The two partitions provide different ways of making progress in every round, and one of the two kinds of progress will always be sufficient for us to prove Theorem 2.14.

We next state our decomposition result, illustrate it with an example, and then state several lemmas quantifying the nature of the progress we can make in each round. Using these lemmas, we prove Theorem 2.14.

Lemma 2.15. *(Decomposition Lemma) For any dataset, there exists a partition of the set of training examples X into a (possibly empty)* zero-loss set *Z and a (possibly empty)* finite-margin set *$F = Z^c \triangleq X \setminus Z$ such that the following hold simultaneously :*

1. *For some positive constant $\gamma > 0$, there exists some vector $\boldsymbol{\eta}^\dagger$ with unit ℓ_1-norm $\|\boldsymbol{\eta}^\dagger\|_1 = 1$ that attains at least γ margin on each example in Z, and exactly zero margin on each example in F*

$$\forall i \in Z : (\mathbf{M}\boldsymbol{\eta}^\dagger)_i \geq \gamma, \qquad \forall i \in F : (\mathbf{M}\boldsymbol{\eta}^\dagger)_i = 0.$$

2. *The optimal loss considering only examples within F is achieved by some finite combination $\boldsymbol{\eta}^*$.*

3. *There is a constant $\mu_{\max} < \infty$, such that for any combination $\boldsymbol{\eta}$ with bounded loss on the finite-margin set, $\sum_{i \in F} e^{-(\mathbf{M}\boldsymbol{\eta})_i} \leq m$, the margin $(\mathbf{M}\boldsymbol{\eta})_i$ for any example i in F lies in the bounded interval $[-\ln m, \mu_{\max}]$.*

A proof is deferred to the next section. The decomposition lemma immediately implies that the vector $\boldsymbol{\eta}^* + \infty \cdot \boldsymbol{\eta}^\dagger$, which denotes $\left(\boldsymbol{\eta}^* + c\boldsymbol{\eta}^\dagger\right)$ in the limit $c \to \infty$, is an optimal solution, achieving zero loss on the zero-loss set, but only finite margins (and hence positive losses) on the finite-margin set (thereby justifying the names).

	$\hbar_1$	$\hbar_2$
a	$+$	$-$
b	$-$	$+$
c	$+$	$+$

Figure 2.4: A dataset requiring $\Omega(1/\varepsilon)$ rounds for convergence.

Before proceeding, we give an example dataset and indicate the zero-loss set, finite-margin set, $\boldsymbol{\eta}^*$ and $\boldsymbol{\eta}^\dagger$ to illustrate our definitions. Consider a dataset with three examples $\{a, b, c\}$ and two hypotheses $\{\hbar_1, \hbar_2\}$ and the feature matrix $\mathbf{M}$ in Figure 2.4. Here $+$ means correct ($M_{ij} = +1$) and $-$ means wrong ($M_{ij} = -1$). The optimal solution is $\infty \cdot (\hbar_1 + \hbar_2)$ with a loss of $2/3$. The finite-margin set is $\{a, b\}$, the zero-loss set is $\{c\}$, $\boldsymbol{\eta}^\dagger = (1/2, 1/2)$ and $\boldsymbol{\eta}^* = (0, 0)$; for this dataset these are unique. This dataset also serves as a lower-bound example in Lemma 2.31, where we show that $2/(9\varepsilon)$ rounds are necessary for AdaBoost to achieve loss at most $(2/3) + \varepsilon$.

Before providing proofs, we introduce some notation. By $\|\cdot\|$ we will mean ℓ_2-norm; every other norm will have an appropriate subscript, such as $\|\cdot\|_1$, $\|\cdot\|_\infty$, etc. The set of all training examples will be denoted by X. By $\ell^{\boldsymbol{\lambda}}(i)$ we mean the exp-loss $e^{-(\mathbf{M}\boldsymbol{\lambda})_i}$ on example i. For any subset $S \subseteq X$ of examples, $\ell^{\boldsymbol{\lambda}}(S) = \sum_{i \in S} \ell^{\boldsymbol{\lambda}}(i)$ denotes the total exp-loss on the set S. Notice $L(\boldsymbol{\lambda}) = (1/m)\ell^{\boldsymbol{\lambda}}(X)$, and that $D_{t+1}(i) = \ell^{\boldsymbol{\lambda}^t}(i)/\ell^{\boldsymbol{\lambda}^t}(X)$, where $\boldsymbol{\lambda}^t$ is the combination found by AdaBoost at the end of round t. By $\delta_S(\boldsymbol{\eta}; \boldsymbol{\lambda})$ we mean the edge obtained on the set S by the vector $\boldsymbol{\eta}$, when the weights over the examples are given by $\ell^{\boldsymbol{\lambda}}(\cdot)/\ell^{\boldsymbol{\lambda}}(S)$:

$$\delta_S(\boldsymbol{\eta}; \boldsymbol{\lambda}) = \left| \frac{1}{\ell^{\boldsymbol{\lambda}}(S)} \sum_{i \in S} \ell^{\boldsymbol{\lambda}}(i)(\mathbf{M}\boldsymbol{\eta})_i \right|.$$

In the rest of the section, by "loss" we mean the unnormalized loss $\ell^{\boldsymbol{\lambda}}(X) = mL(\boldsymbol{\lambda})$ and show that in C/ε rounds AdaBoost converges to within ε of the optimal unnormalized loss $\inf_{\boldsymbol{\lambda}} \ell^{\boldsymbol{\lambda}}(X)$, henceforth denoted by K. Note that this means AdaBoost takes C/ε rounds to converge to within ε/m of the optimal normalized loss, that is to loss at most $\inf_{\boldsymbol{\lambda}} L(\boldsymbol{\lambda}) + \varepsilon/m$. Replacing ε by $m\varepsilon$, it takes $C/(m\varepsilon)$ steps to attain normalized loss at most $\inf_{\boldsymbol{\lambda}} L(\boldsymbol{\lambda}) + \varepsilon$. Thus, whether we use normalized or unnormalized does not substantively affect the result in Theorem 2.14. The progress due to the zero-loss set is now immediate from Item 1 of the decomposition lemma:

Lemma 2.16. *In any round t, the maximum edge δ_t is at least $\gamma \ell^{\boldsymbol{\lambda}^{t-1}}(Z)/\ell^{\boldsymbol{\lambda}^{t-1}}(X)$, where γ is as in Item 1 of the decomposition lemma.*

Proof. Recall the distribution D_t created by AdaBoost in round t puts weight $D_t(i) = \ell^{\boldsymbol{\lambda}^{t-1}}(i)/\ell^{\boldsymbol{\lambda}^{t-1}}(X)$ on each example i. From Item 1 we get

$$\delta_X(\boldsymbol{\eta}^\dagger; \boldsymbol{\lambda}^{t-1}) = \left| \frac{1}{\ell^{\boldsymbol{\lambda}^{t-1}}(X)} \sum_{i \in X} \ell^{\boldsymbol{\lambda}^{t-1}}(i)(\mathbf{M}\boldsymbol{\eta}^\dagger)_i \right| = \frac{1}{\ell^{\boldsymbol{\lambda}^{t-1}}(X)} \sum_{i \in Z} \gamma \ell^{\boldsymbol{\lambda}^{t-1}}(i) = \gamma \left(\frac{\ell^{\boldsymbol{\lambda}^{t-1}}(Z)}{\ell^{\boldsymbol{\lambda}^{t-1}}(X)} \right).$$

Since $(\mathbf{M}\boldsymbol{\eta}^\dagger)_i = \sum_j \eta_j^\dagger (\mathbf{M}\mathbf{e}_j)_i$, we may rewrite the edge $\delta_X(\boldsymbol{\eta}^\dagger; \boldsymbol{\lambda}^{t-1})$ as follows:

$$\begin{aligned}
\delta_X(\boldsymbol{\eta}^\dagger; \boldsymbol{\lambda}^{t-1}) &= \left| \frac{1}{\ell^{\boldsymbol{\lambda}^{t-1}}(X)} \sum_{i \in X} \ell^{\boldsymbol{\lambda}^{t-1}}(i) \sum_j \eta_j^\dagger (\mathbf{M}\mathbf{e}_j)_i \right| \\
&= \left| \sum_j \eta_j^\dagger \frac{1}{\ell^{\boldsymbol{\lambda}^{t-1}}(X)} \sum_{i \in X} \ell^{\boldsymbol{\lambda}^{t-1}}(i) (\mathbf{M}\mathbf{e}_j)_i \right| \\
&= \left| \sum_j \eta_j^\dagger \delta_X(\mathbf{e}_j; \boldsymbol{\lambda}^{t-1}) \right| \leq \sum_j \left| \eta_j^\dagger \right| \delta_X(\mathbf{e}_j; \boldsymbol{\lambda}^{t-1}).
\end{aligned}$$

Since the ℓ_1-norm of $\boldsymbol{\eta}^\dagger$ is 1, the weights $\left|\eta_j^\dagger\right|$ form some distribution p over the columns $1, \ldots, N$. We may therefore conclude

$$\gamma \left(\frac{\ell^{\boldsymbol{\lambda}^{t-1}}(Z)}{\ell^{\boldsymbol{\lambda}^{t-1}}(X)} \right) = \delta_X(\boldsymbol{\eta}^\dagger; \boldsymbol{\lambda}^{t-1}) \leq \mathbb{E}_{j \sim p} \left[\delta_X(\mathbf{e}_j; \boldsymbol{\lambda}^{t-1}) \right] \leq \max_j \delta_X(\mathbf{e}_j; \boldsymbol{\lambda}^{t-1}) \leq \delta_t.$$

$\square$

If the set F were empty, then Lemma 2.16 implies an edge of γ is available in each round. This in fact means that the weak learning assumption holds, and using (2.4), we can show an $O(\ln(1/\varepsilon)\gamma^{-2})$ bound matching the rate bounds of Freund and Schapire [23] and Schapire and Singer [47]. So henceforth, we assume that F is non-empty. Note that this implies that the optimal loss K is at least 1 (since any solution will get non-positive margin on some example in F), a fact we will use later in the proofs.

Lemma 2.16 says that the edge is large if the loss on the zero-loss set is large. On the other hand, when it is small, Lemmas 2.17 and 2.18 together show how AdaBoost can make good progress using the finite margin set. Lemma 2.17 uses second order methods to show how progress is made in the case where there is a finite solution. Similar arguments, under additional assumptions, have earlier appeared in [39].

Lemma 2.17. *Suppose $\boldsymbol{\lambda}$ is a combination such that $m \geq \ell^{\boldsymbol{\lambda}}(F) \geq K$. Then in some coordinate direction the edge is at least $\sqrt{C_0 \left(\ell^{\boldsymbol{\lambda}}(F) - K\right) / \ell^{\boldsymbol{\lambda}}(F)}$, where C_0 is a constant depending only on the feature matrix $\mathbf{M}$.*

Proof. Let $\mathbf{M}_F \in \mathbb{R}^{|F| \times N}$ be the matrix $\mathbf{M}$ restricted to only the rows corresponding to the examples in F. Choose $\boldsymbol{\eta}$ such that $\boldsymbol{\lambda} + \boldsymbol{\eta} = \boldsymbol{\eta}^*$ is an optimal solution over F. Without loss of generality assume that $\boldsymbol{\eta}$ lies in the orthogonal subspace of the null-space $\{\mathbf{u} : \mathbf{M}_F \mathbf{u} = \mathbf{0}\}$ of $\mathbf{M}_F$ (since we can translate $\boldsymbol{\eta}^*$ along the null space if necessary for this to hold). If $\boldsymbol{\eta} = \mathbf{0}$, then $\ell^{\boldsymbol{\lambda}}(F) = K$ and we are done. Otherwise $\|\mathbf{M}_F \boldsymbol{\eta}\| \geq \lambda_{\min} \|\boldsymbol{\eta}\|$, where $\lambda_{\min}^2$ is the smallest positive eigenvalue of the symmetric matrix $\mathbf{M}_F^T \mathbf{M}_F$ (exists since $\mathbf{M}_F \boldsymbol{\eta} \neq \mathbf{0}$). Now define $f : [0,1] \to \mathbb{R}$ as the loss along the (rescaled) segment $[\boldsymbol{\eta}^*, \boldsymbol{\lambda}]$

$$f(x) \triangleq \ell^{(\boldsymbol{\eta}^* - x\boldsymbol{\eta})}(F) = \sum_{i \in F} \ell^{\boldsymbol{\eta}^*}(i) e^{x(\mathbf{M}\boldsymbol{\eta})_i}.$$

This implies that $f(0) = K$ and $f(1) = \ell^{\boldsymbol{\lambda}}(F)$. Notice that the first and second derivatives of $f(x)$ are given by:

$$f'(x) = \sum_{i \in F} (\mathbf{M}_F \boldsymbol{\eta})_i \ell^{(\boldsymbol{\eta}^* - x\boldsymbol{\eta})}(i), \qquad f''(x) = \sum_{i \in F} (\mathbf{M}_F \boldsymbol{\eta})_i^2 \ell^{(\boldsymbol{\eta}^* - x\boldsymbol{\eta})}(i).$$

We next lower bound possible values of the second derivative as follows:

$$f''(x) = \sum_{i' \in F} (\mathbf{M}_F \boldsymbol{\eta})_{i'}^2 \ell^{(\boldsymbol{\eta}^* - x\boldsymbol{\eta})}(i') \geq \sum_{i' \in F} (\mathbf{M}_F \boldsymbol{\eta})_{i'}^2 \min_i \ell^{(\boldsymbol{\eta}^* - x\boldsymbol{\eta})}(i) \geq \|\mathbf{M}_F \boldsymbol{\eta}\|^2 \min_i \ell^{(\boldsymbol{\eta}^* - x\boldsymbol{\eta})}(i).$$

Since both $\boldsymbol{\lambda} = \boldsymbol{\eta}^* - \boldsymbol{\eta}$, and $\boldsymbol{\eta}^*$ suffer total loss at most m, by convexity, so does $\boldsymbol{\eta}^* - x\boldsymbol{\eta}$ for any $x \in [0,1]$. Hence we may apply Item 3 of the decomposition lemma to the vector $\boldsymbol{\eta}^* - x\boldsymbol{\eta}$, for any $x \in [0,1]$, to conclude that $\ell^{(\boldsymbol{\eta}^* - x\boldsymbol{\eta})}(i) = \exp\{-(\mathbf{M}_F(\boldsymbol{\eta}^* - x\boldsymbol{\eta}))_i\} \geq e^{-\mu_{\max}}$ on every example i. Therefore we have,

$$f''(x) \geq \|\mathbf{M}_F \boldsymbol{\eta}\|^2 e^{-\mu_{\max}} \geq \lambda_{\min}^2 e^{-\mu_{\max}} \|\boldsymbol{\eta}\|^2 \text{ (by choice of } \boldsymbol{\eta}) .$$

A standard second-order result is [see e.g. 8, eqn. (9.9)]

$$|f'(1)|^2 \geq 2 \left(\inf_{x \in [0,1]} f''(x) \right) (f(1) - f(0)) .$$

Collecting our results so far, we get

$$\sum_{i \in F} \ell^{\boldsymbol{\lambda}}(i) (\mathbf{M}\boldsymbol{\eta})_i = |f'(1)| \geq \|\boldsymbol{\eta}\| \sqrt{2\lambda_{\min}^2 e^{-\mu_{\max}} \left(\ell^{\boldsymbol{\lambda}}(F) - K\right)}.$$

Next let $\tilde{\boldsymbol{\eta}} = \boldsymbol{\eta}/\|\boldsymbol{\eta}\|_1$ be $\boldsymbol{\eta}$ rescaled to have unit ℓ_1 norm. Then we have

$$\sum_{i \in F} \ell^{\boldsymbol{\lambda}}(i) (\mathbf{M}\tilde{\boldsymbol{\eta}})_i = \frac{1}{\|\boldsymbol{\eta}\|_1} \sum_i \ell^{\boldsymbol{\lambda}}(i) (\mathbf{M}\boldsymbol{\eta})_i \geq \frac{\|\boldsymbol{\eta}\|}{\|\boldsymbol{\eta}\|_1} \sqrt{2\lambda_{\min}^2 e^{-\mu_{\max}} \left(\ell^{\boldsymbol{\lambda}}(F) - K\right)}.$$

Applying the Cauchy-Schwarz inequality, we may lower bound $\frac{\|\boldsymbol{\eta}\|}{\|\boldsymbol{\eta}\|_1}$ by $1/\sqrt{N}$ (since $\boldsymbol{\eta} \in \mathbb{R}^N$). Along with the fact $\ell^{\boldsymbol{\lambda}}(F) \leq m$, we may write

$$\frac{1}{\ell^{\boldsymbol{\lambda}}(F)} \sum_{i \in F} \ell^{\boldsymbol{\lambda}}(i) (\mathbf{M}\tilde{\boldsymbol{\eta}})_i \geq \sqrt{2\lambda_{\min}^2 N^{-1} m^{-1} e^{-\mu_{\max}}} \sqrt{\left(\ell^{\boldsymbol{\lambda}}(F) - K\right) / \ell^{\boldsymbol{\lambda}}(F)}.$$

If we define p to be a distribution on the columns $\{1,\ldots,N\}$ of $\mathbf{M}_F$ which puts probability $p(j)$ proportional to $|\tilde{\boldsymbol{\eta}}_j|$ on column j, then we have

$$\frac{1}{\ell^{\boldsymbol{\lambda}}(F)}\sum_{i\in F}\ell^{\boldsymbol{\lambda}}(i)(\mathbf{M}\tilde{\boldsymbol{\eta}})_i \le \mathbb{E}_{j\sim p}\left|\frac{1}{\ell^{\boldsymbol{\lambda}}(F)}\sum_{i\in F}\ell^{\boldsymbol{\lambda}}(i)(\mathbf{M}\mathbf{e}_j)_i\right| \le \max_j\left|\frac{1}{\ell^{\boldsymbol{\lambda}}(F)}\sum_{i\in F}\ell^{\boldsymbol{\lambda}}(i)(\mathbf{M}\mathbf{e}_j)_i\right|.$$

Notice the quantity inside the max is precisely the edge $\delta_F(\mathbf{e}_j;\boldsymbol{\lambda})$ in direction j. Combining everything, the maximum possible edge is

$$\max_j \delta_F(\mathbf{e}_j;\boldsymbol{\lambda}) \ge \sqrt{C_0\left(\ell^{\boldsymbol{\lambda}}(F)-K\right)/\ell^{\boldsymbol{\lambda}}(F)},$$

where we define $C_0 = 2\lambda_{\min}^2 N^{-1}m^{-1}e^{-\mu_{\max}}$. □

Lemma 2.18. *Suppose, at some stage of boosting, the combination found by AdaBoost is $\boldsymbol{\lambda}$, and the loss is $K+\theta$. Let $\Delta\theta$ denote the drop in the suboptimality θ after one more round; i.e., the loss after one more round is $K+\theta-\Delta\theta$. Then there are constants C_1, C_2 depending only on the feature matrix (and not on θ), such that if $\ell^{\boldsymbol{\lambda}}(Z) < C_1\theta$, then $\Delta\theta \ge C_2\theta$.*

Proof. Let $\boldsymbol{\lambda}$ be the current solution found by boosting. Using Lemma 2.17, pick a direction j in which the edge $\delta_F(\mathbf{e}_j;\boldsymbol{\lambda})$ restricted to the finite loss set is at least $\sqrt{2C_0(\ell^{\boldsymbol{\lambda}}(F)-K)/\ell^{\boldsymbol{\lambda}}(F)}$. We can bound the edge $\delta_X(\mathbf{e}_j;\boldsymbol{\lambda})$ on the entire set of examples as follows:

$$\begin{aligned}
\delta_X(\mathbf{e}_j;\boldsymbol{\lambda}) &= \frac{1}{\ell^{\boldsymbol{\lambda}}(X)}\left|\sum_{i\in F}\ell^{\boldsymbol{\lambda}}(i)(\mathbf{M}\mathbf{e}_j)_i + \sum_{i\in Z}\ell^{\boldsymbol{\lambda}}(i)(\mathbf{M}\mathbf{e}_j)_i\right| \\
&\ge \frac{1}{\ell^{\boldsymbol{\lambda}}(X)}\left(\left|\ell^{\boldsymbol{\lambda}}(F)\delta_F(\mathbf{e}_j;\boldsymbol{\lambda})\right| - \sum_{i\in Z}\ell^{\boldsymbol{\lambda}}(i)\right) \text{ (using the triangle inequality)} \\
&\ge \frac{1}{\ell^{\boldsymbol{\lambda}}(X)}\left(\sqrt{2C_0(\ell^{\boldsymbol{\lambda}}(F)-K)\ell^{\boldsymbol{\lambda}}(F)} - \ell^{\boldsymbol{\lambda}}(Z)\right).
\end{aligned}$$

Now, $\ell^{\boldsymbol{\lambda}}(Z) < C_1\theta$, and $\ell^{\boldsymbol{\lambda}}(F) - K = \theta - \ell^{\boldsymbol{\lambda}}(Z) \ge (1-C_1)\theta$. Further, we will choose $C_1 < 1$, so that $\ell^{\boldsymbol{\lambda}}(F) \ge K \ge 1$. Hence, the previous inequality implies

$$\delta_X(\mathbf{e}_j;\boldsymbol{\lambda}) \ge \frac{1}{K+\theta}\left(\sqrt{2C_0(1-C_1)\theta} - C_1\theta\right).$$

Set $C_1 = \min\left\{1/2, (1/4)\sqrt{C_0/(2m)}\right\}$. Using $\theta \le K+\theta = \ell^{\boldsymbol{\lambda}}(X) \le m$, we can bound the square of the term in brackets on the previous line as

$$\begin{aligned}
\left(\sqrt{2C_0(1-C_1)\theta} - C_1\theta\right)^2 &\ge 2C_0(1-C_1)\theta - 2C_1\theta\sqrt{2C_0(1-C_1)\theta} \\
&\ge 2C_0(1-1/2)\theta - 2\left((1/4)\sqrt{C_0/(2m)}\right)\theta\sqrt{2C_0(1-0)m} \\
&= C_0\theta/2.
\end{aligned}$$

So, if δ is the maximum edge in any direction, then

$$\delta \geq \delta_X(\mathbf{e}_j; \boldsymbol{\lambda}) \geq \sqrt{C_0\theta/(2(K+\theta)^2)} \geq \sqrt{C_0\theta/(2m(K+\theta))},$$

where, for the last inequality, we again used $K + \theta \leq m$. Therefore the loss after one more step is at most $(K+\theta)\sqrt{1-\delta^2} \leq (K+\theta)(1-\delta^2/2) \leq K+\theta - \frac{C_0}{4m}\theta$. Setting $C_2 = C_0/(4m)$ completes the proof. □

Proof of Theorem 2.14. At any stage of boosting, let $\boldsymbol{\lambda}$ be the current combination, and $K+\theta$ be the current loss. We show that the new loss is at most $K+\theta-\Delta\theta$ for $\Delta\theta \geq C_3\theta^2$ for some constant C_3 depending only on the dataset (and not θ). To see this, either $\ell^{\boldsymbol{\lambda}}(Z) < C_1\theta$, in which case Lemma 2.18 applies, and $\Delta\theta \geq C_2\theta \geq (C_2/m)\theta^2$ (since $\theta = \ell^{\boldsymbol{\lambda}}(X) - K \leq m$). Or $\ell^{\boldsymbol{\lambda}}(Z) \geq C_1\theta$, in which case applying Lemma 2.16 yields $\delta \geq \gamma C_1\theta/\ell^{\boldsymbol{\lambda}}(X) \geq (\gamma C_1/m)\theta$. By (2.4), $\Delta\theta \geq \ell^{\boldsymbol{\lambda}}(X)(1-\sqrt{1-\delta^2}) \geq \ell^{\boldsymbol{\lambda}}(X)\delta^2/2 \geq (K/2)(\gamma C_1/m)^2\theta^2$. Using $K \geq 1$ and choosing C_3 appropriately gives the required condition.

If $K+\theta_t$ denotes the loss in round t, then the above claim implies $\theta_t - \theta_{t+1} \geq C_3\theta_t^2$. Applying Lemma 2.32 to the sequence $\{\theta_t\}$ we have $1/\theta_T - 1/\theta_0 \geq C_3T$ for any T. Since $\theta_0 \geq 0$, we have $T \leq 1/(C_3\theta_T)$. Hence to achieve loss $K+\varepsilon$, C_3^{-1}/ε rounds suffice. □

2.3.2 Proof of the decomposition lemma

Throughout this section we only consider (unless otherwise stated) *admissible* combinations $\boldsymbol{\lambda}$ of weak classifiers, which have loss $\ell^{\boldsymbol{\lambda}}(X)$ bounded by m (since such are the ones found by boosting). We prove Lemma 2.15 in three steps. We begin with a simple lemma that rigorously defines the zero-loss and finite-margin sets.

Lemma 2.19. *For any sequence $\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, \ldots,$ of admissible combinations of weak classifiers, we can find a subsequence $\boldsymbol{\eta}_{(1)} = \boldsymbol{\eta}_{t_1}, \boldsymbol{\eta}_{(2)} = \boldsymbol{\eta}_{t_2}, \ldots,$ whose losses converge to zero on all examples in some fixed (possibly empty) subset Z (the zero-loss set), and losses bounded away from zero in its complement $X \setminus Z$ (the finite-margin set)*

$$\forall x \in Z : \lim_{t\to\infty} \ell^{\boldsymbol{\eta}_{(t)}}(x) = 0, \qquad \forall x \in X \setminus Z : \inf_i \ell^{\boldsymbol{\eta}_{(t)}}(x) > 0. \tag{2.8}$$

Proof. We will build a zero-loss set and the final subsequence incrementally. Initially the set is empty. Pick the first example. If the infimal loss ever attained on the example in the sequence is bounded away from zero, then we do not add it to the set. Otherwise we add it, and consider only the subsequence whose t^{th} element attains loss less than $1/t$ on the example. Beginning with this subsequence, we now repeat with other examples. The final sequence is the required subsequence, and the examples we have added form the zero-loss set. □

We apply Lemma 2.19 to some admissible sequence converging to the optimal loss (for instance, the one found by AdaBoost). Let us call the resulting subsequence $\boldsymbol{\eta}^*_{(t)}$, the obtained zero-loss set Z, and the finite-margin set $F = X \setminus Z$. The next lemma shows how to extract a single combination out of the sequence $\boldsymbol{\eta}^*_{(t)}$ that satisfies the properties in Item 1 of the decomposition lemma.

Lemma 2.20. *Suppose* $\mathbf{M}$ *is the feature matrix,* Z *is a subset of the examples, and* $\boldsymbol{\eta}_{(1)}, \boldsymbol{\eta}_{(2)}, \ldots,$ *is a sequence of combinations of weak classifiers such that* Z *is its zero loss set, and* $X \setminus Z$ *its finite loss set, that is,* (2.8) *holds. Then there is a combination* $\boldsymbol{\eta}^\dagger$ *of weak classifiers that achieves positive margin on every example in* Z*, and zero margin on every example in its complement* $X \setminus Z$*, that is:*

$$(\mathbf{M}\boldsymbol{\eta}^\dagger)_i \begin{cases} > 0 & \textit{if } i \in Z, \\ = 0 & \textit{if } i \in X \setminus Z. \end{cases}$$

Proof. Since the $\boldsymbol{\eta}_{(t)}$ achieve arbitrarily large positive margins on Z, $\|\boldsymbol{\eta}_{(t)}\|$ will be unbounded, and it will be hard to extract a useful single solution out of them. On the other hand, the rescaled combinations $\boldsymbol{\eta}_{(t)}/\|\boldsymbol{\eta}_{(t)}\|$ lie on a compact set, and therefore have a limit point, which might have useful properties. We formalize this next.

We prove the statement of the lemma by induction on the total number of training examples $|X|$. If X is empty, then the lemma holds vacuously for any $\boldsymbol{\eta}^\dagger$. Assume inductively for all X of size less than $m > 0$, and consider X of size m. Since translating a vector along the null space of $\mathbf{M}$, $\ker \mathbf{M} = \{\mathbf{x} : \mathbf{M}\mathbf{x} = \mathbf{0}\}$, has no effect on the margins produced by the vector, assume without loss of generality that the $\boldsymbol{\eta}_{(t)}$'s are orthogonal to $\ker \mathbf{M}$. Also, since the margins produced on the zero loss set are unbounded, so are the norms of $\boldsymbol{\eta}_{(t)}$. Therefore assume (by picking a subsequence and relabeling if necessary) that $\|\boldsymbol{\eta}_{(t)}\| > t$. Let $\boldsymbol{\eta}'$ be a limit point of the sequence $\boldsymbol{\eta}_{(t)}/\|\boldsymbol{\eta}_{(t)}\|$, a unit vector that is also orthogonal to the null-space. Then firstly $\boldsymbol{\eta}'$ achieves non-negative margin on every example; otherwise by continuity for some extremely large t, the margin of $\boldsymbol{\eta}_{(t)}/\|\boldsymbol{\eta}_{(t)}\|$ on that example is also negative and bounded away from zero, and therefore $\boldsymbol{\eta}_{(t)}$'s loss is more than m, a contradiction to admissibility. Secondly, the margin of $\boldsymbol{\eta}'$ on each example in $X \setminus Z$ is zero; otherwise, by continuity, for arbitrarily large t the margin of $\boldsymbol{\eta}_{(t)}/\|\boldsymbol{\eta}_{(t)}\|$ on an example in $X \setminus Z$ is positive and bounded away from zero, and hence that example attains arbitrarily small loss in the sequence, a contradiction to (2.8). Finally, if $\boldsymbol{\eta}'$ achieves zero margin everywhere in Z, then $\boldsymbol{\eta}'$, being orthogonal to the null-space, must be $\mathbf{0}$, a contradiction since $\boldsymbol{\eta}'$ is a unit vector. Therefore $\boldsymbol{\eta}'$ must achieve positive margin on some non-empty subset S of Z, and zero margins on every other example.

Next we use induction on the reduced set of examples $X' = X \setminus S$. Since S is non-empty, $|X'| < m$. Further, using the same sequence $\boldsymbol{\eta}_{(t)}$, the zero-loss and finite-loss sets, restricted to X', are $Z' = Z \setminus S$ and $(X \setminus Z) \setminus S = X \setminus Z$ (since $S \subseteq Z$) $= X' \setminus Z'$. By the inductive hypothesis, there exists some $\boldsymbol{\eta}''$ which achieves positive margins on Z', and zero margins on $X' \setminus Z' = X \setminus Z$. Therefore, by setting $\boldsymbol{\eta}^\dagger = \boldsymbol{\eta}' + c\boldsymbol{\eta}''$ for a large enough c, we can achieve the desired properties. □

Applying Lemma 2.20 to the sequence $\boldsymbol{\eta}^*_{(t)}$ yields some convex combination $\boldsymbol{\eta}^\dagger$ having margin at least $\gamma > 0$ (for some γ) on Z and zero margin on its complement, proving Item 1 of the decomposition lemma. The next lemma proves Item 2.

Lemma 2.21. *The optimal loss considering only examples within F is achieved by some finite combination $\boldsymbol{\eta}^*$.*

Proof. The existence of $\boldsymbol{\eta}^\dagger$ with properties as in Lemma 2.20 implies that the optimal loss is the same whether considering all the examples, or just examples in F. Therefore it suffices to show the existence of finite $\boldsymbol{\eta}^*$ that achieves loss K on F, that is, $\ell^{\boldsymbol{\eta}^*}(F) = K$.

Recall $\mathbf{M}_F$ denotes the matrix $\mathbf{M}$ restricted to the rows corresponding to examples in F. Let $\ker \mathbf{M}_F = \{\mathbf{x} : \mathbf{M}_F\mathbf{x} = 0\}$ be the null-space of $\mathbf{M}_F$. Let $\boldsymbol{\eta}^{(t)}$ be the projection of $\boldsymbol{\eta}^*_{(t)}$ onto the orthogonal subspace of $\ker \mathbf{M}_F$. Then the losses $\ell^{\boldsymbol{\eta}^{(t)}}(F) = \ell^{\boldsymbol{\eta}^*_{(t)}}(F)$ converge to the optimal loss K. If $\mathbf{M}_F$ is identically zero, then each $\boldsymbol{\eta}^{(t)} = \mathbf{0}$, and then $\boldsymbol{\eta}^* = \mathbf{0}$ has loss K on F. Otherwise, let λ^2 be the smallest positive eigenvalue of $\mathbf{M}_F^T\mathbf{M}_F$. Then $\|\mathbf{M}\boldsymbol{\eta}^{(t)}\| \geq \lambda\|\boldsymbol{\eta}^{(t)}\|$. By the definition of finite margin set, $\inf_{t\to\infty} \min_{i\in F} \ell^{\boldsymbol{\eta}^{(t)}}(i) = \inf_{t\to\infty} \min_{i\in F} \ell^{\boldsymbol{\eta}^*_{(t)}}(i) > 0$. Therefore, the norms of the margin vectors $\|\mathbf{M}\boldsymbol{\eta}^{(t)}\|$, and hence that of $\boldsymbol{\eta}^{(t)}$, are bounded. Therefore the $\boldsymbol{\eta}^{(t)}$'s have a (finite) limit point $\boldsymbol{\eta}^*$ that must have loss K over F. □

As a corollary, we prove Item 3.

Lemma 2.22. *There is a constant $\mu_{\max} < \infty$, such that for any combination $\boldsymbol{\eta}$ that achieves bounded loss on the finite-margin set, $\ell^{\boldsymbol{\eta}}(F) \leq m$, the margin $(\mathbf{M}\boldsymbol{\eta})_i$ for any example i in F lies in the bounded interval $[-\ln m, \mu_{\max}]$.*

Proof. Since the loss $\ell^{\boldsymbol{\eta}}(F)$ is at most m, therefore no margin may be less than $-\ln m$. To prove a finite upper bound on the margins, we argue by contradiction. Suppose arbitrarily large margins are producible by bounded loss vectors, that is arbitrarily large elements are present in the set $\{(\mathbf{M}\boldsymbol{\eta})_i : \ell^{\boldsymbol{\eta}}(F) \leq m, 1 \leq i \leq m\}$. Then for some fixed example $x \in F$ there exists a sequence of combinations of weak classifiers, whose t^{th} element achieves more than margin t on x but has loss at most m on F. Applying Lemma 2.19 we can find a subsequence $\boldsymbol{\lambda}^{(t)}$ whose tail achieves vanishingly small loss on some non-empty subset S of F containing x, and bounded margins in $F \setminus S$. Applying Lemma 2.20 to $\boldsymbol{\lambda}^{(t)}$ we get some convex combination $\boldsymbol{\lambda}^\dagger$ which has positive margins on S and zero margin on $F \setminus S$. Let $\boldsymbol{\eta}^*$ be as in Lemma 2.21, a finite combination achieving the optimal loss on F. Then $\boldsymbol{\eta}^* + \infty \cdot \boldsymbol{\lambda}^\dagger$ achieves the same loss on every example in $F \setminus S$ as the optimal solution $\boldsymbol{\eta}^*$, but zero loss for examples in S. This solution is strictly better than $\boldsymbol{\eta}^*$ on F, a contradiction to the optimality of $\boldsymbol{\eta}^*$. Therefore our assumption is false, and some finite upper bound $\mu_{\max}$ on the margins $(\mathbf{M}\boldsymbol{\eta})_i$ of vectors satisfying $\ell^{\boldsymbol{\eta}}(F) \leq m$ exists. □

2.3.3 Investigating the constants

In this section, we try to estimate the constant C in Theorem 2.14. We show that it can be arbitrarily large for adversarial feature matrices with real entries (corresponding to confidence rated weak hypotheses), but has an upper-bound doubly exponential in the number of examples when the feature matrix has $\{-1, 0, +1\}$ entries only. We also show that this doubly exponential bound cannot be improved without significantly changing the proof in the previous section.

By inspecting the proofs, we can bound the constant in Theorem 2.14 as follows.

Corollary 2.23. *The constant C in Theorem 2.14 that emerges from the proofs is*

$$C = \frac{32m^3 N e^{\mu_{\max}}}{\gamma^2 \lambda_{\min}^2},$$

where m is the number of examples, N is the number of hypotheses, γ and $\mu_{\max}$ are as given by Items 1 and 3 of the decomposition lemma, and $\lambda_{\min}^2$ is the smallest positive eigenvalue of $\mathbf{M}_F^T\mathbf{M}_F$ ($\mathbf{M}_F$ is the feature matrix restricted to the rows belonging to the finite margin set F).

Our bound on C will be obtained by in turn bounding the quantities $\lambda_{\min}^{-1}, \gamma^{-1}, \mu_{\max}$. These are strongly related to the singular values of the feature matrix $\mathbf{M}$, and in general cannot be easily measured. In fact, when $\mathbf{M}$ has real entries, we have already seen in Section 2.2.3 that the rate can be arbitrarily large, implying these parameters can have very large values. Even when the matrix $\mathbf{M}$ has integer entries (that is, $-1, 0, +1$), the next lemma shows that these quantities can be exponential in the number of examples.

Lemma 2.24. *There are examples of feature matrices with $-1, 0, +1$ entries and at most m rows or columns (where $m > 10$) for which the quantities $\gamma^{-1}, \lambda^{-1}$ and $\mu_{\max}$ are at least $\Omega(2^m/m)$.*

Proof. We first show the bounds for γ and λ. Let $\mathbf{M}$ be an $m \times m$ upper triangular matrix with $+1$ on the diagonal, and -1 above the diagonal. Let $\mathbf{y} = (2^{m-1}, 2^{m-2}, \ldots, 1)^T$, and $\mathbf{b} = (1, 1, \ldots, 1)^T$. Then $\mathbf{My} = \mathbf{b}$, although the $\mathbf{y}$ has much bigger norm than $\mathbf{b}$: $\|\mathbf{y}\| \geq 2^{m-1}$, while $\|\mathbf{b}\| = m$. Since $\mathbf{M}$ is invertible, by the definition of $\lambda_{\min}$, we have $\|\mathbf{My}\| \geq \lambda_{\min}\|\mathbf{y}\|$, so that $\lambda_{\min}^{-1} \geq \|\mathbf{y}\|/\|\mathbf{My}\| \geq 2^m/m$. Next, note that $\mathbf{y}$ produces all positive margins $\mathbf{b}$, and hence the zero-loss set consists of all the examples. In particular, if $\boldsymbol{\eta}^\dagger$ be as in Item 1 of the decomposition lemma, then the vector $\gamma^{-1}\boldsymbol{\eta}^\dagger$ achieves more than 1 margin on each example: $\mathbf{M}(\gamma^{-1}\boldsymbol{\eta}^\dagger) \geq \mathbf{b}$. On the other hand, our matrix is very similar to the one in Lemma 2.10, and the same arguments in the proof of that lemma can be used to show that if for some $\mathbf{x}$ we have $(\mathbf{Mx}) \geq \mathbf{b}$, then $\mathbf{x} \geq \mathbf{y}$. This implies that $\gamma^{-1}\|\boldsymbol{\eta}^\dagger\|_1 \geq \|\mathbf{y}\|_1 = (2^m - 1)$. Since $\boldsymbol{\eta}^\dagger$ has unit ℓ_1-norm, the bound on γ^{-1} follows too.

Next we provide an example showing $\mu_{\max}$ can be $\Omega(2^m/m)$. Consider an $m \times (m-1)$ matrix $\mathbf{M}$. The bottom row of $\mathbf{M}$ is all $+1$. The upper $(m-1) \times (m-1)$ submatrix of $\mathbf{M}$ is a lower triangular matrix with -1 on the diagonal and $+1$ below the diagonal. Observe that if $\mathbf{y}^T = (2^{m-2}, 2^{m-3}, \ldots, 1, 1)$, then $\mathbf{y}^T\mathbf{M} = \mathbf{0}$. Therefore, for any vector $\mathbf{x}$, the inner product of the margins $\mathbf{Mx}$ with $\mathbf{y}$ is zero: $\mathbf{y}^T M\mathbf{x} = 0$. This implies that achieving positive margin on any example forces some other example to receive negative margin. By Item 1 of the decomposition lemma, the zero loss set in this dataset is empty, and all the examples belong to the finite loss set. Next, we choose a combination with at most m loss that nevertheless achieves $\Omega(2^m/m)$

positive margin on some example. Let $\mathbf{x}^T = (1, 2, 4, \ldots, 2^{m-2})$. Then $(\mathbf{M}\mathbf{x})^T = (-1, -1, \ldots, -1, 2^{m-1} - 1)$. Then the margins using $\varepsilon\mathbf{x}$ are $(-\varepsilon, \ldots, -\varepsilon, \varepsilon(2^{m-1} - 1))$ with total loss $(m-1)e^{\varepsilon} + e^{\varepsilon(1-2^{m-1})}$. Choose $\varepsilon = 1/(2m) \leq 1$, so that the loss on examples corresponding to the first $m-1$ rows is at most $e^{\varepsilon} \leq 1 + 2\varepsilon = 1 + 1/m$, where the first inequality holds since $\varepsilon \in [0, 1]$. For $m > 10$, the choice of ε guarantees $1/(2m) = \varepsilon \geq (\ln m)/(2^{m-1} - 1)$, so that the loss on the example corresponding to the bottom most row is $e^{-\varepsilon(2^{m-1}-1)} \leq e^{-\ln m} = 1/m$. Therefore the net loss of $\varepsilon\mathbf{x}$ is at most $(m-1)(1+1/m) + 1/m = m$. On the other hand the margin on the example corresponding to the last row is $\varepsilon(2^{m-1} - 1) = (2^{m-1} - 1)/(2m) = \Omega(2^m/m)$. $\square$

The above result implies any bound on C derived from Corollary 2.23 will be at least $2^{\Omega(2^m/m)}$ in the worst case. This does not imply that the best bound one can hope to prove is doubly exponential, only that our techniques in the previous section do not admit anything better. We next show that the bounds in Lemma 2.24 are nearly the worst possible.

Lemma 2.25. *Suppose each entry of* $\mathbf{M}$ *is* $-1, 0$ *or* $+1$*. Then each of the quantities* $\lambda_{\min}^{-1}, \gamma^{-1}$ *and* $\mu_{\max}$ *are at most* $2^{O(m \ln m)}$*.*

The proof of Lemma 2.25 is rather technical, and we defer it to the Appendix. Lemma 2.25 and Corollary 2.23 together imply a convergence rate of $2^{2^{O(m \ln m)}}/\varepsilon$ to the optimal loss for integer matrices. This bound on C is exponentially worse than the $\Omega(2^m)$ lower bound on C we saw in Section 2.2.3, a price we pay for obtaining optimal dependence on ε. In the next section we will see how to obtain $\mathrm{poly}(2^{m \ln m}, \varepsilon^{-1})$ bounds, although with a worse dependence on ε. We end this section by showing, just for completeness, how a bound on the norm of $\boldsymbol{\eta}^*$ as defined in Item 2 of the decomposition lemma follows as a quick corollary to Lemma 2.25.

Corollary 2.26. *Suppose* $\boldsymbol{\eta}^*$ *is as given by Item 2 of the decomposition lemma. When the feature matrix has only* $-1, 0, +1$ *entries, we may bound* $\|\boldsymbol{\eta}^*\|_1 \leq 2^{O(m \ln m)}$*.*

Proof. Note that every entry of $\mathbf{M}_F\boldsymbol{\eta}^*$ lies in the range $[-\ln m, \mu_{\max} = 2^{O(m \ln m)}]$, and hence $\|\mathbf{M}_F\boldsymbol{\eta}^*\| \leq 2^{O(m \ln m)}$. Next, we may choose $\boldsymbol{\eta}^*$ orthogonal to the null space of $\mathbf{M}_F$; then $\|\boldsymbol{\eta}^*\| \leq \lambda_{\min}^{-1}\|\mathbf{M}_F\boldsymbol{\eta}^*\| \leq 2^{O(m \ln m)}$. Since $\|\boldsymbol{\eta}^*\|_1 \leq \sqrt{N}\|\boldsymbol{\eta}^*\|$, and the number of possible columns N with $\{-1, 0, +1\}$ entries is at most 3^m, the proof follows. $\square$

2.4 Improved Estimates

In this section we shed more light on the rate bounds by cross-application of techniques from Sections 2.2 and 2.3. We obtain both new upper bounds for convergence to the optimal loss, as well as lower bounds for convergence to an arbitrary target loss. We also indicate what we believe might be the optimal bounds for either situation.

We first show how the finite rate bound of Theorem 2.1 along with the decomposition lemma yields a new rate of convergence to the optimal loss. Although the dependence on ε is worse than in Theorem 2.14, the dependence on m is nearly optimal. We will need the following key application of the decomposition lemma.

Lemma 2.27. *When the feature matrix has $-1, 0, +1$ entries, for any $\varepsilon > 0$, there is some solution with ℓ_1-norm at most $2^{O(m \ln m)} \ln(1/\varepsilon)$ that achieves within ε of the optimal loss.*

Proof. Let $\boldsymbol{\eta}^*, \boldsymbol{\eta}^\dagger, \gamma$ be as given by the decomposition lemma. Let $c = \min_{i \in Z} (\mathbf{M}\boldsymbol{\eta}^*)_i$ be the minimum margin produced by $\boldsymbol{\eta}^*$ on any example in the zero-loss set Z. Then $\boldsymbol{\eta}^* - c\boldsymbol{\eta}^\dagger$ produces non-negative margins on Z, and the optimal margins on the finite loss set F. Therefore, the vector $\boldsymbol{\lambda}^* = \boldsymbol{\eta}^* + (\ln(1/\varepsilon)\gamma^{-1} - c)\, \boldsymbol{\eta}^\dagger$ achieves at least $\ln(1/\varepsilon)$ margin on every example in Z, and optimal margins on the finite loss set F. Hence $L(\boldsymbol{\lambda}^*) \leq \inf_{\boldsymbol{\lambda}} L(\boldsymbol{\lambda}) + \varepsilon$. Using $|c| \leq \|\mathbf{M}\boldsymbol{\eta}^*\| \leq m\|\boldsymbol{\eta}^*\|$, and the results in Corollary 2.26 and Lemma 2.25, we may conclude the vector $\boldsymbol{\lambda}^*$ has ℓ_1-norm at most $2^{O(m \ln m)} \ln(1/\varepsilon)$. □

We may now invoke Theorem 2.1 to obtain a $2^{O(m \ln m)} \ln^6(1/\varepsilon)\varepsilon^{-5}$ rate of convergence to the optimal solution. Rate bounds with similar dependence on m and slightly better dependence on ε can be obtained by modifying the proof in Section 2.3 to use first order instead of second order techniques. In that way we may obtain a $\text{poly}(\lambda_{\min}^{-1}, \gamma^{-1}, \mu_{\max})\varepsilon^{-3} = 2^{O(m \ln m)}\varepsilon^{-3}$ rate bound. We omit the the rather long but straightforward proof of this fact. Finally, note that if Conjecture 2.6 is true, then Lemma 2.27 implies a $2^{O(m \ln m)} \ln(1/\varepsilon)\varepsilon^{-1}$ rate bound for converging to the optimal loss, which is nearly optimal in both m and ε. We state this as an independent conjecture.

Conjecture 2.28. *For feature matrices with $-1, 0, +1$ entries, AdaBoost converges to within ε of the optimal loss within $2^{O(m \ln m)}\varepsilon^{-(1+o(1))}$ rounds.*

We next focus on lower bounds on the convergence rate to arbitrary target losses discussed in Section 2.2. We begin by showing the rate dependence on the norm of the solution as given in Lemma 2.9 holds for much more general datasets.

Lemma 2.29. *Suppose a feature matrix has only ± 1 entries, and the finite loss set is non-empty. Then, for any coordinate descent procedure, the number of rounds required to achieve a target loss ϕ^* is at least*

$$\inf \{\|\boldsymbol{\lambda}\|_1 : L(\boldsymbol{\lambda}) \leq \phi^*\} / (1 + \ln m).$$

Proof. It suffices to upper-bound the step size $|\alpha_t|$ in any round t by at most $1 + \ln m$. Notice that when the feature matrix has ± 1 entries, a step in a direction that does not end up increasing the loss is at most of length $(1/2)\ln((1+\delta)/(1-\delta))$, where δ is the edge in that direction. Therefore, if δ_t is the maximum edge achievable in any direction, we have

$$|\alpha_t| \leq \frac{1}{2} \ln\left(\frac{1+\delta_t}{1-\delta_t}\right).$$

Further, by (2.4), a large edge δ_t ensures that for some coordinate step, the new vector $\boldsymbol{\lambda}^t$ will have much smaller loss than the vector $\boldsymbol{\lambda}^{t-1}$ at the beginning of round t: $L(\boldsymbol{\lambda}^t) \leq L(\boldsymbol{\lambda}^{t-1})\sqrt{1-\delta_t^2}$. On the other hand, before the step, the loss is at most 1, $L(\boldsymbol{\lambda}^{t-1}) \leq 1$, and after the step the loss is at most $1/m$ (since the optimal loss on a dataset with non-empty finite set is at least $1/m$): $L(\boldsymbol{\lambda}^t) \geq 1/m$. Combining these inequalities we get

$$1/m \leq L(\boldsymbol{\lambda}^t) \leq L(\boldsymbol{\lambda}^{t-1})\sqrt{1-\delta_t^2} \leq \sqrt{1-\delta_t^2},$$

that is, $\sqrt{1-\delta_t^2} \geq 1/m$. Now the step length can be bounded as

$$|\alpha_t| \leq \frac{1}{2}\ln\left(\frac{1+\delta_t}{1-\delta_t}\right) = \ln(1+\delta_t) - \frac{1}{2}\ln(1-\delta_t^2) \leq \delta_t + \ln m \leq 1 + \ln m.$$

□

We end by showing a new lower bound for the convergence rate to an arbitrary target loss studied in Section 2.2. Corollary 2.11 implies that the rate bound in Theorem 2.1 has to be at least polynomially large in the norm of the solution. We now show that a polynomial dependence on ε^{-1} in the rate is unavoidable too. This shows that rates for competing with a finite solution are different from rates on a dataset where the optimum loss is achieved by a finite solution, since in the latter we may achieve a $O(\ln(1/\varepsilon))$ rate.

Corollary 2.30. *Consider any dataset (e.g. the one in Figure 2.4) for which $\Omega(1/\varepsilon)$ rounds are necessary to get within ε of the optimal loss. If there are constants c and β such that for any $\boldsymbol{\lambda}^*$ and ε, a loss of $L(\boldsymbol{\lambda}^*)+\varepsilon$ can be achieved in at most $O(\|\boldsymbol{\lambda}^*\|_1^c \varepsilon^{-\beta})$ rounds, then $\beta \geq 1$.*

Proof. The decomposition lemma implies that $\boldsymbol{\lambda}^* = \boldsymbol{\eta}^* + \ln(2/\varepsilon)\boldsymbol{\eta}^\dagger$ with ℓ_1-norm $O(\ln(1/\varepsilon))$ achieves loss at most $K + \varepsilon/2$ (recall K is the optimal loss). Suppose the corollary fails to hold for constants c and $\beta \leq 1$. Then $L(\boldsymbol{\lambda}^*) + \varepsilon/2 = K + \varepsilon$ loss can be achieved in $O(\varepsilon^{-\beta})/\ln^c(1/\varepsilon)) = o(1/\varepsilon)$ rounds, contradicting the $\Omega(1/\varepsilon)$ lower bound. □

2.5 Conclusion

In this chapter we studied the convergence rate of AdaBoost with respect to the exponential loss. We showed upper and lower bounds for convergence rates to both an arbitrary target loss achieved by some finite combination of the weak hypotheses, as well as to the infimum loss which may not be realizable. For the first convergence rate, we showed a strong relationship exists between the size of the minimum vector achieving a target loss and the number of rounds of coordinate descent required to achieve that loss. In particular, we showed that a polynomial dependence of the rate on the ℓ_1-norm B of the minimum size solution is absolutely necessary, and that a poly$(B, 1/\varepsilon)$ upper bound holds, where ε is the accuracy parameter. The actual rate we derive has rather large exponents, and we discuss a minor variant of AdaBoost that achieves a much tighter and near optimal rate.

For the second kind of convergence, using entirely separate techniques, we derived a C/ε upper bound, and showed that this is tight up to constant factors. In the

Convergence rate with respect to:	Reference solution (Section 2.2)	Optimal solution (Section 2.3)
Upper bounds:	$13B^6/\varepsilon^5$	$\text{poly}(e^{\mu_{\max}}, \lambda_{\min}^{-1}, \gamma^{-1})/\varepsilon \leq 2^{2^{O(m \ln m)}}/\varepsilon$
		$\text{poly}(\mu_{\max}, \lambda_{\min}^{-1}, \gamma^{-1})/\varepsilon^3 \leq 2^{O(m \ln m)}/\varepsilon^3$
Lower bounds with:	$(B/\varepsilon)^{1-\nu}$ for any $\nu > 0$	$\max\left\{\frac{2^m \ln(1/\varepsilon)}{\ln m}, \frac{2}{9\varepsilon}\right\}$
a) $\{0, \pm 1\}$ entries	$(2^m/\ln m)\ln(1/\varepsilon)$	
b) real entries	Can be arbitrarily large even when m, N, ε are held fixed	
Conjectured upper bounds:	$O(B^2/\varepsilon)$	$2^{O(m \ln m)}/\varepsilon^{1+o(1)}$, if entries in $\{0, \pm 1\}$

Figure 2.5: Summary of our most important results and conjectures regarding the convergence rate of AdaBoost. Here m refers to the number of training examples, and ε is the accuracy parameter. The quantity B is the ℓ_1-norm of the reference solution used in Section 2.2. The parameters $\lambda_{\min}$, γ and $\mu_{\max}$ depend on the dataset and are defined and studied in Section 2.3.

process, we showed a certain decomposition lemma that might be of independent interest. We also study the constants and show how they depend on certain intrinsic parameters related to the singular values of the feature matrix. We estimate the worst case values of these parameters, and considering feature matrices with only $\{-1, 0, +1\}$ entries, this leads to a bound on the rate constant C that is doubly exponential in the number of training examples. Since this is rather large, we also include bounds polynomial in both the number of training examples and the accuracy parameter ε, although the dependence on ε in these bounds is non-optimal.

Finally, for each kind of convergence, we conjecture tighter bounds that are not known to hold presently. A table containing a summary of the results in this chapter is included in Figure 2.5.

2.6 Appendix

2.6.1 Lower bound for convergence to optimal loss

Lemma 2.31. *For any $\varepsilon < 1/3$, to get within ε of the optimum loss on the dataset in Table 2.4, AdaBoost takes at least $2/(9\varepsilon)$ steps.*

Proof. Note that the optimal loss is 2/3, and we are bounding the number of rounds necessary to get within $(2/3) + \varepsilon$ loss for $\varepsilon < 1/3$. We will compute the edge in each round analytically. Let w_a^t, w_b^t, w_c^t denote the normalized-losses (adding up to 1) or weights on examples a, b, c at the beginning of round t, h_t the weak hypothesis chosen in round t, and δ_t the edge in round t. The values of these parameters are shown below for the first 5 rounds, where we have assumed (without loss of generality) that the hypothesis picked in round 1 is $\hbar_b$:

Round	w_a^t	w_b^t	w_c^t	h_t	δ_t
$t = 1:$	1/3	1/3	1/3	$\hbar_b$	1/3
$t = 2:$	1/2	1/4	1/4	$\hbar_a$	1/2
$t = 3:$	1/3	1/2	1/6	$\hbar_b$	1/3
$t = 4:$	1/2	3/8	1/8	$\hbar_a$	1/4
$t = 5:$	2/5	1/2	1/10	$\hbar_b$	1/5.

Based on the patterns above, we first claim that for rounds $t \geq 2$, the edge achieved is $1/t$. In fact we prove the stronger claims, that for rounds $t \geq 2$, the following hold:

1. One of w_a^t and w_b^t is 1/2.

2. $\delta_{t+1} = \delta_t/(1 + \delta_t)$.

Since $\delta_2 = 1/2$, the recurrence on δ_t would immediately imply $\delta_t = 1/t$ for $t \geq 2$. We prove the stronger claims by induction on the round t. The base case for $t = 2$ is shown above and may be verified. Suppose the inductive assumption holds for t. Assume without loss of generality that $1/2 = w_a^t > w_b^t > w_c^t$; note this implies $w_b^t = 1 - (w_a^t + w_c^t) = 1/2 - w_c^t$. Further, in this round, $\hbar_a$ gets picked, and has edge $\delta_t = w_a^t + w_c^t - w_b^t = 2w_c^t$. Now for any dataset, the weights of the examples labeled correctly and incorrectly in a round of AdaBoost are rescaled during the weight update step in a way such that each add up to 1/2 after the rescaling. Therefore, $w_b^{t+1} = 1/2, w_c^{t+1} = w_c^t\left(\frac{1/2}{w_a^t + w_c^t}\right) = w_c^t/(1 + 2w_c^t)$. Hence, $\hbar_b$ gets picked in round $t+1$ and, as before, we get edge $\delta_{t+1} = 2w_c^{t+1} = 2w_c^t/(1 + 2w_c^t) = \delta_t/(1 + \delta_t)$. The proof of our claim follows by induction.

Next we find the loss after each iteration. Using $\delta_1 = 1/3$ and $\delta_t = 1/t$ for $t \geq 2$, the loss after T rounds can be written as

$$\prod_{t=1}^{T} \sqrt{1 - \delta_t^2} = \sqrt{1 - (1/3)^2} \prod_{t=2}^{T} \sqrt{1 - 1/t^2} = \frac{2\sqrt{2}}{3} \sqrt{\prod_{t=2}^{T} \left(\frac{t-1}{t}\right)\left(\frac{t+1}{t}\right)}.$$

The product can be rewritten as follows:

$$\prod_{t=2}^{T}\left(\frac{t-1}{t}\right)\left(\frac{t+1}{t}\right)=\left(\prod_{t=2}^{T}\frac{t-1}{t}\right)\left(\prod_{t=2}^{T}\frac{t+1}{t}\right)=\left(\prod_{t=2}^{T}\frac{t-1}{t}\right)\left(\prod_{t=3}^{T+1}\frac{t}{t-1}\right).$$

Notice almost all the terms cancel, except for the first term of the first product, and the last term of the second product. Therefore, the loss after T rounds is

$$\frac{2\sqrt{2}}{3}\sqrt{\left(\frac{1}{2}\right)\left(\frac{T+1}{T}\right)}=\frac{2}{3}\sqrt{1+\frac{1}{T}}\geq\frac{2}{3}\left(1+\frac{1}{3T}\right)=\frac{2}{3}+\frac{2}{9T},$$

where the inequality holds for $T\geq 1$. Since the initial error is $1=(2/3)+1/3$, therefore, for any $\varepsilon<1/3$, the number of rounds needed to achieve loss $(2/3)+\varepsilon$ is at least $2/(9\varepsilon)$. $\square$

2.6.2 A useful technical result

Here we prove a technical result that was used for proving the various rate upper bounds.

Lemma 2.32. *Suppose $u_0, u_1, \dots,$ are non-negative numbers satisfying*

$$u_t - u_{t+1} \geq c_0 u_t^{1+c_1},$$

for some non-negative constants c_0, c_1. Then, for any t,

$$\frac{1}{u_t^{c_1}}-\frac{1}{u_0^{c_1}}\geq c_1c_0t.$$

Proof. By induction on t. The base case is an identity. Assume the statement holds at iteration t. Then,

$$\frac{1}{u_{t+1}^{c_1}}-\frac{1}{u_0^{c_1}}=\left(\frac{1}{u_{t+1}^{c_1}}-\frac{1}{u_t^{c_1}}\right)+\left(\frac{1}{u_t^{c_1}}-\frac{1}{u_0^{c_1}}\right)\geq\frac{1}{u_{t+1}^{c_1}}-\frac{1}{u_t^{c_1}}+c_1c_0t \text{ (by inductive hypothesis).}$$

Thus it suffices to show $1/u_{t+1}^{c_1}-1/u_t^{c_1}\geq c_1c_0$. Multiplying both sides by $u_t^{c_1}$ and adding 1, this is equivalent to showing $(u_t/u_{t+1})^{c_1}\geq 1+c_1c_0u_t^{c_1}$. We will in fact show the stronger inequality

$$(u_t/u_{t+1})^{c_1}\geq\left(1+c_0u_t^{c_1}\right)^{c_1}. \tag{2.9}$$

Since $(1+a)^b\geq 1+ba$ for a,b non-negative, (2.9) will imply $(u_t/u_{t+1})^{c_1}\geq(1+c_0u_t^{c_1})^{c_1}\geq 1+c_1c_0u_t^{c_1}$, which will complete our proof. To show (2.9), we first rearrange the condition on u_t, u_{t+1} to obtain

$$u_{t+1}\leq u_t\left(1-c_0u_t^{c_1}\right)\implies\frac{u_t}{u_{t+1}}\geq\frac{1}{1-c_0u_t^{c_1}}.$$

Applying the fact $(1 + c_0 u_t^{c_1})(1 - c_0 u_t^{c_1}) \leq 1$ to the previous equation we get,

$$\frac{u_t}{u_{t+1}} \geq 1 + c_0 u_t^{c_1}.$$

Since $c_1 \geq 0$, we may raise both sides of the above inequality to the power of c_1 to show (2.9), finishing our proof. $\square$

2.6.3 Proof of Lemma 2.25

In this section we prove Lemma 2.25, by separately bounding the quantities $\lambda_{\min}^{-1}$, γ^{-1} and $\mu_{\max}$, through a sequence of Lemmas. We will use the next result repeatedly.

Lemma 2.33. *If* $\mathbf{A}$ *is an* $n \times n$ *invertible matrix with* $-1, 0, +1$ *entries, then* $\min_{\mathbf{x}:\|\mathbf{x}\|=1} \|\mathbf{A}\mathbf{x}\|$ *is at least* $1/n! = 2^{-O(n \ln n)}$.

Proof. It suffices to show that $\|\mathbf{A}^{-1}\mathbf{x}\| \leq n!$ for any $\mathbf{x}$ with unit norm. Now $\mathbf{A}^{-1} = \mathrm{adj}(\mathbf{A})/\det(\mathbf{A})$ where $\mathrm{adj}(\mathbf{A})$ is the adjoint of $\mathbf{A}$, whose i,j-th entry is the i,jth cofactor of $\mathbf{A}$ (given by $(-1)^{i+j}$ times the determinant of the $n-1 \times n-1$ matrix obtained by removing the ith row and jth column of $\mathbf{A}$), and $\det(\mathbf{A})$ is the determinant of $\mathbf{A}$. The determinant of any $k \times k$ matrix G can be written as $\sum_\sigma \mathrm{sgn}(\sigma) \prod_{i=1}^k G(i, \sigma(j))$, where σ ranges over all the permutations of $1, \ldots, k$. Therefore each entry of $\mathrm{adj}(\mathbf{A})$ is at most $(n-1)!$, and $\det(\mathbf{A})$ is a non-zero integer. Therefore $\|\mathbf{A}^{-1}\mathbf{x}\| = \|\mathrm{adj}(\mathbf{A})\mathbf{x}\| / \det(\mathbf{A}) \leq n! \|\mathbf{x}\|$, and the proof is complete. $\square$

We first show our bound holds for $\lambda_{\min}$.

Lemma 2.34. *Suppose* $\mathbf{M}$ *has* $-1, 0, +1$ *entries, and let* $\mathbf{M}_F, \lambda_{\min}$ *be as in Corollary 2.23. Then* $\lambda_{\min} \geq 1/m!$.

Proof. Let $\mathbf{A}$ denote the matrix $\mathbf{M}_F$. It suffices to show that $\mathbf{A}$ does not squeeze too much the norm of any vector orthogonal to the null-space $\ker \mathbf{A} \triangleq \{\boldsymbol{\eta} : \mathbf{A}\boldsymbol{\eta} = \mathbf{0}\}$ of $\mathbf{A}$, i.e. $\|\mathbf{A}\boldsymbol{\lambda}\| \geq (1/m!)\|\boldsymbol{\lambda}\|$ for any $\boldsymbol{\lambda} \in \ker \mathbf{A}^\perp$. We first characterize $\ker \mathbf{A}^\perp$ and then study how $\mathbf{A}$ acts on this subspace.

Let the rank of $\mathbf{A}$ be $k \leq m$ (notice $\mathbf{A} = \mathbf{M}_F$ has N columns and fewer than m rows). Without loss of generality, assume the first k columns of $\mathbf{A}$ are independent. Then every column of $\mathbf{A}$ can be written as a linear combination of the first k columns of $\mathbf{A}$, and we have $\mathbf{A} = \mathbf{A}'[\mathbf{I}|\mathbf{B}]$ (that is, the matrix $\mathbf{A}$ is the product of matrices $\mathbf{A}'$ and $[\mathbf{I}|\mathbf{B}]$), where $\mathbf{A}'$ is the submatrix consisting of the first k columns of $\mathbf{A}$, $\mathbf{I}$ is the $k \times k$ identity matrix, and $\mathbf{B}$ is some $k \times (N-k)$ matrix of linear combinations (here $|$ denotes concatenation). The null-space of $\mathbf{A}$ consists of $\mathbf{x}$ such that $\mathbf{0} = \mathbf{A}\mathbf{x} = \mathbf{A}'[\mathbf{I}|\mathbf{B}]\mathbf{x} = \mathbf{A}'(\mathbf{x}_k + \mathbf{B}\mathbf{x}_{-k})$, where $\mathbf{x}_k$ is the first k coordinates of $\mathbf{x}$, and $\mathbf{x}_{-k}$ the remaining $N-k$ coordinates. Since the columns of $\mathbf{A}'$ are independent, this happens if and only if $\mathbf{x}_k = -\mathbf{B}\mathbf{x}_{-k}$. Therefore $\ker \mathbf{A} = \left\{(-\mathbf{B}\mathbf{z}, \mathbf{z}) : \mathbf{z} \in \mathbb{R}^{N-k}\right\}$. Since a vector $\mathbf{x}$ lies in the orthogonal subspace of $\ker \mathbf{A}$ if it is orthogonal to every vector in the latter, we have

$$\ker \mathbf{A}^\perp = \left\{(\mathbf{x}_k, \mathbf{x}_{-k}) : \langle \mathbf{x}_k, \mathbf{B}\mathbf{z} \rangle = \langle \mathbf{x}_{-k}, \mathbf{z} \rangle, \forall \mathbf{z} \in \mathbb{R}^{N-K}\right\}.$$

We next see how $\mathbf{A}$ acts on this subspace. Recall $\mathbf{A} = \mathbf{A}'[\mathbf{I}|\mathbf{B}]$ where $\mathbf{A}'$ has k independent columns. By basic linear algebra, the row rank of $\mathbf{A}'$ is also k, and assume without loss of generality that the first k rows of $\mathbf{A}'$ are independent. Denote by $\mathbf{A}_k$ the $k \times k$ submatrix of $\mathbf{A}'$ formed by these k rows. Then for any vector $\mathbf{x}$,

$$\|\mathbf{A}\mathbf{x}\| = \|\mathbf{A}'[\mathbf{I}|\mathbf{B}]\mathbf{x}\| = \|\mathbf{A}'(\mathbf{x}_k + \mathbf{B}\mathbf{x}_{-k})\| \geq \|\mathbf{A}_k(\mathbf{x}_k + \mathbf{B}\mathbf{x}_{-k})\| \geq \frac{1}{k!}\|\mathbf{x}_k + \mathbf{B}\mathbf{x}_{-k}\|,$$

where the last inequality follows from Lemma 2.33. To finish the proof, it suffices to show that $\|\mathbf{x}_k+\mathbf{B}\mathbf{x}_{-k}\| \geq \|\mathbf{x}\|$ for $\mathbf{x} \in \ker \mathbf{A}^{\perp}$. Indeed, by expanding out $\|\mathbf{x}_k+\mathbf{B}\mathbf{x}_{-k}\|^2$ as inner product with itself, we have

$$\|\mathbf{x}_k + \mathbf{B}\mathbf{x}_{-k}\|^2 = \|\mathbf{x}_k\|^2 + \|\mathbf{B}\mathbf{x}_{-k}\|^2 + 2\left\langle \mathbf{x}_k, \mathbf{B}\mathbf{x}_{-k}\right\rangle \geq \|\mathbf{x}_k\|^2 + 2\|\mathbf{x}_{-k}\|^2 \geq \|\mathbf{x}\|^2,$$

where the first inequality follows since $\mathbf{x} \in \ker \mathbf{A}^{\perp}$ implies $\langle \mathbf{x}_k, \mathbf{B}\mathbf{x}_{-k}\rangle = \langle \mathbf{x}_{-k}, \mathbf{x}_{-k}\rangle$. □

To show the bounds on γ^{-1} and $\mu_{\max}$, we will need an intermediate result.

Lemma 2.35. *Suppose $\mathbf{A}$ is a matrix, and $\mathbf{b}$ a vector, both with $-1, 0, 1$ entries. If $\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ is solvable, then there is a solution satisfying $\|\mathbf{x}\| \leq k \cdot k!$, where $k = \operatorname{rank}(\mathbf{A})$.*

Proof. Pick a solution $\mathbf{x}$ with maximum number of zeroes. Let J be the set of coordinates for which x_i is zero. We first claim that there is no other solution $\mathbf{x}'$ which is also zero on the set J. Suppose there were such an $\mathbf{x}'$. Note any point $\mathbf{p}$ on the infinite line joining $\mathbf{x}, \mathbf{x}'$ satisfies $\mathbf{A}\mathbf{p} = \mathbf{b}$, and $\mathbf{p}_J = \mathbf{0}$ (that is, $p_{i'} = 0$ for $i' \in J$). If i is any coordinate not in J such that $x_i \neq x_i'$, then for some point $\mathbf{p}^i$ along the line, we have $\mathbf{p}^i_{J\cup\{i\}} = \mathbf{0}$. Choose i so that $\mathbf{p}^i$ is as close to $\mathbf{x}$ as possible. Since $\mathbf{x} \geq \mathbf{0}$, by continuity this would also imply that $\mathbf{p}^i \geq \mathbf{0}$. But then $\mathbf{p}^i$ is a solution with more zeroes than $\mathbf{x}$, a contradiction.

The claim implies that the reduced problem $\mathbf{A}'\tilde{\mathbf{x}} = \mathbf{b}, \tilde{\mathbf{x}} \geq \mathbf{0}$, obtained by substituting $\mathbf{x}_J = \mathbf{0}$, has a unique solution. Let $k = \operatorname{rank}(\mathbf{A}')$, $\mathbf{A}_k$ be a $k \times k$ submatrix of $\mathbf{A}'$ with full rank, and $\mathbf{b}_k$ be the restriction of $\mathbf{b}$ to the rows corresponding to those of $\mathbf{A}_k$ (note that $\mathbf{A}'$, and hence $\mathbf{A}_k$, contain only $-1, 0, +1$ entries). Then, $\mathbf{A}_k\tilde{\mathbf{x}} = \mathbf{b}_k, \tilde{\mathbf{x}} \geq \mathbf{0}$ is equivalent to the reduced problem. In particular, by uniqueness, solving $\mathbf{A}_k\tilde{\mathbf{x}} = \mathbf{b}_k$ automatically ensures the obtained $\mathbf{x} = (\tilde{\mathbf{x}}, \mathbf{0}_J)$ is a non-negative solution to the original problem, and satisfies $\|\mathbf{x}\| = \|\tilde{\mathbf{x}}\|$. But, by Lemma 2.33,

$$\|\tilde{\mathbf{x}}\| \leq k!\|\mathbf{A}_k\tilde{\mathbf{x}}\| = k!\|\mathbf{b}_k\| \leq k \cdot k!.$$

□

The bound on γ^{-1} follows easily.

Lemma 2.36. *Let $\gamma, \boldsymbol{\eta}^{\dagger}$ be as in Item 1 of Lemma 2.15. Then $\boldsymbol{\eta}^{\dagger}$ can be chosen such that $\gamma \geq 1/\left(\sqrt{N}m \cdot m!\right) \geq 2^{-O(m \ln m)}$.*

Proof. We know that $\mathbf{M}(\boldsymbol{\eta}^\dagger/\gamma) = \mathbf{b}$, where $\mathbf{b}$ is zero on the set F and at least 1 for every example in the zero loss set Z (as given by Item 1 of Lemma 2.15). Since $\mathbf{M}$ is closed under complementing columns, we may assume in addition that $\boldsymbol{\eta}^\dagger \geq \mathbf{0}$. Introduce slack variables z_i for $i \in Z$, and let $\tilde{\mathbf{M}}$ be $\mathbf{M}$ augmented with the columns $-\mathbf{e}_i$ for $i \in Z$, where $\mathbf{e}_i$ is the standard basis vector with 1 on the ith coordinate and zero everywhere else. Then, by setting $\mathbf{z} = \mathbf{M}(\boldsymbol{\eta}^\dagger/\gamma) - \mathbf{b}$, we have a solution $(\boldsymbol{\eta}^\dagger/\gamma, \mathbf{z})$ to the system $\tilde{\mathbf{M}}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}$. Applying Lemma 2.35, we know there exists some solution $(\mathbf{y}, \mathbf{z}')$ with norm at most $m \cdot m!$ (here $\mathbf{z}'$ corresponds to the slack variables). Observe that $\mathbf{y}/\|\mathbf{y}\|_1$ is a valid choice for $\boldsymbol{\eta}^\dagger$ yielding a γ of $1/\|\mathbf{y}\|_1 \geq 1/(\sqrt{N}m \cdot m!)$. $\square$

To show the bound for $\mu_{\max}$ we will need a version of Lemma 2.35 with strict inequality.

Corollary 2.37. *Suppose $\mathbf{A}$ is a matrix, and $\mathbf{b}$ a vector, both with $-1, 0, 1$ entries. If $\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} > \mathbf{0}$ is solvable, then there is a solution satisfying $\|\mathbf{x}\| \leq 1 + k \cdot k!$, where $k = \operatorname{rank}(\mathbf{A})$.*

Proof. Using Lemma 2.35, pick a solution to $\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ with norm at most $k \cdot k!$. If $\mathbf{x} > \mathbf{0}$, then we are done. Otherwise let $\mathbf{y} > \mathbf{0}$ satisfy $\mathbf{A}\mathbf{x} = \mathbf{b}$, and consider the segment joining $\mathbf{x}$ and $\mathbf{y}$. Every point $\mathbf{p}$ on the segment satisfies $\mathbf{A}\mathbf{p} = b$. Further any coordinate becomes zero at most once on the segment. Therefore, there are points arbitrarily close to $\mathbf{x}$ on the segment with positive coordinates that satisfy the equation, and these have norms approaching that of $\mathbf{x}$. $\square$

We next characterize the feature matrix $\mathbf{M}_F$ restricted to the finite-loss examples, which might be of independent interest.

Lemma 2.38. *If $\mathbf{M}_F$ is the feature matrix restricted to the finite-loss examples F (as given by Item 2 of Lemma 2.15), then there exists a positive linear combination $\mathbf{y} > \mathbf{0}$ such that $\mathbf{M}_F^T\mathbf{y} = \mathbf{0}$.*

Proof. Item 3 of the decomposition lemma states that whenever the loss $\ell^{\mathbf{x}}(F)$ of a vector is bounded by m, then the largest margin $\max_{i \in F}(\mathbf{M}_F\mathbf{x})_i$ is at most $\mu_{\max}$. This implies that there is no vector $\mathbf{x}$ such that $\mathbf{M}_F\mathbf{x} \geq \mathbf{0}$ and at least one of the margins $(\mathbf{M}_F\mathbf{x})_i$ is positive; otherwise, an arbitrarily large multiple of $\mathbf{x}$ would still have loss at most m, but margin exceeding the constant $\mu_{\max}$. In other words, $\mathbf{M}_F\mathbf{x} \geq \mathbf{0}$ implies $\mathbf{M}_F\mathbf{x} = \mathbf{0}$. In particular, the subspace of possible margin vectors $\left\{\mathbf{M}_F\mathbf{x} : \mathbf{x} \in \mathbb{R}^N\right\}$ is disjoint from the convex set Δ_F of distributions over examples in F, which consists of points in $\mathbb{R}^{|F|}$ with all non-negative and at least one positive coordinates. By the Hahn-Banach Separation theorem, there exists a hyperplane separating these two bodies, i.e. there is a $\mathbf{y} \in \mathbb{R}^{|F|}$, such that for any $\mathbf{x} \in \mathbb{R}^N$ and $\mathbf{p} \in \Delta_F$, we have $\langle \mathbf{y}, \mathbf{M}_F\mathbf{x} \rangle \leq 0 < \langle \mathbf{y}, \mathbf{p} \rangle$. By choosing $\mathbf{p} = \mathbf{e}_i$ for various $i \in F$, the second inequality yields $\mathbf{y} > \mathbf{0}$. Since $\mathbf{M}_F\mathbf{x} = -\mathbf{M}_F(-\mathbf{x})$, the first inequality implies that equality holds for all $\mathbf{x}$, i.e. $\mathbf{y}^T\mathbf{M}_F = \mathbf{0}^T$. $\square$

We can finally upper-bound $\mu_{\max}$.

Lemma 2.39. *Let $F, \mu_{\max}$ be as in Items 2,3 of the decomposition lemma. Then $\mu_{\max} \leq \ln m \cdot |F|^{1.5} \cdot |F|! \leq 2^{O(m \ln m)}$.*

Proof. Pick any example $i \in F$ and any combination $\boldsymbol{\lambda}$ whose loss on F, $\sum_{i \in F} e^{-(\mathbf{M}\boldsymbol{\lambda})_i}$, is at most m. Let $\mathbf{b}$ be the i^{th} row of $\mathbf{M}$, and let $\mathbf{A}^T$ be the matrix $\mathbf{M}_F$ without the ith row. Then Lemma 2.38 says that $\mathbf{A}\mathbf{y} = -\mathbf{b}$ for some positive vector $\mathbf{y} > \mathbf{0}$. This implies the margin of $\boldsymbol{\lambda}$ on example i is $(\mathbf{M}\boldsymbol{\lambda})_i = -\mathbf{y}^T\mathbf{A}^T\boldsymbol{\lambda}$. Since the loss of $\boldsymbol{\lambda}$ on F is at most m, each margin on F is at least $-\ln m$, and therefore $\max_{i \in F} \left(-\mathbf{A}^T\boldsymbol{\lambda}\right)_i \leq \ln m$. Hence, the margin on example i can be bounded as $(\mathbf{M}\boldsymbol{\lambda})_i = \left\langle \mathbf{y}^T, -\mathbf{A}^T\boldsymbol{\lambda} \right\rangle \leq \ln m \|\mathbf{y}\|_1$. Using Corollary 2.37, we can find $\mathbf{y}$ with bounded norm, $\|\mathbf{y}\|_1 \leq \sqrt{|F|}\|\mathbf{y}\| \leq \sqrt{|F|}(1 + k \cdot k!)$, where $k = \text{rank}(\mathbf{A}) \leq \text{rank}(\mathbf{M}_F) \leq |F|$. The proof follows. $\square$

Chapter 3

A Theory of Multiclass Boosting

3.1 Introduction

Boosting [46] refers to a general technique of combining rules of thumb, or weak classifiers, to form highly accurate combined classifiers. Minimal demands are placed on the weak classifiers, so that a variety of learning algorithms, also called weak-learners, can be employed to discover these simple rules, making the algorithm widely applicable. The theory of boosting is well-developed for the case of binary classification. In particular, the exact requirements on the weak classifiers in this setting are known: any algorithm that predicts better than random on any distribution over the training set is said to satisfy the weak learning assumption. Further, boosting algorithms that minimize loss as efficiently as possible have been designed. Specifically, it is known that the Boost-by-majority [17] algorithm is optimal in a certain sense, and that AdaBoost [23] is a practical approximation.

Such an understanding would be desirable in the multiclass setting as well, since many natural classification problems involve more than two labels, e.g. recognizing a digit from its image, natural language processing tasks such as part-of-speech tagging, and object recognition in vision. However, for such multiclass problems, a complete theoretical understanding of boosting is lacking. In particular, we do not know the "correct" way to define the requirements on the weak classifiers, nor has the notion of optimal boosting been explored in the multiclass setting.

Straightforward extensions of the binary weak-learning condition to multiclass do not work. Requiring less error than random guessing on every distribution, as in the binary case, turns out to be too weak for boosting to be possible when there are more than two labels. On the other hand, requiring more than 50% accuracy even when the number of labels is much larger than two is too stringent, and simple weak classifiers like decision stumps fail to meet this criterion, even though they often can be combined to produce highly accurate classifiers [21]. The most common approaches so far have relied on reductions to binary classification [3], but it is hardly clear that the weak-learning conditions implicitly assumed by such reductions are the most appropriate.

The purpose of a weak-learning condition is to clarify the goal of the weak-learner, thus aiding in its design, while providing a specific minimal guarantee on performance that can be exploited by a boosting algorithm. These considerations may significantly impact learning and generalization because knowing the correct weak-learning conditions might allow the use of simpler weak classifiers, which in turn can help prevent overfitting. Furthermore, boosting algorithms that more efficiently and effectively minimize training error may prevent underfitting, which can also be important.

In this chapter, we create a broad and general framework for studying multiclass boosting that formalizes the interaction between the boosting algorithm and the weak-learner. Unlike much, but not all, of the previous work on multiclass boosting, we focus specifically on the most natural, and perhaps weakest, case in which the weak classifiers are genuine classifiers in the sense of predicting a single multiclass label for each instance. Our new framework allows us to express a range of weak-learning conditions, both new ones and most of the ones that had previously been assumed (often only implicitly). Within this formalism, we can also now finally make precise what is meant by *correct* weak-learning conditions that are neither too weak nor too strong.

We focus particularly on a family of novel weak-learning conditions that have an especially appealing form: like the binary conditions, they require performance that is only slightly better than random guessing, though with respect to performance measures that are more general than ordinary classification error. We introduce a whole family of such conditions since there are many ways of randomly guessing on more than two labels, a key difference between the binary and multiclass settings. Although these conditions impose seemingly mild demands on the weak-learner, we show that each one of them is powerful enough to guarantee boostability, meaning that some combination of the weak classifiers has high accuracy. And while no individual member of the family is necessary for boostability, we also show that the entire family taken together is necessary in the sense that for every boostable learning problem, there exists one member of the family that is satisfied. Thus, we have identified a family of conditions which, as a whole, is necessary and sufficient for multiclass boosting. Moreover, we can combine the entire family into a single weak-learning condition that is necessary and sufficient by taking a kind of union, or logical OR, of all the members. This combined condition can also be expressed in our framework.

With this understanding, we are able to characterize previously studied weak-learning conditions. In particular, the condition implicitly used by AdaBoost.MH [47], which is based on a one-against-all reduction to binary, turns out to be strictly stronger than necessary for boostability. This also applies to AdaBoost.M1 [21], the most direct generalization of AdaBoost to multiclass, whose conditions can be shown to be equivalent to those of AdaBoost.MH in our setting. On the other hand, the condition implicit to the SAMME algorithm by Zhu et al [57] is too weak in the sense that even when the condition is satisfied, no boosting algorithm can guarantee to drive down the training error. Finally, the condition implicit to AdaBoost.MR [47, 21] (also called AdaBoost.M2) turns out to be exactly necessary and sufficient for boostability.

Employing proper weak-learning conditions is important, but we also need boosting algorithms that can exploit these conditions to effectively drive down error. For a given weak-learning condition, the boosting algorithm that drives down training error most efficiently in our framework can be understood as the optimal strategy for playing a certain two-player game. These games are non-trivial to analyze. However, using the powerful machinery of drifting games [20, 44], we are able to compute the optimal strategy for the games arising out of each weak-learning condition in the family described above. Compared to earlier work, our optimality results hold more generally and also achieve tighter bounds. These optimal strategies have a natural interpretation in terms of random walks, a phenomenon that has been observed in other settings [2, 17].

We also analyze the optimal boosting strategy when using the minimal weak learning condition, and this poses additional challenges. Firstly, the minimal weak learning condition has multiple natural formulations — e.g., as the union of all the conditions in the family described above, or the formulation used in AdaBoost.MR — and each formulation leading to a different game specification. A priori, it is not clear which game would lead to the best strategy. We resolve this dilemma by proving that the optimal strategies arising out of different formulations of the same weak learning condition lead to algorithms that are essentially equally good, and therefore we are free to choose whichever formulation leads to an easier analysis without fear of suffering in performance. We choose the union of conditions formulation, since it leads to strategies that share the same interpretation in terms of random walks as before. However, even with this choice, the resulting games are hard to analyze, and although we can explicitly compute the optimum strategies in general, the computational complexity is usually exponential. Nevertheless, we identify key situations under which efficient computation is possible.

The game-theoretic strategies are non-adaptive in that they presume prior knowledge about the *edge*, that is, how much better than random are the weak classifiers. Algorithms that are adaptive, such as AdaBoost, are much more practical because they do not require such prior information. We show therefore how to derive an adaptive boosting algorithm by modifying the game-theoretic strategy based on the minimal condition. This algorithm enjoys a number of theoretical guarantees. Unlike some of the non-adaptive strategies, it is efficiently computable, and since it is based on the minimal weak learning condition, it makes minimal assumptions. In fact, whenever presented with a boostable learning problem, this algorithm can approach zero training error at an exponential rate. More importantly, the algorithm is effective even beyond the boostability framework. In particular, we show empirical consistency, i.e., the algorithm always converges to the minimum of a certain exponential loss over the training data, whether or not the dataset is boostable. Furthermore, using the results in [35] we can show that this convergence occurs rapidly.

Our focus in this chapter is only on minimizing training error, which, for the algorithms we derive, provably decreases exponentially fast with the number of rounds of boosting under boostability assumptions. Such results can be used in turn to derive bounds on the generalization error using standard techniques that have been applied to other boosting algorithms [49, 23, 27]. Consistency in the multiclass classification setting has been studied by Tewari and Bartlett [53] and has been shown to be trickier than binary classification consistency. Nonetheless, by following the approach in [4] for showing consistency in the binary setting, we are able to extend the empirical consistency guarantees to general consistency guarantees in the multiclass setting: we show that under certain conditions and with sufficient data, our adaptive algorithm approaches the Bayes-optimum error on the *test* dataset.

We present experiments aimed at testing the efficacy of the adaptive algorithm when working with a very weak weak-learner to check that the conditions we have identified are indeed weaker than others that had previously been used. We find that our new adaptive strategy achieves low test error compared to other multiclass boosting algorithms which usually heavily underfit. This validates the potential practical benefit of a better theoretical understanding of multiclass boosting.

Previous work. The first boosting algorithms were given by Schapire [42] and Freund [17], followed by their AdaBoost algorithm [23]. Multiclass boosting techniques include AdaBoost.M1 and AdaBoost.M2 [23], as well as AdaBoost.MH and AdaBoost.MR [47]. Other approaches include the work by Eibl and Pfeiffer [15], Zhu et al [57]. There are also more general approaches that can be applied to boosting including [3, 6, 14, 26]. Two game-theoretic perspectives have been applied to boosting. The first one [22, 37] views the weak-learning condition as a minimax game, while drifting games [44, 17] were designed to analyze the most efficient boosting algorithms. These games have been further analyzed in the multiclass and continuous time setting in [20].

3.2 Framework

We introduce some notation. Unless otherwise stated, matrices will be denoted by bold capital letters like $\mathbf{M}$, and vectors by bold small letters like $\mathbf{v}$. Entries of a matrix and vector will be denoted as $M(i,j)$ or $v(i)$, while $\mathbf{M}(i)$ will denote the ith row of a matrix. Inner product of two vectors $\mathbf{u}, \mathbf{v}$ is denoted by $\langle \mathbf{u}, \mathbf{v} \rangle$. The Frobenius inner product of two matrices $\mathrm{Tr}(\mathbf{M}\mathbf{M}')$ will be denoted by $\mathbf{M} \bullet \mathbf{M}'$, where $\mathbf{M}'$ is the transpose of $\mathbf{M}$. The indicator function is denoted by $\mathbf{1}[\cdot]$. The set of all distributions over the set $\{1, \ldots, k\}$ will be denoted by $\Delta\{1, \ldots, k\}$, and in general, the set of all distributions over any set S will be denoted by $\Delta(S)$.

In multiclass classification, we want to predict the labels of examples lying in some set X. We are provided a training set of labeled examples $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, where each example $x_i \in X$ has a label y_i in the set $\{1, \ldots, k\}$.

Boosting combines several mildly powerful predictors, called *weak classifiers*, to form a highly accurate combined classifier, and has been previously applied for multiclass classification. In this chapter, we only allow weak classifier that predict a single class for each example. This is appealing, since the combined classifier has the same form, although it differs from what has been used in much previous work. Later we will expand our framework to include *multilabel* weak classifiers, that may predict multiple labels per example.

We adopt a game-theoretic view of boosting. A game is played between two players, Booster and Weak-Learner, for a fixed number of rounds T. With binary labels, Booster outputs a distribution in each round, and Weak-Learner returns a weak classifier achieving more than 50% accuracy on that distribution. The multiclass game is an extension of the binary game. In particular, in each round t:

- Booster creates a cost-matrix $\mathbf{C}_t \in \mathbb{R}^{m\times k}$, specifying to Weak-Learner that the cost of classifying example x_i as l is $C_t(i,l)$. The cost-matrix may not be arbitrary, but should conform to certain restrictions as discussed below.
- Weak-Learner returns some weak classifier $h_t\colon X \to \{1,\dots,k\}$ from a fixed space $h_t \in \mathcal{H}$ so that the cost incurred is

$$\mathbf{C}_t \bullet \mathbf{1}_{h_t} = \sum_{i=1}^{m} C_t(i, h_t(x_i)),$$

  is "small enough", according to some conditions discussed below. Here by $\mathbf{1}_h$ we mean the $m \times k$ matrix whose (i,j)-th entry is $\mathbf{1}\left[h(i) = j\right]$.
- Booster computes a weight α_t for the current weak classifier based on how much cost was incurred in this round.

At the end, Booster predicts according to the weighted plurality vote of the classifiers returned in each round:

$$H(x) \triangleq \operatorname*{argmax}_{l\in\{1,\dots,k\}} f_T(x,l), \text{ where } f_T(x,l) \triangleq \sum_{t=1}^{T} \mathbf{1}\left[h_t(x) = l\right]\alpha_t. \tag{3.1}$$

By carefully choosing the cost matrices in each round, Booster aims to minimize the training error of the final classifier H, even when Weak-Learner is adversarial. The restrictions on cost-matrices created by Booster, and the maximum cost Weak-Learner can suffer in each round, together define the *weak-learning condition* being used. For binary labels, the traditional weak-learning condition states: for any non-negative weights $w(1),\dots,w(m)$ on the training set, the error of the weak classifier returned is at most $(1/2 - \gamma/2)\sum_i w_i$. Here γ parametrizes the condition. There are many ways to translate this condition into our language. The one with fewest restrictions on the cost-matrices requires labeling correctly should be less costly than labeling incorrectly:

$$\forall i : C(i,y_i) \le C(i,\bar{y}_i) \text{ (here } \bar{y}_i \ne y_i \text{ is the other binary label)},$$

while the restriction on the returned weak classifier h requires less cost than predicting randomly:

$$\sum_i C(i,h(x_i)) \le \sum_i \left\{\left(\frac{1}{2} - \frac{\gamma}{2}\right) C(i,\bar{y}_i) + \left(\frac{1}{2} + \frac{\gamma}{2}\right) C(i,y_i)\right\}.$$

By the correspondence $w(i) = C(i,\bar{y}_i) - C(i,y_i)$, we may verify the two conditions are the same.

We will rewrite this condition after making some simplifying assumptions. Henceforth, without loss of generality, we assume that the true label is always 1. Let $\mathcal{C}^{\mathrm{bin}} \subseteq \mathbb{R}^{m\times 2}$ consist of matrices $\mathbf{C}$ which satisfy $C(i,1) \leq C(i,2)$. Further, let $\mathbf{U}_\gamma^{\mathrm{bin}} \in \mathbb{R}^{m\times 2}$ be the matrix whose each row is $(1/2+\gamma/2, 1/2-\gamma/2)$. Then, WeakLearner searching space $\mathcal{H}$ satisfies the binary weak-learning condition if: $\forall \mathbf{C} \in \mathcal{C}^{\mathrm{bin}}, \exists h \in \mathcal{H} : \mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{U}_\gamma^{\mathrm{bin}}\right) \leq \mathbf{0}$. There are two main benefits to this reformulation. With linear homogeneous constraints, the mathematics is simplified, as will be apparent later. More importantly, by varying the restrictions $\mathcal{C}^{\mathrm{bin}}$ on the cost vectors and the matrix $\mathbf{U}^{\mathrm{bin}}$, we can generate a vast variety of weak-learning conditions for the multiclass setting $k \geq 2$ as we now show.

Let $\mathcal{C} \subseteq \mathbb{R}^{m\times k}$ and let $\mathbf{B} \in \mathbb{R}^{m\times k}$ be a matrix which we call the *baseline*. We say a weak classifier space $\mathcal{H}$ satisfies the condition $(\mathcal{C}, \mathbf{B})$ if

$$\forall \mathbf{C} \in \mathcal{C}, \exists h \in \mathcal{H} : \quad \mathbf{C} \bullet (\mathbf{1}_h - \mathbf{B}) \leq \mathbf{0}, \quad \text{i.e.,} \sum_{i=1}^m C(i, h(i)) \leq \sum_{i=1}^m \langle \mathbf{C}(i), \mathbf{B}(i) \rangle \tag{3.2}$$

In (3.2), the variable matrix $\mathbf{C}$ specifies how costly each misclassification is, while the baseline $\mathbf{B}$ specifies a weight for each misclassification. The condition therefore states that a weak classifier should not exceed the average cost when weighted according to baseline $\mathbf{B}$. This large class of weak-learning conditions captures many previously used conditions, such as the ones used by AdaBoost.M1 [21], AdaBoost.MH [47] and AdaBoost.MR [21, 47] (see below), as well as novel conditions introduced in the next section.

By studying this vast class of weak-learning conditions, we hope to find the one that will serve the main purpose of the boosting game: finding a convex combination of weak classifiers that has zero training error. For this to be possible, at the minimum the weak classifiers should be sufficiently rich for such a perfect combination to exist. Formally, a collection $\mathcal{H}$ of weak classifiers is *boostable* if it is eligible for boosting in the sense that there exists a distribution $\boldsymbol{\lambda}$ on this space that linearly separates the data: $\forall i : \mathrm{argmax}_{l\in\{1,\ldots,k\}} \sum_{h\in\mathcal{H}} \lambda(h) \mathbf{1}\left[h(x_i) = l\right] = y_i$. The weak-learning condition plays two roles. It rejects spaces that are not boostable, and provides an algorithmic means of searching for the right combination. Ideally, the second factor will not cause the weak-learning condition to impose additional restrictions on the weak classifiers; in that case, the weak-learning condition is merely a reformulation of being boostable that is more appropriate for deriving an algorithm. In general, it could be *too strong*, i.e. certain boostable spaces will fail to satisfy the conditions. Or it could be *too weak* i.e., non-boostable spaces might satisfy such a condition. Booster strategies relying on either of these conditions will fail to drive down error, the former due to underfitting, and the latter due to overfitting. Later we will describe conditions captured by our framework that avoid being too weak or too strong. But before that, we show in the next section how our flexible framework captures weak learning conditions that have appeared previously in the literature.

3.3 Old conditions

In this section, we rewrite, in the language of our framework, the weak learning conditions explicitly or implicitly employed in the multiclass boosting algorithms SAMME [57], AdaBoost.M1 [21], and AdaBoost.MH and AdaBoost.MR [47]. This will be useful later on for comparing the strengths and weaknesses of the various conditions. We will end this section with a curious equivalence between the conditions of AdaBoost.MH and AdaBoost.M1.

Recall that we have assumed the correct label is 1 for every example. Nevertheless, we continue to use y_i to denote the correct label in this section.

3.3.1 Old conditions in the new framework

Here we restate, in the language of our new framework, the weak learning conditions of four algorithms that have earlier appeared in the literature.

SAMME. The SAMME algorithm [57] requires less error than random guessing on any distribution on the examples. Formally, a space $\mathcal{H}$ satisfies the condition if there is a $\gamma' > 0$ such that,

$$\forall d(1), \ldots, d(m) \geq 0, \exists h \in \mathcal{H} : \sum_{i=1}^{m} d(i) \mathbf{1}\left[h(x_i) \neq y_i\right] \leq (1 - 1/k - \gamma') \sum_{i=1}^{m} d(i). \quad (3.3)$$

Define a cost matrix $\mathbf{C}$ whose entries are given by

$$C(i,j) = \begin{cases} d(i) & \text{if } j \neq y_i, \\ 0 & \text{if } j = y_i. \end{cases}$$

Then the left hand side of (3.3) can be written as

$$\sum_{i=1}^{m} C(i, h(x_i)) = \mathbf{C} \bullet \mathbf{1}_h.$$

Next let $\gamma = (1 - 1/k)\gamma'$ and define baseline $\mathbf{U}_\gamma$ to be the multiclass extension of $\mathbf{U}^{\text{bin}}$,

$$U_\gamma(i,l) = \begin{cases} \frac{(1-\gamma)}{k} + \gamma & \text{if } l = y_i, \\ \frac{(1-\gamma)}{k} & \text{if } l \neq y_i. \end{cases}$$

Then the right hand side of (3.3) can be written as

$$\sum_{i=1}^{m} \sum_{l \neq y_i} C(i,l) U_\gamma(i,l) = \mathbf{C} \bullet \mathbf{U}_\gamma,$$

since $C(i, y_i) = 0$ for every example i. Define $\mathcal{C}^{\mathrm{SAM}}$ to be the following collection of cost matrices:

$$\mathcal{C}^{\mathrm{SAM}} \triangleq \left\{ \mathbf{C} : C(i,l) = \begin{cases} 0 & \text{if } l = y_i, \\ t_i & \text{if } l \neq y_i, \end{cases} \text{for non-negative } t_1, \ldots, t_m. \right\}$$

Using the last two equations, (3.3) is equivalent to

$$\forall \mathbf{C} \in \mathcal{C}^{\mathrm{SAM}}, \exists h \in \mathcal{H} : \mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{U}_\gamma \right) \leq 0.$$

Therefore, the weak-learning condition of SAMME is given by $(\mathcal{C}^{\mathrm{SAM}}, \mathbf{U}_\gamma)$.

AdaBoost.M1 AdaBoost.M1 [23] measures the performance of weak classifiers using ordinary error. It requires $1/2 + \gamma/2$ accuracy with respect to any non-negative weights $d(1), \ldots, d(m)$ on the training set:

$$\begin{aligned} \sum_{i=1}^{m} d(i) \mathbf{1}\left[h(x_i) \neq y_i \right] \quad &\leq \quad (1/2 - \gamma/2) \sum_{i=1}^{m} d(i), \qquad (3.4) \\ \text{i.e. } \sum_{i=1}^{m} d(i) [\![h(x_i) \neq y_i]\!] \quad &\leq \quad -\gamma \sum_{i=1}^{m} d(i). \end{aligned}$$

where $[\![\cdot]\!]$ is the ± 1 indicator function, taking value $+1$ when its argument is true, and -1 when false. Using the transformation

$$C(i,l) = [\![l \neq y_i]\!] d(i) \qquad (3.5)$$

we may rewrite (3.5) as

$$\begin{aligned} &\forall C \in \mathbb{R}^{m \times k} \text{ satisfying } 0 \leq -C(i, y_i) = C(i,l) \text{ for } l \neq y_i \qquad (3.6) \\ &\exists h \in \mathcal{H} : \sum_{i=1}^{m} C(i, h(x_i)) \leq \gamma \sum_{i=1}^{m} C(i, y_i) \\ &\text{i.e.} \quad \forall \mathbf{C} \in \mathcal{C}^{\mathrm{M1}}, \exists h \in \mathcal{H} : \mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{B}_\gamma^{\mathrm{M1}} \right) \leq 0, \qquad (3.7) \end{aligned}$$

where $\mathbf{B}_\gamma^{\mathrm{M1}}(i,l) = \gamma \mathbf{1}\left[l = y_i \right]$, and $\mathcal{C}^{\mathrm{M1}} \subseteq \mathbb{R}^{m \times k}$ consists of matrices satisfying the constraints in (3.6).

AdaBoost.MH AdaBoost.MH [47] is a popular multiclass boosting algorithm that is based on the one-against-all reduction, and was originally designed to use weak-hypotheses that return a prediction for every example and every label. The implicit weak learning condition requires that for any matrix with non-negative entries $d(i,l)$, the weak-hypothesis should achieve $1/2 + \gamma$ accuracy

$$\sum_{i=1}^{m} \left\{ \mathbf{1}\left[h(x_i) \neq y_i\right] d(i, y_i) + \sum_{l \neq y_i} \mathbf{1}\left[h(x_i) = l\right] d(i, l) \right\} \;\leq\; \left(\frac{1}{2} - \frac{\gamma}{2}\right) \sum_{i=1}^{m} \sum_{l=1}^{k} d(i, l). \tag{3.8}$$

This can be rewritten as

$$\begin{aligned} &\sum_{i=1}^{m} \left\{ -\mathbf{1}\left[h(x_i) = y_i\right] d(i, y_i) + \sum_{l \neq y_i} \mathbf{1}\left[h(x_i) = l\right] d(i, l) \right\} \\ \leq \;& \sum_{i=1}^{m} \left\{ \left(\frac{1}{2} - \frac{\gamma}{2}\right) \sum_{l \neq y_i} d(i, l) - \left(\frac{1}{2} + \frac{\gamma}{2}\right) d(i, y_i) \right\}. \end{aligned}$$

Using the mapping

$$C(i, l) = \begin{cases} d(i, l) & \text{if } l \neq y_i \\ -d(i, l) & \text{if } l = y_i, \end{cases}$$

their weak-learning condition may be rewritten as follows

$$\forall \mathbf{C} \in \mathbb{R}^{m \times k} \text{ satisfying } C(i, y_i) \leq 0, C(i, l) \geq 0 \text{ for } l \neq y_i, \tag{3.9}$$

$$\begin{aligned} &\exists h \in \mathcal{H} : \\ &\sum_{i=1}^{m} C(i, h(x_i)) \leq \sum_{i=1}^{m} \left\{ \left(\frac{1}{2} + \frac{\gamma}{2}\right) C(i, y_i) + \left(\frac{1}{2} - \frac{\gamma}{2}\right) \sum_{l \neq y_i} C(i, l) \right\}. \end{aligned} \tag{3.10}$$

Defining $\mathcal{C}^{\mathrm{MH}}$ to be the space of all cost matrices satisfying the constraints in (3.9), the above condition is the same as

$$\forall \mathbf{C} \in \mathcal{C}^{\mathrm{MH}}, \exists h \in \mathcal{H} : \mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{B}_{\gamma}^{\mathrm{MH}}\right) \leq 0,$$

where $\mathbf{B}_{\gamma}^{\mathrm{MH}}(i, l) = (1/2 + \gamma [\![l = y_i]\!] / 2)$.

AdaBoost.MR AdaBoost.MR [47] is based on the all-pairs multiclass to binary reduction. Like AdaBoost.MH, it was originally designed to use weak-hypotheses that return a prediction for every example and every label. The weak learning condition for AdaBoost.MR requires that for any non-negative cost-vectors $\{d(i, l)\}_{l \neq y_i}$, the weak-hypothesis returned should satisfy the following:

$$\begin{aligned} \sum_{i=1}^{m} \sum_{l \neq y_i} \left(\mathbf{1}\left[h(x_i) = l\right] - \mathbf{1}\left[h(x_i) = y_i\right]\right) d(i, l) \;&\leq\; -\gamma \sum_{i=1}^{m} \sum_{l \neq y_i} d(i, l) \\ \text{i.e. } \sum_{i=1}^{m} \left\{ -\mathbf{1}\left[h(x_i) = y_i\right] \sum_{l \neq y_i} d(i, l) + \sum_{l \neq y_i} \mathbf{1}\left[h(x_i) = l\right] d(i, l) \right\} \;&\leq\; -\gamma \sum_{i=1}^{m} \sum_{l \neq y_i} d(i, l). \end{aligned}$$

Substituting

$$C(i,l) = \begin{cases} d(i,l) & l \neq y_i \\ -\sum_{l \neq y_i} d(i,l) & l = y_i, \end{cases}$$

we may rewrite AdaBoost.MR's weak-learning condition as

$$\forall \mathbf{C} \in \mathbb{R}^{m \times k} \text{ satisfying } C(i,l) \geq 0 \text{ for } l \neq y_i, C(i,y_i) = -\sum_{l \neq y_i} C(i,l), \quad (3.11)$$

$$\exists h \in \mathcal{H} : \sum_{i=1}^{m} C(i, h(x_i)) \leq -\frac{\gamma}{2} \sum_{i=1}^{m} \left\{ -C(i,y_i) + \sum_{l \neq y_i} C(i,l) \right\}.$$

Defining $\mathcal{C}^{\mathrm{MR}}$ to be the collection of cost matrices satisfying the constraints in (3.11), the above condition is the same as

$$\forall \mathbf{C} \in \mathcal{C}^{\mathrm{MR}}, \exists h \in \mathcal{H} : \mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{B}_\gamma^{\mathrm{MR}} \right) \leq 0,$$

where $\mathbf{B}_\gamma^{\mathrm{MR}}(i,l) = [\![l = y_i]\!] \gamma/2$.

3.3.2 A curious equivalence

We show that the weak learning conditions of AdaBoost.MH and AdaBoost.M1 are identical in our framework. This is surprising because the original motivations behind these algorithms were completely different. AdaBoost.M1 is a direct extension of binary AdaBoost to the multiclass setting, whereas AdaBoost.MH is based on the one-against-all multiclass to binary reduction. This equivalence is a sort of degeneracy, and arises because the weak classifiers being used predict single labels per example. With multilabel weak classifiers, for which AdaBoost.MH was originally designed, the equivalence no longer holds.

The proofs in this and later sections will make use of the following minimax result, that is a weaker version of Corollary 37.3.2 of [40].

Theorem 3.1. *(Minimax Theorem) Let C, D be non-empty closed convex subsets of $\mathbb{R}^m, \mathbb{R}^n$ respectively, and let K be a linear function on $C \times D$. If either C or D is bounded, then*

$$\min_{v \in D} \max_{u \in C} K(u,v) = \max_{u \in C} \min_{v \in D} K(u,v).$$

Lemma 3.2. *A weak classifier space $\mathcal{H}$ satisfies $(\mathcal{C}^{M1}, \mathbf{B}_\gamma^{M1})$ if and only if it satisfies $(\mathcal{C}^{MH}, \mathbf{B}_\gamma^{MH})$.*

Proof. We will refer to $(\mathcal{C}^{\mathrm{M1}}, \mathbf{B}_\gamma^{\mathrm{M1}})$ by M1 and $(\mathcal{C}^{\mathrm{MH}}, \mathbf{B}_\gamma^{\mathrm{MH}})$ by MH for brevity. The proof is in three steps.

Step (i): If $\mathcal{H}$ satisfies MH, then it also satisfies M1. This follows since any constraint (3.4) imposed by M1 on $\mathcal{H}$ can be reproduced by MH by plugging the following values of $d(i,l)$ in (3.8)

$$d(i,l) = \begin{cases} d(i) & \text{if } l = y_i \\ 0 & \text{if } l \neq y_i. \end{cases}$$

Step (ii): If $\mathcal{H}$ satisfies M1, then there is a convex combination $\mathbf{H}_{\boldsymbol{\lambda}^*}$ of the matrices $\mathbf{1}_h \in \mathcal{H}$, defined as

$$\mathbf{H}_{\boldsymbol{\lambda}^*} \triangleq \sum_{h \in \mathcal{H}} \lambda^*(h) \mathbf{1}_h,$$

such that

$$\forall i : \left(\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B}_\gamma^{\text{MH}}\right)(i,l) \begin{cases} \geq 0 & \text{if } l = y_i \\ \leq 0 & \text{if } l \neq y_i. \end{cases} \tag{3.12}$$

Indeed, Theorem 3.1 yields

$$\min_{\boldsymbol{\lambda} \in \Delta(\mathcal{H})} \max_{\mathbf{C} \in \mathcal{C}^{\text{M1}}} \mathbf{C} \bullet \left(\mathbf{H}_{\boldsymbol{\lambda}} - \mathbf{B}_\gamma^{\text{M1}}\right) = \max_{\mathbf{C} \in \mathcal{C}^{\text{M1}}} \min_{h \in \mathcal{H}} \mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{B}_\gamma^{\text{M1}}\right) \leq 0, \tag{3.13}$$

where the inequality is a restatement of our assumption that $\mathcal{H}$ satisfies M1. If $\boldsymbol{\lambda}^*$ is a minimizer of the minimax expression, then $\mathbf{H}_{\boldsymbol{\lambda}^*}$ must satisfy

$$\forall i : \mathbf{H}_{\boldsymbol{\lambda}^*}(i,l) \begin{cases} \geq \frac{1}{2} + \frac{\gamma}{2} & \text{if } l = y_i \\ \leq \frac{1}{2} - \frac{\gamma}{2} & \text{if } l \neq y_i, \end{cases} \tag{3.14}$$

or else some choice of $\mathbf{C} \in \mathcal{C}^{\text{M1}}$ can cause $\mathbf{C} \bullet \left(\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B}^{\text{M1}}\right)$ to exceed 0. In particular, if $\mathbf{H}_{\boldsymbol{\lambda}^*}(i_0, l) < 1/2 + \gamma/2$, then

$$\left(\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B}_\gamma^{\text{M1}}\right)(i_0, y_{i_0}) < \sum_{l \neq y_{i_0}} \left(\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B}_\gamma^{\text{M1}}\right)(i_0, l).$$

Now, if we choose $\mathbf{C} \in \mathcal{C}^{\text{M1}}$ as

$$C(i,l) = \begin{cases} 0 & \text{if } i \neq i_0 \\ 1 & \text{if } i = i_0, l \neq y_{i_0} \\ -1 & \text{if } i = i_0, l = y_{i_0}, \end{cases}$$

then,

$$\mathbf{C} \bullet \left(\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B}_\gamma^{\text{M1}}\right) = -\left(\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B}_\gamma^{\text{M1}}\right)(i_0, y_{i_0}) + \sum_{l \neq y_{i_0}} \left(\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B}_\gamma^{\text{M1}}\right)(i_0, l) > 0,$$

contradicting the inequality in (3.13). Therefore (3.14) holds. Eqn. (3.12), and thus Step (ii), now follows by observing that $\mathbf{B}_\gamma^{\mathrm{MH}}$, by definition, satisfies

$$\forall i : \mathbf{B}_\gamma^{\mathrm{MH}}(i,l) = \begin{cases} \frac{1}{2} + \frac{\gamma}{2} & \text{if } l = y_i \\ \frac{1}{2} - \frac{\gamma}{2} & \text{if } l \neq y_i. \end{cases}$$

Step (iii) If there is some convex combination $\mathcal{H}_{\boldsymbol{\lambda}^*}$ satisfying (3.12), then $\mathcal{H}$ satisfies MH. Recall that $\mathbf{B}^{\mathrm{MH}}$ consists of entries that are non-positive on the correct labels and non-negative for incorrect labels. Therefore, (3.12) implies

$$0 \geq \max_{\mathbf{C} \in \mathcal{C}^{\mathrm{MH}}} \mathbf{C} \bullet \left(\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B}_\gamma^{\mathrm{MH}}\right) \geq \min_{\boldsymbol{\lambda} \in \Delta(\mathcal{H})} \max_{\mathbf{C} \in \mathcal{C}^{\mathrm{MH}}} \mathbf{C} \bullet \left(\mathbf{H}_{\boldsymbol{\lambda}} - \mathbf{B}_\gamma^{\mathrm{MH}}\right).$$

On the other hand, using Theorem 3.1 we have

$$\min_{\boldsymbol{\lambda} \in \Delta(\mathcal{H})} \max_{\mathbf{C} \in \mathcal{C}^{\mathrm{MH}}} \mathbf{C} \bullet \left(\mathbf{H}_{\boldsymbol{\lambda}} - \mathbf{B}_\gamma^{\mathrm{MH}}\right) = \max_{\mathbf{C} \in \mathcal{C}^{\mathrm{MH}}} \min_{h \in \mathcal{H}} \mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{B}_\gamma^{\mathrm{MH}}\right).$$

Combining the two, we get

$$0 \geq \max_{\mathbf{C} \in \mathcal{C}^{\mathrm{MH}}} \min_{h \in \mathcal{H}} \mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{B}_\gamma^{\mathrm{MH}}\right),$$

which is the same as saying that $\mathcal{H}$ satisfies MH's condition.

Steps (ii) and (iii) together imply that if $\mathcal{H}$ satisfies M1, then it also satisfies MH. Along with Step (i), this concludes the proof. □

3.4 Necessary and sufficient weak-learning conditions

The binary weak-learning condition has an appealing form: for any distribution over the examples, the weak classifier needs to achieve error not greater than that of a random player who guesses the correct answer with probability $1/2 + \gamma/2$. Further, this is the weakest condition under which boosting is possible as follows from a game-theoretic perspective [22, 37] . Multiclass weak-learning conditions with similar properties are missing in the literature. In this section we show how our framework captures such conditions.

3.4.1 Edge-over-random conditions

In the multiclass setting, we model a random player as a baseline predictor $\mathbf{B} \in \mathbb{R}^{m \times k}$ whose rows are distributions over the labels, $\mathbf{B}(i) \in \Delta\{1, \ldots, k\}$. The prediction on example i is a sample from $\mathbf{B}(i)$. We only consider the space of *edge-over-random* baselines $\mathcal{B}_\gamma^{\mathrm{eor}} \subseteq \mathbb{R}^{m \times k}$ who have a faint clue about the correct answer. More precisely, any baseline $\mathbf{B} \in \mathcal{B}_\gamma^{\mathrm{eor}}$ in this space is γ more likely to predict the correct label than an incorrect one on every example i: $\forall l \neq 1, B(i,1) \geq B(i,l) + \gamma$, with equality holding for some l, i.e.:

$$B(i,1) = \max\left\{B(i,l) + \gamma : l \neq 1\right\}.$$

Notice that the edge-over-random baselines are different from the baselines used by earlier weak learning conditions discussed in the previous section.

When $k = 2$, the space $\mathcal{B}_\gamma^{\text{eor}}$ consists of the unique player $\mathbf{U}_\gamma^{\text{bin}}$, and the binary weak-learning condition is given by $(\mathcal{C}^{\text{bin}}, \mathbf{U}_\gamma^{\text{bin}})$. The new conditions generalize this to $k > 2$. In particular, define $\mathcal{C}^{\text{eor}}$ to be the multiclass extension of $\mathcal{C}^{\text{bin}}$: any cost-matrix in $\mathcal{C}^{\text{eor}}$ should put the least cost on the correct label, i.e., the rows of the cost-matrices should come from the set $\left\{\mathbf{c} \in \mathbb{R}^k : \forall l, c(1) \leq c(l)\right\}$. Then, for every baseline $\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}$, we introduce the condition $(\mathcal{C}^{\text{eor}}, \mathbf{B})$, which we call an *edge-over-random* weak-learning condition. Since $\mathbf{C} \bullet \mathbf{B}$ is the expected cost of the edge-over-random baseline $\mathbf{B}$ on matrix $\mathbf{C}$, the constraints (3.2) imposed by the new condition essentially require better than random performance.

Also recall that we have assumed that the true label y_i of example i in our training set is always 1. Nevertheless, we may occasionally continue to refer to the true labels as y_i.

We now present the central results of this section. The seemingly mild edge-over-random conditions guarantee boostability, meaning weak classifiers that satisfy any one such condition can be combined to form a highly accurate combined classifier.

Theorem 3.3 (Sufficiency)**.** *If a weak classifier space $\mathcal{H}$ satisfies a weak-learning condition $(\mathcal{C}^{eor}, \mathbf{B})$, for some $\mathbf{B} \in \mathcal{B}_\gamma^{eor}$, then $\mathcal{H}$ is boostable.*

Proof. The proof is in the spirit of the ones in [22]. Applying Theorem 3.1 yields

$$0 \geq \max_{\mathbf{C} \in \mathcal{C}^{\text{eor}}} \min_{h \in \mathcal{H}} \mathbf{C} \bullet (\mathbf{1}_h - \mathbf{B}) = \min_{\boldsymbol{\lambda} \in \Delta(\mathcal{H})} \max_{\mathbf{C} \in \mathcal{C}^{\text{eor}}} \mathbf{C} \bullet (\mathbf{H}_{\boldsymbol{\lambda}} - \mathbf{B}),$$

where the first inequality follows from the definition (3.2) of the weak-learning condition. Let $\boldsymbol{\lambda}^*$ be a minimizer of the min-max expression. Unless the first entry of each row of $(\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B})$ is the largest, the right hand side of the min-max expression can be made arbitrarily large by choosing $\mathbf{C} \in \mathcal{C}^{\text{eor}}$ appropriately. For example, if in some row i, the j_0^{th} element is strictly larger than the first element, by choosing

$$C(i,j) = \begin{cases} -1 & \text{if } j = 1 \\ 1 & \text{if } j = j_0 \\ 0 & \text{otherwise,} \end{cases}$$

we get a matrix in $\mathcal{C}^{\text{eor}}$ which causes $\mathbf{C} \bullet (\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B})$ to be equal to $C(i, j_0) - C(i,1) > 0$, an impossibility by the first inequality.

Therefore, the convex combination of the weak classifiers, obtained by choosing each weak classifier with weight given by $\boldsymbol{\lambda}^*$, perfectly classifies the training data, in fact with a margin γ. □

On the other hand, the family of such conditions, taken as a whole, is necessary for boostability in the sense that every eligible space of weak classifiers satisfies some edge-over-random condition.

Theorem 3.4 (Relaxed necessity)**.** *For every boostable weak classifier space $\mathcal{H}$, there exists a $\gamma > 0$ and $\mathbf{B} \in \mathcal{B}_\gamma^{eor}$ such that $\mathcal{H}$ satisfies the weak-learning condition $(\mathcal{C}^{eor}, \mathbf{B})$.*

Proof. The proof shows existence through non-constructive averaging arguments. We will reuse notation from the proof of Theorem 3.3 above. $\mathcal{H}$ is boostable implies there exists some distribution $\boldsymbol{\lambda}^* \in \Delta(\mathcal{H})$ such that

$$\forall j \neq 1, i : \mathbf{H}_{\boldsymbol{\lambda}^*}(i,1) - \mathbf{H}_{\boldsymbol{\lambda}^*}(i,j) > 0.$$

Let $\gamma > 0$ be the minimum of the above expression over all possible (i,j), and let $\mathbf{B} = \mathbf{H}_{\boldsymbol{\lambda}^*}$. Then $\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}$, and

$$\max_{\mathbf{C}\in\mathcal{C}^{\text{eor}}} \min_{h\in\mathcal{H}} \mathbf{C} \bullet (\mathbf{1}_h - \mathbf{B}) \leq \min_{\boldsymbol{\lambda}\in\Delta(\mathcal{H})} \max_{\mathbf{C}\in\mathcal{C}^{\text{eor}}} \mathbf{C} \bullet (\mathbf{H}_{\boldsymbol{\lambda}} - \mathbf{B}) \leq \max_{\mathbf{C}\in\mathcal{C}^{\text{eor}}} \mathbf{C} \bullet (\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B}) = 0,$$

where the equality follows since by definition $\mathbf{H}_{\boldsymbol{\lambda}^*} - \mathbf{B} = \mathbf{0}$. The max-min expression is at most zero is another way of saying that $\mathcal{H}$ satisfies the weak-learning condition $(\mathcal{C}^{\text{eor}}, \mathbf{B})$ as in (3.2). □

Theorem 3.4 states that any boostable weak classifier space will satisfy some condition in our family, but it does not help us choose the right condition. Experiments in Section 3.10 suggest $\left(\mathcal{C}^{\text{eor}}, \mathbf{U}_\gamma\right)$ is effective with very simple weak-learners compared to popular boosting algorithms. (Recall $\mathbf{U}_\gamma \in \mathcal{B}_\gamma^{\text{eor}}$ is the edge-over-random baseline closest to uniform; it has weight $(1-\gamma)/k$ on incorrect labels and $(1-\gamma)/k + \gamma$ on the correct label.) However, there are theoretical examples showing each condition in our family is too strong.

Theorem 3.5. *For any $\mathbf{B} \in \mathcal{B}_\gamma^{eor}$, there exists a boostable space $\mathcal{H}$ that fails to satisfy the condition $(\mathcal{C}^{eor}, \mathbf{B})$.*

Proof. We provide, for any $\gamma > 0$ and edge-over-random baseline $\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}$, a dataset and weak classifier space that is boostable but fails to satisfy the condition $(\mathcal{C}^{\text{eor}}, \mathbf{B})$.

Pick $\gamma' = \gamma/k$ and set $m > 1/\gamma'$ so that $\lfloor m(1/2+\gamma')\rfloor > m/2$. Our dataset will have m labeled examples $\{(0,y_0),\ldots,(m-1,y_{m-1})\}$, and m weak classifiers. We want the following symmetries in our weak classifiers:

- Each weak classifier correctly classifies $\lfloor m(1/2+\gamma')\rfloor$ examples and misclassifies the rest.
- On each example, $\lfloor m(1/2+\gamma')\rfloor$ weak classifiers predict correctly.

Note the second property implies boostability, since the uniform convex combination of all the weak classifiers is a perfect predictor.

The two properties can be satisfied by the following design. A window is a contiguous sequence of examples that may wrap around; for example

$$\{i, (i+1) \mod m, \ldots, (i+k) \mod m\}$$

is a window containing k elements, which may wrap around if $i + k \geq m$. For each window of length $\lfloor m(1/2+\gamma')\rfloor$ create a hypothesis that correctly classifies within the window, and misclassifies outside. This weak-hypothesis space has size m, and has the required properties.

We still have flexibility as to how the misclassifications occur, and which cost-matrix to use, which brings us to the next two choices:

- Whenever a hypothesis misclassifies on example i, it predicts label
$$\hat{y}_i \triangleq \operatorname{argmin}\left\{B(i,l) : l \neq y_i\right\}. \tag{3.15}$$
- A cost-matrix is chosen so that the cost of predicting $\hat{y}_i$ on example i is 1, but for any other prediction the cost is zero. Observe this cost-matrix belongs to $\mathcal{C}^{\text{eor}}$.

Therefore, every time a weak classifier predicts incorrectly, it also suffers cost 1. Since each weak classifier predicts correctly only within a window of length $\lfloor m(1/2+\gamma')\rfloor$, it suffers cost $\lceil m(1/2-\gamma')\rceil$. On the other hand, by the choice of $\hat{y}_i$ in (3.15),

$$\begin{array}{rcl} B(i,\hat{y}_i) & = & \min\left\{B(i,1)-\gamma, B(i,2),\ldots,B(i,k)\right\} \\ & \leq & \frac{1}{k}\left\{B(i,1)-\gamma+B(i,2)+B(i,3)+\ldots+B(i,k)\right\} \\ & = & 1/k-\gamma/k. \end{array}$$

So the cost of $\mathbf{B}$ on the chosen cost-matrix is at most $m(1/k-\gamma/k)$, which is less than the cost $\lceil m(1/2-\gamma')\rceil \geq m(1/2-\gamma/k)$ of any weak classifier whenever the number of labels k is more than two. Hence our boostable space of weak classifiers fails to satisfy $(\mathcal{C}^{\text{eor}}, \mathbf{B})$. □

Theorems 3.4 and 3.5 can be interpreted as follows. While a boostable space will satisfy *some* edge-over-random condition, without further information about the dataset it is not possible to know *which* particular condition will be satisfied. The kind of prior knowledge required to make this guess correctly is provided by Theorem 3.3: the appropriate weak learning condition is determined by the distribution of votes on the labels for each example that a target weak classifier combination might be able to get. Even with domain expertise, such knowledge may or may not be obtainable in practice before running boosting. We therefore need conditions that assume less.

3.4.2 The minimal weak learning condition

A perhaps extreme way of weakening the condition is by requiring the performance on a cost matrix to be competitive not with a *fixed* baseline $\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}$, but with the *worst* of them:
$$\forall \mathbf{C} \in \mathcal{C}^{\text{eor}}, \exists h \in \mathcal{H} : \mathbf{C} \bullet \mathbf{1}_h \leq \max_{\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}} \mathbf{C} \bullet \mathbf{B}. \tag{3.16}$$

Condition (3.16) states that during the course of the same boosting game, Weak-Learner may choose to beat *any* edge-over-random baseline $\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}$, possibly a different one for every round and every cost-matrix. This may superficially seem much too weak. On the contrary, this condition turns out to be equivalent to boostability. In other words, according to our criterion, it is neither too weak nor too strong as a weak-learning condition. However, unlike the edge-over-random conditions, it also turns out to be more difficult to work with algorithmically.

Furthermore, this condition can be shown to be equivalent to the one used by AdaBoost.MR [47, 21]. This is perhaps remarkable since the latter is based on the apparently completely unrelated all-pairs multiclass to binary reduction. In Section 3.3 we saw that the MR condition is given by $(\mathcal{C}^{\mathrm{MR}}, \mathbf{B}_\gamma^{\mathrm{MR}})$, where $\mathcal{C}^{\mathrm{MR}}$ consists of cost-matrices that put non-negative costs on incorrect labels and whose rows sum up to zero, while $\mathbf{B}_\gamma^{\mathrm{MR}} \in \mathbb{R}^{m\times k}$ is the matrix that has γ on the first column and $-\gamma$ on all other columns. Further, the MR condition, and hence (3.16), can be shown to be neither too weak nor too strong.

Theorem 3.6 (MR)**.** *A weak classifier space $\mathcal{H}$ satisfies AdaBoost.MR's weak-learning condition $(\mathcal{C}^{MR}, \mathbf{B}_\gamma^{MR})$ if and only if it satisfies (3.16). Moreover, this condition is equivalent to being boostable.*

Proof. We will show the following three conditions are equivalent:

(A) $\mathcal{H}$ is boostable

(B) $\exists\gamma > 0$ such that $\forall \mathbf{C} \in \mathcal{C}^{\mathrm{eor}}, \exists h \in \mathcal{H} : \mathbf{C} \bullet \mathbf{1}_h \leq \max_{\mathbf{B}\in\mathcal{B}_\gamma^{\mathrm{eor}}} \mathbf{C} \bullet \mathbf{B}$

(C) $\exists\gamma > 0$ such that $\forall \mathbf{C} \in \mathcal{C}^{\mathrm{MR}}, \exists h \in \mathcal{H} : \mathbf{C} \bullet \mathbf{1}_h \leq \mathbf{C} \bullet \mathbf{B}^{\mathrm{MR}}$.

We will show (A) implies (B), (B) implies (C), and (C) implies (A) to achieve the above.

(A) implies (B): Immediate from Theorem 2.

(B) implies (C): Suppose (B) is satisfied with 2γ. We will show that this implies $\mathcal{H}$ satisfies $(\mathcal{C}^{\mathrm{MR}}, \mathbf{B}_\gamma^{\mathrm{MR}})$. Notice $\mathcal{C}^{\mathrm{MR}} \subset \mathcal{C}^{\mathrm{eor}}$. Therefore it suffices to show that

$$\forall \mathbf{C} \in \mathcal{C}^{\mathrm{MR}}, \mathbf{B} \in \mathcal{B}_{2\gamma}^{\mathrm{eor}} : \mathbf{C} \bullet \left(\mathbf{B} - \mathbf{B}_\gamma^{\mathrm{MR}}\right) \leq 0.$$

Notice that $\mathbf{B} \in \mathcal{B}_{2\gamma}^{\mathrm{eor}}$ implies $\mathbf{B}' = \mathbf{B} - \mathbf{B}_\gamma^{\mathrm{MR}}$ is a matrix whose largest entry in each row is in the first column of that row. Then, for any $\mathbf{C} \in \mathcal{C}^{\mathrm{MR}}$, $\mathbf{C} \bullet \mathbf{B}'$ can be written as

$$\mathbf{C} \bullet \mathbf{B}' = \sum_{i=1}^{m} \sum_{j=2}^{k} C(i,j) \left(B'(i,j) - B'(i,1)\right).$$

Since $C(i,j) \geq 0$ for $j > 1$, and $B'(i,j) - B'(i,1) \leq 0$, we have our result.

(C) implies (A): Applying Theorem 3.1

$$0 \geq \max_{\mathbf{C}\in\mathcal{C}^{\mathrm{MR}}} \min_{h\in\mathcal{H}} \mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{B}_\gamma^{\mathrm{MR}}\right) = \min_{\boldsymbol{\lambda}\in\Delta(\mathcal{H})} \max_{\mathbf{C}\in\mathcal{C}^{\mathrm{MR}}} \mathbf{C} \bullet \left(\mathbf{H}_{\boldsymbol{\lambda}} - \mathbf{B}_\gamma^{\mathrm{MR}}\right).$$

	h_1	h_2
a	1	2
b	1	2

Figure 3.1: A weak classifier space which satisfies SAMME's weak learning condition but is not boostable.

For any i_0 and $l_0 \neq 1$, the following cost-matrix $\mathbf{C}$ satisfies $\mathbf{C} \in \mathcal{C}^{\text{MR}}$,

$$C(i,l) = \begin{cases} 0 & \text{if } i \neq i_0 \text{ or } l \notin \{1, l_0\} \\ 1 & \text{if } i = i_0, l = l_0 \\ -1 & \text{if } i = i_0, l = 1. \end{cases}$$

Let $\boldsymbol{\lambda}$ belong to the argmin of the min max expression. Then $\mathbf{C} \bullet \left(\mathbf{H}_{\boldsymbol{\lambda}} - \mathbf{B}_{\gamma}^{\text{MR}}\right) \leq 0$ implies $\mathbf{H}_{\boldsymbol{\lambda}}(i_0, 1) - \mathbf{H}_{\boldsymbol{\lambda}}(i_0, l_0) \geq 2\gamma$. Since this is true for all i_0 and $l_0 \neq 1$, we conclude that the $(\mathcal{C}^{\text{MR}}, \mathbf{B}_{\gamma}^{\text{MR}})$ condition implies boostability.

This concludes the proof of equivalence. □

Next, we illustrate the strengths of our minimal weak-learning condition through concrete comparisons with previous algorithms.

Comparison with SAMME. The SAMME algorithm of Zhu et al [57] requires the weak classifiers to achieve less error than uniform random guessing for multiple labels; in our language, their weak-learning condition is $(\mathcal{C}^{\text{SAM}}, \mathbf{U}_{\gamma})$, as shown in Section 3.3, where $\mathcal{C}^{\text{SAM}}$ consists of cost matrices whose rows are of the form $(0, t, t, \ldots)$ for some non-negative t. As is well-known, this condition is not sufficient for boosting to be possible. In particular, consider the dataset $\{(a, 1), (b, 2)\}$ with $k = 3, m = 2$, and a weak classifier space consisting of h_1, h_2 which always predict $1, 2$, respectively (Figure 3.1). Since neither classifier distinguishes between a, b we cannot achieve perfect accuracy by combining them in any way. Yet, due to the constraints on the cost-matrix, one of h_1, h_2 will always manage non-positive cost while random always suffers positive cost. On the other hand our weak-learning condition allows the Booster to choose far richer cost matrices. In particular, when the cost matrix $\mathbf{C} \in \mathcal{C}^{\text{eor}}$ is given by

	1	2	3
a	-1	$+1$	0
b	$+1$	-1	0,

both classifiers in the above example suffer more loss than the random player $\mathbf{U}_{\gamma}$, and fail to satisfy our condition.

Comparison with AdaBoost.MH. AdaBoost.MH [47] was designed for use with weak hypotheses that on each example return a prediction for every label. When used in our framework, where the weak classifiers return only a single multiclass prediction per example, the implicit demands made by AdaBoost.MH on the weak classifier space turn out to be too strong. To demonstrate this, we construct a classifier space that satisfies the condition $(\mathcal{C}^{\text{eor}}, \mathbf{U}_\gamma)$ in our family, but cannot satisfy AdaBoost.MH's weak-learning condition. Note that this does not imply that the conditions are too strong when used with more powerful weak classifiers that return multilabel multiclass predictions.

Consider a space $\mathcal{H}$ that has, for every $(1/k+\gamma)m$ element subset of the examples, a classifier that predicts correctly on exactly those elements. The expected loss of a randomly chosen classifier from this space is the same as that of the random player $\mathbf{U}_\gamma$. Hence $\mathcal{H}$ satisfies this weak-learning condition. On the other hand, it was shown in Section 3.3 that AdaBoost.MH's weak-learning condition is the pair $(\mathcal{C}^{\text{MH}}, \mathbf{B}_\gamma^{\text{MH}})$, where $\mathcal{C}^{\text{MH}}$ consists of cost matrices with non-negative entries on incorrect labels and non-positive entries on real labels, and where each row of the matrix $\mathbf{B}_\gamma^{\text{MH}}$ is the vector $(1/2+\gamma/2, 1/2-\gamma/2, \ldots, 1/2-\gamma/2)$. A quick calculation shows that for any $h \in \mathcal{H}$, and $\mathbf{C} \in \mathcal{C}^{\text{MH}}$ with -1 in the first column and zeroes elsewhere, $\mathbf{C} \bullet \left(\mathbf{1}_h - \mathbf{B}_\gamma^{\text{MH}}\right) = 1/2 - 1/k$. This is positive when $k > 2$, so that $\mathcal{H}$ fails to satisfy AdaBoost.MH's condition.

We have seen how our framework allows us to capture the strengths and weaknesses of old conditions, describe a whole new family of conditions and also identify the condition making minimal assumptions. In the next few sections, we show how to design boosting algorithms that employ these new conditions and enjoy strong theoretical guarantees.

3.5 Algorithms

In this section we devise algorithms by analyzing the boosting games that employ weak-learning conditions in our framework. We compute the optimum Booster strategy against a completely adversarial Weak-Learner, which here is permitted to choose weak classifiers without restriction, i.e. the entire space $\mathcal{H}^{\text{all}}$ of all possible functions mapping examples to labels. By modeling Weak-Learner adversarially, we make absolutely no assumptions on the algorithm it might use. Hence, error guarantees enjoyed in this situation will be universally applicable. Our algorithms are derived from the very general drifting games framework [44] for solving boosting games, which in turn was inspired by Freund's Boost-by-majority algorithm [17], which we review next.

The OS Algorithm. Fix the number of rounds T and a weak-learning condition $(\mathcal{C}, \mathbf{B})$. We will only consider conditions that are not *vacuous*, i.e., at least some classifier space satisfies the condition, or equivalently, the space $\mathcal{H}^{\text{all}}$ satisfies $(\mathcal{C}, \mathbf{B})$. Additionally, we assume the constraints placed by $\mathcal{C}$ are on individual rows. In other words, there is some subset $\mathcal{C}_0 \subseteq \mathbb{R}^k$ of all possible rows, such that a cost matrix $\mathbf{C}$ belongs to the collection $\mathcal{C}$ if and only if each of its rows belongs to this subset:

$$\mathbf{C} \in \mathcal{C} \iff \forall i : \mathbf{C}(i) \in \mathcal{C}_0. \tag{3.17}$$

Further, we assume $\mathcal{C}_0$ forms a convex cone i.e $\mathbf{c}, \mathbf{c}' \in \mathcal{C}_0$ implies $t\mathbf{c} + t'\mathbf{c}' \in \mathcal{C}_0$ for any non-negative t, t'. This also implies that $\mathcal{C}$ is a convex cone. This is a very natural restriction, and is satisfied by the space $\mathbf{C}$ used by the weak learning conditions of AdaBoost.MH, AdaBoost.M1, AdaBoost.MR, SAMME as well as every edge-over-random condition. [1] For simplicity of presentation we fix the weights $\alpha_t = 1$ in each round. With $\mathbf{f}_T$ defined as in (3.1), whether the final hypotheses output by Booster makes a prediction error on an example i is decided by whether an incorrect label received the maximum number of votes, $f_T(i, 1) \le \max_{l=2}^{k} f_T(i, l)$. Therefore, the optimum Booster payoff can be written as

$$\min_{\mathbf{C}_1 \in \mathcal{C}} \max_{\substack{h_1 \in \mathcal{H}^{\text{all}}: \\ \mathbf{C}_1 \bullet \left(\mathbf{1}_{h_1} - \mathbf{B}\right) \le \mathbf{0}}} \cdots \min_{\mathbf{C}_T \in \mathcal{C}} \max_{\substack{h_T \in \mathcal{H}^{\text{all}}: \\ \mathbf{C}_T \bullet \left(\mathbf{1}_{h_T} - \mathbf{B}\right) \le \mathbf{0}}} \frac{1}{m} \sum_{i=1}^{m} L^{\text{err}}(f_T(x_i, 1), \ldots, f_T(x_i, k)). \tag{3.18}$$

where the function $L^{\text{err}} : \mathbb{R}^k \to \mathbb{R}$ encodes 0-1 error

$$L^{\text{err}}(\mathbf{s}) = \mathbf{1}\left[s(1) \le \max_{l>1} s(l)\right]. \tag{3.19}$$

In general, we will also consider other loss functions $L : \mathbb{R}^k \to \mathbb{R}$ such as exponential loss, hinge loss, etc. that upper-bound error and are *proper*: i.e. $L(\mathbf{s})$ is increasing in the weight of the correct label $s(1)$, and decreasing in the weights of the incorrect labels $s(l), l \ne 1$.

Directly analyzing the optimal payoff is hard. However, [44] observed that the payoffs can be very well approximated by certain potential functions. Indeed, for any $\mathbf{b} \in \mathbb{R}^k$ define the *potential function* $\phi_t^{\mathbf{b}} : \mathbb{R}^k \to \mathbb{R}$ by the following recurrence:

$$\begin{aligned} \phi_0^{\mathbf{b}} &= L \\ \phi_t^{\mathbf{b}}(\mathbf{s}) &= \min_{\mathbf{c} \in \mathcal{C}_0} \max_{\mathbf{p} \in \Delta\{1,\ldots,k\}} \ \mathbb{E}_{l \sim \mathbf{p}}\left[\phi_{t-1}^{\mathbf{b}}(\mathbf{s} + \mathbf{e}_l)\right] \\ &\qquad \text{s.t.} \quad \mathbb{E}_{l \sim \mathbf{p}}\left[c(l)\right] \le \langle \mathbf{b}, \mathbf{c} \rangle, \end{aligned} \tag{3.20}$$

where $l \sim \mathbf{p}$ denotes that label l is sampled from the distribution $\mathbf{p}$, and $\mathbf{e}_l \in \mathbb{R}^k$ is the unit-vector whose lth coordinate is 1 and the remaining coordinates zero. Notice the recurrence uses the collection of rows $\mathcal{C}_0$ instead of the collection of cost matrices $\mathcal{C}$. When there are $T - t$ rounds remaining (that is, after t rounds of boosting), these potential functions compute an estimate $\phi_{T-t}^{\mathbf{b}}(\mathbf{s}_t)$ of whether an example x will be misclassified, based on its current state $\mathbf{s}_t$ consisting of counts of votes received so far on various classes:

$$s_t(l) = \sum_{t'=1}^{t-1} \mathbf{1}\left[h_{t'}(x) = l\right]. \tag{3.21}$$

[1]All our results hold under the weaker restriction on the space $\mathcal{C}$, where the set of possible cost vectors $\mathcal{C}_0$ for a row i could depend on i. For simplicity of exposition, we stick to the more restrictive assumption that $\mathcal{C}_0$ is common across all rows.

Notice this definition of state assumes that $\alpha_t = 1$ in each round. Sometimes, we will choose the weights differently. In such cases, a more appropriate definition is the weighted state $\mathbf{f}_t \in \mathbb{R}^k$, tracking the weighted counts of votes received so far:

$$f_t(l) = \sum_{t'=1}^{t-1} \alpha_{t'} \mathbf{1}\left[h_{t'}(x) = l\right]. \tag{3.22}$$

However, unless otherwise noted, we will assume $\alpha_t = 1$, and so the definition in (3.21) will suffice.

The recurrence in (3.20) requires the max player's response $\mathbf{p}$ to satisfy the constraint that the expected cost under the distribution $\mathbf{p}$ is at most the inner-product $\langle \mathbf{c}, \mathbf{b} \rangle$. If there is no distribution satisfying this requirement, then the value of the max expression is $-\infty$. The existence of a valid distribution depends on both $\mathbf{b}$ and $\mathbf{c}$ and is captured by the following:

$$\exists \mathbf{p} \in \Delta\{1, \ldots, k\} : \mathbb{E}_{l \sim \mathbf{p}}\left[c(l)\right] \le \langle \mathbf{c}, \mathbf{b} \rangle \iff \min_l c(l) \le \langle \mathbf{b}, \mathbf{c} \rangle. \tag{3.23}$$

In this chapter, the vector $\mathbf{b}$ will always correspond to some row $\mathbf{B}(i)$ of the baseline used in the weak learning condition. In such a situation, the next lemma shows that a distribution satisfying the required constraints will always exist.

Lemma 3.7. *If $\mathcal{C}_0$ is a cone and (3.17) holds, then for any row $\mathbf{b} = \mathbf{B}(i)$ of the baseline and any cost vector $\mathbf{c} \in \mathcal{C}_0$, (3.23) holds unless the condition $(\mathcal{C}, \mathbf{B})$ is vacuous.*

Proof. We show that if (3.23) does not hold, then the condition is vacuous. Assume that for row $\mathbf{b} = \mathbf{B}(i_0)$ of the baseline, and some choice of cost vector $\mathbf{c} \in \mathcal{C}_0$, (3.23) does not hold. We pick a cost-matrix $\mathbf{C} \in \mathcal{C}$, such that no weak classifier h can satisfy the requirement (3.2), implying the condition must be vacuous. The i_0^{th} row of the cost matrix is $\mathbf{c}$, and the remaining rows are $\mathbf{0}$. Since $\mathcal{C}_0$ is a cone, $\mathbf{0} \in \mathcal{C}_0$ and hence the cost matrix lies in $\mathcal{C}$. With this choice for $\mathbf{C}$, the condition (3.2) becomes

$$c(h(x_i)) = C\left(i, h(x_i)\right) \le \langle \mathbf{C}(i), \mathbf{B}(i) \rangle = \langle \mathbf{c}, \mathbf{b} \rangle < \min_l c(l),$$

where the last inequality holds since, by assumption, (3.23) is not true for this choice of $\mathbf{c}, \mathbf{b}$. The previous equation is an impossibility, and hence no such weak classifier h exists, showing the condition is vacuous. $\square$

Lemma 3.7 shows that the expression in (3.20) is well defined, and takes on finite values. We next record an alternate dual form for the same recurrence which will be useful later.

Lemma 3.8. *The recurrence in (3.20) is equivalent to*

$$\phi_t^{\mathbf{b}}(\mathbf{s}) = \min_{\mathbf{c} \in \mathcal{C}_0} \max_{l=1}^{k} \left\{ \phi_{t-1}^{\mathbf{b}}\left(\mathbf{s} + \mathbf{e}_l\right) - \left(c(l) - \langle \mathbf{c}, \mathbf{b} \rangle\right) \right\}. \tag{3.24}$$

Proof. Using Lagrangean multipliers, we may convert (3.20) to an unconstrained expression as follows:

$$\phi_t^{\mathbf{b}}(\mathbf{s}) = \min_{\mathbf{c}\in\mathcal{C}_0} \max_{\mathbf{p}\in\Delta\{1,\ldots,k\}} \min_{\lambda\geq 0} \left\{\mathbb{E}_{l\sim\mathbf{p}}\left[\phi_{t-1}^{\mathbf{b}}\left(\mathbf{s}+\mathbf{e}_l\right)\right] - \lambda\left(\mathbb{E}_{l\sim\mathbf{p}}\left[c(l)\right] - \langle\mathbf{c},\mathbf{b}\rangle\right)\right\}.$$

Applying Theorem 3.1 to the inner min-max expression we get

$$\phi_t^{\mathbf{b}}(\mathbf{s}) = \min_{\mathbf{c}\in\mathcal{C}_0} \min_{\lambda\geq 0} \max_{\mathbf{p}\in\Delta\{1,\ldots,k\}} \left\{\mathbb{E}_{l\sim\mathbf{p}}\left[\phi_{t-1}^{\mathbf{b}}\left(\mathbf{s}+\mathbf{e}_l\right)\right] - \left(\mathbb{E}_{l\sim\mathbf{p}}\left[\lambda c(l)\right] - \langle\lambda\mathbf{c},\mathbf{b}\rangle\right)\right\}.$$

Since $\mathcal{C}_0$ is a cone, $\mathbf{c}\in\mathcal{C}_0$ implies $\lambda\mathbf{c}\in\mathcal{C}_0$. Therefore we may absorb the Lagrange multiplier into the cost vector:

$$\phi_t^{\mathbf{b}}(\mathbf{s}) = \min_{\mathbf{c}\in\mathcal{C}_0} \max_{\mathbf{p}\in\Delta\{1,\ldots,k\}} \mathbb{E}_{l\sim\mathbf{p}}\left[\phi_{t-1}^{\mathbf{b}}\left(\mathbf{s}+\mathbf{e}_l\right) - \left(c(l) - \langle\mathbf{c},\mathbf{b}\rangle\right)\right].$$

For a fixed choice of $\mathbf{c}$, the expectation is maximized when the distribution $\mathbf{p}$ is concentrated on a single label that maximizes the inner expression, which completes our proof. □

The dual form of the recurrence is useful for optimally choosing the cost matrix in each round. When the weak learning condition being used is $(\mathcal{C},\mathbf{B})$, [44] proposed a Booster strategy, called the OS strategy, which always chooses the weight $\alpha_t = 1$, and uses the potential functions to construct a cost matrix $\mathbf{C}_t$ as follows. Each row $\mathbf{C}_t(i)$ of the matrix achieves the minimum of the right hand side of (3.24) with $\mathbf{b}$ replaced by $\mathbf{B}(i)$, t replaced by $T-t$, and $\mathbf{s}$ replaced by current state $\mathbf{s}_t(i)$:

$$\mathbf{C}_t(i) = \operatorname*{argmin}_{\mathbf{c}\in\mathcal{C}_0} \max_{l=1}^{k} \left\{\phi_{T-t-1}^{\mathbf{B}(i)}\left(\mathbf{s}+\mathbf{e}_l\right) - \left(c(l) - \langle\mathbf{c},\mathbf{B}(i)\rangle\right)\right\}. \tag{3.25}$$

The following theorem, proved in the appendix, provides a guarantee for the loss suffered by the OS algorithm, and also shows that it is the game-theoretically optimum strategy when the number of examples is large. Similar results have been proved by Schapire [44], but our theorem holds much more generally, and also achieves tighter lower bounds.

Theorem 3.9 (Extension of results in [44])**.** *Suppose the weak-learning condition is not vacuous and is given by $(\mathcal{C},\mathbf{B})$, where $\mathcal{C}$ is such that, for some convex cone $\mathcal{C}_0\subseteq\mathbb{R}^k$, the condition (3.17) holds. Let the potential functions $\phi_t^{\mathbf{b}}$ be defined as in (3.20), and assume the Booster employs the OS algorithm, choosing $\alpha_t = 1$ and $\mathbf{C}_t$ as in (3.25) in each round t. Then the average potential of the states,*

$$\frac{1}{m}\sum_{i=1}^{m}\phi_{T-t}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i)\right),$$

never increases in any round. In particular, the loss suffered after T rounds of play is at most

$$\frac{1}{m}\sum_{i=1}^{m}\phi_T^{\mathbf{B}(i)}(\mathbf{0}). \tag{3.26}$$

Further, under certain conditions, this bound is nearly tight. In particular, assume the loss function does not vary too much but satisfies

$$\sup_{\mathbf{s},\mathbf{s}'\in\mathcal{S}_T}|L(\mathbf{s})-L(\mathbf{s}')|\leq\varnothing(L,T), \tag{3.27}$$

where $\mathcal{S}_T$, a subset of $\left\{\mathbf{s}\in\mathbb{R}^k:\|\mathbf{s}\|_\infty\leq T\right\}$, is the set of all states reachable in T iterations, and $\varnothing(L,T)$ is an upper bound on the discrepancy of losses between any two reachable states when the loss function is L and the total number of iterations is T. Then, for any $\varepsilon>0$, when the number of examples m is sufficiently large,

$$m\geq\frac{T\varnothing(L,T)}{\varepsilon}, \tag{3.28}$$

no Booster strategy can guarantee to achieve in T rounds a loss that is ε less than the bound (3.26).

In order to implement the near optimal OS strategy, we need to solve (3.25). This is computationally only as hard as evaluating the potentials, which in turn reduces to computing the recurrences in (3.20). In the next few sections, we study how to do this when using various losses and weak learning conditions.

3.6 Solving for any fixed edge-over-random condition

In this section we show how to implement the OS strategy when the weak learning condition is any fixed edge-over-random condition: $(\mathcal{C},\mathbf{B})$ for some $\mathbf{B}\in\mathcal{B}_\gamma^{\text{eor}}$. By our previous discussions, this is equivalent to computing the potential $\phi_t^{\mathbf{b}}$ by solving the recurrence in (3.20), where the vector $\mathbf{b}$ corresponds to some row of the baseline $\mathbf{B}$. Let $\Delta_\gamma^k\subseteq\Delta\{1,\ldots,k\}$ denote the set of all edge-over-random distributions on $\{1,\ldots,k\}$ with γ more weight on the first coordinate:

$$\Delta_\gamma^k=\{\mathbf{b}\in\Delta\{1,\ldots,k\}:b(1)-\gamma=\max\{b(2),\ldots,b(k)\}\}. \tag{3.29}$$

Note, that $\mathcal{B}_\gamma^{\text{eor}}$ consists of all matrices whose rows belong to the set Δ_γ^k. Therefore we are interested in computing $\phi^{\mathbf{b}}$, where $\mathbf{b}$ is an arbitrary edge-over-random distribution: $\mathbf{b}\in\Delta_\gamma^k$. We begin by simplifying the recurrence (3.20) satisfied by such potentials, and show how to compute the optimal cost matrix in terms of the potentials.

Lemma 3.10. *Assume L is proper, and $\mathbf{b} \in \Delta_\gamma^k$ is an edge-over-random distribution. Then the recurrence (3.20) may be simplified as*

$$\phi_t^{\mathbf{b}}(\mathbf{s}) \quad = \quad \mathbb{E}_{l\sim\mathbf{b}}\left[\phi_{t-1}\left(\mathbf{s}+\mathbf{e}_l\right)\right]. \tag{3.30}$$

Further, if the cost matrix $\mathbf{C}_t$ is chosen as follows

$$C_t(i,l) = \phi_{T-t-1}^{\mathbf{b}}(\mathbf{s}_t(i)+\mathbf{e}_l), \tag{3.31}$$

then $\mathbf{C}_t$ satisfies the condition in (3.25), and hence is the optimal choice.

Proof. Let $\mathcal{C}_0^{\text{eor}} \subseteq \mathbb{R}^k$ denote all vectors $\mathbf{c}$ satisfying $\forall l : c(1) \le c(l)$. Then, we have

$$\begin{aligned}
\phi_t^{\mathbf{b}}(\mathbf{s}) &= \begin{array}{ccc} \min\limits_{\mathbf{c}\in\mathcal{C}_0^{\text{eor}}} & \max\limits_{\mathbf{p}\in\Delta\{1,\dots,k\}} & \mathbb{E}_{l\sim\mathbf{p}}\left[\phi_{t-1}\left(\mathbf{s}+\mathbf{e}_l\right)\right] \\ & \text{s.t.} & \mathbb{E}_{l\sim\mathbf{p}}[c(l)] \le \mathbb{E}_{l\sim\mathbf{b}}\left[c(l)\right], \end{array} \quad (\text{ by (3.20) }) \\
&= \min_{\mathbf{c}\in\mathcal{C}_0^{\text{eor}}} \max_{\mathbf{p}\in\Delta} \min_{\lambda\ge 0} \left\{\mathbb{E}_{l\sim\mathbf{p}}\left[\phi_{t-1}^{\mathbf{b}}\left(\mathbf{s}+\mathbf{e}_l\right)\right] + \lambda\left(\mathbb{E}_{l\sim\mathbf{b}}\left[c(l)\right] - \mathbb{E}_{l\sim\mathbf{p}}[c(l)]\right)\right\} \text{ (Lagrangean)} \\
&= \min_{\mathbf{c}\in\mathcal{C}_0^{\text{eor}}} \min_{\lambda\ge 0} \max_{\mathbf{p}\in\Delta} \mathbb{E}_{l\sim\mathbf{p}}\left[\phi_{t-1}^{\mathbf{b}}\left(\mathbf{s}+\mathbf{e}_l\right)\right] + \lambda\left\langle\mathbf{b}-\mathbf{p},\mathbf{c}\right\rangle \text{ (Theorem 3.1)} \\
&= \min_{\mathbf{c}\in\mathcal{C}_0^{\text{eor}}} \max_{\mathbf{p}\in\Delta} \mathbb{E}_{l\sim\mathbf{p}}\left[\phi_{t-1}^{\mathbf{b}}\left(\mathbf{s}+\mathbf{e}_l\right)\right] + \left\langle\mathbf{b}-\mathbf{p},\mathbf{c}\right\rangle \text{ (absorb } \lambda \text{ into } \mathbf{c}) \\
&= \max_{\mathbf{p}\in\Delta} \min_{\mathbf{c}\in\mathcal{C}_0^{\text{eor}}} \mathbb{E}_{l\sim\mathbf{p}}\left[\phi_{t-1}^{\mathbf{b}}\left(\mathbf{s}+\mathbf{e}_l\right)\right] + \left\langle\mathbf{b}-\mathbf{p},\mathbf{c}\right\rangle \text{ (Theorem 3.1)}.
\end{aligned}$$

Unless $b(1)-p(1) \le 0$ and $b(l)-p(l) \ge 0$ for each $l > 1$, the quantity $\langle\mathbf{b}-\mathbf{p},\mathbf{c}\rangle$ can be made arbitrarily small for appropriate choices of $\mathbf{c} \in \mathcal{C}_0^{\text{eor}}$. The max-player is therefore forced to constrain its choices of $\mathbf{p}$, and the above expression becomes

$$\begin{array}{cl} \max\limits_{\mathbf{p}\in\Delta} & \mathbb{E}_{l\sim\mathbf{p}}\left[\phi_{t-1}^{\mathbf{b}}\left(\mathbf{s}+\mathbf{e}_l\right)\right] \\ \text{s.t.} & b(l)-q(l)\begin{cases} \ge 0 & \text{if } l=1, \\ \le 0 & \text{if } l>1. \end{cases} \end{array}$$

Lemma 6 of [44] states that if L is *proper* (as defined here), so is $\phi_t^{\mathbf{b}}$; the same result can be extended to our drifting games. This implies the optimal choice of $\mathbf{p}$ in the above expression is in fact the distribution that puts as small weight as possible in the first coordinate, namely $\mathbf{b}$. Therefore the optimum choice of $\mathbf{p}$ is $\mathbf{b}$, and the potential is the same as in (3.30).

We end the proof by showing that the choice of cost matrix in (3.31) is optimum. Theorem 3.9 states that a cost matrix $\mathbf{C}_t$ is the optimum choice if it satisfies (3.25), that is, if the expression

$$\max_{l=1}^{k}\left\{\phi_{T-t-1}^{\mathbf{B}(i)}\left(\mathbf{s}+\mathbf{e}_l\right) - \left(C_t(i,l) - \left\langle\mathbf{C}_t(i),\mathbf{B}(i)\right\rangle\right)\right\} \tag{3.32}$$

is equal to

$$\min_{\mathbf{c}\in\mathcal{C}_0} \max_{l=1}^{k}\left\{\phi_{T-t-1}^{\mathbf{B}(i)}\left(\mathbf{s}+\mathbf{e}_l\right) - \left(c(l) - \left\langle\mathbf{c},\mathbf{B}(i)\right\rangle\right)\right\} = \phi_{T-t}^{\mathbf{B}(i)}\left(\mathbf{s}\right), \tag{3.33}$$

where the equality in (3.33) follows from (3.24). If $\mathbf{C}_t(i)$ is chosen as in (3.31), then, for any label l, the expression within max in (3.32) evaluates to

$$\begin{aligned}\phi_{T-t-1}^{\mathbf{B}(i)}(\mathbf{s}+\mathbf{e}_l) &- \left(\phi_{T-t-1}^{\mathbf{B}(i)}(\mathbf{s}+\mathbf{e}_l) - \langle \mathbf{C}_t(i), \mathbf{B}(i)\rangle\right) \\ &= \langle \mathbf{B}(i), \mathbf{C}_t(i)\rangle \\ &= \mathbb{E}_{l\sim\mathbf{B}(i)}\left[C_t(i,l)\right] \\ &= \mathbb{E}_{l\sim\mathbf{B}(i)}\left[\phi_{T-t-1}^{\mathbf{B}(i)}(\mathbf{s}+\mathbf{e}_l)\right] \\ &= \phi_{T-t}^{\mathbf{B}(i)}(\mathbf{s}),\end{aligned}$$

where the last equality follows from (3.30). Therefore the max expression in (3.32) is also equal to $\phi_{T-t}^{\mathbf{B}(i)}(\mathbf{s})$, which is what we needed to show. □

Eq. (3.31) in Lemma 3.10 implies the cost matrix chosen by the OS strategy can be expressed in terms of the potentials, which is the only thing left to calculate. Fortunately, the simplification (3.30) of the drifting games recurrence, allows the potentials to be solved completely in terms of a random-walk $\mathcal{R}_{\mathbf{b}}^t(\mathbf{x})$. This random variable denotes the position of a particle after t time steps, that starts at location $\mathbf{x}\in\mathbb{R}^k$, and in each step moves in direction $\mathbf{e}_l$ with probability $b(l)$.

Corollary 3.11. *The recurrence in* (3.30) *can be solved as follows:*

$$\phi_t^{\mathbf{b}}(\mathbf{s}) = \mathbb{E}\left[L\left(\mathcal{R}_{\mathbf{b}}^t(\mathbf{s})\right)\right]. \tag{3.34}$$

Proof. Inductively assuming $\phi_{t-1}^{\mathbf{b}}(\mathbf{x}) = \mathbb{E}\left[L(\mathcal{R}_{\mathbf{b}}^{t-1}(\mathbf{x}))\right]$,

$$\phi_t(\mathbf{s}) = \mathbb{E}_{l\sim\mathbf{b}}\left[L(\mathcal{R}_{\mathbf{b}}^{t-1}(\mathbf{s})+\mathbf{e}_l)\right] = \mathbb{E}\left[L(\mathcal{R}_{\mathbf{b}}^t(\mathbf{s}))\right].$$

The last equality follows by observing that the random position $\mathcal{R}_{\mathbf{b}}^{t-1}(\mathbf{s})+\mathbf{e}_l$ is distributed as $\mathcal{R}_{\mathbf{b}}^t(\mathbf{s})$ when l is sampled from $\mathbf{b}$. □

Lemma 3.10 and Corollary 3.11 together imply:

Theorem 3.12. *Assume L is proper and $\mathbf{b}\in\Delta_\gamma^k$ is an edge-over-random distribution. Then the potential $\phi_t^{\mathbf{b}}$, defined by the recurrence in* (3.20)*, has the solution given in* (3.34) *in terms of random walks.*

Before we can compute (3.34), we need to choose a loss function L. We next consider two options for the loss — the non-convex 0-1 error, and exponential loss.

Exponential Loss. The exponential loss serves as a smooth convex proxy for discontinuous non-convex 0-1 error (3.19) that we would ultimately like to bound, and is given by

$$L_\eta^{\exp}(\mathbf{s}) = \sum_{l=2}^{k} e^{\eta(s_l - s_1)}. \tag{3.35}$$

The parameter η can be thought of as the weight in each round, that is, $\alpha_t = \eta$ in each round. Then notice that the weighted state $\mathbf{f}_t$ of the examples, defined in (3.22), is related to the unweighted states $\mathbf{s}_t$ as $f_t(l) = \eta s_t(l)$. Therefore the exponential loss function in (3.35) directly measures the loss of the weighted state as

$$L^{\exp}(\mathbf{f}_t) = \sum_{l=2}^{k} e^{f_t(l) - f_t(1)}. \tag{3.36}$$

Because of this correspondence, the optimal strategy with the loss function $L^{\exp}$ and $\alpha_t = \eta$ is the same as that using loss $L^{\exp}_{\eta}$ and $\alpha_t = 1$. We study the latter setting so that we may use the results derived earlier. With the choice of the exponential loss $L^{\exp}_{\eta}$, the potentials are easily computed, and in fact have a closed form solution.

Theorem 3.13. *If $L^{\exp}_{\eta}$ is as in (3.35), where η is non-negative, then the solution in Theorem 3.12 evaluates to $\phi^{\mathbf{b}}_t(\mathbf{s}) = \sum_{l=2}^{k} (a_l)^t e^{\eta_l(s_l - s_1)}$, where $a_l = 1 - (b_1 + b_l) + e^{\eta} b_l + e^{-\eta} b_1$.*

The proof by induction is straightforward. By tuning the weight η, each a_l can be always made less than 1. This ensures the exponential loss decays exponentially with rounds. In particular, when $\mathbf{B} = \mathbf{U}_{\gamma}$ (so that the condition is $(\mathcal{C}^{\text{eor}}, \mathbf{U}_{\gamma})$), the relevant potential $\phi_t(\mathbf{s})$ or $\phi_t(\mathbf{f})$ is given by

$$\phi_t(\mathbf{s}) = \phi_t(f) = \kappa(\gamma, \eta)^t \sum_{l=2}^{k} e^{\eta(s_l - s_1)} = \kappa(\gamma, \eta)^t \sum_{l=2}^{k} e^{f_l - f_1} \tag{3.37}$$

where

$$\kappa(\gamma, \eta) = 1 + \frac{(1-\gamma)}{k} \left(e^{\eta} + e^{-\eta} - 2\right) - \left(1 - e^{-\eta}\right) \gamma. \tag{3.38}$$

The cost-matrix output by the OS algorithm can be simplified by rescaling, or adding the same number to each coordinate of a cost vector, without affecting the constraints it imposes on a weak classifier, to the following form

$$C(i, l) = \begin{cases} (e^{\eta} - 1)\, e^{\eta(s_l - s_1)} & \text{if } l > 1, \\ (e^{-\eta} - 1) \sum_{l=2}^{k} e^{\eta(s_l - s_1)} & \text{if } l = 1. \end{cases}$$

Using the correspondence between unweighted and weighted states, the above may also be rewritten as:

$$C(i, l) = \begin{cases} (e^{\eta} - 1)\, e^{f_l - f_1} & \text{if } l > 1, \\ (e^{-\eta} - 1) \sum_{l=2}^{k} e^{f_l - f_1} & \text{if } l = 1. \end{cases} \tag{3.39}$$

With such a choice, Theorem 3.9 and the form of the potential guarantee that the average loss

$$\frac{1}{m} \sum_{i=1}^{m} L^{\exp}_{\eta}(\mathbf{s}_t(i)) = \frac{1}{m} \sum_{i=1}^{m} L^{\exp}(\mathbf{f}_t(i)) \tag{3.40}$$

of the states changes by a factor of at most $\kappa(\gamma, \eta)$ every round. Therefore the final loss, which upper bounds the error, i.e., the fraction of misclassified training examples, is at most $(k-1)\kappa(\gamma,\eta)^T$. Since this upper bound holds for any value of η, we may tune it to optimize the bound. Setting $\eta = \ln(1+\gamma)$, the error can be upper bounded by $(k-1)e^{-T\gamma^2/2}$.

Zero-one Loss. There is no simple closed form solution for the potential when using the zero-one loss L^{err} (3.19). However, we may compute the potentials efficiently as follows. To compute $\phi_t^{\mathbf{b}}(\mathbf{s})$, we need to find the probability that a random walk (making steps according to $\mathbf{b}$) of length t in $\mathbb{Z}^k$, starting at $\mathbf{s}$ will end up in a region where the loss function is 1. Any such random walk will consist of x_l steps in direction $\mathbf{e}_l$ where the non-negative $\sum_l x_l = t$. The probability of each such path is $\prod_l b_l^{x_l}$. Further, there are exactly $\binom{t}{x_1,\ldots,x_k}$ such paths. Starting at state $\mathbf{s}$, such a path will lead to a correct answer only if $s_1 + x_1 > s_l + x_l$ for each $l > 1$. Hence we may write the potential $\phi_t^{\mathbf{b}}(\mathbf{s})$ as

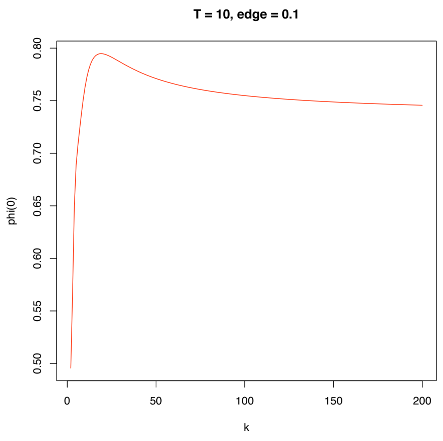
$$\phi_t^{\mathbf{b}}(\mathbf{s}) = 1 - \sum_{x_1,\ldots,x_k}^{t} \binom{t}{x_1,\ldots,x_k} \prod_{l=1}^k b_l^{x_l}$$
$$\begin{array}{lcl} \text{s.t.} & x_1 + \ldots + x_k & = t \\ \forall l: & x_l & \geq 0 \\ \forall l: & x_l + s_l & \leq x_1 + s_1. \end{array}$$

Since the x_l's are restricted to be integers, this problem is presumably hard. In particular, the only algorithms known to the authors that take time logarithmic in t is also exponential in k. However, by using dynamic programming, we can compute the summation in time polynomial in $|s_l|$, t and k. In fact, the run time is always $O(t^3k)$, and at least $\Omega(tk)$.

The bounds on error we achieve, although not in closed form, are much tighter than those obtainable using exponential loss. The exponential loss analysis yields an error upper bound of $(k-1)e^{-T\gamma^2/2}$. Using a different initial distribution, Schapire and Singer [47] achieve the slightly better bound $\sqrt{(k-1)}e^{-T\gamma^2/2}$. However, when the edge γ is small and the number of rounds are few, each bound is greater than 1 and hence trivial. On the other hand, the bounds computed by the above dynamic program are sensible for all values of k, γ and T. When $\mathbf{b}$ is the γ-biased uniform distribution $\mathbf{b} = (\frac{1-\gamma}{k}+\gamma, \frac{1-\gamma}{k}, \frac{1-\gamma}{k}, \ldots, \frac{1-\gamma}{k})$ a table containing the error upper bound $\phi_T^{\mathbf{b}}(0)$ for $k = 6$, $\gamma = 0$ and small values for the number of rounds T is shown in Figure 3.2(a); note that with the exponential loss, the bound is always 1 if the edge γ is 0. Further, the bounds due to the exponential loss analyses seem to imply that the dependence of the error on the number of labels is monotonic. However, a plot of the tighter bounds with edge $\gamma = 0.1$, number of rounds $T = 10$ against various values of k, shown in Figure 3.2(b), indicates that the true dependence is more complicated. Therefore the tighter analysis also provides qualitative insights not obtainable via the exponential loss bound.

T	$\phi_T^{\mathbf{b}}(\mathbf{0})$	T	$\phi_T^{\mathbf{b}}(\mathbf{0})$
0	1.00	6	0.90
1	0.83	7	0.91
2	0.97	8	0.90
3	0.93	9	0.89
4	0.89	10	0.89
5	0.89		

(a)



(b)

Figure 3.2: Plot of potential value $\phi_T^{\mathbf{b}}(\mathbf{0})$ where $\mathbf{b}$ is the γ-biased uniform distribution: $\mathbf{b} = (\frac{1-\gamma}{k} + \gamma, \frac{1-\gamma}{k}, \frac{1-\gamma}{k}, \ldots, \frac{1-\gamma}{k})$. **(a):** Potential values (rounded to two decimal places) for different number of rounds T using $\gamma = 0$ and $k = 6$. These are bounds on the error, and less than 1 even when the edge and number of rounds are small. **(b):** Potential values for different number of classes k, with $\gamma = 0.1$, and $T = 10$. These are tight estimates for the optimal error, and yet not monotonic in the number of classes.

3.7 Solving for the minimal weak learning condition

In the previous section we saw how to find the optimal boosting strategy when using any fixed edge-over-random condition. However as we have seen before, these conditions can be stronger than necessary, and therefore lead to boosting algorithms that require additional assumptions. Here we show how to compute the optimal algorithm while using the weakest weak learning condition, provided by (3.16), or equivalently the condition used by AdaBoost.MR, $(\mathcal{C}^{\mathrm{MR}}, \mathbf{B}_\gamma^{\mathrm{MR}})$. Since there are two possible formulations for the minimal condition, it is not immediately clear which to use to compute the optimal boosting strategy. To resolve this, we first show that the optimal boosting strategy based on any formulation of a necessary and sufficient weak learning condition is the same. Having resolved this ambiguity, we show how to compute this strategy for the exponential loss and 0-1 error using the first formulation.

3.7.1 Game-theoretic equivalence of necessary and sufficient weak-learning conditions

In this section we study the effect of the weak learning condition on the game-theoretically optimal boosting strategy. We introduce the notion of *game-theoretic equivalence* between two weak learning conditions, that determines if the payoffs (3.18) of the optimal boosting strategies based on the two conditions are identical.

This is different from the usual notion of equivalence between two conditions, which holds if any weak classifier space satisfies both conditions or neither condition. In fact we prove that game-theoretic equivalence is a broader notion; in other words, equivalence implies game-theoretic equivalence. A special case of this general result is that any two weak learning conditions that are necessary and sufficient, and hence equivalent to boostability, are also game-theoretically equivalent. In particular, so are the conditions of AdaBoost.MR and (3.16), and the resulting optimal Booster strategies enjoy equally good payoffs. We conclude that in order to derive the optimal boosting strategy that uses the minimal weak-learning condition, it is sound to use either of these two formulations.

The purpose of a weak learning condition $(\mathcal{C}, \mathbf{B})$ is to impose restrictions on the Weak-Learner's responses in each round. These restrictions are captured by subsets of the weak classifier space as follows. If Booster chooses cost-matrix $\mathbf{C} \in \mathcal{C}$ in a round, the Weak-Learner's response h is restricted to the subset $S_{\mathbf{C}} \subseteq \mathcal{H}^{\text{all}}$ defined as

$$S_{\mathbf{C}} = \left\{ h \in \mathcal{H}^{\text{all}} : \mathbf{C} \bullet \mathbf{1}_h \leq \mathbf{C} \bullet \mathbf{B} \right\}.$$

Thus, a weak learning condition is essentially a family of subsets of the weak classifier space. Further, smaller subsets mean fewer options for Weak-Learner, and hence better payoffs for the optimal boosting strategy. Based on this idea, we may define when a weak learning condition $(\mathcal{C}_1, \mathbf{B}_1)$ is *game-theoretically stronger* than another condition $(\mathcal{C}_2, \mathbf{B}_2)$ if the following holds: For every subset $S_{\mathbf{C}_2}$ in the second condition (that is $\mathbf{C}_2 \in \mathcal{C}_2$), there is a subset $S_{\mathbf{C}_1}$ in the first condition (that is $\mathbf{C}_1 \in \mathcal{C}_1$), such that $S_{\mathbf{C}_1} \subseteq S_{\mathbf{C}_2}$. Mathematically, this may be written as follows:

$$\forall \mathbf{C}_1 \in \mathcal{C}_1, \exists \mathbf{C}_2 \in \mathcal{C}_2 : S_{\mathbf{C}_1} \subseteq S_{\mathbf{C}_2}.$$

Intuitively, a game theoretically stronger condition will allow Booster to place similar or stricter restrictions on Weak-Learner in each round. Therefore, the optimal Booster payoff using a game-theoretically stronger condition is at least equally good, if not better. It therefore follows that if two conditions are both game-theoretically stronger than each other, the corresponding Booster payoffs must be equal, that is they must be *game-theoretically equivalent*.

Note that game-theoretic equivalence of two conditions does not mean that they are identical as families of subsets, for we may arbitrarily add large and "useless" subsets to the two conditions without affecting the Booster payoffs, since these subsets will never be used by an optimal Booster strategy. In fact we next show that game-theoretic equivalence is a broader notion than just equivalence.

Theorem 3.14. *Suppose $(\mathcal{C}_1, \mathbf{B}_1)$ and $(\mathcal{C}_2, \mathbf{B}_2)$ are two equivalent weak learning conditions, that is, every space $\mathcal{H}$ satisfies both or neither condition. Then each condition is game-theoretically stronger than the other, and hence game-theoretically equivalent.*

Proof. We argue by contradiction. Assume that despite equivalence, the first condition (without loss of generality) includes a particularly hard subset $S_{\mathbf{C}_1} \subseteq \mathcal{H}^{\text{all}}, \mathbf{C}_1 \in \mathcal{C}_1$ which is not smaller than any subset in the second condition. In particular, for every subset $S_{\mathbf{C}_2}, \mathbf{C}_2 \in \mathcal{C}_2$ in the second condition is satisfied by some weak classifier $h_{\mathbf{C}_2}$ not satisfying the hard subset in the first condition: $h_{\mathbf{C}_2} \in S_{\mathbf{C}_2} \setminus S_{\mathbf{C}_1}$. Therefore, the space

$$\mathcal{H} = \{h_{\mathbf{C}_2} : \mathbf{C}_2 \in \mathcal{C}_2\},$$

formed by just these classifiers satisfies the second condition, but has an empty intersection with $S_{\mathbf{C}_1}$. In other words, $\mathcal{H}$ satisfies the second but not the first condition, a contradiction to their equivalence. □

An immediate corollary is the game theoretic equivalence of necessary and equivalent conditions.

Corollary 3.15. *Any two necessary and sufficient weak learning conditions are game-theoretically equivalent. In particular the optimum Booster strategies based on AdaBoost.MR's condition* $(\mathcal{C}^{MR}, \mathbf{B}_\gamma^{MR})$ *and* (3.16) *have equal payoffs.*

Therefore, in deriving the optimal Booster strategy, it is sound to work with either AdaBoost.MR's condition or (3.16). In the next section, we actually compute the optimal strategy using the latter formulation.

3.7.2 Optimal strategy with the minimal conditions

In this section we compute the optimal Booster strategy that uses the minimum weak learning condition provided in (3.16). We choose this instead of AdaBoost.MR's condition because this description is more closely related to the edge-over-random conditions, and the resulting algorithm has a close relationship to the ones derived for fixed edge-over-random conditions, and therefore more insightful. However, this formulation does not state the condition as a single pair $(\mathbf{C}, \mathbf{B})$, and therefore we cannot directly use the result of Theorem 3.9. Instead, we define new potentials and a modified OS strategy that is still nearly optimal, and this constitutes the first part of this section. In the second part, we show how to compute these new potentials and the resulting OS strategy.

Modified potentials and OS strategy

The condition in (3.16) is not stated as a single pair $(\mathcal{C}^{\text{eor}}, \mathbf{B})$, but uses all possible edge-over-random baselines $\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}$. Therefore, we modify the definitions (3.20) of the potentials accordingly to extract an optimal Booster strategy. Recall that Δ_γ^k is defined in (3.29) as the set of all edge-over-random distributions which constitute the rows of edge-over-random baselines $\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}$. Using these, define new potentials $\phi_t(\mathbf{s})$ as follows:

$$\phi_t(\mathbf{s}) = \min_{\mathbf{c} \in \mathcal{C}_0^{\text{eor}}} \max_{\mathbf{b} \in \Delta_\gamma^k} \begin{array}{ll} \max\limits_{\mathbf{p} \in \Delta\{1,\ldots,k\}} & \mathbb{E}_{l \sim \mathbf{p}}\left[\phi_{t-1}\left(\mathbf{s} + \mathbf{e}_l\right)\right] \\ \text{s.t.} & \mathbb{E}_{l \sim \mathbf{p}}[c(l)] \leq \langle \mathbf{b}, \mathbf{c} \rangle . \end{array} \tag{3.41}$$

The main difference between (3.41) and (3.20) is that while the older potentials were defined using a fixed vector $\mathbf{b}$ corresponding to some row in the fixed baseline $\mathbf{B}$, the new definition takes the maximum over all possible rows $\mathbf{b} \in \Delta_\gamma^k$ that an edge-over-random baseline $\mathbf{B} \in \mathcal{B}_\gamma^{\text{eor}}$ may have. As before, we may write the recurrence in (3.41) in its dual form

$$\phi_t(\mathbf{s}) = \min_{\mathbf{c}\in\mathcal{C}_0^{\text{eor}}} \max_{\mathbf{b}\in\Delta_\gamma^k} \max_{l=1}^{k} \left\{\phi_{t-1}\left(\mathbf{s}+\mathbf{e}_l\right) - \left(c(l) - \langle \mathbf{c}, \mathbf{b}\rangle\right)\right\}. \tag{3.42}$$

The proof is very similar to that of Lemma 3.8 and is omitted. We may now define a new OS strategy that chooses a cost-matrix in round t analogously:

$$\mathbf{C}_t(i) \in \operatorname*{argmin}_{\mathbf{c}\in\mathcal{C}_0^{\text{eor}}} \max_{\mathbf{b}\in\Delta_\gamma^k} \max_{l=1}^{k} \left\{\phi_{t-1}\left(\mathbf{s}+\mathbf{e}_l\right) - \left(c(l) - \langle \mathbf{c}, \mathbf{b}\rangle\right)\right\}. \tag{3.43}$$

where recall that $\mathbf{s}_t(i)$ denotes the state vector (defined in (3.21)) of example i. With this strategy, we can show an optimality result very similar to Theorem 3.9.

Theorem 3.16. *Suppose the weak-learning condition is given by (3.16). Let the potential functions $\phi_t^{\mathbf{b}}$ be defined as in (3.41), and assume the Booster employs the modified OS strategy, choosing $\alpha_t = 1$ and $\mathbf{C}_t$ as in (3.43) in each round t. Then the average potential of the states,*

$$\frac{1}{m}\sum_{i=1}^{m} \phi_{T-t}\left(\mathbf{s}_t(i)\right),$$

never increases in any round. In particular, the loss suffered after T rounds of play is at most $\phi_T(\mathbf{0})$.

Further, for any $\varepsilon > 0$, when the loss function satisfies (3.27) and the number of examples m is as large as in (3.28), no Booster strategy can guarantee to achieve less than $\phi_T(\mathbf{0}) - \varepsilon$ loss in T rounds.

The proof is very similar to that of Theorem 3.9 and is omitted.

Computing the new potentials.

Here we show how to compute the new potentials. The resulting algorithms will require exponential time, and we provide some empirical evidence showing that this might be necessary. Finally, we show how to carry out the computations efficiently in certain special situations.

An exponential time algorithm. Here we show how the potentials may be computed as the expected loss of some random walk, just as we did for the potentials arising with fixed edge-over-random conditions. The main difference is there will be several random walks to choose from.

We first begin by simplifying the recurrence (3.41), and expressing the optimal cost matrix in (3.43) in terms of the potentials, just as we did in Lemma 3.10 for the case of fixed edge-over-random conditions.

Lemma 3.17. *Assume L is proper. Then the recurrence (3.41) may be simplified as*

$$\phi_t(\mathbf{s}) \quad = \quad \max_{\mathbf{b}\in\Delta_\gamma^k} \mathbb{E}_{l\sim\mathbf{b}}\left[\phi_{t-1}\left(\mathbf{s}+\mathbf{e}_l\right)\right]. \tag{3.44}$$

Further, if the cost matrix $\mathbf{C}_t$ is chosen as follows:

$$C_t(i,l) = \phi_{T-t-1}(\mathbf{s}_t(i)+\mathbf{e}_l), \tag{3.45}$$

then $\mathbf{C}_t$ satisfies the condition in (3.43).

The proof is very similar to that of Lemma 3.10 and is omitted. Eq. (3.45) implies that, as before, computing the optimal Booster strategy reduces to computing the new potentials. One computational difficulty created by the new definitions (3.41) or (3.44) is that they require infinitely many possible distributions $\mathbf{b} \in \Delta_\gamma^k$ to be considered. We show that we may in fact restrict our attention to only finitely many of such distributions described next.

At any state $\mathbf{s}$ and number of remaining iterations t, let π be a permutation of the coordinates $\{2,\ldots,k\}$ that sorts the potential values:

$$\phi_{t-1}\left(\mathbf{s}+\mathbf{e}_{\pi(k)}\right) \geq \phi_{t-1}\left(\mathbf{s}+\mathbf{e}_{\pi(k-1)}\right) \geq \ldots \geq \phi_{t-1}\left(\mathbf{s}+\mathbf{e}_{\pi(2)}\right). \tag{3.46}$$

For any permutation π of the coordinates $\{2,\ldots,k\}$, let $\mathbf{b}_a^\pi$ denote the γ-biased uniform distribution on the a coordinates $\{1,\pi_k,\pi_{k-1},\ldots,\pi_{k-a+2}\}$:

$$b_a^\pi(l) = \begin{cases} \frac{1-\gamma}{a}+\gamma & \text{if } l=1 \\ \frac{1-\gamma}{a} & \text{if } l\in\{\pi_k,\ldots,\pi_{k-a+2}\} \\ 0 & \text{otherwise.} \end{cases} \tag{3.47}$$

Then, the next lemma shows that we may restrict our attention to only the distributions $\{\mathbf{b}_2^\pi,\ldots,\mathbf{b}_k^\pi\}$ when evaluating the recurrence in (3.44).

Lemma 3.18. *Fix a state $\mathbf{s}$ and remaining rounds of boosting t. Let π be a permutation of the coordinates $\{2,\ldots,k\}$ satisfying (3.46), and define $\mathbf{b}_a^\pi$ as in (3.47). Then the recurrence (3.44) may be simplified as follows:*

$$\phi_t(\mathbf{s}) = \max_{\mathbf{b}\in\Delta_\gamma^k} \mathbb{E}_{l\sim\mathbf{b}}\left[\phi_{t-1}\left(\mathbf{s}+\mathbf{e}_l\right)\right] = \max_{2\leq a\leq k} \mathbb{E}_{l\sim\mathbf{b}_a^\pi}\left[\phi_{t-1}\left(\mathbf{s}+\mathbf{e}_l\right)\right]. \tag{3.48}$$

Proof. Assume (by relabeling the coordinates if necessary) that π is the identity permutation, that is, $\pi(2)=2,\ldots,\pi(k)=k$. Observe that the right hand side of (3.44) is at least as much the right hand side of (3.48) since the former considers more distributions. We complete the proof by showing that the former is also at most the latter.

By (3.44), we may assume that some optimal $\mathbf{b}$ satisfies

$$\begin{aligned} b(k) = \cdots = b(k-a+2) &= b(1) - \gamma, \\ b(k-a+1) &\leq b(1) - \gamma, \\ b(k-a) = \cdots = b(2) &= 0. \end{aligned}$$

Therefore, $\mathbf{b}$ is a distribution supported on $a+1$ elements, with the minimum weight placed on element $k-a+1$. This implies $b(k-a+1) \in [0, 1/(a+1)]$.

Now, $\mathbb{E}_{l\sim\mathbf{b}}\left[\phi_{t-1}(\mathbf{s}+\mathbf{e}_l)\right]$ may be written as

$$\begin{aligned} & \gamma \cdot \phi_{t-1}(\mathbf{s}+\mathbf{e}_1) + b(k-a+1)\phi_{t-1}(\mathbf{s}+\mathbf{e}_{k-a+1}) \\ + \quad & (1-\gamma-b(k-a+1))\frac{\phi_{t-1}(\mathbf{s}+\mathbf{e}_1) + \phi_{t-1}(\mathbf{s}+\mathbf{e}_{k-a+2}) + \ldots \phi_{t-1}(\mathbf{s}+\mathbf{e}_k)}{a} \\ = \quad & \gamma \cdot \phi_{t-1}(\mathbf{s}+\mathbf{e}_1) + \frac{b(k-a+1)}{1-\gamma}\phi_{t-1}(\mathbf{s}+\mathbf{e}_{k-a+1}) \\ + \quad & (1-\gamma)\left\{ \left(1 - \frac{b(k-a+1)}{1-\gamma}\right) \frac{\phi_{t-1}(\mathbf{s}+\mathbf{e}_1) + \phi_{t-1}(\mathbf{s}+\mathbf{e}_{k-a+2}) + \ldots \phi_{t-1}(\mathbf{s}+\mathbf{e}_k)}{a} \right\} \end{aligned}$$

Replacing $b(k-a+1)$ by x in the above expression, we get a linear function of x. When restricted to $[0, 1/(a+1)]$ the maximum value is attained at a boundary point. For $x = 0$, the expression becomes

$$\gamma \cdot \phi_{t-1}(\mathbf{s}+\mathbf{e}_1) + (1-\gamma)\frac{\phi_{t-1}(\mathbf{s}+\mathbf{e}_1) + \phi_{t-1}(\mathbf{s}+\mathbf{e}_{k-a+2}) + \ldots \phi_{t-1}(\mathbf{s}+\mathbf{e}_k)}{a}.$$

For $x = 1/(a+1)$, the expression becomes

$$\gamma \cdot \phi_{t-1}(\mathbf{s}+\mathbf{e}_1) + (1-\gamma)\frac{\phi_{t-1}(\mathbf{s}+\mathbf{e}_1) + \phi_{t-1}(\mathbf{s}+\mathbf{e}_{k-a+1}) + \ldots \phi_{t-1}(\mathbf{s}+\mathbf{e}_k)}{a+1}.$$

Since $b(k-a+1)$ lies in $[0, 1/(a+1)]$, the optimal value is at most the maximum of the two. However each of these last two expressions is at most the right hand side of (3.48), completing the proof. □

Unraveling (3.48), we find that $\phi_t(\mathbf{s})$ is the expected loss of the final state reached by some random walk of t steps starting at state $\mathbf{s}$. However, the number of possibilities for the random-walk is huge; indeed, the distribution at each step can be any of the $k-1$ possibilities $\mathbf{b}_a^\pi$ for $a \in \{2, \ldots, k\}$, where the parameter a denotes the size of the support of the γ-biased uniform distribution chosen at each step. In other words, at a given state $\mathbf{s}$ with t rounds of boosting remaining, the parameter a determines the number of directions the optimal random walk will consider taking; we will therefore refer to a as the *degree* of the random walk given $(\mathbf{s}, t)$. Now, the total number of states reachable in T steps is $O\left(T^{k-1}\right)$. If the degree assignment every such state, for every value of $t \leq T$ is fixed in advance, $\mathbf{a} = \{a(\mathbf{s}, t) : t \leq T, \mathbf{s} \text{ reachable}\}$, we may identify a unique random walk $\mathcal{R}^{\mathbf{a},t}(\mathbf{s})$ of length t starting at step $\mathbf{s}$. Therefore the

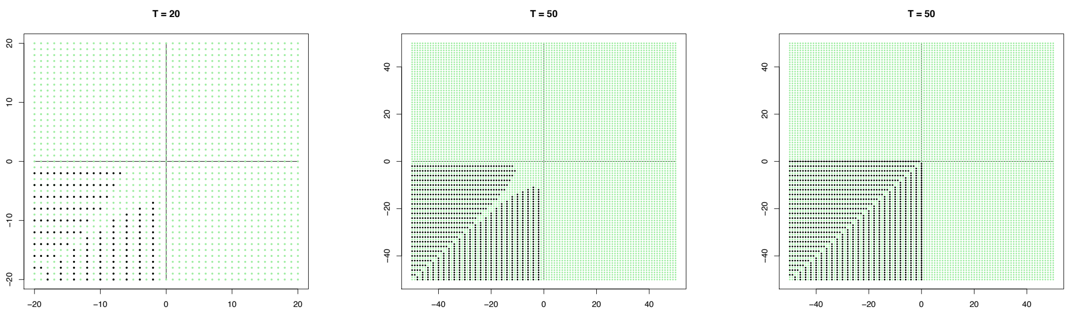


Figure 3.3: Green pixels have degree 3, black pixels have degree 2. Each step is diagonally down (left), and up (if $x < y$) and right (if $x > y$) and both when degree is 3. The rightmost figure uses $\gamma = 0.4$, and the other two $\gamma = 0$. The loss function is 0-1.

potential may be computed as

$$\phi_t(\mathbf{s}) = \max_{\mathbf{a}} \mathbb{E}\left[\mathcal{R}^{\mathbf{a},t}(\mathbf{s})\right]. \tag{3.49}$$

A dynamic programming approach for computing (3.49) requires time and memory linear in the number of different states reachable by a random walk that takes T coordinate steps: $O(T^{k-1})$. This is exponential in the dataset size, and hence impractical. In the next two sections we show that perhaps there may not be any way of computing these efficiently in general, but provide efficient algorithms in certain special cases.

Hardness of evaluating the potentials. Here we provide empirical evidence for the hardness of computing the new potentials. We first identify a computationally easier problem, and show that even that is probably hard to compute. Eq. (3.48) implies that if the potentials were efficiently computable, the correct value of the degree a could be determined efficiently. The problem of determining the degree a given the state $\mathbf{s}$ and remaining rounds t is therefore easier than evaluating the potentials. However, a plot of the degrees against states and remaining rounds, henceforth called a *degree map*, reveals very little structure that might be captured by a computationally efficient function.

We include three such degree maps in Figure 3.3. Only three classes $k = 3$ are used, and the loss function is 0-1 error. We also fix the number T of remaining rounds of boosting and the value of the edge γ for each plot. For ease of presentation, the 3-dimensional states $\mathbf{s} = (s_1, s_2, s_3)$ are compressed into 2-dimensional pixel coordinates $(u = s_2 - s_1, v = s_3 - s_2)$. It can be shown that this does not take away information required to evaluate the potentials or the degree at any pixel (u, v). Further, only those states are considered whose compressed coordinates u, v lie in the range $[-T, T]$; in T rounds, these account for all the reachable states. The degrees are indicated on the plot by colors. Our discussion in the previous sections implies that the possible

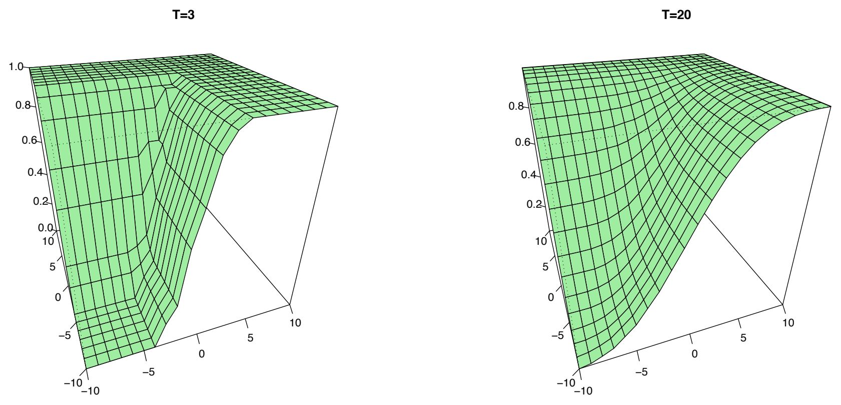


Figure 3.4: Optimum recurrence value. We set $\gamma = 0$. Surface is irregular for smaller values of T, but smoother for larger values, admitting hope for approximation.

values of the degree is 2 or 3. When the degree at a pixel (u, v) is 3, the pixel is colored green, and when the degree is 2, it is colored black.

Note that a random walk over the space $\mathbf{s} \in \mathbb{R}^3$ consisting of distributions over coordinate steps $\{(1,0,0),(0,1,0),(0,0,1)\}$ translates to a random walk over $(u, v) \in \mathbb{R}^2$ where each step lies in the set $\{(-1,-1),(1,0),(0,1)\}$. In Figure 3.3, these correspond to the directions diagonally down, up or right. Therefore at a black pixel, the random walk either chooses between diagonally down and up, or between diagonally down and right, with probabilities $\{1/2 + \gamma/2, 1/2 - \gamma/2\}$. On the other hand, at a green pixel, the random walk chooses among diagonally down, up and right with probabilities $(\gamma + (1-\gamma)/3, (1-\gamma)/3, (1-\gamma)/3)$. The degree maps are shown for varying values of T and the edge γ. While some patterns emerge for the degrees, such as black or green depending on the parity of u or v, the authors found the region near the line $u = v$ still too complex to admit any solution apart from a brute-force computation.

We also plot the potential values themselves in Figure 3.4 against different states. In each plot, the number of iterations remaining, T, is held constant, the number of classes is chosen to be 3, and the edge $\gamma = 0$. The states are compressed into pixels as before, and the potential is plotted against each pixel, resulting in a 3-dimensional surface. We include two plots, with different values for T. The surface is irregular for $T = 3$ rounds, but smoother for 20 rounds, admitting some hope for approximation.

An alternative approach would be to approximate the potential ϕ_t by the potential $\phi_t^{\mathbf{b}}$ for some fixed $\mathbf{b} \in \Delta_\gamma^k$ corresponding to some particular edge-over-random condition. Since $\phi_t \geq \phi_t^{\mathbf{b}}$ for all edge-over-random distributions $\mathbf{b}$, it is natural to approximate by choosing $\mathbf{b}$ that maximizes the fixed edge-over-random potential. (It can be shown that this $\mathbf{b}$ corresponds to the γ-biased uniform distribution.) Two plots of comparing the potential values at $\mathbf{0}$, $\phi_T(\mathbf{0})$ and $\max_{\mathbf{b}} \phi_T^{\mathbf{b}}(\mathbf{0})$, which correspond to the respective error upper bounds, is shown in Figure 3.5. In the first plot, the number of classes k is held fixed at 6, and the values are plotted for different values of iterations T. In the second plot, the number of classes vary, and the num-

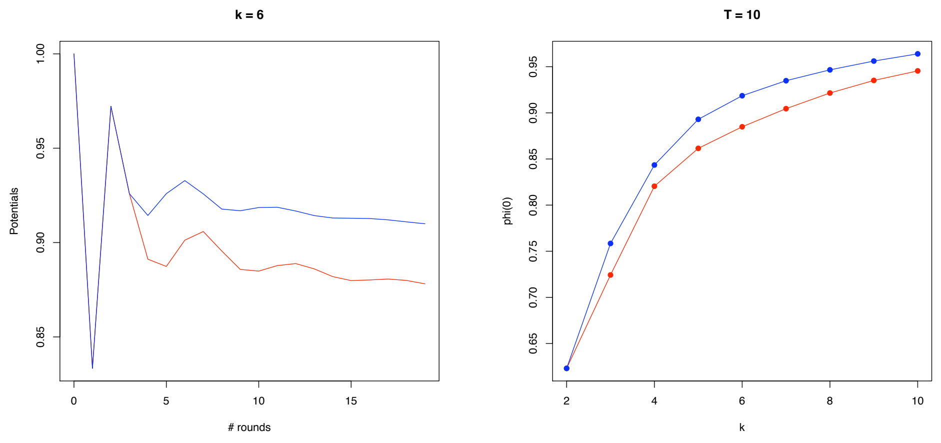


Figure 3.5: Comparison of $\phi_t(\mathbf{0})$ (blue) with $\max_{\mathbf{q}} \phi_t^{\mathbf{q}}(\mathbf{0})$ (red) over different rounds t and different number of classes k. We set $\gamma = 0$ in both.

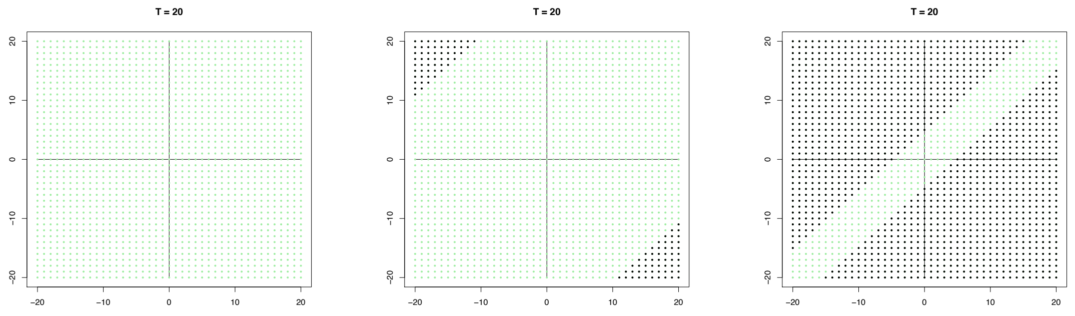


Figure 3.6: Green pixels have degree 3, black pixels have degree 2. Each step is diagonally down (left), and up (if $x < y$) and right (if $x > y$) and both when degree is 3. Each plot uses $T = 20, \gamma = 0.1$. The values of η are 0.08, 0.1 and 0.3, respectively. With smaller values of η, more pixels have degree 3.

ber of iterations is held at 10. Both plots show that the difference in the values is significant, and hence $\max_{\mathbf{b}} \phi_T^{\mathbf{b}}(\mathbf{0})$ would be a rather optimistic upper bound on the error when using the minimal weak learning condition.

If we use exponential loss (3.35), the situation is not much better. The degree maps for varying values of the weight parameter η against fixed values of edge $\gamma = 0.1$, rounds remaining $T = 20$ and number of classes $k = 3$ are plotted in Figure 3.6. Although the patterns are simple, with the degree 3 pixels forming a diagonal band, we found it hard to prove this fact formally, or compute the exact boundary of the band. However the plots suggest that when η is small, all pixels have degree 3. We next find conditions under which this opportunity for tractable computation exists.

Efficient computation in special cases. Here we show that when using the exponential loss, if the edge γ is very small, then the potentials can be computed efficiently. We first show an intermediate result. We already observed empirically that when the weight parameter η is small, the degrees all become equal to k. Indeed, we can prove this fact.

Lemma 3.19. *If the loss function being used is exponential loss (3.35) and the weight parameter η is small compared to the number of rounds*

$$\eta \leq \frac{1}{4} \min\left\{\frac{1}{k-1}, \frac{1}{T}\right\}, \tag{3.50}$$

then the optimal value of the degree a in (3.48) is always k. Therefore, in this situation, the potential ϕ_t using the minimal weak learning condition is the same as the potential $\phi_t^{\mathbf{u}}$ using the γ-biased uniform distribution $\mathbf{u}$,

$$\mathbf{u} = \left(\frac{1-\gamma}{k} + \gamma, \frac{1-\gamma}{k}, \ldots, \frac{1-\gamma}{k}\right), \tag{3.51}$$

and hence can be efficiently computed.

Proof. We show $\phi_t = \phi_t^{\mathbf{u}}$ by induction on the remaining number t of boosting iterations. The base case holds since, by definition, $\phi_0 = \phi_0^{\mathbf{u}} = L_\eta^{\exp}$. Assume, inductively that

$$\phi_{t-1}(\mathbf{s}) = \phi_{t-1}^{\mathbf{u}}(\mathbf{s}) = \kappa(\gamma,\eta)^{t-1} \sum_{l=2}^{k} e^{\eta(s_l - s_1)}, \tag{3.52}$$

where the second equality follows from (3.37). We show that

$$\phi_t(\mathbf{s}) = \mathbb{E}_{l\sim\mathbf{u}}\left[\phi_{t-1}(\mathbf{s}+\mathbf{e}_l)\right]. \tag{3.53}$$

By the inductive hypothesis and (3.30), the right hand side of (3.53) is in fact equal to $\phi_t^{\mathbf{u}}$, and we will have shown $\phi_t = \phi_t^{\mathbf{u}}$. The proof will then follow by induction.

In order to show (3.53), by Lemma 3.18, it suffices to show that the optimal degree a maximizing the right hand side of (3.48) is always k:

$$\mathbb{E}_{l\sim\mathbf{b}_a^\pi}\left[\phi_{t-1}\left(\mathbf{s}+\mathbf{e}_l\right)\right] \leq \mathbb{E}_{l\sim\mathbf{b}_k^\pi}\left[\phi_{t-1}\left(\mathbf{s}+\mathbf{e}_l\right)\right]. \tag{3.54}$$

By (3.52), $\phi_{t-1}\left(\mathbf{s}+\mathbf{e}_{l_0}\right)$ may be written as $\phi_{t-1}(\mathbf{s}) + \kappa(\gamma,\eta)^{t-1}\cdot\xi_{l_0}$, where the term ξ_{l_0} is:

$$\xi_{l_0} = \begin{cases} (e^\eta - 1)e^{\eta(s_{l_0}-s_1)} & \text{if } l_0 \neq 1, \\ (e^{-\eta}-1)\sum_{l=2}^k e^{\eta(s_l - s_1)} & \text{if } l_0 = 1. \end{cases}$$

Therefore (3.54) is the same as: $\mathbb{E}_{l\sim\mathbf{b}_a^\pi}\left[\xi_l\right] \leq \mathbb{E}_{l\sim\mathbf{b}_k^\pi}\left[\xi_l\right]$. Assume (by relabeling if necessary) that π is the identity permutation on coordinates $\{2,\ldots,k\}$. Then the expression $\mathbb{E}_{l\sim\mathbf{b}_a^\pi}\left[\xi_l\right]$ can be written as

$$\begin{aligned}
\mathbb{E}_{l\sim\mathbf{b}_a^\pi}\left[\xi_l\right] &= \left(\frac{1-\gamma}{a}+\gamma\right)\xi_1 + \sum_{l=k-a+2}^{k}\left(\frac{1-\gamma}{a}\right)\xi_l \\
&= \gamma\xi_1 + (1-\gamma)\left\{\frac{\xi_1+\sum_{l=k-a+2}^{k}\xi_l}{a}\right\}.
\end{aligned}$$

It suffices to show that the term in curly brackets is maximized when $a=k$. We will in fact show that the numerator of the term is negative if $a<k$, and non-negative for $a=k$, which will complete our proof. Notice that the numerator can be written as

$$\begin{aligned}
& (e^\eta-1)\left\{\sum_{l=k-a+2}^{k} e^{\eta(s_l-s_1)}\right\} - (1-e^{-\eta})\sum_{l=2}^{k} e^{\eta(s_l-s_1)} \\
= \; & (e^\eta-1)\left\{\sum_{l=k-a+2}^{k} e^{\eta(s_l-s_1)} - \sum_{l=2}^{k} e^{\eta(s_l-s_1)}\right\} + \left\{(e^\eta-1)-(1-e^{-\eta})\right\}\sum_{l=2}^{k} e^{\eta(s_l-s_1)} \\
= \; & \left\{e^\eta+e^{-\eta}-2\right\}\sum_{l=2}^{k} e^{\eta(s_l-s_1)} - (e^\eta-1)\left\{\sum_{l=2}^{k-a+1} e^{\eta(s_l-s_1)}\right\}.
\end{aligned}$$

When $a=k$, the second summation disappears, and we are left with a non-negative expression. However when $a<k$, the second summation is at least $e^{\eta(s_2-s_1)}$. Since $t\le T$, and in t iterations the absolute value of any state coordinate $|s_t(l)|$ is at most T, the first summation is at most $(k-1)e^{2\eta T}$ and the second summation is at least $e^{-2\eta T}$. Therefore the previous expression is at most

$$\begin{aligned}
& (k-1)\left(e^\eta+e^{-\eta}-2\right)e^{2\eta T} - (e^\eta-1)e^{-2\eta T} \\
= \; & (e^\eta-1)e^{-2\eta T}\left\{(k-1)(1-e^{-\eta})e^{4\eta T}-1\right\}.
\end{aligned}$$

We show that the term in curly brackets is negative. Firstly, using $e^x\ge 1+x$, we have $1-e^{-\eta}\le\eta\le 1/(4(k-1))$ by choice of η. Therefore it suffices to show that $e^{4\eta T}<4$. By choice of η again, $e^{4\eta T}\le e^1<4$. This completes our proof. □

The above lemma seems to suggest that under certain conditions, a sort of degeneracy occurs, and the optimal Booster payoff (3.18) is nearly unaffected by whether we use the minimal weak learning condition, or the condition $(\mathcal{C}^{\text{eor}},\mathbf{U}_\gamma)$. Indeed, we next prove this fact.

Theorem 3.20. *Suppose the loss function is as in Lemma 3.19, and for some parameter $\varepsilon>0$, the number of examples m is large enough*

$$m \ge \frac{Te^{1/4}}{\varepsilon}. \tag{3.55}$$

Consider the minimal weak learning condition (3.16), and the fixed edge-over-random condition $(\mathcal{C}^{eor}, \mathbf{U}_\gamma)$ corresponding to the γ-biased uniform baseline $\mathbf{U}_\gamma$. Then the optimal booster payoffs using either condition is within ε of each other.

Proof. We show that the OS strategies arising out of either condition is the same. In other words, at any iteration t and state $\mathbf{s}_t$, both strategies play the same cost matrix and enforce the same constraints on the response of Weak-Learner. The theorem will then follow if we can invoke Theorems 3.9 and 3.16. For that, the number of examples needs to be as large as in (3.28). The required largeness would follow from (3.55) if the loss function satisfied (3.27) with $\varnothing(L,T)$ at most $\exp(1/4)$. Since the largest discrepancy in losses between two states reachable in T iterations is at most $e^{\eta T} - 0$, the bound follows from the choice of η in (3.50). Therefore, it suffices to show the equivalence of the OS strategies corresponding to the two weak learning conditions.

We first show both strategies play the same cost-matrix. Lemma 3.19 states that the potential function using the minimal weak learning condition is the same as when using the fixed condition $(\mathcal{C}^{\text{eor}}, \mathbf{U}_\gamma)$: $\phi_t = \phi_t^{\mathbf{u}}$, where $\mathbf{u}$ is as in (3.51). Since, according to (3.31) and (3.45), given a state $\mathbf{s}_t$ and iteration t, the two strategies choose cost matrices that are identical functions of the respective potentials, by the equivalence of the potential functions, the resulting cost matrices must be the same.

Even with the same cost matrix, the two different conditions could be imposing different constraints on Weak-Learner, which might affect the final payoff. For instance, with the baseline $\mathbf{U}_\gamma$, Weak-Learner has to return a weak classifier h satisfying

$$\mathbf{C}_t \bullet \mathbf{1}_h \leq \mathbf{C}_t \bullet \mathbf{U}_\gamma,$$

whereas with the minimal condition, the constraint on h is

$$\mathbf{C}_t \bullet \mathbf{1}_h \leq \max_{\mathbf{B}\in\mathcal{B}_\gamma^{\text{eor}}} \mathbf{C}_t \bullet \mathbf{B}.$$

In order to show that the constraints are the same we therefore need to show that for the common cost matrix $\mathbf{C}_t$ chosen, the right hand side of the two previous expressions are the same:

$$\mathbf{C}_t \bullet \mathbf{U}_\gamma = \max_{\mathbf{B}\in\mathcal{B}_\gamma^{\text{eor}}} \mathbf{C}_t \bullet \mathcal{B}_\gamma^{\text{eor}}. \tag{3.56}$$

We will in fact show the stronger fact that the equality holds for every row separately:

$$\forall i : \langle \mathbf{C}_t(i), \mathbf{u} \rangle = \max_{\mathbf{b}\in\Delta_\gamma^k} \langle \mathbf{C}_t(i), \mathbf{b} \rangle\,. \tag{3.57}$$

To see this, first observe that the choice of the optimal cost matrix $\mathbf{C}_t$ in (3.45) implies the following identity

$$\langle \mathbf{C}_t(i), \mathbf{b} \rangle = \mathbb{E}_{l\sim\mathbf{b}}\left[\phi_{T-t-1}(\mathbf{s}_t(i) + \mathbf{e}_l)\right].$$

On the other hand, (3.48) and Lemma 3.19 together imply that the distribution $\mathbf{b}$ maximizing the right hand side of the above is the γ-biased uniform distribution, from which (3.57) follows. Therefore, the constraints placed on Weak-Learner by the cost-matrix $\mathbf{C}_t$ is the same whether we use minimum weak learning condition or the fixed condition $(\mathcal{C}^{\text{eor}}, \mathbf{U}_\gamma)$. $\square$

One may wonder why η would be chosen so small, especially since the previous theorem indicates that such choices for η lead to degeneracies. To understand this, recall that η represents the size of the weights α_t chosen in every round, and was introduced as a tunable parameter to help achieve the best possible upper bound on zero-one error. More precisely, recall that the exponential loss $L_\eta^{\text{exp}}(\mathbf{s})$ of the unweighted state, defined in (3.35), is equal to the exponential loss $L^{\text{exp}}(\mathbf{f})$ on the weighted state, defined in (3.36), which in turn is an upper bound on the error $L^{\text{err}}(\mathbf{f}_T)$ of the final weighted state $\mathbf{f}_T$. Therefore the potential value $\phi_T(\mathbf{0})$ based on the exponential loss L_η^{exp} is an upper bound on the minimum error attainable after T rounds of boosting. At the same time, $\phi_T(\mathbf{0})$ is a function of η. Therefore, we may tune this parameter to attain the best bound possible. Even with this motivation, it may seem that a properly tuned η will not be as small as in Lemma 3.19, especially since it can be shown that the resulting loss bound $\phi_T(\mathbf{0})$ will always be larger than a fixed constant (depending on γ, k), no matter how many rounds T of boosting is used. However, the next result identifies conditions under which the tuned value of η is indeed as small as in Lemma 3.19. This happens when the edge γ is very small, as is described in the next theorem. Intuitively, a weak classifier achieving small edge has low accuracy, and a low weight reflects Booster's lack of confidence in this classifier.

Theorem 3.21. *When using the exponential loss function (3.35), and the minimal weak learning condition (3.16), the loss upper bound $\phi_T(\mathbf{0})$ provided by Theorem 3.16 is more than 1 and hence trivial unless the parameter η is chosen sufficiently small compared to the edge γ:*

$$\eta \leq \frac{k\gamma}{1-\gamma}. \tag{3.58}$$

In particular, when the edge is very small:

$$\gamma \leq \min\left\{\frac{1}{2}, \frac{1}{8k}\min\left\{\frac{1}{k}, \frac{1}{T}\right\}\right\}, \tag{3.59}$$

the value of η needs to be as small as in (3.50).

Proof. Comparing solutions (3.49) and (3.34) to the potentials corresponding to the minimal weak learning condition and a fixed edge-over-random condition, we may conclude that the loss bound $\phi_T(\mathbf{0})$ is in the former case is larger than $\phi_T^{\mathbf{b}}(\mathbf{0})$, for any edge-over-random distribution $\mathbf{b} \in \Delta_\gamma^k$. In particular, when $\mathbf{b}$ is set to be the γ-biased uniform distribution $\mathbf{u}$, as defined in (3.51), we get $\phi_T(\mathbf{0}) \geq \phi_T^{\mathbf{u}}(\mathbf{0})$. Now the latter bound, according to (3.37), is $\kappa(\gamma, \eta)^T$, where κ is defined as in (3.38). Therefore, to get non-trivial loss bounds which are at most 1, we need to choose η such that

$\kappa(\gamma, \eta) \leq 1$. By (3.38), this happens when

$$\begin{aligned}
\left(1 - e^{-\eta}\right)\gamma &\geq \left(e^{\eta} + e^{-\eta} - 2\right)\left(\frac{1-\gamma}{k}\right) \\
\text{i.e., } \frac{k\gamma}{1-\gamma} &\geq \frac{e^{\eta} + e^{-\eta} - 2}{1 - e^{-\eta}} = e^{\eta} - 1 \geq \eta.
\end{aligned}$$

Therefore (3.58) holds. When γ is as small as in (3.59), then $1-\gamma \leq \frac{1}{2}$, and therefore, by (3.58), the bound on η becomes identical to that in (3.59). □

The condition in the previous theorem, that of the existence of only a very small edge, is the most we can assume for most practical datasets. Therefore, in such situations, we can compute the optimal Booster strategy that uses the minimal weak learning conditions. More importantly, using this result, we derive, in the next section, a highly efficient and practical *adaptive* algorithm, that is, one that does not require any prior knowledge about the edge γ, and will therefore work with any dataset.

3.8 Variable edges

So far we have required Weak-Learner to beat random by at least a fixed amount $\gamma > 0$ in each round of the boosting game. In reality, the edge over random is larger initially, and gets smaller as the OS algorithm creates harder cost matrices. Therefore requiring a fixed edge is either unduly pessimistic or overly optimistic. If the fixed edge is too small, not enough progress is made in the initial rounds, and if the edge is too large, Weak-Learner fails to meet the weak-learning condition in latter rounds. We fix this by not making any assumption about the edges, but instead *adaptively* responding to the edges returned by Weak-Learner. In the rest of the section we describe the adaptive procedure, and the resulting loss bounds guaranteed by it.

The philosophy behind the adaptive algorithm is a boosting game where Booster and Weak Learner no longer have opposite goals, but cooperate to reduce error as fast as possible. However, in order to create a clean abstraction and separate implementations of the boosting algorithms and the weak learning procedures as much as possible, we assume neither of the players has any knowledge of the details of the algorithm employed by the other player. In particular Booster may only assume that Weak Learner's strategy is barely strong enough to guarantee boosting. Therefore, Booster's demands on the weak classifiers returned by Weak Learner should be minimal, and it should send the weak learning algorithm the "easiest" cost matrices that will ensure boostability. In turn, Weak Learner may only assume a very weak Booster strategy, and therefore return a weak classifier that performs as well as possible with respect to the cost matrix sent by Booster.

At a high level, the adaptive strategy proceeds as follows. At any iteration, based on the states of the examples and number of remaining rounds of boosting, Booster chooses the game-theoretically optimal cost matrix assuming only infinitesimal edges in the remaining rounds. Intuitively, Booster has no high expectations of Weak Learner, and supplies it the easiest cost matrices with which it may be able to boost.

However, in the adaptive setting, Weak-Learner is no longer adversarial. Therefore, although only infinitesimal edges are anticipated by Booster, Weak Learner cooperates in returning weak classifiers that achieve as large edges as possible, which will be more than just inifinitesimal. Based on the exact edge received in each round, Booster chooses the weight α_t adaptively to reach the most favourable state possible. Therefore, Booster plays game theoretically assuming an adversarial Weak Learner and expecting only the smallest edges in the future rounds, although Weak Learner actually cooperates, and Booster adaptively exploits this favorable behavior as much as possible. This way the boosting algorithm remains robust to a poorly performing Weak Learner, and yet can make use of a powerful weak learning algorithm whenever possible.

We next describe the details of the adaptive procedure. With variable weights we need to work with the weighted state $\mathbf{f}_t(i)$ of each example i, defined in (3.22). To keep the computations tractable, we will only be working with the exponential loss $L^{\exp}(\mathbf{f})$ on the weighted states. We first describe how Booster chooses the cost-matrix in each round. Following that we describe how it adaptively computes the weights in each round based on the edge of the weak classifier received.

Choosing the cost-matrix. As discussed before, at any iteration t and state $\mathbf{f}_t$ Booster assumes that it will receive an infinitesimal edge γ in each of the remaining rounds. Since the step size is a function of the edge, which in turn is expected to be the same tiny value in each round, we may assume that the step size in each round will also be some fixed value η. We are therefore in the setting of Theorem 3.21, which states that the parameter η in the exponential loss function (3.35) should also be tiny to get any non-trivial bound. But then the loss function satisfies the conditions in Lemma 3.19, and by Theorem 3.20, the game theoretically optimal strategy remains the same whether we use the minimal condition or $(\mathcal{C}^{\text{eor}}, \mathbf{U}_\gamma)$. When using the latter condition, the optimal choice of the cost-matrix at iteration t and state $\mathbf{f}_t$, according to (3.39), is

$$C_t(i,l) = \begin{cases} (e^{\eta} - 1)\, e^{f_{t-1}(i,j) - f_{t-1}(i,1)} & \text{if } l > 1, \\ (e^{-\eta} - 1) \sum_{j=2}^{k} e^{f_{t-1}(i,j) - f_{t-1}(i,1)} & \text{if } l = 1. \end{cases} \tag{3.60}$$

Further, when using the condition $(\mathcal{C}^{\text{eor}}, \mathbf{U}_\gamma)$, the average potential of the states $\mathbf{f}_t(i)$, according to (3.37), is given by the average loss (3.40) of the state times $\kappa(\gamma,\eta)^{T-t}$, where the function κ is defined in (3.38). Our goal is to choose η as a function of γ so that $\kappa(\gamma,\eta)$ is as small as possible. Now, there is no lower bound on how small the edge γ may get, and, anticipating the worst, it makes sense to choose an infinitesimal γ, in the spirit of [19]. Eq. (3.38) then implies that the choice of η should also be infinitesimal. Then the above choice of the cost matrix becomes the following (after some rescaling):

$$\begin{aligned}
C_t(i,l) &= \lim_{\eta\to 0} C_\eta(i,l) \triangleq \frac{1}{\eta}\begin{cases}(e^{\eta}-1)\,e^{f_{t-1}(i,j)-f_{t-1}(i,1)} & \text{if } l>1,\\ (e^{-\eta}-1)\sum_{j=2}^{k} e^{f_{t-1}(i,j)-f_{t-1}(i,1)} & \text{if } l=1.\end{cases}\\
&= \begin{cases}e^{f_{t-1}(i,j)-f_{t-1}(i,1)} & \text{if } l>1,\\ -\sum_{j=2}^{k} e^{f_{t-1}(i,j)-f_{t-1}(i,1)} & \text{if } l=1.\end{cases} \qquad (3.61)
\end{aligned}$$

We have therefore derived the optimal cost matrix played by the adaptive boosting strategy, and we record this fact.

Lemma 3.22. *Consider the boosting game using the minimal weak learning condition* (3.16)*. Then, in iteration t at state $\mathbf{f}_t$, the game-theoretically optimal Booster strategy chooses the cost matrix $\mathbf{C}_t$ given in* (3.61)*.*

We next show how to adaptively choose the weights α_t.

Adaptively choosing weights. Once Weak Learner returns a weak classifier h_t, Booster chooses the optimum weight α_t so that the resulting states $\mathbf{f}_t = \mathbf{f}_{t-1} + \alpha_t \mathbf{1}_{ht}$ are as favorable as possible, that is, minimizes the total potential of its states. By our previous discussions, these are proportional to the total loss given by $Z_t = \sum_{i=1}^{m}\sum_{l=2}^{k} e^{f_t(i,l)-f_t(i,1)}$. For any choice of α_t, the difference $Z_t - Z_{t-1}$ between the total loss in rounds $t-1$ and t is given by

$$\begin{aligned}
&(e^{\alpha_t}-1)\sum_{i\in S_-} e^{f_{t-1}(i,h_t(i))-f_{t-1}(i,1)} - \left(1-e^{-\alpha_t}\right)\sum_{i\in S_+} L^{\exp}(\mathbf{f}_{t-1}(i))\\
=\;& (e^{\alpha_t}-1)\,A_-^t - \left(1-e^{-\alpha_t}\right)A_+^t\\
=\;& \left(A_+^t e^{-\alpha_t} + A_-^t e^{\alpha_t}\right) - \left(A_+^t + A_-^t\right),
\end{aligned}$$

where S_+ denotes the set of examples that h_t classifies correctly, S_- the incorrectly classified examples, and A_-^t, A_+^t denote the first and second summations, respectively. Therefore, the task of choosing α_t can be cast as a simple optimization problem minimizing the previous expression. In fact, the optimal value of α_t is given by the following closed form expression

$$\alpha_t = \frac{1}{2}\ln\left(\frac{A_+^t}{A_-^t}\right). \qquad (3.62)$$

With this choice of weight, one can show (with some straightforward algebra) that the total loss of the state falls by a factor less than 1. In fact the factor is exactly

$$(1-c_t) - \sqrt{c_t^2 - \delta_t^2}, \qquad (3.63)$$

where

$$c_t = (A_+^t + A_-^t)/Z_{t-1}, \qquad (3.64)$$

and δ_t is the edge of the returned classifier h_t on the supplied cost-matrix $\mathbf{C}_t$. Notice that the quantity c_t is at most 1, and hence the factor (3.63) can be upper bounded by $\sqrt{1-\delta_t^2}$. We next show how to compute the edge δ_t. The definition of the edge depends on the weak learning condition being used, and in this case we are using the minimal condition (3.16). Therefore the edge δ_t is the largest γ such that the following still holds

$$\mathbf{C}_t \bullet \mathbf{1}_h \leq \max_{\mathbf{B}\in\mathcal{B}_\gamma^{\mathrm{eor}}} \mathbf{C}_t \bullet \mathbf{B}.$$

However, since $\mathbf{C}_t$ is the optimal cost matrix when using exponential loss with a tiny value of η, we can use arguments in the proof of Theorem 3.20 to simplify the computation. In particular, eq. (3.56) implies that the edge δ_t may be computed as the largest γ satisfying the following simpler inequality

$$\begin{aligned}
\delta_t &= \sup\left\{\gamma : \mathbf{C}_t \bullet \mathbf{1}_{h_t} \leq \mathbf{C}_t \bullet \mathbf{U}_\gamma\right\}\\
&= \sup\left\{\gamma : \mathbf{C}_t \bullet \mathbf{1}_{h_t} \leq -\gamma \sum_{i=1}^m \sum_{l=2}^k e^{f_{t-1}(i,l)-f_{t-1}(i,1)}\right\}\\
\implies \delta_t &= \gamma : \ \mathbf{C}_t \bullet \mathbf{1}_{h_t} = -\gamma \sum_{i=1}^m \sum_{l=2}^k e^{f_{t-1}(i,l)-f_{t-1}(i,1)}\\
\implies \delta_t &= \frac{-\mathbf{C}_t \bullet \mathbf{1}_{h_t}}{\sum_{i=1}^m \sum_{l=2}^k e^{f_{t-1}(i,l)-f_{t-1}(i,1)}} = \frac{-\mathbf{C}_t \bullet \mathbf{1}_{h_t}}{Z_t}, \qquad (3.65)
\end{aligned}$$

where the first step follows by expanding $\mathbf{C}_t \bullet \mathbf{U}_\gamma$. We have therefore an adaptive strategy which efficiently reduces error. We record our results.

Lemma 3.23. *If the weight α_t in each round is chosen as in (3.62), and the edge δ_t is given by (3.65), then the total loss Z_t falls by the factor given in (3.63), which is at most $\sqrt{1-\delta_t^2}$.*

The choice of α_t in (3.62) is optimal, but depends on quantities other than just the edge δ_t. We next show a way of choosing α_t based only on δ_t that still causes the total loss to drop by a factor of $\sqrt{1-\delta_t^2}$.

Lemma 3.24. *Suppose cost matrix $\mathbf{C}_t$ is chosen as in (3.61), and the returned weak classifier h_t has edge δ_t i.e. $\mathbf{C}_t \bullet \mathbf{1}_{h_t} \leq \mathbf{C}_t \bullet \mathbf{U}_{\delta_t}$. Then choosing any weight $\alpha_t > 0$ for h_t makes the loss Z_t at most a factor*

$$1 - \frac{1}{2}(e^{\alpha_t} - e^{-\alpha_t})\delta_t + \frac{1}{2}(e^{\alpha_t} + e^{-\alpha_t} - 2)$$

of the previous loss Z_{t-1}. In particular by choosing

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1+\delta_t}{1-\delta_t}\right), \qquad (3.66)$$

the drop factor is at most $\sqrt{1-\delta_t^2}$.

Proof. We borrow notation from earlier discussions. The edge-condition implies

$$A_-^t - A_+^t = \mathbf{C}_t \bullet \mathbf{1}_{h_t} \le \mathbf{C}_t \bullet \mathbf{U}_{\delta_t} = -\delta_t Z_{t-1} \implies A_+^t - A_-^t \ge \delta_t Z_{t-1}.$$

On the other hand, the drop in loss after choosing h_t with weight α_t is

$$\begin{aligned} & \left(1 - e^{-\alpha_t}\right) A_+^t - \left(e^{\alpha_t} - 1\right) A_-^t \\ = & \left(\frac{e^{\alpha_t} - e^{-\alpha_t}}{2}\right) \left(A_+^t - A_-^t\right) - \left(\frac{e^{\alpha_t} + e^{-\alpha_t} - 2}{2}\right) \left(A_+^t + A_-^t\right). \end{aligned}$$

We have already shown that $A_+^t - A_-^t \ge \delta_t Z_{t-1}$. Further, $A_+^t + A_-^t$ is at most Z_{t-1}. Therefore the loss drops by a factor of at least

$$1 - \frac{1}{2}(e^{\alpha_t} - e^{-\alpha_t})\delta_t + \frac{1}{2}(e^{\alpha_t} + e^{-\alpha_t} - 2) = \frac{1}{2}\left\{(1-\delta_t)e^{\alpha_t} + (1+\delta_t)e^{-\alpha_t}\right\}.$$

Tuning α_t as in (3.66) causes the drop factor to be at least $\sqrt{1-\delta_t^2}$. □

Algorithm 1 contains pseudocode for the adaptive algorithm, and includes both ways of choosing α_t. We call both versions of this algorithm AdaBoost.MM. With the approximate way of choosing the step length in (3.67), AdaBoost.MM turns out to be identical to AdaBoost.M2 [23] or AdaBoost.MR [47], provided the weak classifier space is transformed in an appropriate way to be acceptable by AdaBoost.M2 or AdaBoost.MR. We emphasize that AdaBoost.MM and AdaBoost.M2 are products of very different theoretical considerations, and this similarity should be viewed as a coincidence arising because of the particular choice of loss function, infinitesimal edge and approximate step size. For instance, when the step sizes are chosen instead as in (3.68), the training error falls more rapidly, and the resulting algorithm is different.

As a summary of all the discussions in the section, we record the following theorem.

Theorem 3.25. *The boosting algorithm AdaBoost.MM, shown in Algorithm 1, is the optimal strategy for playing the adaptive boosting game, and is based on the minimal weak learning condition. Further if the edges returned in each round are $\delta_1, \ldots, \delta_T$, then the error after T rounds is $(k-1)\prod_{t=1}^{T}\sqrt{1-\delta_t^2} \le (k-1)\exp\left\{-(1/2)\sum_{t=1}^{T}\delta_t^2\right\}$.*

In particular, if a weak hypothesis space is used that satisfies the optimal weak learning condition (3.16), for some γ, then the edge in each round is large, $\delta_t \ge \gamma$, and therefore the error after T rounds is exponentially small, $(k-1)e^{-T\gamma^2/2}$.

The theorem above states that as long as the minimal weak learning condition is satisfied, the error will decrease exponentially fast. Even if the condition is not satisfied, the error rate will keep falling rapidly provided the edges achieved by the weak classifiers are relatively high. However, our theory so far can provide no guarantees on these edges, and therefore it is not clear what is the best error rate achievable in this case, and how quickly it is achieved. The assumptions of boostability, and hence our minimal weak learning condition does not hold for the vast majority of practical datasets, and as such it is important to know what happens in such settings. In particular, an important requirement is *empirical consistency*, where we want that for

Algorithm 1 AdaBoost.MM

```
Require: Number of classes $k$, number of examples $m$.
Require: Training set $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ with $y_i \in \{1, \ldots, k\}$ and $x_i \in X$.
  • Initialize $m \times k$ matrix $f_0(i, l) = 0$ for $i = 1, \ldots, m$, and $l = 1, \ldots, k$.
  for $t = 1$ to $T$ do
    • Choose cost matrix $\mathbf{C}_t$ as follows:
        $C_t(i,l) = \begin{cases} e^{f_{t-1}(i,l) - f_{t-1}(i,y_i)} & \text{if } l \neq y_i, \\ -\sum_{l \neq y_i} e^{f_{t-1}(i,j) - f_{t-1}(i,y_i)} & \text{if } l = 1. \end{cases}$
    • Receive weak classifier $h_t : X \to \{1, \ldots, k\}$ from weak learning algorithm
    • Compute edge $\delta_t$ as follows:
        $\delta_t = \frac{-\sum_{i=1}^{m} C_t(i, h_t(x_i))}{\sum_{i=1}^{m} \sum_{l \neq y_i} e^{f_{t-1}(i,l) - f_{t-1}(i,y_i)}}$
    • Choose $\alpha_t$ either as
        $\alpha_t = \frac{1}{2} \ln \left( \frac{1 + \delta_t}{1 - \delta_t} \right),$   (3.67)
      or, for a slightly bigger drop in the loss, as
        $\alpha_t = \frac{1}{2} \ln \left( \frac{\sum_{i: h_t(x_i) = y_i} \sum_{l \neq y_i} e^{f_{t-1}(i,l) - f_{t-1}(i,y_i)}}{\sum_{i: h_t(x_i) \neq y_i} e^{f_{t-1}(i, h_t(x_i)) - f_{t-1}(i,y_i)}} \right)$   (3.68)
    • Compute $\mathbf{f}_t$ as:
        $f_t(i,l) = f_{t-1}(i,l) + \alpha_t \mathbf{1}[h_t(x_i) = l].$
  end for
  • Output weighted combination of weak classifiers $F_T : X \times \{1, \ldots, k\} \to \mathbb{R}$ defined as:
        $F_T(x, l) = \sum_{t=1}^{T} \alpha_t \mathbf{1}[h_t(x) = l].$   (3.69)
  • Based on $F_T$, output a classifier $H_T : X \to \{1, \ldots, k\}$ that predicts as
        $H_T(x) = \operatorname{argmax}_{l=1}^{k} F_T(x, l).$   (3.70)
```

any given weak classifier space, the algorithm converge, if allowed to run forever, to the weighted combination of classifiers that minimizes error on the training set. Another important criterion is *universal consistency*, which requires that the algorithm converge, when provided sufficient training data, to the classifier combination that minimizes error on the test dataset. In the next section, we show that AdaBoost.MM satisfies such consistency requirements. Both the choice of the minimal weak learning condition as well as the setup of the adaptive game framework will play crucial roles in ensuring consistency. These results therefore provide evidence that game theoretic considerations can have strong statistical implications.

3.9 Consistency of the adaptive algorithm

The goal in a classification task is to design a classifier that predicts with high accuracy on unobserved or test data. This is usually carried out by ensuring the classifier fits training data well without being overly complex. Assuming the training and test data are reasonably similar, one can show that the above procedure achieves high test accuracy, or is consistent. Here we work in a probabilistic setting that connects training and test data by assuming both consist of examples and labels drawn from a common, unknown distribution.

Consistency for multiclass classification in the probabilistic setting has been studied by Tewari and Bartlett [53], who show that, unlike in the binary setting, many natural approaches fail to achieve consistency. In this section, we show that AdaBoost.MM described in the previous section avoids such pitfalls and enjoys various consistency results. We begin by laying down some standard assumptions and setting up some notation. Then we prove our first result showing that our algorithm minimizes a certain exponential loss function on the training data at a fast rate. Next, we build upon this result and improve along two fronts: firstly we change our metric from exponential loss to the more relevant classification error metric, and secondly we show fast convergence on not just training data, but also the test set. For the proofs, we heavily reuse existing machinery in the literature.

Throughout the rest of this section we consider the version of AdaBoost.MM that picks weights according to the approximate rule in (3.67). All our results most probably hold with the other rule for picking weights in (3.68) as well, but we did not verify that. These results hold without any boostability requirements on the space $\mathcal{H}$ of weak classifiers, and are therefore widely applicable in practice. While we do not assume any weak learning condition, we will require a fully cooperating Weak Learner. In particular, we will require that in each round Weak Learner picks the weak classifier suffering minimum cost with respect to the cost matrix provided by the boosting algorithm, or equivalently achieves the highest edge as defined in (3.65). Such assumptions are both necessary and standard in the literature, and are frequently met in practice.

In order to state our results, we will need to setup some notation. The space of examples will be denoted by $\mathcal{X}$, and the set of labels by $\mathcal{Y} = \{1, \ldots, k\}$. We also fix a finite weak classifier space $\mathcal{H}$ consisting of classifiers $h : \mathcal{X} \to \mathcal{Y}$. We will be interested in functions $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that assign a score to every example and label pair. Important examples of such functions are the weighted majority combinations (3.69) output by the adaptive algorithm. In general, any such combination of the weak classifiers in space $\mathcal{H}$ is specified by some weight function $\alpha : \mathcal{H} \to \mathbb{R}$; the resulting function is denoted by $F_\alpha : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, and satisfies:

$$F_\alpha(x, l) = \sum_{h \in \mathcal{H}} \alpha(h) \mathbf{1}\left[h(x) = l\right].$$

We will be interested in measuring the average exponential loss of such functions. To measure this, we introduce the $\widehat{\text{risk}}$ operator:

$$\widehat{\text{risk}}(F) \triangleq \frac{1}{m} \sum_{i=1}^{m} \sum_{l \neq y_i} e^{F(x_i,l) - F(x_i,y_i)}. \tag{3.71}$$

With this setup, we can now state our simplest consistency result, which ensures that the algorithm converges to a weighted combination of classifiers in the space $\mathcal{H}$ that achieves the minimum exponential loss over the training set at an efficient rate.

Lemma 3.26. *The $\widehat{\text{risk}}$ of the predictions F_T, as defined in (3.69), converges to that of the optimal predictions of any combination of the weak classifiers in $\mathcal{H}$ at the rate $O(1/T)$:*

$$\widehat{\text{risk}}(F_T) - \inf_{\alpha : \mathcal{H} \to \mathbb{R}} \widehat{\text{risk}}(F_\alpha) \leq \frac{C}{T}, \tag{3.72}$$

where C is a constant depending only on the dataset.

A slightly stronger result would state that the average exponential loss when measured with respect to the *test set*, and not just the empirical set, also converges. The test set is generated by some target distribution D over example label pairs, and we introduce the risk_D operator to measure the exponential loss for any function $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ with respect to D:

$$\text{risk}_D(F) = \mathbb{E}_{(x,y)\sim D}\left[\sum_{l \neq y} e^{F(x,l) - F(x,y)}\right].$$

We show this stronger result holds if the function F_T is modified to the function $\bar{F}_T : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that takes values in the range $[0, -C]$, for some large constant C:

$$\bar{F}_T(x, l) \triangleq \max\left\{-C, F_T(x, l) - \max_{l'} F_T(x, l')\right\}. \tag{3.73}$$

Lemma 3.27. *If $\bar{F}_T$ is as in (3.73), and the number of rounds T is set to $T_m = \sqrt{m}$, then its risk_D converges to the optimal value as $m \to \infty$ with high probability:*

$$\Pr\left[\text{risk}_D\left(\bar{F}_{T_m}\right) \leq \inf_{F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}} \text{risk}_D(F) + O\left(m^{-c}\right)\right] \geq 1 - \frac{1}{m^2}, \tag{3.74}$$

where $c > 0$ is some absolute constant, and the probability is over the draw of training examples.

We prove Lemmas 3.26 and 3.27 by demonstrating a strong correspondence between AdaBoost.MM and binary AdaBoost, and then leveraging almost identical known consistency results for AdaBoost [4]. Our proofs will closely follow the exposition in Chapter 12 of [46] on the consistency of AdaBoost, and are deferred to the appendix.

So far we have focused on risk_D, but a more desirable consistency result would state that the test *error* of the final classifier output by AdaBoost.MM converges to the Bayes optimal error. The test error is measured by the err_D operator, and is given by

$$\mathrm{err}_D(H) = \Pr_{(x,y)\sim D}\left[H(x) \neq y\right]. \tag{3.75}$$

The Bayes optimal classifier H_{opt} is a classifier achieving the minimum error among all possible classifying functions

$$\mathrm{err}_D(H_{\mathrm{opt}}) = \inf_{H:\mathcal{X}\to\mathcal{Y}} \mathrm{err}_D(H), \tag{3.76}$$

and we want our algorithm to output a classifier whose err_D approaches $\mathrm{err}_D(H_{\mathrm{opt}})$. In designing the algorithm, our main focus was on reducing the exponential loss, captured by risk_D and $\widehat{\mathrm{risk}}$. Unless these loss functions are aligned properly with classification error, we cannot hope to achieve optimal error. The next result shows that our loss functions are correctly aligned, or more technically *Bayes consistent*. In other words, if a scoring function $F : \mathcal{X}\times\mathcal{Y} \to \mathbb{R}$ is close to achieving optimal risk_D, then the classifier $H : \mathcal{X} \to \mathcal{Y}$ derived from it as follows:

$$H(x) \in \operatorname*{argmax}_{l\in\mathcal{Y}} F(x, y), \tag{3.77}$$

also approaches the Bayes optimal error.

Lemma 3.28. *Suppose F is a scoring function achieving close to optimal risk*

$$\mathrm{risk}_D(F) \leq \inf_{F':\mathcal{X}\times\mathcal{Y}\to\mathbb{R}} \mathrm{risk}_D(F') + \varepsilon, \tag{3.78}$$

for some $\varepsilon \geq 0$. If H is the classifier derived from it as in (3.77), then it achieves close to the Bayes optimal error

$$\mathrm{err}_D(H) \leq \mathrm{err}_D(H_{\mathrm{opt}}) + \sqrt{2\varepsilon}. \tag{3.79}$$

Proof. The proof is similar to that of Theorem 12.1 in [46], which in turn is based on the work by Zhang [55] and Bartlett et al [5]. Let $p(x) = \Pr_{(x',y')\sim D}(x' = x)$ denote the the marginalized probability of drawing example x from D, and let $p_y^x = \Pr_{(x',y')\sim D}[y' = y|x' = x]$ denote the conditional probability of drawing label y given we have drawn example x. We first rewrite the difference in errors between H and H_{opt} using these probabilities. Firstly note that the accuracy of any classifier H' is given by

$$\sum_{x\in\mathcal{X}} D(x, H'(x)) = \sum_{x\in\mathcal{X}} p(x) p^x_{H'(x)}.$$

If $\mathcal{X}'$ is the set of examples where the predictions of H and H_{opt} differ, $\mathcal{X}' = \{x \in \mathcal{X} : H(x) \neq H_{\mathrm{opt}}(x)\}$, then we may bound the error differences as

$$\mathrm{err}_D(H) - \mathrm{err}_D(H_{\mathrm{opt}}) = \sum_{x\in\mathcal{X}'} p(x)\left(p^x_{H_{\mathrm{opt}}(x)} - p^x_{H(x)}\right). \tag{3.80}$$

We next relate this expression to the difference of the losses.

Notice that for any scoring function F', the risk_D can be rewritten as follows :

$$\mathrm{risk}_D(F') = \sum_{x\in\mathcal{X}} p(x) \sum_{l<l'} \left\{ p^x_l e^{F'(x,l')-F'(x,l)} + p^x_{l'} e^{F'(x,l)-F'(x,l')} \right\}.$$

Denote the inner summation in curly brackets by $L^{l,l'}_{F'}(x)$, and notice this quantity is minimized if

$$e^{F'(x,l)-F'(x,l')} = \sqrt{p^x_l / p^x_{l'}}, \quad \text{i.e., if } F'(x,l) - F'(x,l') = \tfrac{1}{2}\ln p^x_l - \tfrac{1}{2}\ln p^x_{l'}.$$

Therefore, defining $F^*(x,l) = \frac{1}{2}\ln p^x_l$ leads to a risk_D minimizing function F^*. Furthermore, for any example and pair of labels l, l', the quantity $L^{l,l'}_{F^*}(x)$ is at most $L^{l,l'}_{F}(x)$, and therefore the difference in losses of F^* and F may be lower bounded as follows:

$$\begin{aligned}
\varepsilon \geq \mathrm{risk}_D(F) - \mathrm{risk}_D(F^*) &= \sum_{x\in\mathcal{X}} p(x) \sum_{l\neq l'} \left(L^{l,l'}_F - L^{l,l'}_{F^*} \right) \\
&\geq \sum_{x\in\mathcal{X}'} p(x) \left\{ L^{H(x),H_{\mathrm{opt}}(x)}_F - L^{H(x),H_{\mathrm{opt}}(x)}_{F^*} \right\}. \qquad (3.81)
\end{aligned}$$

We next study the term in the curly brackets for a fixed x. Let A and B denote $H(x)$ and $H_{\mathrm{opt}}(x)$, respectively. We have already seen that $L^{A,B}_{F^*} = 2\sqrt{p^x_A p^x_B}$. Further, by definition of Bayes optimality, $p^x_A \geq p^x_B$. On the other hand, since $x \in \mathcal{X}'$, we know that $B \neq A$, and hence, $F(x,A) \geq F(x,B)$. Let $e^{F(x,B)-F(x,A)} = 1 + \eta$, for some $\eta \geq 0$. The quantity $L^{A,B}_F$ may be lower bounded as:

$$\begin{aligned}
L^{A,B}_F &= p^x_A e^{F(x,B)-F(x,A)} + p^x_B e^{F(x,A)-F(x,B)} \\
&= (1+\eta)p^x_A + (1+\eta)^{-1} p^x_B \\
&\geq (1+\eta)p^x_A + (1-\eta)p^x_B \\
&= p^x_A + p^x_B + \eta(p^x_A - p^x_B) \geq p^x_A + p^x_B.
\end{aligned}$$

Combining we get

$$L^{A,B}_F - L^{A,B}_{F^*} \geq p^x_A + p^x_B - 2\sqrt{p^x_A p^x_B} = \left(\sqrt{p^x_A} - \sqrt{p^x_B}\right)^2.$$

Plugging back into (3.81) we get

$$\sum_{x\in\mathcal{X}'} p(x)\left(\sqrt{p^x_{H(x)}} - \sqrt{p^x_{H_{\mathrm{opt}}(x)}}\right)^2 \le \varepsilon. \tag{3.82}$$

Now we connect (3.80) to the previous expression as follows

$$\begin{aligned}
&\{\mathrm{err}_D(H) - \mathrm{err}_D(H_{\mathrm{opt}})\}^2 \\
&\quad= \left\{\sum_{x\in\mathcal{X}'} p(x)\left(p^x_{H_{\mathrm{opt}}(x)} - p^x_{H(x)}\right)\right\}^2 \\
&\quad\le \left(\sum_{x\in\mathcal{X}'} p(x)\right)\left(\sum_{x\in\mathcal{X}'} p(x)\left(p^x_{H_{\mathrm{opt}}(x)} - p^x_{H(x)}\right)^2\right) \quad \text{(Cauchy-Schwartz)} \\
&\quad\le \sum_{x\in\mathcal{X}'} p(x)\left(\sqrt{p^x_{H_{\mathrm{opt}}(x)}} - \sqrt{p^x_{H(x)}}\right)^2\left(\sqrt{p^x_{H_{\mathrm{opt}}(x)}} + \sqrt{p^x_{H(x)}}\right)^2 && (3.83) \\
&\quad\le 2\sum_{x\in\mathcal{X}'} p(x)\left(\sqrt{p^x_{H_{\mathrm{opt}}(x)}} - \sqrt{p^x_{H(x)}}\right)^2 && (3.84) \\
&\quad\le 2\varepsilon, \ \text{(by (3.82))}
\end{aligned}$$

where (3.83) holds since

$$\sum_{x\in\mathcal{X}'} p(x) = \Pr_{(x',y')\sim D}[x' \in \mathcal{X}'] \le 1,$$

and (3.84) holds since

$$\begin{aligned}
p^x_{H(x)} + p^x_{H_{\mathrm{opt}}(x)} &= \Pr_{(x',y')\sim D}[y' \in \{H(x), H_{\mathrm{opt}}(x)\} \,|x] \le 1 \\
\implies \sqrt{p^x_{H(x)}} + \sqrt{p^x_{H_{\mathrm{opt}}(x)}} &\le \sqrt{2}.
\end{aligned}$$

Therefore, $\mathrm{err}_D(H) - \mathrm{err}_D(H_{\mathrm{opt}}) \le \sqrt{2\varepsilon}$. □

Note that the classifier $\bar{H}_T$, derived from the truncated scoring function $\bar{F}_T$ in the manner provided in (3.77), makes identical predictions to, and hence has the same err_D as, the classifier H_T output by the adaptive algorithm. Further, Lemma 3.27 seems to suggest that $\bar{F}_T$ satisfies the condition in (3.78), which, combined with our previous observation $\mathrm{err}_D(H) = \mathrm{err}_D(\bar{H}_T)$, would imply H_T approaches the optimal error. However, the condition (3.78) requires achieving optimal risk over all scoring functions, and not just ones achievable as a combination of weak classifiers in $\mathcal{H}$. Therefore, in order to use Lemma 3.28, we require the weak classifier space to be sufficiently rich, so that some combination of the weak classifiers in $\mathcal{H}$ attains risk_D arbitrarily close to the minimum attainable by any function:

$$\inf_{\alpha:\mathcal{H}\to\mathbb{R}} \mathrm{risk}_D(F_\alpha) = \inf_{F:\mathcal{X}\times\mathcal{Y}\to\mathbb{R}} \mathrm{risk}_D(F). \tag{3.85}$$

The richness condition, along with our previous arguments and Lemma 3.27, immediately imply the following result.

Theorem 3.29. *If the weak classifier space $\mathcal{H}$ satisfies the richness condition (3.85), and the number of rounds T is set to $\sqrt{m}$, then the error of the final classifier H_T approaches the Bayes optimal error:*

$$\Pr\left[\mathrm{err}_D\left(H_{\sqrt{m}}\right) \leq \mathrm{err}_D(H_{\mathrm{opt}}) + O\left(m^{-c}\right)\right] \geq 1 - \frac{1}{m^2}, \tag{3.86}$$

where $c > 0$ is some positive constant, and the probability is over the draw of training examples.

A consequence of the theorem is our strongest consistency result:

Corollary 3.30. *Let H_{opt} be the Bayes optimal classifier, and let the weak classifier space $\mathcal{H}$ satisfy the richness condition (3.85). Suppose m example and label pairs $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ are sampled from the distribution D, the number of rounds T is set to be $\sqrt{m}$, and these are supplied to AdaBoost.MM. Then, in the limit $m \to \infty$, the final classifier $H_{\sqrt{m}}$ output by AdaBoost.MM achieves the Bayes optimal error almost surely:*

$$\Pr\left[\left\{\lim_{m\to\infty} \mathrm{err}_D(H_{\sqrt{m}})\right\} = \mathrm{err}_D(H_{\mathrm{opt}})\right] = 1, \tag{3.87}$$

where the probability is over the randomness due to the draw of training examples.

The proof of Corollary 3.30, based on the Borel-Cantelli Lemma, is very similar to that of Corollary 12.3 in [46], and so we omit it. When $k = 2$, AdaBoost.MM is identical to AdaBoost. For Theorem 3.29 to hold for AdaBoost, the richness assumption (3.85) is necessary, since there are examples due to Long and Servedio [29] showing that the theorem may not hold when that assumption is violated.

Although we have seen that technically AdaBoost.MM is consistent under broad assumptions, intuitively perhaps it is not clear what properties were responsible for this desirable behavior. We next briefly study the high level ingredients necessary for consistency in boosting algorithms.

Key ingredients for consistency. We show here how both the choice of the loss function as well as the weak learning condition play crucial roles in ensuring consistency. If the loss function were not Bayes consistent as in Lemma 3.28, driving it down arbitrarily could still lead to high test error. For example, the loss employed by SAMME [57] does not upper bound the error, and therefore although it can manage to drive down its loss arbitrarily when supplied by the dataset discussed in Figure 3.1, although its error remains high.

Equally important is the weak learning condition. Even if the loss function is chosen to be error, so that it is trivially Bayes consistent, choosing the wrong weak learning condition could lead to inconsistency. In particular, if the weak learning condition is stronger than necessary, then, even on a boostable dataset where the error can be driven to zero, the boosting algorithm may get stuck prematurely because its stronger than necessary demands cannot be met by the weak classifier space. We have already seen theoretical examples of such datasets, and we will see some practical instances of this phenomenon in the next section.

On the other hand, if the weak learning condition is too weak, then a lazy Weak Learner may satisfy the Booster's demands by returning weak classifiers belonging only to a non-boostable subset of the available weak classifier space. For instance, consider again the dataset in Figure 3.1, and assume that this time the weak classifier space is much richer, and consists of all possible classifying functions. However, in any round, Weak Learner searches through the space, first trying hypotheses h_1 and h_2 shown in the figure, and only if neither satisfy the Booster, search for additional weak classifiers. In that case, any algorithm using SAMME's weak learning condition, which is known to be too weak and satisfiable by just the two hypotheses $\{h_1, h_2\}$, would only receive h_1 or h_2 in each round, and therefore be unable to reach the optimum accuracy. Of course, if the Weak Learner is extremely generous and helpful, then it may return the right collection of weak classifiers even with a null weak learning condition that places no demands on it. However, in practice, many Weak Learners used are similar to the lazy weak learner described since these are computationally efficient.

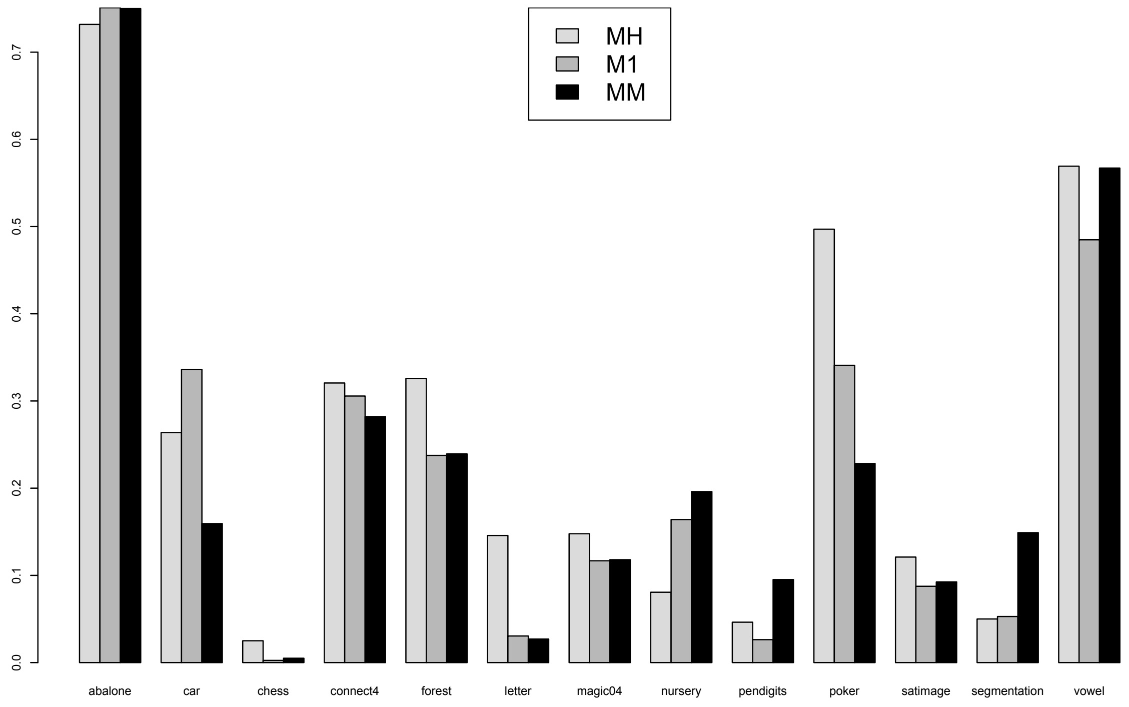
To see the effect of inconsistency arising from too weak learning conditions in practice, we need to test boosting algorithms relying on such datasets on significantly hard datasets, where only the strictest Booster strategy can extract the necessary service from Weak Learner for creating an optimal classifier. We did not include such experiments, and it will be an interesting empirical conjecture to be tested in the future. However, we did include experiments that illustrate the consequence of using too strong conditions, and we discuss those in the next section.

3.10 Experiments

In the final section of this chapter, we report preliminary experimental results on 13 UCI datasets: letter, nursery, pendigits, satimage, segmentation, vowel, car, chess, connect4, forest, magic04, poker, abalone. These datasets are all multiclass except for magic04, have a wide range of sizes, contain all combinations of real and categorical features, have different number of examples to number of features per example ratios, and are drawn from a variety of real-life situations. Most sets come with prespecified train and test splits which we use; if not, we picked a random $4 : 1$ split. Throughout this section by `MM` we refer to the version of AdaBoost.MM studied in the consistency section, which uses the approximate step size (3.67).

There were two kinds of experiments. In the first, we took a standard implementation `M1` of AdaBoost.M1 with C4.5 as weak learner, and the Boostexter implementation `MH` of AdaBoost.MH using stumps [48], and compared it against our method `MM` with a naive greedy tree-searching weak-learner `Greedy`. The size of trees to be used can be specified to our weak learner, and was chosen to be the of the same order as the tree sizes used by `M1`. The test-error after 500 rounds of boosting for each algorithm and dataset is bar-plotted in Figure 3.7. The performance is comparable

Figure 3.7: This is a plot of the final test-errors of standard implementations of M1, MH and MM after 500 rounds of boosting on different datasets. Both M1 and MM achieve comparable error, which is often larger than that achieved by MH. This is because M1 and MM used trees of comparable sizes which were often much larger and powerful than the decision stumps that MH boosted.



with M1 and far better than MH (understandably since stumps are far weaker than trees), even though our weak-learner is very naive. The convergence rates of error with rounds of M1 and MM are also comparable, as shown in Figure 3.8 (we omitted the curve for MH since it lay far above both M1 and MM).

We next investigated how each algorithm performs with less powerful weak-learners. We modified MH so that it uses a tree returning a single multiclass prediction on each example. For MH and MM we used the Greedy weak learner, while for M1 we used a more powerful-variant Greedy-Info whose greedy criterion was information gain rather than error (we also ran M1 on top of Greedy but Greedy-Info consistently gave better results so we only report the latter). We tried all tree-sizes in the set {10, 20, 50, 100, 200, 500, 1000, 2000, 4000} up to the tree-size used by M1 on C4.5 for each data-set. We plotted the error of each algorithm against tree size for each data-set in Figure 3.9. As predicted by our theory, our algorithm succeeds in boosting the accuracy even when the tree size is too small to meet the stronger weak learning assumptions of the other algorithms. More insight is provided by plots in Figure 3.10 of the rate of convergence of error with rounds when the tree size allowed is very small (5). Both M1 and MH drive down the error for a few rounds. But since boosting keeps creating harder distributions, very soon the

Figure 3.8: Plots of the rates at which `M1`(black,dashed) and `MM`(red,solid) drive down test-error on different data-sets when using trees of comparable sizes as weak classifiers. `M1` called C4.5, and `MM` called `Greedy`, respectively, as weak-learner. The tree sizes returned by C4.5 were used as a bound on the size of the trees that `Greedy` was allowed to return. This bound on the tree-size depended on the dataset, and are shown next to the dataset labels.

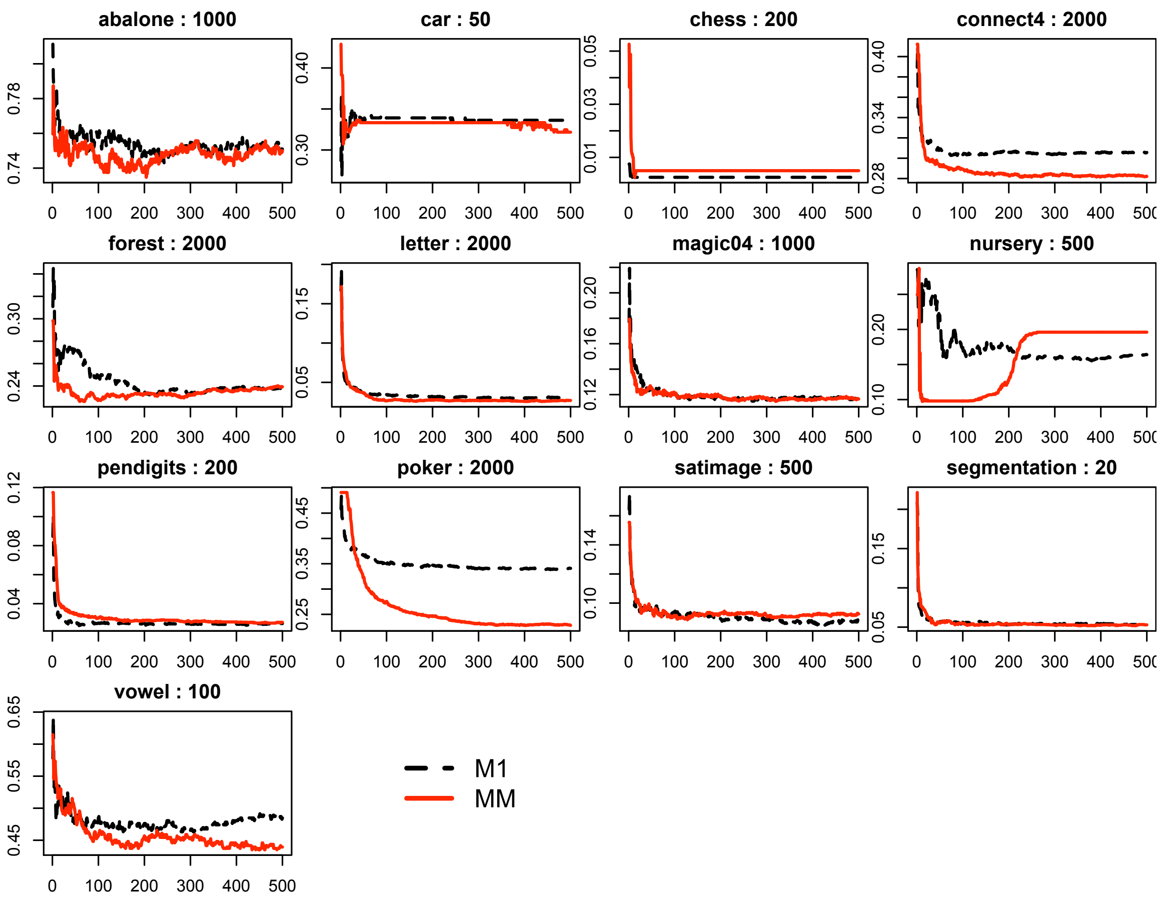
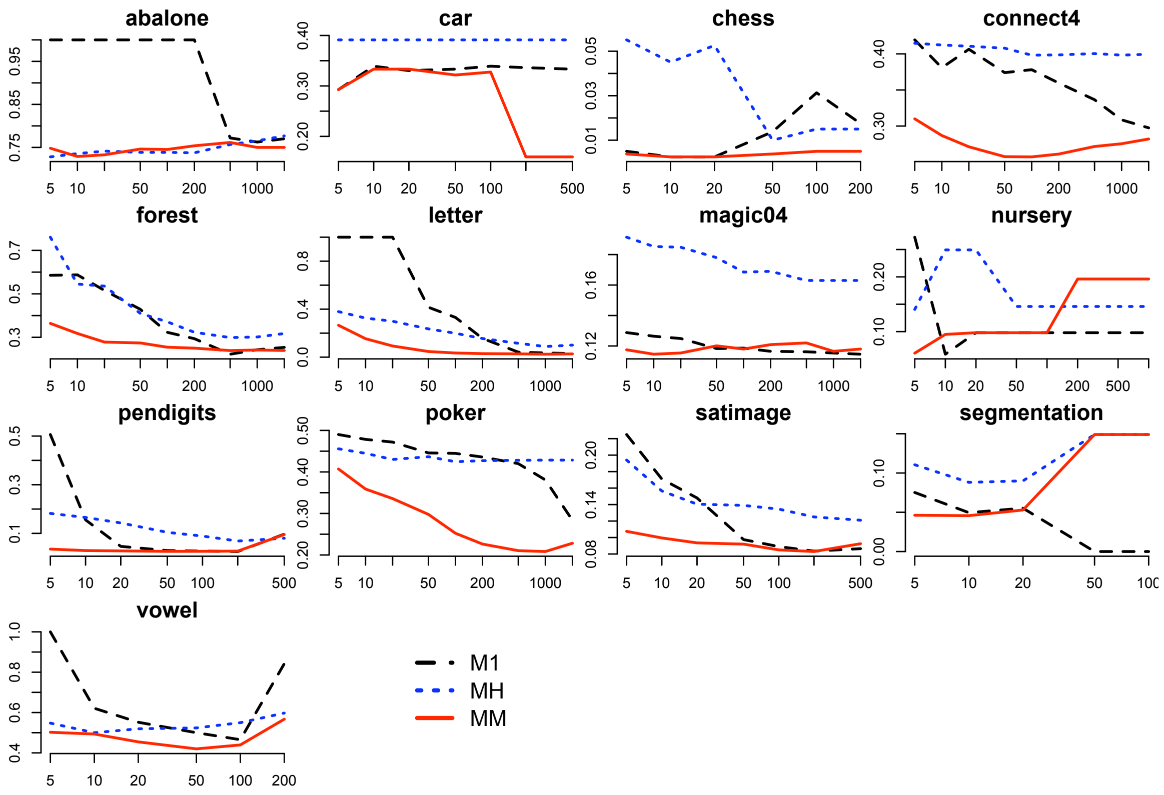


Figure 3.9: For this figure, `M1`(black, dashed), `MH`(blue, dotted) and `MM`(red,solid) were designed to boost decision trees of restricted sizes. The final test-errors of the three algorithms after 500 rounds of boosting are plotted against the maximum tree-sizes allowed for the weak classifiers. `MM` achieves much lower error when the weak classifiers are very weak, that is, with smaller trees.

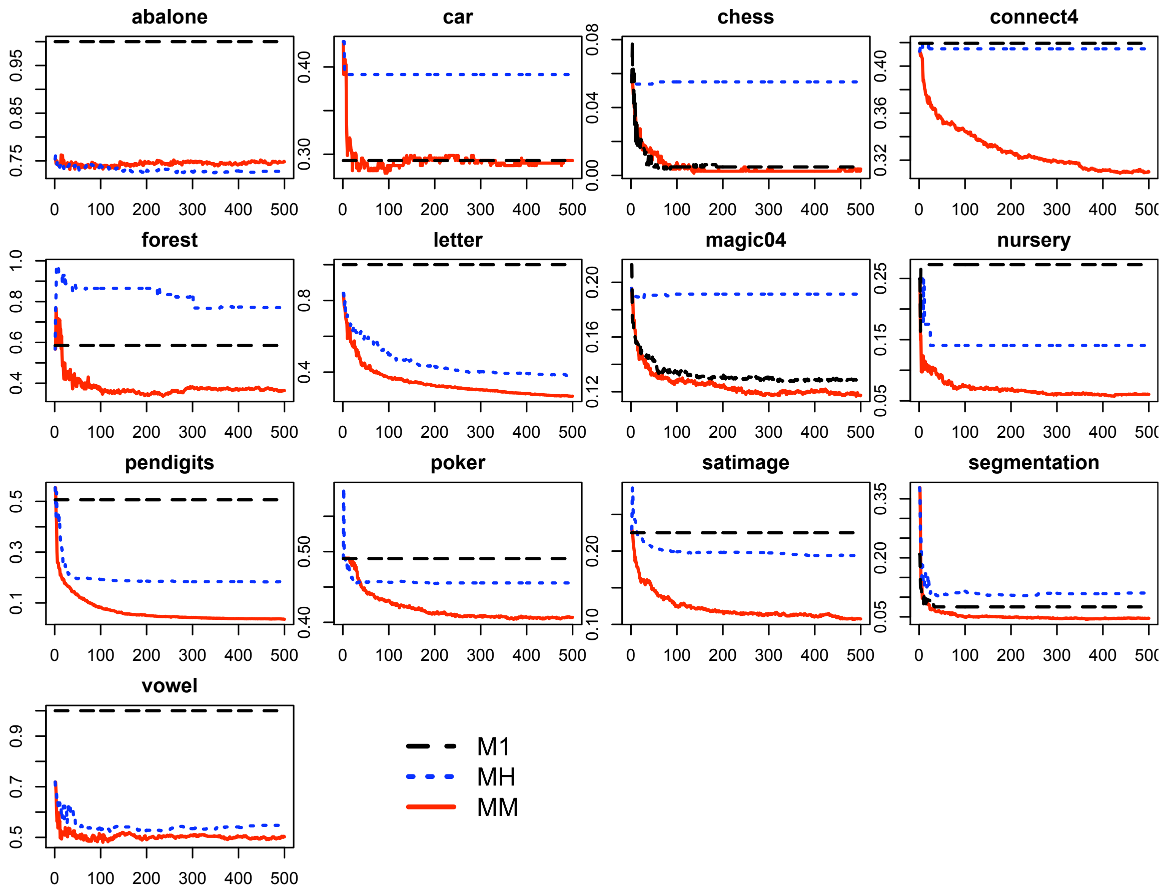


small-tree learning algorithms `Greedy` and `Greedy-Info` are no longer able to meet the excessive requirements of `M1` and `MH` respectively. However, our algorithm makes more reasonable demands that are easily met by `Greedy`.

3.11 Conclusion

In summary, we create a new framework for studying multiclass boosting. This framework is very general and captures the weak learning conditions implicitly used by many earlier multiclass boosting algorithms as well as novel conditions, including the minimal condition under which boosting is possible. We also show how to design boosting algorithms relying on these weak learning conditions that drive down training error rapidly. These algorithms are the optimal strategies for playing certain two player games. Based on this game-theoretic approach, we also design a multiclass boosting algorithm that is consistent, i.e., approaches the minimum empirical risk, and under some basic assumptions, the Bayes optimal test error. Preliminary exper-

Figure 3.10: A plot of how fast the test-errors of the three algorithms drop with rounds when the weak classifiers are trees with a size of at most 5. Algorithms `M1` and `MH` make strong demands which cannot be met by the extremely weak classifiers after a few rounds, whereas `MM` makes gentler demands, and is hence able to drive down error through all the rounds of boosting.



iments show that this algorithm can achieve much lower error compared to existing algorithms when used with very weak classifiers.

Although we can efficiently compute the game-theoretically optimal strategies under most conditions, when using the minimal weak learning condition, and non-convex 0-1 error as loss function, we require exponential computational time to solve the corresponding boosting games. Boosting algorithms based on error are potentially far more noise tolerant than those based on convex loss functions, and finding efficiently computable near-optimal strategies in this situation is an important problem left for future work. Further, we primarily work with weak classifiers that output a single multiclass prediction per example, whereas weak hypotheses that make multilabel multiclass predictions are typically more powerful. We believe that multilabel predictions do not increase the power of the weak learner in our framework, and our theory can be extended without much work to include such hypotheses, but we do not address this here. Finally, it will be interesting to see if the notion of minimal weak learning condition can be extended to boosting settings beyond classification, such as ranking.

3.12 Appendix

3.12.1 Optimality of the OS strategy

Here we prove Theorem 3.9. The proof of the upper bound on the loss is very similar to the proof of Theorem 2 in [44]. For the lower bound, a similar result is proved in Theorem 3 in [44]. However, the proof relies on certain assumptions that may not hold in our setting.

We first show that the average potential of states does not increase in any round. The dual form of the recurrence (3.24) and the choice of the cost matrix $\mathbf{C}_t$ in (3.25) together ensure that for each example i,

$$\begin{aligned} \phi_{T-t}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i)\right) &= \max_{l=1}^{k}\left\{\phi_{T-t-1}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i)+\mathbf{e}_l\right) - \left(\mathbf{C}_t(i)(l) - \langle \mathbf{C}_t(i), \mathbf{B}(i)\rangle\right)\right\} \\ &\geq \phi_{T-t-1}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i)+\mathbf{e}_{h_t(x_i)}\right) - \left(C_t(i, h_t(x_i)) - \langle \mathbf{C}_t(i), \mathbf{B}(i)\rangle\right). \end{aligned}$$

Summing up the inequalities over all examples, we get

$$\sum_{i=1}^{m}\phi_{T-t-1}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i)+\mathbf{e}_{h_t(x_i)}\right) \leq \sum_{i=1}^{m}\phi_{T-t}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i)\right) + \sum_{i=1}^{m}\left\{C_t(i, h_t(x_i)) - \langle \mathbf{C}_t(i), \mathbf{B}(i)\rangle\right\}$$

The first two summations are the total potentials in round $t+1$ and t, respectively, and the third summation is the difference in the costs incurred by the weak-classifier h_t returned in iteration t and the baseline $\mathbf{B}$. By the weak learning condition, this difference is non-positive, implying that the average potential does not increase.

Next we show that the bound is tight. In particular choose any accuracy parameter $\varepsilon > 0$, and total number of iterations T, and let m be as large as in (3.28). We show that in any iteration $t \leq T$, based on Booster's choice of cost-matrix $\mathbf{C}$, an adversary can choose a weak classifier $h_t \in \mathcal{H}^{\text{all}}$ such that the weak learning condition is satisfied, and the average potential does not fall by more than an amount ε/T. In fact, we show how to choose labels $l_1, \ldots, l_m$ such that the following hold simultaneously:

$$\sum_{i=1}^{m} C(i, l_i) \leq \sum_{i=1}^{m} \langle \mathbf{C}(i), \mathbf{B}(i) \rangle \tag{3.88}$$

$$\sum_{i=1}^{m} \phi_{T-t}^{\mathbf{B}(i)}(\mathbf{s}_t(i)) \leq \frac{m\varepsilon}{T} + \sum_{i=1}^{m} \phi_{T-t-1}^{\mathbf{B}(i)}(\mathbf{s}_t(i) + \mathbf{e}_{l_i}) \tag{3.89}$$

This will imply that the final potential or loss is at least ε less than the bound in (3.26).

We first construct, for each example i, a distribution $\mathbf{p}_i \in \Delta\{1, \ldots, k\}$ such that the size of the support of $\mathbf{p}_i$ is either 1 or 2, and

$$\phi_{T-t}^{\mathbf{B}(i)}(\mathbf{s}_t(i)) = \mathbb{E}_{l \sim \mathbf{p}_i}\left[\phi_{T-t-1}^{\mathbf{B}(i)}(\mathbf{s}_t(i) + \mathbf{e}_l)\right]. \tag{3.90}$$

To satisfy (3.90), by (3.20), we may choose $\mathbf{p}_i$ as any optimal response of the max player in the minimax recurrence when the min player chooses $\mathbf{C}(i)$:

$$\mathbf{p}_i \in \operatorname*{argmax}_{\mathbf{p} \in \mathcal{P}_i}\left\{\mathbb{E}_{l \sim \mathbf{p}}\left[\phi_{t-1}^{\mathbf{B}(i)}(\mathbf{s} + \mathbf{e}_l)\right]\right\} \tag{3.91}$$

$$\text{where } \mathcal{P}_i = \{\mathbf{p} \in \Delta\{1, \ldots, k\} : \mathbb{E}_{l \sim \mathbf{p}}[C(i, l)] \leq \langle \mathbf{C}(i), \mathbf{B}(i) \rangle\}. \tag{3.92}$$

The existence of $\mathbf{p}_i$ is guaranteed, since, by Lemma 3.7, the polytope $\mathcal{P}_i$ is non-empty for each i. The next result shows that we may choose $\mathbf{p}_i$ to have a support of size 1 or 2.

Lemma 3.31. *There is a $\mathbf{p}_i$ satisfying (3.91) with either 1 or 2 non-zero coordinates.*

Proof. Let $\mathbf{p}^*$ satisfy (3.91), and let its support set be S. Let μ_i denote the mean cost under this distribution:

$$\mu_i = \mathbb{E}_{l \sim \mathbf{p}^*}[C(i, l)] \leq \langle \mathbf{C}(i), \mathbf{B}(i) \rangle.$$

If the support has size at most 2, then we are done. Further, if each non-zero coordinate $l \in S$ of $\mathbf{p}^*$ satisfies $C(i, l) = \mu_i$, then the distribution $\mathbf{p}_i$ that concentrates all its weight on the label $l^{\min} \in S$ minimizing $\phi_{t-1}^{\mathbf{B}(i)}(\mathbf{s} + \mathbf{e}_{l^{\min}})$ is an optimum solution with support of size 1. Otherwise, we can pick labels $l_1^{\min}, l_2^{\min} \in S$ such that

$$C(i, l_1^{\min}) < \mu_i < C(i, l_2^{\min}).$$

Then we may choose a distribution $\mathbf{q}$ supported on these two labels with mean μ_i:

$$\mathbb{E}_{l \sim \mathbf{q}}[C(i, l)] = q(l_1^{\min})C(i, l_1^{\min}) + q(l_2^{\min})C(i, l_2^{\min}) = \mu_i.$$

Choose λ as follows:

$$\lambda = \min\left\{\frac{p^*(l_1^{\min})}{q(l_1^{\min})}, \frac{p^*(l_2^{\min})}{q(l_2^{\min})}\right\},$$

and write $\mathbf{p}^* = \lambda\mathbf{q} + (1-\lambda)\mathbf{p}$. Then both $\mathbf{p}, \mathbf{q}$ belong to the polytope $\mathcal{P}_i$, and have strictly fewer non-zero coordinates than $\mathbf{p}^*$. Further, by linearity, one of $\mathbf{q}, \mathbf{p}$ is also optimal. We repeat the process on the new optimal distribution till we find one which has only 1 or 2 non-zero entries. $\square$

We next show how to choose the labels $l_1, \ldots, l_m$ using the distributions $\mathbf{p}_i$. For each i, let $\left\{l_i^+, l_i^-\right\}$ be the support of $\mathbf{p}_i$ so that

$$C\left(i, l_i^+\right) \le \mathbb{E}_{l\sim\mathbf{p}_i}\left[C(i,l)\right] \le C\left(i, l_i^-\right).$$

(When $\mathbf{p}_i$ has only one non-zero element, then $l_i^+ = l_i^-$.) For brevity, we use p_i^+ and p_i^- to denote $p_i\left(l_i^+\right)$ and $p_i\left(l_i^-\right)$, respectively. If the costs of both labels are equal, we assume without loss of generality that $\mathbf{p}_i$ is concentrated on label l_i^-:

$$C\left(i, l_i^-\right) - C\left(i, l_i^-\right) = 0 \implies p_i^+ = 0, p_i^- = 1. \tag{3.93}$$

We will choose each label l_i from the set $\left\{l_i^-, l_i^+\right\}$. In fact, we will choose a partition S_+, S_- of the examples $1, \ldots, m$ and choose the label depending on which side S_ξ, for $\xi \in \{-, +\}$, of the partition element i belongs to:

$$l_i = l_i^\xi \text{ if } i \in S_\xi.$$

In order to guide our choice for the partition, we introduce parameters a_i, b_i as follows:

$$\begin{aligned} a_i &= C(i, l_i^-) - C(i, l_i^+), \\ b_i &= \phi_{T-t-1}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i) + \mathbf{e}_{l_i^-}\right) - \phi_{T-t-1}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i) + \mathbf{e}_{l_i^+}\right). \end{aligned}$$

Notice that for each example i and each sign-bit $\xi \in \{-1, +1\}$, we have the following relations:

$$C(i, l_i^\xi) = \mathbb{E}_{l\sim\mathbf{p}_i}\left[C(i,l)\right] - \xi(1-p_i^\xi)a_i \tag{3.94}$$

$$\phi_{T-t-1}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i) + \mathbf{e}_{l_i^\xi}\right) = \mathbb{E}_{l\sim\mathbf{p}_i}\left[\phi_{T-t}^{\mathbf{B}(i)}(i,l)\right] - \xi(1-p_i^\xi)b_i. \tag{3.95}$$

Then the cost incurred by the choice of labels can be expressed in terms of the parameters a_i, b_i as follows:

$$\begin{aligned}
\sum_{i\in S_+} C(i, l_i^+) + \sum_{i\in S_-} C(i, l_i^-) &= \sum_{i\in S_+} \left\{\mathbb{E}_{l\sim \mathbf{p}_i}\left[C(i,l)\right] - a_i + p_i^+ a_i\right\} \\
&\quad + \sum_{i\in S_-} \left\{\mathbb{E}_{l\sim \mathbf{p}_i}\left[C(i,l)\right] + p_i^+ a_i\right\} \\
&= \sum_{i=1}^{m} \mathbb{E}_{l\sim \mathbf{p}_i}\left[C(i,l)\right] + \left(\sum_{i=1}^{m} p_i^+ a_i - \sum_{i\in S_+} a_i\right) \\
&\le \sum_{i=1}^{m} \langle \mathbf{C}(i), \mathbf{B}(i)\rangle + \left(\sum_{i=1}^{m} p_i^+ a_i - \sum_{i\in S_+} a_i\right), \qquad (3.96)
\end{aligned}$$

where the first equality follows from (3.94), and the inequality follows from the constraint on $\mathbf{p}_i$ in (3.92). Similarly, the potential of the new states is given by

$$\sum_{i\in S_+} \phi_{T-t-1}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i) + \mathbf{e}_{l_i^+}\right) + \sum_{i\in S_-} \phi_{T-t-1}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i) + \mathbf{e}_{l_i^-}\right) \qquad (3.97)$$

$$\begin{aligned}
&= \sum_{i\in S_+} \left\{\mathbb{E}_{l\sim \mathbf{p}_i}\left[\phi_{T-t-1}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i) + \mathbf{e}_l\right)\right] - b_i + p_i^+ b_i\right\} \\
&\quad + \sum_{i\in S_-} \left\{\mathbb{E}_{l\sim \mathbf{p}_i}\left[\phi_{T-t-1}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i) + \mathbf{e}_l\right)\right] + p_i^+ b_i\right\} \\
&= \sum_{i=1}^{m} \mathbb{E}_{l\sim \mathbf{p}_i}\left[\phi_{T-t-1}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i) + \mathbf{e}_l\right)\right] + \left(\sum_{i=1}^{m} p_i^+ b_i - \sum_{i\in S_+} b_i\right) \\
&= \sum_{i=1}^{m} \phi_{T-t}^{\mathbf{B}(i)}\left(\mathbf{s}_t(i)\right) + \left(\sum_{i=1}^{m} p_i^+ b_i - \sum_{i\in S_+} b_i\right), \qquad (3.98)
\end{aligned}$$

where the first equality follows from (3.95), and the last equality from an optimal choice of $\mathbf{p}_i$ satisfying (3.90). Now, (3.96) and (3.98) imply that in order to satisfy (3.88) and (3.89), it suffices to choose a subset S_+ satisfying

$$\sum_{i\in S_+} a_i \ge \sum_{i=1}^{m} p_i^+ a_i, \qquad\qquad \sum_{i\in S_+} b_i \le \frac{m\varepsilon}{T} + \sum_{i=1}^{m} p_i^+ b_i. \qquad (3.99)$$

We simplify the required conditions. Notice the first constraint tries to ensure that S_+ is big, while the second constraint forces it to be small, provided the b_i are non-negative. However, if $b_i < 0$ for any example i, then adding this example to S_+ only helps both inequalities. In other words, if we can always construct a set S_+ satisfying (3.99) in the case where the b_i are non-negative, then we may handle the more general situation by just adding the examples i with negative b_i to the set S_+ that would be constructed by considering only the examples $\{i : b_i \geq 0\}$. Therefore we may assume without loss of generality that the b_i are non-negative. Further, assume (by relabeling if necessary) that $a_1, \ldots, a_{m'}$ are positive and $a_{m'+1}, \ldots a_m = 0$, for some $m' \leq m$. By (3.93), we have $p_i^+ = 0$ for $i > m'$. Therefore, by assigning the examples $m'+1, \ldots, m$ to the opposite partition S_-, we can ensure that (3.99) holds if the following is true:

$$\sum_{i \in S_+} a_i \quad \geq \quad \sum_{i=1}^{m'} p_i^+ a_i, \tag{3.100}$$

$$\sum_{i \in S_+} b_i \quad \leq \quad \max_{i=1}^{m'} |b_i| + \sum_{i=1}^{m'} p_i^+ b_i, \tag{3.101}$$

where, for (3.101), we additionally used that, by the choice of m (3.28) and the bound on loss variation (3.27), we have

$$\frac{m\varepsilon}{T} \geq \varnothing(L, T) \geq b_i \text{ for } i = 1, \ldots, m.$$

The next lemma shows how to construct such a subset S_+, and concludes our lower bound proof.

Lemma 3.32. *Suppose $a_1, \ldots, a_{m'}$ are positive and $b_1, \ldots, b_{m'}$ are non-negative reals, and $p_1^+, \ldots, p_{m'}^+ \in [0,1]$ are probabilities. Then there exists a subset $S_+ \subseteq \{1, \ldots, m'\}$ such that* (3.100) *and* (3.101) *hold.*

Proof. Assume, by relabeling if necessary, that the following ordering holds:

$$\frac{a(1) - b(1)}{a(1)} \geq \cdots \geq \frac{a(m') - b(m')}{a(m')}. \tag{3.102}$$

Let $I \leq m'$ be the largest integer such that

$$a_1 + a_2 + \cdots + a_I < \sum_{i=1}^{m'} p_i^+ a_i. \tag{3.103}$$

Since the p_i^+ are at most 1, I is in fact at most $m' - 1$. We will choose S_+ to be the first $I + 1$ examples $S_+ = \{1, \ldots, I+1\}$. Observe that (3.100) follows immediately from the definition of I. Further, (3.101) will hold if the following is true

$$b_1 + b_2 + \cdots + b_I \leq \sum_{i=1}^{m'} p_i^+ b_i, \tag{3.104}$$

since the addition of one more example $I+1$ can exceed this bound by at most $b_{I+1} \leq \max_{i=1}^{m'} |b_i|$. We prove (3.104) by showing that the left hand side of this equation is not much more than the left hand side of (3.103). We first rewrite the latter summation differently. The inequality in (3.103) implies we can pick $\tilde{p}_1^+, \ldots, \tilde{p}_{m'}^+ \in [0,1]$ (e.g., by simply scaling the p_i^+'s appropriately) such that

$$\begin{aligned} a_1 + \ldots + a_I &= \sum_{i=1}^{m'} \tilde{p}_i^+ a_i && (3.105) \\ \text{for } i = 1, \ldots, m' \colon \tilde{p}_i^+ &\le p_i. && (3.106) \end{aligned}$$

By subtracting off the first I terms in the right hand side of (3.105) from both sides we get

$$(1 - \tilde{p}_1^+) a_1 + \cdots + (1 - \tilde{p}_I^+) a_I = \tilde{p}_{I+1}^+ a_{I+1} + \cdots + \tilde{p}_{m'}^+ a_{m'}.$$

Since the terms in the summations are non-negative, we may combine the above with the ordering property in (3.102) to get

$$\begin{aligned} &(1 - \tilde{p}_1^+) a_1 \left(\frac{a_1 - b_1}{a_1} \right) + \cdots + (1 - \tilde{p}_I^+) a_I \left(\frac{a_I - b_I}{a_I} \right) \\ \ge\ & \tilde{p}_{I+1}^+ a_{I+1} \left(\frac{a_{I+1} - b_{I+1}}{a_{I+1}} \right) + \cdots + \tilde{p}_{m'}^+ a_{m'} \left(\frac{a_{m'} - b_{m'}}{a_{m'}} \right). \qquad (3.107) \end{aligned}$$

Adding the expression

$$\tilde{p}_1^+ a_1 \left(\frac{a_1 - b_1}{a_1} \right) + \cdots + \tilde{p}_I^+ a_I \left(\frac{a_I - b_I}{a_I} \right)$$

to both sides of (3.107) yields

$$\begin{aligned} \sum_{i=1}^{I} a_i \left(\frac{a_i - b_i}{a_i} \right) &\ge \sum_{i=1}^{m'} \tilde{p}_i^+ a_i \left(\frac{a_i - b_i}{a_i} \right) \\ \text{i.e. } \sum_{i=1}^{I} a_i - \sum_{i=1}^{I} b_i &\ge \sum_{i=1}^{m'} \tilde{p}_i^+ a_i - \sum_{i=1}^{m'} \tilde{p}_i^+ b_i \\ \text{i.e. } \sum_{i=1}^{I} b_i &\le \sum_{i=1}^{m'} \tilde{p}_i^+ b_i, && (3.108) \end{aligned}$$

where the last inequality follows from (3.105). Now (3.104) follows from (3.108) using (3.106) and the fact that the b_i's are non-negative. □

This completes the proof of the lower bound.

3.12.2 Consistency proofs

Here we sketch the proofs of Lemmas 3.26 and 3.27. Our approach will be to relate our algorithm to AdaBoost and then use relevant known results on the consistency of AdaBoost. We first describe the correspondence between the two algorithms, and then state and connect the relevant results on AdaBoost to the ones in this section.

For any given multiclass dataset and weak classifier space, we will obtain a transformed binary dataset and weak classifier space, such that the run of AdaBoost.MM on the original dataset will be in perfect correspondence with the run of AdaBoost on the transformed dataset. In particular, the loss and error on both the training and test set of the combined classifiers produced by our algorithm will be exactly equal to those produced by AdaBoost, while the space of functions and classifiers on the two datasets will be in correspondence.

Intuitively, we transform our multiclass classification problem into a single binary classification problem in a way similar to the all-pairs multiclass to binary reduction. A very similar reduction was carried out by [23]. Borrowing their terminology, the transformed dataset roughly consists of *mislabel* triples (x, y, l) where y is the true label of the example and l is an incorrect example. The new binary label of a mislabel triple is always -1, signifying that l is not the true label. A multiclass classifier becomes a binary classifier that predict ± 1 on the mislabel triple (x, y, l) depending on whether the prediction on x matches label l; therefore error on the transformed binary dataset is low whenever the multiclass accuracy is high. The details of the transformation are provided in Figure 3.11.

Some of the properties between the functions and their transformed counterparts are described in the next lemma, showing that we are essentially dealing with similar objects.

Lemma 3.33. *The following are identities for any scoring function $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ and weight function $\alpha : \mathcal{H} \to \mathbb{R}$:*

$$\widehat{\text{risk}}\left(F_\alpha\right) = \widetilde{\widehat{\text{risk}}}\left(\widetilde{F}_{\widetilde{\alpha}}\right) \tag{3.109}$$

$$\text{risk}_D\left(\bar{F}\right) = \widetilde{\text{risk}_D}\left(\bar{\widetilde{F}}\right). \tag{3.110}$$

The proofs involve doing straightforward algebraic manipulations to verify the identities and are omitted.

The next lemma connects the two algorithms. We show that the scoring function output by AdaBoost when run on the transformed dataset is the transformation of the function output by our algorithm. The proof again involves tedious but straightforward checking of details and is omitted.

Lemma 3.34. *If AdaBoost.MM produces scoring function F_α when run for T rounds with the training set S and weak classifier space $\mathcal{H}$, then AdaBoost produces the scoring function $\widetilde{F}_{\widetilde{\alpha}}$ when run for T rounds with the training set $\widetilde{S}$ and space $\widetilde{\mathcal{H}}$. We assume that for both the algorithms, Weak Learner returns the weak classifier in each round that achieves the maximum edge. Further we consider the version of AdaBoost.MM that chooses weights according to the approximate rule* (3.67).

We next restate the result for AdaBoost corresponding to Lemma 3.26 which we have already seen in Chapter 2.

	AdaBoost.MM	AdaBoost
Labels	$\mathcal{Y} = \{1, \ldots, k\}$	$\widetilde{\mathcal{Y}} = \{-1, +1\}$
Examples	$\mathcal{X}$	$\widetilde{\mathcal{X}} = \mathcal{X} \times ((\mathcal{Y} \times \mathcal{Y}) \setminus \{(y, y) : y \in \mathcal{Y}\})$
Weak classifiers	$h : \mathcal{X} \to \mathcal{Y}$	$\widetilde{h} : \widetilde{X} \to \{-1, 0, +1\}$, where $\widetilde{h}(x, y, l) = \mathbf{1}[h(x) = l] - \mathbf{1}[h(x) = y]$
Classifier space	$\mathcal{H}$	$\widetilde{\mathcal{H}} = \left\{\widetilde{h} : h \in \mathcal{H}\right\}$
Scoring function	$F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$	$\widetilde{F} : \widetilde{\mathcal{X}} \to \mathbb{R}$ where $\widetilde{F}(x, y, l) = F(x, l) - F(x, y)$
Clamped function	$\bar{F}(x, y) =$ $\max\{-C, F(x, l) - \max_{l'} F_T(x, l')\}$	$\bar{\widetilde{F}}(x, y, l) = \widetilde{F}(x, y, l)$, if $\left\lvert\widetilde{F}(x, y, l)\right\rvert \leq C$ $\bar{\widetilde{F}}(x, y, l) = C$, if $\left\lvert\widetilde{F}(x, y, l)\right\rvert > C$
Classifier weights	$\alpha : \mathcal{H} \to \mathbb{R}$	$\widetilde{\alpha} : \widetilde{\mathcal{H}} \to \mathbb{R}$ where $\widetilde{\alpha}\left(\widetilde{h}\right) = \alpha(h)$
Combined hypothesis	F_α where $F_\alpha(x, l) = \sum_{h \in \mathcal{H}} \alpha(h)\mathbf{1}[h(x) = l]$	$\widetilde{F}_{\widetilde{\alpha}}$ where $\widetilde{F}_{\widetilde{\alpha}}(x, y, l) = \sum_{\widetilde{h} \in \widetilde{\mathcal{H}}} \widetilde{\alpha}\left(\widetilde{h}\right)\widetilde{h}(x, y, l)$
Training set	$S = \{(x_i, y_i) : 1 \leq i \leq m\}$	$\widetilde{S} =$ $\{((x_i, y_i, l), \xi) : \xi = -1, l \neq y_i, 1 \leq i \leq m\}$
Test distribution	D over $\mathcal{X} \times \mathcal{Y}$	$\widetilde{D}$ over $\widetilde{X} \times \widetilde{Y}$ where $\widetilde{D}((x, y, l), -1) = D(x, y)/(k-1)$ $\widetilde{D}((x, y, l), +1) = 0$
Empirical risk	$\widehat{\text{risk}}(F) =$ $\frac{1}{m}\sum_{i=1}^m \sum_{l \neq y_i} e^{F(x_i, l) - F(x_i, y_i)}$	$\widetilde{\widehat{\text{risk}}}\left(\widetilde{F}\right)$ $\frac{1}{m(k-1)}\sum_{i=1}^m \sum_{l \neq y_i} e^{-\xi \widetilde{F}(x_i, y_i, l)}$
Test risk	$\text{risk}_D(F) =$ $\mathbb{E}_{(x,y)\sim D}\left[\sum_{l \neq y} e^{F(x,l) - F(x,y)}\right]$	$\widetilde{\text{risk}_D}\left(\widetilde{F}\right) =$ $\mathbb{E}_{((x,y,l),\xi)\sim\widetilde{D}}\left[e^{-\xi\widetilde{F}(x,y,l)}\right]$

Figure 3.11: Details of transformation between AdaBoost.MM and AdaBoost.

Lemma 3.35. *[Theorem 2.14] Suppose AdaBoost produces the scoring function $\widetilde{F}_{\widetilde{\alpha}}$ when run for T rounds with the training set $\widetilde{S}$ and space $\widetilde{\mathcal{H}}$. Then*

$$\widetilde{\widetilde{\text{risk}}}\left(\widetilde{F}_{\widetilde{\alpha}}\right) \leq \inf_{\widetilde{\beta}:\widetilde{\mathcal{H}}\to\mathbb{R}} \widetilde{\widetilde{\text{risk}}}\left(\widetilde{F}_{\widetilde{\beta}}\right) + C/T, \tag{3.111}$$

where the constant C depends only on the dataset.

The previous lemma, along with (3.109) immediately proves Lemma 3.26. The result for AdaBoost corresponding to Lemma 3.27 appears in [46].

Lemma 3.36 (Theorem 12.2 in [46])**.** *Suppose AdaBoost produces the scoring function $\widetilde{F}$ when run for $T = \sqrt{m}$ rounds with the training set $\widetilde{S}$ and space $\widetilde{\mathcal{H}}$. Then*

$$\Pr\left[\text{risk}_D\left(\bar{\widetilde{F}}\right) \leq \inf_{\widetilde{F'}:\widetilde{\mathcal{X}}\to\mathbb{R}} \text{risk}_D(\widetilde{F'}) + O\left(m^{-c}\right)\right] \geq 1 - \frac{1}{m^2}, \tag{3.112}$$

where the constant C depends only on the dataset.

The proof of Lemma 3.27 follows immediately from the above lemma and (3.110).

Chapter 4

Learning with Continuous Experts Using Drifting Games

Although boosting is mainly studied as a statistical learning algorithm, the theory of boosting has benefited greatly from viewpoints borrowed from related areas. One such area is online learning, where the goal is to design algorithms that can constantly learn from a stream of signals supplied to it. At least superficially, this is similar to the boosting framework, where the boosting algorithm has to constantly make use of signals in the form of weak classifiers that are supplied to it each round. In this chapter we flesh out this connection by focusing on a particular online learning problem that will lead to boosting algorithms with certain optimality properties.

We consider the problem of learning to predict as well as the best in a group of experts. Our model consists of a series of rounds. In each round, experts make predictions in $[-1, +1]$. This can be interpreted as giving a binary prediction, for example, if it will rain or not, with a certain degree of *confidence*. In particular, this means that an expert can choose to *abstain* from giving any prediction at all, in which case it predicts 0. The problem is to design a master algorithm that combines the expert predictions in each round to give its own binary prediction in $\{-1, +1\}$. At the end of each round, nature (or an adversary) reveals the truth, which is a value in $\{-1, +1\}$. The experts and the master suffer loss that depends on the amount by which their predictions deviated from the truth. Our goal is to ensure that our master algorithm does not suffer much loss relative to the best expert.

An important feature of our model, which we define rigorously in Section 4.1, is that we assume the master algorithm has prior knowledge of a bound k on the total loss that the best expert will suffer. With binary experts, outputting predictions in $\{-1, +1\}$, this problem was essentially solved entirely by Cesa-Bianchi et al. [11] who proposed the Binomial Weights (BW) algorithm. However, their work cannot be applied to our setting since here the experts are continuous, with predictions in $[-1, +1]$. In such a setting, other methods, notably exponential-weight algorithms [12, 23, 28], can be used instead. However, such algorithms do not enjoy the same level of tight optimality of the BW algorithm, and it has been an open problem since the introduction of BW as to whether this method can be generalized to continuous experts.

In this chapter, we present just such a generalization. In Section 4.2, we propose a new master strategy that gives the best possible performance for this problem. Our algorithm predicts using a weighted majority of the experts' predictions in each round, where the weights are carefully chosen to ensure that the master's loss is small relative to k. We also show that our algorithm runs in polynomial time.

Our algorithm is based on the *drifting games* framework introduced by Schapire [44]. This framework generalizes a number of on-line and boosting learning algorithms, including boost-by-majority [17], AdaBoost [23], the weighted majority algorithm [28] and Binomial Weights [11]. We apply the drifting games framework directly both to derive our algorithm, and to analyze its performance which, as seen in Section 4.3, relies heavily on properties of drifting games.

We also provide in Section 4.4 new lower bound constructions for master algorithms which employ weighted-majority predictions. These are slightly weaker than those provided by Cesa-Bianchi et al. [11] which already show that our algorithm is nearly the best possible when the number of experts is very large. However, their techniques are based on Spencer's [51] sophisticated results for Ulam's game, and require an enormous number of experts. In contrast, our lower bounds use simpler arguments based on the drifting games framework, and are meaningful for any number of experts. In the Appendix, we show how to extend the Spencer's [51] and Cesa-Bianchi et al.'s [11] approaches to achieve exactly tight lower bounds.

A consequence of our results is that learning in our framework with continuous experts is no harder than learning with abstaining experts, i.e., experts whose predictions are restricted to be in $\{-1, 0, +1\}$ (assuming that $2k$ is an integer), although there is a small gap between abstaining and binary experts.

Our results also have applications in boosting, where it leads to optimal algorithms for combining weak hypotheses that return confidence-rated predictions instead of just binary labels [47]. Although the algorithm turns out to be more complicated than in the expert learning setting, our theory ensures that it remains efficiently computable.

Other related work. Abernethy et al. [1] extended the BW algorithm to the setting where the experts remain binary, but the master is allowed to predict continuously. Their results do not directly apply to our setting, but with some effort can be used to obtain sub-optimal bounds. For completeness, a sketch is included in Section 4.1.1.

Continuous-time versions of drifting games, with potential applications to on-line learning, were studied by Freund and Opper [20]. In their setting, learning rounds are no longer discrete, but are instead continuous.

4.1 Expert Learning Model

Our expert learning model can be viewed as the following game. The players of the game are a fixed set of m experts, a master algorithm, and an adversary. The game proceeds through T rounds. In each round t, the following happen:

- The master chooses real weights $w_1^t, \ldots, w_m^t$ over the experts.

- Each expert i makes a prediction $x_i^t \in [-1, +1]$. The experts' predictions are controlled by the adversary; we distinguish between the experts and the adversary for clarity of exposition.

- The master predicts $\hat{y}^t \triangleq \operatorname{sign}(\sum_i w_i^t x_i^t) \in \{-1, 0, +1\}$. The sign function maps positive reals to 1, negative reals to -1 and 0 to itself.

- The adversary then chooses a label $y^t \in \{-1, +1\}$, causing expert i to suffer loss $\frac{1}{2}|y^t - x_i^t|$, and the master to suffer loss $\mathbf{1}(y^t \neq \hat{y})$, where $\mathbf{1}$ is the indicator function. Note that predicting 0 counts as a mistake.

The total loss of any player is the sum of the losses in each round. It is guaranteed that some expert will suffer at most k total loss, where k is known ahead of time to the master. The goal of the master is to come up with a strategy to choose distributions $\mathbf{w}^t$ in each round, so as to minimize its loss against the worst possible adversary. The performance of every fixed strategy will thus be a function of m and k.

We will only consider *conservative* master algorithms, i.e., algorithms that *ignore* rounds where it does not make a mistake, so that the weights it chooses in a certain round depend only on past rounds where it made a mistake. This will also allow us to assume that as long as the game can continue, the master makes a mistake in every round. Since one can easily convert any master algorithm in a mistake bounded model like ours to a conservative one without loss of performance, we do not lose generality with this assumption.

4.1.1 The Binning Algorithm

Abernethy et al. [1] study a similar game, where the Master predicts continuously, but the expert-predictions are binary. Experts in their model may split themselves into two parts of varying masses; each part independently makes a prediction and suffers integral loss. The game terminates when the total mass of experts with error at most k is less than one. They develop an optimal algorithm for their model, called binning. Superficially, the two models appear to be the same, but in fact, critical differences exist.

The first main difference between the models assumed by binning and ours is that experts suffer integral losses in binning, whereas in our case experts suffer continuous losses. However their experts are amorphous, and can split themselves into different parts which make different predictions and suffer different integral losses. The game-state in the binning model is captured by the total mass of experts that have suffered losses $0, \ldots, k$, respectively. Mathematically, their state-space is $[0, m]^{k+1}$. Their terminal states are those in which the sum of coordinates is less than 1. In contrast, the state of our expert learning game in any given round is given by the (possibly fractional) cumulative losses of each expert. Mathematically, our state-space is $[0, k+1]^m$; the terminal states are the ones where each coordinate is greater than k.

The next main difference is that whereas we find the optimal *deterministic* Master, Abernethy et al. [1] bound the *expected* error of a Master that predicts *randomly.* The very point of their work was to compute the improvement that a random Master could achieve over the best possible deterministic master, which, in their setting, was known to be the Binomial Weights algorithm of Cesa-Bianchi et al. [11]. Hence, their techniques are suited for finding optimal random Master algorithms, which is not the question we study here.

4.2 A master strategy for choosing weights

We describe a strategy of the master for choosing a distribution on the experts in each round. Computing this strategy requires playing a different type of game called a *drifting game* introduced by Schapire ([44]), which we have already briefly encountered in Chapter 3. We begin with an abstract definition of the game, and then go on to show how such games can be used to derive the BW algorithm [11], which is the optimal (in the broadest possible sense) master strategy in the case of binary experts. We then show how similar ideas can be used to derive a master algorithm for continuous experts.

4.2.1 Drifting Games

A drifting game is played by a *shepherd* and m *sheep* (also known as *chips* in [44]). In general, sheep exist in $\mathbb{R}^d$; however, in this chapter, we only consider drifting games in which $d = 1$. The game proceeds through T rounds. In each round t, the following happen:

- The shepherd chooses a weight $w_i^t \in \mathbb{R}$ for each sheep i. The sign indicates the direction he intends the sheep to move, and the magnitude encodes the importance he places on that sheep.

- Sheep i responds by shifting by z_i^t, where z_i^t belongs to a fixed set of directions B. Additionally, the following drifting constraint is obeyed

$$\sum_{i=1}^{m} w_i^t z_i^t \geq \delta \sum_{i=1}^{m} |w_i|. \tag{4.1}$$

  Here $\delta \geq 0$ and $B \subseteq \mathbb{R}$ are parameters of the game.

The shepherd suffers a loss $L(s)$ for every sheep that is at location $s \in \mathbb{R}$ at the end of the game; here $L : \mathbb{R} \to \mathbb{R}$ is a real function on the space. Initially, all the sheep are at the origin, so at the end of the game, sheep i is at $\sum_{t=1}^{T} z_i^t$. The goal of the shepherd is to choose weights in a way that would minimize its average loss $\frac{1}{m}\sum_i L(\sum_t z_i^t)$, assuming the worst behavior from the sheep.

Schapire [44] suggests a shepherd strategy OS based on a set of potential functions $\phi_t : \mathbb{R}^d \to \mathbb{R}$ defined recursively as follows:

- $\phi_T(x) = L(x)$

- $\phi_{t-1}(x) = \min_{w\in\mathbb{R}} \max_{z\in B}(\phi_t(x+z) + wz - \delta|w|)$.

Denoting by s_i^t the position of sheep i at time t, the OS algorithm chooses w_i^t as follows

$$w_i^t \in \arg\min_{w\in\mathbb{R}} \max_{z\in B}(\phi_{t+1}(s_i^t + z) + wz - \delta|w|). \tag{4.2}$$

(In this chapter, we regard arg min or arg max as returning the set of all values realizing the minimum or maximum.) Note that the time subscripts t on the potentials and the weights run in the opposite direction to that used in Chapter 3. This is only a superficial notational difference, and does not affect the results. We choose this over here since they match more closely the notation in the paper [44], whose results we will be heavily relying on. Schapire [44] provides an upper bound on the performance of the OS algorithm, and argues that (under some natural assumptions) it is optimal when the number of sheep m is very large. We record the results in the theorem below.

Theorem 4.1 (Drifting Games [44]). *If B is a bounded subset of $\mathbb{R}$, and L is locally bounded, then the loss suffered by the OS algorithm is upper bounded by $\phi_0(0)$, where ϕ is defined as above. Additionally, if B contains both positive and negative numbers of magnitude greater than δ, and L is globally bounded, then, given any $\varepsilon > 0$, for sufficiently large m, the sheep can force any shepherd algorithm to suffer at least $\phi_0(0) - \varepsilon$ loss at the end of the game.*

4.2.2 Learning with binary experts using drifting games

Consider our expert learning model with the change that experts make $\{-1, +1\}$ instead of continuous predictions. The Binomial Weights algorithm [11] is the best possible master strategy for this problem, even among master algorithms not restricted to predicting a weighted majority of the experts' predictions at each stage. We show how a master can simulate a drifting game to derive a strategy for choosing weights on the experts so as to perform as well as the BW algorithm.

The drifting game parameters are $B = \{-1, +1\}$ and $\delta = 0$, and the loss function is $L(s) = \mathbf{1}(x \leq 2k - T)$. The number of rounds is $T = T_0 + 1$ where T_0 will be specified later. For every expert, there is a sheep. At the beginning of a round, the master uses the shepherd's choice $w_1, \ldots, w_m$ for that round to assign weights to experts. After seeing the expert predictions x_i and the label y produced by the adversary, the master causes sheep i to drift by $z_i = -yx_i$. The drifting constraint (4.1) holds since we are in the conservative setting and assume a mistake is made by the master in each round. Schapire [44] shows that the resulting algorithm is equivalent to BW, for a certain choice of T_0.

We use the notation $\binom{q}{\leq k}$ to denote $\sum_{i=0}^{k} \binom{q}{i}$.

Theorem 4.2 (Learning with Binary Experts [11],[44]). *Consider the expert learning model described in Section 4.1, with the change that the expert predictions lie in*

$\{-1, +1\}$. *For this problem, when T_0 is set to be*

$$\max\left\{q \in \mathbb{N} : q \leq \lg m + \lg \binom{q}{\leq k}\right\} \tag{4.3}$$

the number of mistakes made by the master algorithm described in this section is upper bounded by T_0. Further, the resulting algorithm can be computed efficiently.

Proof. At the heart of the proof, and the reason behind our choice of T_0, lies the following result which was proved by Schapire [44]: If a drifting game with parameters $\delta = 0, B = \{-1, +1\}$ and loss function $L(x) = \mathbf{1}(x \leq 2k - T)$ is played for T rounds, then ϕ_t can be computed exactly, yielding

$$\phi_0(0) = 2^{-T}\binom{T}{\leq k}. \tag{4.4}$$

Note that the position of a sheep after t rounds is $2M - t$, where M is the loss suffered by the corresponding expert until then. Since we are guaranteed a mistake bound of at most k on some expert, we always have $\sum_i L(s_i^t) \geq 1$, where s_i^t is sheep i's position after t rounds of play. If the game could continue for $T = 1 + T_0$ rounds, using Theorem 4.1 and (4.4) we would have the following contradiction:

$$\frac{1}{m} \leq \frac{1}{m}\sum_i L(s_i^T) \leq \phi_0(0) = 2^{-T}\binom{T}{\leq k} < \frac{1}{m}.$$

The last inequality follows from the fact that T_0 was chosen to satisfy $T_0 = \max\{\text{number of rounds} : \phi_0(\mathbf{0}) \geq \frac{1}{m}\}$. This upper bounds the maximum number of rounds for which the game can continue, or equivalently, the maximum number of mistakes our master algorithm makes, by T_0. □ □

4.2.3 Drifting games for continuous experts

The same approach from the previous section can be applied to our expert learning model, where experts make $[-1, +1]$ predictions. The drifting game parameter B changes to $B = [-1, +1]$, and a new expression for T_0 has to be chosen; everything else remains the same. We summarize the master strategy in Algorithm 2, where we choose $T_0 = \max\left[q \in \mathbb{N} : q \leq \lg m + \lg \binom{T+1}{\leq k}\right]$. We can now state our first main result.

Theorem 4.3 (Learning with Continuous Experts). *Consider the expert learning model described in Section 4.1. For that problem, the loss of the master algorithm described in Algorithm 2 is upper bounded by T_0, which is equal to*

$$\max\left[q \in \mathbb{N} : q \leq \lg m + \lg \binom{q+1}{\leq k}\right]. \tag{4.5}$$

Algorithm 2 Master algorithm for continuous experts

```
Require: $k$ - mistake bound, $m$ - number of experts
  $T_0 \leftarrow \max\left[q \in \mathbb{N} : q \le \lg m + \lg \binom{T+1}{\le k}\right]$
  $B \leftarrow [-1, +1]$, $L \leftarrow \mathbf{1}(x \le 2k - T)$
  $\delta \leftarrow 0$
  Setup drifting game with $T = 1 + T_0, B, \delta, L$, and shepherd OS
  {Note: Game cannot continue beyond $T_0$ rounds}
  for $t = 1$ to $T_0$ do
    Accept $w_1^t, \ldots, w_m^t$ from shepherd (Algorithm 3)
    Accept predictions $x_1^t, \ldots, x_m^t$ from experts.
    Predict $\hat{y}^t = \text{sign}(\sum_i w_i x_i^t)$
    Accept label $y^t$ from adversary.
    For each $i$, make sheep $i$ drift by $z_i^t \triangleq -y^t x_i^t$
  end for
```

The bound for continuous experts looks very similar to the one for binary experts, except $\lg \binom{q}{\le k}$ in (4.3) is replaced by $\lg \binom{q+1}{\le k}$ in (4.5). Roughly, this means the continuous bound is twice as much as the binary one.

As in learning with binary experts, the choice of T_0 in Theorem 4.3 is dictated by the analysis of the drifting game used for playing with continuous experts. This analysis also constitutes our main technical contribution, and is summarized in the next theorem, but we defer a proof until the next section.

Theorem 4.4 (Drifting Games for $[-1, +1]$ Experts)**.** *Consider the drifting game with parameters $\delta = 0$, $B = [-1, +1]$, total number of rounds T and loss function $L(x) = \mathbf{1}(x \le 2k - T)$. The value of the potential function for this game at any integer point $s + 2k - T$ is given by*

$$\phi_{T-t}(s + 2k - T) = \begin{cases} 1 & \text{if } s \le 0 \\ 1 - 2^{-t} \sum_{i=0}^{s-1} \binom{t}{\lceil \frac{t+i}{2} \rceil} & \text{else.} \end{cases} \tag{4.6}$$

In particular we have

$$\phi_0(0) = 2^{-T} \binom{T+1}{\le k}.$$

Further, the OS strategy for this game can be computed efficiently.

Proof of Theorem 4.3. Observe that $T_0 = \max\{\text{number of rounds} : \phi_0(0) \ge \frac{1}{m}\}$, where ϕ_0 is the potential associated with the drifting game in Theorem 4.4. The rest of the proof is the same as that for Theorem 4.2. □

We can loosely upper bound the expression for the number of mistakes in Theorem 4.3 by

$$2k + \ln m \left(1 + \sqrt{1 + \frac{4k}{\ln m}}\right) - 1.$$

In Section 4.4 we will prove that, when the number of experts m is around 2^k, the mistake guarantee given in Theorem 4.3 is the best possible, up to an additive $O(\log k)$ term, when considering master algorithms that predict a weighted majority of the experts' predictions in each round. When $m \geq 2^{2^k}$, our upper bound is exactly tight as shown in the Appendix.

4.3 Analysis of drifting games for continuous experts

In this section we analyze the continuous drifting game and prove Theorem 4.4. Throughout we will be using the following two facts: ϕ_t is decreasing, and takes values in $[0, 1]$. These facts were proved more generally by Schapire [44].

We begin with a technical result necessary for proving Theorem 4.4.

Theorem 4.5 (Piecewise Convexity). *For every round t, ϕ_t is piecewise convex with pieces breaking at integers, i.e., for every integer n, ϕ_t is convex in $[n, n + 1]$.*

The proof of this theorem is complicated and we defer it to Section 4.5. The proof relies on Lemma 4.6, which will also be useful otherwise. The lemma gives us a tool for recursively computing the potentials ϕ_t. It can be proved using a more general result in [44], but here we give a direct proof for the case of interest.

Lemma 4.6. *If ϕ_t is piecewise convex with pieces breaking at integers, then for $s \notin \mathbb{Z}$,*

$$\phi_{t-1}(s) = \max\left\{\frac{z\phi_t(s+z') - z'\phi_t(s+z)}{z - z'} : z, z' \in Z, zz' < 0\right\} \tag{4.7}$$

where

$$Z = \{z \in [-1, +1] : s + z \in \mathbb{Z}\} \cup \{-1, +1\}. \tag{4.8}$$

For s integral, $\phi_{t-1}(s)$ is the maximum of $\phi_t(s)$ and the above expression.

Proof. By definition

$$\phi_{t-1}(s) = \min_w \max_{z\in[-1,+1]} (\phi_t(s + z) + wz).$$

For fixed s and w, our assumptions imply that $\phi_t(s + z) + wz$ is piecewise convex in z. As z varies over the convex set $[-1, +1]$, the maximum will be realized either at an endpoint, -1 or 1, or when $s + z$ lies at one of the endpoints of the convex pieces, which happens at the integers. This shows that we can restrict z to Z while evaluating $\phi_{t-1}(s)$.

Denote by Δ the simplex of distributions over Z. By the discussion above,

$$\begin{aligned}
\phi_{t-1}(s) &= \min_w \max_{z\in Z} (\phi_t(s + z) + wz) \\
&= \min_w \max_{p\in\Delta} \mathbb{E}_{z\sim p} \{\phi_t(s + z) + wz\} \\
&= \max_{p\in\Delta} \min_w \mathbb{E}_{z\sim p} \{\phi_t(s + z) + wz\}
\end{aligned}$$

where the last equality comes from Corollary 37.3.2 of [40]. Interpreting the right side as the Lagrangian dual we may compute $\phi_{t-1}(s)$ as the solution to the following optimization problem

$$\begin{array}{cl} \max_{p\in\Delta} & \mathbb{E}_{z\sim p}\left\{\phi_t(s+z)\right\} \\ s.t. & \mathbb{E}_{z\sim p}[z] = 0. \end{array}$$

The above is a linear program and is hence optimized at vertices of the polytope $\{p \in \Delta : \mathbb{E}_{z\sim p}[z] = 0\}$, which are mean-zero distributions supported on two points z, z' of opposite signs, or concentrated on 0 when feasible, i.e., when $s \in Z$. Maximizing $\mathbb{E}_{z\sim p}\left\{\phi_t(s+z)\right\}$ over such vertices p yields the lemma. □ □

As a corollary, we show how the OS algorithm may use the potentials to compute weights according to (4.2).

Corollary 4.7. *In round $t-1$, the OS algorithm puts the following weight on a sheep at location s*

$$w^* \triangleq -\left(\frac{\phi_t(s+z_2) - \phi_t(s+z_1)}{z_2 - z_1}\right), \tag{4.9}$$

where $z_1 < 0 < z_2$ realize the maximum in the right hand side of (4.7).

Proof. We need to show

$$\forall z \in [-1,+1] : \phi_t(s+z) + w^* z \le \phi_{t-1}(s).$$

As in the proof of Lemma 4.6, the maximum of the left hand side is attained by some z lying in Z (defined in (4.8)). Positive and negative z need to be handled separately but are symmetric (the case $z = 0 \in Z$ occurs only when s is integral; but for such s, Lemma 4.6 tells us $\phi_{t-1}(s) \ge \phi_t(s) = \phi_t(s+0) + w^* \cdot 0$ anyway). Therefore it suffices to show

$$\text{if } z \in Z, z > 0, \text{ then } \phi_t(s+z) + w^* z \le \phi_{t-1}(s).$$

Call the expression being maximized in (4.7) $f(z', z)$ and rewrite it as

$$f(z', z) = \phi_t(s+z') - s(z', z)z'.$$

where $s(z', z)$ denotes the slope $(\phi_t(s+z) - \phi_t(s+z'))\,/\,(z - z')$. Note both s, f are symmetric functions of their arguments. Since

$$z_1 < 0 < z_2 \in \arg \max_{z,z' \in Z : z' < 0 < z} \phi_t(s+z') - s(z', z)z',$$

we may conclude

$$w^* = -\max_{0<z\in Z} s(z_1, z). \tag{4.10}$$

Finish by observing

$$\begin{aligned}\phi_{t-1}(s) &\geq \max_{0<z\in Z} f(z,z_1) \text{ (Lemma 4.6)} \\ &= \max_{0<z\in Z} \phi_t(s+z) - s(z_1,z)z \\ &\geq \max_{0<z\in Z} \phi_t(s+z) + w^* z \text{ (by (4.10))} \quad \square\end{aligned}$$

□

It is now straightforward to prove Theorem 4.4.

Proof of Theorem 4.4: Theorem 4.5 and Lemma 4.6 imply that for integer points s

$$\phi_{t-1}(s) = \max\left\{\phi_t(s), \frac{\phi_t(s-1)+\phi_t(s+1)}{2}\right\}. \tag{4.11}$$

One can finish the proof by directly substituting into (4.11) the expression for $\phi_t(s)$ given in (4.6), and verifying that the inequality holds. We omit calculations.

Corollary 4.7 shows how the shepherd computes weights in each round. The OS routine is summarized in Algorithm 3. All we now need is a way to efficiently compute the potentials using recurrences (4.7) and (4.11). Note the value of ϕ_t at point s depends upon values at $\mathbb{Z}$ and $s+\mathbb{Z}$. An easy induction yields for all t, $\phi_t(s) = 1$ for $s \leq 2k - T$ and $\phi_t(s) = 0$ for $s \geq T$. Standard dynamic programming techniques can now be used to compute $\phi_t(s)$ in time polynomial in T. □

Algorithm 3 OS subroutine: initialized with T, B, δ, L from Algorithm 2

```
Require: Current round $t$ and positions $s_1^t, \ldots, s_m^t$ of each sheep : $s_i^t = \sum_{t'<t} z_i^t$
  (see Algorithm 2 for a definition of $z_i^t$)

  for each sheep $i$ do
    $Z \leftarrow [-1, \lfloor s_i^t \rfloor - s_i^t, \lceil s_i^t \rceil - s_i^t, +1]$ as in (4.8)
    Pick $z, z' \in Z$ maximizing, as in (4.7),
        $\frac{z\phi_{T-(t-1)}(s+z') - z'\phi_{T-(t-1)}(s+z)}{z-z'}$
    Set $w_i^t \leftarrow \frac{\phi_{T-(t-1)}(s+z') - \phi_{T-(t-1)}(s+z)}{z-z'}$ as in (4.9)
  end for
  return $w_1^t, \ldots, w_m^t$
```

Notice that (4.11) is the same as what we would get if the sheep were allowed to drift only by $-1, 0, +1$ at each time step. Correspondingly, in terms of provable upper bounds, our algorithm performs no worse with continuous experts than it does with abstaining experts, while with binary experts, the upper bound on performance is a tiny bit better. Combined with our lower bound results, this implies that, for sufficiently many experts, abstaining experts are exactly as powerful as continuous ones, while binary experts are only slightly less powerful.

4.4 Lower Bounds

In this section we provide lower bounds for on-line learning with continuous experts which almost match the upper bounds of Theorem 4.3, thus showing that the drifting game based on Algorithm 2 is near optimal. The bounds we obtain are weaker than those provided in the Appendix (which exactly match the upper-bounds derived in the previous section). We nevertheless include these, since the arguments are much simpler, and do not require the number of experts to depend on the mistake bound. The main result of this section is the following theorem.

Theorem 4.8 (Lower bound for expert learning)**.** *Consider the expert learning model defined in Section 4.1. For every master algorithm, the adversary can choose labels and cause the experts to make predictions in each round in a manner so as to force the master algorithm to suffer a loss of*

$$\max\left\{q \in \mathbb{N} : q < \lg\left(\frac{m}{\sqrt{k}}\right) + \lg\binom{q+1}{\leq k} + \Theta(1)\right\}, \tag{4.12}$$

where $\Theta(1)$ is a quantity bounded between some absolute constants c_1 and c_2.

The loss bound given above, and the upper bound in (4.5) define the smallest integer T such that $2^{-T}\binom{T+1}{\leq k}$ is less than $O(\frac{\sqrt{k}}{m})$ and $\frac{1}{m}$, respectively. Since $O(\log m)$ rounds will always be necessary, and $2^{-T}\binom{T+1}{\leq k}$ decreases exponentially fast when $T > 3k$, we see that the gap between the upper and lower bounds is only $O(\log k)$ when the number of experts m is around 2^k. When the number of experts is much larger ($m > 2^{2^k}$), a very different and highly involved analysis included in the Appendix shows that there is essentially no gap between the upper and lower bounds.

The proof of Theorem 4.8 consists of showing how an adversary in the expert model can exploit adversarial sheep movement in the drifting game to force any master algorithm to suffer high loss. This is the converse of what we saw in Section 4.2.3, where a well performing shepherd algorithm gave rise to master algorithms suffering low loss. At the heart of the proof of Theorem 4.8 is the following result on drifting games, showing how the sheep may drift without violating the drifting constraint, and yet cause any shepherd a large amount of loss.

Theorem 4.9. *Consider the drifting game with parameters $\delta = 0$, $B = [-1, +1]$, number of rounds T and loss function $\mathbf{1}(x \leq 2k - T)$. For any shepherd algorithm, there exists a strategy for the sheep that causes the shepherd to suffer a loss of*

$$\phi_0(0) - \frac{\Theta(\sqrt{k})}{m}$$

at the end of the game.

We prove Theorem 4.9 in the next section, but first we show how it can be used to prove Theorem 4.8.

Proof of Theorem 4.8: The adversary in our expert model (defined in Section 4.1) simulates a drifting game in $\mathbb{R}$, with parameters as above. The drifting game is played for T rounds, where T is given by the expression (4.12). For every expert, there is a sheep. At the beginning of a round, if the master places weights $w_1, \ldots, w_m$ on the experts, the adversary causes the shepherd to drive each sheep i in direction w_i. If the sheep drift in direction $z_1, \ldots, z_m$, the adversary causes expert i to predict $x_i = z_i$ (remember the adversary controls expert predictions). The drifting constraint $\sum_i w_i z_i \geq \delta = 0$ ensures that the weighted majority prediction of the master is 0 or 1. The adversary then outputs the label $y = -1$, causing the master to make a mistake in each round.

Note that the position of a sheep after t rounds is $2M - t$ where M is the loss suffered by the corresponding expert until then; thus an expert has suffered at most k loss if and only if the corresponding sheep lies at a point less than $2k - T$ at the end of the game. Hence, by our choice of loss function, the mistake bound on the experts is equivalent to ensuring the constraint that the loss suffered by the shepherd algorithm is strictly positive at the end of the game, so that at least one sheep has a final loss of 1.

Theorem 4.9 guarantees that the sheep can drift in a way so that the shepherd suffers at least $\phi_0(0) - \Theta(\sqrt{k})/m$ loss, where we know from Theorem 4.4 that $\phi_0(0) = \binom{T+1}{\leq k}$. Our choice of T satisfies $\phi_0(0) - \Theta(\sqrt{k})/m > 0$, completing the proof. $\square$

4.4.1 Lower bound for drifting game

We prove Theorem 4.9. Schapire ([44]) provides a similar though slightly weaker lower bound ($\phi_0(0) - O(T/\sqrt{m})$ instead of $\phi_0(0) - (\sqrt{k}/m)$) which leads to considerably weaker expert learning lower bounds. The reason is that Schapire's arguments hold for much more general drifting games. By carefully tailoring his proof to our specific learning model, we achieve significant improvements.

Proof of Theorem 4.9: We will show that on round t, the sheep can choose to drift in directions z_i so that

$$\frac{1}{m}\sum_i \phi_{t+1}(s_i^{t+1}) \geq \frac{1}{m}\sum_i \phi_t(s_i^t) - \frac{U_t}{m}. \tag{4.13}$$

Here s_i^t is the position of sheep i in round t, and

$$U_t \triangleq \max_{s^t} \frac{\phi_{t+1}(s^t - 1) - \phi_{t+1}(s^t + 1)}{2} \tag{4.14}$$

where the maximum is taken over all possible *integral* positions s^t of any sheep in round t. Note that this is different from the set of all possible positions, since the movement of the sheep is restricted to change by at most $+1$ or -1 in each round.

Among the possible positions, we take supremum over only those positions which happen to lie at an integer.

Repeatedly applying the above yields

$$\frac{1}{m}\sum_i L(s_i^T) \geq \phi_0(0) - \frac{1}{m}\sum_t U_t.$$

Appealing to Lemma 4.12 will then produce the desired bound.

For each i, $s_i^0 = 0$. Our sheep strategy will choose every drift to be in $\{-1, 0, 1\}$. Hence we may assume $s_i^t \in \mathbb{Z}$ for each i, t.

Fix a round t. From Lemma 4.6 we have

$$\phi_t(s) = \max\left\{\phi_{t+1}(s), \frac{\phi_{t+1}(s-1) + \phi_{t+1}(s+1)}{2}\right\}.$$

Let $I = \{1, \ldots, m\}$, $I_0 = \{i : \phi_t(s_i^t) = \phi_{t+1}(s_i^t)\}$, $I_1 = I \setminus I_0$. For each $i \in I_0$ we set $z_i^t = 0$. This ensures

$$\sum_{i \in I_0} \phi_{t+1}(s_i^{t+1}) = \sum_{i \in I_0} \phi_t(s_i^t).$$

For each $i \in I_1$ we must have

$$\phi_t(s_i^t) = \frac{\phi_{t+1}(s_i^t - 1) + \phi_{t+1}(s_i^t + 1)}{2}.$$

For such i we will choose z_i^t in $\{-1, +1\}$. Define $a_i^t \triangleq \frac{\phi_{t+1}(s_i^t-1) - \phi_{t+1}(s_i^t+1)}{2}$. Then, for each $i \in I_1$,

$$\phi_{t+1}(s_i^{t+1}) = \phi_t(s_i^t) - z_i^t a_i^t$$

since $s_i^{t+1} = s_i^t + z_i^t$. Thus

$$\sum_{i \in I_1} \phi_{t+1}(s_i^{t+1}) = \sum_{i \in I_1} \phi_t(s_i^t) - \sum_{i \in I_1} z_i^t a_i^t$$

Note that $a_i^t \in [0, U_t]$ by definition. If the shepherd weights for this round are $w_1^t, \ldots, w_m^t$, it suffices to ensure that $\sum_{i \in I_1} w_i^t z_i^t \geq 0$ while keeping $\sum_{i \in I_1} a_i^t z_i^t$ below U_t.

By Lemma 4.10, there exists a subset $P \subseteq I_1$ such that

$$\left|\sum_{i \in P} a_i^t - \sum_{j \in I_1 \setminus P} a_j^t\right| \leq U_t.$$

Assume without loss of generality that $\sum_{i \in P} w_i^t - \sum_{i \in I_1 \setminus P} w_j^t \geq 0$. Then assigning $z_i^t = +1$ for $i \in P$ and $z_i^t = -1$ for $i \in I_1 \setminus P$ would ensure the drifting constraints as well as (4.13), completing our proof. $\square$

Lemma 4.10. *For any sequence $a_1, \ldots, a_n$ of numbers in $[0, U]$*

$$\min_{P \subseteq I} \left| \sum_{i \in P} a_i - \sum_{j \in I \setminus P} a_j \right| \leq U$$

where $I = \{1, \ldots, n\}$.

Proof. Define discrepancy to be the argument of the min, and let P^* realize the minimum. If P^*'s discrepancy were greater than U, we could transfer any a_i from the heavier group to get a partition with lower discrepancy, a contradiction. □ □

Notice that the U_t can be trivially bounded by 1, since the ϕ_t take values in $[0, 1]$. That would give us a lower bound of $\phi(0) - \frac{T}{m}$. By being more careful, we get the following improvement.

Lemma 4.11. *Define U_t as in (4.14). Then*

$$U_{T-t} = \begin{cases} 2^{-t}\binom{t}{k} & \text{if } t > 2k \\ 2^{-t}\binom{t}{\lceil \frac{t}{2} \rceil} & \text{if } t \leq 2k. \end{cases}$$

Proof. Using (4.6), we have, for $s \geq 0$

$$\begin{aligned} &\frac{1}{2}[\phi_{T-t}(s - 1 + 2k - T) - \phi_{T-t}(s + 1 + 2k - T)] \\ &= 2^{-t-1}\left(\binom{t}{\lceil \frac{t+s-1}{2} \rceil} + \binom{t}{\lceil \frac{t+s}{2} \rceil}\right). \end{aligned} \tag{4.15}$$

Let $s + 2k - T$ be the position of a sheep at the end of $T - t$ rounds. Since it can drift by at most -1 in the negative direction in any round, we have $s + 2k - T \geq t - T$ so that $s \geq t - 2k$. We take two cases, depending on the value of k.

Suppose $t > 2k$. Then $s \geq t - 2k \geq 1$. Since (4.15) is larger for smaller (and non-negative) s, we can plug in $s = t - 2k$ to compute U_{T-t}.

$$U_{T-t} = 2^{-t-1}\left(\binom{t}{\lceil \frac{2t-2k-1}{2} \rceil} + \binom{t}{\lceil \frac{2t-2k}{2} \rceil}\right) = 2^{-t}\binom{t}{t-k} = 2^{-t}\binom{t}{k}.$$

When $t \leq 2k$, s can be less than 1. If $s < 0$, the left hand side of (4.15) is zero. If $s = 0$, the right hand side of the same equation equals $2^{-t}\binom{t}{\lceil \frac{t}{2} \rceil}$. Hence for $t \leq 2k$, $U_{T-t} = 2^{-t}\binom{t}{\lceil \frac{t}{2} \rceil}$. □ □

Lemma 4.12. *Define U_t as in (4.14). Then, $\sum_t U_t \leq \Theta(\sqrt{k})$.*

Proof. Lemma 4.11 yields

$$\sum_t U_t = \sum_{t>2k} U_t + \sum_{t\le 2k} U_t = \sum_{t>2k} 2^{-t}\binom{t}{k} + \sum_{t\le 2k} 2^{-t}\binom{t}{\lceil \frac{t}{2}\rceil}.$$

The terms in the first summation decrease by at least a factor of $3/4$ successively, so that we can upper bound it by $4\binom{2k}{k}$. Stirling's approximation yields $\binom{t}{\lceil \frac{t}{2}\rceil} < \frac{O(1)}{\sqrt{t}}$ for all positive integers t. Hence we have

$$\sum_t U_t \le \frac{4}{\sqrt{2k}} + \sum_{t\le 2k} \frac{O(1)}{\sqrt{t}} = \Theta(\sqrt{k})$$

completing our proof. □ □

4.5 Proof of Theorem 4.5

In this section we prove Theorem 4.5, which states that the potentials arising out of drifting games against continuous experts are not too complicated, but are in fact piecewise convex. Without this result, none of the computations would be possible. This is also the main reason lying behind the surprising fact that continuous experts are no stronger than abstaining ones, given a large number of experts.

We prove the Theorem by (backward) induction on time steps. Our proofs would be far simpler if we could inductively assume convex, rather than piecewise-convex, potentials at later time steps. Unfortunately, this is not the case, e.g. the end-potential is the non-convex $0-1$ loss function. The considerably weaker piecewise-convex condition causes many technical complications, and our proof is rather long and messy. The main ingredient is Lemma 5, proved below. We first show how this lemma suffices to prove the theorem.

Proof of Theorem 4.5: Lemma 4.13 shows that ϕ_t is convex in $(n, n+1)$. Since ϕ_t is decreasing, it is right-convex at n. We are left to show ϕ_t is left convex at $n+1$; since ϕ_t is convex and decreasing in $(n, n+1)$, this is equivalent to showing ϕ_t is left-continuous at $n+1$. Inductively, ϕ_{t+1} is decreasing and convex in $[n, n+1]$, and hence necessarily left-continuous at $n+1$. Since $\phi_t(n+1)$ is decreasing, it suffices to show

$$\phi_t(n+1) \ge \lim_{s\to(n+1)^-} \phi_t(s).$$

By (4.7), for $s \in (n, n+1)$, $\phi_t(s)$ is either

$$\frac{\phi_{t+1}(s-z_1) + z_1\phi_{t+1}(s+1)}{1+z_1},$$

where $z_1 \in [1, s-n]$, or

$$\frac{(n+1-s)\phi_{t+1}(s-z_1) + z_1\phi_{t+1}(n+1)}{n+1-s+z_1}.$$

As $s \to (n+1)^-$, the first expression tends to

$$\begin{aligned}&\frac{\lim_{s_1 \to n^-} \phi_{t+1}(s_1) + \lim_{s_2 \to (n+2)^-} \phi_{t+1}(s_2)}{2} \\ =& \frac{\phi_{t+1}(n) + \phi_{t+1}(n+2)}{2},\end{aligned}$$

where the equality holds since by induction, ϕ_{t+1} is left continuous at integers. Similarly, the second expression tends to $\phi_{t+1}(n+1)$. Therefore,

$$\lim_{s \to (n+1)^-} \phi_t(s) \leq \max\left[\phi_{t+1}(n+1), \frac{\phi_{t+1}(n) + \phi_{t+1}(n+2)}{2}\right].$$

But the right hand side of the above equation is $\phi_t(n+1)$ by (4.11). This completes the proof. □

Lemma 4.13. *For every integer n and round t, ϕ_t is convex in $(n, n+1)$.*

Proof. By backwards induction on t. The base case holds since ϕ_T is the loss function $\mathbf{1}(x \leq 2k - T)$. Assume ϕ_{t+1} is piecewise convex. Fix any integer $n \in \mathbb{Z}$. We have to show that ϕ_t is convex in $(n, n+1)$. Recall that, for non-integral points s, Eqs (4.7) and (4.8) state that $\phi_t = \max\{H_{13}, H_{23}, H_{14}, H_{24}\}$ where

$$H_{ij}(s) = \frac{z_i \phi_{t+1}(s + z_j) - z_j \phi_{t+1}(s + z_i)}{z_i - z_j},$$

and $(z_1, z_2, z_3, z_4) = (-1, n - s, n + 1 - s, +1)$. Checking that H_{14} and H_{23} are convex is straightforward. It turns out that H_{13} and H_{24} need not be convex. However, below we show that $\max\{H_{23}, H_{24}\}$ is convex, and a very similar proof works for showing $\max\{H_{23}, H_{13}\}$ is convex. As the supremum of convex functions is convex (Theorem 5.5 [40]), and because ϕ_t can be written as $\phi_t = \max\{\max\{H_{23}, H_{13}\}, \max\{H_{23}, H_{24}\}, H_{14}\}$, we are done.

We begin by making our task of showing $\max\{H_{23}, H_{13}\}$ is convex a little easier. The next lemma shows that it suffices to show only local convexity, meaning every point in the domain has a neighborhood over which the function is convex. The proofs of this and other technical lemmas are given later.

Lemma 4.14. *A locally convex function on $(0, 1)$ is convex.*

We eliminate a degenerate case before proceeding. If $\phi_{t+1}(n) = 0$, then $\phi_{t+1}(s) = 0$ for $s \geq n$, and H_{24} ends up being the 0 function. Then $\max\{H_{23}, H_{24}\} = H_{23}$ is convex. So assume $\phi_{t+1}(n) > 0$.

If H_{24} were locally convex, then it would immediately follow that $\max\{H_{23}, H_{24}\}$ is also locally convex. Unfortunately, H_{24} may fail to be convex in some neighborhood. We instead show that in any neighborhood, either H_{24} is convex, or $H_{23} \geq H_{24}$, which suffices. The conditions for each fact to hold are given in the next two lemmas. Since we are in the non-degenerate case ($\phi_{t+1}(n) > 0$), we can algebraically simplify the conditions by introducing some notation. To that end, we define functions $f, g : (0,1) \to \mathbb{R}$

$$f(x) = \frac{\phi_{t+1}(n+1+x)}{\phi_{t+1}(n)} \quad , \quad g(x) = \frac{f(x)-1}{1+x} \tag{4.16}$$

and we continuously extend them at 0.

Lemma 4.15. *Let f, g be as in (4.16). Then* $\max\{H_{23}, H_{24}\} = H_{23}$ *at a point $(n+x)$ if $g(0) \geq g(x)$.*

Proof. Using ϕ_{t+1} is decreasing, $f(0) \leq \frac{\phi_{t+1}(n+1)}{\phi_{t+1}(n)}$. The rest is simple algebra. □ □

Lemma 4.16. *Let f, g be as in (4.16). Then the left derivative f^L of f exists. Further, if $g(x) \leq f^L(x)$ in some open set U, then H_{24} is convex in the neighborhood $n + U$.*

The conditions in Lemmas 4.15 and 4.16 motivate the following definition.

Definition 4.17. *Let $f : [0,1) \to \mathbb{R}$ have a left-derivative f^L at all points, and define $g : [0,1) \to \mathbb{R}$ as in (4.16). Then f satisfies the* max-convex condition *if around every point there is a neighborhood $U \subseteq [0,1)$ for which at least one of the following holds:*

- $\forall x \in U : g(0) \geq g(x)$.
- $\forall x \in U : g(x) \leq f^L(x)$.

If f satisfies the max-convex condition, then our proof is complete, since then either Lemma 4.15 or Lemma 4.16 will apply. In either case, $\max\{H_{23}, H_{24}\}$ is locally convex. A picture providing intuition for why this might happen is given in Figure 4.1.

Continuing with our proof, observe that (4.16) defines f to be a positive scaling of ϕ_{t+1}, which is convex by the inductive assumption. By construction, f is continuous at 0 and hence convex in $[0,1)$. By Theorem 10.1 of [40], convexity of f implies continuity in $(0,1)$ as well. It turns out that by the next lemma these properties are sufficient.

Lemma 4.18. *Every convex, continuous $f : [0,1) \to \mathbb{R}$ satisfies the max-convex condition.*

□

We are now done showing Lemma 4.13, except for proving Lemmas 4.14, 4.16 and 4.18, which we do next. We will use the following standard fact about convex functions (Theorems 23.1 and 24.1 in [40]).

Lemma 4.19. *If f is convex in a neighborhood, then its left derivative f^L exists and may be defined as*

$$f^L(x) = \sup_{y<x} \frac{f(x)-f(y)}{x-y}. \tag{4.17}$$

Further, f^L is non-decreasing and left continuous.

Proof of Lemma 4.14. Suppose a function F is convex on (a,b) and (c,d) with $a < c < b < d$. We show F is convex on (a,d). Take any three points $x < y < z \in (a,d)$. It suffices to show $s_{x,y} \leq s_{y,z}$ where $s_{p,q}$ denotes the slope between points $(p, F(p))$ and $(q, F(q))$. Consider any two points $u, v \in (c,b)$. Let the set of five points $\{x,y,z,p,q\}$ in increasing order be $p_1, \ldots, p_5$. Then every three adjacent points lie entirely in (a,b) or (c,d); hence the slopes $s_{p_i,p_{i+1}}$ are increasing, and it follows that $s_{x,y} \leq s_{y,z}$ will hold.

Now consider any compact set $[a,b]$ with $0 < a \leq b < 1$. Since F is locally convex, every point in $(0,1)$ has an open interval containing it where F is convex. These form an open cover of $[a,b]$ and hence there is a minimal finite sub-cover $(a_1,b_1), \ldots, (a_N,b_N)$ with $a_1 < \cdots < a_N$ and $b_1 < \cdots < b_N$ by minimality. Using the procedure outlined above, we may conclude that F is convex over $[a,b]$. Since this holds for arbitrary $0 < a$ and $b < 1$, F is convex over $(0,1)$. $\square$

Proof of Lemma 4.16. As noted in the proof of Lemma 4.13, f is convex, so that by Lemma 4.19, its left derivative exists. Next observe that function h, defined as

$$h(x) \triangleq \frac{1 + xf(x)}{1+x} = \frac{H_{24}(n+x)}{\phi_{t+1}(n)},$$

is convex in a neighborhood U if and only if H_{24} is convex in $n+U$ (for geometric intuition about h, refer to Figure 4.1). For any points $0 < x < y < 1$, and convex combination $z = \lambda x + \mu y$, we get, after some algebra,

$$\lambda h(x) + \mu h(y) - h(z) = \frac{\lambda\mu(y-x)}{1+z}(g(y) - g(x)) + \frac{z(\lambda f(x) + \mu f(y) - f(z))}{1+z}.$$

The second term is non-negative since f is convex, and the first term is non-negative if g is non-decreasing. Hence h, and thus H_{24}, is convex in a region where g is not decreasing, which happens if $0 \leq g^L(x) = \frac{f^L(x) - g(x)}{1+x}$, i.e., $f^L(x) \geq g(x)$. $\square$

Proof of Lemma 4.18. We will need the following fact. For any points $x < y \in (0,1)$

$$g(y) \text{ is a weighted average of } g(x) \text{ and } \frac{f(y)-f(x)}{y-x}. \tag{4.18}$$

We take cases to show that for every point x, there is a neighborhood U containing it where either $g(0) \geq g(y) \forall y \in U$, or $f^L(y) \geq g(y) \forall y \in U$.

case 1: $g(0) > g(x)$: By the continuity of f and hence g, we get $g(0) \geq g(y)$ for y in an interval containing x.

case 2: $g(0) < g(x)$: We have

$$g(x) < \frac{f(x) - f(0)}{x} \leq f^L(x),$$

since the first inequality follows from (4.18), and the second one from (4.17). By left continuity, $f^L(y) > g(y)$ in a left neighborhood of x. For any $y > x$,

$$\begin{aligned} g(y) &\leq \max\{g(x), \frac{f(y)-f(x)}{y-x}\} \text{ (by (4.18))} \\ &\leq \max\{g(x), f^L(y)\} \text{ (by (4.17))} \\ &= f^L(y). \end{aligned}$$

The last equality holds since f^L is increasing and $f^L(x) > g(x)$.

We have therefore shown $f^L(y) \geq g(y)$ holds in a neighbourhood of x.

case 3: $g(0) = g(x)$: We have

$$g(x) = \frac{f(x)-f(0)}{x} \leq f^L(x),$$

since, the equality follows from (4.18), and the inequality from (4.17). If strict inequality holds, then we are done as in case 2. Otherwise we have $f^L(x) = \frac{f(x)-f(0)}{x}$. By Lemma 4.19, $f^L(x) = \sup_{y<x} \frac{f(y)-f(x)}{y-x}$, so that for any $y < x$,

$$\frac{f(x)-f(y)}{x-y} \leq \frac{f(x)-f(0)}{x}.$$

If strict inequality holds then, since $\frac{f(x)-f(0)}{x}$ is a weighted average of $\frac{f(x)-f(y)}{x-y}$ and $\frac{f(y)-f(0)}{y}$, we get

$$f^L(x) = \frac{f(x)-f(0)}{x} < \frac{f(y)-f(0)}{y} \leq f^L(y),$$

a contradiction, since f convex implies f^L is non-decreasing. Hence $\frac{f(x)-f(y)}{x-y} = \frac{f(x)-f(0)}{x}$ for all $y < x$ and the segment of the curve f between $(0, x)$ is a straight line. It follows from (4.18) that $g(y) = \frac{f(x)-f(0)}{x}$, which in turn is equal to $f^L(y)$, for y in a left neighborhood around x; since $f^L(x) \geq g(x)$ implies $f^L(y) \geq g(y)$ for y in a right neighborhood of x (as shown in case 2), we have $f^L(y) \geq g(y)$ in some neighborhood of x.

We have considered all cases. The proof follows. $\square$

4.6 Connections to boosting with confidence-rated hypotheses

An important application of the theory we derived so far is the design of optimally efficient boosting algorithms when using confidence rated weak hypotheses. Such weak hypotheses map examples not to just binary labels $\{-1, +1\}$, but to a real number in the unit interval $[-1, +1]$. The sign of the prediction denotes the label,

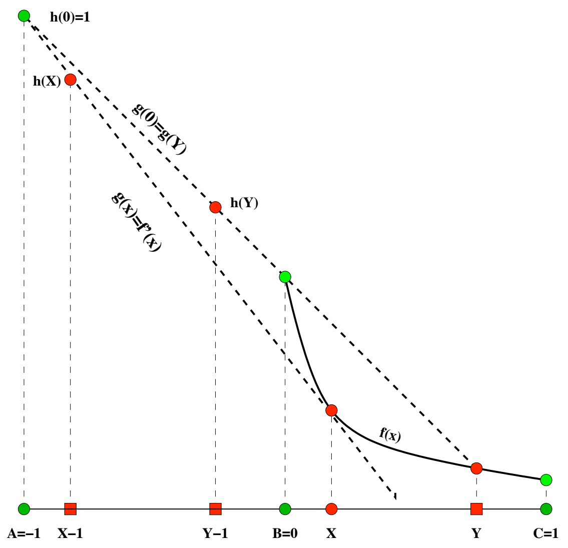


Figure 4.1: The diagram shows how any convex, continuous $f : [0, 1) \to \mathbb{R}$ satisfies the *max-convex condition* in Definition 4.17. The slopes of the dotted lines trace the function g, while the bold curved line indicates f. X is the x-coordinate of the point of contact of the tangent from $(A, 1)$ to f. The value Y is the x-coordinate of the point where the line joining $(A, 1)$ and $(B, f(0))$ hits the curve f again. For every point z in the region (B, Y), $g(0) \geq g(z)$. The other case, i.e. $g(z) \leq f^L(z)$ happens for every point in the region (X, C). Also included in the figure is a geometric intuition for the function $h(x) = \frac{1+xf(x)}{1+x}$, used in the proof of Lemma 4.16.

and the magnitude denotes the confidence of the hypothesis in its own prediction. Schapire and Singer [47] have shown that appropriate use of such weak hypotheses can lead to very efficient boosting algorithms. In this section, we see how to apply the results of the previous sections to derive such algorithms.

The results of Schapire [44], discussed in Theorem 4.1, show that computing the optimal boosting strategy with confidence rated hypotheses boils down to solving the drifting games recurrence when the response set of the adversary is the unit interval, $B = [-1, +1]$. Accordingly, the relevant recurrence is:

$$\begin{aligned} \phi_0 &= L \\ \phi_{t-1}(x) &= \min_{w\in\mathbb{R}} \max_{z\in[-1,+1]} (\phi_t(x+z) + wz - \delta|w|), \end{aligned} \tag{4.19}$$

where δ represents the *edge* over random guessing that the weak hypothesis must achieve in each round. Theorem 4.4 solves the above recurrence when the value $\delta = 0$, and a key ingredient was Theorem 4.5 which showed that the potential functions ϕ_t were piecewise convex, with pieces breaking at integers. A variation of this latter result can be shown using our previous techniques for non-zero values of δ. The potential still remains piecewise convex, but the break-points could be at any point that is the sum of an integer and a multiple of δ: $\{a + b\delta : a, b \in \mathbb{Z}\}$. The proofs are tedious but straightforward, and are omitted. Figure 4.2 shows the break points of

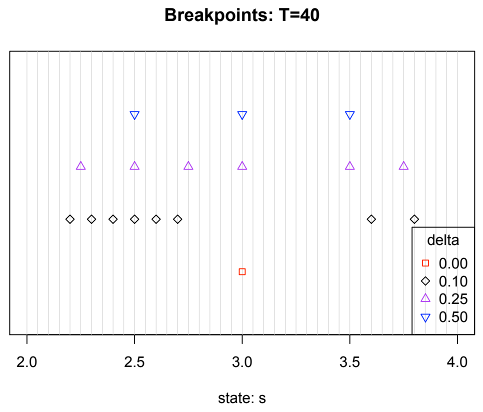


Figure 4.2: Breakpoints of the intervals on which the potential $\phi_0(\mathbf{s})$ is piecewise convex, for various values of δ. Only breakpoints for values of state $\mathbf{s}$ between 2 and 4 are considered, and the number of rounds is fixed at $T = 40$. The potential satisfies the recurrence in (4.19), where the loss function is set to be 0-1 error: $L(x) = \mathbf{1}(x \leq 0)$. Different values of $\delta \in \{0, 0.10, 0.25, 0.50\}$ are considered. The gridlines are drawn at every multiple of 0.05. We can see that the breakpoints occur only at values of the state $\mathbf{s}$ which are a sum of some integer multiple of δ and some integer.

the potential function $\phi_0(s)$ computed using 0-1 loss $L(x) = \mathbf{1}(x \leq 0)$ and number of rounds $T = 40$, for different values of δ. As a consequence, our neat formula (4.6) for calculating the potentials no longer applies for non-zero δ, although a dynamic programming procedure for calculating the value at any point in time $\text{poly}(T, 1/\delta)$ can be derived, using the fact that the potentials still remain piecewise convex, with at most $\text{poly}(T, 1/\delta)$ pieces. Further, by continuity, the potential value is very well approximated by (4.6) for small values of δ. We already saw in Section 3.8 how adaptive strategies arise when the edge is assumed to be infinitesimal, and therefore Theorem 4.4 may prove useful in designing optimally efficient adaptive algorithms for 0-1 error with confidence rated hypotheses. We note that such an approach has already been implemented in [18] for the case of weak hypotheses that return binary predictions without any confidence attached to them.

4.7 Conclusion

In this chapter we designed the optimal deterministic Master algorithm for continuous experts in the mistake bounded model, and computed the exact worst case error it suffers. Computing the optimal random Master algorithm against continuous experts is an open question. A combination of our techniques and the binning algorithm [1] might prove useful there. Finally we show how our techniques may be useful in deriving highly efficient boosting algorithms when the weak hypotheses are confidence rated.

4.8 Appendix: Tight lower bounds

We establish exact tightness of our upper bound (4.5) . As a consequence we are able to conclusively show that abstaining experts are as powerful as continuous experts for sufficiently many experts, as well as show that binary experts are strictly less powerful (although only slightly) than abstaining experts for infinitely many games. Our arguments are based on minor modifications of the reasoning in Section 2.3 of [11] and [51].

We setup some notation. An (m, k) game is an on-line expert game with m experts and k mistake bound. The optimal number of mistakes made by the Master in an (m, k) game assuming optimal play is denoted by $q^*_{\text{abs}}(m, k)$ and $q^*_{\text{bin}}(m, k)$ for abstaining and binary experts, respectively. We define $\text{Abs}(q, k) \triangleq 2^q / \binom{q+1}{\le k}$, and $\text{Bin}(q, k) \triangleq 2^q / \binom{q}{\le k}$. Then the upper bounds obtained by us (4.5) and the Binomial Weights(BW) [11] algorithm (4.3) are given by $\max\{q : \text{Abs}(q, k) \le m\}$ and $\max\{q : \text{Bin}(q, k) \le m\}$, respectively.

4.8.1 Bounds are tight for abstaining

Theorem 4.20. *For any k and $m > 2^{2^k}$, let $Abs(q, k) \le m < Abs(q + 1, k)$. Then,*

1. *$q \le q^*_{abs}(m, k) + 1$.*
2. *If $m \ge Abs(q, k) + 2^k$, then $q = q^*_{abs}(m, k)$.*

It is not difficult to check, that for $q \gg k$, $\text{Abs}(q + 1, k) - \text{Abs}(q, k) \gg 2^k$, so that condition 2 is satisfied by almost all games.

Proof of Theorem 4.20. We assume familiarity with Section 2.3 of [11] and [51]. Consider the following chip game with two players Paul and Carole. There are $k + 2$ bins on the *non-negative integer line*, initially all at 0. In every round, Paul *abstains* on an arbitrary subset of the chips, and splits the remaining chips arbitrarily into two sets. Then Carole chooses one of the two sets and advances each chip in it by one. Abstaining twice on the same chip causes it to advance by one bin. The game ends when all chips are located beyond k. Carole wants the game to end soon, whereas Paul wants it to drag. Reasoning very similar to that in Section 2.3 of [11] shows that, assuming optimal play, the number of rounds in the chip game is $q^*_{\text{abs}}(m, k)$. In the rest of the proof, we describe a strategy for Paul that ensures the game lasts for at least $q - 1$ rounds, and when condition 2 is met, for q rounds.

The strategy consists of three stages. In the first two stages, Paul never abstains, so the state of the game can be described by a configuration $I = (I_0, \ldots, I_k)$, where I_j denotes the number of chips in position j (chips beyond k do not matter). For any r, define the weight $W_r(I)$ of configuration I to be $\sum_j I_j \binom{r+1}{\le j}$. Chips in position k are called pennies. At the end of the first two stages, there will be at most one non-penny remaining.

By choice of q, the weight of the initial configuration given by W_q is $m2^q$. The first stage lasts for k steps at the end of which, like in the proof of Theorem 5 in

[11], Paul can reach a configuration I^k where $W_{\tilde{q}}(I^k) = 2^{\tilde{q}}$, and $I^k_k > c(k)\tilde{q}^k$, where $c(k)$ is sufficiently large. If condition 2 holds then $\tilde{q} = q - k$ or else $\tilde{q} = q - k - 1$. We will show that the game lasts for $\tilde{q}$ more rounds which will complete the proof of the theorem. Henceforth, the weight of a configuration is given by W_r where r is the number of rounds remaining.

Next, in the second stage, we apply fictitious play as described in [51]. The analysis of the First Steps, Middle, Late Middle and Early End stages in the proof of **Main Theorem** in [51] carries through with our modified weight function W as well, as straightforward calculations show. Therefore, we halve the weight of the configuration every round, until, as [51] shows, we reach a configuration with at most one non-penny.

Finally, in the third stage, we modify the **Endgame Lemma** of [51] in Lemma 4.21 to finish off the proof. This is the only place where Paul might use abstaining moves. □

Lemma 4.21. *Let $(x_0, \ldots, x_k)$ be a configuration with $x_0 \leq 1$, $x_1 = \ldots = x_{k-1} = 0$ and $W_j(x_0, \ldots, x_k) = 2^j$. Then Paul can play for j more rounds.*

Proof. Note that $W_j(x_0, \ldots, x_k) = x_0\binom{j+1}{\leq k} + x_k$. If $x_0 = 0$, then $x_k = 2^j$, and Paul may continue halving for j more rounds. So assume $x_0 > 0$. If $j \leq 2k$, then Paul may use abstaining moves on the chip at position 0 to make the game last for at least $2k \geq j$ moves. If $j > 2k$, then both $\binom{j}{\leq k}, \binom{j}{\leq k-1}$, are less than 2^{j-1}. Since $\binom{j+1}{\leq k} = \binom{j}{\leq k} + \binom{j}{\leq k-1}$, there is a way to split the chips into two parts of exactly equal weight. The proof follows by induction. □ □

4.8.2 Abstaining is Continuous

Whenever condition 2 of Theorem 4.20 is satisfied, the upper-bound for continuous experts matches the lower bound for abstaining experts, and hence both classes of experts are equally powerful. For number-theoretic reasons, the upper-bound might occasionally be one less than the optimum. The same situation occurs for the BW upper-bound and the true optimum in games against binary experts. Appendix A of [11] contain an enhanced version of the BW algorithm which always achieves the exact optimum. It is straightforward to check that our algorithms for continuous and abstaining experts may be enhanced similarly to achieve exactly optimum performance for all (m, k) games with $m > 2^{2^k}$. This implies, that whenever m is sufficiently large with respect to k, abstaining and continuous experts are game-theoretically equally powerful.

4.8.3 Binary less powerful than Abstaining

Basic calculations show the next useful Lemma.

Lemma 4.22. *The following hold:*

1. *For $q > 2^k$, $Bin(q, k) < Abs(q + 1, k) < Bin(q + 1, k)$*

2. For $q \gg k$, $Bin(q+1,k) - Abs(q+1,k) \gg 2^k$.

A straightforward application of the above and Theorem 4.20 now yields the following separation, which is the best possible according to point 1 in Lemma 4.22.

Corollary 4.23. *For any k, define*

$$M_k \triangleq \cup_{q>2^k}\{Abs(q+1,k)+2^k,\ldots,Bin(q+1,k)-1\}.$$

Then, M_k has constant density, and if $m \in M_k$, $q^*_{abs}(m,k) = q^*_{bin}(m,k)+1$.

Bibliography

[1] Abernethy J, Langford J, Warmuth MK (2006) Continuous experts and the binning algorithm. In: 19th Annual Conference on Learning Theory

[2] Abernethy J, Bartlett PL, Rakhlin A, Tewari A (2008) Optimal stragies and minimax lower bounds for online convex games. In: COLT, pp 415–424

[3] Allwein EL, Schapire RE, Singer Y (2000) Reducing multiclass to binary: A unifying approach for margin classifiers. Journal of Machine Learning Research 1:113–141

[4] Bartlett PL, Traskin M (2007) AdaBoost is consistent. Journal of Machine Learning Research 8:2347–2368

[5] Bartlett PL, Jordan MI, McAuliffe JD (2006) Convexity, classification, and risk bounds. Journal of the American Statistical Association 101(473):138–156

[6] Beygelzimer A, Langford J, Ravikumar P (2009) Error-correcting tournaments. In: Algorithmic Learning Theory: 20th International Conference, pp 247–262

[7] Bickel PJ, Ritov Y, Zakai A (2006) Some theory for generalized boosting algorithms. Journal of Machine Learning Research 7:705–732

[8] Boyd S, Vandenberghe L (2004) Convex Optimization. Cambridge University Press

[9] Breiman L (1999) Prediction games and arcing classifiers. Neural Computation 11(7):1493–1517

[10] Caruana R, Niculescu-Mizil A (2006) An empirical comparison of supervised learning algorithms. In: Proceedings of the 23rd International Conference on Machine Learning

[11] Cesa-Bianchi N, Freund Y, Helmbold DP, Warmuth MK (1996) On-line prediction and conversion strategies. Machine Learning 25:71–110

[12] Cesa-Bianchi N, Freund Y, Haussler D, Helmbold DP, Schapire RE, Warmuth MK (1997) How to use expert advice. Journal of the Association for Computing Machinery 44(3):427–485

[13] Collins M, Schapire RE, Singer Y (2002) Logistic regression, AdaBoost and Bregman distances. Machine Learning 48(1/2/3)

[14] Dietterich TG, Bakiri G (1995) Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research 2:263–286

[15] Eibl G, Pfeiffer KP (2005) Multiclass boosting for weak classifiers. Journal of Machine Learning Research 6:189–210

[16] Frean M, Downs T (1998) A simple cost function for boosting. Tech. rep., Department of Computer Science and Electrical Engineering, University of Queensland

[17] Freund Y (1995) Boosting a weak learning algorithm by majority. Information and Computation 121(2):256–285

[18] Freund Y (1999) An adaptive version of the boost by majority algorithm. In: Proceedings of the Twelfth Annual Conference on Computational Learning Theory, pp 102–113

[19] Freund Y (2001) An adaptive version of the boost by majority algorithm. Machine Learning 43(3):293–318

[20] Freund Y, Opper M (2002) Continuous drifting games. Journal of Computer and System Sciences pp 113–132

[21] Freund Y, Schapire RE (1996) Experiments with a new boosting algorithm. In: Machine Learning: Proceedings of the Thirteenth International Conference, pp 148–156

[22] Freund Y, Schapire RE (1996) Game theory, on-line prediction and boosting. In: Proceedings of the Ninth Annual Conference on Computational Learning Theory, pp 325–332

[23] Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1):119–139

[24] Friedman J, Hastie T, Tibshirani R (2000) Additive logistic regression: A statistical view of boosting. Annals of Statistics 28(2):337–374

[25] Friedman JH (2001) Greedy function approximation: A gradient boosting machine. Annals of Statistics 29(5)

[26] Hastie T, Tibshirani R (1998) Classification by pairwise coupling. Annals of Statistics 26(2):451–471

[27] Koltchinskii V, Panchenko D (2002) Empirical margin distributions and bounding the generalization error of combined classifiers. Annals of Statistics 30(1)

[28] Littlestone N, Warmuth MK (1994) The weighted majority algorithm. Information and Computation 108:212–261

[29] Long PM, Servedio RA (2010) Random classification noise defeats all convex potential boosters. Machine Learning 78:287–304

[30] Luenberger DG, Ye Y (2008) Linear and nonlinear programming, 3rd edn. Springer

[31] Luo ZQ, Tseng P (1992) On the convergence of the coordinate descent method for convex differentiable minimization. Journal of Optimization Theory and Applications 72(1):7–35

[32] Mason L, Baxter J, Bartlett P, Frean M (2000) Boosting algorithms as gradient descent. In: Advances in Neural Information Processing Systems 12

[33] Mukherjee I, Schapire RE (2010) Learning with continuous experts using drifting games. Theoretical Computer Science 411(29-30):2670–2683

[34] Mukherjee I, Schapire RE (2010) A theory of multiclass boosting. In: Twenty Third Annual Conference on Neural Information Processing Systems

[35] Mukherjee I, Rudin C, Schapire RE (2011) The rate of convergence of AdaBoost. In: The 24th Annual Conference on Learning Theory

[36] Onoda T, Rätsch G, Müller KR (1998) An asymptotic analysis of AdaBoost in the binary classification case. In: Proceedings of the 8th International Conference on Artificial Neural Networks, pp 195–200

[37] Rätsch G, Warmuth MK (2005) Efficient margin maximizing with boosting. Journal of Machine Learning Research 6:2131–2152

[38] Rätsch G, Onoda T, Müller KR (2001) Soft margins for AdaBoost. Machine Learning 42(3):287–320

[39] Rätsch G, Mika S, Warmuth MK (2002) On the convergence of leveraging. In: Advances in Neural Information Processing Systems 14

[40] Rockafellar RT (1970) Convex Analysis. Princeton University Press

[41] Rudin C, Schapire RE, Daubechies I (2007) Analysis of boosting algorithms using the smooth margin function. Annals of Statistics 35(6):2723–2768

[42] Schapire RE (1990) The strength of weak learnability. Machine Learning 5(2):197–227

[43] Schapire RE (1991) The design and analysis of efficient learning algorithms. PhD thesis, Massachusetts Institute of Technology, supervised by Ronald L. Rivest. Technical Report MIT/LCS/TR-493, MIT Laboratory for Computer Science

[44] Schapire RE (2001) Drifting games. Machine Learning 43(3):265–291

[45] Schapire RE (2010) The convergence rate of AdaBoost. In: The 23rd Conference on Learning Theory, open problem

[46] Schapire RE, Freund Y (2012) Boosting: Foundations and Algorithms. MIT Press

[47] Schapire RE, Singer Y (1999) Improved boosting algorithms using confidence-rated predictions. Machine Learning 37(3):297–336

[48] Schapire RE, Singer Y (2000) BoosTexter: A boosting-based system for text categorization. Machine Learning 39(2/3):135–168

[49] Schapire RE, Freund Y, Bartlett P, Lee WS (1998) Boosting the margin: A new explanation for the effectiveness of voting methods. Annals of Statistics 26(5):1651–1686

[50] Shalev-Shwartz S, Singer Y (2008) On the equivalence of weak learnability and linear separability: New relaxations and efficient boosting algorithms. In: 21st Annual Conference on Learning Theory

[51] Spencer J (1992) Ulam's searching game with a fixed number of lies. Theoret Comput Sci 95(2):307–321

[52] Telgarsky M (2011) The convergence rate of AdaBoost and friends, http://arxiv.org/abs/1101.4752

[53] Tewari A, Bartlett PL (2007) On the Consistency of Multiclass Classification Methods. Journal of Machine Learning Research 8:1007–1025

[54] Wu X, Kumar V, Quinlan R, Ghosh J, Yang Q, Motoda H, Mclachlan G, Ng A, Liu B, Yu P, Zhou ZH, Steinbach M, Hand D, Steinberg D (2008) Top 10 algorithms in data mining. Knowledge and Information Systems 14(1):1–37

[55] Zhang T (2004) Statistical behavior and consistency of classification methods based on convex risk minimization. Annals of Statistics 32(1):56–134

[56] Zhang T, Yu B (2005) Boosting with early stopping: Convergence and consistency. Annals of Statistics 33(4):1538–1579

[57] Zhu J, Zou H, Rosset S, Hastie T (2009) Multi-class AdaBoost. Statistics and Its Interface 2:349360