Journal of Machine Learning Research 5 (2004) 1557–1595 Submitted 3/2004; Published 12/2004

The Dynamics of AdaBoost: Cyclic Behavior and Convergence of Margins

Cynthia Rudin[*] CRUDIN@PRINCETON.EDU
Ingrid Daubechies INGRID@MATH.PRINCETON.EDU
Program in Applied and Computational Mathematics
Fine Hall
Washington Road
Princeton University
Princeton, NJ 08544-1000, USA

Robert E. Schapire SCHAPIRE@CS.PRINCETON.EDU
Princeton University
Department of Computer Science
35 Olden St.
Princeton, NJ 08544, USA

Editor: Dana Ron

Abstract

In order to study the convergence properties of the AdaBoost algorithm, we reduce AdaBoost to a nonlinear iterated map and study the evolution of its weight vectors. This dynamical systems approach allows us to understand AdaBoost's convergence properties completely in certain cases; for these cases we find stable cycles, allowing us to explicitly solve for AdaBoost's output.

Using this unusual technique, we are able to show that AdaBoost does not always converge to a maximum margin combined classifier, answering an open question. In addition, we show that "non-optimal" AdaBoost (where the weak learning algorithm does not necessarily choose the best weak classifier at each iteration) may fail to converge to a maximum margin classifier, even if "optimal" AdaBoost produces a maximum margin. Also, we show that if AdaBoost cycles, it cycles among "support vectors", i.e., examples that achieve the same smallest margin.

Keywords: boosting, AdaBoost, dynamics, convergence, margins

1. Introduction

Boosting algorithms are currently among the most popular and most successful algorithms for pattern recognition tasks (such as text classification). AdaBoost (Freund and Schapire, 1997) was the first practical boosting algorithm, and due to its success, a number of similar boosting algorithms have since been introduced (see the review paper of Schapire, 2002, for an introduction, or the review paper of Meir and Rätsch, 2003). Boosting algorithms are designed to construct a "strong" classifier using only a training set and a "weak" learning algorithm. A "weak" classifier produced by the weak learning algorithm has a probability of misclassification that is slightly below 50%, i.e., each weak classifier is only required to perform slightly better than a random guess. A "strong"

∗. C. Rudin's present address is New York University / Howard Hughes Medical Institute, 4 Washington Place, Room 809, New York, NY 10003-6603, USA. Her present email is rudin@nyu.edu.

classifier has a much smaller probability of error on test data. Hence, these algorithms "boost" the weak learning algorithm to achieve a stronger classifier. In order to exploit the weak learning algorithm's advantage over random guessing, the data is reweighted (the relative importance of the training examples is changed) before running the weak learning algorithm at each iteration. That is, AdaBoost maintains a distribution (set of weights) over the training examples, and selects a weak classifier from the weak learning algorithm at each iteration. Training examples that were misclassified by the weak classifier at the current iteration then receive higher weights at the following iteration. The end result is a final combined classifier, given by a thresholded linear combination of the weak classifiers.

AdaBoost does not often seem to suffer from overfitting, even after a large number of iterations (Breiman, 1998; Quinlan, 1996). This lack of overfitting has been explained to some extent by the *margin theory* of Schapire, Freund, Bartlett, and Lee (1998). The *margin* of a boosted classifier is a number between -1 and 1, that according to the margin theory, can be thought of as a confidence measure of a classifier's predictive ability, or as a guarantee on the generalization performance. If the margin of a classifier is large, then it tends to perform well on test data. If the margin is small, then the classifier tends not to perform so well. (The margin of a boosted classifier is also called the *minimum margin over training examples*.) Although the empirical success of a boosting algorithm depends on many factors (e.g., the type of data and how noisy it is, the capacity of the weak learning algorithm, the number of boosting iterations, regularization, entire margin distribution over the training examples), the margin theory does provide a reasonable explanation (though not a complete explanation) of AdaBoost's success, both empirically and theoretically.

Since the margin tends to give a strong indication of a classifier's performance in practice, a natural goal is to find classifiers that achieve a maximum margin. Since the AdaBoost algorithm was invented before the margin theory, the algorithm became popular due to its practical success rather than for its theoretical success (its ability to achieve large margins). Since AdaBoost was not specifically designed to maximize the margin, the question remained whether in fact it does actually maximize the margin. The objective function that AdaBoost minimizes (the exponential loss) is not related to the margin in the sense that one can minimize the exponential loss while simultaneously achieving an arbitrarily bad (small) margin. Thus, AdaBoost does not, in fact, optimize a cost function of the margins (see also Wyner, 2002). It was shown analytically that AdaBoost produces *large* margins, namely, Schapire et al. (1998) showed that AdaBoost achieves at least half of the maximum margin, and Rätsch and Warmuth (2002) have recently tightened this bound slightly. However, because AdaBoost does not necessarily make progress towards increasing the margin at each iteration, the usual techniques for analyzing coordinate algorithms do not apply; for all the extensive theoretical and empirical study of AdaBoost prior to the present work, it remained unknown whether or not AdaBoost always achieves a maximum margin solution.

A number of other boosting algorithms emerged over the past few years that aim more explicitly to maximize the margin at each iteration, such as AdaBoost* (Rätsch and Warmuth, 2002), arc-gv (Breiman, 1999), Coordinate Ascent Boosting and Approximate Coordinate Ascent Boosting (Rudin et al., 2004c,b,a; Rudin, 2004), the linear programming (LP) boosting algorithms including LP-AdaBoost (Grove and Schuurmans, 1998) and LPBoost (Demiriz et al., 2002). (Also see the ε-boosting literature, for example, Rosset et al., 2004.) However, AdaBoost is still used in practice, because it often empirically seems to produce maximum margin classifiers with low generalization error. In fact, under tightly controlled tests, it was shown empirically that the maximum margin algorithms arc-gv and LP-AdaBoost tend to perform *worse* than AdaBoost (Breiman, 1999; Grove

and Schuurmans, 1998). In the experiments of Grove and Schuurmans (1998), AdaBoost achieved margins that were almost as large, (but not quite as large) as those of the LP algorithms when stopped after a large number of iterations, yet often achieved lower generalization error. AdaBoost is also easy to program, and in our trials, it seems to converge the fastest (with respect to the margin) among the coordinate-based boosting algorithms.

Another surprising result of empirical trials is that AdaBoost does seem to be converging to maximum margin solutions *asymptotically* in the numerical experiments of Grove and Schuurmans (1998) and Rätsch and Warmuth (2002). Grove and Schuurmans have questioned whether AdaBoost is simply a "general, albeit very slow, LP solver". If AdaBoost is simply a margin-maximization algorithm, then why are other algorithms that achieve the same margin performing worse than AdaBoost? Is AdaBoost simply a fancy margin-maximization algorithm in disguise, or is it something more? As we will see, the answers are sometimes yes and sometimes no. So clearly the margins do not tell the whole story.

AdaBoost, as shown repeatedly (Breiman, 1997; Friedman et al., 2000; Rätsch et al., 2001; Duffy and Helmbold, 1999; Mason et al., 2000), is actually a coordinate descent algorithm on a particular exponential loss function. However, minimizing this function in other ways does not necessarily achieve large margins; the process of coordinate descent must be somehow responsible. Hence, we look to AdaBoost's dynamics to understand the process by which the margin is generated.

In this work, we took an unusual approach to this problem. We simplified AdaBoost to reveal a nonlinear iterated map for AdaBoost's weight vector. This iterated map gives a direct relation between the weights at time t and the weights at time $t+1$, including renormalization, and thus provides a much more concise mapping than the original algorithm. We then analyzed this dynamical system in specific cases. Using a small toolbox of techniques for analyzing dynamical systems, we were able to avoid the problem that progress (with respect to the margin) does not occur at every iteration. Instead, we measure progress another way; namely, via the convergence towards limit cycles.

To explain this way of measuring progress more clearly, we have found that for some specific cases, the weight vector of AdaBoost produces limit cycles that can be analytically stated, and are stable. When stable limit cycles exist, the convergence of AdaBoost can be understood. Thus, we are able to provide the key to answering the question of AdaBoost's convergence to maximum margin solutions: a collection of examples in which AdaBoost's convergence can be completely understood.

Using a very low-dimensional example (8×8, i.e., 8 weak classifiers and 8 training examples), we are able to show that AdaBoost does not always produce a maximum margin solution, finally answering the open question.

There are two interesting cases governing the dynamics of AdaBoost: the case where the optimal weak classifiers are chosen at each iteration (the "optimal" case), and the case where permissible non-optimal weak classifiers may be chosen (the "non-optimal" case). In the optimal case (which is the case we usually consider), the weak learning algorithm is required to choose a weak classifier that has the largest edge at every iteration, where the edge measures the performance of the weak learning algorithm. In the non-optimal case, the weak learning algorithm may choose any weak classifier as long as its edge exceeds ρ, the maximum margin achievable by a combined classifier. This is a natural notion of non-optimality for boosting, thus it provides a natural sense in which to measure robustness. Based on large scale experiments and a gap in theoretical bounds, Rätsch and Warmuth (2002) conjectured that AdaBoost does not necessarily converge to a maximum margin

classifier in the non-optimal case, i.e., that AdaBoost is not robust in this sense. In practice, the weak classifiers are generated by CART or another weak learning algorithm, implying that the choice need not always be optimal.

In Section 8, we show this conjecture to be true using a 4×5 example. That is, we show that "non-optimal AdaBoost" (AdaBoost in the non-optimal case) may not converge to a maximum margin solution, even in cases where "optimal AdaBoost" does.

Empirically, we have found very interesting and remarkable cyclic dynamics in many different low-dimensional cases (many more cases than the ones analyzed in this paper), for example, those illustrated in Figure 6. In fact, we have empirically found that AdaBoost produces cycles on randomly generated matrices – even on random matrices with hundreds of dimensions. On low-dimensional random matrices, cycles are almost always produced in our experiments. Thus, the story of AdaBoost's dynamics does not end with the margins; it is important to study AdaBoost's dynamics in more general cases where these cycles occur in order to understand its convergence properties.

To this extent, we prove that if AdaBoost cycles, it cycles only among a set of "support vectors" that achieve the same smallest margin among training examples. In this sense, we confirm observations of Caprile et al. (2002) who previously studied the dynamical behavior of boosting, and who also identified two sorts of examples which they termed "easy" and "hard." In addition, we give sufficient conditions for AdaBoost to achieve a maximum margin solution when cycling occurs. We also show that AdaBoost treats identically classified examples as one example, in the sense we will describe in Section 6. In Section 10, we discuss a case in which AdaBoost exhibits indications of chaotic behavior, namely sensitivity to initial conditions, and movement into and out of cyclic behavior.

We proceed as follows. In Section 2 we introduce some notation and state the AdaBoost algorithm. Then in Section 3 we decouple the dynamics for AdaBoost in the binary case so that we have a nonlinear iterated map. In Section 4, we analyze these dynamics for a simple case: the case where each weak classifier has one misclassified training example. In a 3×3 example, we find that the weight vectors always converge to one of two stable limit cycles, allowing us to calculate AdaBoost's output vector directly. From this, we can prove the output of AdaBoost yields the best possible margin. We generalize this case to $m \times m$ in Section 5. In Section 6 we discuss identically classified examples. Namely, we show that the weights on identically classified training examples can be shifted among these examples while preserving the cycle; that is, manifolds of stable cycles can occur. For an extension of the simple 3×3 case, we show that manifolds of cycles exist and are stable. In Section 7 we show that the training examples AdaBoost cycles upon are "support vectors" in that they all achieve the same margin. In the process, we provide a formula to directly calculate the margin from the cycle parameters. We also give sufficient conditions for AdaBoost to produce a maximum margin classifier when cycling occurs. Then in Section 8 we produce an example to show non-robustness of AdaBoost in the non-optimal case. In Section 9, we produce the example discussed above to show that AdaBoost may not converge to a maximum margin solution. And finally in Section 10, we provide a case for which AdaBoost exhibits indications of chaotic behavior.

2. Notation and Introduction to AdaBoost

The training set consists of examples with labels $\{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,m}$, where $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \{-1, 1\}$. The space $\mathcal{X}$ never appears explicitly in our calculations. Let $\mathcal{H} = \{h_1, \ldots, h_n\}$ be the set of all possible weak classifiers that can be produced by the weak learning algorithm, where $h_j : \mathcal{X} \to \{1, -1\}$. We assume that if h_j appears in $\mathcal{H}$, then $-h_j$ also appears in $\mathcal{H}$. Since our classifiers are binary, and since we restrict our attention to their behavior on a finite training set, we can assume the number of weak classifiers n is finite. We typically think of n as being very large, $m \ll n$, which makes a gradient descent calculation impractical because n, the number of dimensions, is too large; hence, AdaBoost uses coordinate descent instead, where only one weak classifier is chosen at each iteration.

We define an $m \times n$ matrix $\mathbf{M}$ where $M_{ij} = y_i h_j(\mathbf{x}_i)$, i.e., $M_{ij} = +1$ if training example i is classified correctly by weak classifier h_j, and -1 otherwise. We assume that no column of $\mathbf{M}$ has all +1's, that is, no weak classifier can classify all the training examples correctly. (Otherwise the learning problem is trivial. In this case, AdaBoost will have an undefined step size.) Although $\mathbf{M}$ is too large to be explicitly constructed in practice, mathematically, it acts as the only "input" to AdaBoost in this notation, containing all the necessary information about the weak learning algorithm and training examples.

AdaBoost computes a set of coefficients over the weak classifiers. At iteration t, the (unnormalized) coefficient vector is denoted $\boldsymbol{\lambda}_t$; i.e., the coefficient of weak classifier h_j determined by AdaBoost at iteration t is $\lambda_{t,j}$. The final combined classifier that AdaBoost outputs is $f_{\boldsymbol{\lambda}_{tmax}}$ given via $\boldsymbol{\lambda}_{t_{max}} / \|\boldsymbol{\lambda}_{t_{max}}\|_1$:

$$f_{\boldsymbol{\lambda}} = \frac{\sum_{j=1}^{n} \lambda_j h_j}{\|\boldsymbol{\lambda}\|_1} \quad \text{where} \quad \|\boldsymbol{\lambda}\|_1 = \sum_{j=1}^{n} |\lambda_j|.$$

In the specific examples we provide, either h_j or $-h_j$ remains unused over the course of AdaBoost's iterations, so all values of $\lambda_{t,j}$ are non-negative. The *margin of training example i* is defined by $y_i f_{\boldsymbol{\lambda}}(\mathbf{x}_i)$. Informally, one can think of the margin of a training example as the distance (by some measure) from the example to the decision boundary, $\{\mathbf{x} : f_{\boldsymbol{\lambda}}(\mathbf{x}) = 0\}$.

A boosting algorithm maintains a distribution, or set of weights, over the training examples that is updated at each iteration t. This distribution is denoted $\mathbf{d}_t \in \Delta_m$, and $\mathbf{d}_t^T$ is its transpose. Here, Δ_m denotes the simplex of m-dimensional vectors with non-negative entries that sum to 1. At each iteration t, a weak classifier h_{j_t} is selected by the weak learning algorithm. The *probability of error* at iteration t, denoted d_-, for the selected weak classifier h_{j_t} on the training examples (weighted by $\mathbf{d}_t$) is $\sum_{\{i:M_{ij_t}=-1\}} d_{t,i}$. Also, denote $d_+ := 1 - d_-$. Note that d_+ and d_- depend on t; although we have simplified the notation, the iteration number will be clear from the context. The *edge* of weak classifier j_t at time t with respect to the training examples is $(\mathbf{d}_t^T \mathbf{M})_{j_t}$, which can be written as

$$(\mathbf{d}_t^T \mathbf{M})_{j_t} = \sum_{i:M_{ij_t}=1} d_{t,i} - \sum_{i:M_{ij_t}=-1} d_{t,i} = d_+ - d_- = 1 - 2d_-.$$

Thus, a smaller edge indicates a higher probability of error. For the *optimal* case (the case we usually consider), we will require the weak learning algorithm to give us the weak classifier with the largest possible edge at each iteration,

$$j_t \in \operatorname*{argmax}_j (\mathbf{d}_t^T \mathbf{M})_j,$$

i.e., j_t is the weak classifier that performs the best on the training examples weighted by $\mathbf{d}_t$. For the *non-optimal* case (which we consider in Section 8), we only require a weak classifier whose edge exceeds ρ, where ρ is the largest possible margin that can be attained for $\mathbf{M}$, i.e.,

$$j_t \in \{j : (\mathbf{d}_t^T \mathbf{M})_j \geq \rho\}.$$

(The value ρ is defined formally below.) The edge for the chosen weak classifier j_t at iteration t is denoted r_t, i.e., $r_t = (\mathbf{d}_t^T \mathbf{M})_{j_t}$. Note that $d_+ = (1+r_t)/2$ and $d_- = (1-r_t)/2$.

The margin theory developed via a set of generalization bounds that are based on the margin distribution of the training examples (Schapire et al., 1998; Koltchinskii and Panchenko, 2002). These bounds can be reformulated (in a slightly weaker form) in terms of the minimum margin, which was the focus of previous work by Breiman (1999), Grove and Schuurmans (1998), and Rätsch and Warmuth (2002). Thus, these bounds suggest maximizing the minimum margin over training examples to achieve a low probability of error over test data. Hence, our goal is to find a normalized vector $\tilde{\boldsymbol{\lambda}} \in \Delta_n$ that maximizes the minimum margin over training examples, $\min_i (\mathbf{M}\tilde{\boldsymbol{\lambda}})_i$ (or equivalently $\min_i y_i f_{\boldsymbol{\lambda}}(\mathbf{x}_i)$). That is, we wish to find a vector

$$\tilde{\boldsymbol{\lambda}} \in \operatorname*{argmax}_{\boldsymbol{\lambda} \in \Delta_n} \min_i (\mathbf{M}\bar{\boldsymbol{\lambda}})_i.$$

We call this minimum margin over training examples (i.e., $\min_i (\mathbf{M}\boldsymbol{\lambda})_i / \|\boldsymbol{\lambda}\|_1$) the ℓ_1-*margin* or simply *margin* of classifier $\boldsymbol{\lambda}$. Any training example that achieves this minimum margin will be called a *support vector*. Due to the von Neumann Min-Max Theorem for 2-player zero-sum games,

$$\min_{\mathbf{d} \in \Delta_m} \max_j (\mathbf{d}^T \mathbf{M})_j = \max_{\tilde{\boldsymbol{\lambda}} \in \Delta_n} \min_i (\mathbf{M}\tilde{\boldsymbol{\lambda}})_i.$$

That is, the minimum value of the edge (left hand side) corresponds to the maximum value of the margin (i.e., the maximum value of the minimum margin over training examples, right hand side). We denote this value by ρ. One can think of ρ as measuring the worst performance of the best combined classifier, $\min_i (\mathbf{M}\tilde{\boldsymbol{\lambda}})_i$.

The "unrealizable" or "non-separable" case where $\rho = 0$ is fully understood (Collins et al., 2002). For this work, we assume $\rho > 0$ and study the less understood "realizeable" or "separable" case. In both the non-separable and separable cases, AdaBoost converges to a minimizer of the empirical loss function

$$F(\boldsymbol{\lambda}) := \sum_{i=1}^{m} e^{-(\mathbf{M}\boldsymbol{\lambda})_i}.$$

In the non-separable case, the $\mathbf{d}_t$'s converge to a fixed vector (Collins et al., 2002). In the separable case, the $\mathbf{d}_t$'s cannot converge to a fixed vector, and the minimum value of F is 0, occurring as $||\boldsymbol{\lambda}||_1 \to \infty$. It is important to appreciate that this tells us nothing about the value of the margin achieved by AdaBoost or any other procedure designed to minimize F. To see why, consider any $\bar{\boldsymbol{\lambda}} \in \Delta_n$ such that $(\mathbf{M}\bar{\boldsymbol{\lambda}})_i > 0$ for all i (assuming we are in the separable case so such a $\bar{\boldsymbol{\lambda}}$ exists). Then $\lim_{a \to \infty} a\bar{\boldsymbol{\lambda}}$ will produce a minimum value for F, but the original normalized $\bar{\boldsymbol{\lambda}}$ need not yield a maximum margin. To clarify, any normalized $\bar{\boldsymbol{\lambda}}$ for which $(\mathbf{M}\bar{\boldsymbol{\lambda}})_i > 0$ for all i produces a classifier that classifies all training examples correctly, has unnormalized counterparts that attain values of F arbitrarily close to 0, yet may produce a classifier with arbitrarily *small* margin. In other words, an arbitrary algorithm that minimizes F can achieve an arbitrarily bad margin. So it must be the *process*

AdaBoost ("optimal" case):

1. **Input:** Matrix $\mathbf{M}$, No. of iterations t_{max}
2. **Initialize:** $\lambda_{1,j} = 0$ for $j = 1,...,n$
3. **Loop for** $t = 1,...,t_{max}$
   (a) $d_{t,i} = e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_i} / \sum_{\tilde{i}=1}^{m} e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_{\tilde{i}}}$ for $i = 1,...,m$
   (b) $j_t \in \operatorname{argmax}_j (\mathbf{d}_t^T \mathbf{M})_j$
   (c) $r_t = (\mathbf{d}_t^T \mathbf{M})_{j_t}$
   (d) $\alpha_t = \frac{1}{2} \ln\left(\frac{1+r_t}{1-r_t}\right)$
   (e) $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \alpha_t \mathbf{e}_{j_t}$, where $\mathbf{e}_{j_t}$ is 1 in position j_t and 0 elsewhere.
4. **Output:** $\boldsymbol{\lambda}_{t_{max}} / \|\boldsymbol{\lambda}_{t_{max}}\|_1$

Figure 1: Pseudocode for the AdaBoost algorithm.

of coordinate descent that awards AdaBoost its ability to increase margins, not simply AdaBoost's ability to minimize F. The value of the function F tells us very little about the value of the margin; even asymptotically, it only tells us whether the margin is positive or not.

Figure 1 shows pseudocode for the AdaBoost algorithm. Usually the $\boldsymbol{\lambda}_1$ vector is initialized to zero, so that all the training examples are weighted equally during the first iteration. The weight vector $\mathbf{d}_t$ is adjusted so that training examples that were misclassified at the previous iteration are weighted more highly, so they are more likely to be correctly classified at the next iteration. The weight vector $\mathbf{d}_t$ is determined from the vector of coefficients $\boldsymbol{\lambda}_t$, which has been updated. The map from $\mathbf{d}_t$ to $\mathbf{d}_{t+1}$ also involves renormalization, so it is not a very direct map when written in this form. Thus on each round of boosting, the distribution $\mathbf{d}_t$ is updated and renormalized (Step 3a), classifier j_t with maximum edge (minimum probability of error) is selected (Step 3b), and the weight of that classifier is updated (Step 3e). Note that $\lambda_{t,j} = \sum_{\tilde{t}=1}^{t} \alpha_{\tilde{t}} \mathbf{1}_{j_{\tilde{t}}=j}$ where $\mathbf{1}_{j_{\tilde{t}}=j}$ is 1 if $j_{\tilde{t}} = j$ and 0 otherwise.

3. The Iterated Map Defined By AdaBoost

AdaBoost can be reduced to an iterated map for the $\mathbf{d}_t$'s, as shown in Figure 2. This map gives a direct relationship between $\mathbf{d}_t$ and $\mathbf{d}_{t+1}$, taking the normalization of Step 3a into account automatically. For the cases considered in Sections 4, 5, and 6, we only need to understand the dynamics of Figure 2 in order to compute the final coefficient vector that AdaBoost will output. Initialize $d_{1,i} = 1/m$ for $i = 1,...,m$ as in the first iteration of AdaBoost. Also recall that all values of r_t are nonnegative since $r_t \geq \rho > 0$.

To show the equivalence with AdaBoost, consider the iteration defined by AdaBoost and reduce as follows. Since:

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1+r_t}{1-r_t}\right), \text{ we have } e^{-(M_{ij_t}\alpha_t)} = \left(\frac{1-r_t}{1+r_t}\right)^{\frac{1}{2}M_{ij_t}} = \left(\frac{1-M_{ij_t}r_t}{1+M_{ij_t}r_t}\right)^{\frac{1}{2}}.$$

Iterated Map Defined by AdaBoost

1. $j_t \in \operatorname{argmax}_j (\mathbf{d}_t^T \mathbf{M})_j$
2. $r_t = (\mathbf{d}_t^T \mathbf{M})_{j_t}$
3. $d_{t+1,i} = \frac{d_{t,i}}{1+M_{ij_t} r_t}$ for $i = 1, \ldots, m$

Figure 2: The nonlinear iterated map obeyed by AdaBoost's weight vectors. This dynamical system provides a direct map from $\mathbf{d}_t$ to $\mathbf{d}_{t+1}$.

Here, we have used the fact that $\mathbf{M}$ is a binary matrix. The iteration defined by AdaBoost combined with the equation above yields:

$$d_{t+1,i} = \frac{e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_i} e^{-(M_{ij_t}\alpha_t)}}{\sum_{\bar{i}=1}^m e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_{\bar{i}}} e^{-(M_{\bar{i}j_t}\alpha_t)}} = \frac{d_{t,i}}{\sum_{\bar{i}=1}^m d_{t,\bar{i}} \left(\frac{1-M_{\bar{i}j_t} r_t}{1+M_{\bar{i}j_t} r_t}\right)^{\frac{1}{2}} \left(\frac{1+M_{ij_t} r_t}{1-M_{ij_t} r_t}\right)^{\frac{1}{2}}}.$$

Here, we have divided numerator and denominator by $\sum_{\bar{i}=1}^m e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_{\bar{i}}}$. For each i such that $M_{ij_t} = 1$, we find:

$$\begin{aligned} d_{t+1,i} &= \frac{d_{t,i}}{\sum_{\{\bar{i}:M_{\bar{i}j_t}=1\}} d_{t,\bar{i}} \left(\frac{1-r_t}{1+r_t}\right)^{\frac{1}{2}} \left(\frac{1+r_t}{1-r_t}\right)^{\frac{1}{2}} + \sum_{\{\bar{i}:M_{\bar{i}j_t}=-1\}} d_{t,\bar{i}} \left(\frac{1+r_t}{1-r_t}\right)^{\frac{1}{2}} \left(\frac{1+r_t}{1-r_t}\right)^{\frac{1}{2}}} \\ &= \frac{d_{t,i}}{d_+ + d_- \left(\frac{1+r_t}{1-r_t}\right)} = \frac{d_{t,i}}{\frac{1+r_t}{2} + \frac{1-r_t}{2} \left(\frac{1+r_t}{1-r_t}\right)} = \frac{d_{t,i}}{1+r_t}. \end{aligned}$$

Likewise, for each i such that $M_{ij_t} = -1$, we find $d_{t+1,i} = \frac{d_{t,i}}{1-r_t}$. Thus our reduction is complete. To check that $\sum_{i=1}^m d_{t+1,i} = 1$, i.e., that renormalization has been taken into account by the iterated map, we calculate:

$$\sum_{i=1}^m d_{t+1,i} = \frac{1}{1+r_t} d_+ + \frac{1}{1-r_t} d_- = \frac{(1+r_t)}{2(1+r_t)} + \frac{(1-r_t)}{2(1-r_t)} = 1.$$

For the iterated map, fixed points (rather than cycles or other dynamics) occur when the training data fails to be separable by the set of weak classifiers. In that case, the analysis of Collins, Schapire, and Singer (2002) shows that the iterated map will converge to a fixed point, and that the $\boldsymbol{\lambda}_t's$ will asymptotically attain the minimum value of the convex function $F(\boldsymbol{\lambda}) := \sum_{i=1}^m e^{-(\mathbf{M}\boldsymbol{\lambda})_i}$, which is strictly positive in the non-separable case. Consider the possibility of fixed points for the $\mathbf{d}_t$'s in the separable case $\rho > 0$. From our dynamics, we can see that this is not possible, since $r_t \geq \rho > 0$ and for any i such that $d_{t,i} > 0$,

$$d_{t+1,i} = \frac{d_{t,i}}{(1+M_{i,j_t} r_t)} \neq d_{t,i}.$$

Thus, we have shown that AdaBoost does not produce fixed points in the separable case.

4. The Dynamics of AdaBoost in the Simplest Case : The 3×3 Case

In this section, we will introduce a simple 3×3 input matrix (in fact, the simplest non-trivial matrix) and analyze the convergence of AdaBoost in this case, using the iterated map of Section 3. We will show that AdaBoost does produce a maximum margin solution, remarkably through convergence to one of two stable limit cycles. We extend this example to the $m \times m$ case in Section 5, where AdaBoost produces at least $(m-1)!$ stable limit cycles, each corresponding to a maximum margin solution. We will also extend this example in Section 6 to include manifolds of cycles.

Consider the input matrix

$$\mathbf{M} = \begin{pmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix}$$

corresponding to the case where each classifier misclassifies one of three training examples. We could add columns to include the negated version of each weak classifier, but those columns would never be chosen by AdaBoost, so they have been removed for simplicity. The value of the margin for the best combined classifier defined by $\mathbf{M}$ is $1/3$. How will AdaBoost achieve this result? We will proceed step by step.

Assume we are in the optimal case, where $j_t \in \text{argmax}_j(\mathbf{d}_t^T\mathbf{M})_j$. Consider the dynamical system on the simplex Δ_3 defined by our iterated map in Section 3. In the triangular region with vertices $(0,0,1),(\frac{1}{3},\frac{1}{3},\frac{1}{3}),(0,1,0)$, j_t will be 1 for Step 1 of the iterated map. That is, within this region, $d_{t,1} < d_{t,2}$ and $d_{t,1} < d_{t,3}$, so j_t will be 1. Similarly, we have regions for $j_t = 2$ and $j_t = 3$ (see Figure 3(a)).

AdaBoost was designed to set the edge of the previous weak classifier to 0 at each iteration, that is, $\mathbf{d}_{t+1}$ will always satisfy $(\mathbf{d}_{t+1}^T\mathbf{M})_{j_t} = 0$. To see this using the iterated map,

$$\begin{aligned}(\mathbf{d}_{t+1}^T\mathbf{M})_{j_t} &= \sum_{\{i:M_{ij_t}=1\}} d_{t,i}\frac{1}{1+r_t} - \sum_{\{i:M_{ij_t}=-1\}} d_{t,i}\frac{1}{1-r_t} \\ &= d_+\frac{1}{1+r_t} - d_-\frac{1}{1-r_t} = \frac{1+r_t}{2}\frac{1}{1+r_t} - \frac{1-r_t}{2}\frac{1}{1-r_t} = 0. \end{aligned} \quad (1)$$

This implies that after the first iteration, the $\mathbf{d}_t$'s are restricted to

$$\{\mathbf{d} : [(\mathbf{d}^T\mathbf{M})_1 = 0] \bigcup [(\mathbf{d}^T\mathbf{M})_2 = 0] \bigcup [(\mathbf{d}^T\mathbf{M})_3 = 0]\}.$$

Thus, it is sufficient for our dynamical system to be analyzed on the edges of a triangle with vertices $\left(0,\frac{1}{2},\frac{1}{2}\right)$, $\left(\frac{1}{2},0,\frac{1}{2}\right)$, $\left(\frac{1}{2},\frac{1}{2},0\right)$ (see Figure 3(b)). That is, within one iteration, the 2-dimensional map on the simplex Δ_3 reduces to a 1-dimensional map on the edges of the triangle.

Consider the possibility of periodic cycles for the $\mathbf{d}_t$'s. If there are periodic cycles of length T, then the following condition must hold for $\mathbf{d}_1^{cyc},...,\mathbf{d}_T^{cyc}$ in the cycle: For each i, either

- $d_{1,i}^{cyc} = 0$, or
- $\prod_{t=1}^{T}(1 + M_{ij_t}r_t^{cyc}) = 1$,

where $r_t^{cyc} = (\mathbf{d}_t^{cycT}\mathbf{M})_{j_t}$. (As usual, $\mathbf{d}_t^{cycT} := (\mathbf{d}_t^{cyc})^T$, superscript T denotes transpose.) The statement above follows directly from the reduced map iterated T times. In fact, the first condition $d_{1,i}^{cyc} = 0$ implies $d_{t,i}^{cyc} = 0$ for all $t \in \{1,...,T\}$. We call the second condition the *cycle condition*.

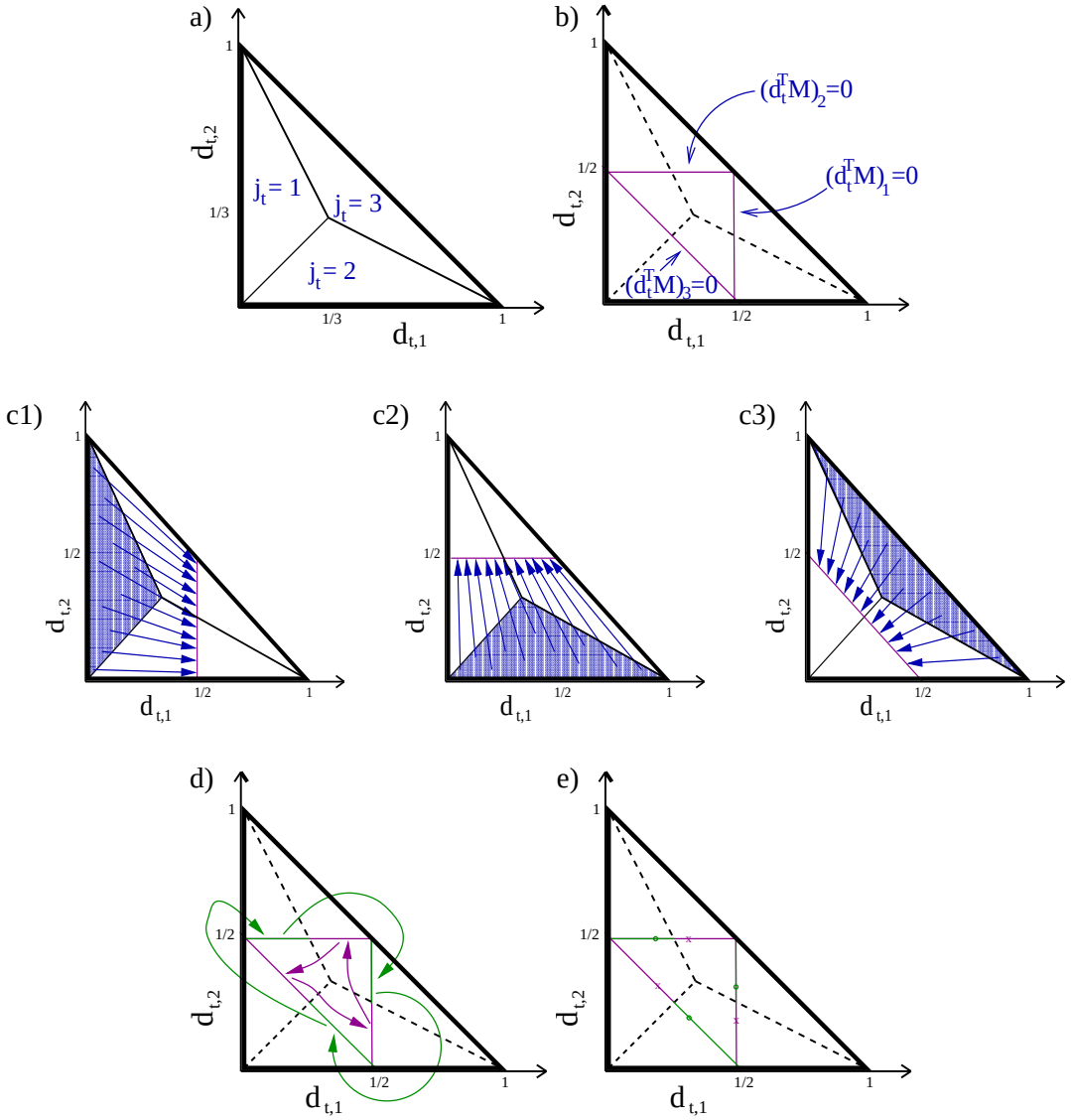

Figure 3: (a) Regions of $\mathbf{d}_t$-space where classifiers $j_t = 1, 2, 3$ will respectively be selected for Step 1 of the iterated map of Figure 2. Since $d_{t,3} = 1 - d_{t,2} - d_{t,1}$, this projection onto the first two coordinates $d_{t,1}$ and $d_{t,2}$ completely characterizes the map. (b) Regardless of the initial position $\mathbf{d}_1$, the weight vectors at all subsequent iterations $\mathbf{d}_2, \ldots, \mathbf{d}_{t_{max}}$ will be restricted to lie on the edges of the inner triangle which is labelled. (c1) Within one iteration, the triangular region where $j_t = 1$ maps to the line $\{\mathbf{d} : (\mathbf{d}^T \mathbf{M})_1 = 0\}$. The arrows indicate where various points in the shaded region will map at the following iteration. The other two regions have analogous dynamics as shown in (c2) and (c3). (d) There are six total subregions of the inner triangle (two for each of the three edges). Each subregion is mapped to the interior of another subregion as indicated by the arrows. (e) Coordinates for the two 3-cycles. The approximate positions $\mathbf{d}_1^{cyc}$, $\mathbf{d}_2^{cyc}$, and $\mathbf{d}_3^{cyc}$ for one of the 3-cycles are denoted by a small 'o', the positions for the other cycle are denoted by a small 'x'.

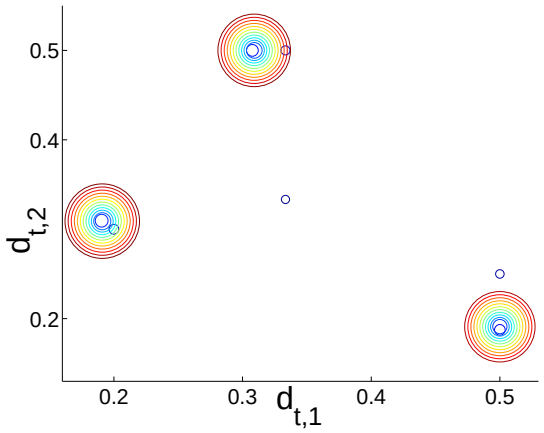

Figure 4: 50 iterations of AdaBoost showing convergence of $\mathbf{d}_t$'s to a cycle. Small rings indicate earlier iterations of AdaBoost, while larger rings indicate later iterations. There are many concentric rings at positions $\mathbf{d}_1^{cyc}$, $\mathbf{d}_2^{cyc}$, and $\mathbf{d}_3^{cyc}$.

Consider the possibility of a periodic cycle of length 3, cycling through each weak classifier once. For now, assume $j_1 = 1, j_2 = 2, j_3 = 3$, but without loss of generality one can choose $j_1 = 1, j_2 = 3, j_3 = 2$, which yields another cycle. Assume $d_{1,i}^{cyc} > 0$ for all i. From the cycle condition,

$$\begin{aligned}
1 &= (1+M_{ij_1}r_1^{cyc})(1+M_{ij_2}r_2^{cyc})(1+M_{ij_3}r_3^{cyc}) \quad \text{for } i=1,2, \text{ and } 3, \text{ i.e.,} \\
1 &= (1-r_1^{cyc})(1+r_2^{cyc})(1+r_3^{cyc}) \quad \text{for } i=1, && (2)\\
1 &= (1+r_1^{cyc})(1-r_2^{cyc})(1+r_3^{cyc}) \quad \text{for } i=2, && (3)\\
1 &= (1+r_1^{cyc})(1+r_2^{cyc})(1-r_3^{cyc}) \quad \text{for } i=3. && (4)
\end{aligned}$$

From (2) and (3),

$$(1-r_1^{cyc})(1+r_2^{cyc}) = (1+r_1^{cyc})(1-r_2^{cyc}),$$

thus $r_1^{cyc} = r_2^{cyc}$. Similarly, $r_2^{cyc} = r_3^{cyc}$ from (3) and (4), so $r_1^{cyc} = r_2^{cyc} = r_3^{cyc}$. Using either (2), (3), or (4) to solve for $r := r_1^{cyc} = r_2^{cyc} = r_3^{cyc}$ (taking positive roots since $r > 0$), we find the value of the edge for every iteration in the cycle to be equal to the golden ratio minus one, i.e.,

$$r = \frac{\sqrt{5}-1}{2}.$$

Now, let us solve for the weight vectors in the cycle, $\mathbf{d}_1^{cyc}$, $\mathbf{d}_2^{cyc}$, and $\mathbf{d}_3^{cyc}$. At $t = 2$, the edge with respect to classifier 1 is 0. Again, it is required that each $\mathbf{d}_t^{cyc}$ lies on the simplex Δ_3.

$$\begin{aligned}
(\mathbf{d}_2^{cyc\,T}\mathbf{M})_1 = 0 \quad &\text{and} \quad \sum_{i=1}^{3} d_{2,i}^{cyc} = 1, \quad \text{that is,} \\
-d_{2,1}^{cyc} + d_{2,2}^{cyc} + d_{2,3}^{cyc} = 0 \quad &\text{and} \quad d_{2,1}^{cyc} + d_{2,2}^{cyc} + d_{2,3}^{cyc} = 1, \\
\text{thus,} \quad d_{2,1}^{cyc} \quad &= \quad \frac{1}{2}.
\end{aligned}$$

Since $d^{cyc}_{2,1} = \frac{1}{2}$, we have $d^{cyc}_{2,2} = \frac{1}{2} - d^{cyc}_{2,3}$. At $t = 3$, the edge with respect to classifier 2 is 0. From the iterated map, we can write $\mathbf{d}^{cyc}_3$ in terms of $\mathbf{d}^{cyc}_2$.

$$0 = (\mathbf{d}^{cyc\,T}_3 \mathbf{M})_2 = \sum_{i=1}^{3} \frac{M_{i2} d^{cyc}_{2,i}}{1 + M_{i2} r} = \frac{\frac{1}{2}}{1+r} - \frac{\frac{1}{2} - d^{cyc}_{2,3}}{1-r} + \frac{d^{cyc}_{2,3}}{1+r}, \quad \text{so}$$

$$d^{cyc}_{2,3} = \frac{r}{2} = \frac{\sqrt{5}-1}{4} \quad \text{and thus} \quad d^{cyc}_{2,2} = \frac{1}{2} - d^{cyc}_{2,3} = \frac{3-\sqrt{5}}{4}.$$

Now that we have found $\mathbf{d}^{cyc}_2$, we can recover the rest of the cycle:

$$\begin{aligned}
\mathbf{d}^{cyc}_1 &= \left(\frac{3-\sqrt{5}}{4}, \frac{\sqrt{5}-1}{4}, \frac{1}{2} \right)^T, \\
\mathbf{d}^{cyc}_2 &= \left(\frac{1}{2}, \frac{3-\sqrt{5}}{4}, \frac{\sqrt{5}-1}{4} \right)^T, \\
\mathbf{d}^{cyc}_3 &= \left(\frac{\sqrt{5}-1}{4}, \frac{1}{2}, \frac{3-\sqrt{5}}{4} \right)^T.
\end{aligned}$$

To check that this actually is a cycle, starting from $\mathbf{d}^{cyc}_1$, AdaBoost will choose $j_t = 1$. Then $r_1 = (\mathbf{d}^{cyc\,T}_1 \mathbf{M})_1 = \frac{\sqrt{5}-1}{2}$. Now, computing $\frac{d^{cyc}_{1,i}}{1+M_{i,1} r_1}$ for all i yields $\mathbf{d}^{cyc}_2$. In this way, AdaBoost will cycle between weak classifiers $j = 1, 2, 3, 1, 2, 3$, etc.

The other 3-cycle can be determined similarly:

$$\begin{aligned}
\mathbf{d}^{cyc'}_1 &= \left(\frac{3-\sqrt{5}}{4}, \frac{1}{2}, \frac{\sqrt{5}-1}{4} \right)^T, \\
\mathbf{d}^{cyc'}_2 &= \left(\frac{1}{2}, \frac{\sqrt{5}-1}{4}, \frac{3-\sqrt{5}}{4} \right)^T, \\
\mathbf{d}^{cyc'}_3 &= \left(\frac{\sqrt{5}-1}{4}, \frac{3-\sqrt{5}}{4}, \frac{1}{2} \right)^T.
\end{aligned}$$

Since we always start from the initial condition $\mathbf{d}_1 = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)^T$, the initial choice of j_t is arbitrary; all three weak classifiers are within the argmax set in Step 1 of the iterated map. This arbitrary step, along with another arbitrary choice at the second iteration, determines which of the two cycles the algorithm will choose; as we will see, the algorithm must converge to one of these two cycles.

To show that these cycles are globally stable, we will show that the map is a contraction from each subregion of the inner triangle into another subregion. We only need to consider the one-dimensional map defined on the edges of the inner triangle, since within one iteration, every trajectory starting within the simplex Δ_3 lands somewhere on the edges of the inner triangle. The edges of the inner triangle consist of 6 subregions, as shown in Figure 3(d). We will consider one subregion, the segment from $\left(0, \frac{1}{2}, \frac{1}{2}\right)^T$ to $\left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right)^T$, or simply $\left(x, \frac{1}{2}, \frac{1}{2} - x\right)^T$ where $x \in (0, \frac{1}{4})$. (We choose not to deal with the endpoints since we will show they are unstable; thus the dynamics never reach or

converge to these points. For the first endpoint the map is not defined, and for the second, the map is ambiguous; not well-defined.) For this subregion $j_t = 1$, and the next iterate is

$$\left(\frac{x}{1-(1-2x)}, \frac{\frac{1}{2}}{1+(1-2x)}, \frac{\frac{1}{2}-x}{1+(1-2x)}\right)^T = \left(\frac{1}{2}, \frac{1}{4(1-x)}, \frac{1}{2} - \frac{1}{4(1-x)}\right)^T.$$

To compare the length of the new interval with the length of the previous interval, we use the fact that there is only one degree of freedom. A position on the previous interval can be uniquely determined by its first component $x \in (0, \frac{1}{4})$. A position on the new interval can be uniquely determined by its second component taking values $\frac{1}{4(1-x)}$, where we still have $x \in (0, \frac{1}{4})$. The map

$$x \mapsto \frac{1}{4(1-x)}$$

is a contraction. To see this, the slope of the map is $\frac{1}{4(1-x)^2}$, taking values within the interval $(\frac{1}{4}, \frac{4}{9})$. Thus the map is continuous and monotonic, with absolute slope strictly less than 1. The next iterate will appear within the interval $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right)^T$ to $\left(\frac{1}{2}, \frac{1}{3}, \frac{1}{6}\right)^T$, which is strictly contained within the subregion connecting $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right)^T$ with $\left(\frac{1}{2}, \frac{1}{2}, 0\right)^T$. Thus, we have a contraction. A similar calculation can be performed for each of the subregions, showing that each subregion maps monotonically to an area strictly within another subregion by a contraction map. Figure 3(d) illustrates the various mappings between subregions. After three iterations, each subregion maps by a monotonic contraction to a strict subset of itself. Thus, any fixed point of the three-iteration cycle must be the unique attracting fixed point for that subregion, and the domain of attraction for this point must be the whole subregion. In fact, there are six such fixed points, one for each subregion, three for each of the two cycles. The union of the domains of attraction for these fixed points is the whole triangle; every position $\mathbf{d}$ within the simplex Δ_3 is within the domain of attraction of one of these 3-cycles. Thus, these two cycles are globally stable.

Since the contraction is so strong at every iteration (as shown above, the absolute slope of the map is much less than 1), the convergence to one of these two 3-cycles is very fast. Figure 5(a) shows where each subregion of the "unfolded triangle" will map after the first iteration. The "unfolded triangle" is the interval obtained by traversing the triangle clockwise, starting and ending at $\left(0, \frac{1}{2}, \frac{1}{2}\right)$. Figure 5(b) illustrates that the absolute slope of the second iteration of this map at the fixed points is much less than 1; the cycles are strongly attracting.

The combined classifier that AdaBoost will output is

$$\boldsymbol{\lambda}_{\text{combined}} = \frac{\left(\frac{1}{2}\ln\left(\frac{1+r_1^{cyc}}{1-r_1^{cyc}}\right), \frac{1}{2}\ln\left(\frac{1+r_2^{cyc}}{1-r_2^{cyc}}\right), \frac{1}{2}\ln\left(\frac{1+r_3^{cyc}}{1-r_3^{cyc}}\right)\right)^T}{\text{normalization constant}} = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)^T,$$

and since $\min_i(\mathbf{M}\boldsymbol{\lambda}_{\text{combined}})_i = \frac{1}{3}$, we see that AdaBoost always produces a maximum margin solution for this input matrix.

Thus, we have derived our first convergence proof for AdaBoost in a specific separable case. We have shown that at least in some cases, AdaBoost is in fact a margin-maximizing algorithm. We summarize this first main result.

Theorem 1 *For the* 3×3 *matrix* $\boldsymbol{M}$*:*

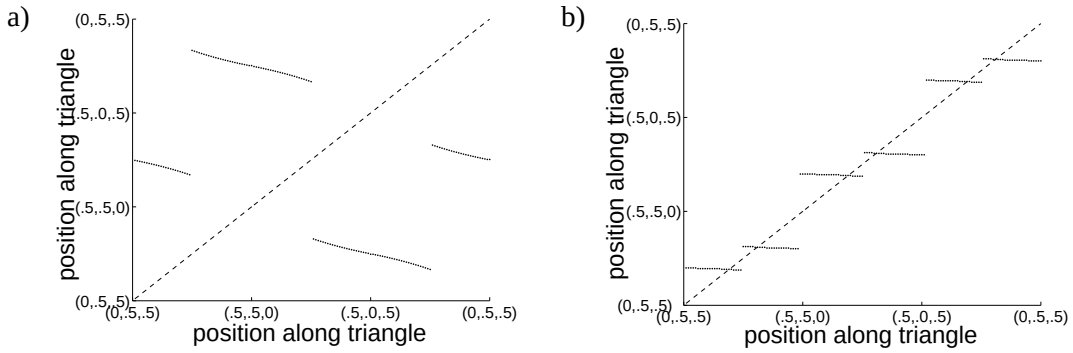

Figure 5: (a) The iterated map on the unfolded triangle. Both axes give coordinates on the edges of the inner triangle in Figure 3(b). The plot shows where $\mathbf{d}_{t+1}$ will be, given $\mathbf{d}_t$. (b) The map from (a) iterated twice, showing where $\mathbf{d}_{t+3}$ will be, given $\mathbf{d}_t$. For this "triple map", there are 6 stable fixed points, 3 for each cycle.

- *The weight vectors* $\mathbf{d}_t$ *converge to one of two possible stable cycles. The coordinates of the cycles are:*

$$\begin{aligned}
\mathbf{d}_1^{cyc} &= \left(\frac{3-\sqrt{5}}{4}, \frac{\sqrt{5}-1}{4}, \frac{1}{2}\right)^T, \\
\mathbf{d}_2^{cyc} &= \left(\frac{1}{2}, \frac{3-\sqrt{5}}{4}, \frac{\sqrt{5}-1}{4}\right)^T, \\
\mathbf{d}_3^{cyc} &= \left(\frac{\sqrt{5}-1}{4}, \frac{1}{2}, \frac{3-\sqrt{5}}{4}\right)^T,
\end{aligned}$$

and

$$\begin{aligned}
\mathbf{d}_1^{cyc'} &= \left(\frac{3-\sqrt{5}}{4}, \frac{1}{2}, \frac{\sqrt{5}-1}{4}\right)^T, \\
\mathbf{d}_2^{cyc'} &= \left(\frac{1}{2}, \frac{\sqrt{5}-1}{4}, \frac{3-\sqrt{5}}{4}\right)^T, \\
\mathbf{d}_3^{cyc'} &= \left(\frac{\sqrt{5}-1}{4}, \frac{3-\sqrt{5}}{4}, \frac{1}{2}\right)^T.
\end{aligned}$$

- *AdaBoost produces a maximum margin solution for this matrix* $\boldsymbol{M}$*.*

5. Generalization to m Classifiers, Each with One Misclassified Example

This simple 3 classifier case can be generalized to m classifiers, each having one misclassified training example; we will find solutions of a similar nature to the ones we found for the 3×3 case,

where there is a rotation of the coordinates at every iteration and a contraction. Here,

$$\mathbf{M} = \begin{pmatrix} -1 & 1 & 1 & \cdots & 1 \\ 1 & -1 & 1 & \cdots & 1 \\ 1 & 1 & -1 & & \vdots \\ \vdots & & & \ddots & 1 \\ 1 & \cdots & \cdots & 1 & -1 \end{pmatrix}.$$

Theorem 2 *For the $m \times m$ matrix above:*

- *The dynamical system for AdaBoost's weight vectors contains at least $(m-1)!$ stable periodic cycles of length m.*
- *AdaBoost converges to a maximum margin solution when the weight vectors converge to one of these cycles.*

The proof of Theorem 2 can be found in Appendix A.

6. Identically Classified Examples and Manifolds of Cycles

In this section, we show how manifolds of cycles appear automatically from cyclic dynamics when there are sets of identically classified training examples. We show that the manifolds of cycles that arise from a variation of the 3×3 case are stable. One should think of a "manifold of cycles" as a continuum of cycles; starting from a position on any cycle, if we move along the directions defined by the manifold, we will find starting positions for infinitely many other cycles. These manifolds are interesting from a theoretical viewpoint. In addition, their existence and stability will be an essential part of the proof of Theorem 7.

A set of training examples $\overline{I}$ is *identically classified* if each pair of training examples i and i' contained in $\overline{I}$ satisfy $y_i h_j(\mathbf{x}_i) = y_{i'} h_j(\mathbf{x}_{i'}) \; \forall j$. That is, the rows i and i' of matrix $\mathbf{M}$ are identical; training examples i and i' are misclassified by the same set of weak classifiers. When AdaBoost cycles, it treats each set of identically classified training examples as one training example, in a specific sense we will soon describe.

For convenience of notation, we will remove the '*cyc*' notation so that $\mathbf{d}_1$ is a position within the cycle (or equivalently, we could make the assumption that AdaBoost starts on a cycle). Say there exists a cycle such that $d_{1,i} > 0 \; \forall i \in \overline{I}$, where $\mathbf{d}_1$ is a position within the cycle and $\mathbf{M}$ possesses some identically classified examples $\overline{I}$. ($\overline{I}$ is not required to include all examples identically classified with $i \in \overline{I}$.) We know that for each pair of identically classified examples i and i' in $\overline{I}$, we have $M_{ij_t} = M_{i'j_t} \; \forall t = 1, ..., T$. Let perturbation $\mathbf{a} \in \mathbb{R}^m$ obey

$$\sum_{i \in \overline{I}} a_i = 0, \text{ and also } a_i = 0 \text{ for } i \notin \overline{I}.$$

Now, let $\mathbf{d}_1^a := \mathbf{d}_1 + \mathbf{a}$. We accept only perturbations $\mathbf{a}$ so that the perturbation does not affect the value of any j_t in the cycle. That is, we assume each component of $\mathbf{a}$ is sufficiently small; since the dynamical system defined by AdaBoost is piecewise continuous, it is possible to choose $\mathbf{a}$ small enough so the perturbed trajectory is still close to the original trajectory after T iterations. Also, $\mathbf{d}_1^a$

must still be a valid distribution, so it must obey the constraint $\mathbf{d}_1^a \in \Delta_m$, i.e., $\sum_i a_i = 0$ as we have specified. Choose any elements i and $i' \in \overline{I}$. Now,

$$\begin{aligned}
r_1^a &= (\mathbf{d}_1^{a\,T}\mathbf{M})_{j_1} = (\mathbf{d}_1{}^T\mathbf{M})_{j_1} + (\mathbf{a}^T\mathbf{M})_{j_1} = r_1 + \sum_{\bar{i}\in\overline{I}} a_{\bar{i}} M_{\bar{i}j_1} = r_1 + M_{i'j_1}\sum_{\bar{i}\in\overline{I}} a_{\bar{i}} = r_1 \\
d_{2,i}^a &= \frac{d_{1,i}^a}{1+M_{ij_1}r_1^a} = \frac{d_{1,i}^a}{1+M_{ij_1}r_1} = \frac{d_{1,i}}{1+M_{ij_1}r_1} + \frac{1}{1+M_{i'j_1}r_1}a_i = d_{2,i} + \frac{1}{1+M_{i'j_1}r_1}a_i \\
r_2^a &= (\mathbf{d}_2^{a\,T}\mathbf{M})_{j_2} = (\mathbf{d}_2{}^T\mathbf{M})_{j_2} + \frac{1}{1+M_{i'j_1}r_1}(\mathbf{a}^T\mathbf{M})_{j_2} = r_2 + \frac{1}{1+M_{i'j_1}r_1}\sum_{\bar{i}\in\overline{I}} a_{\bar{i}} M_{\bar{i}j_2} \\
&= r_2 + \frac{M_{i'j_2}}{1+M_{i'j_1}r_1}\sum_{\bar{i}\in\overline{I}} a_{\bar{i}} = r_2 \\
d_{2,i}^a &= \frac{d_{2,i}^a}{1+M_{ij_2}r_2^a} = \frac{d_{2,i}^a}{1+M_{ij_2}r_2} = \frac{d_{2,i}}{1+M_{ij_2}r_2} + \frac{1}{(1+M_{i'j_2}r_2)(1+M_{i'j_1}r_1)}a_i \\
&= d_{3,i} + \frac{1}{(1+M_{i'j_2}r_2)(1+M_{i'j_1}r_1)}a_i \\
&\vdots \\
d_{T+1,i}^a &= d_{T+1,i} + \frac{1}{\prod_{t=1}^T(1+M_{i'j_t}r_t)}a_i = d_{1,i} + a_i = d_{1,i}^a.
\end{aligned}$$

The cycle condition was used in the last line. This calculation shows that if we perturb any cycle in the directions defined by $\overline{I}$, we will find another cycle. An entire manifold of cycles then exists, corresponding to the possible nonzero acceptable perturbations $\mathbf{a}$. Effectively, the perturbation shifts the distribution among examples in $\overline{I}$, with the total weight remaining the same. For example, if a cycle exists containing vector $\mathbf{d}_1$ with $d_{1,1} = .20, d_{1,2} = .10$, and $d_{1,3} = .30$, where $\{1,2,3\} \subset \overline{I}$, then a cycle with $d_{1,1} = .22, d_{1,2} = .09$, and $d_{1,3} = .29$ also exists, assuming none of the j_t's change; in this way, groups of identically distributed examples may be treated as one example, because they must share a single total weight (again, only within the region where none of the j_t's change).

We will now consider a simple case where manifolds of cycles exist, and we will show that these manifolds are stable in the proof of Theorem 3.

The form of the matrix $\mathbf{M}$ is

$$\begin{pmatrix}
-1 & 1 & 1 \\
\vdots & \vdots & \vdots \\
-1 & 1 & 1 \\
1 & -1 & 1 \\
\vdots & \vdots & \vdots \\
1 & -1 & 1 \\
1 & 1 & -1 \\
\vdots & \vdots & \vdots \\
1 & 1 & -1 \\
1 & 1 & 1 \\
\vdots & \vdots & \vdots \\
1 & 1 & 1
\end{pmatrix}.$$

To be more specific, the first q_1 training examples are misclassified only by h_1, the next q_2 examples are misclassified only by h_2, the next q_3 examples are misclassified only by h_3, and the last q_4

examples are always correctly classified (their weights converge to zero). Thus we consider the components of $\mathbf{d}$ as belonging to one of four pieces; as long as

$$\left(\sum_{i=1}^{q_1} d_i, \sum_{i=q_1+1}^{q_1+q_2} d_i, \sum_{i=q_1+q_2+1}^{q_1+q_2+q_3} d_i\right)^T = \mathbf{d}_1^{cyc}, \mathbf{d}_2^{cyc}, \mathbf{d}_3^{cyc}, \mathbf{d}_1^{cyc'}, \mathbf{d}_2^{cyc'}, \text{ or } \mathbf{d}_3^{cyc'} \text{ from Section 4,}$$

then $\mathbf{d}$ lies on a 3-cycle as we have just shown.

Theorem 3 *For the matrix $\boldsymbol{M}$ defined above, manifolds of cycles exist (there is a continuum of cycles). These manifolds are stable.*

The proof of Theorem 3 can be found in Appendix B.

7. Cycles and Support Vectors

Our goal is to understand general properties of AdaBoost in cases where cycling occurs, to broaden our understanding of the phenomenon we have observed in Sections 4, 5, and 6. Specifically, we show that if cyclic dynamics occur, the training examples with the smallest margin are the training examples whose $d_{t,i}$ values stay non-zero (the "support vectors"). In the process, we provide a formula that allows us to directly calculate AdaBoost's asymptotic margin from the edges at each iteration of the cycle. Finally, we give sufficient conditions for AdaBoost to produce a maximum margin solution when cycling occurs.

As demonstrated in Figure 6, there are many low-dimensional matrices $\mathbf{M}$ for which AdaBoost empirically produces cyclic behavior. The matrices used to generate the cycle plots in Figure 6 are contained in Figure 7. These matrices were generated randomly and reduced (rows and columns that did not seem to play a role in the asymptotic behavior were eliminated). We observe cyclic behavior in many more cases than are shown in the figure; almost every low-dimensional random matrix that we tried (and even some larger matrices) seems to yield cyclic behavior. Our empirical observations of cyclic behavior in many cases leads us to build an understanding of AdaBoost's general asymptotic behavior in cases where cycles exist, though there is not necessarily a contraction at each iteration so the dynamics may be harder to analyze. (We at least assume the cycles AdaBoost produces are stable, since it is not likely we would observe them otherwise.) These cyclic dynamics may not persist in very large experimental cases, but from our empirical evidence, it seems plausible (even likely) that cyclic behavior might persist in cases in which there are very few support vectors.

When AdaBoost converges to a cycle, it "chooses" a set of rows and a set of columns, that is:

- The j_t's cycle amongst some of the columns of $\mathbf{M}$, but not necessarily all of the columns. In order for AdaBoost to produce a maximum margin solution, it must choose a set of columns such that the maximum margin for $\mathbf{M}$ can be attained using only those columns.

- The values of $d_{t,i}$ (for a fixed value of i) are either always 0 or always strictly positive throughout the cycle. A *support vector* is a training example i such that the $d_{t,i}$'s in the cycle are strictly positive. These support vectors are similar to the support vectors of a support vector machine in that they all attain the minimum margin over training examples (as we will show). These are training examples that AdaBoost concentrates the hardest on. The remaining training examples have zero weight throughout the cycle; these are the examples that are easier

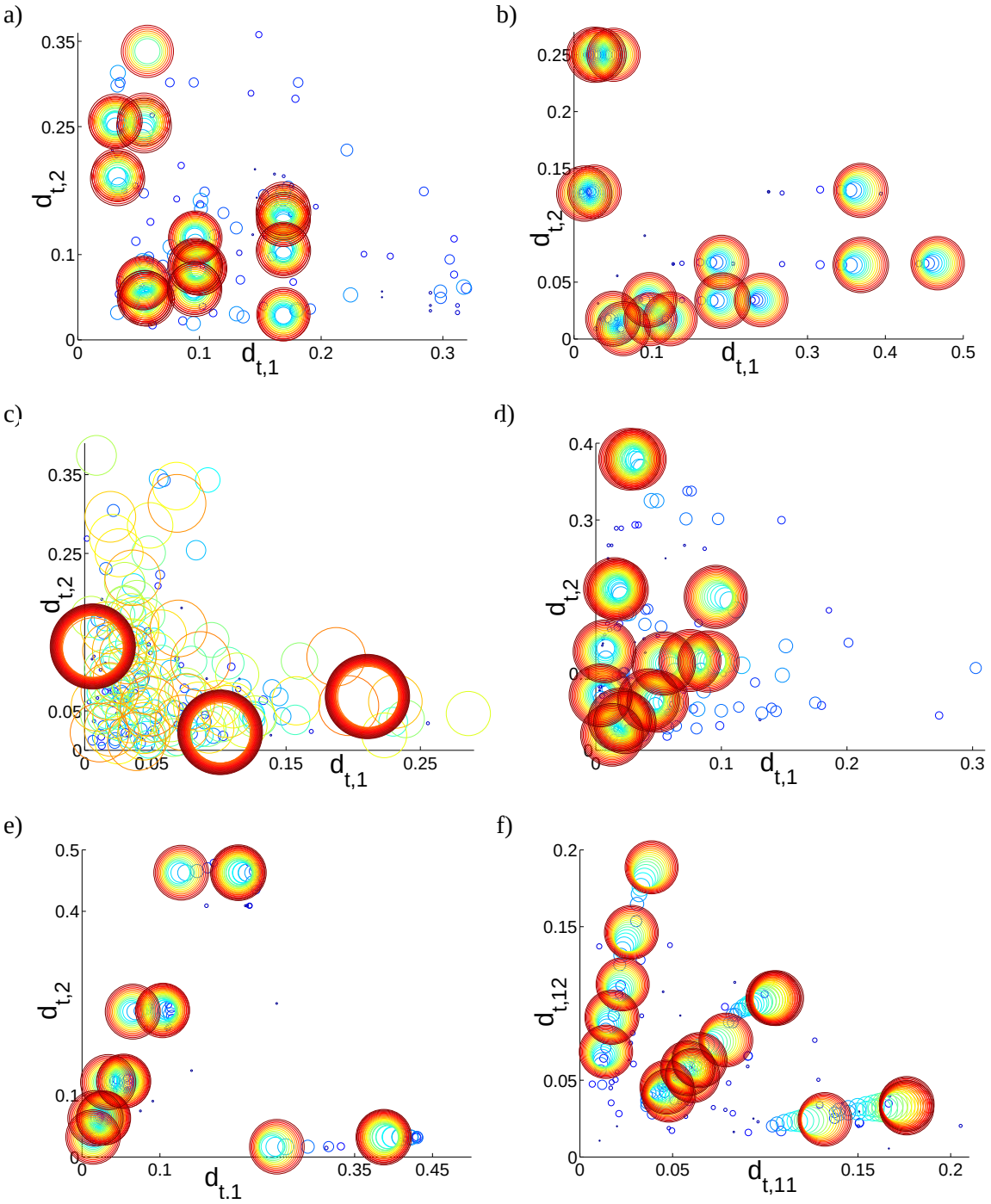

Figure 6: Examples of cycles from randomly generated matrices **M**. An image of **M** for each plot appears in Figure 7. These plots show a projection onto the first two components of AdaBoost's weight vector, e.g., the axes might be $d_{t,1}$ vs. $d_{t,2}$. Smaller circles indicate earlier iterations, and larger circles indicate later iterations. For (a), (d) and (f), 400 iterations were plotted, and for (b) and (e), 300 iterations were plotted. Plot (c) shows 5500 iterations, but only every 20th iteration was plotted. This case took longer to converge, and converged to a simple 3-cycle.

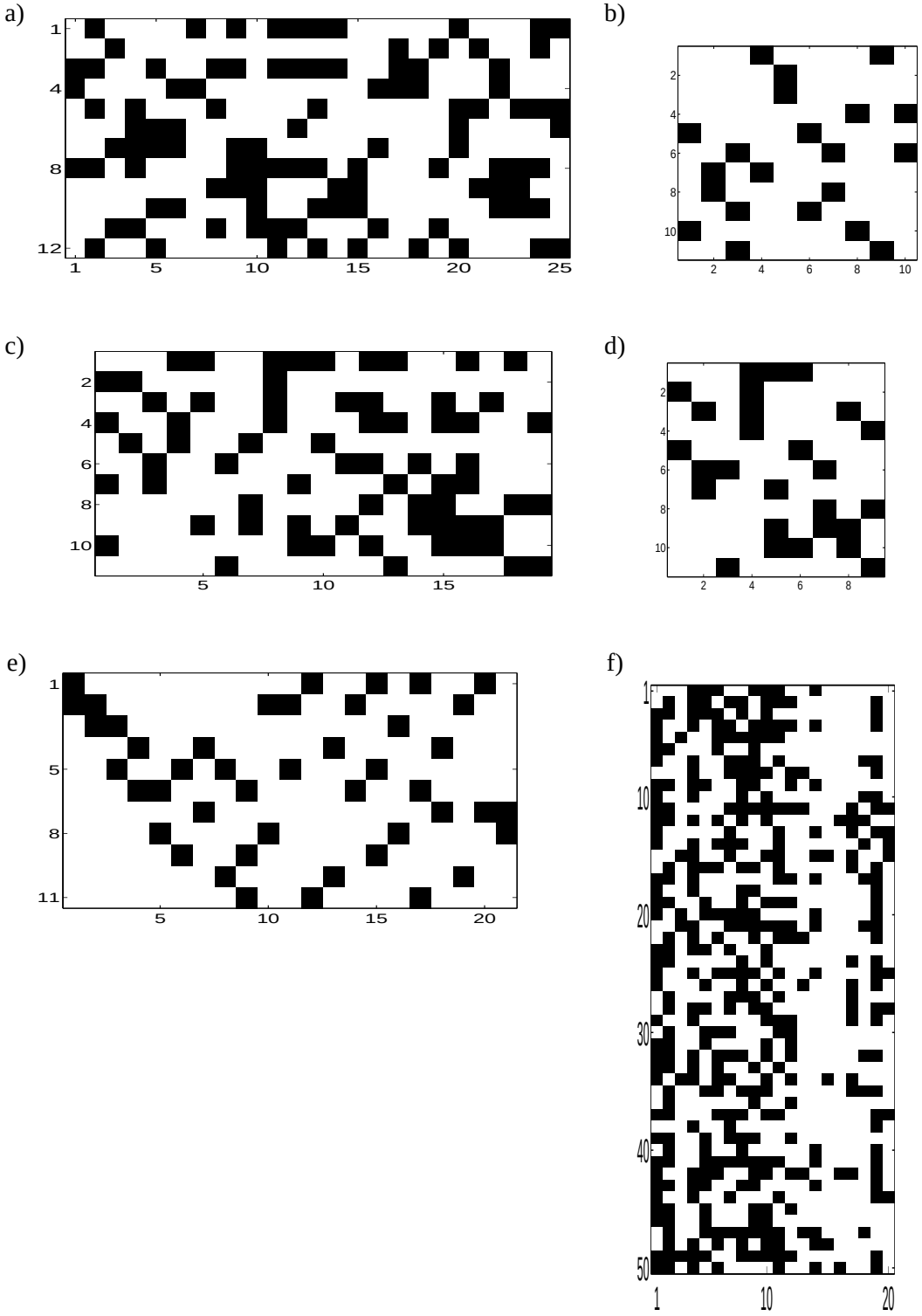

Figure 7: The matrices **M** used to generate the plots in Figure 6. White indicates a value of 1, and black indicates a value of -1. The size of **M** does not seem to have a direct correlation on either the number of iterations per cycle, or the speed of convergence to a cycle.

for the algorithm to classify, since they have margin larger than the support vectors. For support vectors, the cycle condition holds, $\prod_{t=1}^{T}(1+M_{ij_t}r_t^{cyc})=1$. (This holds by Step 3 of the iterated map.) For non-support vectors, $\prod_{t=1}^{T}(1+M_{ij_t}r_t^{cyc})>1$ so the $d_{t,i}$'s converge to 0 (the cycle must be stable).

Theorem 4 *AdaBoost produces the same margin for each support vector and larger margins for other training examples. This margin can be expressed in terms of the cycle parameters* $r_1^{cyc},...,r_T^{cyc}$.

Proof Assume AdaBoost is cycling. Assume $\mathbf{d}_1$ is within the cycle for ease of notation. The cycle produces a normalized output $\boldsymbol{\lambda}^{cyc} := \lim_{t\to\infty}\boldsymbol{\lambda}_t/||\boldsymbol{\lambda}_t||_1$ for AdaBoost. (This limit always converges when AdaBoost converges to a cycle.) Denote

$$z_{cyc} := \sum_{t=1}^{T}\alpha_t = \sum_{t=1}^{T}\frac{1}{2}\ln\left(\frac{1+r_t}{1-r_t}\right).$$

Let i be a support vector. Then,

$$\begin{aligned}
(\mathbf{M}\boldsymbol{\lambda}^{cyc})_i &= \frac{1}{z_{cyc}}\sum_{t=1}^{T}M_{ij_t}\alpha_t = \frac{1}{z_{cyc}}\sum_{t=1}^{T}M_{ij_t}\frac{1}{2}\ln\left(\frac{1+r_t}{1-r_t}\right)\\
&= \frac{1}{2z_{cyc}}\sum_{t=1}^{T}\ln\left(\frac{1+M_{ij_t}r_t}{1-M_{ij_t}r_t}\right) = \frac{1}{2z_{cyc}}\ln\left[\prod_{t=1}^{T}\frac{1+M_{ij_t}r_t}{1-M_{ij_t}r_t}\right]\\
&= \frac{1}{2z_{cyc}}\ln\left[\left(\prod_{t=1}^{T}\frac{1+M_{ij_t}r_t}{1-M_{ij_t}r_t}\right)\left(\prod_{t=1}^{T}\frac{1}{(1+M_{ij_t}r_t)}\right)^2\right]\\
&= \frac{1}{2z_{cyc}}\ln\left[\prod_{t=1}^{T}\frac{1}{(1-M_{ij_t}r_t)(1+M_{ij_t}r_t)}\right]\\
&= \frac{1}{2z_{cyc}}\ln\left[\prod_{t=1}^{T}\frac{1}{1-r_t^2}\right] = -\frac{1}{2}\frac{\ln\prod_{t=1}^{T}(1-r_t^2)}{\sum_{\tilde{t}=1}^{T}\frac{1}{2}\ln\left(\frac{1+r_{\tilde{t}}}{1-r_{\tilde{t}}}\right)}\\
&= -\frac{\ln\prod_{t=1}^{T}(1-r_t^2)}{\ln\prod_{\tilde{t}=1}^{T}\left(\frac{1+r_{\tilde{t}}}{1-r_{\tilde{t}}}\right)}. \qquad (5)
\end{aligned}$$

The first line uses the definition of α_t from the AdaBoost algorithm, the second line uses the fact that $\mathbf{M}$ is binary, the third line uses the fact that i is a support vector, i.e., $\prod_{t=1}^{T}(1+M_{ij_t}r_t)=1$. Since the value in (5) is independent of i, this is the value of the margin that AdaBoost assigns to every support vector i. We denote the value in (5) as μ_{cycle}, which is only a function of the cycle parameters, i.e., the edge values.

Now we show that every non-support vector achieves a larger margin than μ_{cycle}. For a non-support vector i, we have $\prod_{t=1}^{T}(1+M_{ij_t}r_t)>1$, that is, the cycle is stable. Thus, $0>\ln\left[\frac{1}{\prod_{t=1}^{T}(1+M_{ij_t}r_t)}\right]^2$.

Now,

$$\begin{aligned}(\mathbf{M}\boldsymbol{\lambda}^{cyc})_i &= \frac{1}{z_{cyc}}\sum_{t=1}^{T} M_{ij_t}\alpha_t = \frac{1}{2z_{cyc}}\ln\left[\frac{\prod_t(1+M_{ij_t}r_t)}{\prod_{\tilde{t}}(1-M_{ij_{\tilde{t}}}r_{\tilde{t}})}\right] \\ &> \frac{1}{2z_{cyc}}\ln\left[\frac{\prod_t(1+M_{ij_t}r_t)}{\prod_{\tilde{t}}(1-M_{ij_{\tilde{t}}}r_{\tilde{t}})}\right] + \frac{1}{2z_{cyc}}\ln\left[\frac{1}{\prod_{\tilde{t}}(1+M_{ij_{\tilde{t}}}r_{\tilde{t}})}\right]^2 \\ &= \frac{-\ln\prod_{t=1}^{T}(1-r_t^2)}{\ln\prod_{\tilde{t}=1}^{T}\left(\frac{1+r_{\tilde{t}}}{1-r_{\tilde{t}}}\right)} = \mu_{cycle}.\end{aligned}$$

Thus, non-support vectors achieve larger margins than support vectors. ■

The previous theorem shows that the asymptotic margin of the support vectors is the same as the asymptotic margin produced by AdaBoost; this asymptotic margin can be directly computed using (5). AdaBoost may not always produce a maximum margin solution, as we will see in Sections 8 and 9; however, there are sufficient conditions such that AdaBoost will automatically produce a maximum margin solution when cycling occurs. Before we state these conditions, we define the matrix $\mathbf{M}_{cyc} \in \{-1,1\}^{m_{cyc}\times n_{cyc}}$, which contains certain rows and columns of $\mathbf{M}$. To construct $\mathbf{M}_{cyc}$ from $\mathbf{M}$, we choose only the rows of $\mathbf{M}$ that correspond to support vectors (eliminating the others, whose weights vanish anyway), and choose only the columns of $\mathbf{M}$ corresponding to weak classifiers that are chosen in the cycle (eliminating the others, which are never chosen after cycling begins anyway). Here, m_{cyc} is the number of support vectors chosen by AdaBoost, and n_{cyc} is the number of weak classifiers in the cycle.

Theorem 5 *Suppose AdaBoost is cycling, and that the following are true:*

1. $$\max_{\hat{\boldsymbol{\lambda}}\in\Delta_{n_{cyc}}} \min_i (\boldsymbol{M}_{cyc}\hat{\boldsymbol{\lambda}})_i = \max_{\tilde{\boldsymbol{\lambda}}\in\Delta_n} \min_i (\boldsymbol{M}\tilde{\boldsymbol{\lambda}})_i = \rho$$

   (AdaBoost cycles among columns of $\boldsymbol{M}$ that can be used to produce a maximum margin solution.)

2. *There exists $\boldsymbol{\lambda}_\rho \in \Delta_{n_{cyc}}$ such that $(\boldsymbol{M}_{cyc}\boldsymbol{\lambda}_\rho)_i = \rho$ for $i = 1,\ldots,m_{cyc}$. (AdaBoost chooses support vectors corresponding to a maximum margin solution for $\boldsymbol{M}_{cyc}$.)*

3. *The matrix $\boldsymbol{M}_{cyc}$ is invertible.*

Then AdaBoost produces a maximum margin solution.

The first two conditions specify that AdaBoost cycles among columns of $\mathbf{M}$ that can be used to produce a maximum margin solution, and chooses support vectors corresponding to this solution. The first condition specifies that the maximum margin, ρ, (corresponding to the matrix $\mathbf{M}$) must be the same as the maximum margin corresponding to $\mathbf{M}_{cyc}$. Since the cycle is stable, all other training examples achieve larger margins; hence ρ is the best possible margin $\mathbf{M}_{cyc}$ can achieve. The second condition specifies that there is at least one analytical solution $\boldsymbol{\lambda}_\rho$ such that all training examples of $\mathbf{M}_{cyc}$ achieve a margin of exactly ρ.

Proof By Theorem 4, AdaBoost will produce the same margin for all of the rows of $\mathbf{M}_{cyc}$, since they are all support vectors. We denote the value of this margin by μ_{cycle}.

Let $\chi_{m_{cyc}} := (1,1,1,\ldots,1)^T$, with m_{cyc} components. From 2, we are guaranteed the existence of λ_ρ such that

$$\mathbf{M}_{cyc}\lambda_\rho = \rho\chi_{m_{cyc}}.$$

We already know

$$\mathbf{M}_{cyc}\lambda^{cyc} = \mu_{cycle}\chi_{m_{cyc}}$$

since all rows are support vectors for our cycle. Since $\mathbf{M}_{cyc}$ is invertible,

$$\lambda^{cyc} = \mu_{cycle}\mathbf{M}_{cyc}^{-1}\chi_{m_{cyc}} \quad \text{and} \quad \lambda_\rho = \rho\mathbf{M}_{cyc}^{-1}\chi_{m_{cyc}},$$

so we have $\lambda^{cyc} = \text{constant} \cdot \lambda_\rho$. Since λ^{cyc} and λ_ρ must both be normalized, the constant must be 1. Thus $\rho = \mu_{cycle}$. ■

It is possible for the conditions of Theorem 5 not to hold, for example, condition 1 does not hold in the examples of Sections 8 and 9; in these cases, a maximum margin solution is not achieved. It can be shown that the first two conditions are necessary but the third one is not. It is not hard to understand the necessity of the first two conditions; if it is not possible to produce a maximum margin solution using the weak classifiers and support vectors AdaBoost has chosen, then it is not possible for AdaBoost to produce a maximum margin solution. The third condition is thus quite important, since it allows us to uniquely identify λ^{cyc}. Condition 3 does hold for the cases studied in Sections 4 and 5.

8. Non-Optimal AdaBoost Does Not Necessarily Converge to a Maximum Margin Solution, Even if Optimal AdaBoost Does

Based on large scale experiments and a gap in theoretical bounds, Rätsch and Warmuth (2002) conjectured that AdaBoost does not necessarily converge to a maximum margin classifier in the non-optimal case, i.e., that AdaBoost is not robust in this sense. In practice, the weak classifiers are generated by CART or another weak learning algorithm, implying that the choice need not always be optimal.

We will consider a 4×5 matrix $\mathbf{M}$ for which AdaBoost fails to converge to a maximum margin solution if the edge at each iteration is required only to exceed ρ (the non-optimal case). That is, we show that "non-optimal AdaBoost" (AdaBoost in the non-optimal case) may not converge to a maximum margin solution, even in cases where "optimal AdaBoost" does.

Theorem 6 *AdaBoost in the non-optimal case may fail to converge to a maximum margin solution, even if optimal AdaBoost does. An example illustrating this is*

$$\boldsymbol{M} = \begin{pmatrix} -1 & 1 & 1 & 1 & -1 \\ 1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 & 1 \end{pmatrix}.$$

Proof For this matrix, the maximum margin ρ is 1/2. Actually, in the optimal case, AdaBoost will produce this value by cycling among the first four columns of $\mathbf{M}$. Recall that in the non-optimal case:

$$j_t \in \{j : (\mathbf{d}_t^T \mathbf{M})_j \geq \rho\}.$$

Consider the following initial condition for the dynamics:

$$\mathbf{d}_1 = \left(\frac{3-\sqrt{5}}{8}, \frac{3-\sqrt{5}}{8}, \frac{1}{2}, \frac{\sqrt{5}-1}{4} \right)^T.$$

Since $(\mathbf{d}_1^T \mathbf{M})_5 = (\sqrt{5}-1)/2 > 1/2 = \rho$, we are justified in choosing $j_1 = 5$, although here it is not the optimal choice. Another iteration yields

$$\mathbf{d}_2 = \left(\frac{1}{4}, \frac{1}{4}, \frac{\sqrt{5}-1}{4}, \frac{3-\sqrt{5}}{4} \right)^T,$$

satisfying $(\mathbf{d}_1^T \mathbf{M})_4 > \rho$ for which we choose $j_2 = 4$. At the following iteration, we choose $j_3 = 3$, and at the fourth iteration we find $\mathbf{d}_4 = \mathbf{d}_1$. This cycle is the same as one of the cycles considered in Section 4 (although there is one extra dimension). There is actually a whole manifold of 3-cycles, since $\tilde{\mathbf{d}}_1^T := (\varepsilon, \frac{3-\sqrt{5}}{4} - \varepsilon, \frac{1}{2}, \frac{\sqrt{5}-1}{4})$ lies on a (non-optimal) cycle for any ε, $0 \leq \varepsilon \leq \frac{3-\sqrt{5}}{4}$. In any case, the value of the margin produced by this cycle is 1/3, not 1/2. ■

We have thus established that AdaBoost is not robust in the sense we described; if the weak learning algorithm is not required to choose the optimal weak classifier at each iteration, but is required only to choose a sufficiently good weak classifier $j_t \in \{j : (\mathbf{d}_t^T \mathbf{M})_j \geq \rho\}$, a maximum margin solution will not necessarily be attained, even if optimal AdaBoost would have produced a maximum margin solution. We are not saying that the only way for AdaBoost to converge to a non-maximum margin solution is to fall into the wrong cycle; it is conceivable that there may be many other, non-cyclic, ways for the algorithm to fail to converge to a maximum margin solution.

Note that for some matrices $\mathbf{M}$, the maximum value of the margin may still be attained in the non-optimal case; an example is the 3×3 matrix we analyzed in Section 4. If one considers the 3×3 matrix in the non-optimal case, the usual 3-cycle may not persist. Oddly, a 4-cycle may emerge instead. If AdaBoost converges to this 4-cycle, it will still converge to the same (maximum) margin of 1/3. See Appendix C for the coordinates of such a 4-cycle. Thus, there is no guarantee as to whether the non-optimal case will produce the same asymptotic margin as the optimal case.

In Figure 8, we illustrate the evolution of margins in the optimal and non-optimal cases for matrix $\mathbf{M}$ of Theorem 6. Here, optimal AdaBoost converges to a margin of $1/2$ via convergence to a 4-cycle, and non-optimal AdaBoost converges to a margin of 1/3 via convergence to a 3-cycle.

9. Optimal AdaBoost Does Not Necessarily Converge to a Maximum Margin Solution

In this section, we produce a low-dimensional example that answers the question of whether AdaBoost always converges to a maximum margin in the optimal case.

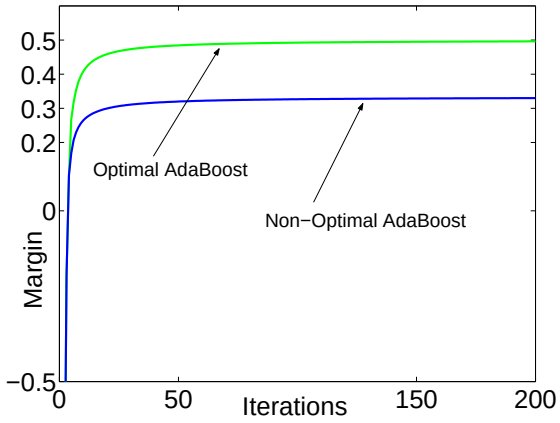

Figure 8: AdaBoost in the optimal case (higher curve) and in the non-optimal case (lower curve). Optimal AdaBoost converges to a margin of $1/2$ via convergence to a 4-cycle, and non-optimal AdaBoost converges to a margin of 1/3 via convergence to a 3-cycle. In both cases we start with $\boldsymbol{\lambda}_1 = \mathbf{0}$.

Theorem 7 *Consider the following matrix whose image appears in Figure 9 (one can see the natural symmetry more easily in the imaged version):*

$$\boldsymbol{M} = \begin{bmatrix} -1 & 1 & 1 & 1 & 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 & -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & 1 & -1 \end{bmatrix}. \tag{6}$$

For this matrix, it is possible for AdaBoost to fail to converge to a maximum margin solution.

Proof The dynamical system corresponding to this matrix contains a manifold of strongly attracting 3-cycles. The cycles we will analyze alternate between weak classifiers 3, 2, and 1. If we consider only weak classifiers 1, 2, and 3, we find that training examples $i = 1$ and 2 are identically classified, i.e., rows 1 and 2 of matrix $\mathbf{M}$ are the same (only considering columns 1, 2, and 3). Similarly, examples 3, 4 and 5 are identically classified, and additionally, examples 6 and 7. Training example 8 is correctly classified by each of these weak classifiers. Because we have constructed $\mathbf{M}$ to have such a strong attraction to a 3-cycle, there are many initial conditions (initial values of $\boldsymbol{\lambda}$) for which AdaBoost will converge to one of these cycles, including the vector $\boldsymbol{\lambda} = \mathbf{0}$. For the first iteration, we chose $j_t = 1$ to achieve the cycle we will analyze below; there are a few different choices for j_t within the first few iterations, since the argmax set sometimes contains more than one element. The dynamics may converge to a different 3-cycle, depending on which values of j_t are chosen within the first few iterations. (Oddly enough, there are initial values of $\boldsymbol{\lambda}$ where AdaBoost converges to

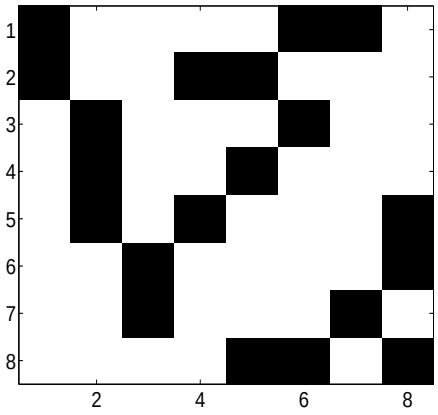

Figure 9: The image of the matrix **M** in (6). White indicates +1, black indicates -1. This matrix has natural symmetry.

a cycle in which a maximum margin solution is produced, although finding such a cycle requires some work.)

To show that a manifold of 3-cycles exists, we present a vector $\mathbf{d}_1$ such that $\mathbf{d}_4 = \mathbf{d}_1$, namely:

$$\mathbf{d}_1 = \left(\frac{3-\sqrt{5}}{8}, \frac{3-\sqrt{5}}{8}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{\sqrt{5}-1}{8}, \frac{\sqrt{5}-1}{8}, 0 \right)^T. \tag{7}$$

To see this, we iterate the iterated map 4 times.

$$\mathbf{d}_1^T \mathbf{M} = \left(\frac{\sqrt{5}-1}{2}, 0, \frac{3-\sqrt{5}}{2}, \frac{3\sqrt{5}-1}{12}, \frac{3\sqrt{5}-1}{12}, \frac{3\sqrt{5}-1}{12}, \frac{1}{2}, \frac{11-3\sqrt{5}}{12} \right),$$

and here $j_1 = 1$,

$$\begin{aligned} \mathbf{d}_2 &= \left(\frac{1}{4}, \frac{1}{4}, \frac{\sqrt{5}-1}{12}, \frac{\sqrt{5}-1}{12}, \frac{\sqrt{5}-1}{12}, \frac{3-\sqrt{5}}{8}, \frac{3-\sqrt{5}}{8}, 0 \right)^T \\ \mathbf{d}_2^T \mathbf{M} &= \left(0, \frac{3-\sqrt{5}}{2}, \frac{\sqrt{5}-1}{2}, \frac{4-\sqrt{5}}{6}, \frac{4-\sqrt{5}}{6}, \frac{4-\sqrt{5}}{6}, \frac{\sqrt{5}-1}{4}, \frac{5+\sqrt{5}}{12} \right), \end{aligned}$$

and here $j_2 = 3$,

$$\begin{aligned} \mathbf{d}_3 &= \left(\frac{\sqrt{5}-1}{8}, \frac{\sqrt{5}-1}{8}, \frac{3-\sqrt{5}}{12}, \frac{3-\sqrt{5}}{12}, \frac{3-\sqrt{5}}{12}, \frac{1}{4}, \frac{1}{4}, 0 \right)^T \\ \mathbf{d}_3^T \mathbf{M} &= \left(\frac{3-\sqrt{5}}{2}, \frac{\sqrt{5}-1}{2}, 0, \frac{3}{4} - \frac{\sqrt{5}}{12}, \frac{3}{4} - \frac{\sqrt{5}}{12}, \frac{3}{4} - \frac{\sqrt{5}}{12}, \frac{3-\sqrt{5}}{4}, \frac{\sqrt{5}}{6} \right), \end{aligned}$$

and here, $j_3 = 2$, and then $\mathbf{d}_4 = \mathbf{d}_1$.

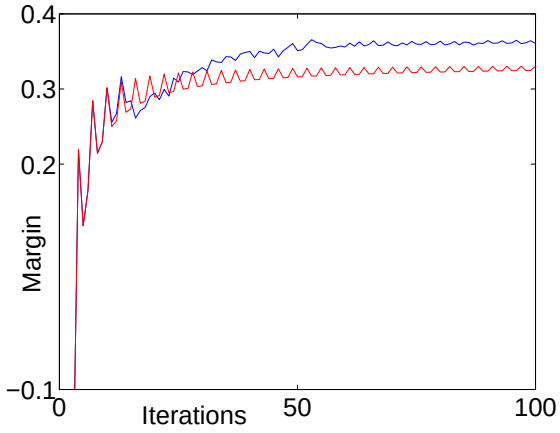

Figure 10: AdaBoost (lower curve) and Approximate Coordinate Ascent Boosting (higher curve), using the 8×8 matrix $\mathbf{M}$ given in Section 9 and initial condition $\boldsymbol{\lambda} = \mathbf{0}$. AdaBoost converges to a margin of 1/3, yet the value of ρ is 3/8. Thus, AdaBoost does not converge to a maximum margin solution for this matrix $\mathbf{M}$.

Hence the 3-cycle exists, and since there are identically classified examples, a manifold of cycles exists; it is automatically stable due to the calculation in the proof of Theorem 3. The margin produced by one of these 3-cycles is always 1/3, yet the maximum margin for this matrix is 3/8. To see that a margin of 3/8 can be obtained, note that $\mathbf{M} \times [2,3,4,1,2,2,1,1]^T \times 1/16 = 3/8$ for all i.■

In Figure 10, we have plotted the evolution of the margin over time for $\mathbf{M}$, for both AdaBoost and Approximate Coordinate Ascent Boosting. Approximate Coordinate Ascent Boosting (Rudin et al., 2004c,b,a; Rudin, 2004) is an algorithm similar to AdaBoost that converges to a maximum margin solution, and runs in polynomial time. AdaBoost rapidly converges to the cycle analyzed above and does not produce a maximum margin solution.

Again, we are not saying that the only way for AdaBoost to converge to a non-maximum margin solution is to fall into the wrong cycle since there may be many other non-cyclic ways for the algorithm to fail to converge to a maximum margin solution. However, many high dimensional cases can be reduced to low dimensional cases simply by restricting our attention to support vectors and weak classifiers that are actually chosen by the algorithm. Thus, a "bad" cycle may not be as uncommon as one would expect, even in a realistic setting.

10. Indications of Chaos

Although we do observe cyclic behavior for many random low-dimensional matrices, we have found an example for which AdaBoost exhibits behavior resembling that of a chaotic dynamical system. In particular, this case exhibits:

- Sensitivity to initial conditions.

- Movement into and out of cyclic behavior.

The matrix **M** we consider for this section is given in Figure 7(a).

Figure 11 shows AdaBoost's edge r_t for t ranging from 0 to 10,000; a number of different initial conditions were considered, which are of the form $\lambda_{1,i} = a$ for all i, for $a = 0, 0.01, 0.02, 0.03, 0.05, 0.06, 0.07, 0.08, 0.09$ and 0.1, (a-j) respectively. In many of these cases, cyclic behavior occurs after some time. In fact, for $a = 0.08$, AdaBoost converges to a 3-cycle. Sometimes, AdaBoost seems to migrate in and out of cyclic behavior, and its behavior is not at all clear. Thus, this example suggests sensitivity to initial conditions. This sensitivity makes sense, since the iterated map is not continuous, it is only *piecewise* continuous. That is, if the argmax set contains two different elements, say j_1 and j_2, the arbitrary choice between them may cause the dynamics to change spectacularly. If we are near such a boundary of a j_t region, a small perturbation may change the choice of j_t chosen and the trajectory may change dramatically. (Note that the Li and Yorke "Period-3-Implies-Chaos" result does not apply to the dynamical system defined by AdaBoost since the iterated map is not continuous, as illustrated in Figure 5 for the 3×3 case.)

Within Figure 11, we can see AdaBoost moving into and out of cyclic behavior, for example, in Figure 11(j). In order to closely examine the switch between the large region of cycling within approximately iterations 8500-9381 and the following chaotic region, we focus our attention on the iterations just before the switch into chaotic behavior. This switch does seem to occur due to a change in region. In other words, as AdaBoost cycles for many iterations (in a cycle of length 14), the weight vectors (viewed every 14th iteration, as in Figure 12) migrate towards the edge of a region and eventually cross over this edge. Where previously, at every 14^{th} iteration AdaBoost would choose $j_t = 19$, it instead chooses $j_t = 3$. Figure 12 shows the values of $(\mathbf{d}_t^T\mathbf{M})_3$ and $(\mathbf{d}_t^T\mathbf{M})_{19}$ at every 14^{th} iterate preceding the switch into chaotic behavior at iteration 9381. Figure 13, which shows the evolution of two components of the weight vector, also illustrates the switches into and out of chaotic behavior.

Eventually, the dynamics drift back towards the same cycle and actually seem to converge to it, as shown in Figure 14(a). Here, the weight vectors do not cross regions, since the values of the largest two components of $(\mathbf{d}^T\mathbf{M})$ do not cross, as shown in Figure 14(b).

Thus, there are many open questions regarding AdaBoost's dynamics that could help us understand its asymptotic behavior; for example, is AdaBoost chaotic in some cases or does it perhaps always produce cyclic behavior asymptotically?

11. Conclusions

We have used the nonlinear iterated map defined by AdaBoost to understand its update rule in low-dimensional cases and uncover remarkable cyclic dynamics. We describe many aspects of AdaBoost's dynamical traits including the fact that AdaBoost does not necessarily converge to a maximum margin solution. We have also proved the conjecture that AdaBoost is not robust to the choice of weak classifier. The key to answering these questions was our analysis of cases in which AdaBoost's asymptotic behavior could be completely determined. Thus an understanding of simple cases has yielded answers to important open questions concerning AdaBoost's large-scale asymptotic behavior.

We leave open many interesting questions. To what extent is AdaBoost chaotic? For what cases does AdaBoost produce maximum margin solutions? Does AdaBoost always exhibit cyclic behavior in the limit? If AdaBoost can behave chaotically without converging to a cycle, how large does the

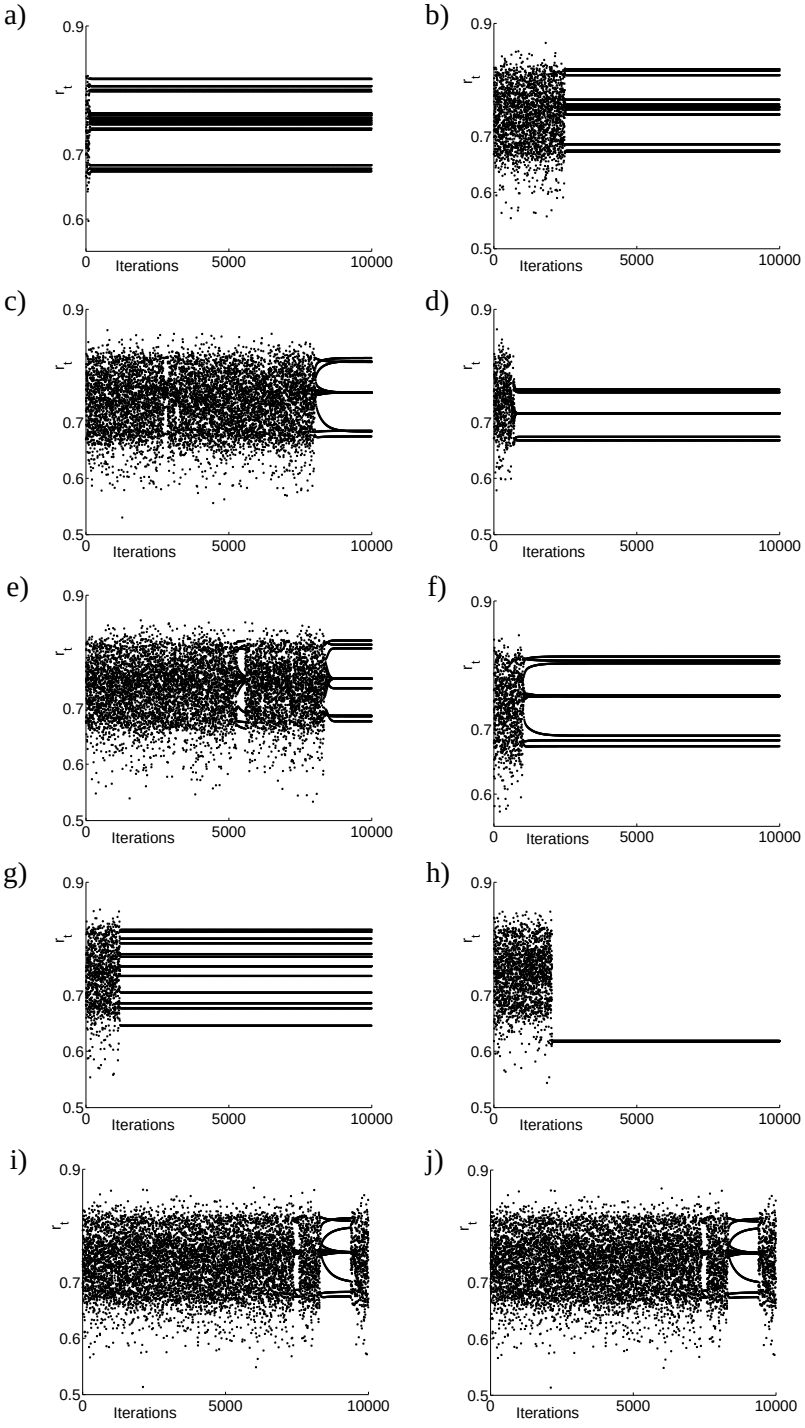

Figure 11: AdaBoost is sensitive to initial conditions. Value of the edge r_t at each iteration t, for many different runs of AdaBoost. For all plots we used the matrix $\mathbf{M}$ shown in Figure 7(a), but with slightly different initial conditions. Some of these plots look somewhat chaotic except for a few regions where AdaBoost seems to be converging to a cycle before becoming chaotic again. In (h), AdaBoost converges to a simple 3-cycle after a significant number of iterations.

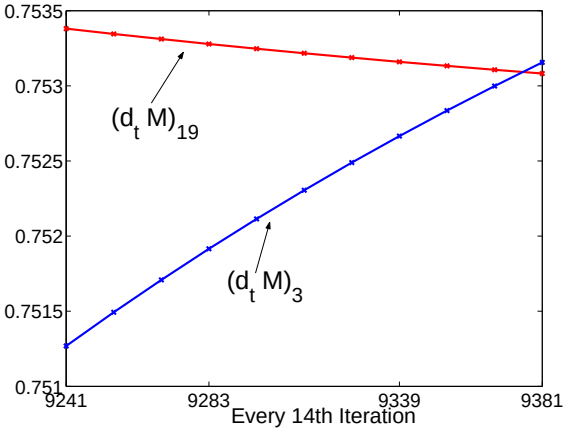

Figure 12: The values of $(\mathbf{d}_t^T\mathbf{M})_3$ and $(\mathbf{d}_t^T\mathbf{M})_{19}$ at every 14^{th} iterate preceding the switch into chaotic behavior of Figure 11(j) where $a = 0.1$. AdaBoost switches from a 14-cycle into chaotic behavior after iteration 9381 when it switches regions, from the region where $j_t = 19$ into the region where $j_t = 3$.

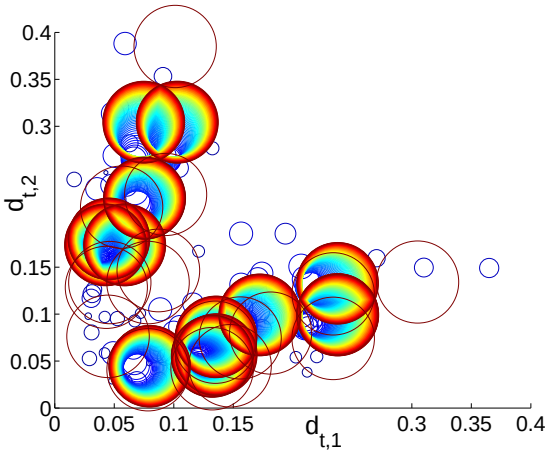

Figure 13: Scatter plot of $d_{t,1}$ vs. $d_{t,2}$ for the iterations surrounding the slow convergence to the cycle in Figure 11(j), where the initial condition is $\lambda_{1,j} = 0.1$ for all j. By examining the circles (especially the smallest and largest ones) one can see the switch into the cyclic behavior, the slow migration towards a cycle, and the fast switch back into chaotic behavior. Again, smaller circles indicate earlier iterations and larger circles indicate later iterations.

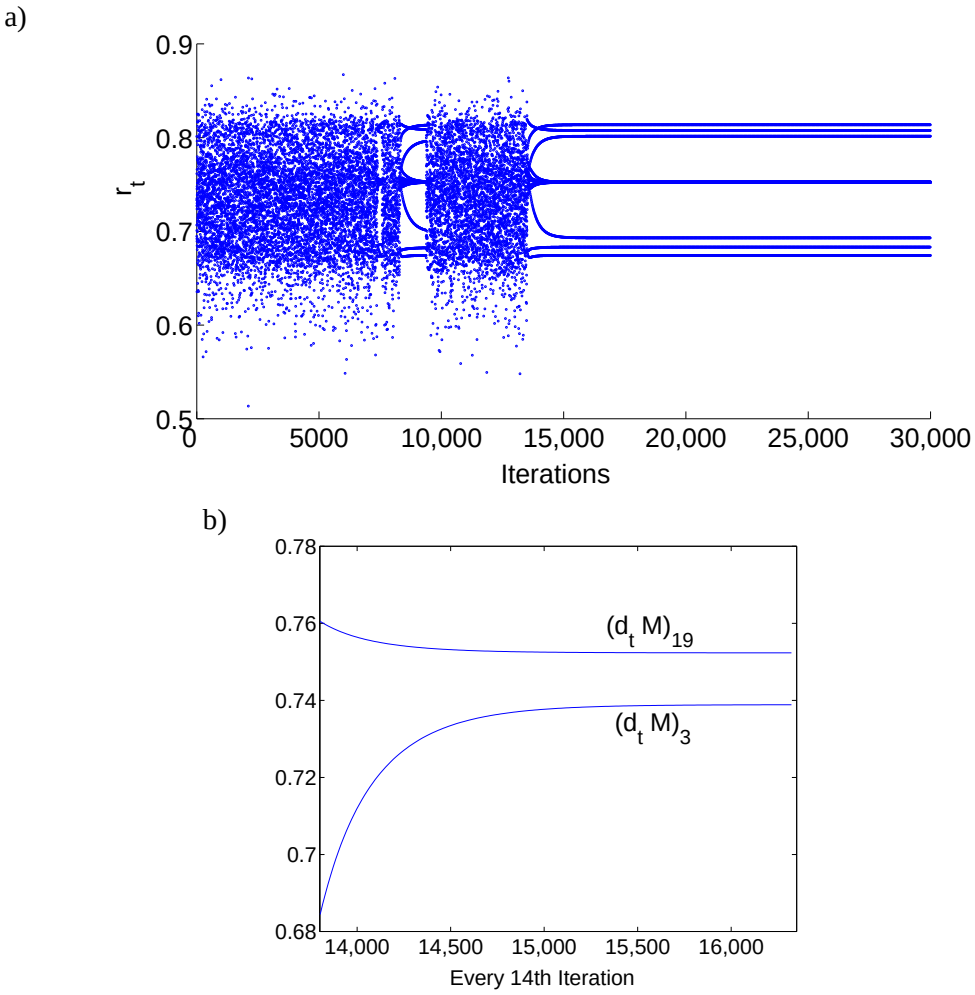

Figure 14: (a) This is the same plot as in Figure 11(j), extended to 30,000 iterations. After more than 13,000 iterations, AdaBoost finally seems to settle on the 14-cycle. (b) The same plot as in Figure 12, but for a different set of iterations. In Figure 12 the edges corresponding to $j = 19$ and $j = 3$ cross, whereas here, the edges are well separated. Thus, AdaBoost is able to maintain the cycle.

matrix **M** need to be in order to produce such behavior? And finally, how does AdaBoost achieve its strong generalization performance if it does not simply maximize the margin? We hope the analytical tools provided in this work, namely the reduction of AdaBoost to a dynamical system and the analysis of its asymptotic behavior in special cases, will help yield answers to these interesting questions.

Acknowledgments

This material is based upon work partially supported by the National Science Foundation under grant numbers CCR-0325463, IIS-0325500 and DMS-9810783. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. This work is also partially supported by AFOSR award F49620-01-1-0099.

Appendix A. Proof of Theorem 2

Let us first assume a cycle with rotating coordinates exists for this case, and then we will prove its existence and stability. Denote the first position in our cycle as follows (we drop the (cyc) notation here):

$$\mathbf{d}_1 = (\beta_1, \cdots, \beta_i, \cdots, \beta_{m-1}, \beta_m)^T,$$

where $0 < \beta_1 < \beta_2 < \cdots < \beta_m$. Note that $\beta_m = \frac{1}{2}$, again because $(\mathbf{d}_2^T \mathbf{M})_1 = 0$ and $\sum_i d_{2,i} = 1$. Now,

$$\mathbf{d}_1^T \mathbf{M} = ((1-2\beta_1), \cdots, (1-2\beta_{m-1}), 0)^T,$$

so $j_1 = 1$ and $r_1 = 1 - 2\beta_1$. Then, using the iterated map,

$$\mathbf{d}_2 = \left(\frac{1}{2}, \frac{\beta_2}{2(1-\beta_1)}, \cdots, \frac{\beta_i}{2(1-\beta_1)}, \cdots, \frac{\beta_{m-1}}{2(1-\beta_1)}, \frac{\beta_m}{2(1-\beta_1)} \right)^T.$$

Assuming that the coordinates cycle,

$$\beta_{i-1} = \frac{\beta_i}{2(1-\beta_1)} \quad \text{for} \quad i = 2, ..., m, \tag{8}$$

in order for the current iterate to agree with the next iterate. This recursive relation gives

$$\beta_{i-(i-1)} = \frac{\beta_i}{[2(1-\beta_1)]^{i-1}},$$

so that

$$\beta_1 = \frac{\beta_m}{[2(1-\beta_1)]^{m-1}} \tag{9}$$

and

$$\beta_i = \beta_1 [2(1-\beta_1)]^{i-1}, \quad \text{for } i = 1, ..., m. \tag{10}$$

Thus, substituting (9) into (10), recalling that $\beta_m = 1/2$,

$$\beta_i = \frac{\beta_m}{[2(1-\beta_1)]^{m-1}} [2(1-\beta_1)]^{i-1} = \frac{1}{2} [2(1-\beta_1)]^{i-m}. \tag{11}$$

It remains to show that there is a viable solution for β_1 to prove the existence of a cycle. (We require a solution to obey $\beta_1 \leq 1/m$ so that it is possible for $\sum_i d_{1,i} = 1$.) The condition $\sum_i d_{1,i} = 1$ can be rewritten as

$$1 = \frac{1}{2}\sum_{i=1}^{m}[2(1-\beta_1)]^{i-m}.$$

Substituting $\varsigma = 2(1-\beta_1)$ and multiplying both sides by $2\varsigma^{m-1}$, we have a geometric series:

$$2\varsigma^{m-1} = \sum_{i=1}^{m}\varsigma^{i-m+m-1} = \sum_{i=1}^{m}\varsigma^{i-1} = \frac{1-\varsigma^m}{1-\varsigma},$$

that is,

$$\varsigma^m - 2\varsigma^{m-1} + 1 = 0. \tag{12}$$

Substituting back for ς,

$$2^m(1-\beta_1)^{m-1}[(1-\beta_1)-1]+1 = 0,$$

or more simply,

$$1 - \beta_1 2^m (1-\beta_1)^{m-1} = 0.$$

To show a solution exists for $m \geq 4$ (we have already handled the $m = 3$ case), we will apply the Intermediate Value Theorem to the function

$$\varphi(\bar{\beta_1}, m) := 1 - \bar{\beta_1} 2^m (1-\bar{\beta_1})^{m-1}.$$

We know $\varphi(0,m) = 1 > 0$. Consider

$$\varphi\left(\frac{1}{10}, m\right) = 1 - 2^m \frac{1}{10}\left(\frac{9}{10}\right)^{m-1} = 1 - \left(\frac{9}{5}\right)^m \frac{1}{9}.$$

Plugging in $m = 4$, we find that $\varphi(1/10, 4) = -104/625 < 0$. On the other hand, extending the definition of φ to non-integer values of m, we have

$$\frac{\partial\varphi(1/10,m)}{\partial m} = -\frac{1}{9}\left[\ln\left(\frac{9}{5}\right)\right]\left(\frac{9}{5}\right)^m < 0 \quad \text{for all } m \geq 4.$$

Hence, $\varphi(1/10, m) \leq -104/625 < 0$ for all $m \geq 4$. By the Intermediate Value Theorem, there is a root β_1 of $\varphi(\cdot, m)$ for any $m \geq 4$ with $0 \leq \beta_1 \leq 1/10$. Since

$$\frac{\partial\varphi(\bar{\beta_1}, m)}{\partial\bar{\beta_1}} = 2^m(1-\bar{\beta_1})^{m-2}(m\bar{\beta_1} - 1),$$

we have $\frac{\partial\varphi}{\partial\beta_1}(\bar{\beta_1}, m) = 0$ only when $\bar{\beta_1} = 1/m$, and that $\varphi(\cdot, m)$ decreases for $0 \leq \bar{\beta_1} < 1/m$ and increases for $\bar{\beta_1} > 1/m$ (where $\bar{\beta_1} < 1$). If $m \leq 10$, since $\varphi(\cdot, m)$ decreases for $0 \leq \bar{\beta_1} < 1/m$, the Intermediate Value Theorem provides the unique root $0 \leq \beta_1 \leq 1/10 \leq 1/m$. If $m > 10$, $\varphi(\cdot, m)$ decreases for $0 \leq \beta_1 < 1/m$ and increases to the value $\varphi(1/10, m)$, which is negative. Thus, there is a unique root $0 \leq \beta_1 \leq 1/m$. Hence, the root exists and is unique for $m \geq 4$. Now we have shown the existence and uniqueness of our cycle, namely the cycle starting from

$$\mathbf{d}_1 = \left(\beta_1, 2\beta_1(1-\beta_1), 4\beta_1^2(1-\beta_1)^2, \ldots, 1/2\right)^T.$$

Of course, this is not the only periodic orbit. Any permutation of the components in $\mathbf{d}_1$ will lie on a periodic cycle. If (without loss of generality) we fix the first iteration of each cycle to start with the same first component $d_{1,i} = \beta_1$, then the number of permutations of the other components (and thus the number of periodic cycles we have defined by relabelling the coordinates) is $(m-1)!$.

We now show that these $(m-1)!$ cycles are stable. It is sufficient to show that just one cycle is stable, since the others are obtained by simply relabelling the order of the coordinates (without loss of generality say $j_t = t$ for $t = 1, ..., m$). We add a perturbation $\varepsilon\mathbf{a}$ to $\mathbf{d}_1$, small enough so that none of the j_t's chosen within the cycle are affected. (Note that choosing such a perturbation is possible, since the map is piecewise continuous, and $\beta_1 < ... < \beta_m$ without equality between the β_i's. This will ensure that no $\mathbf{d}_t$ lies on the boundary of a region, so that the $\text{argmax}_j(\mathbf{d}_t^T\mathbf{M})_j$ set contains exactly one element.) Also, we require $\sum_{i=1}^m a_i = 0$ so the perturbed starting point still lies on the simplex Δ_m. Assume $\|\mathbf{a}\|_1$ is $O(1)$, and that ε is small. Our perturbed starting point is

$$\mathbf{d}_1^a := \mathbf{d}_1 + \varepsilon\mathbf{a}.$$

Now, since $r_1 = 1 - 2\beta_1$,

$$r_1^a \quad = \quad (\mathbf{d}_1^{aT}\mathbf{M})_1 = (\mathbf{d}_1{}^T\mathbf{M})_1 + \varepsilon(\mathbf{a}^T\mathbf{M})_1 = 1 - 2\beta_1 + \varepsilon(\mathbf{a}^T\mathbf{M})_1.$$

Again we use the iterated map. Recall that $\beta_m = \frac{1}{2}$, and $d_{2,1}^a = \frac{1}{2}$. For all other i,

$$d_{2,i}^a = \frac{\beta_i + \varepsilon a_i}{1 + r_1^a} = \frac{\beta_i + \varepsilon a_i}{2 - 2\beta_1 + \varepsilon(\mathbf{a}^T\mathbf{M})_1}.$$

To see whether the perturbation has shrunk due to the dynamics, we compute $\varepsilon\tilde{\mathbf{a}} := \mathbf{d}_2^a - \mathbf{d}_2$. If $\|\tilde{\mathbf{a}}\|_1 \leq C\|\mathbf{a}\|_1$ where C is a constant less than 1, the map is a contraction. Recall that

$$\mathbf{d}_2 = \left(\frac{1}{2}, \beta_1, \cdots, \beta_{m-1}\right)^T.$$

Thus,

$$\varepsilon\tilde{a}_1 = d_{2,1}^a - d_{2,1} = 0,$$

and for other i,

$$\varepsilon\tilde{a}_i = d_{2,i}^a - d_{2,i} = \frac{\beta_i + \varepsilon a_i}{2 - 2\beta_1 + \varepsilon(\mathbf{a}^T\mathbf{M})_1} - \beta_{i-1}.$$

We are done with the $i = 1$ term. For all other terms, we will do an approximation to first order in ε. Using a first order Taylor expansion $\frac{1}{1+x} \approx 1 - x$, we obtain

$$\frac{1}{1 + r_1^a} = \frac{1}{2 - 2\beta_1 + \varepsilon(\mathbf{a}^T\mathbf{M})_1} = \frac{1}{2(1-\beta_1)\left(1 + \frac{\varepsilon(\mathbf{a}^T\mathbf{M})_1}{2(1-\beta_1)}\right)} \approx \frac{1 - \frac{\varepsilon(\mathbf{a}^T\mathbf{M})_1}{2(1-\beta_1)}}{2(1-\beta_1)}.$$

Our expansion yields:

$$\varepsilon\tilde{a}_i = d_{2,i}^a - d_{2,i} \quad \approx \quad \frac{(\beta_i + \varepsilon a_i)\left(1 - \frac{\varepsilon(\mathbf{a}^T\mathbf{M})_1}{2(1-\beta_1)}\right)}{2(1-\beta_1)} - \beta_{i-1}.$$

Grouping terms in orders of ε, we can use (8) to show the first term vanishes, and we find:

$$\begin{aligned}\varepsilon\tilde{a}_i &\approx 0+\varepsilon\left(\frac{a_i}{2(1-\beta_1)}-\frac{\beta_i(\mathbf{a}^T\mathbf{M})_1}{4(1-\beta_1)^2}\right)+O(\varepsilon^2), \quad \text{so that}\\ \tilde{a}_i &\approx \frac{a_i}{2(1-\beta_1)}-\frac{\beta_i(\mathbf{a}^T\mathbf{M})_1}{4(1-\beta_1)^2}+O(\varepsilon).\end{aligned}$$

We will show that the perturbation shrinks at every iteration. Since ε is small, we do not care about the $O(\varepsilon)$ contribution to $\tilde{a}_i$. Recall that $\sum_i \beta_i = 1$, so that:

$$\begin{aligned}\|\tilde{\mathbf{a}}\|_1 &\leq \frac{1}{2(1-\beta_1)}\sum_{i=1}^m |a_i| + \frac{\left|(\mathbf{a}^T\mathbf{M})_1\right|}{4(1-\beta_1)^2}\sum_{i=1}^m \beta_i + O(\varepsilon)\\ &= \frac{1}{2(1-\beta_1)}\|\mathbf{a}\|_1 + \frac{\left|(\mathbf{a}^T\mathbf{M})_1\right|}{4(1-\beta_1)^2} + O(\varepsilon)\\ &\leq \frac{1}{2(1-\beta_1)}\|\mathbf{a}\|_1 + \frac{1}{4(1-\beta_1)^2}\|\mathbf{a}\|_1 + O(\varepsilon)\\ &= \frac{3-2\beta_1}{4(1-\beta_1)^2}\|\mathbf{a}\|_1 + O(\varepsilon).\end{aligned}$$

For the third line, we used the fact that the entries of $\mathbf{M}$ are always within $\{-1,+1\}$. In order to have

$$\frac{3-2\beta_1}{4(1-\beta_1)^2} < 1,$$

we would need

$$3-2\beta_1 < 4(1-\beta_1)^2 = 4-8\beta_1+4\beta_1^2,$$

i.e.,

$$0 < 1-6\beta_1+4\beta_1^2,$$

or more simply,

$$\beta_1 < (3-\sqrt{5})/4.$$

This condition does hold, since

$$0 \leq \beta_1 \leq 1/10 < (3-\sqrt{5})/4.$$

Thus we have shown a contraction of the perturbation at the first iteration. An identical calculation (one must simply reorder the components in the vectors) will yield a contraction at every iteration, so our cycle is stable. We have thus proven the existence and stability of $(m-1)!$ cycles for the case with m weak classifiers, each with 1 misclassified example.

Appendix B. Proof of Theorem 3

The existence of manifolds of cycles follows from the fact that $\mathbf{M}$ has the same 3-cycles as the 3×3 case, except that the weight is distributed among identically classified examples. (Recall that the weights of the last q_4 examples vanish, since these examples are always correctly classified.) In order to move along the manifold, just shift the weights among identically classified examples; this new weight vector will lie directly on another 3-cycle.

In order to show the manifold is stable, we will show that any vector $\mathbf{d}^a$ that lies sufficiently near the manifold will be attracted towards it. More specifically, we will:

- choose an arbitrary vector $\mathbf{d}_1^a$ sufficiently close to the manifold of stable cycles.
- carefully choose a corresponding vector $\mathbf{d}_1$ on the manifold.
- show there is a contraction, sending the successive $\mathbf{d}_t^a$ vectors closer to the $\mathbf{d}_t$ vectors at every iteration. In this way, the path of $\mathbf{d}_t^a$'s will converge to the cycle obeyed by the $\mathbf{d}_t$'s.

Consider an arbitrary vector $\mathbf{d}_1^a$, which we assume to be close enough to the manifold so that the distance between vector $\mathbf{d}_1^a$ and vector $\mathbf{d}_1$ (defined below) is $O(\varepsilon)$. Define

$$k_1^a := \sum_{i=1}^{q_1} d_{1,i}^a, \quad k_2^a := \sum_{i=q_1+1}^{q_1+q_2} d_{1,i}^a, \quad k_3^a := \sum_{i=q_1+q_2+1}^{q_1+q_2+q_3} d_{1,i}^a, \quad \text{and} \quad k_4^a := \sum_{i=q_1+q_2+q_3+1}^{m} d_{1,i}^a.$$

Assume that $(k_1^a, k_2^a, k_3^a)^T$ is $O(\varepsilon)$ close to either $\mathbf{d}_1^{cyc}, \mathbf{d}_2^{cyc}, \mathbf{d}_3^{cyc}, \mathbf{d}_1^{cyc'}, \mathbf{d}_2^{cyc'}$, or $\mathbf{d}_3^{cyc'}$ from Section 4. Since we are permitted to freely shuffle the rows and columns of $\mathbf{M}$ without loss of generality, we assume that $(k_1^a, k_2^a, k_3^a)^T$ is $O(\varepsilon)$ close to $\mathbf{d}_1^{cyc}$, i.e., that k_1^a is $O(\varepsilon)$ close to $k_1 := (3-\sqrt{5})/4$, k_2^a is close to $k_2 := (\sqrt{5}-1)/4$, k_3^a is close to $k_3 := 1/2$, and k_4^a is $O(\varepsilon)$. Now we will carefully choose a corresponding vector $\mathbf{d}_1$ on the manifold, namely

$$d_{1,i} := \left\{ \begin{array}{ccc} \frac{k_1}{k_1^a} d_{1,i}^a & \text{if} & i \le q_1 \\ \frac{k_2}{k_2^a} d_{1,i}^a & \text{if} & q_1 < i \le q_1+q_2 \\ \frac{k_3}{k_3^a} d_{1,i}^a & \text{if} & q_1+q_2 < i \le q_1+q_2+q_3 \\ 0 & \text{otherwise} & \end{array} \right\}. \tag{13}$$

Define $\mathbf{a}_1$ as follows:

$$\varepsilon a_{1,i} := d_{1,i}^a - d_{1,i} = \left\{ \begin{array}{ccc} \left(\frac{k_1^a}{k_1} - 1\right) d_{1,i} & \text{if} & i \le q_1 \\ \left(\frac{k_2^a}{k_2} - 1\right) d_{1,i} & \text{if} & q_1 < i \le q_1+q_2 \\ \left(\frac{k_3^a}{k_3} - 1\right) d_{1,i} & \text{if} & q_1+q_2 < i \le q_1+q_2+q_3 \\ d_{1,i}^a & \text{otherwise} & \end{array} \right\}.$$

The assumption that $\mathbf{d}_1^a$ is sufficiently close to the manifold amounts to the assumption that $\|\mathbf{a}_1\|_1$ is $O(1)$. It is important to note that each of the four pieces of $\mathbf{a}_1$ is proportional to a piece of $\mathbf{d}_1$, and thus proportional to a piece of $\mathbf{d}_1^a$. Recalling that $r_1 = r_2 = r_3 = r = (\sqrt{5}-1)/2$, we have:

$$\begin{aligned} r_1^a &= (\mathbf{d}_1^a \mathbf{M})_1 = (\mathbf{d}_1 \mathbf{M})_1 + \varepsilon(\mathbf{a}_1^T \mathbf{M})_1 = r + \varepsilon(\mathbf{a}_1^T \mathbf{M})_1, \quad \text{so} \\ \frac{1}{1-r_1^a} &= \frac{1}{1-r-\varepsilon(\mathbf{a}_1^T \mathbf{M})_1} = \frac{1}{(1-r)\left(1-\frac{\varepsilon(\mathbf{a}_1^T \mathbf{M})_1}{1-r}\right)} \\ &= \frac{1}{1-r}\left(1+\frac{\varepsilon(\mathbf{a}_1^T \mathbf{M})_1}{1-r}\right) + O(\varepsilon^2) \\ \frac{1}{1+r_1^a} &= \frac{1}{1+r+\varepsilon(\mathbf{a}_1^T \mathbf{M})_1} = \frac{1}{(1+r)\left(1+\frac{\varepsilon(\mathbf{a}_1^T \mathbf{M})_1}{1+r}\right)} \\ &= \frac{1}{1+r}\left(1-\frac{\varepsilon(\mathbf{a}_1^T \mathbf{M})_1}{1+r}\right) + O(\varepsilon^2). \end{aligned}$$

According to the iterated map, for $i \le q_1$, removing terms of $O(\varepsilon^2)$,

$$\begin{aligned} d^a_{2,i} &= \frac{d^a_{1,i}}{1-r^a_1} \\ &\approx \frac{d_{1,i}}{1-r}\left(1+\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1-r}\right)+\frac{\varepsilon a_{1,i}}{1-r} \\ &= \frac{d_{1,i}}{1-r}\left[1+\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1-r}+\frac{\varepsilon a_{1,i}}{d_{1,i}}\right] \\ &= \frac{d_{1,i}}{1-r}\left[1+\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1-r}+\left(\frac{k^a_1}{k_1}-1\right)\right] \\ &= d_{2,i}\left[\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1-r}+\frac{k^a_1}{k_1}\right]. \end{aligned} \tag{14}$$

Since the term in brackets does not depend on i, we know $d^a_{2,i}$ is proportional to $d_{2,i}$ for $i \le q_1$. Recall that for the 3-cycle, the edge of weak classifier 1 must be 0 at iteration 2. Thus, $(\mathbf{d}^a_2\mathbf{M})_1 = 0$, and $\sum_{i=1}^m d^a_{2,i} = 1$, and as before:

$$-\sum_{i=1}^{q_1} d^a_{2,i} + \sum_{i=q_1+1}^{m} d^a_{2,i} = 0 \qquad \text{and} \qquad \sum_{i=1}^{q_1} d^a_{2,i} + \sum_{i=q_1+1}^{m} d^a_{2,i} = 1, \text{ so}$$

$$\sum_{i=1}^{q_1} d^a_{2,i} = \frac{1}{2} \quad \text{and we also have} \quad \sum_{i=1}^{q_1} d_{2,i} = \frac{1}{2}. \tag{15}$$

Combining (14) and (15),

$$\frac{1}{2} = \sum_{i=1}^{q_1} d^a_{2,i} \approx \left[\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1-r}+\frac{k^a_1}{k_1}\right]\sum_{i=1}^{q_1} d_{2,i} = \left[\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1-r}+\frac{k^a_1}{k_1}\right]\frac{1}{2},$$

thus

$$d^a_{2,i} \approx d_{2,i} \quad \text{for} \quad i \le q_1. \tag{16}$$

A similar calculation for $q_1 < i \le q_1 + q_2$ yields

$$\begin{aligned} d^a_{2,i} &= \frac{d^a_{1,i}}{1+r^a_1} \\ &\approx \frac{d_{1,i}}{1+r}\left(1-\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1+r}\right)+\frac{\varepsilon a_{1,i}}{1+r} \qquad (17) \\ &= \frac{d_{1,i}}{1+r}\left[1-\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1+r}+\frac{\varepsilon a_{1,i}}{d_{1,i}}\right] \\ &= \frac{d_{1,i}}{1+r}\left[1-\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1+r}+\left(\frac{k^a_2}{k_2}-1\right)\right] \\ &= d_{2,i}\left[-\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1+r}+\frac{k^a_2}{k_2}\right]. \qquad (18) \end{aligned}$$

And similarly, for $q_1 + q_2 < i \le q_1 + q_2 + q_3$,

$$d^a_{2,i} \approx \frac{d_{1,i}}{1+r}\left(1-\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1+r}\right)+\frac{\varepsilon a_{1,i}}{1+r} = d_{2,i}\left[-\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1+r}+\frac{k^a_3}{k_3}\right]. \tag{19}$$

For the remaining $q_1+q_2+q_3 < i \le m$,

$$d^a_{2,i} = \frac{\varepsilon a_{1,i}}{1+r^a} \approx \frac{\varepsilon a_{1,i}}{1+r}\left[1 - \frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1+r}\right] \approx \frac{\varepsilon a_{1,i}}{1+r}. \tag{20}$$

This last set of components will always shrink as t increases, since the last q_4 examples are always correctly classified. From (16), (18), (19), and (20), we can see that each of the first three sections of $\mathbf{d}_2^a$ is proportional to the corresponding section of $\mathbf{d}_2$ to $O(\varepsilon^2)$, and that the last section of $\mathbf{d}_2^a$ is vanishing.

Now calculating the vector $\mathbf{a}_2$ to $O(\varepsilon)$, incorporating equations (16), (17), (19), and (20):

$$\varepsilon a_{2,i} = d^a_{2,i} - d_{2,i} \approx \left\{ \begin{array}{ccc} 0 & \text{if} & i \le q_1 \\ -\frac{\varepsilon d_{1,i}(\mathbf{a}_1^T\mathbf{M})_1}{(1+r)^2} + \frac{\varepsilon a_{1,i}}{1+r} & \text{if} & q_1 < i \le q_1+q_2+q_3 \\ \frac{\varepsilon a_{1,i}}{1+r} & \text{otherwise} & \end{array} \right\}.$$

We will now show that $\mathbf{a}_1$ undergoes a contraction via the iterated map.

$$\begin{aligned} \varepsilon\|\mathbf{a}_2\|_1 = \|\mathbf{d}_2^a - \mathbf{d}_2\|_1 &\le \varepsilon\left[\frac{\left(\sum_{i=q_1+1}^m d_{1,i}\right)|(\mathbf{a}_1^T\mathbf{M})_1|}{(1+r)^2} + \frac{\|\mathbf{a}\|_1}{1+r}\right] + O(\varepsilon^2) \\ &\le \varepsilon\left[\frac{\left(\sum_{i=q_1+1}^m d_{1,i}\right)\|\mathbf{a}_1\|_1}{(1+r)^2} + \frac{\|\mathbf{a}_1\|_1}{1+r}\right] + O(\varepsilon^2). \end{aligned}$$

Aside, we note $\left(\sum_{i=q_1+1}^m d_{1,i}\right) = 1/2 + (\sqrt{5}-1)/4 = (\sqrt{5}+1)/4$. Also, $r = (\sqrt{5}-1)/2$, so $1/(1+r) = (\sqrt{5}-1)/2$. Now,

$$\begin{aligned} \|\mathbf{a}_2\|_1 &\le \|\mathbf{a}_1\|_1\left[\frac{1+\sqrt{5}}{4}\frac{(\sqrt{5}-1)^2}{4} + \frac{\sqrt{5}-1}{2}\right] + O(\varepsilon) \\ &= \|\mathbf{a}_1\|_1\frac{3(\sqrt{5}-1)}{4} + O(\varepsilon) < \|\mathbf{a}_1\|_1. \end{aligned}$$

Thus, we have shown a contraction at the first iteration. The same calculation (with indices changed accordingly) is valid for each iteration, since the relation between $\mathbf{d}_2^a$ and $\mathbf{d}_2$ is now the same as the relation between $\mathbf{d}_1^a$ and $\mathbf{d}_1$, that is, each section of $\mathbf{d}_2^a$ is proportional to the corresponding section of $\mathbf{d}_2$ from (16), (18), and (19) (and the last section vanishes). Since the calculation is valid for each iteration, there is a contraction to the manifold at every iteration. Hence, the manifold is stable.

Appendix C. 4 Cycle for the 3×3 matrix in the Non-Optimal Case

In this appendix, we prove the following theorem:

Theorem 8 *For the 3×3 matrix analyzed in Section 4, AdaBoost in the non-optimal case may produce a 4-cycle which yields a maximum margin solution.*

Proof We will show the existence of such a 4-cycle by presenting its coordinates, and omit a proof of stability. One coordinate on the cycle is given by

$$\mathbf{d}_1^{cyc} = \left(\frac{1}{2}, \frac{1}{2} - \frac{\sqrt{2}}{4}, \frac{\sqrt{2}}{4}\right)^T.$$

The only choice for j_1 is $j_1 = 2$, and the edge value is $(\mathbf{d}_1^{cycT}\mathbf{M})_2 = 1/\sqrt{2}$, which is larger than 1/3. Now, we compute $\mathbf{d}_2^{cyc}$ using the iterated map:

$$\mathbf{d}_2^{cyc} = \left(\frac{1}{2+\sqrt{2}}, \frac{1}{2}, \frac{1}{\sqrt{2}} - \frac{1}{2}\right)^T.$$

We now choose $j_2 = 1$ even though it is not the optimal choice. We are justified in this choice, since the edge is $(\mathbf{d}_2^{cycT}\mathbf{M})_1 = \sqrt{2} - 1 > 1/3$. Now iterating again, we obtain $\mathbf{d}_3^{cyc}$:

$$\mathbf{d}_3^{cyc} = \left(\frac{1}{2}, \frac{\sqrt{2}}{4}, \frac{1}{2} - \frac{\sqrt{2}}{4}\right)^T.$$

Again, we must choose $j_3 = 3$ since it is the only permissible edge, $(\mathbf{d}_3^{cycT}\mathbf{M})_3 = 1/\sqrt{2}$. Iterating,

$$\mathbf{d}_4^{cyc} = \left(\frac{1}{2+\sqrt{2}}, \frac{1}{\sqrt{2}} - \frac{1}{2}, \frac{1}{2}\right)^T.$$

With the choice $j_4 = 1$, the edge value is $(\mathbf{d}_4^{cycT}\mathbf{M})_1 = \sqrt{2} - 1 > 1/3$, and the next iterate of the map will yield $\mathbf{d}_1^{cyc}$. We have completed the proof. ■

References

Leo Breiman. Arcing the edge. Technical Report 486, Statistics Department, University of California at Berkeley, 1997.

Leo Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–849, 1998.

Leo Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493–1517, 1999.

Bruno Caprile, Cesare Furlanello, and Stefano Merler. Highlighting hard patterns via adaboost weights evolution. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems, Lecture Notes in Computer Science 2364*, pages 72–80. Springer, 2002.

Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 48(1/2/3), 2002.

Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1/2/3):225–254, 2002.

N. Duffy and D. Helmbold. A geometric approach to leveraging weak learners. In *Computational Learning Theory: Fourth European Conference, EuroCOLT '99*. Springer-Verlag, 1999.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 38(2):337–374, April 2000.

Adam J. Grove and Dale Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998.

Vladimir Koltchinskii and Dmitry Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics*, 30(1), February 2002.

T. Y. Li and J. A. Yorke. Period 3 implies chaos. *Amer. Math. Monthly*, 82(10):985–992, 1975.

Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems 12*, 2000.

R. Meir and G. Rätsch. An introduction to boosting and leveraging. In S. Mendelson and A. Smola, editors, *Advanced Lectures on Machine Learning*, LNCS, pages 119–184. Springer Verlag, 2003.

J. R. Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730, 1996.

G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3): 287–320, 2001.

Gunnar Rätsch and Manfred Warmuth. Efficient margin maximizing with boosting. unpublished manuscript, 2002.

Saharon Rosset, Ji Zhu, and Trevor Hastie. Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5:941–973, August 2004.

Cynthia Rudin. *Boosting, Margins and Dynamics*. PhD thesis, Princeton University, April 2004.

Cynthia Rudin, Ingrid Daubechies, and Robert E. Schapire. On the dynamics of boosting. In *Advances in Neural Information Processing Systems 16*, 2004a.

Cynthia Rudin, Robert E. Schapire, and Ingrid Daubechies. Analysis of boosting algorithms using the smooth margin function : A study of three algorithms. Submitted, 2004b.

Cynthia Rudin, Robert E. Schapire, and Ingrid Daubechies. Boosting based on a smooth margin. In *Proceedings of the Sixteenth Annual Conference on Computational Learning Theory*, pages 502–517, 2004c.

Robert E. Schapire. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, October 1998.

Abraham Wyner. Boosting and the exponential loss. In *Proceedings of the Ninth Annual Conference on AI and Statistics*, 2002.