Heleen Otten

A theoretical analysis of boosting algorithms

Bachelor's thesis

Supervisor: Dr. Tim van Erven

Date bachelor exam: June 19, 2016

Mathematical Institute, University of Leiden

1 Abstract

Boosting is an important concept in machine learning to create classification algorithms. AdaBoost and NH-Boost.DT are two existing boosting algorithms, which both use a different online allocation algorithm as subroutine. However, there is a third online allocation algorithm that has not been used for boosting yet, named Squint.

In this thesis we have created a new boosting algorithm, SquintBoost, that uses Squint as online allocation algorithm. The advantage of Squint over the online allocation algorithms that are used for AdaBoost and NH-Boost.DT is that it has a better regret bound. By zooming in on the training error, we prove that this advantage also gives a lower upper bound for the training error of SquintBoost.

Contents

2 Introduction

2.1 Motivation

The main goal of this thesis is to see whether a better boosting algorithm can be created by using Squint as online allocation algorithm. AdaBoost and NH-Boost.DT are two boosting algorithms that use respectively Hedge and Normal-Hedge.DT as online allocation algorithm. We are going to prove what the upper bound of the training error for the new boosting algorithm is and compare this to the upper bounds of the existing algorithms. In Vente [7] these algorithms are tested in experiments to be able to compare their performance in practice.

Before we will zoom in on the existing algorithms, we discuss what the use is of boosting for classification problems and how it can be used. In Chapter 3 the technical set up for boosting is discussed and in Chapter 4 the theorems about the upper bounds of the training error of AdaBoost and NH-Boost.DT will be proven according to the proofs given in Freund and Schapire [3] and Luo and Schapire [6]. Then, in Chapter 5, the new boosting algorithm SquintBoost is introduced and it is proven what the upper bound for the training error of this new algorithm is.

2.2 Classification

The boosting algorithms that are discussed in this thesis are meant to solve classification problems. This means that they identify for an input vector to which of a set of categories $\mathcal{Y}$ it belongs. Given a training set containing a set of observations with corresponding output, the algorithm learns to classify input correctly. The training set is of the following form:

$$\text{For } i \in \{1, \dots, N\} : \begin{pmatrix} y_i \\ \\ \boldsymbol{x_i} \end{pmatrix} \text{ with } \boldsymbol{x_i} \in \mathbb{R}^d,\ d \in \mathbb{N} \text{ and } y_i \in \mathcal{Y}.$$

The vector $\boldsymbol{x}_i$ consists of d properties and y_i is the desired output. A classification algorithm is used to predict what the output will be given any new input vector $\boldsymbol{x}$ that is not in the training set.

Example 2.1. Handwritten digit recognition (see also Hastie et al. [4]).

Consider a set letters with handwritten zip codes. Now the algorithm is meant to decide which digits are written on the basis of the given pixels. This is an example of a classification problem. In this example y_i would be the i-th number that is used for training and $\boldsymbol{x_i}$ would consist of all the characteristics of the pixels of this number. Some digits are easier than others, since for example the 8 does not really look like any other digit, while the 1 and 7 quite look alike in certain handwritings. If this classification can be done accurate enough, the resulting algorithm could be used as part of an automatic sorting procedure for these letters. Note that it is very important that the error is low for this algorithm, since it would be a problem if letters were misdirected. An option to achieve such a low error, is to classify certain digits which are hard to classify to an extra category that has to be sorted by hand afterwards.

In this thesis we will mainly address binary classification problems. For these problems the set of labels $\mathcal{Y}$ has only two elements. As discussed in Freund and Schapire [3], algorithms used for binary classification problems can be generalized to classification problems with n categories by splitting the problem in $\frac{1}{2}n(n-1)$ binary problems. Then the boosting is done separately on each of the binary problems.

2.3 Boosting

Boosting is a part of machine-learning which uses weak learners to create one strong learner. A weak learner is a classifier that performs only slightly better than random guessing. So for binary classification problems, the weak learner only has to be correct just a little more than half of the time. A good way to create such a weak learner is described in Hastie et al. [4]. They explain how decision stumps can be used to create a weak learner. A decision stump is a binary tree with a single split. So these decision stumps take only one property of the input into account and classify on the basis of this property. As long as the error of this decision stump is not equal to $\frac{1}{2}$ it is useful for boosting. Note that if the error is bigger than $\frac{1}{2}$, the weak learner just has to classify the other way around. The weak learner determines its hypothesis on the basis of weights that are assigned to each training example. The boosting algorithm uses these examples to extract the hard cases and assigns a higher weight to those cases than to the easy examples. By repeating this on the training data, the weights are updated and so a strong learner, a classifier with much higher accuracy, is created. For the updating of the weights, an online allocation algorithm is used. These algorithms will be discussed in the next paragraph.

AdaBoost, as described in Freund and Schapire [3], is a boosting algorithm that uses Hedge as online allocation algorithm to create a strong learner out of a weak learner. At first, all the training examples get the same weight and a weak learning algorithm produces a hypothesis on the basis of these examples, but then the algorithm determines which examples are harder than others. The hard examples get a higher weight according to the Hedge algorithm to reduce the error of the algorithm. After T rounds in which the weak learner has produced T hypotheses h_t for $t \in \{1, \ldots, T\}$, the final classification hypothesis is determined on the basis of all these T hypotheses.

NH-Boost.DT, as described in Luo and Schapire [6], is a boosting algorithm that is computationally faster than AdaBoost. This advantage is achieved by ignoring a large number of easy examples in each round. Since NH-Boost.DT sets multiple weights to zero, these examples do not have to be taken into account by the weak learner. As more rounds have been run, the examples with zero weights increase, so the algorithm gets faster each round. For this boosting algorithm, the online allocation algorithm NormalHedge.DT is used. As will be proven in Chapter 4, the training error of NH-Boost.DT has an upper bound which is comparable to AdaBoost. Since the algorithm is faster, the training error will thus decrease faster than the training error of AdaBoost.

2.4 Online allocation algorithm

Assume there are N strategies and T is the number of iterations. An online allocation algorithm is used to choose for every $t \in \{1, \ldots, T\}$ a distribution $\boldsymbol{p}_t$ over these N strategies such that the suffered loss is as small as possible. For an online allocation algorithm, the loss $\boldsymbol{l}_t$ is defined dependent on the "game" it is used for such that the goal of the algorithm is to minimize its cumulative loss. The loss can be interpreted as the prediction error. Since $\boldsymbol{p}_t$ is a distribution, we have $\sum_{i=1}^{N} p_{t,i} = 1$, where $p_{t,i} \geq 0$ is the amount allocated to strategy i. Now on iteration t the suffered loss is defined as $\boldsymbol{p}_t \cdot \boldsymbol{l}_t = \sum_{i=1}^{N} p_{t,i} l_{t,i}$.

The regret R_T gives the difference between the loss of the algorithm and the loss of the best strategy, so $R_T = \sum_{t=1}^{T} \boldsymbol{p}_t \cdot \boldsymbol{l}_t - \min_i \sum_{t=1}^{T} l_{t,i}$. So when this difference is small, the algorithm performs well.

Hedge is introduced in 1997 by Freund and Schapire [3]. It is an algorithm used for online allocation problems. Hedge is nowadays still widely used for multiple purposes. It updates the given weights such that the suffered loss is small. To do so, it calculates on every iteration the suffered loss and the weights of the strategies that have suffered much loss are relatively decreased with respect to the weights of the strategies that have suffered few loss. Twelve years after Hedge was introduced, Chaudhuri, Freund and Hsu have invented a new algorithm called NormalHedge [2] and in 2014 Luo and Shapire introduced NormalHedge.DT in [6]. With this last algorithm they created a new boosting algorithm, named NH-Boost.DT. NormalHedge.DT is comparable to Hedge but chooses the weights in a different manner. Its regret bound is comparable to that of Hedge too. Finally, Squint, as introduced in Koolen and van Erven [5], is proven to perform significantly better on easy data, since it has a better regret bound.

As shown in Freund and Schapire [3], for the regret of Hedge the following holds:

$$R_T = O\left(\sqrt{T \ln N}\right) \tag{1}$$

For NormalHedge.DT we consider the upper bound for the ϵ-regret. The ϵ-regret is defined as $R_T^{\epsilon} = \sum_{t=1}^{T} \boldsymbol{p}_t \cdot \boldsymbol{l}_t - \sum_{t=1}^{T} l_{t,i_\epsilon}$, where i_ε is the index of the action that is the $\lceil N\varepsilon \rceil$-th element of the list of actions sorted by their total losses after T rounds from smallest to largest. For the ϵ-regret of NormalHedge.DT we have:

$$R_T^{\epsilon} = O\left(\sqrt{T \ln\left(\tfrac{1}{\epsilon}\right) + T \ln(\ln T)}\right) \tag{2}$$

For Squint, we consider the regret with respect to a set of strategies $\mathcal{K}$, which are referred to as "experts" in Koolen and van Erven [5]. The regret is defined as $R_T^i = \sum_{t=1}^{T} \boldsymbol{p}_t \cdot \boldsymbol{l}_t - \sum_{t=1}^{T} l_{t,i}$ and the regret with respect to a set of strategies as $R_T^{\mathcal{K}} = \mathbb{E}_{D(i|\mathcal{K})}(R_T^i)$, with D the prior distribution on the strategies. For $\mathcal{K}$ the set of strategies with index smaller than or equal to i_ϵ in combination with a uniform prior, you get the ϵ-regret. So this regret $R_T^{\mathcal{K}}$ is even more general than the ϵ-regret. Denote by V_T^i the variance of the i-th strategy: $V_T^i = \sum_{t=1}^{T} (\boldsymbol{p}_t \cdot \boldsymbol{l}_t - l_{t,i})^2$. Now we define $V_T^{\mathcal{K}} = \mathbb{E}_{D(i|\mathcal{K})}(V_T^i)$. For Squint the regret with respect to a set of strategies is of the following order:

$$R_T^{\mathcal{K}} = O\left(\sqrt{V_T^{\mathcal{K}} \ln\left(\frac{\ln T}{D(\mathcal{K})}\right)} + \ln\left(\frac{\ln T}{D(\mathcal{K})}\right)\right) \tag{3}$$

As is explained in Koolen and van Erven [5], the variance $V_T^{\mathcal{K}}$ can be much smaller than T and can not be larger than T, which implies that the upper bound for Squint is smaller than the upper bounds of the other algorithms. Because of that we are going to try to create a boosting algorithm with Squint and evaluate whether this advantage of Squint can be an advantage for boosting too.

Firstly, we have to find a way to convert Squint into a boosting algorithm. Freund and Schapire [3] do not mention how a boosting algorithm can be created in general, since AdaBoost has immediately incorporated Hedge in it. Secondly, we are going to prove what the upper bound will be for the new algorithm that is created with Squint. To do this, we first take a closer look at the upper bounds which were found for AdaBoost and NH-Boost.DT.

3 The boosting set up

A boosting algorithm is used for classification problems. Let d be the number of properties taken into account and let $\mathcal{Y}$ be the set of labels. As mentioned before, the algorithm needs training examples as input. So it needs N labeled examples, where the training examples are of the following form:

$$\text{For } i \in \{1, \ldots, N\} : \begin{pmatrix} y_i \\ \boldsymbol{x_i} \end{pmatrix} \text{ with } \boldsymbol{x_i} \in \mathbb{R}^d \text{ and some } d \in \mathbb{N} \text{ and } y_i \in \mathcal{Y}.$$

The vector $\boldsymbol{x}_i$ consists for every example of d properties and y_i is the desired output. Moreover, the algorithm needs an integer T which denotes the number of iterations. Now $w_{t,i}$ is the weight assigned to example i on iteration t. For the first weights $\boldsymbol{w}_1$, distribution D is used. So $w_{1,i} = D(i)$ and since D is a distribution, it follows that $\sum_{i=1}^{N} w_{1,i} = 1$.

Algorithm 1 Hedge(β)

Require:

$\beta \in [0, 1]$

initial weight vector $\boldsymbol{w}_1 \in [0, 1]^N$ with $\sum_{i=1}^{N} w_{1,i} = 1$

integer T specifying number of iterations

1: **for** $t = 1, 2, \ldots, T$ **do**

2: Choose allocation $\boldsymbol{p}_t = \frac{\boldsymbol{w}_t}{\sum_{i=1}^{N} w_{t,i}}$

3: Receive loss vector $\boldsymbol{l}_t \in [0, 1]^N$ from environment

4: Suffer loss $\boldsymbol{p}_t \cdot \boldsymbol{l}_t$.

5: Set the new weights vector to be

$$w_{t+1,i} = w_{t,i} \cdot \beta^{l_{t,i}}$$

6: **end for**

For AdaBoost, the online allocation algorithm Hedge is used for updating the weights on every iteration. The pseudo-code for Hedge, as given in [3], is shown in Fig. 1. This algorithm needs $\beta \in [0, 1]$ as input and updates the weights on

the basis of the loss vector. On every iteration of AdaBoost, this β is calculated dependent on the error ε_t of the hypothesis for iteration t.

With the weights, the distribution $\boldsymbol{p}_t$ is set to $\boldsymbol{p}_t = \frac{\boldsymbol{w}_t}{\sum_{i=1}^{N} w_{t,i}}$. The weak learning algorithm **WeakLearn** is provided with this distribution and generates a hypothesis $h_t : X \to [0,1]$. If we have $h_t(\boldsymbol{x}_i) \neq y_i$, the hypothesis makes a mistake. Now the loss is set to be $l_{t,i} := 1 - |h_t(\boldsymbol{x}_i) - y_i|$ and for every iteration the error is $\varepsilon_t = \sum_{i=1}^{N} p_{t,i} |h_t(\boldsymbol{x}_i) - y_i|$. Moreover, β_t is chosen to be $\beta_t = \varepsilon_t/(1-\varepsilon_t)$ and the weights are updated according to this β_t and loss $l_{t,i}$. After T iterations, the final hypothesis is determined on the basis of the T hypotheses h_t for $t \in \{1, \ldots, T\}$.

Thus, the AdaBoost algorithm is as shown in Fig. 2.

Algorithm 2 AdaBoost

Require:

sequence of N labeled examples $\langle (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N) \rangle$, $y_i \in \{0,1\}$
distribution D over the N examples
weak learning algorithm **WeakLearn**
integer T specifying number of iterations

1: **procedure Boosting**
2: **Initialize** the weight vector $w_{1,i} = D(i)$ for $i = 1, \ldots, N$.
3: **for** $t = 1, 2, \ldots, T$ **do**
4: Set $\boldsymbol{p}_t = \frac{\boldsymbol{w}_t}{\sum_{i=1}^{N} w_{t,i}}$.
5: Call **WeakLearn**, providing it with the distribution $\boldsymbol{p}_t$ and receive a hypothesis $h_t : X \to [0,1]$.
6: Calculate the error of h_t : $\varepsilon_t = \sum_{i=1}^{N} p_{t,i} |h_t(\boldsymbol{x}_i) - y_i|$.
7: Set $\beta_t = \varepsilon_t/(1-\varepsilon_t)$.
8: Update the weights:

$$w_{t+1,i} = w_{t,i} \cdot \beta_t^{1-|h_t(\boldsymbol{x}_i)-y_i|}$$

9: **end for**
10: **return** final hypothesis

$$h_f(\boldsymbol{x}) : \mathbb{R}^d \to \{0,1\}, h_f(\boldsymbol{x}) := \begin{cases} 1 & \text{if } \sum_{t=1}^{T} (\log \frac{1}{\beta_t}) h_t(\boldsymbol{x}) \geq \frac{1}{2} \sum_{t=1}^{T} \log \frac{1}{\beta_t} \\ 0 & \text{otherwise} \end{cases}$$

11: **end procedure**

For NH-Boost.DT, the hedging algorithm NormalHedge.DT is used, so the biggest difference between AdaBoost and NH-Boost.DT is how the weights are updated. Instead of multiplying the weights by a factor $\beta_t^{1-|h_t(\boldsymbol{x}_i)-y_i|}$ on every iteration, the weights are set proportional to $\exp\left(\frac{[s_{t-1,i}-1]_-^2}{3t} - 1\right) - \exp\left(\frac{[s_{t-1,i}+1]_-^2}{3t} - 1\right)$, where $s_{t-1,i}$ is determined according to the algorithm NormalHedge.DT. The notation $[s]_-$ stands for $\min\{0, s\}$. Moreover, the final hypothesis for NormalHedge.DT is just a majority vote of all the hypotheses h_t for $t \in \{1, \ldots, T\}$. Note that NH-Boost.DT uses label set $\mathcal{Y} = \{-1, 1\}$, while AdaBoost uses $\mathcal{Y} = \{0, 1\}$, since this makes in both cases the proof of the upper bound easier. The algorithm

NH-Boost.DT is thus as shown in Fig. 3.

Algorithm 3 NH-Boost.DT

Require:
sequence of N labeled examples $\langle(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\rangle$, $y_i \in \{-1, 1\}$
weak learning algorithm **WeakLearn**
integer T specifying number of iterations

procedure Boosting
  Set $s_0 = 0$.
  for $t = 1, 2, \ldots, T$ **do**
    Set $p_{t,i} \propto \exp([s_{t-1,i} - 1]_-^2/3t) - \exp([s_{t-1,i} + 1]_-^2/3t)$, for all i.
    Call **WeakLearn**, providing it with the distribution $\boldsymbol{p}_t$ and receive a hypothesis $h_t : X \to \{-1, 1\}$ with edge $\gamma_t = \frac{1}{2}\sum_{i=1}^{N} p_{t,i} y_i h_t(\boldsymbol{x}_i)$.
    Set $s_{t,i} = s_{t-1,i} + \frac{1}{2} y_i h_t(\boldsymbol{x}_i) - \gamma_t$ for all i.
  end for
return final hypothesis

$$h_f(\boldsymbol{x}) : \mathbb{R}^d \to \{-1, 1\}, h_f(\boldsymbol{x}) := \text{sign}\left(\sum_{t=1}^{T} h_t(\boldsymbol{x})\right)$$

end procedure

4 Analysis of the existing boosting algorithms

In [3], Freund and Shapire find an upper bound for the training error of the AdaBoost algorithm. First we are going to prove that this upper bound indeed holds for AdaBoost. Moreover, we zoom in on the proof of the upper bound of the training error of NH-Boost.DT. Then we can analyze how to find a way to create a boosting algorithm with Squint and find an upper bound for the training error of this new algorithm.

4.1 Upper bound of the error of AdaBoost

For the proof of the upper bound for the training error of AdaBoost, the following lemma is needed.

Lemma 4.1. *For every $\alpha \geq 0$ and $r \in [0, 1]$ the following holds:*

$$\alpha^r \leq 1 - (1 - \alpha)r \tag{4}$$

Proof. When taking the second derivative of the difference, the following is found.

$$\frac{d^2}{dr^2}\left(\alpha^r - 1 + (1 - \alpha)r\right) = \alpha^r \cdot \ln^2(r) \geq 0 \tag{5}$$

since $\alpha \geq 0$, so the difference is convex. $f(r) = \alpha^r$ and $g(r) = 1 - (1 - \alpha)r$ intersect for $r = 0$ and $r = 1$, since $f(0) = \alpha^0 = 1 = 1 - (1 - \alpha) \cdot 0 = g(0)$ and $f(1) = \alpha^1 = \alpha = 1 - (1 - \alpha) \cdot 1 = g(1)$. So the difference is 0 for $r = 0$

and $r = 1$ and is convex in between, so $\alpha^r \geq 1 - (1-\alpha)r$ for $r \in [0,1]$ or $\alpha^r \leq 1 - (1-\alpha)r$ for $r \in [0,1]$. We find that $\frac{d}{dr}\alpha^r|_{r=0} = \alpha^r \ln(r)|_{r=0} = -\infty$. Moreover, $\frac{d}{dr}(1-(1-\alpha)r)|_{r=0} = 1-\alpha$, so $\frac{d}{dr}\alpha^r|_{r=0} < \frac{d}{dr}1-(1-\alpha)r|_{r=0}$, so $\alpha^r \leq 1-(1-\alpha)r$ for $r \in [0,1]$. □

Now we can prove the following theorem, as proven in Freund and Schapire [3], about the upper bound for the training error of AdaBoost.

Theorem 4.2. *Let $\varepsilon_1, \ldots, \varepsilon_T$ be the errors of the generated hypotheses of the weak learning algorithm WeakLearn, when called by AdaBoost. Then the training error $\varepsilon = \sum_{i=1}^{N} D(i)\mathbb{1}\{h_f(\boldsymbol{x}_i) \neq y_i\}$ of the final hypothesis h_f output by AdaBoost is bounded above by*

$$\varepsilon \leq 2^T \prod_{t=1}^{T} \sqrt{\varepsilon_t(1-\varepsilon_t)} \tag{6}$$

Proof. Since $h_t(\boldsymbol{x}_i) \in [0,1]$ and $y_i \in \{0,1\}$ we have that $|h_t(\boldsymbol{x}_i) - y_i| \in [0,1]$, so $1 - |h_t(\boldsymbol{x}_i) - y_i| \in [0,1]$. Note that $\sum_{i=1}^{N} p_{t,i} = 1$ and $\sum_{i=1}^{N} p_{t,i}|h_t(\boldsymbol{x}_i) - y_i| = \varepsilon_t$ by definition. Moreover, $p_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^{N} w_{t,j}}$, so $w_{t,i} = p_{t,i} \cdot \sum_{j=1}^{N} w_{t,j}$. Since $\beta_t \geq 0$ by definition, Lemma 4.1 can be used and it follows that

$$\begin{aligned}
\sum_{i=1}^{N} w_{t+1,i} &= \sum_{i=1}^{N} w_{t,i}\beta_t^{1-|h_t(\boldsymbol{x}_i)-y_i|} \\
&\leq \sum_{i=1}^{N} w_{t,i}(1-(1-\beta_t)(1-|h_t(\boldsymbol{x}_i)-y_i|)) \\
&= \sum_{i=1}^{N} w_{t,i} - (1-\beta_t)\sum_{i=1}^{N} p_{t,i}\left(\sum_{j=1}^{N} w_{t,j}\right)(1-|h_t(\boldsymbol{x}_i)-y_i|) \\
&= \left(\sum_{i=1}^{N} w_{t,i}\right)(1-(1-\beta_t)(1-\varepsilon_t))
\end{aligned} \tag{7}$$

Note that $\beta_t \in [0,1]$ for all $t \in \{1,\ldots,T\}$ and $\varepsilon_t \in [0,1]$, so $1-(1-\beta_t)(1-\varepsilon_t) \geq 0$ for all t. By repeating this inequality, we get that

$$\begin{aligned}
\sum_{i=1}^{N} w_{T+1,i} &\leq \left(\sum_{i=1}^{N} w_{T,i}\right)(1-(1-\beta_T)(1-\varepsilon_t)) \\
&\leq \left(\sum_{i=1}^{N} w_{T-1,i}\right)(1-(1-\beta_{T-1})(1-\varepsilon_{T-1}))\,(1-(1-\beta_T)(1-\varepsilon_t)) \\
&\leq \cdots \leq \left(\sum_{i=1}^{N} w_{1,i}\right)\prod_{t=1}^{T}(1-(1-\beta_t)(1-\varepsilon_t)) \\
&= \prod_{t=1}^{T}(1-(1-\beta_t)(1-\varepsilon_t))
\end{aligned} \tag{8}$$

First suppose that $y_i = 0$. Then h_f makes a mistake on instant i if $h_f(\boldsymbol{x}_i) = 1$, so then, according to the AdaBoost algorithm, the following holds

$$\begin{aligned} &\sum_{t=1}^{T} \log(1/\beta_t) h_t(\boldsymbol{x}_i) \geq \frac{1}{2} \sum_{t=1}^{T} \log(1/\beta_t) \\ \Rightarrow \; &-\sum_{t=1}^{T} \log(\beta_t) h_t(\boldsymbol{x}_i) \geq -\frac{1}{2} \sum_{t=1}^{T} \log(\beta_t) \end{aligned} \tag{9}$$

Now we get, since $h_t(\boldsymbol{x}_i) \geq 0$, that

$$\prod_{t=1}^{T} \beta_t^{-|h_t(\boldsymbol{x}_i) - y_i|} = \prod_{t=1}^{T} \beta_t^{-|h_t(\boldsymbol{x}_i) - 0|} = e^{-\sum_{t=1}^{T} \log(\beta_t) h_t(\boldsymbol{x}_i)} \tag{10}$$

$$\geq e^{-\frac{1}{2}\sum_{t=1}^{T} \log(\beta_t)} = \prod_{t=1}^{T} e^{\log(\beta_t^{-\frac{1}{2}})} = \left(\prod_{t=1}^{T} \beta_t \right)^{-\frac{1}{2}} \tag{11}$$

Now suppose that $y_i = 1$. Then h_f makes a mistake on instant i if $h_f(\boldsymbol{x}_i) = 0$, so then, according to the AdaBoost algorithm, the following holds

$$\begin{aligned} &\sum_{t=1}^{T} \log(1/\beta_t) h_t(\boldsymbol{x}_i) < \frac{1}{2} \sum_{t=1}^{T} \log(1/\beta_t) \\ \Rightarrow \quad &\sum_{t=1}^{T} \log(\beta_t) h_t(\boldsymbol{x}_i) > \frac{1}{2} \sum_{t=1}^{T} \log(\beta_t) \end{aligned} \tag{12}$$

Now we get, since $h_t(\boldsymbol{x}_i) \in [0, 1]$ and thus $h_t(\boldsymbol{x}_i) - 1 \leq 0$, that

$$\begin{aligned} \prod_{t=1}^{T} \beta_t^{-|h_t(\boldsymbol{x}_i) - y_i|} &= \prod_{t=1}^{T} \beta_t^{-|h_t(\boldsymbol{x}_i) - 1|} = \prod_{t=1}^{T} \beta_t^{h_t(\boldsymbol{x}_i)} \cdot \prod_{s=1}^{T} \beta_s^{-1} \\ &= e^{\sum_{t=1}^{T} \log(\beta_t) \cdot h_t(\boldsymbol{x}_i)} \cdot \prod_{s=1}^{T} \beta_s^{-1} > e^{\frac{1}{2}\sum_{t=1}^{T} \log(\beta_t)} \cdot \prod_{s=1}^{T} \beta_s^{-1} \\ &= \prod_{t=1}^{T} e^{\log(\beta_t^{\frac{1}{2}})} \prod_{s=1}^{T} \beta_s^{-1} = \left(\prod_{t=1}^{T} \beta_t \right)^{-\frac{1}{2}} \end{aligned} \tag{13}$$

Since $y_i \in \{0, 1\}$, we have dealt with all the cases, so h_f only makes a mistake on instance i if

$$\prod_{t=1}^{T} \beta_t^{-|h_t(\boldsymbol{x}_i) - y_i|} \geq \left(\prod_{t=1}^{T} \beta_t \right)^{-\frac{1}{2}} \tag{14}$$

The fifth step of the algorithm gives us that

$$\begin{aligned} w_{T+1,i} &= w_{T,i} \beta_T^{1-|h_T(\boldsymbol{x}_i) - y_i|} = w_{T-1,i} \cdot \beta_{T-1}^{1-|h_{T-1}(\boldsymbol{x}_i) - y_i|} \cdot \beta_T^{1-|h_T(\boldsymbol{x}_i) - y_i|} \\ &= \cdots = w_{1,i} \cdot \prod_{t=1}^{T} \beta_t^{1-|h_t(\boldsymbol{x}_i) - y_i|} = D(i) \prod_{t=1}^{T} \beta_t^{1-|h_t(\boldsymbol{x}_i) - y_i|} \end{aligned} \tag{15}$$

Combining (14) and (15), we find, since $w_{T+1,i} \geq 0$ for all $i \in \{1, \ldots, N\}$, that

$$\begin{aligned}
\sum_{i=1}^{N} w_{T+1,i} &\geq \sum_{i:h_f(\boldsymbol{x}_i)\neq y_i} w_{T+1,i} \\
&= \sum_{i:h_f(\boldsymbol{x}_i)\neq y_i} D(i) \prod_{t=1}^{T} \beta_t^{1-|h_t(\boldsymbol{x}_i)-y_i|} \\
&\geq \sum_{i:h_f(\boldsymbol{x}_i)\neq y_i} D(i) \prod_{t=1}^{T} \beta_t \cdot \left(\prod_{t=1}^{T} \beta_t\right)^{-\frac{1}{2}} \\
&= \varepsilon \cdot \left(\prod_{t=1}^{T} \beta_t\right)^{\frac{1}{2}} \qquad (16)
\end{aligned}$$

So, it follows, since $\prod_{t=1}^{T} \beta_t \geq 0$, that

$$\begin{aligned}
\varepsilon &\leq \sum_{i=1}^{N} w_{T+1,i} \cdot \left(\prod_{t=1}^{T} \beta_t\right)^{-\frac{1}{2}} \\
&\leq \prod_{j=1}^{T} (1-(1-\varepsilon_j)(1-\beta_j)) \cdot \left(\prod_{t=1}^{T} \beta_t\right)^{-\frac{1}{2}} \\
&= \prod_{t=1}^{T} \frac{1-(1-\varepsilon_t)(1-\beta_t)}{\sqrt{\beta_t}} \qquad (17)
\end{aligned}$$

Now by calculating the derivative of this upper bound, we get

$$\frac{d}{d\beta_t}\left(\frac{1-(1-\varepsilon_t)(1-\beta_t)}{\sqrt{\beta_t}}\right) = -\tfrac{1}{2}\beta_t^{-\frac{3}{2}} \cdot (1-(1-\varepsilon_t)(1-\beta_t)) + \beta_t^{-\frac{1}{2}}(1-\varepsilon_t) \qquad (18)$$

To find out for which β the upper bound is the smallest, we set the derivative equal to zero. This gives us the following

$$\begin{aligned}
&\beta_t^{-\frac{1}{2}}(1-\varepsilon_t) = \tfrac{1}{2}\beta_t^{-\frac{3}{2}}(1-(1-\varepsilon_t)(1-\beta_t)) \\
&\Rightarrow \qquad \beta_t = \frac{\varepsilon_t}{1-\varepsilon_t} \qquad (19)
\end{aligned}$$

If this β_t is plugged into the upper bound, as done by AdaBoost, it follows that

$$\begin{aligned}
\varepsilon_t &\leq \prod_{t=1}^{T} \frac{1-(1-\varepsilon_t)(1-\frac{\varepsilon_t}{1-\varepsilon_t})}{\sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}}} \\
&= 2^T \prod_{t=1}^{T} \sqrt{(1-\varepsilon_t)\varepsilon_t} \qquad (20)
\end{aligned}$$

□

So the upper bound of AdaBoost only depends on T, the number of time steps, and ε_t for every $t \in \{1, 2, \ldots, T\}$. Since the error ε_t lies in the interval $[0, 1]$ for every $t \in \{1, 2, \ldots, T\}$, we find that $\varepsilon_t(1 - \varepsilon_t) \in [0, \frac{1}{4}]$ and thus $\sqrt{\varepsilon_t(1 - \varepsilon_t)} \in [0, \frac{1}{2}]$. So for every extra time step, the upper bound of the error is multiplied by $2\sqrt{\varepsilon_t(1 - \varepsilon_t)} \leq 1$. By increasing the number of time steps the upper bound will already decrease if the error of the hypothesis is only slightly smaller than $\frac{1}{2}$. Moreover, note that the upper bound of the error does not only depend on the hypothesis with the biggest error, which is mostly the case for other algorithms, but depends on all hypotheses.

To find an upper bound that is easier to interpret, we prove the following lemma, which is also stated and proven in Freund and Schapire [3].

Lemma 4.3. *Suppose the setting is the same as in Theorem 4.2. The error* $\varepsilon = \sum_{i=1}^{N} D(i)\mathbb{1}\{h_f(\boldsymbol{x}_i) \neq y_i\}$ *of the final hypothesis* h_f *output by AdaBoost is bounded above by*

$$\varepsilon \leq \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right) \tag{21}$$

with $\gamma_t = \frac{1}{2} - \varepsilon_t$*. In the case that the errors* ε_t *of all the hypotheses are smaller than or equal to* $\frac{1}{2} - \gamma$*, Eq. (21) implies that*

$$\varepsilon \leq \exp(-2T\gamma^2) \tag{22}$$

Proof. As we have proven in Theorem 4.2, we already know that the error is bounded above by $\varepsilon \leq 2^T \prod_{t=1}^{T} \sqrt{\varepsilon_t\,(1 - \varepsilon_t)}$.

We find that

$$\begin{aligned}
\varepsilon &\leq 2^T \prod_{t=1}^{T} \sqrt{\varepsilon_t\,(1 - \varepsilon_t)} \\
&= 2^T \prod_{t=1}^{T} \sqrt{\left(\tfrac{1}{2} - \gamma_t\right)\left(1 - \left(\tfrac{1}{2} - \gamma_t\right)\right)} \\
&= \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2}
\end{aligned} \tag{23}$$

Now we use the Kullback-Leibler divergence, so that Pinsker's inequality can be used. As described in [1], this divergence is defined as

$$\mathrm{kl}\,(p, q) = p \ln\left(\tfrac{p}{q}\right) + (1 - p)\ln\left(\tfrac{1-p}{1-q}\right) \tag{24}$$

By choosing $p = \frac{1}{2}$ and $q = \frac{1}{2} - \gamma_t$ we get

$$\begin{aligned}
\mathrm{kl}\left(\tfrac{1}{2}, \tfrac{1}{2} - \gamma_t\right) &= \tfrac{1}{2}\ln\left(\frac{\frac{1}{2}}{\frac{1}{2} - \gamma_t}\right) + \left(1 - \tfrac{1}{2}\right)\ln\left(\frac{1 - \frac{1}{2}}{1 - \left(\frac{1}{2} - \gamma_t\right)}\right) \\
&= \tfrac{1}{2}\ln\left(\tfrac{1}{1 - 2\gamma_t}\right) + \tfrac{1}{2}\ln\left(\tfrac{1}{1 + 2\gamma_t}\right) \\
&= -\ln\left(\sqrt{1 - 4\gamma_t^2}\right)
\end{aligned} \tag{25}$$

Now it follows that

$$\begin{aligned}\varepsilon &\leq \prod_{t=1}^{T} \sqrt{1-4\gamma_t^2} \\ &= \exp\left(-\sum_{t=1}^{T} \mathrm{kl}\left(\tfrac{1}{2}, \tfrac{1}{2}-\gamma_t\right)\right) \end{aligned} \tag{26}$$

As stated in equation (2.8) in the article of Bubeck and Cesa-Bianchi [1], for every $p, q \in \mathbb{R}$, the following holds:

$$\mathrm{kl}(p,q) \geq 2(p-q)^2.$$

Plugging in $p = \frac{1}{2}$ and $q = \frac{1}{2} - \gamma_t$, gives

$$\mathrm{kl}(\tfrac{1}{2}, \tfrac{1}{2}-\gamma_t) \geq 2(\tfrac{1}{2} - (\tfrac{1}{2}-\gamma_t))^2 = 2\gamma_t^2 \tag{27}$$

The following upper bound for ε follows:

$$\begin{aligned}\varepsilon &\leq \prod_{t=1}^{T} \sqrt{1-4\gamma_t^2} = \exp\left(-\sum_{t=1}^{T} \mathrm{kl}\left(\tfrac{1}{2}, \tfrac{1}{2}-\gamma_t\right)\right) \\ &\leq \exp\left(-2\sum_{t=1}^{T} \gamma_t^2\right) \end{aligned} \tag{28}$$

Note that if the errors ε_t of all the hypotheses are smaller than or equal to $\frac{1}{2} - \gamma$, this implies

$$\varepsilon \leq \exp\left(-2T\gamma^2\right)$$

□

4.2 Normal Hedge

Now we consider NH-Boost.DT based on NormalHedge.DT as described by Luo and Schapire [6]. For $t \in \{1, \ldots, T\}$, we define edge $\gamma_t = \frac{1}{2}\sum_{i=1}^{N} p_{t,i} y_i h_t(\boldsymbol{x}_i)$. This edge γ_t can be interpreted as the advantage of hypothesis h_t on iteration t over random guessing. So hypothesis h_t is correct for an example with probability $\frac{1}{2} + \gamma_t$. Now NH-Boost.DT guarantees that there exists a small edge γ such that $\gamma \leq \gamma_t$ for all $t \in \{1, \ldots, T\}$. Let $(\boldsymbol{x}_i, y_i)_{i=1,\ldots,N}$ be the set of training examples where $\boldsymbol{x}_i \in \mathbb{R}^d$ is an example and $y_i \in \{-1, 1\}$ its label. Now we can prove the following theorem, as stated and proven in Luo and Schapire [6], for NH-Boost.DT.

Theorem 4.4. *Let $\mathcal{Y} = \{-1, 1\}$ be the set of labels. After T rounds, the training error of NH-Boost.DT is at most $\exp(-\frac{1}{3}T\gamma^2 + \ln(\ln T^{\frac{3}{2}} + \frac{5}{2}))$, which is up to logarithmic factors of order $\tilde{O}(\exp(-\frac{1}{3}T\gamma^2))$.*

Proof. Let $(\tilde{\boldsymbol{x}}_i, \tilde{y}_i)_{i=1,2,\ldots,N}$ be the set examples, ordered such that

$$\sum_{t=1}^{T} \tilde{y}_1 h_t(\tilde{\boldsymbol{x}}_1) \leq \sum_{t=1}^{T} \tilde{y}_2 h_t(\tilde{\boldsymbol{x}}_2) \leq \cdots \leq \sum_{t=1}^{T} \tilde{y}_N h_t(\tilde{\boldsymbol{x}}_N) \tag{29}$$

Set $l_{t,i} = \mathbb{1}\{h_t(\boldsymbol{x}_i) = y_i\} = \frac{1}{2} y_i h_t(\boldsymbol{x}_i) + \frac{1}{2}$ and denote by $\tilde{l}_{t,j}$ the loss on the sorted examples: $\tilde{l}_{t,j} = \mathbb{1}\{h_t(\tilde{\boldsymbol{x}}_j) = \tilde{y}_j\}$. Note that $h_t(\tilde{\boldsymbol{x}}_i)\tilde{y}_i = 1$ if and only if $h_t(\tilde{\boldsymbol{x}}_i) = \tilde{y}_i$. Besides, we have $h_t(\tilde{\boldsymbol{x}}_i)\tilde{y}_i = -1$ if and only if $h_t(\tilde{\boldsymbol{x}}_i) \neq \tilde{y}_i$. Now for every $i \in \{1, \ldots, N\}$ the following holds:

$$\begin{aligned}
\sum_{t=1}^{T} h_t(\tilde{\boldsymbol{x}}_i)\tilde{y}_i &= \sum_{t=1}^{T} \mathbb{1}\{h_t(\tilde{\boldsymbol{x}}_i) = \tilde{y}_i\} - \sum_{t=1}^{T} \mathbb{1}\{h_t(\tilde{\boldsymbol{x}}_i) \neq \tilde{y}_i\} \\
&= \sum_{t=1}^{T} \mathbb{1}\{h_t(\tilde{\boldsymbol{x}}_i) = \tilde{y}_i\} - \left(T - \sum_{t=1}^{T} \mathbb{1}\{h_t(\tilde{\boldsymbol{x}}_i) = \tilde{y}_i\}\right) \\
&= 2\sum_{t=1}^{T} \tilde{l}_{t,i} - T
\end{aligned} \tag{30}$$

This implies that if $i, j \in \{1, \ldots, N\}$ such that $j \leq i$, and thus $\sum_{t=1}^{T} \tilde{y}_j h_t(\tilde{\boldsymbol{x}}_j) \leq \sum_{t=1}^{T} \tilde{y}_i h_t(\tilde{\boldsymbol{x}}_i)$, we have

$$\sum_{t=1}^{T} \tilde{y}_j h_t(\tilde{\boldsymbol{x}}_j) = 2\sum_{t=1}^{T} \tilde{l}_{t,j} - T \leq \sum_{t=1}^{T} \tilde{y}_i h_t(\tilde{\boldsymbol{x}}_i) = 2\sum_{t=1}^{T} \tilde{l}_{t,i} - T$$

$$\text{So } \sum_{t=1}^{T} \tilde{l}_{t,j} \leq \sum_{t=1}^{T} \tilde{l}_{t,i}$$

Now we use for every $t \in \{1, \ldots, T\}$:

$$\gamma \leq \gamma_t$$

So it follows that

$$\begin{aligned}
\tfrac{1}{2} + \gamma &\leq \tfrac{1}{T}\sum_{t=1}^{T} \tfrac{1}{2} + \gamma_t \\
&= \tfrac{1}{2} + \frac{1}{T}\sum_{t=1}^{T} \frac{1}{2} \sum_{i=1}^{N} p_{t,i} y_i h_t(\boldsymbol{x}_i) \\
&= \frac{1}{2} + \frac{1}{T}\sum_{t=1}^{T} \left(\sum_{i=1}^{N} p_{t,i} l_{t,i} - \frac{1}{2}\right) \\
&= \frac{1}{T}\sum_{t=1}^{T} \left(\sum_{i=1}^{N} p_{t,i} l_{t,i}\right)
\end{aligned}$$

Consider the ϵ-regret R_T^{ϵ} for $\epsilon = j/N$. Then for all $j \in \{1, \ldots, N\}$ the following holds:

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T} \tilde{l}_{t,j} + \frac{R_T^{j/N}}{T} &= \frac{1}{T}\sum_{t=1}^{T} \tilde{l}_{t,j} + \frac{1}{T}\left(\sum_{t=1}^{T} \boldsymbol{p}_t \cdot \boldsymbol{l}_t - \sum_{t=1}^{T} l_{t,i_{j/N}}\right) \\
&= \frac{1}{T}\sum_{t=1}^{T} \left(\tilde{l}_{t,j} + \left(\sum_{i=1}^{N} p_{t,i} \cdot l_{t,i}\right) - \tilde{l}_{t,j}\right) \\
&= \frac{1}{T}\sum_{t=1}^{T} \left(\sum_{i=1}^{N} p_{t,i} l_{t,i}\right)
\end{aligned} \tag{31}$$

So for all $j \in \{1, \ldots, N\}$ we have:

$$\tfrac{1}{2} + \gamma \leq \frac{1}{T} \sum_{t=1}^{T} \left(\sum_{i=1}^{N} p_{t,i} l_{t,i} \right) = \frac{1}{T} \sum_{i=1}^{T} \tilde{l}_{t,j} + \frac{R_T^{j/N}}{T} \tag{32}$$

Note that $\gamma_t = \frac{1}{2} \sum_i p_{t,i} y_i h_t(\boldsymbol{x}_i) = \sum_i p_{t,i} l_{t,i} - \sum_i \frac{1}{2} p_{t,i} = \boldsymbol{p}_t \cdot \boldsymbol{l}_t - \frac{1}{2}$. For the $s_{t,i}$ of the NH-Boost.DT algorithm we find

$$s_{t,i} = s_{t-1,i} + \tfrac{1}{2} y_i h_t(\boldsymbol{x}_i) - \gamma_t = s_{t-1,i} + l_{t,i} - \boldsymbol{p}_t \cdot \boldsymbol{l}_t \tag{33}$$

for all $i \in \{1, \ldots, N\}$ and $t \in \{1, \ldots, T\}$. So the weights in NH-Boost.DT are updated according to the General Hedge Algorithm described in Algorithm 3 of Luo and Schapire [6], where the loss is set to be $l_{t,i} = \mathbb{1}\{h_t(\boldsymbol{x}_i) = y_i\}$ and $\phi_T(s) = \exp\left(\frac{[s]_-^2}{3T}\right)$, with $[s]_- = \min\{0, s\}$.

In Corollary 2 of Luo and Schapire [6], it is found that the regret now is bounded above as follows.

$$R_T^{\varepsilon} \leq \sqrt{3T \ln\left(\frac{1}{2\varepsilon}(e^{4/3} - 1)(\ln T + 1) + 1 \right)} \tag{34}$$

Plugging $\frac{j}{N}$ in for ε, we find

$$\begin{aligned} R_T^{\frac{j}{N}} &\leq \sqrt{3T \ln\left(\frac{1}{2j/N}(e^{4/3} - 1)(\ln T + 1) + 1 \right)} \\ &\leq \sqrt{3T \ln\left(\frac{3N}{2j}(\ln T + 1) + 1 \right)} \\ &\leq \sqrt{3T \ln\left(\frac{N}{j}(\ln T^{\frac{3}{2}} + \tfrac{5}{2}) \right)} \end{aligned} \tag{35}$$

Now Eq. (32) gives us that

$$\begin{aligned} \tfrac{1}{2} + \gamma &\leq \frac{1}{T} \sum_{i=1}^{T} \tilde{l}_{t,j} + \frac{R_T^{j/N}}{T} \\ &\leq \frac{1}{T} \sum_{i=1}^{T} \tilde{l}_{t,j} + \sqrt{\frac{3 \ln\left(\frac{N}{j}(\ln T^{\frac{3}{2}} + \frac{5}{2}) \right)}{T}} \end{aligned} \tag{36}$$

Suppose j is such that $\gamma > \sqrt{\frac{3 \ln\left(\frac{N}{j}(\ln T^{\frac{3}{2}} + \frac{5}{2}) \right)}{T}}$. Then it follows that $\frac{1}{T} \sum_{t=1}^{T} \tilde{l}_{t,j} = \frac{1}{T} \sum_{t=1}^{T} \mathbb{1}\{h_t(\tilde{\boldsymbol{x}}_j) = \tilde{y}_j\} > \frac{1}{2}$. Since the final hypothesis $h_f(\boldsymbol{x})$ is a majority vote, this means the final hypothesis will be correct for example $(\tilde{\boldsymbol{x}}_j, \tilde{y}_j)$. Since we know for $i \geq j$ that $\sum_{t=1}^{T} \tilde{l}_{t,i} \geq \sum_{t=1}^{T} \tilde{l}_{t,j}$, the final hypothesis will be correct for all $i \geq j$. So the training error will be at most $\frac{j-1}{N}$, since the final hypothesis

can only be wrong for the first $j-1$ examples. So we want to find the smallest j such that

$$\gamma > \sqrt{\frac{3\ln\left(\frac{N}{j}(\ln T^{\frac{3}{2}} + \frac{5}{2})\right)}{T}} \tag{37}$$

Now the following must hold:

$$j > Ne^{-\frac{1}{3}T\gamma^2}(\ln T^{\frac{3}{2}} + \tfrac{5}{2}) \tag{38}$$

Note that the theorem is vacuous if we have

$$-\tfrac{1}{3}T\gamma^2 + \ln(\ln T^{\frac{3}{2}} + \tfrac{5}{2}) \geq 0 \tag{39}$$

so without loss of generality we can assume that the smallest j for which Eq. (37) holds is smaller than N, so there exists such a j. Since we took the smallest j for which (37) holds, the training error has the following upper bound:

$$\varepsilon \leq \frac{j-1}{N} < \exp(-\tfrac{1}{3}T\gamma^2 + \ln(\ln T^{\frac{3}{2}} + \tfrac{5}{2})) \tag{40}$$

□

5 Plugging Squint into a boosting algorithm

As stated before, Squint is an online allocation algorithm that is proven to have a better regret bound than Hedge and NormalHedge.DT. Since it is not mentioned in Freund and Schapire [3] how online allocation algorithms can be plugged into a boosting algorithm, we first have to determine how this can be done for Squint.

5.1 SquintBoost

Comparing the NH-Boost.DT algorithm with NormalHedge.DT, we can see how the online allocation algorithm is plugged into the boosting algorithm. By setting the weights in this algorithm according to Squint instead of NormalHedge.DT, we create a new boosting algorithm SquintBoost. The new algorithm is given in Fig. 4. Note that for SquintBoost we need a prior distribution D. For Theorem 5.2, we choose the uniform distribution as prior distribution.

5.2 Upper bound of the training error of SquintBoost

The regret bound of Squint is given in Theorem 4 of Koolen and van Erven [5]. To make the notation more consistent, we define for Squint $w_t^i = p_{t,i}$ where t denotes the iteration and i the expert.

Algorithm 4 Squint-Boost

Require:
sequence of N labeled examples $\langle(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\rangle$, $y_i \in \{-1, 1\}$
weak learning algorithm **WeakLearn**
integer T specifying number of iterations
prior D over the N examples

1: **procedure Boosting**
2: **for** $t = 1, 2, \ldots, T$ **do**
3: Set $p_{t,i} \propto D(i) \int_0^{\frac{1}{2}} \exp\left(\eta \sum_{s=1}^{t-1} \left(\frac{1}{2} \sum_{j=1}^{N} p_{s,j} y_j h_s(\boldsymbol{x}_j)\right) - p_{s,i} y_i h_s(\boldsymbol{x}_i) - \eta^2 \sum_{s=1}^{t-1} \left(\left(\frac{1}{2} \sum_{j=1}^{N} p_{s,j} y_j h_s(\boldsymbol{x}_j)\right) - p_{s,i} y_i h_s(\boldsymbol{x}_i)\right)^2\right) d\eta$ for all i.
4: Call **WeakLearn**, providing it with the distribution $\boldsymbol{p}_t$ and receive a hypothesis $h_t : X \to \{-1, 1\}$ with edge $\gamma_t = \frac{1}{2} \sum_i p_{t,i} y_i h_t(\boldsymbol{x}_i)$.
5: **end for**
6: **return** final hypothesis

$$h_f(\boldsymbol{x}) : \mathbb{R}^d \to \{-1, 1\}, h_f(\boldsymbol{x}) := \operatorname{sign}\left(\sum_{t=1}^{T} h_t(\boldsymbol{x})\right)$$

7: **end procedure**

Theorem 5.1. *With respect to any subset of experts $\mathcal{K}$, $i = |\mathcal{K}|$, the regret of Squint with improper prior, which chooses weights*

$$p_{T+1,i} \propto D(i) \int_0^{1/2} e^{\eta R_T^i - \eta^2 V_T^i} d\eta$$

is bounded by

$$R_T^{\mathcal{K}} \leq \sqrt{2V_T^{\mathcal{K}}} \left(1 + \sqrt{2 \ln\left(\frac{\frac{1}{2} + \ln(T+1)}{D(\mathcal{K})}\right)}\right) + 5 \ln\left(1 + \frac{1 + 2\ln(T+1)}{D(\mathcal{K})}\right)$$

Let $(\boldsymbol{x}_i, y_i)_{i=1,\ldots,N}$ be the set of training examples where $\boldsymbol{x}_i \in \mathbb{R}^d$ is an example and $y_i \in \{-1, 1\}$ its label. Now we can prove the following theorem for Squint-Boost, which is the main result of this thesis.

Theorem 5.2. *Let $\mathcal{Y} = \{-1, 1\}$ be the set of labels. After T rounds, the training error of Squint-Boost with the uniform distribution as prior distribution is at most*

$$\exp\left(-\frac{1}{2}\left(\frac{T\gamma - 5\ln\left(2N(1+\ln(T+1))\right)}{\sqrt{2V_T^{\mathcal{K}}}} - 1\right)^2 + \ln(1 + 2\ln(T+1))\right),$$

which is up to logarithmic factors of order $\tilde{O}\left(\exp\left(-\frac{1}{4}\frac{T^2\gamma^2}{V_T^{\mathcal{K}}}\right)\right)$.

It is stated in Koolen and van Erven [5] that often the variance is small, so $V_T^{\mathcal{K}} \ll T$. For large T the upper bound for the training error is of the order

$$\exp\left(-\frac{1}{4}\frac{T^2\gamma^2}{V_T^{\mathcal{K}}}\right) \ll \exp\left(-\frac{1}{4}T\gamma^2\right) \tag{41}$$

Proof. Let $(\tilde{\boldsymbol{x}}_i, \tilde{y}_i)_{i=1,2,\ldots,N}$ be the set examples, ordered such that

$$\sum_{t=1}^{T} \tilde{y}_1 h_t(\tilde{\boldsymbol{x}}_1) \leq \sum_{t=1}^{T} \tilde{y}_2 h_t(\tilde{\boldsymbol{x}}_2) \leq \cdots \leq \sum_{t=1}^{T} \tilde{y}_N h_t(\tilde{\boldsymbol{x}}_N) \tag{42}$$

Set $l_{t,i} = \mathbb{1}\{h_t(\boldsymbol{x}_i) = y_i\} = \frac{1}{2}y_i h_t(\boldsymbol{x}_i) + \frac{1}{2}$ and denote by $\tilde{l}_{t,i}$ the loss on the sorted examples: $\tilde{l}_{t,j} = \mathbb{1}\{h_t(\tilde{\boldsymbol{x}}_j) = \tilde{y}_j\}$. Recall from the proof for NH-Boost.DT that if $i, j \in \{1, \ldots, N\}$ such that $j \leq i$, and thus $\sum_{t=1}^{T} \tilde{y}_j h_t(\tilde{\boldsymbol{x}}_j) \leq \sum_{t=1}^{T} \tilde{y}_i h_t(\tilde{\boldsymbol{x}}_i)$, we have that

$$\sum_{t=1}^{T} \tilde{l}_{t,j} \leq \sum_{t=1}^{T} \tilde{l}_{t,i} \tag{43}$$

In the same way as in the proof for NH-Boost.DT, it can be found that

$$\tfrac{1}{2} + \gamma \leq \frac{1}{T}\sum_{t=1}^{T}\left(\sum_{i=1}^{N} p_{t,i} l_{t,i}\right)$$

For all $j \in \{1, \ldots, N\}$ we choose $\mathcal{K}$ to be the set of experts with loss smaller or equal to the loss of the j-th ordered example. So $\tilde{l}_{t,i} \leq \tilde{l}_{t,j}$ for all $i \in \mathcal{K}$. It follows that $\mathbb{E}_{D(k|\mathcal{K})}(l_{t,k}) \leq \tilde{l}_{t,j}$. This leads to the following result.

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T} \tilde{l}_{t,j} + \frac{R_T^{\mathcal{K}}}{T} &= \frac{1}{T}\left(\sum_{t=1}^{T} \tilde{l}_{t,j} + \mathbb{E}_{D(k|\mathcal{K})} R_T^k\right) \\
&= \frac{1}{T}\sum_{t=1}^{T}\left(\tilde{l}_{t,j} + \left(\sum_{i=1}^{N} p_{t,i} l_{t,i}\right) - \mathbb{E}_{D(k|\mathcal{K})}(l_{t,k})\right) \\
&\geq \frac{1}{T}\sum_{t=1}^{T}\left(\tilde{l}_{t,j} + \left(\sum_{i=1}^{N} p_{t,i} l_{t,i}\right) - \tilde{l}_{t,j}\right) \\
&= \frac{1}{T}\sum_{t=1}^{T}\left(\sum_{i=1}^{N} p_{t,i} l_{t,i}\right)
\end{aligned} \tag{44}$$

Now for all $j \in \{1, \ldots, N\}$ with $\mathcal{K}$ chosen as before, we have that

$$\tfrac{1}{2} + \gamma \leq \frac{1}{T}\sum_{t=1}^{T}\left(\sum_{i=1}^{N} p_{t,i} l_{t,i}\right) \leq \frac{1}{T}\sum_{t=1}^{T} \tilde{l}_{t,j} + \frac{R_T^{\mathcal{K}}}{T} \tag{45}$$

As stated in Theorem 5.1, the regret of Squint, which is used in Squint-Boost, is bounded from above as follows.

$$R_T^{\mathcal{K}} \leq \sqrt{2V_T^{\mathcal{K}}}\left(1 + \sqrt{2\ln\left(\frac{\frac{1}{2} + \ln(T+1)}{D(\mathcal{K})}\right)}\right) + 5\ln\left(1 + \frac{1 + 2\ln(T+1)}{D(\mathcal{K})}\right)$$

Plugging in the fact that $|\mathcal{K}| = j$ and thus $D(\mathcal{K}) = \frac{j}{N}$ for D the uniform distribution, this implies

$$\begin{aligned} R_T^{\mathcal{K}} &\leq \sqrt{2V_T^{\mathcal{K}}}\left(1+\sqrt{2\ln\left(\frac{\frac{1}{2}+\ln(T+1)}{\frac{j}{N}}\right)}\right)+5\ln\left(1+\frac{1+2\ln(T+1)}{\frac{j}{N}}\right) \\ &\leq \sqrt{2V_T^{\mathcal{K}}}\left(1+\sqrt{2\ln\left(\frac{N}{j}\left(\frac{1}{2}+\ln(T+1)\right)\right)}\right)+5\ln\left(2N(1+\ln(T+1))\right) \end{aligned}$$

Define $\alpha := 5\ln\left(2N(1+\ln(T+1))\right)$ to make the equation more transparent. Now Eq. (45) gives us that

$$\begin{aligned} \tfrac{1}{2}+\gamma &\leq \frac{1}{T}\sum_{t=1}^{T}\tilde{l}_{t,j}+\frac{R_T^{\mathcal{K}}}{T} \\ &\leq \frac{1}{T}\sum_{t=1}^{T}\tilde{l}_{t,j}+\frac{\sqrt{2V_T^{\mathcal{K}}}}{T}\left(1+\sqrt{2\ln\left(\frac{N}{j}\left(\frac{1}{2}+\ln(T+1)\right)\right)}\right)+\frac{\alpha}{T} \end{aligned} \tag{46}$$

Suppose j is such that

$$\gamma > \frac{\sqrt{2V_T^{\mathcal{K}}}}{T}\left(1+\sqrt{2\ln\left(\frac{N}{j}\left(\frac{1}{2}+\ln(T+1)\right)\right)}\right)+\frac{\alpha}{T} \tag{47}$$

Then we find that $\frac{1}{T}\sum_{t=1}^{T}\tilde{l}_{t,j} = \frac{1}{T}\sum_{t=1}^{T}\mathbb{1}\{h_t(\tilde{\boldsymbol{x}}_j)=\tilde{y}_j\} > \frac{1}{2}$. Exactly as in the proof of NH-Boost.DT it follows that the training error will be at most $\frac{j-1}{N}$. So we want to find the smallest j for which Eq. (47) holds.

Solving for j, this means that the following must hold:

$$j > N\exp\left(-\frac{1}{2}\left(\frac{T\gamma-\alpha}{\sqrt{2V_T^{\mathcal{K}}}}-1\right)^2\right)\left(1+2\ln(T+1)\right) \tag{48}$$

Note that the theorem is vacuous if we have

$$-\frac{1}{2}\left(\frac{T\gamma-\alpha}{\sqrt{2V_T^{\mathcal{K}}}}-1\right)^2+\ln(1+2\ln(T+1)) \geq 0 \tag{49}$$

so without loss of generality we can assume that the smallest j for which Eq. (47) holds is smaller than N, so there exists such a j. Since we took the smallest j for which (47) holds, the training error has the following upper bound:

$$\varepsilon < \exp\left(-\frac{1}{2}\left(\frac{T\gamma-5\ln\left(2N(1+\ln(T+1))\right)}{\sqrt{2V_T^{\mathcal{K}}}}-1\right)^2+\ln(1+2\ln(T+1))\right)$$

□

5.3 Comparing upper bounds

Now we have found the upper bounds for the different algorithms. For large T the upper bounds are in the order of the values in the table below.

Algorithm	Upper bound
AdaBoost	$\exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right)$
NH-Boost.DT	$\exp\left(-\frac{1}{3}T\gamma^2\right)$
SquintBoost	$\exp\left(-\frac{1}{4}\frac{T^2\gamma^2}{V_T^{\mathcal{K}}}\right)$

As is mentioned before, the first two upper bounds are comparable, but since NH-Boost.DT is faster per iteration, NH-Boost.DT can run more iterations in the same amount of time. So the training error decreases faster for NH-Boost.DT than for AdaBoost, which is illustrated in the experiments in appendix G of Luo and Schapire [6]. After the same number of rounds, the training error of AdaBoost and NH-Boost.DT do not differ much, but NH-Boost.DT needs less time, so could have produced a smaller training error in the same amount of time.

Now we have found an upper bound for SquintBoost, which is significantly lower than the upper bounds of AdaBoost and NormalHedge if $V_T^{\mathcal{K}} \ll T$. In Vente [7] it is tested with experiments whether this also means the training error will be lower in practice too. Surprisingly, these experiments show that the theoretical lower upper bound for the training error of SquintBoost does not result in an algorithm with lower generalized error than NH-Boost.DT. The results show that after the same number of iterations, the generalized error of NH-Boost.DT is smaller than that of SquintBoost.

6 Summary and future work

We have constructed a new boosting algorithm, SquintBoost, using the online allocation algorithm Squint. For this algorithm we have proved that if $V_T^{\mathcal{K}}$ is significantly smaller than T, the upper bound of the training error is lower than the known upper bounds of the training errors of the existing boosting algorithms AdaBoost and NH-Boost.DT. However, in Vente [7] it is shown that this lower upper bound does not result in a lower generalized error in practice. NH-Boost.DT outperforms SquintBoost in these experiments.

An issue for future work is to find out why SquintBoost does not benefit of this lower upper bound for the training error in practice. If the cause of this is found, it could perhaps indicate how the algorithm can be improved. By taking a closer look at the values of $V_T^{\mathcal{K}}$, it could be determined whether the variance indeed gets much smaller than T in practice, as is assumed in Koolen and van Erven [5]. If this is not the case, the upper bound is not better than the upper bounds of the other algorithms and this could explain why the performance of SquintBoost is not better than that of NH-Boost.DT.

References

[1] Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Found. Trends Mach. Learn.*, 5 (1):1–122, 2012.

[2] Kamalika Chaudhuri, Yoav Freund, and Daniel J. Hsu. A parameter-free hedging algorithm. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 297–305. 2009.

[3] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. System Sci.*, 55 (1, part 2):119–139, 1997.

[4] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning*. Springer Series in Statistics. Springer, New York, second edition, 2009. ISBN 978-0-387-84857-0. Data mining, inference, and prediction.

[5] Wouter M. Koolen and Tim van Erven. Second-order quantile methods for experts and combinatorial games. *Proceedings of The 28th Conference on Learning Theory*, 2015.

[6] Haipeng Luo and Robert E. Schapire. A drifting-games analysis for online learning and applications to boosting. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1368–1376. Curran Associates, Inc., 2014.

[7] Daniel Vente. A practical comparison of boosting algorithms, 2016. To appear at `https://www.math.leidenuniv.nl/nl/theses/year/2016/`.