Maximum Drawdown of a Brownian Motion and AlphaBoost: A Boosting Algorithm

Thesis by

Amrit Pratap

In Partial Fulfillment of the Requirements
for the Degree of
Master of Science

California Institute of Technology
Pasadena, California

2004
(Submitted May 28, 2004)

Acknowledgements

I'd like to thank my advisor, Yaser Abu-Mostafa for his assistance and guidance throughout my research and studies. I'd also like to thank Alexander Nicholson, Ling Li, Malik Magdon-Ismail, Amir Atiya, Gentian Buzi, Dustin Boswell, Carl Gold and Nathan Gray for valuable discussions and meetings.

The work on in chapter 2 on Maximal Drawdown was done in collaboration with Malik Magdon-Ismail and Amir Atiya. I'd especially like to thank Ling Li for a lot of helpful discussions and suggestions on AlphaBoost. Finally I'd like to thank my family and friends for their continuing love and support.

This work was supported by Caltech Center for Neuromorphic Systems Engineering, a National Science Foundation supported Engineering Research Center, under NSF Cooperative Agreement EEC-9402726

Abstract

We study two problems, one in the field of computational finance and the other one in machine learning.

Firstly we study the Maximal drawdown statistics of the Brownian random walk. We give the infinite series representation of its distribution and consider its expected value. For the case when drift is zero, we give an exact expression of the expected value and for the other cases, we give an infinite series representation. For all the cases, we compute the limiting behavior of the expected value.

Secondly, we propose a new algorithm for boosting, AlphaBoost, which does better than AdaBoost in reducing the cost function. We study its generalization properties and compare it to AdaBoost. However, this algorithm does not always give better out-of-sample performance.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Maximum Drawdown of a Brownian Random Walk

The maximum drawdown of a series is defined as the maximum drop from a peak to bottom during a specified period of time. For any trading strategy, the maximum drawdown represents a risk measure. Here we consider the maximum drawdown of a Brownian motion. Let $W(t)$, $0 \leq t \leq T$, be a standard Wiener process and let $X(t)$ be a Brownian motion given by $X(t) = \sigma W(t) + \mu t$, where $\mu \in \Re$ is the drift and $\sigma \geq 0$ is the diffusion parameter. The maximum drawdown is defined as

$$MDD(T;\mu,\sigma) = \max_{s\in[0,T]} (\max_{r\in[0,s]} X(r) - X(s)) \tag{1.1}$$

The maximum drawdown is a very commonly used risk measure for evaluating the performances of funds. It represents the maximum loss that someone can incur if they invest in a stock at the worst possible time. The drawdown can be computed from the historical values of the series. However, it depends on a lot of factors such as the mean return, standard deviation, frequency at which the values are recorded and the time period over which it is computed. These factors have to be taken into consideration while comparing two strategies. Despite its popularity as a risk measure, there is no known theoretical analysis of Maximum Drawdown. Since its a single number computed from the historical data, it'll have a large error associated with it. So in this thesis, we derive the expected maximum drawdown in a specific period for a given return and standard deviation. This will provide guidelines as to what maximum drawdown to expect and to analyze the observed maximum drawdown.

We've computed the distribution of MDD, if we denote $G_{MDD}(h) = \Pr[MDD \geq h]$, we get that

$$G_{MDD}(h) = 2\sigma^4 \sum_{n=1}^{\infty} \frac{\theta_n \sin\theta_n}{\sigma^4\theta_n^2 + \mu^2h^2 - \sigma^2\mu h} e^{-\frac{\mu h}{\sigma^2}} \left(1 - e^{-\frac{\sigma^2\theta_n^2 T}{2h^2}} e^{-\frac{\mu^2 T}{2\sigma^2}}\right) + L \tag{1.2}$$

where L is given by

$$L = \begin{cases} 0 & \mu < \frac{\sigma^2}{h} \\ \frac{3}{e}\left(1 - e^{-\frac{\mu^2 T}{2\sigma^2}}\right) & \mu = \frac{\sigma^2}{h} \\ \frac{2\sigma^2\eta\sinh\eta}{(\sigma^2\mu h - \mu^2h^2 + \sigma^4\eta^2)} e^{-\frac{\mu h}{\sigma^2}} \left(1 - e^{-\frac{\mu^2 T}{2\sigma^2}} e^{-\frac{\sigma^2\eta^2 T}{2h^2}}\right) & \mu > \frac{\sigma^2}{h} \end{cases} \tag{1.3}$$

and θ_n are the positive solution of

$$\tan\theta_n = \frac{\sigma^2}{\mu h}\theta_n \tag{1.4}$$

We've also computed the expected value of MDD as $E[MDD] = \frac{2\sigma^2}{\mu} Q_{MDD}(\alpha^2)$ where $\alpha = \mu\sqrt{T/2\sigma^2}$ and

$$Q_{MDD}(x) = \begin{cases} Q_p(x) & \mu > 0 \\ \gamma\sqrt{2x} & \mu = 0 \\ Q_n(x) & \mu < 0 \end{cases}$$

where $\gamma = \sqrt{\pi/8}$ is a constant and $Q_p$and Q_n are functions whose exact expression are given in Section 2.4

This gives the exact dependence of the expected value of MDD on the drift and diffusion parameter and also on the time period over which it was computed.

1.2 AlphaBoost

Boosting[14] is general technique to improve the accuracy of any weak learning algorithm. It iteratively generates a linear combination of hypothesis from the weak learning algorithm. AdaBoost is the most popular of the boosting methods and in each iteration, it emphasizes the wrongly classified examples from the previous iteration thereby forcing the weak learner to focus on the "difficult" examples. Under some restrictions on the base classifier, AdaBoost is guaranteed to produce a final hypothesis with zero training error after a finite number of iterations.

In practice, AdaBoost has been observed to decrease the out-of-sample error even when the in-sample error is zero. This is contrary to the traditional belief that since AdaBoost is producing increasingly complex hypothesis, it should quickly "overfit" the training data and the out-of-sample performance should go down.

One explanation of the effectiveness of AdaBoost is that it generates large margin classifiers [15]. It has been observed that even after zero training error has been achieved, AdaBoost tends to increase the vote it gives to the correct label which leads to better generalization performance. The out-of-sample error of the final hypothesis produced by AdaBoost can be bounded in terms of the margin of the training points.

Mason et al. [12] present a generalized view of Boosting as a gradient descent procedure on a cost functional. They bound the out-of-sample performance of the final hypothesis in terms of the in-sample value of the cost function. In Chapter 3, we present a more aggressive scheme of optimizing the cost function which leads to smaller in-sample values and then study its out-of-sample performance.

Chapter 2

Maximum Drawdown of a Brownian Random Walk

Maximum Drawdown is an essential aspect of risk assessment in any trading strategy. It offers a more natural measure of the real market risk as it refers to a physical reality, and is less abstract than other measures of risk like variance.

2.1 Maximum Drawdown

The maximum drawdown is defined as the largest drop from a peak to a bottom during a specified period. Figure 2.1 shows the drawdown for a hypothetical Brownian motion.

Nakou et al. [5] discuss the features of Maximum Drawdown as a measure of risk. The two main factors which will affect drawdown are the return μ of the strategy and its variability σ. The value of a strategy which has a high positive return will drift upwards over time. This upward drift will contain some stochastic variations which will cause the value of the strategy to fall from time to time. So, if the upward drift is large or if the variability of the process is low, then the drawdown will be small. So, drawdown is a function of the drift and variability of the underlying process.

Another important factor which affects the drawdown is the frequency of the measurements taken and the time period over which the drawdown is computed. The maximum drawdown will be greater for a longer time series. So if we want to compare the drawdown of two strategies which are reported over different time period,

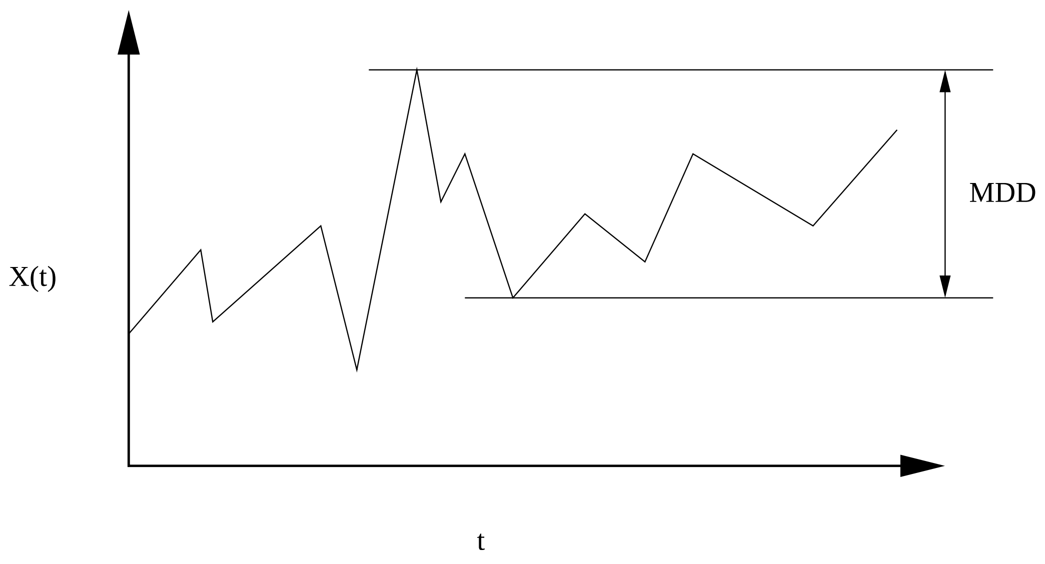


Figure 2.1: Maximum Drawdown

we would need to scale the value of the drawdown appropriately.

Thus, to make drawdown a more informative statistic, we must correct for the length of measurement, volatility and the measurement interval and also know the statistical behavior of the drawdown.

2.2 Notation

Let $X(t)$ be a random walk with drift parameter μ and standard deviation σ. Here t is either discrete of continuous time and we're interested in the drawdown of this random walk in the interval $t \in [0, T]$. We've $X(0) = 0$ and $X(t)$ follows the a path given by an dynamical equation.

2.2.1 Continuous time

A standard Brownian motion with a drift parameter μ and variance parameter σ will have the following dynamics

$$dX(t) = \mu dt + \sigma dW(t) \tag{2.1}$$

here $W(t)$ is the standard wiener process with the following properties

$$E[\frac{dW(t)}{dt}] = 0 \tag{2.2}$$

$$E[\frac{dW(t)}{dt}\frac{dW(s)}{ds}] = \delta(t-s) \tag{2.3}$$

here $\delta(.)$ is the Dirac delta function. Usually $dW(t)$is chosen as normally distributed white noise with $E[dW(t)] = 0$ and $E[dW(t)^2] = dt$. Under these assumptions, the expectations of $X(t)$ can be computed as $E[X(t)] = \mu t$ and $V[X(t)] = \sigma^2 t$.

2.2.2 Discrete time

In this case, the interval of interest $[0, T]$ is broken into n intervals of equal length $\Delta t = T/n$. Defining $X_i = X(i\Delta t)$ for $i = 0, 1, ..n$, and we assume that X_i follows the dynamics

$$X_{i+1} = \begin{cases} X_i + \delta & \text{with prob } p \\ X_i - \delta & \text{with prob. } q\text{=1-}p \end{cases} \tag{2.4}$$

We choose the parameters δ and p such that the random walk converges, in the limit $\Delta t \to 0$, to a Brownian motion on $[0, T]$ with drift μ and diffusion parameter σ. For this, the parameters $\delta$and p must satisfy the following constraints

$$E\left[X_n - X_0\right] = n(p-q)\delta = \mu T \tag{2.5}$$

$$V[X_n - X_0] = \delta^2 = n\sigma^2 \Delta T \tag{2.6}$$

Solving these two equations, we get

$$\delta = \sigma\sqrt{t}\left(1+\frac{\mu^2\Delta t}{\sigma^2}\right)^{1/2} \tag{2.7}$$

$$p = \frac{1}{2}\left(1+\frac{\mu\sqrt{\Delta t}}{\sigma}\left(1+\frac{\mu^2\Delta t}{\sigma^2}\right)^{1/2}\right) \tag{2.8}$$

$$q = \frac{1}{2}\left(1-\frac{\mu\sqrt{\Delta t}}{\sigma}\left(1+\frac{\mu^2\Delta t}{\sigma^2}\right)^{1/2}\right) \tag{2.9}$$

Asymptotically, as $\Delta t \to 0$, $p \to 0.5$ and $\delta \to 0$, both at a rate $\sqrt{\Delta t}$. So, in any statistic of the random walk, to get the corresponding statistic of the continuous case, we take the limit as Δt goes to zero. So asymptotically, we have

$$\delta \to \sigma\sqrt{\Delta t} \tag{2.10}$$

$$p \to \frac{1}{2}\left(1+\frac{\mu\sqrt{\Delta t}}{\sigma}\right) \tag{2.11}$$

2.3 Drawdown of a Discrete Random Walk

Corresponding to the random walk X_t, we define an associated walk for the drawdown D_t, the drawdown from the previous maximum value with $D_0 = 0$. D_t has dynamics very similar to X_t. If X_t goes down, then D_t goes up, and if X_t goes up, then D_t goes down, with the exception that D_t cannot go below zero. So, if X_t follows a random walk with probability p, then D_t follows a random walk with probability $1-p$ and a reflective barrier at 0. The maximum drawdown is then given by

$$MDD = \max_t D_t \tag{2.12}$$

To compute the probability that MDD is greater than h, we add an absorbing barrier at h. So, the random walk D_t will get absorbed if $D_t \geq h$ for some t in $[0, T]$. So, the probability that $MDD \geq h$ is the same as the absorption probability of the random walk. Let $G(h|T)$ be the probability that $MDD \geq h$. Let $f(i|h)$ be the probability that the random walk gets absorbed in the i^{th} time step, then

$$G(h|T) = P[absorbtion \in [0,T]] = \sum_{i=0}^{T/\Delta t} f(i|h) \tag{2.13}$$

$f(i|h)$ was computed by Weesakul [19] for a special case when $p/q < (1+1/N)^2$, the more general case was computed by Blasi [3], which after some corrections is given by

$$f(i|h) = \begin{cases} \tilde{f}(1) & \frac{p}{q} < \left(1+\frac{1}{N}\right)^2 \\ \tilde{f}(2) + \frac{3}{2}\frac{2^i p^{\frac{1}{2}(i-N)} q^{\frac{1}{2}(i+N)}}{(N+1)(N+\frac{1}{2})} & \frac{p}{q} = \left(1+\frac{1}{N}\right)^2 \\ \tilde{f}(3) + \frac{2^i p^{\frac{1}{2}(i-N)} q^{\frac{1}{2}(i+N)} q^{\frac{1}{2}} \cosh^{i-1}\beta \sinh^2\beta}{(N+1)q^{\frac{1}{2}}\cosh(N+1)\beta - Np^{\frac{1}{2}}\cosh N\beta} & \frac{p}{q} > \left(1+\frac{1}{N}\right)^2 \end{cases} \tag{2.14}$$

where $N = h/\delta$ and

$$\tilde{f}(k) = -2^i p^{\frac{1}{2}(i-N)} q^{\frac{1}{2}(i+N)} \sum_{v=k}^{N} \frac{q^{\frac{1}{2}} \cos^{i-1}\alpha_v \sin^2\alpha_v}{(N+1)q^{\frac{1}{2}}\cos(N+1)\alpha_v - Np^{\frac{1}{2}}\cos N\alpha_v} \tag{2.15}$$

where $\alpha_v \in \left(\frac{v\pi}{N-1}, \frac{(v+1)\pi}{N-1}\right)$ satisfies

$$q^{\frac{1}{2}} \sin(N+1)\alpha_v - p^{\frac{1}{2}} \sin N\alpha_v = 0 \tag{2.16}$$

and β satisfies

$$q^{\frac{1}{2}} \sinh(N+1)\beta - p^{\frac{1}{2}} \sinh N\beta = 0 \tag{2.17}$$

Substituting 2.14 and 2.15 in 2.13, we get the distribution function in the discrete case. Using δ and p as given in 2.11, and taking the limit as $\Delta t \to 0$, will give us the continuous case. Suppose we let $\hat{f}_\tau(t|h)$ be the corresponding density function of absorption in $[t, t+\Delta t]$ defined by $\hat{f}_\tau(t|h)\Delta t = f(t/\Delta t|h)$, then

$$G(h|T) = \sum_{i=0}^{T/\Delta t} \Delta t f_\tau(i\hat{\Delta} t|h) \tag{2.18}$$

Since in the limit as $\Delta t \to 0$, we have $\hat{f}_\tau(t|h) \to f_\tau(t|h)$, the continuous time absorption density, we have in the limit $\Delta t \to 0$

$$G(h|T) = \int_0^T dt f_\tau(t|h) \tag{2.19}$$

Denote $\lambda = \frac{\mu\sqrt{\Delta t}}{\sigma}\left(1 + \frac{\mu^2 \Delta t}{\sigma^2}\right)^{1/2}$, so $p = \frac{1}{2}(1+\lambda)$ and in the limit $\Delta t \to 0$, we have $\lambda \to \frac{\mu\sqrt{\Delta t}}{\sigma}$

Using the identity $\lim_{x\to\infty}\left(1+\frac{1}{x}\right)^x = e$, we get

$$2^i p^{\frac{1}{2}(i-N)} q^{\frac{1}{2}(i+N)} = (1-\lambda^2)^{\frac{i}{2}} \left(\frac{1-\lambda}{1+\lambda}\right)^{\frac{N}{2}} \to e^{-\frac{\mu^2 t}{2\sigma^2}} e^{-\frac{\mu h}{\sigma^2}} \tag{2.20}$$

Expanding the eigen value condition for α_v in 2.16 to the first order in λ, and on simplification, we get

$$\tan\left(N+\frac{1}{2}\right)\alpha_v \cos\alpha_v = \frac{2}{\lambda}\sin\frac{\alpha_v}{2}$$

Since $\alpha_v \in \left(\frac{v\pi}{N-1}, \frac{(v+1)\pi}{N-1}\right)$, let $\theta_v = \left(N+\frac{1}{2}\right)\alpha_v$. For fixed v, $\alpha_v \to 0$, so we take the first order expansion in α_v to get

$$\tan\theta_v = \frac{\sigma^2}{\mu h}\theta_v$$

with $\theta_v \in \left(v\pi\frac{N+\frac{1}{2}}{N-1}, (v+1)\pi\frac{N+\frac{1}{2}}{N-1}\right) \to (v\pi, (v+1)\pi]$. Similarly analyzing the eigen value condition for β, we get

$$\tanh\left(N+\frac{1}{2}\right)\beta\cosh\beta = \frac{2}{\lambda}\sinh\frac{\beta}{2}$$

Defining $\eta = \left(N+\frac{1}{2}\right)\beta$, and taking the limit, we get

$$\tanh\eta = \frac{\sigma^2}{\mu h}\eta$$

Consider the summand in $\tilde{f}$. Since $\sin\alpha_v \to \frac{\theta_v}{N+\frac{1}{2}}$, the summand becomes

$$\frac{\theta_v^2 \cos^{i-1}\alpha_v}{(N+\frac{1}{2})^2(N+1)\left[\cos(\theta_v+\frac{1}{2}\alpha_v) - A\cos(\theta_v-\frac{1}{2}\alpha_v)\right]} \tag{2.21}$$

where $A = \frac{(1+\lambda)^{\frac{1}{2}}}{(1+\frac{1}{N})(1-\lambda)^{\frac{1}{2}}}$. Using the identity $\lim_{x\to 0}\cos^{1/x^2} x = e^{-1/2}$, we get that $\cos^{i-1}\alpha_v \to e^{-\frac{\sigma^2\theta_v^2 t}{2h^2}}$. Thus, after simplifications and taking the limit, we have that the summand in $\tilde{f}$ becomes

$$\frac{\Delta t\sigma^2}{h^2}\frac{\theta_v^2 e^{-\frac{\sigma^2\theta_v^2 t}{2h^2}}}{\left(1-\frac{\mu h}{\sigma^2}\right)\cos\theta_v - \theta_v\sin\theta_v} = \frac{-\theta_v\sin\theta_v[\sigma^4\theta_v^2+\mu^2h^2]}{[\sigma^4\theta_v^2+\mu^2h^2-\mu h\sigma^2]} \tag{2.22}$$

Plugging all these results into 2.15 and dividing by Δt, we get the continuous time limit of $\tilde{f}$,

$$\tilde{f}(k) \to e^{-\frac{\mu^2 t}{2\sigma^2}} e^{-\frac{\mu h}{\sigma^2}} \frac{\sigma^2}{h^2}\sum_{v=k}^{\infty}\frac{\theta_v\sin\theta_v[\sigma^4\theta_v^2+\mu^2h^2]e^{-\frac{\sigma^2\theta_v^2 t}{2h^2}}}{[\sigma^4\theta_v^2+\mu^2h^2-\mu h\sigma^2]} \tag{2.23}$$

The three cases in 2.14, correspond in the limit to the conditions $\mu < \frac{\sigma^2}{h}$, $\mu = \frac{\sigma^2}{h}$ and $\mu > \frac{\sigma^2}{h}$.

The last two cases have an additional term whose limits can be computed using 2.20. For the second case, when $\frac{p}{q} = \left(1+\frac{1}{N}\right)^2$, using 2.20 and dividing by Δt, we get the limit of the extra term as

$$e^{-\frac{\mu^2 t}{2\sigma^2}}\frac{3\sigma^2}{2eh^2} \tag{2.24}$$

For the third case, $\frac{p}{q} > \left(1+\frac{1}{N}\right)^2$, defining $\eta = \left(N+\frac{1}{2}\right)\beta$, the identity $\lim_{x\to 0}\cosh^{1/x^2} x = e^{1/2}$, we get $\cosh^{i-1}\beta \to e^{\frac{\sigma^2\eta^2 t}{2h^2}}$. So, we get the additional term in this case as

$$\frac{\Delta t\sigma^2}{h^2}\frac{\eta^2 e^{-\frac{\sigma^2\eta^2 t}{2h^2}}}{N(1-A)\cosh\eta\cosh\frac{1}{2}\beta - N(1+A)\sinh\eta\sinh\frac{1}{2}\beta} = \frac{\eta\sinh\eta[\mu^2h^2-\sigma^4\eta^2]}{[\sigma^4\eta^2-\mu^2h^2+\mu h\sigma^2]} \tag{2.25}$$

Again using 2.20 and dividing by Δt, we get the limit of the extra term as

$$e^{-\frac{\mu^{2}t}{2\sigma^2}}\frac{\sigma^2}{h^2}\frac{(\mu^2h^2-\sigma^4\eta^2)\eta\sinh\eta}{(\sigma^2\mu h-\mu^2h^2+\sigma^4\eta^2)}e^{-\frac{\mu h}{\sigma^2}}e^{-\frac{\sigma^2\eta^2t}{2h^2}} \tag{2.26}$$

Combining 2.23 ,2.24 and 2.26 in 2.14 , we compute the continuous limit of the discrete time density as

$$f_\tau(t|h)=e^{-\frac{\mu^2t}{2\sigma^2}}\left[\frac{\sigma^2}{h}\sum_{n=0}^{\infty}\frac{(\sigma^4\theta_n^2+\mu^2h^2)\theta_n\sin\theta_n}{(\sigma^4\theta_n^2+\mu^2h^2-\sigma^2\mu h)}e^{-\frac{\mu h}{\sigma^2}}e^{-\frac{\sigma^2\theta_n^2t}{2h^2}}+K\right] \tag{2.27}$$

where θ_n are the positive solutions to the eigen value condition

$$\tan\theta_n=\frac{\sigma^2}{\mu h}\theta_n \tag{2.28}$$

and K is given by

$$K=\begin{cases}0 & \mu<\frac{\sigma^2}{h}\\ \frac{3\sigma^2}{2eh^2} & \mu=\frac{\sigma^2}{h}\\ \frac{\sigma^2}{h^2}\frac{(\mu^2h^2-\sigma^4\eta^2)\eta\sinh\eta}{(\sigma^2\mu h-\mu^2h^2+\sigma^4\eta^2)}e^{-\frac{\mu h}{\sigma^2}}e^{-\frac{\sigma^2\eta^2t}{2h^2}} & \mu>\frac{\sigma^2}{h}\end{cases} \tag{2.29}$$

where η is the positive solution to the eigen value condition

$$\tanh\eta=\frac{\sigma^2}{\mu h}\eta \tag{2.30}$$

Substituting 2.27 in 2.19, and taking the integration, we get

$$G_{MDD}(h|T)=2\sigma^4\sum_{n=1}^{\infty}\frac{\theta_n\sin\theta_n}{\sigma^4\theta_n^2+\mu^2h^2-\sigma^2\mu h}e^{-\frac{\mu h}{\sigma^2}}\left(1-e^{-\frac{\sigma^2\theta_n^2T}{2h^2}}e^{-\frac{\mu^2T}{2\sigma^2}}\right)+L \tag{2.31}$$

where L is given by

$$L = \begin{cases} 0 & \mu < \frac{\sigma^2}{h} \\ \frac{3}{e}\left(1 - e^{-\frac{\mu^2 T}{2\sigma^2}}\right) & \mu = \frac{\sigma^2}{h} \\ \frac{2\sigma^2 \eta \sinh \eta}{(\sigma^2 \mu h - \mu^2 h^2 + \sigma^4 \eta^2)} e^{-\frac{\mu h}{\sigma^2}} \left(1 - e^{-\frac{\mu^2 T}{2\sigma^2}} e^{-\frac{\sigma^2 \eta^2 T}{2h^2}}\right) & \mu > \frac{\sigma^2}{h} \end{cases} \tag{2.32}$$

2.4 Expected Drawdown

To get the expectation of the maximum drawdown, we use the identity $E[MDD|T] = \int_0^\infty G(h|t)dh$, which holds for positive valued random variables. The behavior of the expectation is different depending on the sign on μ. We will analyze the expectation under the following three cases on the sign of μ.

2.4.1 $\mu = 0$

For the case when $\mu = 0$, the eigen value condition 2.28 is solved by $\theta_n = (n - \frac{1}{2})\pi$. Thus, we have

$$\begin{aligned} G(h|T) &= 2\sum_{n=1}^{\infty} \frac{\sin(n - \frac{1}{2})\pi}{(n - \frac{1}{2})\pi} \left(1 - e^{-\frac{\sigma^2 (n-\frac{1}{2})^2 \pi^2 T^2}{2h^2}}\right) && (2.33) \\ &= \frac{2}{\pi} \sum_{n=0}^{\infty} \frac{(-1)^n}{(n + \frac{1}{2})\pi} \left(1 - e^{-\frac{\sigma^2 (n+\frac{1}{2})^2 \pi^2 T^2}{2h^2}}\right) && (2.34) \end{aligned}$$

The expected value of the maximum drawdown can then be calculated as

$$\begin{aligned} E[MDD] &= \int_0^\infty G(h|T)dh && (2.35) \\ &= \frac{2}{\pi} \int_0^\infty \sum_{n=0}^{\infty} \frac{(-1)^n}{(n + \frac{1}{2})\pi} \left(1 - e^{-\frac{\sigma^2 (n+\frac{1}{2})^2 \pi^2 T^2}{2h^2}}\right) dh && (2.36) \\ &= 2\sigma\sqrt{T} \int_0^\infty \sum_{n=0}^{\infty} \frac{(-1)^n}{(n + \frac{1}{2})\pi} \left(1 - e^{-\frac{(n+\frac{1}{2})^2}{2h^2}}\right) dh && (2.37) \end{aligned}$$

Defining the constant γ by

$$\gamma = \int_0^\infty \sum_{n=0}^\infty \frac{(-1)^n}{(n+\frac{1}{2})\pi} \left(1 - e^{-\frac{(n+\frac{1}{2})^2}{2h^2}}\right) dh \approx 0.6276 \tag{2.38}$$

we have that $E[MDD|T] = 2\sigma\gamma\sqrt{T}$. The value of γ was obtained by numerical integration. Alternatively, the value of γ was computed analytically by Greg Bond [4], as $\gamma = \sqrt{\pi/8} \approx 0.6267$.

2.4.2 $\mu < 0$

Taking the eigen condition

$$\begin{aligned} E[MDD] &= \int_0^\infty G(h|T)dh & (2.39)\\ &= 2\int_0^\infty e^{-\frac{\mu h}{\sigma^2}} \sum_{n=1}^\infty \frac{\sin^3\theta_n}{\theta_n - \cos\theta_n \sin\theta_n} \left(1 - e^{-\frac{\mu^2 T}{2\sigma^2\cos^2\theta_n}}\right) dh & (2.40)\end{aligned}$$

making change of variables to $u = -\mu h/\sigma^2$, we get

$$E[MDD] = -2\frac{\sigma^2}{\mu} \int_0^\infty e^u \sum_{n=1}^\infty \frac{\sin^3\theta_n}{\theta_n - \cos\theta_n \sin\theta_n} \left(1 - e^{-\frac{\mu^2 T}{2\sigma^2\cos^2\theta_n}}\right) du \tag{2.41}$$

where $\tan\theta_n = -\theta_n/u$. So we get

$$E[MDD] = -\frac{2\sigma^2}{\mu} Q_n(\alpha^2) \tag{2.42}$$

where $\alpha = \mu\sqrt{\frac{T}{2\sigma^2}}$ and for some function $Q_n(.)$. The numerical computation of $Q_n(x)$ is not straightforward. The summation in the integrand is a function of u that decreases faster than e^{-u}. Since the magnitude of the n^{th} term is approximately $1/n$, we need to take $\Omega(e^u)$ terms in the summation to make sure that the next term left out has a magnitude less than the size of the sum. Appendix A gives the approximate values of $Q_n(x)$ for various values of x, computed using numerical integration.

We know that $Q_n(x) \to \gamma\sqrt{2x}$ when $x \to 0^+$, since in the limit, we must recover the $\mu \to 0$ behavior. For the behavior of Q_n as $x \to \infty$, we first bound the MDD in terms of the Range R and the Low L of the Brownian motion. The range R is defined as the difference between the maximum value and the minimum value attained by the random walk. We've

$$R \geq MDD \geq L \tag{2.43}$$

The range and low were analyzed by Malik et al. [10]. From this we get

$$\frac{\alpha^2}{2} + \frac{Q_R(-\alpha)}{2} \leq Q_n(\alpha^2) \leq Q_R(-\alpha) \tag{2.44}$$

where $Q_R(x) = erf(x)\left(\frac{1}{2} + x^2\right) + \frac{1}{\sqrt{\pi}} x e^{-x^2}$. Asymptotically, as $\alpha \to \infty$, this gives

$$\alpha^2 + \frac{1}{4} \leq Q_n(\alpha^2) \leq \alpha^2 + \frac{1}{2} \tag{2.45}$$

So, $Q_n(x) \to x + \epsilon(x)$ where $\frac{1}{4} \leq \epsilon(x) \leq \frac{1}{2}$. Since $Q_n(x)$ is a monotonically increasing function of x, we conclude that $\epsilon(x) \to D_\infty$ for some constant $D_\infty \in [\frac{1}{4}, \frac{1}{2}]$. Asymptotically, since $\mu < 0$, we'll have that $E[MDD] = E[R]$ and so, $D_\infty = \frac{1}{2}$. This fact can be verified numerically.

So, we have

$$Q_n(x) \to \begin{cases} \gamma\sqrt{2x} & x \to 0^+ \\ x + \frac{1}{2} & x \to \infty \end{cases} \tag{2.46}$$

2.4.3 $\mu > 0$

In this case, for $h > \frac{\sigma^2}{\mu}$ in the integral, the third case for L adds another term. So have have

$$\begin{aligned} E[MDD] &= \int_0^\infty G(h|T)dh \qquad (2.47)\\ &= 2\int_0^\infty e^{-\frac{\mu h}{\sigma^2}} \sum_{n=1}^{\infty} \frac{\sin^3\theta_n}{\theta_n - \cos\theta_n \sin\theta_n}\left(1 - e^{-\frac{\mu^2 T}{2\sigma^2 \cos\theta_n}}\right) dh - \\ & \quad 2\int_{\sigma^2/h}^\infty e^{-\frac{\mu h}{\sigma^2}} \frac{\sinh^3\eta}{\eta - \cosh\eta \sinh\eta}\left(1 - e^{-\frac{\mu T^2}{2\sigma^2 \cosh^2\eta}}\right) dh \qquad (2.48) \end{aligned}$$

The second integral can be reduced by a change of variables $u = \eta(h)$. Since $\tanh u = \sigma^2 u/\mu h$, we have

$$\frac{dh}{du} = \frac{\sigma^2}{\mu}\frac{\cosh u \sinh u - u}{\sinh^2 u} \qquad (2.49)$$

So, the integral reduces to

$$-\frac{\sigma^2}{\mu}\int_0^\infty e^{-\frac{u}{\tanh u}}\left(1 - e^{-\frac{\mu^2 T}{2\sigma^2 \cosh^2 u}}\right) du \qquad (2.50)$$

Changing variable in the first integral to $u = \mu h/\sigma^2$, we get

$$E[MDD] = 2\frac{\sigma^2}{\mu}\int_0^\infty e^{-u}\sum_{n=1}^{\infty}\frac{\sin^3\theta_n}{\theta_n - \cos\theta_n\sin\theta_n}\left(1 - e^{-\frac{\mu^2 T}{2\sigma^2\cos\theta_n}}\right) + e^{-\frac{u}{\tanh u}}\sinh u\left(1 - e^{-\frac{\mu^2 T}{2\sigma^2\cosh^2 u}}\right) du \qquad (2.51)$$

where $\tan\theta_n = \frac{\theta_n}{u}$. So, we have

$$E[MDD] = \frac{2\sigma^2}{\mu}Q_p(\alpha^2) \qquad (2.52)$$

for some function Q_p and $\alpha = \mu\sqrt{T/2\sigma^2}$. We know that $Q_p(x) \rightarrow \gamma\sqrt{2x}$ when $x \rightarrow 0^+$. For the case when $x \rightarrow \infty$, we will evaluate the two terms in Q_p separately. Consider $I_1(x)$ given by

$$I_1(x) = \int_0^\infty e^{-u} \sum_{n=1}^\infty \frac{\sin^3 \theta_n}{\theta_n - \cos\theta_n \sin\theta_n} \left(1 - e^{-\frac{x}{\cos\theta_n}}\right) du \tag{2.53}$$

Since $0 \leq \cos^2 x \leq 1$ and $x \to \infty$, the term in the brackets is rapidly approaching 1. Since e^{-u} is rapidly decreasing, we can interchange the summation and the integration. Using the change of variable $v = \theta_n(u)$ and the identity

$$du = \frac{\cos v \sin v - v}{\sin^2 v} dv \tag{2.54}$$

we get

$$I_1(x) = \sum_{n=0}^\infty \int_{n\pi}^{n+\frac{1}{2}\pi} e^{-\frac{v}{\tan v}} \sin v \left(1 - e^{-\frac{x}{\cos^2 v}}\right) dv \tag{2.55}$$

By translating each integral by $n\pi$ and bringing the summation inside the integral, the summation becomes a geometric series which can be evaluated in closed form to get

$$I_1(x) = \int_0^{\pi/2} \frac{e^{-\frac{v}{\tan v}} \sin v \left(1 - e^{-\frac{x}{\cos^2 v}}\right)}{1 + e^{-\frac{\pi i}{\tan v}}} dv \tag{2.56}$$

So, we have

$$(1 - e^{-x})\beta_1 \leq I_1(x) \leq \beta_1 \tag{2.57}$$

where

$$\beta_1 = \int_0^{\pi/2} \frac{e^{-\frac{v}{\tan v}} \sin v}{1 + e^{-\frac{\pi i}{\tan v}}} dv \tag{2.58}$$

So, we'll have that $I_1(x) \to \beta_1$ as $x \to \infty$. β_1 can be evaluated numerically to give $\beta_1 = 0.4575$.

Consider the second term in Q_p,

$$I_2(x) = \int_0^\infty e^{-\frac{u}{\tanh u}} \sinh u \left(1 - e^{-\frac{\mu^2 T}{2\sigma^2 \cosh^2 u}}\right) du \tag{2.59}$$

The term in the brackets is the only place where x appears. When x is large, this term is very close to 1 until u gets large enough so that $\cosh u \sim x$, from which point, the term quickly decays to 0. The term $e^{-\frac{u}{\tanh u}} \sinh u$ is always less than $\frac{1}{2}$ and rapidly increases from 0 to $\frac{1}{2}$. So, we write

$$\begin{aligned} I_2(x) &= \int_0^\infty \left(e^{-\frac{u}{\tanh u}} \sinh u - \frac{1}{2} + \frac{1}{2}\right)\left(1 - e^{-\frac{x}{\cosh^2 u}}\right) du \qquad (2.60) \\ &= \frac{1}{2}\int_0^\infty \left(1 - e^{-\frac{x}{\cosh^2 u}}\right) du - \int_0^\infty \left(\frac{1}{2} - e^{-\frac{u}{\tanh u}} \sinh u\right) du + \\ &\quad \int_0^\infty e^{-\frac{x}{\cosh^2 u}} \left(\frac{1}{2} - e^{-\frac{u}{\tanh u}} \sinh u\right) du \qquad (2.61) \end{aligned}$$

The third integral approaches 0 as $x \to \infty$ since the first term is small when u is small and the second term is small when u is large. The second integral is a constant independent of x and can be evaluated numerically to give $\beta_2 \approx 0.4575$, which is numerically equal to β_1.

For bounding the first integral, we use the fact that $\cosh u \geq \frac{1}{2}e^u$ and for $u \geq A$, $\cosh u \leq \frac{1}{2}e^{\lambda(A)u}$, where $\lambda(A) = 1 + \frac{e^{-2A}}{A}$. Denoting the first integral by $F(x)$, we get the following bound

$$A\left(1 - e^{-\frac{x}{\cosh^2 A}}\right) + \int_A^\infty (1 - e^{-4xe^{-2\lambda(A)u}})du \leq 2F(x) \leq \int_A^\infty (1 - e^{-4xe^{-2u}})du \quad (2.62)$$

which holds for any A. A change of variable $v = xe^{-2\lambda(A)u}$in the lower bound and $v = xe^{-2u}$ in the upper bound, gives the following bound

$$A\left(1 - e^{-\frac{x}{\cosh^2 A}}\right) + \int_0^{xe^{-2\lambda(A)}} \frac{1}{2\lambda(A)v}(1 - e^{-4v})dv \leq 2F(x) \leq \int_0^x \frac{1}{2v}(1 - e^{-4v})dv \quad (2.63)$$

Now suppose $x > 1$, then the following identity holds

$$\int_0^x \frac{1}{v}(1-e^{-4v})dv = \int_0^1 \frac{1}{v}(1-e^{-4v})dv + \log x - \int_1^x e^{-4v}\frac{dv}{v} \tag{2.64}$$

As $x \to \infty$, the last term converges to $-Ei(-4)$ which can be computed numerically. The first term can also be evaluated numerically and so as $x \to \infty$, we get

$$\int_0^x \frac{1}{v}(1-e^{-4v})dv = \log x + C$$

where $C \approx 1.9635$. Applying this identity to the bounds, we get

$$\frac{1}{2\lambda(A)}(\log x + C) - Ae^{-\frac{x}{\cosh^2 A}} \leq 2F(x) \leq \frac{1}{2}\log x + \frac{C}{2} \tag{2.65}$$

Since A is arbitrary, it can be chosen to grow with x. If we take $A = \frac{1}{2}(1+\epsilon)\log x$, then $\lambda(A) \to 1$ and the second term goes to 0. So, asymptotically as $x \to \infty$,we have

$$F(X) \to \frac{1}{4}\log x + D \tag{2.66}$$

where $D = \frac{C}{4} \approx 0.49088$.

Combining all the asymptotic behaviors, we get the behavior of Q_p,

$$Q_p(x) \to \begin{cases} \gamma\sqrt{2x} & x \to 0^+ \\ \frac{1}{4}\log x + D & x \to \infty \end{cases}$$

here we've used the fact that $\beta_1 \approx \beta_2$.

2.5 Sterling Ratio

A commonly used performance measure which uses the maximum drawdown is the Sterling Ratio

$$Sterling(T) = \frac{\mu}{E[MDD]} \tag{2.67}$$

It basically measures how much return to expect for every unit of drawdown risk. This is similar to the widely used Sharpe Ratio which uses the standard deviation as a measure of risk.

$$Shape(T) = \frac{\mu}{\sigma} \tag{2.68}$$

For the case $\mu > 0$, using equation 2.52 we have

$$Sterling(T) = \frac{(Sharpe(T))^2}{Q_p((Sharpe(T))^2 T)} \tag{2.69}$$

In the asymptotic case as $T \to \infty$, we have

$$Sterling(T \to \infty) = \frac{(Sharpe)^2}{-0.5416 + 0.5\log(T) + log(Sharpe)} \tag{2.70}$$

These indicate a direct relationship between the Sterling Ratio and the Sharpe Ratio.

2.6 Conclusion

The Maximum Drawdown is an important risk measure. For it to be an effective indicator of the risk, its analytical properties have to be studied. We've presented the distribution and the expected value of Maximum drawdown. We found that in the limiting case, the drawdown statistic scales in three different ways $(\log(T), \sqrt{T}, T)$ depending on the sign of μ. This can be used as a way of testing the hypothesis : is the return positive, zero or negative.

Chapter 3

AlphaBoost

In any learning scenario, the main goal is to find a hypothesis that performs well on the unseen examples. In recent years, there has been a growing interest in voting algorithms, which combine the output of many hypothesis to produce a final output. These algorithms take a given "base" learning algorithm and apply it repeatedly to re-weighted versions of the original dataset, thus producing an ensemble of hypothesis which are then combined via a weighted voting scheme to form a final aggregate hypothesis.

It is believed that the reason for their good performance is that they tend to produce hypothesis with large margins, however it has been seen that directly maximizing the margin doesn't always lead to better performance. Another explanation is that they really are a gradient descent procedure in a functional space. In this chapter, we present an algorithm which aggressively minimizes the cost functional to see if that really leads to a better out-of-sample performance.

3.1 Notation

We assume that the examples (x, y) are randomly generated from some unknown probability distribution $\mathcal{D}$ on $X \times Y$ where X is the input space and Y is the output space. We'll only be dealing with binary classifiers, so in general we'll have $Y = \{-1, 1\}$. Boosting algorithms produce a voted combination of classifiers of the form $sgn(F(x))$ where

$$F(x) = \sum_{t=1}^{T} \alpha_t f_t(x)$$

where $f_t : X \rightarrow \{-1, 1\}$ are base classifiers from some fixed hypothesis class $\mathcal{F}$ and $\alpha_t \in \Re^+$and $\sum_{t=1}^{T} \alpha_t = 1$ are the weights of the classifiers. The class of all convex combinations of functions from the base classifiers will be called $con(\mathcal{F})$.

The margin(Δ) of an example (x, y) for the classifier $sgn(F(x))$ is defined as $yF(x)$. Margin is a measure of the confidence on the decision. A large positive margin implies a confident correct decision.

Given a set $S = \{(x_1, y_1), ..., (x_n, y_n)\}$ of examples drawn from $\mathcal{D}$, the goal of learning is to construct an hypothesis, in our case a voted combination of classifier so that it minimizes the out-of-sample error which is defined as $P_{\mathcal{D}}[sgn(F(x)) \neq y]$, i.e. the probability that F wrongly classifies a random point drawn from the distribution $\mathcal{D}$.

3.2 AdaBoost

AdaBoost is one of the most popular boosting algorithms. It has been shown to work very well in a number of real world situations[17, 18, 16, 9, 6]. It takes a base learning algorithm and repeatedly applies it to re-weighted versions of the original training sample, producing a linear combination of hypothesis from the base learner. At each iteration, AdaBoost emphasizes the misclassified examples from the previous iteration thereby forcing the weak learner to focus on the "difficult" examples. Algorithm 1 gives the pseudo-code of AdaBoost.

The effectiveness of AdaBoost has been attributed to the fact that it tends to produce classifiers with large margins on the training points. Theorem 1 bounds the generalization error of an voted classifier in terms of the fraction of points with small margin.

Theorem 1 *Let S be a sample of N examples chosen independently at random according to $\mathcal{D}$. Assume that the base hypothesis space $\mathcal{F}$ has VC-dimension d, and let*

Algorithm 1 AdaBoost(S,T)[7]

```
• Input: $S = (x_1, y_1), ..., (x_N, y_N)$
• Input: $T$ the number of iterations
• Initialize $w_i = \frac{1}{N}$ for $i = 1, .., N$
• For $t = 1$ to $T$ do
    – Train the weak learner on the weighted dataset $(S, w)$ and obtain $h_{t:} : X \to \{-1, 1\}$
    – Calculate the weighted training error $\epsilon_t$ of $h_t$:
          $$\epsilon_t = \sum_{i=1}^{N} w_i I[h_t(x_i) \neq y_i]$$
    – Calculate the weight $\alpha_t$ as:
          $$\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}$$
    – Update weights
          $$w_i^{new} = w_i \exp\{-\alpha_t y_n h_t(x_n)\}/Z_t$$
      where $Z_t$ is a normalization constant
    – if $\epsilon_t = 0$ or $\epsilon_t \geq \frac{1}{2}$ then break and set $T = t - 1$
• Output $F_T(x) = \sum_{t=0}^{T} \alpha_t h_t(x)$
```

$\delta > 0$. *Then with probability at least* $1 - \delta$ *over the random choice of the training set* S*,every weighted function* $F \in lin(\mathcal{F})$ *satisfies the following bound for all* $\gamma > 0$

$$P_D[sgn(F(x)) \neq y] \quad \leq \quad P_S[yF(x) \leq \gamma] + O\left(\sqrt{\frac{\gamma^{-2} d \log(N/d) + \log(1/\delta)}{N}}\right)$$

AdaBoost has been found to be particularly effective in increasing the margin of "difficult" examples (those with small margin), even at the price of reducing the margin of other examples. So, it seems that the effectiveness of AdaBoost comes from maximizing the minimum margin. Grove and Schuurmans [8] devised an algorithm

LPBoost which maximizes the minimum margin, and achieve better minimum margin than AdaBoost. However, this lead to a worse out-of-sample performance. They concluded that no simple version of the minimum margin explanation can be complete.

3.3 AnyBoost: Boosting as Gradient Descent

Mason et al.[12] presents a generalized view of boosting algorithms as a gradient descent procedure in the functional space. Their algorithm AnyBoost iteratively minimizes the cost function by gradient descent in the functional space.

The base hypothesis and their linear combinations can be viewed as elements of an inner product space $(\mathcal{X}, \langle , \rangle)$ where $\mathcal{X}$ is a linear space of functions that contain $lin(\mathcal{F})$. The algorithm AnyBoost starts with the zero function F and iteratively finds a function $f \in \mathcal{F}$ to add to F so as to minimize the cost $C(F + \epsilon f)$ for some small ϵ. The new function added f is chosen such that the cost function is maximally decreased. The desired "direction" is the negative of the functional derivative of C at F, $-\nabla C(F)$, where

$$\nabla C(F)(x) := \frac{\partial C(F + \delta 1_x)}{\partial \delta}|_{\delta=0}$$

where 1_x is the indicator function of x. In general it is not possible to choose the new function as the negative of the gradient since we're restricted to picking from $\mathcal{F}$, so instead AnyBoost searches for f which maximizes the inner product $\langle f, -\nabla C(F) \rangle$

Most of the boosting algorithms use a cost function of the margin of points.

$$C(F) = \frac{1}{N} \sum_{i=1}^{N} c(y_i F(x_i))$$

where $c : \Re \rightarrow \Re^+$is a monotonically nondecreasing function. In this case, the inner product can be defined as

$$\langle f, g \rangle = \frac{1}{N} \sum_{i=1}^{N} f(x_i) g(x_i) \tag{3.1}$$

Algorithm 2 AnyBoost(C, S, T)[12]

```
Requires:

• An inner product space $(\mathcal{X}, \langle , \rangle)$containing functions mapping $X$ to $Y$
• A class of base classifier $\mathcal{F}$
• A differentiable cost functional $C : lin(\mathcal{F}) \rightarrow \Re$
• A weak learner $\mathcal{L}(F)$ that accepts $F \in lin(\mathcal{F})$ and returns $f \in \mathcal{F}$ with a large
  value of $-\langle \nabla C(F), f \rangle$
• Input: $S = (x_1, y_1), ..., (x_N, y_N)$
• Input: $T$ is the number of iterations
    – Let $F_0(x) := 0$
    – for $t = 0$ to $T$ do
        ∗ Let $f_{t+1} := \mathcal{L}(F_t)$
        ∗ if $-\langle f, -\nabla C(F) \rangle \leq 0$
            · return $F_t$
        ∗ Choose $\alpha_{t+1}$
        ∗ Let $F_{t+1} = F_t + \alpha_{t+1} f_{t+1}$
    – return $F_{T+1}$
```

So,

$$\langle -\nabla C(F), f \rangle = \frac{1}{N^2} \sum_{i=1}^{N} y_i F(x_i) c'(y_i F(x_i)) \tag{3.2}$$

So, maximizing $\langle -\nabla C(F), f \rangle$ is equivalent to minimizing the training error with examples weights, $D(i) \propto -c'(y_i F(x_i))$

AdaBoost can be seen as a special case of AnyBoost with the cost function $c(yF(x)) = e^{-yF(x)}$and the inner product $\langle F(x), G(x) \rangle = \frac{1}{N} \sum_{i=1}^{N} F(x_i) G(x_i)$. Many of the most successful voting methods are special cases of AnyBoost with the appropriate cost function and step size.

Table 3.1: Voting Methods seen as special case of AnyBoost

Algorithm	Cost Function	Step Size
AdaBoost	$e^{-yF(x)}$	Line Search
ARC-X4	$(1 - yF(x))^5$	$1/t$
LogitBoost	$ln(1 + e^{-yF(x)})$	Newton-Rapson

3.4 AlphaBoost

AlphaBoost improves on AnyBoost by minimizing the cost function more aggressively. It can achieve significantly lower values of cost function, much faster than AnyBoost.

3.4.1 Algorithm

AlphaBoost starts out by calling AnyBoost to obtain a linear combination of hypothesis from the base learner. It then optimizes the weights given to each of the classifier in the combination to further reduce the cost function. This is done by doing a conjugate gradient descent [13] in the weight space. Algorithm 3 gives the pseudo code of AlphaBoost. We used conjugate gradient instead of normal gradient descent because it uses the second order information of the cost function and so the cost is reduced faster.

The first step is to visualize the cost function as a function of the weight of the hypothesis instead of the aggregate hypothesis. So once we've fixed the weak learners which will vote to form the aggregate hypothesis, we've a cost function $C(\alpha)$ which depends on the weight which each hypothesis gets in the aggregate. We can then do a conjugate gradient descent in the weight space to minimize the cost function and thus find the optimal weights.

Suppose AnyBoost returns $F_T(x){=}\sum_{i=1}^{T} \alpha_i h_i(x)$ as the final hypothesis. Then we have,

$$C(\alpha) = \sum_{i=1}^{n} c(y_i F_T(x_i)) \tag{3.3}$$

and so, the gradient can be computed as

$$\frac{\partial C(\alpha)}{\partial \alpha_t} = \sum_{i=1}^{n} y_i h_t(x_i) c'(y_i F_T(x_i)) \tag{3.4}$$

So the descent direction at stage t is computed as $d_t = -\nabla C(\alpha_t)$. Instead of using this direction directly, we choose the search direction as

$$d_t = -\nabla C(\alpha_t) + \beta_t d_{t-1} \tag{3.5}$$

where $\beta_t \in \Re$ and d_{t-1} is the last search direction. The value of β_t controls how much of the previous direction affects the current direction. It is computed using the Polak-Ribiere formula

$$\beta_t = \frac{\langle d_t, d_t - d_{t-1} \rangle}{\langle d_{t-1}, d_{t-1} \rangle} \tag{3.6}$$

Algorithm 3 uses a fixed step size to perform conjugate descent. We can also do a line search to find an optimal step size at each iteration.

Algorithm 3 AlphaBoost(C, S,T, A)

```
Requires:

• An inner product space $(\mathcal{X}, \langle , \rangle)$containing functions mapping $X$ to $Y$
• A class of base classifier $\mathcal{F}$
• Input: $S = (x_1, y_1), ..., (x_N, y_N)$
• Input : T the number of iterations of AnyBoost and $A$ is the number of conjugate gradient steps
• Input: $C$ is a differentiable cost functional $C : lin(\mathcal{F}) \rightarrow \Re$
    – Let $F_T(x)$=$\alpha.H(x)$ be the output of AnyBoost($C$,$S$,$T$)
    – $d_0 = 0$ and $\alpha_0 = \alpha$
    – For $t = 1$ to $A$
        ∗ $d_t = -\nabla C(\alpha_t)$
        ∗ $\beta_t = \frac{\langle d_t, d_t - d_{t-1} \rangle}{\langle d_{t-1}, d_{t-1} \rangle}$
        ∗ $d\alpha_t = -d_t + \beta_t d\alpha_{t-1}$
        ∗ $\alpha_t = \alpha_{t-1} + \eta d\alpha_{t-1}$
```

3.4.2 Generalization Performance

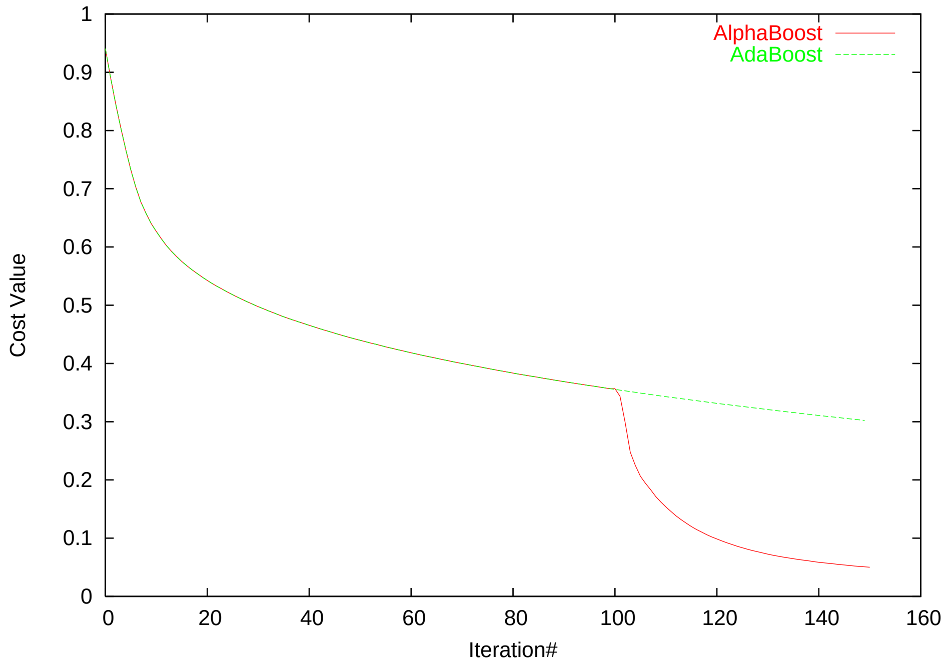
For analyzing the out-of-sample performance of AlphaBoost, we used AdaBoost's exponential cost function. Once we fix a cost function, AnyBoost/AdaBoost and AlphaBoost are essentially two algorithms which try to optimize the same cost function. For comparing the generalization performance improvement that AlphaBoost obtains by decreasing the cost function further, we generated random target function using the Caltech Data Engine [1].Targets were chosen at varying levels of complexity and from different models. For each target, 50 random training and testing datasets of size 500 and 5000 were generated. Decision stumps were used as the base learner in both the algorithms. For AlphaBoost, AdaBoost was first run for 100 iterations to get the initial hypothesis and then conjugate gradient descent with line search was done for 50 steps on $\alpha's$. This was compared with running AdaBoost for 150 iterations. All the results were averaged over 100 independent trials. The cost function and out-of-sample error at each iteration for one such run is shown in figure 3.1.

The cost function at the end of AlphaBoost is significantly lower than the cost function at the end of AdaBoost. However, the out-of-sample error achieved by AdaBoost is significantly lower. The results on other random functions are summarized in Appendix B.

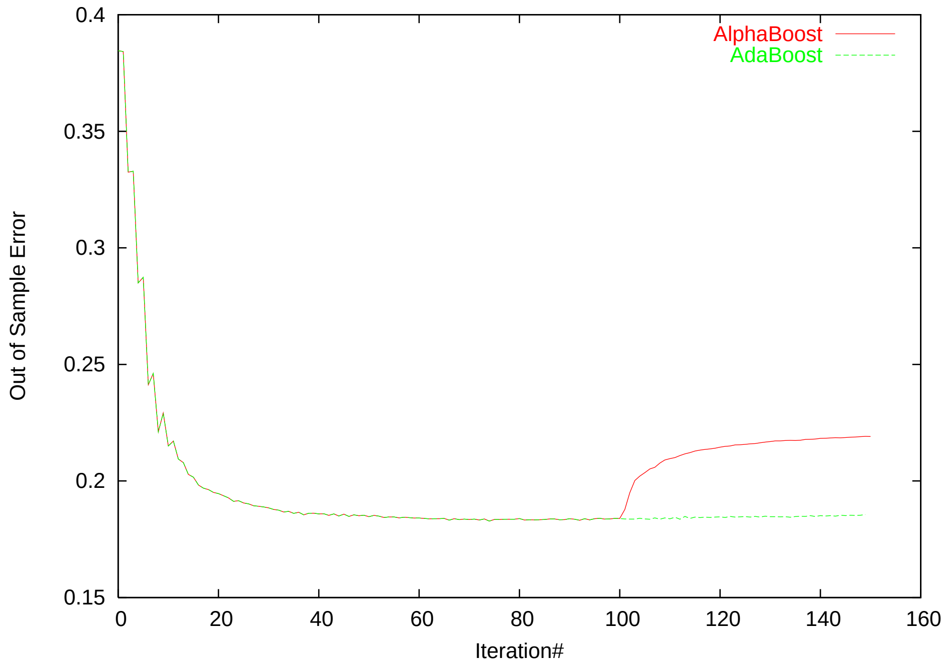
3.4.3 Experimental Results

We tested AlphaBoost on six datasets from the UCI machine learning repository[2]. The datasets chosen were Pima Indians Diabetes Database, Sonar database, heart disease diagnosis database from V.A. Medical Center, Long Beach and Cleveland Clinic Foundation collected by Robert Detrano, M.D., Ph.D., John Hopkins University Ionosphere database, 1984 United States Congressional Voting Records Database and breast cancer databases from the University of Wisconsin Hospitals [11]. Decision stumps were used as the base learner in all the experiments. For the experiments, the dataset was randomly divided into two sets of size 80% and 20% and they were used for training and testing the algorithms. AlphaBoost was composed of 100 steps of AdaBoost followed 50 steps of conjugate gradient with line search as in the previous section and it was compared with AdaBoost running for 150 iterations. All the results were averaged over 50 runs.

Table 3.2 shows the final values obtained by the two algorithms. As expected, AlphaBoost was able to achieve significantly lower value of the cost function. Though AdaBoost achieved better out-of-sample error than AlphaBoost, the error bars are high in these limited datasets to make any statistically significant conclusions. Figures 3.2-3.7 show the cost function value and out-of-sample error as a function of the number of iterations. For the first 100 iterations, AdaBoost and AlphaBoost perform exactly the same thing, so the curves coincide for this period. The cost function takes a steep dip around iteration 101 for AlphaBoost when it starts doing conjugate gradient descent. The out-of-sample error increases during this stage. Also shown are the distribution of the margins at the end of 100 iterations of AdaBoost and at the end



(a) Cost function Value at each iteration

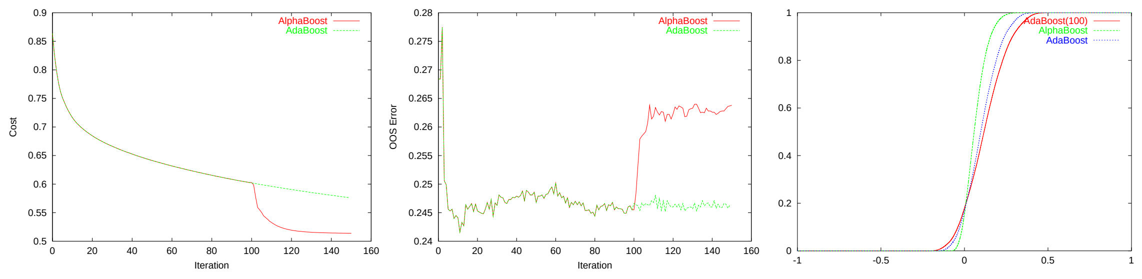


(b) Out of sample error at each iteration

Figure 3.1: Performance of AlphaBoost and AdaBoost on a random target

Table 3.2: Experimental Results on UCI Data Sets

Data Set	Algorithm	Cost	OOS Error	min(Δ)	$\overline{\Delta}$
Pima Indian	AlphaBoost	**0.5139**	0.2638	**−0.0979**	0.0658
	AdaBoost	0.5763	**0.2463**	−0.1576	**0.1023**
Sonar	AlphaBoost	**0**	0.1833	**0.1099**	**0.2128**
	AdaBoost	0.0003	**0.167**	0.1088	0.2099
Cleveland	AlphaBoost	**0.0386**	0.2421	**−0.183**	0.0979
	AdaBoost	0.0651	**0.2047**	−0.0747	**0.1366**
Ionosphere	AlphaBoost	**0**	0.1	**0.0652**	0.1849
	AdaBoost	0.0094	**0.0894**	0.0459	**0.1886**
Vote	AlphaBoost	**0.3576**	0.2222	**−0.1359**	0.2724
	AdaBoost	0.3756	**0.2155**	−0.1757	**0.3104**
Cancer	AlphaBoost	**0.0015**	0.0483	**−0.0052**	0.2377
	AdaBoost	0.0509	**0.0419**	−0.0511	**0.2638**



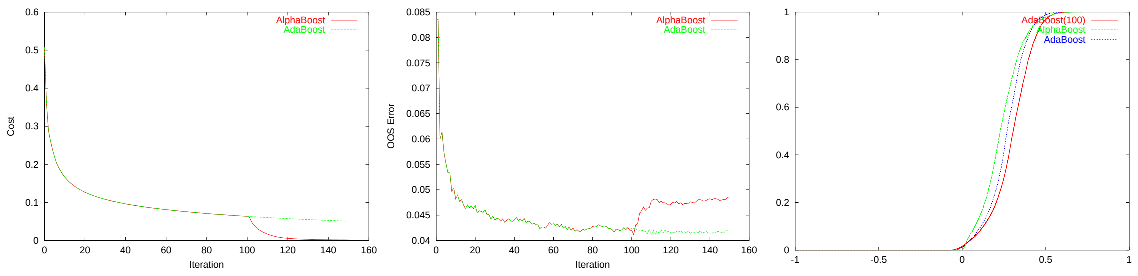
(a) Cost Function (b) Out of Sample Error (c) Margin CDF

Figure 3.2: Pima Indian Diabetes

of 150 iterations of AdaBoost and 50 iterations of conjugate gradient descent. The distribution of the margins at the end of 100 iterations of AdaBoost is the starting point for both the algorithms and from there on, they do different things for the next 50 iterations. AlphaBoost tends to maximize the minimum margin better than AdaBoost at the expense of lowering the maximum margin.

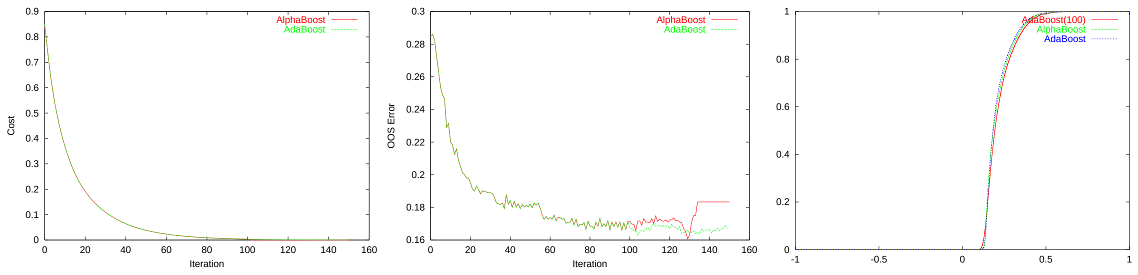
3.5 Conclusion

We've presented an improved algorithm for optimizing the cost function used in many boosting algorithms. AlphaBoost was able to achieve significantly lower values of



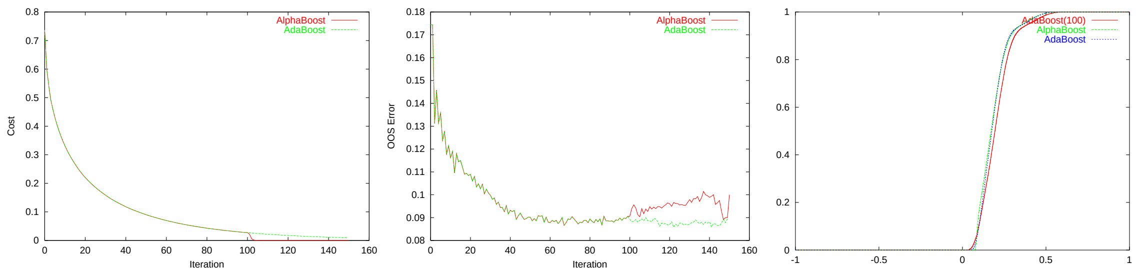
(a) Cost Function (b) Out of Sample Error (c) Margin CDF

Figure 3.3: Wisconsin Breast Cancer



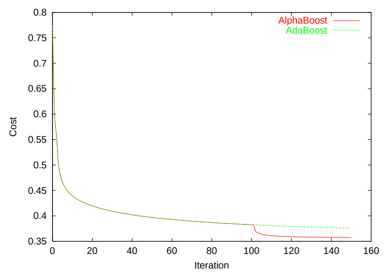
(a) Cost Function (b) Out of Sample Error (c) Margin CDF

Figure 3.4: Sonar

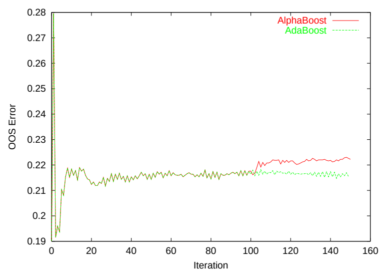


(a) Cost Function (b) Out of Sample Error (c) Margin CDF

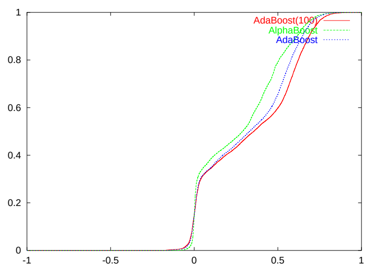
Figure 3.5: Ionosphere



(a) Cost Function

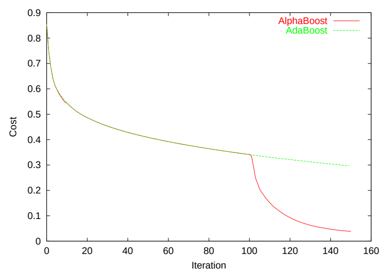


(b) Out of Sample Error

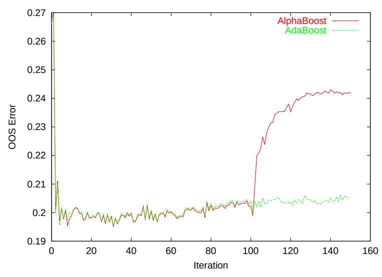


(c) Margin CDF

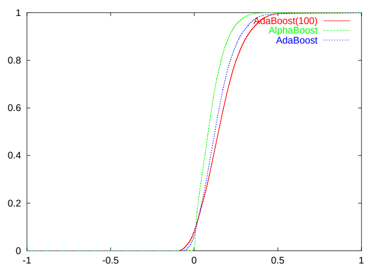
Figure 3.6: Vote84 Dataset



(a) Cost Function



(b) Out of Sample Error



(c) Margin CDF

Figure 3.7: Cleveland Heart Dataset

cost function. However, in terms of out-of-sample performance, AdaBoost was still the better algorithm. This shows that aggressive optimization of the exponential cost function can lead to overfitting.

Chapter 4

Conclusions

We've addressed two problems, one in the field of computational finance and another in the field of machine learning. In Chapter 2, we addressed the problem of estimating the distribution of the maximum drawdown of a Brownian motion. We also computed the expected value of the maximum drawdown and determined its limiting behavior.

In Chapter 3, we presented a new boosting algorithm AlphaBoost which is an extension of the popular AdaBoost algorithm, and does a better job of optimizing the exponential cost function. We tested the new algorithm using Caltech Data Engine and datasets from UCI machine learning repository, and found that it achieved lower values of cost function. We also found that the exponential cost function used in AdaBoost is vulnerable to overfitting, so AlphaBoost achieved worse out-of-sample performance than AdaBoost.

Appendix A

Table of Numerical values for Q(.)

x	$Q_p(x), \mu > 0$
$x \to 0$	$\gamma\sqrt{2x}$
0.0005	0.019690
0.0010	0.027694
0.0020	0.038896
0.0050	0.060721
0.0100	0.084693
0.0200	0.117503
0.0300	0.141842
0.0400	0.161817
0.0500	0.179015
0.0600	0.194248
0.0700	0.207999
0.0800	0.220581
0.0900	0.232212
0.1000	0.243050
0.2000	0.325071
0.3000	0.382016
0.4000	0.426452
0.5000	0.463159
1.5000	0.668992
2.5000	0.775976
3.5000	0.849298
4.5000	0.905305
10.000	1.088998
20.000	1.253794
30.000	1.351794
40.000	1.421860
50.000	1.476457
150.0000	1.747485
250.0000	1.874323
$x \to \infty$	$\frac{1}{4}\log x + 0.49088$

x	$Q_n(x), \mu < 0$
$x \to 0$	$\gamma\sqrt{2x}$
0.0005	0.019965
0.0010	0.028394
0.0015	0.034874
0.0020	0.040369
0.0025	0.045256
0.0050	0.064633
0.0075	0.079746
0.0100	0.092708
0.0150	0.114814
0.0200	0.133772
0.0300	0.166229
0.0400	0.194489
0.0500	0.220056
0.0600	0.243374
0.0700	0.265472
0.0800	0.286406
0.0900	0.306393
0.0950	0.316066
0.1000	0.325585
0.1500	0.413136
0.2000	0.491599
0.2500	0.564333
0.3000	0.633007
0.3500	0.698849
0.4000	0.762455
0.5000	0.884593
1.0000	1.445520
2.0000	2.483960
5.0000	5.498820
$x \to \infty$	$x + \frac{1}{2}$

Appendix B

Generalization Performance of AlphaBoost

To compare the generalization performance of AlphaBoost, we used the Caltech Data Engine to generate random target functions. The Caltech Data Engine is a computer program that contains several predefined data models, such as neural networks, support vector machines (SVM), and radial basis functions (RBF). When requested for data, it randomly picks a model, generates (also randomly) parameters for that model, and produces random examples according to the generated model. A complexity factor can be specified which controls the complexity of the generated model. The engine can be prompted repeatedly to generate independent data sets from the same model to achieve small error bars in testing and comparing learning algorithms.

The two algorithms were compared using function of varying complexity and from different models. For each target, 100 independent training sets of size 500 were generated. The algorithms were tested on a independently generated test set of size 5000. In all the runs, AlphaBoost obtained significantly lower values of cost function. However, the out-of-sample performance of AdaBoost was significantly better. The final cost and out-of-sample error obtained by the two algorithms on different models of Data Engine are given in Tables B.1 to B.3. For each model, results for 10 different target functions are shown. The error bar on all the runs was less than 10^{-3} of the values, and so are not reported in the tables.

Table B.1: Generalization Performance on NNet Model

	AlphaBoost Cost	AlphaBoost OOS error	AdaBoost Cost	AdaBoost OOS error
1	$9.3565888e-14$	$3.0266667e-02$	$2.7597247e-03$	$2.9089333e-02$
2	$1.2035998e-13$	$4.4191011e-02$	$1.8804671e-02$	$4.2529333e-02$
3	$1.6344770e-14$	$4.4392500e-02$	$2.0653736e-02$	$4.3245333e-02$
4	$4.2481317e-17$	$4.1800000e-02$	$1.7477263e-02$	$4.0553333e-02$
5	$4.2698023e-18$	$3.8835088e-02$	$1.0045108e-02$	$3.8081333e-02$
6	$4.3135297e-14$	$3.2375000e-02$	$6.5214255e-03$	$3.2469333e-02$
7	$1.0915530e-15$	$4.6779747e-02$	$2.1924053e-02$	$4.4280000e-02$
8	$1.0915530e-15$	$4.5810959e-02$	$2.0753478e-02$	$4.4768000e-02$
9	$3.8035720e-18$	$2.8975000e-02$	$5.0236339e-03$	$2.8206667e-02$
10	$4.7306306e-15$	$4.5177778e-02$	$1.7291558e-02$	$4.3353333e-02$

Table B.2: Generalization Performance on SVM Model

	AlphaBoost Cost	AlphaBoost OOS error	AdaBoost Cost	AdaBoost OOS error
1	$1.5322601e-04$	$8.2038202e-02$	$7.6928569e-02$	$7.7200000e-02$
2	$1.3788141e-13$	$4.2930612e-02$	$1.9828926e-02$	$4.1520000e-02$
3	$4.8592162e-15$	$4.9562963e-02$	$2.8406560e-02$	$4.8784000e-02$
4	$5.3670424e-04$	$9.5224490e-02$	$9.5785479e-02$	$8.6446000e-02$
5	$5.0031784e-05$	$7.3665116e-02$	$7.2252861e-02$	$6.8764000e-02$
6	$4.8371905e-10$	$6.3383721e-02$	$4.4560837e-02$	$5.9480000e-02$
7	$1.0375764e-04$	$6.6155056e-02$	$4.4763318e-02$	$6.2090000e-02$
8	$1.0375764e-04$	$7.4686420e-02$	$5.9636413e-02$	$6.7576000e-02$
9	$2.2833023e-11$	$5.8356962e-02$	$3.7523566e-02$	$5.4996000e-02$
10	$1.2634642e-03$	$9.4135556e-02$	$1.0156811e-01$	$8.3704000e-02$

Table B.3: Generalization Performance on SVM2 Model

	AlphaBoost Cost	AlphaBoost OOS error	AdaBoost Cost	AdaBoost OOS error
1	0.517253	0.3126293	0.62887978	0.263594
2	0.5031718	0.3200826	0.61548533	0.285921
3	0.5391776	0.3443026	0.6450533	0.314052
4	0.438985	0.2986720	0.552395	0.272044
5	0.7194488	0.455278	0.8077336	0.4283666
6	0.5417086	0.33590	0.645625	0.299938
7	0.7265202	0.2355866	0.8178936	0.452702
8	0.4332155	0.2620933	0.55019133	0.212585
9	0.555306	0.337778	0.66236691	0.2949906
10	0.65321	0.25642	0.76012	0.20311

Bibliography

[1] Pratap A. Caltech data engine. Technical report.

[2] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[3] Blasi. On a random walk between a reflecting and an absorbing barrier. *Annals of Probability*, 4(4):695–696, 1976.

[4] Greg C. Bond. Green's function for a zero drift reflected brownian motion and the expected maximum drawdown. 2003. Submitted to Baker Investment Group.

[5] Georgia Nakou David Harding and Ali Nejjar. The pros and cons of "drawdown" as a statistical measure of risk for investments. http://www.wintoncapital.com/pdfs/Drawdown.pdf.

[6] Gerard Escudero, Lluís Màrquez, and German Rigau. Boosting applied to word sense disambiguation. In Ramon López de Mántaras and Enric Plaza, editors, *Proceedings of ECML-00, 11th European Conference on Machine Learning*, pages 129–141, Barcelona, ES, 2000. Springer Verlag, Heidelberg, DE.

[7] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[8] Adam J. Grove and Dale Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *AAAI/IAAI*, pages 692–699, 1998.

[9] G. Guo and H. Zhang. Boosting for fast face recognition, 2001.

[10] Amrit Pratap Malik Magdon-Ismail, Amir F. Atiya and Yaser S. Abu-Mostafa. The sharpe ratio, range and maximal drawdown of a brownian motion. Technical Report TR 02-13, RPI Computer Science, September 2002.

[11] O. L. Mangasarian and W. H. Wolberg. Cancer diagnosis via linear programming. *SIAM News*, 23(5), September 1990.

[12] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent, 2000.

[13] Fletcher R. and Reeves C.V. Function minimization by conjugate gradients. *Computer Journal*, pages 149–154, 1964.

[14] Robert E. Schapire. A brief introduction to boosting. In *IJCAI*, pages 1401–1406, 1999.

[15] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. 14th International Conference on Machine Learning*, pages 322–330. Morgan Kaufmann, 1997.

[16] Robert E. Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[17] H. Schwenk. Using boosting to improve a hybrid hmm/neural network speech recogniser, 1999.

[18] Holger Schwenk and Yoshua Bengio. Adaboosting neural networks: Application to on-line character recognition. In *ICANN*, pages 967–972, 1997.

[19] B. Weesakul. The random walk between a reflecting and an absorbing barrier. *Annals of Mathematical Statistics*, 32(3):765–769, 1961.