Machine Learning, 43, 293–318, 2001
© 2001 Kluwer Academic Publishers. Manufactured in The Netherlands.

# An Adaptive Version of the Boost by Majority Algorithm

YOAV FREUND
*AT&T Labs, 180 Park Avenue, Florham Park, NJ 07932, USA*

**Editor:** Yoram Singer

**Abstract.** We propose a new boosting algorithm. This boosting algorithm is an adaptive version of the boost by majority algorithm and combines bounded goals of the boost by majority algorithm with the adaptivity of AdaBoost.

The method used for making boost-by-majority adaptive is to consider the limit in which each of the boosting iterations makes an infinitesimally small contribution to the process as a whole. This limit can be modeled using the differential equations that govern Brownian motion. The new boosting algorithm, named BrownBoost, is based on finding solutions to these differential equations.

The paper describes two methods for finding approximate solutions to the differential equations. The first is a method that results in a provably polynomial time algorithm. The second method, based on the Newton-Raphson minimization procedure, is much more efficient in practice but is not known to be polynomial.

**Keywords:** boosting, Brownian motion, AdaBoost, ensamble learning, drifting games

## 1. Introduction

The AdaBoost boosting algorithm has become over the last few years a very popular algorithm to use in practice. The two main reasons for this popularity are simplicity and adaptivity. We say that AdaBoost is *adaptive* because the amount of update is chosen as a function of the weighted error of the hypotheses generated by the weak learner. In contrast, the previous two boosting algorithms (Schapire, 1990; Freund, 1995) were designed based on the assumption that a uniform upper bound, strictly smaller than $1/2$, exists on the weighted error of all weak hypotheses. In practice, the common behavior of learning algorithms is that their error gradually increases with the number of boosting iterations and as a result, the number of boosting iterations required for AdaBoost is far smaller than the number of iterations required for the previous boosting algorithms.

The "boost by majority" algorithm (BBM), suggested by Freund (1995), has appealing optimality properties but has rarely been used in practice because it is not adaptive. In this paper we present and analyze an adaptive version of BBM which we call BrownBoost (the reason for the name will become clear shortly). We believe that BrownBoost will be a useful algorithm for real-world learning problems.

While the success of AdaBoost is indisputable, there is increasing evidence that the algorithm is quite susceptible to noise. One of the most convincing experimental studies that establish this phenomenon has been recently reported by Dietterich (2000). In his

experiments Diettrich compares the performance of AdaBoost and bagging (Breiman, 1996) on some standard learning benchmarks and studies the dependence of the performance on the addition of *classification noise* to the training data. As expected, the error of both AdaBoost and bagging increases as the noise level increases. However, the increase in the error is much more significant in AdaBoost. Diettrich also gives a convincing explanation of the reason of this behavior. He shows that boosting tends to assign the examples to which noise was added much higher weight than other examples. As a result, hypotheses generated in later iterations cause the combined hypothesis to over-fit the noise.

In this paper we consider only binary classification problems. We denote an example by $(x, y)$ where $x$ is the instance and $y \in \{-1, +1\}$ is the label. The weights that AdaBoost assigns to example $(x, y)$ is $e^{-r(x,y)}$ where $r(x, y) = h(x)y$, $y \in \{-1, +1\}$ is the label in the training set and $h(x)$ is the combined "strong" hypothesis which is a weighted sum of the weak hypotheses that corresponds to the instance $x$:

$$h(x) = \sum_i \ln \frac{1 - \epsilon_i}{\epsilon_i} h_i(x),$$

where $\epsilon_i$ denotes the weighted error of $h_i$ with respect to the weighting that was used for generating it. We call $r(x, y)$ the "margin" of the example $(x, y)$. It is easy to observe that if, for a given example $(x, y)$ the prediction of most hypotheses is incorrect, i.e. $h_i(x)y = -1$ then $r(x, y)$ becomes a large negative number and the weight of the example increases very rapidly and without bound.

To reduce this problem several authors have suggested using weighting schemes that use functions of the margin that increase more slowly than $e^x$, for example, the algorithm "Gentle-Boost" of Friedman, Hastie, and Tibshirani (1998); however, none of the suggestions have the formal boosting property as defined in the PAC framework.

The experimental problems with AdaBoost observed by Dietterich and the various recent attempts to overcome these problems have motivated us to have another look at BBM. The weights that are assigned to examples in that algorithm are also functions of the margin.[1] However, the form of the function that relates the margin and the weight is startlingly different than the one used in AdaBoost, as is depicted schematically in figure 1. The weight is a *non-monotone* function of the margin. For small values of $r(x, y)$ the weight increases as $r(x, y)$ decreases in a way very similar to AdaBoost; however, from some point onwards, the weight *decreases* as $r(x, y)$ decreases.

The reason for this large difference in behavior is that BBM is an algorithm that is optimized to minimize the training error *within a pre-assigned number of boosting iterations*. As the algorithm approaches its predetermined end, it becomes less and less likely that examples which have large negative margins will eventually become correctly labeled. Thus it is more optimal for the algorithms to "give up" on those examples and concentrate its effort on those examples whose margin is a small negative number.

To use BBM one needs to pre-specify an upper bound $1/2 - \gamma$ on the error of the weak learner and a "target" error $\epsilon > 0$. In this paper we show how to get rid of the parameter $\gamma$. The parameter $\epsilon$ still has to be specified. Its specification is very much akin to making a "bet" as to how accurate we can hope to make our final hypothesis, in other words, how much inherent noise there is in the data. As we shall show, setting $\epsilon$ to zero

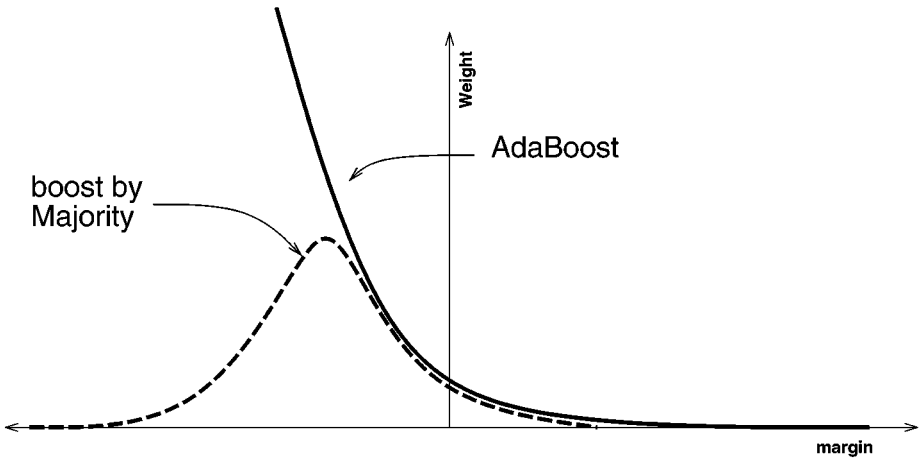


*Figure 1.* A schematic comparison between examples weights as a function of the margin for AdaBoost and BBM.

transforms the new algorithm back into AdaBoost. It can thus be said that AdaBoost is a special case of the new algorithm where the initial "bet" is that the error can be reduced to zero. This intuition agrees well with the fact that AdaBoost performs poorly on noisy datasets.

To derive BrownBoost we analyzed the behavior of BBM in the limit where each boosting iteration makes a very small change in the distribution and the number of iterations increases to infinity. In this limit we show that BBM's behavior is closely related to Brownian motion with noise. This relation leads us to the design of the new algorithm as well as to the proof of its main property.

The paper is organized as follows. In Section 2 we show the relationship between BBM and Brownian motion and derive the main ingredients of BrownBoost. In Section 4 we describe BrownBoost and prove our main theorem regarding its performance. In Section 5 we show the relationship between BrownBoost and AdaBoost and suggest a heuristic for choosing a value for BrownBoost's parameter. In Section 6 we prove that (a variant of) BrownBoost is indeed a boosting algorithm in the PAC sense. In Section 7 we present two solutions to the numerical problem that BrownBoost needs to solve in order to calculate the weights of the hypotheses. We present two solutions, the first is an approximate solution that is guaranteed to work in polynomial time, the second is probably much faster in practice, but we don't yet have a proof that it is efficient in all cases. In Section 8 we make a few remarks on the generalization error we expect for BrownBoost. Finally, in Section 9 we describe some open problems and future work.

## 2. Derivation

In this section we describe an intuitive derivation of algorithm BrownBoost. The derivation is based on a "thought experiment" in which we consider the behavior of BBM when the bound on the error of the weak learner is made increasingly close to $1/2$. This thought

experiment shows that there is a close relation between boost by majority and Brownian motion with drift. This relation gives the intuition behind BrownBoost. The claims made in this section are not fully rigorous and there are not proofs; however, we hope it will help the reader understand the subsequent more formal sections.

In order to use BBM two parameters have to be specified ahead of time: the desired accuracy $\epsilon > 0$ and a non-negative parameter $\gamma > 0$ such that the weak learning algorithm is guaranteed to always generate a hypothesis whose error is smaller than $1/2 - \gamma$. The weighting of the examples on each boosting iteration depends directly on the pre-specified value of $\gamma$. Our goal here is to get rid of the parameter $\gamma$ and, instead, create a version of BBM that "adapts" to the error of the hypotheses that it encounters as it runs in a way similar to AdaBoost.

We start by fixing $\delta$ to some small positive value, small enough that we expect most of the weak hypotheses to have error smaller than $1/2 - \delta$. Given a hypothesis $h$ whose error is $1/2 - \gamma$, $\gamma > \delta$ we define $h'$ to be

$$h'(x) = \begin{cases} h(x), & \text{with probability } \delta/\gamma \\ 0, & \text{with prob. } (1 - \delta/\gamma)/2 \\ 1, & \text{with prob. } (1 - \delta/\gamma)/2 \end{cases}.$$

It is easy to check that the error of $h'(x)$ is exactly $1/2 - \delta$. Assume we use this hypothesis and proceed to the following iteration.

Note that if $\delta$ is very small, then the change in the weighting of the examples that occurs after each boosting iteration is also very small. It is thus likely that the same hypothesis would have an error of less than $1/2 - \delta$ for many consecutive iterations. In other words, instead of calling the weak learner at each boosting iteration, we can instead reuse the same hypothesis over and over until its error becomes larger than $1/2 - \delta$, or, in other words very close[2] to $1/2$. Using BBM in this fashion will result in a combined hypothesis that is a *weighted* majority of weak hypotheses, where each hypothesis has an integer coefficient that corresponds to the number of boosting iterations that it "survived". We have thus arrived at an algorithm whose behavior is very similar to the variant of AdaBoost suggested by Schapire and Singer (To appear). Instead of choosing the weights of weak hypotheses according to their error, we choose it so that the error of the last weak hypothesis on the altered distribution is (very close to) $1/2$.

Note that there is something really strange about the altered hypotheses that we are combining: they contain a large amount of artificially added noise. On each iteration where we use $h(x)$ we add to it some new independent noise; in fact, if $\delta$ is very small then the behavior of $h'(x)$ is *dominated* by the random noise! the contribution of the actual "useful" hypothesis $h$ being proportional to $\delta/\gamma$. Still, there is no problem in principle, in using this modification of BBM and, as long as we can get at least $O(\delta^{-2}\ln(1/\epsilon))$ boosting iterations we are guaranteed that the expected error of the final hypothesis would be smaller than $\epsilon$.

In a sense, we have just described an adaptive version of BBM. We have an algorithm which adapts to the performance of its weak hypotheses, and generates a *weighted* majority vote as its final hypothesis. The more accurate the weak hypothesis, the more iterations of boosting it participates in, and thus the larger the weight it receive in the combined final hypothesis. This is exactly what we were looking for. However, our need to set $\delta$ to be very

small, together with the fact that the number of iterations increases like $O(\delta^{-2})$ makes the running time of this algorithm prohibitive.

To overcome this problem, we push the thought experiment a little further; we let $\delta$ *approach* 0 and characterize the resulting "limit algorithm". Consider some fixed example $x$ and some fixed "real" weak hypothesis $h$. How can we characterize the behavior of the sum of randomly altered versions of $h$, i.e. of $h'_1(x) + h'_2(x) + \cdots$ where $h'_1, h'_2, \ldots$ are randomized alterations of $h$ as $\delta \to 0$?

As it turns out, this limit can be characterized, under the proper scaling, as a well-known stochastic process called *Brownian motion with drift*.[3] More precisely, let us define two real valued variables $t$ and $x$ which we can think about as "time" and "position". We set the time to be $t = \delta^2 i$, so that the time at the end of the boosting process, after $O(\delta^{-2})$ iterations, is a constant independent of $\delta$, and we define the "location" by[4]

$$r_\delta(t) = \delta \sum_{j=1}^{\lceil t/\delta^2 \rceil} h'_j(x).$$

Then as $\delta \to 0$ the stochastic process $r_\delta(t)$ approaches a well defined continuous time stochastic process $r(t)$ which follows a Brownian motion characterized by its mean $\mu(t)$ and variance $\sigma^2(t)$ which are equal to

$$\mu(t) = \int_0^t \frac{1}{\gamma(s)}\, ds, \quad \sigma^2(t) = t,$$

where $1/2 - \gamma(t)$ is the weighted error of the hypothesis $h$ at time $t$. Again, this derivation is not meant as a proof, so we make no attempt to formally define the notion of limit used here; this is merely a bridge to get us to a continuous-time notion of boosting. Note that in this limit we consider the *distribution* of the prediction of the example, i.e. the normal distribution that results from the Brownian motion of the sum $h'_1(x) + h'_2(x) + \cdots$.

So far we have described the behavior of the altered weak hypothesis. To complete the picture we now describe the corresponding limit for the weighting function defined in BBM. The weighting function used there is the binomial distribution (Eq. (1) in Freund (1995)):

$$\alpha_r^i = \binom{k_\delta - i - 1}{\lfloor \frac{k_\delta}{2} \rfloor - r} \left(\frac{1}{2} + \delta\right)^{\lfloor \frac{k_\delta}{2} \rfloor - r} \left(\frac{1}{2} - \delta\right)^{\lceil \frac{k_\delta}{2} \rceil - i - 1 + r}. \tag{1}$$

where $k_\delta$ is the total number of boosting iterations, $i$ is the index of the current iteration, and $r$ is the number of correct predictions made so far. Using the definitions for $t$ and $r_\delta$ given above and letting $\delta \to 0$ we get that $\alpha_r^i$ approaches a limit which is (up to an irrelevant constant factor)

$$\alpha(t, r) = \exp\left(-\frac{(r(t) + c - t)^2}{c - t}\right) \tag{2}$$

where $c = \lim_{\delta \to 0} \delta^2 k_\delta$. Similarly, we find that the limit of the potential function (Eq. (6) in Freund (1995))

$$\beta_r^i = \sum_{j=0}^{\lfloor \frac{k_\delta}{2} \rfloor - r} \binom{k_\delta - i}{j} \left(\frac{1}{2} + \delta\right)^j \left(\frac{1}{2} - \delta\right)^{k_\delta - i - j}. \tag{3}$$

is

$$\beta(t,r) = \frac{1}{2}\left(1 - \mathrm{erf}\left(\frac{r(t)+c-t}{\sqrt{c-t}}\right)\right) \tag{4}$$

where $\mathrm{erf}(\cdot)$ is the so-called "error function":

$$\mathrm{erf}(a) = \frac{2}{\pi}\int_0^a e^{-x^2}\,dx.$$

To simplify our notation, we shall use a slightly different potential function. The use of this potential function will be essentially identical to the use of $\beta_r^i$ in Freund (1995).

Given the definitions of $t, r(t), \alpha(t,r)$ and $\beta(t,r)$ we can now translate the BBM algorithm into the continuous time domain. In this domain we can, instead of running BBM a huge number of very small steps, solve a differential equation which defines the value of $t$ that corresponds to a distribution with respect to which the error of the weak hypothesis is exactly one half.

However, instead of following this route, we now abandon the intuitive trail that lead us here, salvage the definitions of the variables $t, r, c, \alpha(t,r)$ and $\beta(t,r)$ and describe algorithm BrownBoost using these functions directly, without referring to the underlying intuitions.

## 3. Preliminaries

We assume the label $y$ that we are to predict is either $+1$ or $-1$. As in Schapire and Singer's work (Schapire & Singer, 1998), we allow "confidence rated" predictions which are real numbers from the range $[-1,+1]$. The error of a prediction $\hat{y}$ is defined to be

$$\text{error}\,(\hat{y}, y) = |y - \hat{y}| = 1 - y\hat{y}.$$

We can interpret $\hat{y}$ as a randomized prediction by predicting $+1$ with probability $(1+\hat{y})/2$ and $-1$ with probability $(1-\hat{y})/2$. In this case $\text{error}\,(\hat{y}, y)/2$ is the probability that we make a mistaken prediction. A hypothesis $h$ is a mapping from instances into $[-1,+1]$. We shall be interested in the average error of a hypothesis with respect to a training set. A perfect hypothesis is one for which $h(x)y = 1$ for all instances, and a completely random one is one for which $h_i(x) = 0$ for all instances. We call a hypothesis a "weak" hypothesis if its error is slightly better than that of random guessing, which, in our notation, correspond to an average error slightly smaller than 1. It is convenient to measure the strength of a weak hypothesis by its *correlation* with the label which is $(1/m)\sum_{j=1}^m h(x_j)y_j = 1 - (1/m)\sum_{j=1}^m \text{error}\,(h(x_j), y_j)$.

Boosting algorithms are algorithms that define different distributions over the training examples and use a "weak" learner (sometimes called a "base" learner) to generate hypotheses that are slightly better than random guessing with respect to the generated distributions. By combining a number of these weak hypotheses the boosting algorithm creates a combined

hypothesis which is very accurate. We denote the correlation of the hypothesis $h_i$ by $\gamma_i$ and assume that the $\gamma_i$'s are significantly different from zero.

## 4. The algorithm

Algorithm BrownBoost is described in figure 2. It receives as an input parameter a positive real number $c$. This number determines our target error rate; the larger it is, the smaller

---

**Inputs:**
**Training Set:** A set of $m$ labeled examples: $T = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$.
**WeakLearn** — A weak learning algorithm.
$c$ — a positive real valued parameter.
$\nu > 0$ — a small constant used to avoid degenerate cases.

**Data Structures:**
**prediction value:** With each example we associate a real valued *margin*. The margin of example $(x, y)$ on iteration $i$ is denoted $r_i(x, y)$.The initial prediction values of all examples is zero $r_1(x, y) = 0$.

**Initialize** "remaining time" $s_1 = c$.
**Do for** $i = 1, 2, \ldots$

1. Associate with each example a positive weight
$$W_i(x, y) = e^{-(r_i(x,y)+s_i)^2/c}$$
2. Call **WeakLearn** with the distribution defined by normalizing $W_i(x, y)$ and receive from it a hypothesis $h_i(x)$ which has some advantage over random guessing $\sum_{(x,y)} W_i(x, y)h_i(x)y = \gamma_i > 0$.
3. Let $\boldsymbol{\gamma}$, $\boldsymbol{\alpha}$ and $\mathbf{t}$ be real valued variables that obey the following differential equation:
$$\frac{d\mathbf{t}}{d\boldsymbol{\alpha}} = \boldsymbol{\gamma} = \frac{\sum_{(x,y)\in T} \exp\left(-\frac{1}{c}(r_i(x, y) + \boldsymbol{\alpha} h_i(x)y + s_i - \mathbf{t})^2\right) h_i(x)y}{\sum_{(x,y)\in T} \exp\left(-\frac{1}{c}(r_i(x, y) + \boldsymbol{\alpha} h_i(x)y + s_i - \mathbf{t})^2\right)} \tag{5}$$
Where $r_i(x, y)$, $h_i(x)y$ and $s_i$ are all constants in this context.
Given the boundary conditions $\mathbf{t} = 0, \boldsymbol{\alpha} = 0$ solve the set of equations to find $t_i = \mathbf{t}^* > 0$ and $\alpha_i = \boldsymbol{\alpha}^*$ such that either $\boldsymbol{\gamma}^* \leq \nu$ or $\mathbf{t}^* = s_i$.
4. Update the prediction value of each example to
$$r_{i+1}(x, y) = r_i(x, y) + \alpha_i h_i(x)y$$
5. Update "remaining time" $s_{i+1} = s_i - t_i$

**Until** $s_{i+1} \leq 0$
**Output** the final hypothesis,
$$\text{if } \mathrm{p}(x) \in [-1, +1] \text{ then } \quad \mathrm{p}(x) = \mathrm{erf}\left(\frac{\sum_{i=1}^{N} \alpha_i h_i(x)}{\sqrt{c}}\right)$$
$$\text{if } \mathrm{p}(x) \in \{-1, +1\} \text{ then } \quad \mathrm{p}(x) = \mathrm{sign}\left(\sum_{i=1}^{N} \alpha_i h_i(x)\right)$$

---

*Figure 2*. Algorithm BrownBoost.

the target error. This parameter is used to initialize the variable $s$, which can be seen as the "count-down" clock. The algorithm stops when this clock reaches zero. The algorithm maintains, for each example $(x, y)$ in the training set its *margin* $r(x, y)$. The overall structure of the algorithm is similar to that of AdaBoost. On iteration $i$ a weight $W_i(x, y)$ is associated with each example and then the weak learner is called to generate a hypothesis $h_i(\cdot)$ whose correlation is $\gamma_i$.

Given the hypothesis $h_i$ the algorithm chooses a weight $\alpha_i$ and, in addition, a positive number $t_i$ which represents the amount of time that we can subtract from the count-down clock $s_i$. To calculate $t_i$ and $\alpha_i$ the algorithm calculates the solution to the differential Eq. (5). In a lemma below we show that such a solution exists and in later sections we discuss in more detail the amount of computation required to calculate the solution. Given $\alpha_i$ and $t_i$ we update the margins and the count-down clock and repeat. Unlike AdaBoost we cannot stop the algorithm at an arbitrary point but rather have to wait until the count-down clock, $s_i$, reaches the value zero. At that point we stop the algorithm and output our hypothesis. Interestingly, the natural hypothesis to output is a stochastic rule. However, we can use a thresholded truncation of the stochastic rule and get a deterministic rule whose error bound is at most twice as large as the stochastic rule.

In order to simplify our notation in the rest of the paper, we shall use the following shorter notation when referring to a specific iteration. We drop the iteration index $i$ and use the index $j$ to refer to specific examples in the training set. For example $(x_j, y_j)$ and parameters $\boldsymbol{\alpha}$ and $\mathbf{t}$ we define the following quantities:

$$
\begin{aligned}
v_j &= r_i(x_j, y_j) + s_i && \text{old margin}\\
u_j &= h_i(x_j) y_j && \text{step}\\
d_j &= v_j + \boldsymbol{\alpha} u_j - \mathbf{t} && \text{new margin}\\
w_j &= \frac{\exp\left(-d_j^2/c\right)}{\sum_k \exp\left(-d_k^2/c\right)} && \text{weight}
\end{aligned}
$$

Using this notation we can rewrite the definition of $\gamma$ as

$$\gamma(\boldsymbol{\alpha}, \mathbf{t}) = \frac{\sum_j \exp\left(-d_j^2/c\right) u_j}{\sum_j \exp\left(-d_j^2/c\right)} = \sum_j w_j u_j$$

which we shall also write as

$$\gamma(\boldsymbol{\alpha}, \mathbf{t}) = E_w[u]$$

which is a short hand notation that describes the average value of $u_j$ with respect to the distribution defined by $w_j$.

The two main properties of the differential equation that are used in the analysis are the equations for the partial derivatives:

$$\frac{\partial}{\partial \mathbf{t}} \gamma = \frac{2}{c}\left(E_w[du] - E_w[d] E_w[u]\right) = \frac{2}{c} \operatorname{Cov}_w[d, u]. \tag{6}$$

Where $\mathrm{Cov}_w[d, u]$ stands for the covariance of $d_j$ and $u_j$ with respect to the distribution defined by $w_j$. Similarly we find that[5]

$$\frac{\partial}{\partial \boldsymbol{\alpha}}\gamma = \frac{2}{c}(E_w[u]E_w[du] - E_w[du^2]) = -\frac{2}{c}\,\mathrm{Cov}_w[du, u]. \tag{7}$$

The following lemma shows that the differential Eq. (5) is guaranteed to have a solution.

**Lemma 1.** *For any set of real valued constants $a_1, \ldots, a_n$, $b_1, \ldots, b_n$. There is one and only one function $\mathbf{t} : \mathbb{R} \to \mathbb{R}$ such that $\mathbf{t}(0) = 0$ and*

$$\frac{d\mathbf{t}(\boldsymbol{\alpha})}{d\boldsymbol{\alpha}} = f(\boldsymbol{\alpha}, \mathbf{t}) = \frac{\sum_j \exp\left(-\frac{(a_j+\boldsymbol{\alpha}b_j-\mathbf{t})^2}{c}\right)b_j}{\sum_j \exp\left(-\frac{(a_j+\boldsymbol{\alpha}b_j-\mathbf{t})^2}{c}\right)}$$

*and this function is continuous and continuously differentiable.*

**Proof:** In Appendix A. □

The lemma guarantees that there exists a function that solves the differential equation and passes through $\boldsymbol{\alpha} = 0$, $\mathbf{t} = 0$. As $\gamma_i > 0$ we know that the derivative of this function at zero is positive. As the solution is guaranteed to have a continuous first derivative, there are only two possibilities. Either we reach the boundary condition $\gamma = \nu$ or the derivative remains larger than $\nu$, in which case we will reach the boundary condition $\mathbf{t} = s_i$. It is also clear that within the range $\mathbf{t} \in [0, \mathbf{t}^*]$ there is a one-to-one relationship between $\mathbf{t}$ and $\boldsymbol{\alpha}$. We can thus use either $\mathbf{t}$ or $\boldsymbol{\alpha}$ as an index to the solution of the differential equation.

We now prove the main theorem of this paper, which states the main property of BrownBoost. Note that there are no inequalities in the statement or in the proof, only strict equalities!

**Theorem 2.** *If algorithm BrownBoost exits the main loop (i.e. there exists some finite $i$ such that $\sum_i t_i \geq c$) then the final hypothesis obeys*:

$$\frac{1}{m}\sum_{j=1}^{m} |y_j - p(x_j)| = 1 - \mathrm{erf}(\sqrt{c}).$$

**Proof:** We define the potential of the example $(x, y)$ at time $\mathbf{t} > 0$ on iteration $i$ to be

$$\beta_{i,\mathbf{t}}(x, y) = \mathrm{erf}\left(\frac{1}{\sqrt{c}}(r_i(x, y) + \boldsymbol{\alpha}h_i(x)y + s_i - \mathbf{t})\right),$$

and the weight of the example to be

$$W_{i,\mathbf{t}}(x, y) = \exp\left(-\frac{1}{c}(r_i(x, y) + \boldsymbol{\alpha}h_i(x)y + s_i - \mathbf{t})^2\right)$$

where $r_i(x, y)$, $s_i$ and $h_i(x)y$ depend only on $i$ and $\boldsymbol{\alpha}$ depends on the "time" $\mathbf{t}$.

The central quantity in the analysis of BrownBoost is the average potential over the training data. As we show below, this quantity is an *invariant* of the algorithm. In other words, the average potential remains constant throughout the execution of BrownBoost.

When the algorithm starts, $r_1(x, y) = 0$ for all examples, $\boldsymbol{\alpha} = 0$, $s_1 = c$ and $t = 0$; thus the potential of each example is $\mathrm{erf}(\sqrt{c})$ and the average potential is the same.

Equating the average potential at the beginning to the average potential at the end we get that

$$\begin{aligned} m\ \mathrm{erf}\big(\sqrt{c}\big) &= \sum_{(x,y)} \beta_{N+1,0}(x, y) \\ &= \sum_{(x,y)} \mathrm{erf}\left(\frac{r_{N+1}(x, y)}{\sqrt{c}}\right) \\ &= \sum_{(x,y)} \mathrm{erf}\left(\frac{\big(\sum_{i=1}^{N} \alpha_i h_i(x)\big) y}{\sqrt{c}}\right) \\ &= \sum_{(x,y)} y\ \mathrm{erf}\left(\frac{\sum_{i=1}^{N} \alpha_i h_i(x)}{\sqrt{c}}\right). \end{aligned}$$

Plugging in the definition of the final prediction $p(x)$, dividing both sides of the equation by $-m$ and adding 1 to each side we get:

$$1 - \mathrm{erf}\big(\sqrt{c}\big) = 1 - \frac{1}{m} \sum_{(x,y)} y p(x) = \frac{1}{m} \sum_{(x,y)} |y - p(x)|$$

Which is the statement of the theorem.

We now show that the average potential does not change as a function of time.

It follows directly from the definitions that for any iteration $i$ and any example $(x, y)$, the potential of the example does not change between boosting iterations: $\beta_{i,t_i}(x, y) = \beta_{i+1,0}(x, y)$. Thus the average potential does not change at the boundary between boosting iterations.

It remains to show that the average potential does not change within an iteration. To simplify the equations here we use, for a fixed iteration $i$, the notation $\boldsymbol{\beta}_j(\mathbf{t})$ and $\mathbf{W}_j(t)$ to replace $\beta_{i,\mathbf{t}}(x_j, y_j)$ and $W_{i,\mathbf{t}}(x_j, y_j)$ respectively. For a single example we get

$$\frac{d}{d\mathbf{t}} \boldsymbol{\beta}_j(\mathbf{t}) = \frac{2}{c\pi} \mathbf{W}_j(\mathbf{t}) \left[ b_j \frac{d\boldsymbol{\alpha}}{d\mathbf{t}} - 1 \right]$$

The solution to the differential equation requires that

$$\frac{d\boldsymbol{\alpha}}{d\mathbf{t}} = \frac{1}{\gamma}.$$

By using the definition of $\mathbf{W}_j(\mathbf{t})$ and $\gamma$ and averaging over all of the examples we get:

$$\begin{aligned}\frac{d}{d\mathbf{t}}\frac{1}{m}\sum_{j=1}^{m}\beta_j(\mathbf{t}) &= \frac{2}{cm\pi}\left[\frac{1}{\gamma}\sum_{j=1}^{m}(\mathbf{W}_j(\mathbf{t})b_j) - \sum_{j=1}^{m}\mathbf{W}_j(\mathbf{t})\right]\\ &= \frac{2}{cm\pi}\left[\frac{\sum_{j=1}^{m}\mathbf{W}_j(\mathbf{t})}{\sum_{j=1}^{m}(\mathbf{W}_j(\mathbf{t})b_j)}\sum_{j=1}^{m}(\mathbf{W}_j(\mathbf{t})b_j) - \sum_{j=1}^{m}\mathbf{W}_j(\mathbf{t})\right]\\ &= 0\end{aligned}$$

which shows that the average potential does not change with time and completes the proof. □

## 5. Choosing the value of *c*

Running BrownBoost requires choosing a parameter $c$ ahead of time. This might cause us to think that we have not improved much on the situation we had with BBM. There we had two parameters to choose ahead of time: $\gamma$ and $\epsilon$. With BrownBoost we have to choose only one parameter: $c$, but this still seems to be not quite as good as we had it with AdaBoost. There we have no parameters whatsoever! Or do we?

In this section we show that in fact there *is* a hidden parameter setting in AdaBoost. AdaBoost is equivalent to setting the target error $\epsilon$ in BrownBoost to zero.

Observe the functional relationship between $c$ and $\epsilon$ we give in Theorem 2: $\epsilon = 1 - \mathrm{erf}(\sqrt{c})$. Second, note that if we let $\epsilon \to 0$ then we get that $c \to \infty$. It would thus be interesting to characterize the behavior of our algorithm as we let $c \to \infty$.

The solution of Eq. (5) in figure 2 implies that, if the algorithm reaches the "normal" solution where $\gamma(\mathbf{t}) = \nu$ and $\nu = 0$ then the solution $\mathbf{t}^*$, $\boldsymbol{\alpha}^*$ satisfies

$$\frac{\sum_{j=1}^{m}\exp\left(-\frac{(a_j+\boldsymbol{\alpha}^* b_j-\mathbf{t}^*)^2}{c}\right)b_j}{\sum_{j=1}^{m}\exp\left(-\frac{(a_j+\boldsymbol{\alpha}^* b_j-\mathbf{t}^*)^2}{c}\right)} = 0$$

Now, assume that $|h_i(x)|$, $\alpha_i$ and $t_i$ are all bounded by some constant $M$ for iteration $i = 1, \ldots, n$ and let $c \to \infty$; it is easy to see that under these conditions $\lim_{c\to\infty}(a_j/c) = 1$ while all other terms remain bounded by $Mn$. We thus have

$$\begin{aligned}&\lim_{c\to\infty}\frac{\sum_{j=1}^{m}\exp\left(-\frac{(a_j+\boldsymbol{\alpha}^* b_j-\mathbf{t}^*)^2}{c}\right)b_j}{\sum_{j=1}^{m}\exp\left(-\frac{(a_j+\boldsymbol{\alpha}^* b_j-\mathbf{t}^*)^2}{c}\right)}\\ &= \lim_{c\to\infty}\frac{\sum_{j=1}^{m}\exp\left(-\frac{a_j^2+2a_j(b_j\boldsymbol{\alpha}^*-\mathbf{t}^*)}{c}\right)b_j}{\sum_{j=1}^{m}\exp\left(-\frac{a_j^2+2a_j(b_j\boldsymbol{\alpha}^*-\mathbf{t}^*)}{c}\right)}\\ &= \frac{\sum_{j=1}^{m}\exp(-2(a_j+b_j\alpha^*))b_j}{\sum_{j=1}^{m}\exp(-2(a_j+b_j\alpha^*))}\\ &= 0\end{aligned}$$

Note that in this limit there are no dependencies on $c$ or on $\mathbf{t}^*$ which cancel with the denominator. Plugging in the definitions of $a_j$ and $b_j$ we get that the condition for the choice of $\boldsymbol{\alpha}$ is

$$\frac{\sum_{(x,y)} \exp\bigl(-2\bigl(\sum_{i'=1}^{i-1} \alpha_{i'} h_{i'}(x) y + \boldsymbol{\alpha}^* h_{i'}(x) y\bigr)\bigr) h_i(x) y}{\exp\bigl(-2\bigl(\sum_{i'=1}^{i-1} \alpha_{i'} h_{i'}(x) y + \boldsymbol{\alpha}^* h_{i'}(x) y\bigr)\bigr)} = 0$$

If we stare at this last equation sufficiently long we realize that the condition that it defines on the choice of the weight of the $i$'th hypothesis $\alpha_i = \boldsymbol{\alpha}^*$ is *identical* to the one defined by Schapire and Singer in their generalized version of AdaBoost (Schapire & Singer, 1999, Theorem 3).

Note however that another effect of letting $c$ increase without bound is that our algorithm will never reach the condition to exit the loop, and thus we cannot apply Theorem 2 to bound the error of the combined hypothesis. On the other hand, we can use the bounds proven for AdaBoost.[6]

If we set $c = 0$ we get trivially that the algorithm exits the loop immediately. We can thus devise a reasonable heuristic to choose $c$. Start by running AdaBoost (which corresponds to setting $c$ very large in our algorithm) and measure the error of the resulting combined hypothesis on a held-out test set. If this error is very small then we are done. On the other hand, if the error is large, then we set $c$ so that the observed error is equal to $1 - \mathrm{erf}(\sqrt{c})$ and run our algorithm again to see whether we can reach the loop-exit condition. If not—we decrease $c$ further, if yes, we increase $c$. Repeating this binary search we can identify a locally optimal value of $c$, i.e. a value of $c$ for which BrownBoost exits the loop and the theoretical bound holds while slightly larger setting of $c$ will cause BrownBoost to never achieve $\sum_i t_i \geq c$ and exit the loop.

It remains open whether this is also the global maximum of $c$, i.e., whether the legitimate values of $c$ form an (open or closed) segment between 0 and some $c_{\max} > 0$.

## 6. BrownBoost is a boosting algorithm

So far we have shown that BrownBoost has some interesting properties that relate it to BBM and to AdaBoost. However, we have not yet shown that it is indeed a boosting algorithm in the PAC sense. In other words, that it provides a polynomial time transformation of any weak PAC learning algorithm to a strong PAC learning algorithm.

There are two parts to showing that the algorithm is a PAC learning algorithm. First, we should show that when the errors of the weak hypotheses are all smaller than $1/2 - \gamma$ for some $\gamma > 0$ then the algorithm will reach any desired error level $\epsilon$ within a number of boosting iterations that is poly-logarithmic in $1/\epsilon$. Secondly, we need to show that solving the differential equation can be done efficiently, i.e. in polynomial time.

In this section we show the first part, the issue of efficiency will be addressed in the next section. In order to show that a poly-logarithmic number of iterations suffices, we need to show that the "remaining time" parameter, $s_i$ decreases by a constant if the errors are uniformly bounded away from 1/2. As it turns out, this is not the case for

**Parameter:** $\theta > 0$

1. Solve Equation (5) to find $\boldsymbol{\alpha}^*, \mathbf{t}^*$.
2. Let $A = 32\sqrt{c \ln \frac{2}{\theta}}$.
3. If $\mathbf{t}^* \geq \gamma_i^2/A$ then let $(\alpha_i, t_i) = (\boldsymbol{\alpha}^*, \mathbf{t}^*)$.
4. If $\mathbf{t}^* < \gamma_i^2/A$ then find $0 \leq \mathbf{t}' \leq \mathbf{t}^*$ for which $\sum_j \mathbf{W}_j(\mathbf{t}') \leq \theta$ and the corresponding $\boldsymbol{\alpha}'$ and let $(\alpha_i, t_i) = (\boldsymbol{\alpha}', \mathbf{t}' + \gamma_i^2/A)$.

*Figure 3.* A variant of step 3 in algorithm BrownBoost (figure 2). This variant is provably a boosting algorithm in the PAC sense.

BrownBoost itself. Indeed, the decrease in $s_i$ can be arbitrarily small even when the error is constant. However, as we shall see, in this case there is a very simple choice for $\mathbf{t}$ and $\alpha$ in which $\mathbf{t}$ is sufficiently large. This choice is not an exact solution of the differential equation, but, as we shall see, its influence on the average potential is sufficiently small.

In figure 3 we describe the variant of BrownBoost which utilizes this observation. The desired property of this variant is stated in the following theorem.

**Theorem 3.** *Assume that we are using the variant of BrownBoost described in figure* 3. *Let* $1/10 > \epsilon > 0$ *be a desired accuracy and set* $c = (erf^{-1}(1-\epsilon))^2$ *and* $\theta = (\epsilon/c)^2$.

*If the advantages of the weak hypotheses satisfy*

$$\sum_{i=1}^{m} \left(\gamma_i^2 - \nu^2\right) \geq Ac = 32c^{3/2}\sqrt{\ln 2 + 2\ \ln \frac{c}{\epsilon}}$$

*Then the algorithm terminates and the training error of the final hypothesis is at most* $2\epsilon$.

**Corollary 4.** *If* $\gamma_i > \gamma$ *for all* $i$ *then the number of boosting iterations required by BrownBoost to generate a hypothesis whose error is* $\epsilon$ *is* $\tilde{O}((\gamma - \nu)^{-2}(\ln(1/\epsilon))^2)$ (*ignoring factors of order* $\ln \ln 1/\epsilon$.)

**Proof:** As $c \to \infty$, $1 - \mathrm{erf}(\sqrt{c}) = e^{-c}/\sqrt{\pi c}(1 + o(1))$. Thus it is sufficient to set $c = \ln 1/\epsilon$ to guarantee that the initial potential is smaller than $\epsilon$. Plugging this choice of $c$ into the statement of Theorem 3 proves the corollary. □

## 7. Solving the differential equation

In order to show that BrownBoost is an *efficient* PAC boosting algorithm it remains to be shown how we can efficiently solve the differential Eq. (5). We shall show two methods for doing that, the first is of theoretical interest, as we can prove that it requires only polynomial time. The second is a method which in practice is much more efficient but for which we have yet to prove that it requires only polynomial number of steps.

### 7.1. *A polynomial time solution*

The solution described in this section is based on calculating a finite step solution to the differential equation. In other words, we start with $\mathbf{t}_0 = 0$, $\boldsymbol{\alpha}_0 = 0$. Given $\mathbf{t}_k$, $\boldsymbol{\alpha}_k$ we calculate $\gamma_k$ and using it we calculate a small update of $\mathbf{t}_k$, $\boldsymbol{\alpha}_k$ to arrive at $\boldsymbol{\alpha}_{k+1}$. We repeat this process until $\gamma_k < \nu$ at which point we stop and go to the next iteration of BrownBoost. We also check at each point if the total weight is smaller that $\theta$ and, if it is, follow the prescription in figure 3 and set $\mathbf{t}_{k+1} = \mathbf{t}_k = \gamma_k^s/A$, $\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k$.

These small updates do not solve the differential equation exactly, however, we can show that the total decrease in the average potential that they cause can be made arbitrarily small.

This solution method corresponds to solving the differential equation by a small-step approximation. Clearly, this is a crude approximation and the constants we use are far from optimal. The point here is to show that the required calculation can be done in polynomial time. In the next section we describe a solution method which is much more efficient in practice, but for which we don't yet have a proof of efficiency.

**Theorem 5.** *For any* $1/2 > \epsilon > 0$, *if we choose* $c$ *so that* $c \geq \min(1, 18\ln(c/\epsilon))$, *and use, in step* 3 *of BrownBoost the settings* $\alpha_i = \min(\epsilon, \gamma_i)$, $t_i = \alpha_i^2/3$, *then the total decrease in the average potential is at most* $12\epsilon/\sqrt{c}$.

**Proof:** in Appendix C. □

Given this approximation we get that BrownBoost is indeed a poly-time boosting algorithm in the PAC sense. We sketch the analysis of this algorithm below.

Let $\epsilon > 0$ be a small constant which is our desired accuracy. Suppose we have a weak learning algorithm which can, for any distribution of examples, generate a hypothesis whose correlation is larger than $\epsilon$. Then we set $c$ to be large enough so that $c \geq \max(c\ln(c/\epsilon))$ and $\epsilon < 1 - \mathrm{erf}(\sqrt{(c)})$. Both conditions are satisfied by $c = O(\log(1/\epsilon))$. If we now use BrownBoost with the approximate solution described in Theorem 5. We are guaranteed to stop within $3(c/\epsilon^2) = O(\ln(1/\epsilon)/\epsilon^2)$ iterations. The training error of the generated hypothesis, which is the final potential, is at most $(1 + 12/\sqrt{c})\epsilon$.

### 7.2. *A more practical solution*

In this section we describe an alternative method for solving the differential equation, which, we believe (and some initial experiments indicate), is much more efficient and accurate then the previous method.

As we know from the proof of Theorem 2, the exact solution to Eq. (5) is guaranteed to leave the average potential of the examples in the training set unchanged. In other words, the solution for $\boldsymbol{\alpha}$ and $\mathbf{t}$ should satisfy

$$\sum_{j=1}^{m} \mathrm{erf}\left(\frac{a_j + \boldsymbol{\alpha} b_j - \mathbf{t}}{\sqrt{c}}\right) = \sum_{j=1}^{m} \mathrm{erf}\left(\frac{a_j}{\sqrt{c}}\right)$$

On the other hand, the boundary condition $\gamma = 0$ corresponds to the equation

$$\sum_{j=1}^{m} \exp\left(-\frac{(a_j + \boldsymbol{\alpha} b_j - \mathbf{t})^2}{c}\right) b_j = 0. \tag{8}$$

We thus have two nonlinear equations in two unknowns, $t$ and $\boldsymbol{\alpha}$, to which we wish to find a simultaneous solution.

We suggest using Newton-Raphson method for finding the solution. In order to simplify the notation, we use $j$ to index the examples in the sample. Recall that $i$ is the boosting iteration that we will keep fixed in this derivation.

We define $\vec{z} \doteq [\boldsymbol{\alpha}, \mathbf{t}]$, and $\vec{v}_j = [b_j, -1]$. Using this notation we write the two non-linear equations as the components of a function from $\mathbb{R}^2$ to $\mathbb{R}^2$:

$$f(\vec{z}) = \left[\sum_j b_j \ \exp\left(-\frac{1}{c}(a_j + \vec{v}_j \cdot \vec{z})^2\right), \sum_j \left(\operatorname{erf}\left(\frac{a_j + \vec{v}_j \cdot \vec{z}}{\sqrt{c}}\right) - \operatorname{erf}\left(\frac{a_j}{\sqrt{c}}\right)\right)\right]$$

Our goal is to find $\vec{z}$ such that $f(\vec{z}) = [0, 0]$. The Newton-Raphson method generates a sequence of approximate solutions $\vec{z}_1, \vec{z}_2, \ldots, \vec{z}_k$ using the following recursion:

$$\vec{z}_{k+1} = \vec{z}_k - (Df(\vec{z}))^{-1} f(\vec{z}_k),$$

where $Df$ is the Jacobian of the function $f$.

Using the notation $d_j = a_j + \vec{v}_j \cdot \vec{z}$, $w_j = e^{-d_j^2/c}$ and

$$W = \sum_j w_j, \qquad U = \sum_j w_j d_j b_j,$$
$$B = \sum_j w_j b_j, \quad V = \sum_j w_j d_j b_j^2.$$

we can write the Jacobian as follows:

$$Df(\vec{z}) = 2\begin{bmatrix} -V/c & U/c \\ B/\sqrt{\pi c} & -W/\sqrt{\pi c} \end{bmatrix}$$

In order to calculate the inverse of $Df(\vec{z})$ we first calculate the determinant of $Df(\vec{z})$, which is:

$$\det(Df(\vec{z})) = \frac{4}{c\sqrt{\pi c}}(VW - UB)$$

Using the adjoint of $Df(\vec{z})$ we can express the inverse as:

$$(Df(\vec{z}))^{-1} = \frac{-2}{\det(Df(\vec{z}))}\begin{bmatrix} W/\sqrt{\pi c} & U/c \\ B/\sqrt{\pi c} & V/c \end{bmatrix}$$

Combining these Equations, using the subscript $k$ to denote the value of "constants" on the $k$'th iteration of Newton-Raphson, and denoting

$$E_k = \sum_j \left( \mathrm{erf}\left( \frac{d_{j,k}}{\sqrt{c}} \right) - \mathrm{erf}\left( \frac{a_{j,k}}{\sqrt{c}} \right) \right)$$

we find that the Newton-Raphson update step is:

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k + \frac{cW_k B_k + \sqrt{\pi c} U_k E_k}{2(V_k W_k - U_k B_k)}$$

and

$$t_{k+1} = t_k + \frac{cB_k^2 + \sqrt{\pi c} V_k E_k}{2(V_k W_k - U_k B_k)}$$

If we divide the enumerator and denominator by $W_k^2$ we get an expression of the update step that is a function of expected values with respect to the distribution defined by normalizing the weights $w_j$:

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k + \frac{c\hat{B}_k + \sqrt{\pi c} \hat{U}_k \frac{E_k}{W_k}}{2(\hat{V}_k - \hat{U}_k \hat{B}_k)}$$

and

$$t_{k+1} = t_k + \frac{c\hat{B}_k^2 + \sqrt{\pi c} \hat{V}_k \frac{E_k}{W_k}}{2(\hat{V}_k - \hat{U}_k \hat{B}_k)}$$

where

$$\hat{B}_k = \frac{B_k}{W_k}, \qquad \hat{V}_k = \frac{V_k}{W_k}, \qquad \hat{U}_k = \frac{U_k}{W_k}$$

How efficient is this solution method? Newton-Raphson methods are guaranteed to have an asymptotically quadratic rate of convergence for twice differentiable conditions. This means that the error decreases at the rate of $O(e^{-i^2})$ when the starting point is "sufficiently close" to the correct solution. We are currently trying to show that the error in the solution decreases at a similar rate when we start from an easy to calculate starting point, such as the one suggested in Theorem 5.

## 8. The generalization error of BrownBoost

In Schapire et al. (1998) prove theorems (Theorems 1 and 2) which bound the generalization error of a convex combination of classifiers as a function of the margin distribution

of the same combination. Clearly, this theorem can be applied to the output of BrownBoost. Moreover, we claim that BrownBoost is more appropriate than AdaBoost for minimizing these bounds. This is because the bounds consist of two terms: the first is equal to the fraction of training examples whose margin is smaller than $\theta$ and the second is proportional to $1/\theta$. In cases where the data is very noisy one can clearly get better bounds by "giving up" on some of the noisy training examples and allocating them to the first term and by doing that increasing $\theta$ and decreasing the second term. Unlike AdaBoost, BrownBoost can be tuned, using the parameter $c$, to achieve this effect.

One issue that might be important is controlling the $l_1$ norm of the coefficients of the weak hypotheses. In the theorem we assume that $\sum_i |\alpha_i| = 1$. As stated, BrownBoost does not have any control over the norm of the coefficients. However, a simple trick can be used to make sure that the $l_1$ norm is always bounded by 1. Suppose that the weak learner generates the hypothesis $h_i$. Instead of finding the coefficient $\alpha_i$ for $h_i$, we can use the following altered version of $h_i$:

$$\hat{h}_i(x) = h_i(x) - \sum_{j=1}^{i-1} \alpha_j h_j(x).$$

Suppose that $\sum_{j=1}^{i-1} |\alpha_j| = 1$, then, as long as $\alpha_i \leq 1$ all of the coefficients remain positive and their sum remains 1. The case where $\alpha_i = 1$ is degenerate in this case because it effectively eliminates all of the previous hypotheses from the new combination and only the new hypothesis remains. In this case we can remove all of the previous hypotheses from the combination and starting the algorithm with the combined hypothesis being $h_i(x)$.

## 9. Conclusions and future work

We have shown that BrownBoost is a boosting algorithm that possesses some interesting properties. We are planning to experiment with this algorithm extensively in the near future to see how it performs in practice.

There are several technical issues that we would like to resolve regarding BrownBoost. We would like to show that the Newton-Raphson method, or something similar to it, is guaranteed to converge quickly to the solution of the differential equation. We would like to know whether there can be more than one local maximum for $c$. And we would also like to formalize the noise resistant properties of the algorithm and characterize the types of noise it can overcome.

It seems that BrownBoost is optimizing a function of the margin that is much more closely related to the bound proven in Schapire et al. (1998) than AdaBoost. In this regard it seems like it can be a method for "direct optimization of margins" as suggested by Mason, Bartlett, and Baxter (1998). Experiments are needed in order to see whether this theoretical advantage pans out in practice.

The relationship between boosting and Brownian motion has been studied further by Schapire (1999) and by Freund and Opper (2000).

## Appendix

### A. *Proof of Lemma 1*

**Proof:** To prove the lemma we use a standard Lipschitz condition on ordinary differential equations, which we state again here in a slightly simplified form

**Theorem 6** (*Theorem* 7.1.1 *in* Stoler and Bulrisch (1992)). *Let $f$ be defined and continuous on the strip $S := \{(x, y) \mid a \le x \le b, y \in \mathbb{R}\}$, $a, b$ finite. Further let there be a constant $L$ such that*

$$|f(x, y_1) - f(x, y_2| \le L|y_1 - y_2|$$

*for all $x \in [a, b]$ and all $y_1, y_2 \in \mathbb{R}$. Then for every $x_0 \in [a, b]$ and every $y_0 \in \mathbb{R}$ there exists exactly one function $y(x)$ such that*

1. *$y(x)$ is continuous and continuously differentiable for $x \in [a, b]$*
2. *$y'(x) = f(x, y(x))$, for $x \in [a, b]$*
3. *$y(x_0) = y_0$*

As $f$ is infinitely differentiable it suffices to prove a bound on the partial derivative of $f$ with respect to $t$. In our case $f(\boldsymbol{\alpha}, t) = \boldsymbol{\gamma}$. From Eq. (6) we know that

$$\frac{\partial}{\partial t}\boldsymbol{\gamma} = \frac{2}{c}\,\mathrm{Cov}_w[d, u] \tag{9}$$

So it is sufficient if we prove that within any strip $-A \le \alpha \le A$ the value of $d_i$ and $u_i$ are also uniformly bounded. There is a finite number of examples thus $U = \max_i |u_i|$ and $V = \max_i |v_i|$ are finite.[7] It remains to show an upper bound on $d_i = v_i + \boldsymbol{\alpha} u_i - \mathbf{t}$. Unfortunately $\mathbf{t}$ is not bounded on the strip so we need to work a little harder.

To overcome the problem with $\mathbf{t}$ potentially being unbounded we fix some real number $B > 0$ and define $\boldsymbol{\gamma}_B$ to be a clipped version of the function $\boldsymbol{\gamma}$:

$$\boldsymbol{\gamma}_B(\boldsymbol{\alpha}, \mathbf{t}) = \boldsymbol{\gamma}(\boldsymbol{\alpha}, \max(-B, \min(B, \mathbf{t})))$$

Note the $\boldsymbol{\gamma}_B$ is equal to $\boldsymbol{\gamma}$ whenever $|\mathbf{t}| \le A$, and is continuous everywhere. The partial derivative $\partial/\partial\mathbf{t}\boldsymbol{\gamma}_B$ is equal to that that of $\boldsymbol{\gamma}$ when $\mathbf{t} < A$, zero when $\mathbf{t} > A$ and undefined at $\mathbf{t} = A$. When $\mathbf{t} < A$ the magnitude of $d_i$ is bounded by $|d_i| \le V + AU + B$. Using Eq. (9) we conclude that $\boldsymbol{\gamma}_B$ satisfies the conditions of Theorem 6 for the strip $\boldsymbol{\alpha} \in [-A, A]$, from which we conclude that there is one and only one function $\mathbf{t}(\boldsymbol{\alpha})$ which satisfies both $\mathbf{t}(0) = 0$ and $(d\mathbf{t}(\boldsymbol{\alpha})/d\boldsymbol{\alpha}) = \boldsymbol{\gamma}_B(\boldsymbol{\alpha}, \mathbf{t})$ on that strip. Note however that $|\boldsymbol{\gamma}(\boldsymbol{\alpha}, \mathbf{t})| < U$ which implies that also $|\boldsymbol{\gamma}_A(\boldsymbol{\alpha}, \mathbf{t})| < U$. Thus also the derivative $|(d\mathbf{t}(\boldsymbol{\alpha})/d\boldsymbol{\alpha})| \le U$ and thus within the range $\boldsymbol{\alpha} \in [-A, A]$ the function is bounded in $|\mathbf{t}(\boldsymbol{\alpha})| < AU$. Setting $B > AU$ we conclude that the solution of the differential equation defined by $\boldsymbol{\gamma}_A(\boldsymbol{\alpha}, \mathbf{t})$ is the same as the solution of the differential equation defined by $\boldsymbol{\gamma}(\boldsymbol{\alpha}, \mathbf{t})$.

Finally, a solution exists for any setting of $A > 0$ and all of these solutions must conform to each other. Thus there is one solution of the differential equation for the whole real line. □

*B. Proof of Theorem 3*

**Proof:** The proof consists of two parts, corresponding to the two cases that the algorithm can follow on each iteration. In each case we show two properties. First, we show that the difference in the "remaining time" $s_i - s_{i+1}$ is always at least $\gamma_i^2/A$. Second, we show that the total decrease in the average potential from time $s_1 = c$ until $s_i = 0$ is at most $\epsilon$. The parameter $c$ is chosen so that the initial potential is $1 - \epsilon$. Combining these claims we get that the final potential is at least $1 - 2\epsilon$. From this lower bound on the final potential, using the same argument as in the proof of Theorem 2 we find that the error of the final hypothesis is at most $2\epsilon$.

*Case I: Average weight at least $\theta$ throughout the iteration.* The idea of this part of the proof is the following. The initial boundary of the differential equation is $\gamma(0) = \gamma_i$, $\boldsymbol{\alpha} = 0$, $\mathbf{t} = 0$, the final boundary is $\gamma(\mathbf{t}^*) = \nu$. We shall give a lower bound denoted $B$ on $\frac{d\gamma(\boldsymbol{\alpha})}{d\boldsymbol{\alpha}}$ and then use the following integral:

$$\begin{aligned} t^* &= \int_0^{\alpha^*} \frac{dt}{d\boldsymbol{\alpha}}\, d\boldsymbol{\alpha} = \int_0^{\alpha^*} \gamma(\boldsymbol{\alpha})\, d\boldsymbol{\alpha} \\ &\geq \int_0^{\alpha^*} (\gamma(0) - B\boldsymbol{\alpha})\, d\boldsymbol{\alpha} = \gamma(0) + \nu/over2\gamma(0) - \nu/overB \\ &= \frac{\gamma(0)^2 - \nu^2}{2B} \end{aligned} \tag{10}$$

We now compute the lower bound $B$. Using Eqs. (6, 7) we get that

$$\begin{aligned} \frac{d\gamma(\boldsymbol{\alpha})}{d\boldsymbol{\alpha}} &= \frac{\partial}{\partial\boldsymbol{\alpha}}\gamma + \frac{d\mathbf{t}}{d\boldsymbol{\alpha}}\frac{\partial}{\partial t}\gamma \\ &= \frac{\partial}{\partial\boldsymbol{\alpha}}\gamma + \gamma\frac{\partial}{\partial t}\gamma \\ &= \frac{2}{c}(E_w[u]E_w[du] - E_w[du^2]) + \frac{2\gamma}{c}(E_w[du] - E_w[d]E_w[u]) \\ &= \frac{2}{c}(2\gamma E_w[du] - E_w[du^2] - \gamma^2 E_w[d]). \end{aligned}$$

To bound this sum we bound the absolute value of each term. By definition $\gamma \leq 1$. All the other expectations can be bounded using the following lemma.

**Lemma 7.** *Let $b_1, \ldots, b_m, a_1, \ldots, a_m$ be real valued numbers such that $|a_i| \leq |b_i|$. If there exists $0 < \theta \leq 2/e$ $(e = 2.71\ldots)$ such that*

$$\frac{1}{m}\sum_{i=1}^{m} e^{-b_i^2} \geq \theta,$$

*then*

$$\left|\frac{\sum_{i=1}^{m} a_i e^{-b_i^2}}{\sum_{i=1}^{m} e^{-b_i^2}}\right| \le 2\sqrt{\ln\left(\frac{2}{\theta}\right)}.$$

The proof of the lemma is given later.

We continue with the proof of case I. By assumption, in this case $(1/m)\sum_i e^{-d_i^2/c} \ge \theta$ for all $0 \le t \le t^*$. Setting $b_i = d_i/\sqrt{c}, a_i = u_i d_i/\sqrt{c}$, noting that $|u_i| \le 1$ we can apply Lemma 7 and find that $|E_w[du]| \le 2\sqrt{c\ln(2/\theta)}$. Similarly, by setting $a_i = d_i/\sqrt{c}$ or $a_i = u_i^2 d_i/\sqrt{c}$ we find that $|E_w[d]| \le 2\sqrt{c\ln(2/\theta)}$ and $|E_w[du^2]| \le 2\sqrt{c\ln(2/\theta)}$. Combining these bounds we get that

$$\left|\frac{d}{d\boldsymbol{\alpha}}\gamma(\boldsymbol{\alpha})\right| \le \frac{16}{\sqrt{c}}\ln\frac{2}{\theta} \doteq B \tag{11}$$

Combining Eqs. (10) and (11) we find that, on iterations where the total weight remains above $\theta$, $s_{i+1} - s_i \ge (\gamma_i^2 - \nu^2)/2B = (\gamma_i^2 - \nu^2)/A$.

As for conservation of the average potential, we already know, from the proof of Theorem 2, that on iterations where we use the exact solution of the differential equation the total potential does not change. This completes the proof for case I.

*Case II: Average weight smaller than $\theta$ at some point within the iteration.* In this case the claim that $s_i - s_{i+1} \ge \gamma_i^2/A$ follows directly from the construction of the algorithm. What remains to be shown in this case is that the decrease in the average potential is sufficiently small. To do this we show that the speed of the decrease in the potential as a function of time is smaller than $\epsilon/c$, as the total time is $c$ this gives that the maximal total decrease in the potential is $\epsilon$.

The derivative of the average potential w.r.t. $\mathbf{t}$ is the average weight because:

$$\frac{d}{d\mathbf{t}}\beta(\mathbf{t}, \boldsymbol{\alpha}) = -\boldsymbol{W}(\mathbf{t}, \boldsymbol{\alpha}). \tag{12}$$

It remains to be shown that if the average weight at some point is smaller than $\theta$ that it will remain smaller than $\epsilon/c$ for when $\boldsymbol{\alpha}$ is kept unchanged and $\mathbf{t}$ is increased by a quantity smaller or equal to $1/A$ (recall that $\gamma_i^2 < 1$). To this end we use the following lemma.

**Lemma 8.** *If $a_1, \ldots, a_m$ and $\theta > 0$ are real numbers such that*

$$\frac{1}{m}\sum_{j=1}^{m} e^{-a_j^2} \le \theta \tag{13}$$

*then for all $x > 0$*

$$\frac{1}{m}\sum_{j=1}^{m} e^{-(a_j - x)^2} \le e^{-(\sqrt{\ln(1/\theta)} - x)^2} \tag{14}$$

Recall the definitions of $\mathbf{t}'$ and $\boldsymbol{\alpha}'$ in figure 3. We set $\theta = (\epsilon/c)^2$ and $a_j = (v_j + \boldsymbol{\alpha}' u_j - \mathbf{t}')/\sqrt{c}$ and $x = t/\sqrt{c}$ in Lemma 8 and get that for any $0 \le t \le 1/A$

$$\begin{aligned}
\frac{1}{m}\sum_{j=1}^{m} W_j(\mathbf{t}' + t, \boldsymbol{\alpha}') &= \frac{1}{m}\sum_{j=1}^{m} e^{-(a_j - t)^2/c} \\
&\le \exp\left(-\frac{1}{c}\left[\sqrt{c\ln\frac{1}{\theta}} - \frac{1}{A}\right]^2\right) \\
&\le \exp\left(-\left[\sqrt{\ln\frac{1}{\theta}} - \frac{1}{32}\left(\ln\frac{1}{\theta}\right)^{-1/2}\right]^2\right) \\
&\le \sqrt{\theta} = \frac{\epsilon}{c}
\end{aligned}$$

where the last inequality follows from the constraint $\epsilon < 1/10$ which implies that $\theta \le 1/10$. This completes the proof of case II. □

**Proof of Lemma 7:** It is easy to see that to prove the lemma it is sufficient to show for the case $a_i = b_i > 0$ that

$$\sum_{i=1}^{m} b_i e^{-b_i^2} \le 2\sqrt{\ln\left(\frac{2}{\theta}\right)}\sum_{i=1}^{m} e^{-b_i^2} \tag{15}$$

We separate the sum in the LHS of Eq. (15) into three parts:

$$\sum_{i=1}^{m} b_i e^{-b_i^2} = \sum_{b_i \le A} b_i e^{-b_i^2} + \sum_{A < b_i \le 2A} b_i e^{-b_i^2} + \sum_{2A < b_i} b_i e^{-b_i^2} \tag{16}$$

where $A = \sqrt{\ln(2/\theta)}$. Note that as $\theta \le 2/e$, $A \ge 1$.

First, we show that, under the assumption of the lemma, the number of terms for which $b_i \le A$ is large. Denote the number of such terms by $\alpha m$, then the assumption of the lemma implies that

$$\begin{aligned}
\theta m &< \sum_{b_i \le A} e^{-b_i^2} \sum_{b_i > A} e^{-b_i^2} \le \alpha m + (1-\alpha) m e^{-A^2} \\
&= \alpha m + (1-\alpha) m(\theta/2) = m(\theta/2 + \alpha(1 - \theta/2))
\end{aligned}$$

which implies that $\alpha > \theta/2$.

Next we show that the third sum in Eq. (16) is small relative to an expression related to the first sum. Observe that for $x > 1/\sqrt{2}$ the function $xe^{-x^2}$ is monotonically decreasing. Thus, as $2A \ge 2 > 1/\sqrt{2}$ we have that:

$$\sum_{2A < b_i} b_i e^{-b_i^2} \le m 2A e^{-4A^2} \le 2m\sqrt{\ln(2/\theta)}\left(\frac{\theta}{2}\right)^4$$

$$
\begin{aligned}
&< m\left(\frac{\theta}{2}\right)^2 = m\frac{\theta}{2}e^{-A^2} \\
&< \alpha m e^{-A^2} \le \sum_{b_i \le A} e^{-b_i^2}
\end{aligned}
\tag{17}
$$

Combining Eqs. (16) and (17) we get

$$
\begin{aligned}
\sum_{i=1}^{m} b_i e^{-b_i^2} &\le \sum_{b_i \le A} b_i e^{-b_i^2} + \sum_{A<b_i \le 2A} b_i e^{-b_i^2} + \sum_{2A<b_i} b_i e^{-b_i^2} \\
&\le A \sum_{b_i \le A} e^{-b_i^2} + 2A \sum_{A<b_i \le 2A} e^{-b_i^2} + \sum_{b_i \le A} e^{-b_i^2} \\
&\le 2A \sum_{b_i \le 2A} e^{-b_i^2} \le 2A \sum_{i=1}^{m} e^{-b_i^2}
\end{aligned}
$$

which proves Eq. (15) and completes the proof of the lemma. □

**Proof of Lemma 8:** We fix $x > 0$ and maximize the LHS of Eq. (14) under the constraint defined by Eq. (13).

Note first that if for some $1 \le i \le m$, $a_i < 0$, then replacing $a_i$ with $-a_i$ does not change the constrained equation, and increases the LHS of Eq. (14). We can thus assume w.l.o.g. that $\forall i$, $a_i \ge 0$. Furthermore, if $x > a_i \ge 0$ then setting $a_i = x$ reduces the sum in Eq. (13) and increases the sum in Eq. (14). We can thus assume w.l.o.g. that $\forall i$, $a_i \ge x$.

Using the Lagrange method for constrained maximization we find that, for each $j$:

$$
\frac{\partial}{\partial a_j}\left(\sum_{i=1}^{m} e^{-(a_i-x)^2} + \lambda \sum_{i=1}^{m} e^{-a_i^2}\right) = 2(x-a_j)e^{-(a_j-x)^2} + 2\lambda a_j e^{-a_j^2} = 0
$$

Which implies that for all $j$

$$
\lambda = \frac{a_j - x}{a_j} e^{x(2a_j - x)}.
$$

As we assume that $a_j \ge x > 0$ the last expression is positive and monotonically increasing in $a_j$. Thus the only extremum, which is the maximum, occurs when all the $a_j$s are equal and thus all equal to $\sqrt{\ln(1/\theta)}$. Plugging this value into all of the $a_j$s in Eq. (14) completes the proof. □

### C. *Proof of Theorem 5*

**Proof:** The proof follows the same line as the proof of Theorem 2. The difference here is that rather than showing that the average potential stays completely constant, we show that its decrease on any iteration is small.

In what follows we fix the boosting iteration $i$. We denote the potential of an example $(x, y)$ on iteration $i$ by

$$\beta_i(x, y) \doteq \operatorname{erf}\left(\frac{a_i(x, y) + z(x, y)}{c}\right)$$

where $a(x, y) = r_i(x, y) + s_i$ is viewed as the constant part and $z(x, y) = \boldsymbol{\alpha} h_i(x) y - \mathbf{t}$ is the variable part.

We start by focusing on the change in the potential of a single example $(x, y)$ and a single iteration $i$. Later, we will bound the change in the average potential over all examples in a specific iteration. Finally, we will sum the change over all iterations. For this first part we fix and drop the indices $i$ and $(x, y)$. We concentrate on the change in $\beta$ as a function of $z$.

As $\beta(z)$ has an infinite number of derivatives with respect to $z$ we can use the Taylor expansion of third order to estimate it:

$$\beta(z) = \beta(0) + z \frac{\partial}{\partial z}\bigg|_{z=0} \beta(z) + \frac{z^2}{2} \frac{\partial^2}{\partial z^2}\bigg|_{z=0} \beta(z) + \frac{z^3}{6} \frac{\partial^3}{\partial z^3}\bigg|_{z=\theta} \beta(z)$$

where $\theta$ is some number in the range $(0, z)$. Computing the derivatives we get:

$$\begin{aligned} \beta(z) - \beta(0) \;=\; & \frac{2}{\sqrt{\pi c}} \exp\left(-\frac{a^2}{c}\right)\left[z - \frac{a}{c} z^2\right] \\ & + \frac{4}{3\sqrt{\pi} c^{5/2}} ((a+\theta)^2 - c/2) \exp\left(-\frac{(a+\theta)^2}{c}\right) z^3. \end{aligned}$$

By considering the variability of the last term with $\theta$ we get the following bound:

$$\beta(z) - \beta(0) \geq \frac{2}{\sqrt{\pi c}} \exp\left(-\frac{a^2}{c}\right)\left[z - \frac{a}{c} z^2\right] - \frac{2}{3\sqrt{\pi} c^{3/2}} z^3 \tag{18}$$

From our choice of $\alpha_i$ and $t_i$ we get the following bound on $|z(x, y)|$ for all examples $(x, y)$:

$$|z(x, y)| = |\alpha_i h_i(x) y - t_i| = |\alpha_i h_i(x) y| + |t_i| \leq \sqrt{2} |\alpha_i| \tag{19}$$

In order to get an upper bound on the decrease in the average potential on iteration $i$ we sum Inequality (18) over the examples in the training set. We estimate the sum for each power of $z$ in (18) separately.

To bound the sum of the first term in (18) we use the definition of $\gamma_i$ and the fact that $\alpha_i \leq \gamma_i$.

$$\sum_{(x,y)} z(x, y) \frac{2}{\sqrt{\pi c}} \exp\left(-\frac{a_i(x, y)^2}{c}\right)$$

$$
\begin{aligned}
&= \sum_{(x,y)} (\alpha_i h_i(x) y - t_i) \frac{2}{\sqrt{\pi c}} \exp\left(-\frac{a_i(x,y)^2}{c}\right) \\
&= \frac{2}{\sqrt{\pi c}} \left(\sum_{(x,y)} \exp\left(-\frac{a_i(x,y)^2}{c}\right)\right) \left[\alpha_i \gamma_i - \alpha_i^2/3\right] \\
&\geq \frac{2}{\sqrt{\pi c}} \left(\sum_{(x,y)} \exp\left(-\frac{a_i(x,y)^2}{c}\right)\right) \frac{2}{3}\alpha_i^2 \qquad (20)
\end{aligned}
$$

To upper bound the sum of the second term in (18) we use the bound on $|z(x,y)|$. We separate the sum into two parts according to the value of $a_i(x,y)$. For $a_i(x,y) < c/3$ we have

$$
\begin{aligned}
&\frac{2}{\sqrt{\pi c}} \sum_{\substack{(x,y) \\ a_i(x,y) \leq c/3}} \frac{a_i(x,y)}{c} \exp\left(-\frac{a_i(x,y)^2}{c}\right) (z(x,y))^2 \\
&\leq \frac{2}{\sqrt{\pi c}} \left(\sum_{(x,y)} \frac{(z(x,y))^2}{3} \exp\left(-\frac{a_i(x,y)^2}{c}\right)\right) \\
&\leq \frac{2}{\sqrt{\pi c}} \left(\sum_{(x,y)} \exp\left(-\frac{a_i(x,y)^2}{c}\right)\right) \frac{2}{3}\alpha_i^2. \qquad (21)
\end{aligned}
$$

For the case $a(x,y) > c/3$ we use the following technical lemma whose proof is given later.

**Lemma 9.** *If $a, c, \epsilon$ are non-negative real numbers such that $c > \min(1, 18\ln(c/\epsilon))$ and $a > c/3$ then*

$$
\frac{a}{c} \exp\left(-\frac{a^2}{c}\right) \leq \frac{\epsilon}{c}
$$

Combining Lemma 9 with the condition on $c$ and the setting of $t_i$ given in the statement of the theorem we get

$$
\begin{aligned}
&\frac{2}{\sqrt{\pi c}} \sum_{\substack{(x,y) \\ a(x,y) > c/3}} \frac{a_i(x,y)}{c} \exp\left(-\frac{a_i(x,y)^2}{c}\right) (z(x,y))^2 \\
&\leq \frac{2}{\sqrt{\pi c}} m \frac{\epsilon}{c} 2\alpha_i^2 \leq \frac{12m}{\sqrt{\pi c}} \epsilon \frac{t_i}{c} \qquad (22)
\end{aligned}
$$

where the first inequality follows from the fact that are at most $m$ terms in the sum.

Finally, we bound the last term in (18) by using the assumption that $\alpha_i \leq \epsilon$ and the bound on $z$ given in Eq. (19)

$$
\sum_{(x,y)} \frac{2}{3\sqrt{\pi} c^{3/2}} z^3 \leq \frac{2}{3\sqrt{\pi} c^{3/2}} m 2^{3/2} \alpha_i^3 \leq \frac{8m}{\sqrt{\pi c}} \epsilon \frac{t_i}{c} \qquad (23)
$$

Combining the bounds given in Eqs. (20)–(23) we get the following bound on the decrease in the average potential on iteration $i$:

$$\begin{aligned}
&\frac{1}{m}\sum_{(x,y)}(\beta_z(x,y)-\beta_0(x,y))\\
&\quad\geq \frac{2}{m\sqrt{\pi c}}\left(\sum_{(x,y)}\exp\left(-\frac{a_i(x,y)^2}{c}\right)\right)\left(\frac{2}{3}\alpha_i^2-\frac{2}{3}\alpha_i^2\right)-\frac{20}{\sqrt{\pi c}}\epsilon\frac{t_i}{c}\\
&\quad\geq -\frac{20}{\sqrt{\pi c}}\epsilon\frac{t_i}{c}
\end{aligned}$$

Summing this bound over all iterations $i$ and using the assumption that $\sum_i t_i = c$ we get that the total decrease from the initial potential to the final potential is

$$\sum_i \frac{20}{\sqrt{\pi c}}\epsilon\frac{t_i}{c} \leq \frac{20}{\sqrt{\pi c}}\epsilon.$$

Using the same argument as in the proof of Theorem 2 we get the statement of this theorem. □

**Proof of Lemma 9:** We consider two possible ranges for $a$.

If $c/3 \leq a \leq c/\epsilon$ then

$$\frac{a}{c}\exp\left(-\frac{a^2}{c}\right) \leq \frac{1}{\epsilon}\exp\left(-\frac{1}{9}c\right) \leq \frac{\epsilon}{c}$$

because $c \geq 18\ln\frac{c}{\epsilon}$ and $c \geq 1$.

If $a > c/\epsilon$ then

$$\frac{a}{c}\exp\left(-\frac{a^2}{c}\right) \leq \frac{a}{c}\exp\left(-\frac{a}{c}\frac{c}{\epsilon}\right) = \frac{\epsilon}{c}\left(\frac{a}{\epsilon}\exp\left(-\frac{a}{\epsilon}\right)\right) \leq \frac{\epsilon}{c}\frac{1}{e} < \frac{\epsilon}{c}$$

because $xe^{-x} \leq 1/e$ for all $x$. □

## Acknowledgments

Special thanks to Eli Shamir for pointing out to me the similarities between the boost-by-majority and Brownian motion with drift. Thanks to Roland Freund for help with some problems in numerical analysis. Thanks to Rob Schapire for several helpful discussions and insights.

## Notes

1. As BBM generates majority rules in which all of the hypotheses have the same weight, the margin is a linear combination of the *number* of weak hypotheses that are correct and the number of iterations so far.

2. If the error of $h$ is larger than $1/2 + \delta$ then the error of $-h(x)$ is smaller than $1/2 - \delta$.
3. For an excellent introduction to Brownian motion see Breiman (1992); especially relevant is Section 12.2, which describes the limit used here.
4. Note that the sum contains $O(\delta^{-2})$ terms of constant average magnitude and is multiplied by $\delta$ rather than $\delta^2$, thus the maximal value of the sum diverges to $\pm\infty$ as $\delta \to 0$; however, the *variance* of $r_\delta(t)$ converges to a limit.
5. These clean expressions for the derivatives of $\gamma$ are reminiscent of derivatives of the partition function that are often used in Statistical Mechanics. However, we don't at this point have a clear physical interpretation of these quantities.
6. This proof, in fact, follows a similar route to the proof of Theorem 2, but in this case the potential function and the weight function are essentially the same because $\int e^{-x}\,dx = -e^{-x}$, while in our case $\int e^{-x^2} = \mathrm{erf}(x)$.
7. In fact, we assume in the algorithm that $U = 1$. However, we use this more general proof in order to note that the bounded range assumption is not a requirement for the existance of a solution to the differential equation.

## References

Breiman, L. (1992). *Probability*. SIAM, classics edition. Original edition first published in 1968.

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24:2*, 123–140.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning, 40:2*, 139–158.

Freund, Y. (1995). Boosting a weak learning algorithm by majority. *Information and Computation, 121:2*, 256–285.

Freund, Y., & Opper, M. (2000). Continuous drifting games. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory* (pp. 126–132). Morgan Kaufman.

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

Friedman, J., Hastie, T., & Tibshirani, R. (1998). Additive logistic regression: A statistical view of boosting. Technical Report.

Mason, L., Bartlett, P., & Baxter, J. (1998). Direct optimization of margins improves generalization in combined classifiers. Technical report, Deparment of Systems Engineering, Australian National University.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning, 5:2*, 197–227.

Schapire, R. E. (2001). Drifting games. *Machine Learning, 43:3*, 265–291.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics, 26:5*, 1651–1686.

Schapire, R. E., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning, 37:3*, 297–336.

Stoler, J., & Bulrisch, R. (1992). *Introduction to Numerical Analysis*. Springer-Verlag.

Received October 25, 1999
Revised October 25, 1999
Accepted March 3, 2000
Final manuscript October 3, 2000