Boaz Leskes

The Value of Agreement

a new boosting algorithm

Master Thesis

Section Computational Science

and

Institute for Logic, Language & Computation

University of Amsterdam

Supervisior: Leen Torenvliet

Abstract

In the past few years unlabeled examples and their potential advantage have received a lot of attention. In this paper a new boosting algorithm is presented where unlabeled examples are used to enforce agreement between several different learning algorithms. Not only do the learning algorithms learn from the given training set but they are supposed to do so while agreeing on the unlabeled examples. Similar ideas have been proposed before (for example, the Co-Training algorithm by Mitchel and Blum), but without a proof or under strong assumptions. In our setting, it is only assumed that all learning algorithms are equally adequate for the tasks. A new generalization bound is presented where the use of unlabeled examples results in a better ratio between training-set size and the the resulting classifier's quality. The extent of this improvement depends on the diversity of the learners—a more diverse group of learners will result in a larger improvement whereas using two copies of a single algorithm gives no advantage at all. As a proof of concept, the algorithm, named AgreementBoost, is applied to two test problems. In both cases, using AgreementBoost results in an up to 40% reduction in the number of labeled examples.

Contents

Chapter 1

Introduction

Imagine a young pupil being taught for the first time to read a foreign language. The teacher might begin the first lesson by writing the entire alphabet of the language on the board announcing that today they are going to learn the first letter, say '$\aleph$.' After a rather cumbersome hour of staring at letters, the pupil is surprised by the teacher's last announcement: the following lesson will begin with a short quiz covering today's material. The quiz will take the following form: each pupil will receive a sheet of paper written in that foreign language and will have to mark all occurrences of the letter '$\aleph$.' Furthermore, imagine that the teacher of this young pupil is a particularly bad one. She does not give the pupil any guidance nor explanation with respect to how an '$\aleph$' should be recognized out of the whole alphabet. Rather the teacher spends the whole lesson showing the students transparencies of different letters, saying for each one whether it is a nice example of the lesson's subject ('$\aleph$') or that it is a letter they will study at a later time.

The problem faced by this poor pupil can be modeled by one of the simplest but popular models in machine learning called *supervised learning.* This model represents a scenario where a 'learner' is required to solve a *classification problem.* The model assumes the existence of a set of possible examples $\mathcal{X}$ which are divided in some way into a set of classes $\mathcal{Y}$ (often called labels). In the pupil's case, the example space can be the set of all possible images of letters in the alphabet (one letter per image) and the classes are '$\aleph$' and non-'$\aleph$.' The pupil then comes up with a mapping $f : \mathcal{X} \to \mathcal{Y}$, saying for each letter in the quiz whether it is an '$\aleph$' or not. In order to evaluate the quality of the pupil's mapping it is assumed that there exists a distribution P over $\mathcal{X} \times \mathcal{Y}$ which represents the 'chance' to see a specific example and it's label[1] in real life. The quality of the pupil's mapping, or classifier, is then measured by the probability of it making a mistake. In other words, the lower the probability of the classifier making a mistake, the better the classifier.

In class, the only interaction between teacher and pupil is achieved by showing the pupil a set of examples along with their correct label. This form of interaction is modeled by assuming that the learner is given a finite sample of labeled examples S, drawn at random and independently from P. Since this is the only information the pupil receives, she must use this sample to generate the resulting classifier.

More formally, the learning model is defined as follows: it is assumed that the learning algorithm is given only a finite sequence of training examples. The training

[1]Note that this setting allows an example to belong to different classes, or have several labels. This is helpful when modeling noise or non-deterministic processes.

examples are given in the form of a set of ordered pairs, $S = \left\{(x_j, y_j)\right\}_{j=1}^{n_s}$, where each instance x_j belongs to some *instance space* $\mathcal{X}$ and is accompanied by a single label out of a *label space*[2] $\mathcal{Y} \subseteq [-1, +1]$. Furthermore, it is assumed that the training set S was generated by repeatedly and independently sampling some arbitrary probability distribution P over $\mathcal{X} \times \mathcal{Y}$. The task of a learning algorithm is then to use the training set to construct a classifier, or a hypothesis that best predicts the labels given to examples from $\mathcal{X}$ by P. Typically the learning algorithm can only choose a hypothesis from a limited set of possible hypotheses, which is referred to as the hypothesis space $H \subseteq \mathcal{Y}^{\mathcal{X}}$.

Though the above learning scenario is somewhat contrived, many real life applications fall nicely within this model. Problems like OCR, web pages classification (as done in Internet directories) and detection of spam e-mail are only a few of many problems that fit into this scheme. Note that the assumption regarding the poor didactic skills of the teacher is made to serve a very practical reason: how does one explain to an e-mail program how to detect a spam e-mail? If this was easy, it would have been much simpler to write a program to identify spam rather then running a learning algorithm. Devising concrete rules that precisely discern spam from non-spam is much harder (to humans) than the pure proclamation "This is spam." In other words, the algorithm is used to transfer our somehow vague notion of "This is spam" to a concrete rule that can be efficiently executed by a computer. Therefore, an algorithm that is able to 'learn' concrete classification rules from a labeled sample alone demands less from the 'teacher'—the user of the learning algorithm.

In all the examples above and in many others, it is relatively hard to obtain a large sample of labeled examples. The sample has to be carefully analyzed and labeled by humans—a costly and time consuming task. However in many situations it is fairly easy to obtain unlabeled examples: examples from the example space $\mathcal{X}$ *without* the class that they belong to. This process can be easily mechanized and preformed by a machine, much faster then any human-plausible rate. This difference between labeled and unlabeled examples has encouraged researchers in the recent years to study the benefits that unlabeled examples may have in various learning scenarios. For example, will it help the young pupil if she will be given a booklet written in that foreign language to take home? Can this booklet and the abundant unlabeled-examples within be used to improve the pupil's results in the coming exam?

1.1 The Co-training model

At first glance it might seem that nothing is to be gained from unlabeled examples. After all, unlabeled examples lack the most important piece of information—the class to which they belong. However, this is not necessarily the case. In some theoretical settings, it is beneficial to gain knowledge over the examples' marginal distribution $P(x)$ (for example in [7]). In these cases, having extra examples, with or without their label, provides this extra information. On the other hand, there exist situations (for example in [10]) where knowing $P(x)$ is not helpful and unlabeled examples do not help at all. The main goal of this sort of research is to determine the amount of information that can be extracted from unlabeled examples. However, unlabeled examples have also been used by algorithms in a more practical way: as a sort of a communication

[2] For technical reasons, the label space is restricted to $[-1, +1]$. In the general case, $\mathcal{Y}$ may be different. For example, in a multi class problem one typically chooses $\mathcal{Y}$ to be $\{1, 2, 3..., M\}$ where M is the number of possible classes.

platform between two different learning algorithms. One such usage is the so called Co-Training model or strategy.

A typical example of Co-Training can be found in [5], a paper often cited with respect to unlabeled examples. In their paper, Blum and Mitchel provide both an algorithm and a theoretical framework where unlabeled examples are used to communicate an 'opinion' about an unlabeled example from one algorithm to another. As a case study, the algorithm is then applied to a web-page classification problem involving identifying courses' homepages out of a collection of web-pages.

In their work, Blum and Mitchel assume that the example space can be split into two 'views' $\mathcal{X}^1$ and $\mathcal{X}^2$ i.e., $\mathcal{X} = \mathcal{X}^1 \times \mathcal{X}^2$. For example, in the web-page classification problem, a web-page can be represented as the words appearing on the web-page itself (view 1) but also as words appearing on links pointing to it. Further they assume that examples are labeled by two target functions f^1, f^2, one for each view. In order to comply with the fact that each example belongs to a single class, they further assume that $P\left(\left\{x^1 \times x^2 : f^1\left(x^1\right) \neq f^2\left(x^2\right)\right\}\right) = 0$. Lastly, they assume that there exist two learning algorithms L^1 and L^2 such that L^1 can learn using the first view alone and L^2 can learn using the second view and in the presence of noise. Furthermore, they suppose that both views are sufficient for learning the problem.

The main theoretical result in their paper is the following: under some assumptions, a classifier that uses the first view and that is slightly better than random guessing is equivalent to a noisy source when looked upon from the point of view of L^2. Furthermore, the noise rate of the source is strictly smaller than $\frac{1}{2}$ and therefore it contains some information on the original labels. If L^1 is trained on the labeled examples to achieve such a classifier, it can then be used to label unlabeled examples and supply a noisy source for L^2 to learn from. Since L^2 can learn in the presence of noise, it can then use the abundant newly labeled examples to produce a good classifier.

This ability to transform a deterministic process to random noise entails a very severe assumption: for every fixed example $\left(\hat{x}^1, \hat{x}^2\right) \in \mathcal{X}$ of non-zero probability it must hold that:

$$\begin{aligned} P\left(X^1 = \hat{x}^1 \mid X^2 = \hat{x}^2\right) &= P\left(X^1 = \hat{x}^1 \mid f^2\left(X^2\right) = f^2\left(\hat{x}^2\right)\right) \\ P\left(X^2 = \hat{x}^2 \mid X^1 = \hat{x}^1\right) &= P\left(X^2 = \hat{x}^2 \mid f^1\left(X^1\right) = f^1\left(\hat{x}^1\right)\right). \end{aligned} \tag{1.1}$$

In other words, that $X^1$and X^2 are conditionally independent given the label. As the authors themselves state, only four hypotheses comply with this assumption (assuming that P allows for it).

The algorithm presented by Blum and Mitchel (see Algorithm 1) has been shown to produce better classifiers in the web-pages problem and in other experiments (for example, [12, 11], for more detailed analysis and limitations see [13, 16]). However, the theory presented can only be used as a motivation or a general intuition for the algorithm's success. The algorithm does not train one learner and uses it to label *all* unlabeled examples, which are in turn given to the second learner. Instead, both learners are allowed to choose *some* unlabeled examples for labeling. These unlabeled examples are then added to the pool of labeled examples. Therefore, after being labeled, an unlabeled example assumes the same role as a labeled example: a true representation of the target function. As the authors themselves remark, this process encourages the learners to slowly *agree* on the labels of the unlabeled examples. In the last iteration of the Co-Training algorithm, the classifiers produced are the result of underlying learners doing their best to fit the same set of labeled and newly-labeled examples. If the number of newly-labeled examples is considerately larger, this implies that the learners

Algorithm 1 The Co-Training algorithm, as presented in [5]

```
Inputs:

  • A set of labeled training examples ($S$)
  • A set of unlabeled examples ($U$)

1. Create a pool $U'$ of examples by choosing $n_u$ unlabeled examples, at random, from $U$.
2. Loop for $k$ iterations:
    (a) Use $S$ and $L^1$ to train a classifier $h^1$ that considers only the $x^1$ portion of the examples.
    (b) Use $S$ and $L^2$ to train a classifier $h^2$ that considers only the $x^2$ portion of the examples.
    (c) Allow $h^1$ to label $p$ positive and $n$ negative examples from $U'$.
    (d) Allow $h^2$ to label $p$ positive and $n$ negative examples from $U'$.
    (e) Add these self-labeled examples to $S$.
    (f) Randomly choose $2p+2n$ examples from $U$ to replenish $U'$.
```

are encouraged to agree on a set of examples whose labels were not given to them in advance. Before starting the algorithm, there is no guarantee as to what label an unlabeled example will receive but only to the fact that the end classifiers are highly likely to agree on that label. This type of agreement is a side effect of many other variants of the co-training model [15, 14].

This intuition that agreement is useful and can assist in the task of learning is elaborated and made more precise in this paper. A theoretical framework is presented where agreement between different learners has a clear advantage[3]. Furthermore, these ideas are carried over to the field of boosting, where the theoretical settings are especially applicable. This is achieved in the form of a new algorithm that builds upon this theory. A similar attempt can be found in [17] where a boosting algorithm is presented, based on the above intuition. However, no proof is provided that the algorithm does result in agreeing classifiers nor for the advantage of such an agreement. A proof for the latter (in a more general settings) was provided by Dasgupta et al. in [18]. Nevertheless, for the proof to hold one still has to use the strong assumption of view-independence (Equation 1.1). Another example for the use of unlabeled examples in boosting can be found in [23].

Before delving into the theoretical benefits of agreement, we present a short introduction to boosting.

1.2 Boosting

The basic idea behind boosting is to iteratively combine relatively simple hypotheses to create a more complex classifier—a classifier that is superior to all underlying hypothe-

[3]Assuming that all learners are different yet adequate for the task.

Algorithm 2 AdaBoost [19]

```
Input: A sample $S = \{(x_i, y_i)\}_{i=1}^{n_s}$ and an underlying learning algorithm $L$.

1. Initialize: $d_i^{(1)} = 1/n_s$ for all $i = 1, \ldots, n_s$.
2. Do for $t = 1, \ldots, T$:
    (a) Use $L$ to obtain a hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$ trained on $S$, re-weighted
        according to $\mathbf{d}^{(t)}$.
    (b) Calculate the weighted training error $\varepsilon_t$ of $h_t$:
        $\varepsilon_t = \sum_{i=1}^{n_s} d_i^{(t)} \mathbf{I}(y_i \neq h_t(x_i))$ where $\mathbf{I}(y_i \neq h_t(x_i)) = \left[ \begin{array}{ll} 1 & \text{if } y_i \neq h_t(x_i) \\ 0 & \text{otherwise} \end{array} \right.$
    (c) Set $\alpha_t = \frac{1}{2} \log \frac{1-\varepsilon_t}{\varepsilon_t}$.
    (d) Update weights: $d_i^{(t+1)} = d_i^{(t)} \exp(-\alpha_t y_i h_t(x_i)) / Z_t$, where $Z_t$ is a normal-
        ization constant such that $\sum_{i=1}^{n_s} d_i^{(t+1)} = 1$.
3. Output $sign\left(\sum_{t=1}^{T} \alpha_t h_t\right)$ as resulting classifier.
```

ses. While coming up with a classifier (and an algorithm to construct it) that can adequately solve a supervised learning problem is hard, devising a 'rule of thumb' which is just slightly better than random guessing is fairly easy in many problems. Take, for example, the web-page classification problem from before. One can use the existence of the word 'course' in a web-page as a relatively good indication that this is a course home page. While this criterion is far from perfect (for example, a university schedule is also highly likely to contain the word 'course') it is still much better than random guessing. Other such words may include 'homework','class','lesson','assignment' etc.

In the last decade boosting has grown in popularity and has received considerable attention. One of the first practical boosting algorithms was AdaBoost (**Ada**ptive **Boost**ing, see Algorithm 2), introduced by Freund and Schapire in [19]. Since then, it was thoroughly researched and many variants of it were derived. Since AdaBoost is such a typical and simple boosting algorithm, it will be used here as a representative example.

The main idea behind AdaBoost (and many other boosting algorithms) is to assign to each of the training examples a weight. An underlying learning algorithm is then used to generate a (weak) hypothesis based on this artificially *weighted* training set. This means that the algorithm receives the original training set S along with the weights of examples in it. The learning algorithm is expected to treat this weighing as if it is the true distribution of examples in S.

The weight of an example is determined at each iteration according to the hypothesis, h_t, returned by the weak learner. The weight of examples that are correctly classified by h_t is decreased and the weight of those that are misclassified is increased. In this way, the learner is forced to focus on the more difficult examples in the training set. The final hypothesis is then constructed by combining the hypotheses of all rounds giving more weight to hypotheses that had a lower classification error.

Freund and Schapire provide a bound for the training error made by the ensemble classifier built by AdaBoost:

$$\frac{1}{n_s}\sum_{i=1}^{n_s}\mathbf{I}\left(y_i \neq sign\left(f\left(x_i\right)\right)\right) \leq 2^T \prod_{t=1}^{T}\sqrt{\varepsilon_t\left(1-\varepsilon_t\right)}.$$

If it is further assumed that weak learning algorithm always returns a hypothesis whose error is at least a constant away from random guessing (i.e., $\varepsilon_t \leq \frac{1}{2}-\gamma$), this bound turns into:

$$\frac{1}{n_s}\sum_{i=1}^{n_s}\mathbf{I}\left(y_i \neq sign\left(f\left(x_i\right)\right)\right) \leq \exp\left(-2T\gamma^2\right).$$

Therefore, under the above assumption, the training error decays exponentially fast. This quick convergence has also been observed in practice and is one of the most important traits of AdaBoost.

Further experiments have revealed that this analysis is not enough to explain the behaviour of AdaBoost. Typically, the reduction of training error to zero was combined with a generalization[4] bound connecting the training error to the global true error of the classifier (for an example, see Theorem 12 in Chapter 2.3). This means that once a classifier has reached zero training error the bounds have reached their full power in predicting the quality of it. However, in practice, the test-error of the classifiers produced by AdaBoost continue to improve long after the training error has reached zero.

Beginning with [20], a whole new set of generalization bounds was introduced that can explain this behavior. Two of these bounds will be presented in the following chapters as an example. Describing the rest of the generalization bounds and the huge amount of research into and improvements of AdaBoost is far beyond the scope of this work. For further information and an excellent introduction, the reader is referred to [1].

[4]The term generalization refers to the ability of a classifier to perform well in *general* and not only on the training set. Generalization bounds are theorems bounding its global error with terms relating to some measurement on the training set and other terms that are independent of the specific classifier used. Typically these terms measure the complexity of the set of possible classifiers. For more details see Chapter 2.

Chapter 2

The Value of Agreement

In this chapter, a theoretical foundation and justification of the advantage in combing several different learning algorithms will be presented.

A typical approach in the supervised learning model is to design an algorithm that chooses a hypothesis that in some way best fits the training sample. It will be shown that an advantage can be gained by taking several such learning algorithms and demanding that they not only best learn the training set but also ‘agree’ with each other by outputting identical hypotheses[1].

The discussion below involves several learning algorithms and their accompanying hypothesis spaces. To avoid confusion, any enumeration or index that relates to different learners or hypotheses is enumerated using superscripts (typically l). All other indices, such as algorithm iterations and different examples, are denoted using a subscript.

2.1 Preliminaries

Since the learning algorithm is only given a finite sample of examples, it can only select a hypothesis based on limited information. However, the task of the algorithm is a global one. The resulting classifier f must perform well with respect to all examples in $\mathcal{X}$. The probability of error $P(\{(x,y) : f(x) \neq y\})$ must be small. In order to transfer the success of a classifier on the training set to the global case, there exist numerous *generalization bounds* (two such theorems will be given below). Typically these theorems involve some measure of the complexity or richness of the available hypothesis space. In some cases it is assumed that the labels of examples from $\mathcal{X}$ are given by one of the hypotheses in H. In other cases, the algorithm's task is to construct a classifier that best predicts an arbitrary distribution P (for example, in the case of noise). In both cases, if the hypothesis space is not too rich, any hypothesis able to correctly classify the given examples cannot be to far from the target distribution. However, if H is very rich and can classify correctly any finite sample using different functions, success on a finite sample does not necessarily imply good global behavior.

One such hypotheses-space complexity measure, which is particularly useful in the boosting scenario (see Theorem 7) is the *Rademacher Complexity*. A definition from [2] will be given here:

[1]By identical it is meant that the hypotheses represent the same function from $\mathcal{X}$ to $\mathcal{Y}$. In practice the hypotheses will be very different (for example, Bayes classifiers vs. decision trees).

Definition 1. Let $X_1, \ldots, X_n$ be independent samples drawn according to some distribution P on a set $\mathcal{X}$. For a class of functions F, mapping $\mathcal{X}$ to $\mathbb{R}$, define the random variable

$$\hat{R}_n(F) = \mathbf{E}\left[\sup_{f \in F}\left|\frac{2}{n}\sum_{i=1}^{n}\sigma_i f(X_i)\right|\right]$$

where the expectance is taken with respect to $\sigma_1, \ldots, \sigma_n$, independent uniform $\{\pm 1\}$-valued random variables. Then the *Rademacher complexity* of F is $R_n(F) = \mathbf{E}\hat{R}_n(F)$ where the expectance is now taken over $X_1, \ldots, X_n$.

The Rademacher complexity is in fact a measure of how well a hypothesis space is expected to fit an arbitrary labeling. A rich class of functions will do this well and the term $\sup_{f \in F}\left|\frac{2}{n}\sum_{i=1}^{n}\sigma_i f(X_i)\right|$ will be high for many examples. Therefore, the Rademacher complexity of such a function set will be high as well. Smaller and less rich spaces are expected to lead to a smaller $R_n(F)$ value. In the special case where F consists of binary functions, one can show that $R_N(F) = O\left(\sqrt{VCdim(F)/N}\right)$ (derived from [2]), connecting the Rademacher complexity to the well-known VC dimension.

One example of a generalization bound is the following (adapted from Theorem 3 in [1] and proved in [3]):

Theorem 2. *Let F be a class of real-valued functions from $\mathcal{X}$ to $[-1, +1]$ and let $\theta \in [0, 1]$. Let P be a probability distribution on $\mathcal{X} \times \{-1, +1\}$ and suppose that a sample of N examples $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ is generated independently at random according to P. Then for any integer N, with probability at least $1 - \delta$ over samples of length N, every $f \in F$ satisfies*

$$P(y \neq \text{sign}(f(x))) \leq \hat{L}^{\theta}(f) + \frac{2R_N(F)}{\theta} + \sqrt{\frac{\log(2/\delta)}{2N}}$$

where $\hat{L}^{\theta}(f) = \frac{1}{N}\sum_{n=1}^{N}\mathbf{I}(y_n f(x_n) \leq \theta)$ and $\mathbf{I}(y_n f(x_n) \leq \theta) = 1$ if $y_n f(x_n) \leq \theta$ and 0 otherwise.

Theorem 2 introduces a new concept named *margin*.

Definition 3. The *margin* of a function $h : \mathcal{X} \to [-1, 1]$ on an example $x \in \mathcal{X}$ with a label $y \in \{\pm 1\}$ is $yh(x)$.

The margin of a function h on an example x can be interpreted as the confidence of the classification. If the margin is negative, $sign(h)$ misclassifies x. A positive margin measures the distance of $h(x)$ from the crucial 0-value. In these terms, Theorem 2 can be interpreted as saying that a more 'confident' hypotheses has a lower general error. Margins have been used to give a new explanations to the success of boosting algorithms, such as AdaBoost, in decreasing the global error long after a perfect classification of the training examples has been achieved [4]. Typically, one would expect a learning algorithm to eventually over fit the training sample, resulting in an increase in global error.

Theorem 2 represents a rather general type of generalization bounds. Instead of assuming that the labels are generated by one of the hypotheses in H, it gives a connection between the empirical error on samples drawn from *any* distribution and the global expected error. As can be seen, the complexity of H plays a crucial role in this relation. If a small hypothesis space was able to fit well a relatively large sample (resulting in a low L^θ and a large θ), the probability distribution must correlate well with the selected function. However, if H is very rich its success in fitting a finite sample is of lesser importance.

From the above discussion it is clear that if one was able to reduce the hypothesis space H without harming its ability to fit the sampled data, the resulting classifier is expected to have a smaller global error.

2.2 Formal Settings

Let $H^1, \ldots, H^L$ be a set of hypothesis spaces, each with a fitting learning algorithm, A^l. Further suppose that the intersection of all hypothesis spaces is not empty and all learning algorithms are forced to agree and output the same hypothesis. Then effectively the algorithms select a hypothesis from a smaller hypothesis space: $\bigcap_{l=1}^{L} H^l$. If it is further assumed that the hypothesis that best fits the training set belongs to every H^l (thus available to all algorithms), this scheme produces a hypothesis from a potentially much *smaller* hypothesis space which is just as good on the training sample. Hence, the generalization capability of such a hypothesis, as drawn from theorems such as Theorem 2, is potentially much better than the hypotheses outputted from any algorithm operating alone.

While the above discussion would yield the expected theoretical gain, it is very hard to implement. First, demanding that the algorithms output *exactly* the same hypothesis entails an ability that is unlikely to be easily available. Typically the different hypothesis spaces would consist of classifiers as different as neural networks and Bayes classifiers. The need to test (or construct) whether two classifiers are absolutely equal in functionality would render this scheme impractical. Existing algorithms could not be used as is, if at all, and would have to be thoroughly changed. Second, it is also unrealistic to demand that the hypothesis spaces will have an intersection which is rich enough to be useful to correctly classify different target distributions. While this might be feasible for $L = 2$ (such as the assumption in [5]) it is highly unlikely for a bigger number of learners. A more relaxed agreement demand will now be presented, along with a simple way of checking it: unlabeled examples.

Definition 4.

1. Define the *variance* of a vector in $\mathbb{R}^L$ to be:

$$V(y^1, \ldots, y^L) = \frac{1}{L} \sum_{l=1}^{L} \left(y^l\right)^2 - \left(\frac{1}{L} \sum_{l=1}^{L} y^l\right)^2 .$$

2. Furthermore, define the variance of a set of classifiers $f^1, \ldots, f^L$ to be the expected variance on examples from $\mathcal{X}$:

$$V\left(f^1, \ldots, f^L\right) = \mathbf{E} V\left(f^1(x), \ldots, f^L(x)\right) .$$

The variance of a set of classifiers will be used in the following relaxed definition of intersection as a measures of their disagreement:

Definition 5. For any $\nu > 0$, define the ν-*intersection* of a set of hypothesis spaces, $H^1, \dots, H^L$ to be:

$$\nu - \bigcap_{l=1}^{L} H^l = \left\{ f^1, \dots, f^L \, : \forall l,\ f^l \in H^l, \text{ and } V(f^1, \dots, f^L) \le \nu \right\}.$$

In effect the ν-intersection of $H^1, \dots, H^L$ contains all the hypotheses whose difference with some of the members of other hypothesis spaces is hard to discover. This relaxed definition of intersection will be used as the space from which the algorithms can draw their hypotheses. Note that for $\nu = 0$, the 0-intersection is precisely[2] the normal intersection of $H^1, \dots, H^L$.

As mentioned before, unlabeled examples will be used to measure the level of agreement between the various learners. Therefore, let $U = \{u_j\}_{j=1}^{n_u}$ be a set of unlabeled examples, drawn independently from the same distribution P but without the label being available. It will now be shown that if enough unlabeled examples are drawn, the disagreement measured on them is a good representative of the global disagreement. This will be done by defining a new hypothesis space $V(H^1, \dots, H^L)$ and a target distribution $\tilde{P}$ and using a generalization bound resembling Theorem 2.

Definition 6.

1. Let $V\left(H^1, \dots, H^L\right) = \left\{V \circ \left(f^1, \dots, f^L\right) \, : \, f^1 \in H^1, \dots, f^L \in H^L\right\}$ where $V \circ \left(f^1, \dots, f^l\right) : \mathcal{X} \to [0,1]$ is defined by:
$$V \circ \left(f^1, \dots, f^L\right)(x) = V\left(f^1(x), \dots, f^L(x)\right).$$

2. Let $\tilde{P}$ be a probability distribution over $\mathcal{X} \times [0, \infty]$ which is defined by:
$$(\forall A \subseteq \mathcal{X} \times [0,\infty]) \left[\tilde{P}(A) = P\left(\{(x,y) \in \mathcal{X} \times \mathcal{Y} : (x,0) \in A\}\right)\right].$$
In essence, $\tilde{P}$ labels all examples in $\mathcal{X}$ with 0, while giving them same marginal probability as before.

Before we can use the generalizing bound, we need to establish the Rademacher complexity of the new hypothesis space $V(H^1, \dots, H^L)$. This will be done using the following theorem from [2] (Theorem 10) which gives some structural properties of the Rademacher complexity.

Theorem 7. *Let $F, F_1, \dots, F_k$ and H be classes of real functions. Then*

1. *If $F \subseteq H$, $R_n(F) \le R_n(H)$.*
2. *$R_n(F) = R_n(convF)$ where*
$$convF = \left\{ \sum_{t=1}^{T} \alpha_t f_t \, : \forall t\ f_t \in F \text{ and } \alpha_t \ge 0, \sum_{t=1}^{T} \alpha_t = 1 \right\}$$
is the class of all finite convex combinations of functions from F.

[2]This assumes that no element in $\mathcal{X}$ has zero marginal probability. In the case that $\mathcal{X}$ contains non-empty groups of zero probability, the difference between the 0-intersection and true intersection will be never be discovered.

3. *For every $c \in \mathbb{R}$, $R_n(cF) = |c|\, R_n(F)$ where $cF = \{cf : f \in F\}$.*
4. *If $\phi : \mathbb{R} \to \mathbb{R}$ is Lipschitz with a constant L_ϕ and satisfies $\phi(0) = 0$, then $R_n(\phi(F)) \leq 2L_\phi R_n(F)$ where $\phi(F) = \{\phi \circ f : f \in F\}$.*
5. *$R_n\left(\sum_{i=1}^k F_i\right) \leq \sum_{i=1}^k R_n(F_i)$ where $\sum_{i=1}^k F_i = \left\{\sum_{i=1}^k f_i : \forall i\, f_i \in F_i\right\}$.*

Note that clause 2 in Theorem 7 is what makes the Rademacher complexity so attractive in boosting. Since the complexity of all convex combinations of classifiers from a weak hypothesis space is the same as the complexity of the space it self, it results in good generalization bounds.

Corollary 8. $R_n\left(V\left(H^1,\ldots,H^L\right)\right) \leq 8\max_l R_n\left(H^l\right)$.

Proof. The result will follow from the following fact:

$$V\left(H^1,\ldots,H^L\right) \subseteq \frac{1}{L}\sum_{l=1}^{L}\phi\left(H^l\right) + \left[-\phi\left(\frac{1}{L}\sum_{l=1}^{L}H^l\right)\right]$$

where $\phi(z) = z^2$ and is Lipschitz on $\mathcal{Y} \subseteq [-1,+1]$ with $L_\phi = 2$.

Therefore, from Theorem 7

$$\begin{aligned}
R_n\left(V\left(H^1,\ldots,H^L\right)\right) &\leq R_n\left(\frac{1}{L}\sum_{l=1}^{L}\phi\left(H^l\right) + \left[-\phi\left(\frac{1}{L}\sum_{l=1}^{L}H^l\right)\right]\right)\\
&\leq \frac{1}{L}\sum_{l=1}^{L}R_n\left(\phi\left(H^l\right)\right) + R_n\left(\phi\left(\frac{1}{L}\sum_{l=1}^{L}H^l\right)\right)\\
&\leq \frac{1}{L}\sum_{l=1}^{L}2L_\phi R_n\left(H^l\right) + 2L_\phi\frac{1}{L}\sum_{l=1}^{L}R_n\left(H^l\right)\\
&\leq 8\max_l R_n\left(H^l\right).
\end{aligned}$$

□

Before proving the main theorems of the section, the following generalization bound (adapted from [2]) is presented. This theorem allows the use of an arbitrary loss function and does not use the concepts of margins.

Theorem 9. *Consider a loss function $\mathcal{L} : \mathcal{Y} \times \mathbb{R} \to [0,1]$ and let F be a class of functions mapping $\mathcal{X}$ to $\mathcal{Y}$. Let $\{(x_i,y_i)\}_{i=1}^n$ be a sample independently selected according to some probability measure P. Then, for any integer n and any $0 < \delta < 1$, with probability of at least $1-\delta$ over samples of length n, every $f \in F$ satisfies*

$$\mathbf{E}\mathcal{L}(Y, f(X)) \leq \hat{\mathbf{E}}_n\mathcal{L}(Y, f(X)) + R_n\left(\tilde{L} \circ F\right) + \sqrt{\frac{8\log(2/\delta)}{n}}$$

where $\hat{\mathbf{E}}_n$ is the expectance measured on the samples and

$$\tilde{L} \circ F = \{(x,y) \mapsto \mathcal{L}(y, f(x)) - \mathcal{L}(y,0) : f \in F\}.$$

The scene is now set to give the first of the two main theorems of this chapter—a connection between function's agreement on a finite sample set and their true disagreement:

Theorem 10. *Let $H^1, \ldots, H^L$ be sets of functions from $\mathcal{X}$ to $\mathcal{Y}$ and let $U = \left\{u_j\right\}_{j=1}^{n}$ be a set of unlabeled examples drawn independently according to a distribution P over $\mathcal{X} \times \mathcal{Y}$. Then for any integer n and $0 < \delta < 1$, with probability of at least $1-\delta$ every set of functions $f^l \in H^l$, $l = 1 \ldots L$ satisfies:*

$$V(f^1, \ldots, f^L) \leq \hat{V}(f^1, \ldots, f^L) + 8 \max_l R_n(H^l) + \sqrt{\frac{8 \log(2/\delta)}{n}}$$

where $\hat{V}(f^1, \ldots, f^L)$ is the sampled expected variance, as measured on $U = \left\{u_j\right\}_{j=1}^{n}$.

Proof. The theorem follows directly from Theorem 9 when applied to the function set $V(H^1, \ldots, H^L)$ with $\tilde{P}$ as target distribution. The loss function is defined by $\mathcal{L}(y,z) = \min\{|y-z|, 1\}$. Since $\tilde{P}$ assigns zero probability to all non-zero labels,

$$\mathbf{E}\mathcal{L}\left(Y, V\left(f^1(X), \ldots, f^L(X)\right)\right) = \mathbf{E}V\left(f^1(X), \ldots, f^L(X)\right) = V(f^1, \ldots, f^L)$$

and

$$\hat{\mathbf{E}}\mathcal{L}\left(Y, V\left(f^1(X), \ldots, f^L(X)\right)\right) = \hat{V}(f^1, \ldots, f^L).$$

Furthermore $R_n\left(\tilde{L} \circ F\right) = R_n\left(V(H^1, \ldots, H^L)\right)$ since

$$\tilde{P}\left(\mathcal{L}\left(y, V \circ (f^1, \ldots, f^L)(x)\right) - \mathcal{L}(y, 0) \neq V \circ (f^1, \ldots, f^L)(x)\right) = 0.$$

Therefore, Theorem 9 implies that, with a probability of at least $1-\delta$,

$$\begin{aligned} V(f^1, \ldots, f^L) &\leq \hat{V}(f^1, \ldots, f^L) + R_n\left(V(H^1, \ldots, H^L)\right) + \sqrt{\frac{8 \log(2/\delta)}{n}} \\ &\leq \hat{V}(f^1, \ldots, f^L) + 8 \max_l R_n(H^l) + \sqrt{\frac{8 \log(2/\delta)}{n}} \end{aligned}$$

where the last inequality holds due to the corollary of Theorem 7. □

Theorem 10 allows us to use a finite set of unlabeled examples to make sure (with high probability) that the classifiers selected by the learning algorithms are indeed in the desired ν-intersection of the hypothesis spaces. This allows us to adapt generalization bounds to use smaller hypothesis spaces. As an example, an adapted version of Theorem 2 is presented.

Theorem 11. *Let $H^1, \ldots, H^L$ be a class of real-valued functions from $\mathcal{X}$ to $[-1, +1]$ and let $\theta \in [0, 1]$. Let P be a probability distribution on $\mathcal{X} \times \{-1, +1\}$ and suppose that a sample of n_s labeled examples $S = \left\{(x_j, y_j)\right\}_{j=1}^{n_s}$ and n_u unlabeled examples $U = \left\{u_j\right\}_{j=1}^{n_u}$ is generated independently at random according to P. Then for any integer n_s, $\nu > 0$, $0 < \delta < 1$ and n_u such that $8 \max_l R_{n_u}(H^l) + \sqrt{\frac{8 \log(4/\delta)}{n_u}} \leq \frac{\nu}{2}$, with a probability at least $1-\delta$, every $f^1 \in H^1, \ldots, f^L \in H^L$ whose disagreement $\hat{V}$ on U is at most $\frac{\nu}{2}$ satisfies*

$$\forall l \, P(y \neq \operatorname{sign}(f^l(x))) \leq \hat{L}^\theta(f^l) + \frac{2R_{n_s}(\nu - \bigcap_{\hat{l}} H^{\hat{l}})}{\theta} + \sqrt{\frac{\log(4/\delta)}{2n_s}}$$

where $\hat{L}^\theta(f^l) = \frac{1}{n_s} \sum_{j=1}^{n_s} \mathbf{I}\left(y_i f^l(x_i) \leq \theta\right)$.

Proof. From Theorem 10, with a probability of at least $1-\frac{\delta}{2}$, $V(f^1,\dots f^L) \leq \nu$ and therefore $\forall l \;\; f^l \in \nu - \bigcap_{\hat{l}} H^{\hat{l}}$. Applying Theorem 2 to $\nu - \bigcap_{\hat{l}} H^{\hat{l}}$, with a probability of at least $1-\frac{\delta}{2}$,

$$\forall f \in \nu - \bigcap_{\hat{l}} H^{\hat{l}} P(y \neq \operatorname{sign}(f(x))) \leq \hat{L}^{\theta}(f) + \frac{2R_{n_s}(\nu - \bigcap_{\hat{l}} H^{\hat{l}})}{\theta} + \sqrt{\frac{\log(4/\delta)}{2n_s}}$$

using the union bound and combining the results proves the theorem. □

To conclude this section, we note that the proposed settings has the following desired property: it doesn't help to have duplicate copies of the same hypothesis space. To have any advantage, $\nu - \bigcap_{\hat{l}} H^{\hat{l}}$ must be considerably smaller then any of the base hypothesis spaces. Therefore, using only duplicate copies of the same hypothesis space $H = H^1,\dots H^L$ gives $\nu - \bigcap_{\hat{l}} H^{\hat{l}} = H$ and hence no improvement. Furthermore, any duplicates within the set of different hypothesis spaces can be removed without changing the results.

2.3 Reduction of labeled examples

The previous section presented a formal setting where agreement was used to reduce the complexity of the set of possible hypotheses. The immediate implication is that training error serves as a better approximation for global true error. Therefore, for a given number of labeled examples, if the learning algorithm has produced a classifier with a low training error one can expect a lower global error. However this reduction in complexity can be also viewed from a different, though very related, point of view.

Since most algorithms can reduce the training error to a very low level, the number of labeled examples necessary is usually determined as to decrease the two other terms in generalization bounds: the complexity of the hypothesis space and the certainty in the success of the whole procedure (δ). Using more labeled examples typically allows using a lower δ value without hindering the expected error of the resulting classifier (for example, Theorem 2 involves a $\sqrt{\frac{\log(2/\delta)}{2N}}$ term). The second result of increasing the number of labeled examples is reduction in the Rademacher complexity (or similar complexity terms). Therefore, decreasing the term relating to hypothesis space complexity, enables one to use *less* labeled examples while achieving the same bound.

To illustrate this consider Blumer et al. result [6] concerning the simple case of consistent learners:

Theorem 12. *(adapted from 2.1.ii in [6]) For any $\varepsilon > 0$, $\delta > 0$, a probability distribution P over X and a hypothesis space $H \subseteq \{-1,+1\}^X$ of finite VC dimension, d. Let $\hat{f} \in H$ be some target function and $S = \left\{\left(x_j, \hat{f}(x_j)\right)\right\}_{j=1}^{n_s}$ a sample independently drawn from X where n_s is larger than $\max\left\{\frac{4}{\varepsilon}\log\frac{2}{\delta}, \frac{8d}{\varepsilon}\log\frac{13}{\varepsilon}\right\}$. Then with probability of at least $1-\delta$ over samples of size n_s, for any function $f \in H$ which correctly classify all sampled examples, $P(f(x) \neq \hat{f}(x)) < \varepsilon$.*

To prove the theorem, Blumer et al. showed that with high probability, a sample of size $\max\left\{\frac{4}{\varepsilon}\log\frac{2}{\delta}, \frac{8d}{\varepsilon}\log\frac{13}{\varepsilon}\right\}$ is sufficient to disqualify any function in H that is too 'far' from the target $\hat{f}$. If H is made smaller, the number of functions which need to be excluded is reduced. Therefore, less labeled examples are needed in order to exclude high error functions.

In many learning applications one often comes across a phenomena called over-fitting. In these cases, the learning algorithm produces a classifier which is overly trained on the training sample. The classifier is too specialized in classifying the learning sample and looses its generalization powers. In other words, the algorithm learns information that is accidentally available in the training sample but is specific to it. For an iterative learning algorithm (improving its current classifier with every iteration), one typically observes a reduction in both training and test (global) error in the beginning of the algorithm's run. However, from some point on, the test error will start to increase, while the training error is still being reduced[21, 22].

Generalization bounds such as those presented above typically deal with over-fitting using the following idea: if the algorithm is given enough labeled examples it will not over-fit. Since the training sample is representative enough of of target function, specializing in it does no harm. In the extreme, this leads to theorems such as the one of Blumer et al. concerning consistent learning algorithms. In the setting proposed here, the learning algorithm needs not only fit its training data but also agree with a couple of other algorithms. If the algorithms are sufficiently different, forcing them to agree inhibits their specialization on the training data, allowing to use a less representative training sample, or less labeled examples.

Chapter 3

The Algorithm

In this chapter, we propose a new boosting algorithm named AgreementBoost (Algorithm 3), which exploits the benefits suggested by the theory presented in the previous chapter. Like AdaBoost, the algorithm is designed to operate in Boolean scenarios where each example can belong to one of two possible classes denoted by ± 1 (i.e., $\mathcal{Y} = \{+1, -1\}$).

As in many boosting algorithms, AgreementBoost creates combined classifiers or ensembles. However, instead of just one such classifier, AgreementBoost creates L ensembles, one for each hypothesis space. The ensembles are constructed using L underlying learning algorithms, one for each of the L hypothesis spaces $\left\{H^l\right\}_{l=1}^{L}$. At each iteration, one of the learning algorithms is presented with a weighing of both labeled and unlabeled examples in the form of a weight vector $w(x)$ and pseudo-labels for the unlabeled examples ($y(u)$). The underlying learner is then expected to return a hypothesis f_t^l with a near-optimal[1] edge[2]: $\gamma = \sum_{(x_j, y_j) \in S} w(x_j) y_j f^l(x_j) + \sum_{u_j \in U} w(u_j) y(u_j) f^l(u_j)$. The returned hypothesis is then added to the corresponding ensemble. The weight given to f_t^l (α_t^l, selected at step 2.a.iii) is chosen such that the cost function F is minimized. This can be done by any numeric line minimization algorithm.

The proposed AgreementBoost can be described as a particular instance of Any-Boost [21], a boosting algorithm allowing for arbitrary cost functions. Agreement-Boost's cost function F has been chosen to incorporate the ensembles' disagreement into the normal margin terms. This is achieved using a weighted sum of two terms: an error or margin-related term ($\sum_{l=1}^{L} \sum_{j=1}^{n_s} er\left(-y_j g^l(x_j)\right)$) and a disagreement term $\sum_{j=1}^{n_u} er\left(V\left(u_j\right)\right)$. Despite of the fact that the these terms capture different notions, they are very similar. Both terms use the same underlying function, $er(x)$, to assign a cost to some example-related measure: The first penalizes low (negative) margins while the second condemns high variance (and hence disagreement). AgreementBoost allows choosing any function as $er(x)$, so long as it is convex and strictly increasing. This freedom allows using different cost schemes and thus for future cost function analysis (as done, for example, in [21]). In the degenerate case where no unlabeled examples are used ($n_u = 0$) and e^x is used as $er(x)$, AgreementBoost is equivalent to L independent

[1]For the exact definition of 'near-optimal', see Chapter 4.

[2]Note that the edge can be rewritten in terms of the weighted error of $f^l : \gamma = 1 - 2\varepsilon$ where $\varepsilon = \sum_{(x_j, y_j) \in S} w(x_j) \mathbf{I}\left[y_j \neq f^l(x_j)\right] + \sum_{u_j \in U} w(u_j) \mathbf{I}\left[y(u_j) \neq f^l(u_j)\right]$. Therefore f^l having a near-optimal edge is equivalent to it having a near-minimal weighted error.

Algorithm 3 Agreement Boost

Denote $F\left(g^1,\ldots,g^L\right)=\sum_{l=1}^{L}\sum_{j=1}^{n_s} er\left(-y_j g^l(x_j)\right)+\eta L\sum_{j=1}^{n_u} er\left(V\left(u_j\right)\right)$ where $V(u)=\frac{1}{L}\sum\limits_{l=1}^{L} g^l(u)^2-\left[\frac{1}{L}\sum\limits_{l=1}^{L} g^l(u)\right]^2$, $\eta\in\mathbb{R}^+$ is some positive real number and $er:\mathbb{R}\to\mathbb{R}$ is some convex, strictly increasing function with continuous second derivative.

```
1. Set $g^l \equiv 0$ for $l = 1\ldots L$.
2. Iterate until done (counter $t$):
    (a) Iterate over $l = 1\ldots L$:
        i. Set $w(x_j) = er'\left(-y_j g^l(x_j)\right) y_j / Z$ for all $(x_j, y_j) \in S$ and
           $w(u_j) = 2\eta \left|\frac{1}{L}\sum_{\hat{l}=1}^{L} g^{\hat{l}}(u_j) - g^l(u_j)\right| er'\left(V(u_j)\right)/Z$ for all $u_j \in U$
           where $Z$ is a renormalization factor s.t. $\sum\limits_{x_j} w(x_j) + \sum\limits_{u_j} w(u_j) = 1$.
           Use $y(u_j) = sign\left(\frac{1}{L}\sum_{\hat{l}=1}^{L} g^{\hat{l}}(u_j) - g^l(u_j)\right)$ as pseudo-labels for $u_j$.
        ii. Receive hypothesis $f_t^l$ from learner $l$ using the above weights and labels.
        iii. Find $\alpha_t^l \geq 0$ that minimizes $F\left(g^1,\ldots,g^l + \alpha_t^l f_t^l,\ldots,g^L\right)$.
        iv. Set $g^l = g^l + \alpha_t^l f_t^l$.
3. Output classifier $sign(g^l)$ whose error on the samples is minimal out of the $L$ classifiers.
```

runs of AdaBoost (using the L underlying learners).

Chapter 4

Proof of convergence

In this chapter a convergence proof for Algorithm 3 is given. The proof considers two scenarios. The first assumes that the intersection of all $conv\left(H^l\right)$ is able to correctly classify all labeled examples using classifiers which agree on all unlabeled examples. Under this assumption, it is shown that the algorithm will produce classifiers, which in the limit are fully correct and agree on all unlabeled examples. In other cases, where this assumption is not valid, the algorithm will produce ensembles which minimize a function representing a compromise between correctness and agreement.

Both Mason et al. [21] and Rätsch et al. [24] provide similar convergence proofs for AnyBoost-like algorithms. While both proofs can be used (with minor modifications) in our settings, they do not fully cover both scenarios. The proof in [24] demands that the sum of the α_t^l coefficients will be bounded and thus cannot be used in cases where the theoretical assumptions hold. This can be seen easily in the case of AdaBoost, where a fully correct hypothesis will be assigned an infinite weight. While Agreement-Boost will never assign an infinite weight to a hypotheses (due to the disagreement term), it is easy to come up with a similar scenario where the coefficient sum grows to infinity. In [21], Mason et al. present a theorem very similar to Theorem 18 below. However, they assume that the underlying learner performs perfectly and always returns the *best* hypothesis from the hypothesis space. Such a severe assumption is not needed in the proof presented here. Furthermore, due to the generality of AnyBoost, the result in [21] apply to the cost function alone and is not translated back to training error terms.

The proofs below are based on two assumptions concerning the learning algorithms and the hypothesis spaces. It is assumed that when presented with an example set[1] S and a weighing $w(x)$, the underlying learning algorithms return a hypothesis f^l for which:

$$\sum_{\left(x_j,y_j\right)\in S} w(x_j)y_j f^l(x_i) \geq \delta \max_{\hat{f}\in H^l}\left(\sum_{x_j\in S} w(x_j)y_j\hat{f}(x_i)\right) \text{ for some } \delta > 0.$$

The second assumption concerns the hypothesis spaces: it is assumed[2] that for every l and some $f^l \in H^l$ the negation of f^l is also in H^l i.e.: $f \in H^l \Rightarrow -f \in H^l$. This allows

[1] Note that this may include examples labeled by the algorithm, rather the being drawn according to the underlying distribution P.

[2] Note that this is assumed for simplicity alone. By allowing the algorithm to use negative values for α_t^l all hypothesis spaces become $H^l \cup -H^l$, which are closed under negation. This inflicts no increase in the Rademacher complexity of the spaces. To keep the assumption with respect to the hypothesis returned by the

us to using absolute value in the previous assumption:

$$\sum_{x_j \in S} w(x_j) y_j f^l(x_j) \geq \delta \max_{\hat{f} \in H^l} \left| \sum_{x_j \in S} w(x_j) y_j \hat{f}(x_i) \right| \text{ for some } \delta > 0.$$

In the Lemmas and Theorems to follow, it will sometimes be assumed that the hypothesis spaces are finite. Due to the fact that there is only finite amount of ways to classify a finite set of examples with a ± 1 label, if some of the hypothesis spaces are infinite it will be indistinguishable when restricted to S and U. Therefore, without loss of generality, one can assume that the number of hypotheses is finite.

The convergence of the algorithm is proven taking a different point of view to the ensembles built by the algorithm. The ensembles can be seen as a mix of all possible functions in the hypothesis spaces rather then as an accumulation of hypotheses:

Definition 13.

1. Let $H^l = \left\{ f_i^l \right\}_{i \in I_l}$ be an enumeration of functions in H^l. One can rewrite the ensembles g^l built by AgreementBoost as functions from $X \times \mathbb{R}^{|H^l|}$ to $\mathbb{R}$: $g^l(x, \beta^l) = \sum_i \beta_i^l f_i^l(x)$ for $\beta^l = \left(\beta_1^l, \beta_2^l, \dots\right) \in \mathbb{R}^{|H^l|}$ and $l = 1 \dots L$. Further denote $\beta = \left(\beta^1, \dots, \beta^L\right)$. Note that β_i^l is the sum of all α_t^l chosen by the algorithm such that $f_t^l \equiv f_i^l$.

2. Let the *variance* of $g^1, \dots, g^L$ on an example u be

$$V(u, \beta) = \frac{1}{L} \sum_{l=1}^{L} g^l(u, \beta^l)^2 - \left[\frac{1}{L} \sum_{l=1}^{L} g^l(u, \beta^l) \right]^2.$$

3. Whenever it is clear from context what are the β parameters, $V(u)$ and $g^l(u)$ will be used for brevity.

4. Let $er : \mathbb{R} \to \mathbb{R}^+$ be a convex monotonically increasing function. Denote by

$$\begin{aligned} F(\beta) &= E(\beta) + \eta D(\beta) \\ E(\beta) &= \sum_{l=1}^{L} \sum_{j=1}^{n_S} er\left(-y_j g^l(x_j)\right) \\ D(\beta) &= L \sum_{j=1}^{n_u} er\left(V\left(u_j\right)\right) \end{aligned} \tag{4.1}$$

for some $\eta > 0$.

$F(\beta)$ represents a weighing between correctness and disagreement. $E(\beta)$, being a sum of loss functions penalizing negative margins, relates to the current error of the ensemble classifiers. $D(\beta)$ captures the ensembles' disagreement over the unlabeled examples.

Using the above notations and the new point of view, the margins of hypotheses becomes proportional to the partial derivative of $F(\beta)$ with respect to the corresponding

underlying learning algorithm, one has to present two queries: the original query and a query where all labels are negated. Choosing the the hypothesis with a higher margin of the two results is the desired hypothesis.

coefficient. Replacing the examples' weight and labels according to the definition of AgreementBoost, we have that:

$$\begin{aligned}
&\sum_{x_j\in S} w(x_j) y_j f_i^l(x_j) + \sum_{u_j\in U} w(u) y(u_j) f_i^l(u_j) = \\
&\quad \tfrac{1}{Z}\sum_{x_j\in S} er'\left(-y_j g^l(x_j)\right) y_j f_i^l(x_j) + \\
&\qquad \tfrac{2\eta}{Z}\sum_{u_j\in U} er'\left(V(u_j)\right)\left(\tfrac{1}{L}\textstyle\sum_{\hat{l}=1}^{L} g^{\hat{l}}(u_j) - g^l(u_j)\right) f_i^l(u_j) \\
&\qquad\qquad = -\tfrac{1}{Z}\tfrac{\partial F}{\partial \beta_i^l}(\beta.)
\end{aligned}$$

Therefore the underlying learners return hypotheses whose corresponding partial derivatives maintain the following inequality:

$$-\frac{\partial F}{\partial \beta_i^l}(\beta) \geq \delta \max_{\hat{i}} -\frac{\partial F}{\partial \beta_{\hat{i}}^l}(\beta).$$

Furthermore, since the hypothesis spaces are assumed to be closed under negation, the following holds as well:

$$-\frac{\partial F}{\partial \beta_i^l}(\beta) \geq \delta \max_{\hat{i}} -\frac{\partial F}{\partial \beta_{\hat{i}}^l}(\beta) = \delta \max_{\hat{i}} \left|\frac{\partial F}{\partial \beta_{\hat{i}}^l}(\beta)\right|.$$

Note that this ensures that the partial derivative with respect to the returned function coefficient is non-positive and hence the choice of α_t^l in step 2.a.ii of Algorithm 3 is in fact the global optimum[3] over all $\mathbb{R}$. Since in every iteration only one coefficient is changed to a value which minimizes $F(\beta)$, Algorithm 3 is equivalent to a coordinate descent minimization algorithm (for more information about minimization algorithms see, for example, [9]).

As a last preparation before the convergence proof, it will be shown that $F(\beta)$ is convex. Apart from having other technical advantages, this guaranties that the algorithm will not get stuck in a local minimum:

Claim 14.

1. $\forall u \in U$, $er(V(u,\beta))$ is convex with respect to β.
2. $\forall x_j \in X$, $er(-y_j g^l(x_j,\beta^l))$ is convex with respect to β^l.

Proof. The claim will follow from the following facts:

1. The operator $g(\beta) = \left(g^1(\beta^1),\ldots,g^L(\beta^L)\right)$ is linear. Therefore, for every β, $\hat{\beta}$ and $\forall \alpha,\gamma \in [0,1]$, we have that $g(\alpha\beta + \gamma\hat{\beta}) = \alpha g(\beta) + \gamma g(\hat{\beta})$.
2. Let $A, B \in \mathbb{R}^L$ be two vectors and $\alpha,\gamma \in [0,1]$ two numbers such that $\alpha+\gamma = 1$. Further denote the variance of a vector $v = (v_1,\ldots,v_L)$ by $V(v) = \frac{1}{L}\sum_{i=1}^{L} v_i^2 - \frac{1}{L^2}\left(\sum_{i=1}^{L} v_i\right)^2$. Then the following holds:

$$\begin{aligned}
V(\alpha A + \gamma B) &= \alpha^2 V(A) + \gamma^2 V(B) + 2\alpha\gamma cov(A,B) \\
&\leq \alpha^2 V(A) + \gamma^2 V(B) + \alpha\gamma(V(A) + V(B)) \\
&= \alpha V(A) + \gamma V(B)
\end{aligned}$$

[3]This involves the convexity of $F(\beta)$ that will be discussed below.

where $cov(A,B)$ is the covariance of the two vectors and is defined in a similar fashion to $V(v)$.

3. $\forall c \in \mathbb{R}$ the function $er(cx)$ is convex with respect to x.
4. er is monotonically increasing.

Therefore, for every β, $\hat{\beta}$ and $\forall \alpha, \gamma \in [0,1]$ such that $\alpha + \gamma = 1$, using fact 1 and fact 2 above, we have that $V(u, \alpha\beta + \gamma\hat{\beta}) \leq \alpha V(u,\beta) + \gamma V(u,\hat{\beta})$. Using fact no. 4 and er's convexity, we have that

$$er\left(V(u,\alpha\beta+\gamma\hat{\beta})\right) \leq er\left(\alpha V(u,\beta)+\gamma V(u,\hat{\beta})\right) \leq \alpha er\left(V(u,\beta)\right)+\gamma er\left(V(u,\hat{\beta})\right)$$

proving the first part of the claim.

For the second part, Facts 1 and 3 are used together, giving that

$$\begin{aligned} er(-y_j g^l(x_j, \alpha\beta^l + \gamma\hat{\beta}^l)) &= er\left(-y_j\left[\alpha g^l(x_j,\beta^l)+\gamma g^l(x_j,\hat{\beta}^l)\right]\right) \\ &\leq \alpha er\left(-y_j g^l(x_j,\beta^l)\right)+\gamma er\left(-y_j g^l(x_j,\hat{\beta}^l)\right) \end{aligned}$$

proving the second part of the claim. □

Lemma 15. *The function $F(\beta)$ is convex with respect to β.*

Proof. This follows immediately from Claim 14 and the fact that a sum of convex functions is convex. □

Lemma 16. *Let $\{\beta_n\}$ be a sequence of points generated by an iterative linear search algorithm A, i.e., $\beta_{n+1} = A(\beta_n)$ minimizing a non-negative convex function*[4] *$F \in C^2$. Denote the direction in which the algorithm minimizes F in every step by $v_n = \frac{\beta_{n+1}-\beta_n}{\|\beta_{n+1}-\beta_n\|_\infty}$ and $F_n(\alpha) = F(\beta_n + \alpha v_n)$ (i.e., A minimizes $F_n(\alpha)$ in every iteration by a linear search). Then, if $\exists M \in \mathbb{R}^+$ such that $\forall n$ $\frac{d^2F_n}{d\alpha^2}(\alpha) \leq M$ for every 'feasible' α (i.e., when $F_n(\alpha) \leq F(\beta_n)$) then $\lim\limits_{n\to\infty} \frac{dF_n}{d\alpha}(0) = 0$.*

Proof. By Taylor expansion $F_n(\alpha) = F_n(0) + F_n'(0)\alpha + \frac{F_n''(\xi)}{2}\alpha^2$ for some ξ between 0 and α. Therefore $F_n(0) - F_n(\alpha) = -F_n'(0)\alpha - \frac{F_n''(\xi)}{2}\alpha^2 \geq -F_n'(0)\alpha - \frac{1}{2}M\alpha^2$ since by assumption $\frac{d^2F_n}{d\alpha^2}(\xi) \leq M$ (by the convexity of F, if α is feasible, all points between it and 0 are also feasible). From the previous inequalities it follows that

$$\begin{aligned} F(\beta_n) - F(\beta_{n+1}) &= F(\beta_n) - \min_{feasible\,\alpha} F_n(\alpha) = \max_{feasible\,\alpha}(F_n(0)-F_n(\alpha)) \\ &\geq \max_{feasible\,\alpha}\left(-F_n'(0)\alpha - \frac{1}{2}M\alpha^2\right) \\ &= \max_{\alpha\in\mathbb{R}}\left(-F_n'(0)\alpha - \frac{1}{2}M\alpha^2\right) = \frac{F_n'(0)^2}{2M} \geq 0. \end{aligned}$$

Note that the last inequality ensures that the last expression is maximized by a feasible α, making the equality argument valid.

Now, since $F(\beta_n)$ is a monotonically decreasing sequence and is bounded from below, it must converge. Therefore $\lim\limits_{n\to\infty}(F(\beta_n) - F(\beta_{n+1})) = 0$, which together with the above inequality, gives the result. □

[4] By C^2 we denote the set of functions having a continuous second derivative.

Lemma 17. *Let $\{\beta_n\}$ be a sequence of points generated by an iterative linear search algorithm A (i.e., $\beta_{n+1} = A(\beta_n)$) minimizing a non-negative convex function $F \in C^2$. If in addition to the conditions of Lemma 16 (and using the same notations),$\exists m > 0 \in \mathbb{R}$ such that $\forall n \frac{d^2F_n}{d\alpha^2}(\alpha) \geq m$ for every 'feasible' α (i.e., when $F_n(\alpha) \leq F(\beta_n)$) then $\lim\limits_{n\to\infty} \|\beta_{n+1} - \beta_n\|_\infty = 0$.*

Proof. The proof is similar to the one of Lemma 16.
By Taylor expansion $F_n(\alpha) = F_n(0) + F_n'(0)\alpha + \frac{F_n''(\xi)}{2}\alpha^2$ for some ξ between 0 and α. Therefore $F_n(0) - F_n(\alpha) = -F_n'(0)\alpha - \frac{F_n''(\xi)}{2}\alpha^2 \leq -F_n'(0)\alpha - \frac{1}{2}m\alpha^2$ since by assumption $\frac{d^2F_n}{d\alpha^2}(\xi) \geq m$. It follows that for the α chosen at step n, $|\alpha|$ must be smaller or equal to $\left|\frac{2F_n'(0)}{m}\right|$. By Lemma 16 $\lim\limits_{n\to\infty} F_n'(0) = 0$, and therefore $\|\beta_{n+1} - \beta_n\|_\infty = \|\alpha v_n\|_\infty = |\alpha| \longrightarrow 0$. □

Theorem 18. *For some non-empty sets of labeled examples S and unlabeled examples U, suppose that the underlying learners are guaranteed to return a hypothesis $\hat{f}$ such that $\sum\limits_x w(x) y_x \hat{f}(x) \geq \delta\left(\max\limits_f \left|\sum\limits_x w(x) y_x f(x)\right|\right)$ for some constant $\delta > 0$ and every weighing $w(x)$ of their examples. Further let $er : \mathbb{R} \to \mathbb{R}^+$ be a non constant convex monotonically increasing function such that:*

1. *$er \in C^2$ and $er'(0) > 0$.*
2. *$\exists M \in \mathbb{R}^+$ for which $er(x) \leq \max\left\{L\left(|S| + \eta\,|U|\right)er(0), \frac{1}{\eta}\left(|S| + \eta\,|U|\right)er(0)\right\}$ implies that $er''(x) < M$.*

Then it holds that $\lim\limits_{n\to\infty} \|\nabla F(\beta_n)\|_\infty = 0$.

Proof. Since Algorithm 3 is equivalent to performing a coordinate descent which minimizes

$$F(\beta) = \sum_l \sum_{x_i} er(-y_i g^l(x_i)) + \eta L \sum_{u_j} er(V(u_j))$$

the result will follow from Lemma 16 and Lemma 17.

Using the same notation as in Lemma 16, one must show that $\forall n \frac{d^2F_n}{d\alpha^2}(\alpha) \leq M'$ for every 'feasible' α and some $M' \in \mathbb{R}^+$. Since Algorithm 3 only changes one coordinate at a time, it is enough to show that the second derivative with respect to each coordinate is bounded.

$$\begin{aligned} \frac{\partial^2 F}{\partial\beta_i^l \partial\beta_i^l} &= \sum_{x_j \in S} er''\left(-y_j g^l(x_j)\right)\left[y_j f_i^l(x_j)\right]^2 \\ &\quad + 4\frac{\eta}{L} \sum_{u_j \in U} er''\left(V(u_j)\right)\left[\frac{1}{L}\sum_{\hat{l}} g^{\hat{l}}(u_j) - g^l(u_j)\right]^2 f_i^l(u_j)^2 \qquad (4.2) \\ &\quad + 2\eta \sum_{u_j \in U} er'\left(V(u_j)\right)\frac{L-1}{L} f_i^l(u_j)^2. \end{aligned}$$

Denote $L_\eta = \max\left\{L, \frac{1}{\eta}\right\}$ and $\xi = \sup\left\{x : er(x) \leq L_\eta\left(|S| + \eta\,|U|\right)er(0)\right\}$ (note that er convexity and the fact that it is not a constant function imply that $\lim\limits_{x\to\infty} er(x) = \infty$ and therefore ξ is finite). Now, from the convexity of $er(x)$ it follows that $er'(x)$ is monotonically increasing. Hence it holds that $er(x) \leq L_\eta\left(|S|+\eta\,|U|\right)er(0)$ implies $er'(x) \leq er'(\xi)$. Therefore, without loss of generality, we can assume that $er(x) \leq L_\eta\left(|S|+\eta\,|U|\right)er(0)$ implies that both $er'(x)$ and $er''(x)$ are bounded by M.

Let $\{\beta_n\}$ be the sequence of coefficient vectors generated by Algorithm 3. Since β_0 is the all zero vector, $F(\beta_0) = L\left(|S|+\eta\,|U|\right)er(0)$. Therefore for every 'feasible' β which the algorithm might consider, it must hold that $F(\beta) \leq F(\beta_0)$. Since $(\forall x)\,[er(x) \geq 0]$, this implies that

$$(\forall l,j)\left[er(-y_j g^l(x_j)) \leq L\left(|S|+\eta\,|U|\right)er(0)\right]$$

and

$$(\forall j)\left[er(V(u_j)) \leq \frac{1}{\eta}\left(|S|+\eta\,|U|\right)er(0)\right].$$

Therefore,

$$\left[\frac{1}{L}\sum_{\hat{l}} g^{\hat{l}}(u_j) - g^l(u_j)\right]^2 \leq LV(u) \leq L\xi. \tag{4.3}$$

From which it follows that for every feasible β:

$$\begin{aligned}\frac{\partial^2 F}{\partial\beta_i^l\partial\beta_i^l}(\beta) \quad &\leq \quad \sum_{x_j\in S} M\left[y_j f_i^l(x_j)\right]^2 \\ &\qquad + 4\frac{\eta}{L}\sum_{u_j\in U} ML\xi f_i^l(x)^2 \\ &\qquad + 2\eta\sum_{u_j\in U} M\frac{L-1}{L} f_i^l(x)^2 \\ &\leq \quad M\left(|S|+2\eta\,|U|\,(2\xi+1)\right),\end{aligned} \tag{4.4}$$

where the last inequality follows from the fact that $y_j, f_i^l(x_j) \in \{-1,1\}$. This completes the necessary requirements of Lemma 16 and therefore $\lim_{n\to\infty}\frac{\partial F}{\partial\beta_{i_n}^{l_n}} = 0$ where l_n and i_n are the indices of the hypothesis returned by the underlying learner at iteration n. By the assumptions regarding the base learners and the hypothesis spaces

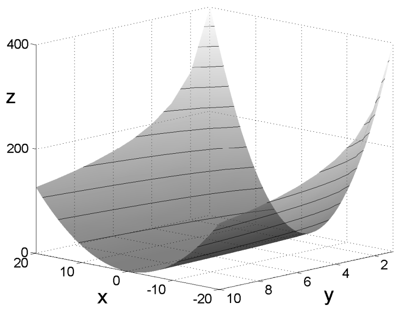
$$-\frac{\partial F}{\partial\beta_{i_n}^{l_n}} \geq \delta\max_i -\frac{\partial F}{\partial\beta_i^{l_n}} = \delta\max_i\left|\frac{\partial F}{\partial\beta_i^{l_n}}\right|$$

giving that $\forall i \in I_{l_n}\ \lim_{n\to\infty}\left|\frac{\partial F}{\partial\beta_i^{l_n}}(\beta_n)\right| = 0$.

Since in Algorithm 3, $l_n = (n \mod L)+1$, in order to complete the proof it is enough to show that $\forall\hat{l} = 1\ldots L$, $\forall i \in I_{\widehat{l_n}}\ \lim_{n\to\infty}\left|\frac{\partial F}{\partial\beta_i^{\widehat{l_n}}}(\beta_n)\right| = 0$ where $\widehat{l_n} = \left(n+\hat{l} \mod L\right)+1$. Using the above as base for an induction argument (for $\hat{l} = L$), it is enough to show that if the claim holds for $\hat{l}+1$, it also holds for $\hat{l}$. Let i be some index in $I_{\widehat{l_n}}$. By the inductive assumption, it holds that

$$\lim_{n\to\infty}\left|\frac{\partial F}{\partial\beta_i^{\widehat{l_n}}}(\beta_{n-1})\right| = 0 \text{ (note that } \widehat{l_n} = \left(n-1+\hat{l}+1 \mod L\right)+1).$$

Figure 4.1: $z = \frac{1}{\sqrt{y}}x^2 + \frac{1}{y}$ and its level curves



It is therefore enough to show that $\left|\frac{\partial F}{\partial \beta_i^{\widehat{l_n}}}(\beta_n) - \frac{\partial F}{\partial \beta_i^{\widehat{l_n}}}(\beta_{n-1})\right|$ also converges to 0 as n grows to infinity.

Using the bound in Equation 4.3 which is valid for all l, it holds that

$$\forall l \left|\frac{1}{L}\sum_{\hat{l}} g^{\hat{l}}(u_j) - g^l(u_j)\right| \le \sqrt{L\xi}$$

for all 'feasible' β and therefore for all β between β_{n-1} and β_n. Denoting again by i_{n-1} the function index returned by learner l_{n-1} in iteration $n-1$, one can show in a very similar way to that used to develop Equation 4.4 that

$$\left|\frac{\partial^2 F}{\partial \beta_{i_{n-1}}^{l_{n-1}} \partial \beta_i^{\widehat{l_n}}}(\beta)\right| \le M\left(|S| + 2\eta\,|U|\,(2\xi+1)\right) \tag{4.5}$$

for all β between β_{n-1} and β_n, regardless to whether $l_{n-1} = \widehat{l_n}$ or not.

From Equation 4.5, since β_n and β_{n-1} differ in only one coordinate, it follows that

$$\left|\frac{\partial F}{\partial \beta_i^{\widehat{l_n}}}(\beta_n) - \frac{\partial F}{\partial \beta_i^{\widehat{l_n}}}(\beta_{n-1})\right| \le M\left(|S| + 2\eta\,|U|\,(2\xi+1)\right)\|\beta_n - \beta_{n-1}\|_\infty .$$

Furthermore, since $er'(V(u)) \ge er'(0)$, it follows from Equation 4.2 that

$$\frac{\partial^2 F}{\partial \beta_i^l \partial \beta_i^l} \ge 2\eta \frac{L-1}{L}\,|U|\,er'(0)$$

allowing to invoke Lemma 17 and conclude the proof. □

Theorem 18 ensures that the gradient of $F(\beta)$ converges to zero along the sequence generated by AgreementBoost. Combined with the fact that $F(\beta)$ is convex, this suffices to ensure that AgreementBoost converges to the global minimum of F, if such a minimum exists. However, since β is not restrained, it may very well be that F has no minimum. As a matter of fact, if one assumes that indeed all learners are able to correctly classify the labeled examples while agreeing on the unlabeled examples, it is guaranteed that F will have no finite minimum. In other words, regardless of the algorithm's current point, one can always do better. Therefore, we can only hope that in the limit, the algorithm converges to the infimum of F's domain, i.e., $F(\beta_n) \to \inf\{F(\beta)\}$.

Unlike the one-dimensional case, the derivative's convergence to 0 is not sufficient to ensure the convergence of a multimultidimensional function to its infimum. As an example, consider the two-dimensional function $f(x,y) = \frac{1}{\sqrt{y}}x^2 + \frac{1}{y}$ (shown in Figure 4.1) and the sequence $\left\{(t,t^4)\right\}_{t=1}^{\infty}$. While the function's gradient $\nabla f(t,t^2) = \left(\frac{2}{t}, -\frac{1}{2t^4} - \frac{1}{t^8}\right)$ converges to 0, the function itself stays above the value of 1: $f(t,t^4) = 1 + \frac{1}{t^4}$ —far away from the desired infimum. In the rest of this chapter, it will be shown that even when $F(\beta)$ has no finite minimum, AgreementBoost converges to its infimum.

Lemma 19. *Let $T : V_1 \to V_2$ be a linear transformation between two finite dimension vector spaces over $\mathbb{R}$ and let $\{v_n\} \in V_1$ be a sequence of vectors. Denote the orthogonal complement of the kernel of T by $Ker(T)^{\perp}$. Then if $\lim_{n\to\infty} \|v_n\|_2 = \infty$ and $(\forall n)\left[v_n \in Ker(T)^{\perp}\right]$ then $\lim_{n\to\infty} \|T(v_n)\|_2 = \infty$ when the norms are taken with respect to some arbitrary orthonormal bases.*

Proof. Let T' be the restriction of T to $Ker(T)^{\perp}$. Since $Ker(T') = \{0\}$, T' is invertible when its range is restricted to its image. Denoting by $\xi_n = T(v_n)$, suppose for a contradiction that $\lim_{n\to\infty} \|\xi_n\|_2 \neq \infty$. Therefore, there exists a sub-sequence of ξ_n which is bounded and hence a converging sub-sequence ξ_{n_i}. Denote its limit by ξ. Since the image of T' is a finite dimension vector space it is closed and therefore $\xi \in Domain\left(T'^{-1}\right)$. However, since T'^{-1} is continuous,

$$\lim_{n\to\infty} \left\|T'^{-1}(\xi_{n_i})\right\|_2 = \lim_{n\to\infty} \|v_{n_i}\|_2 = \left\|T'^{-1}(\xi)\right\|_2 .$$

This is in contradiction to $\lim_{n\to\infty} \|v_n\|_2 = \infty$ since v_n cannot have a bounded sub-sequence. □

Lemma 20. *Let $\{\beta_n\}$ be a sequence of coefficients. Furthermore let*

$$B = \{\beta : \forall u \in U, V(u,\beta) = 0\}$$

and $\tilde{\beta}_n$ be the closest point in B to β_n. Then if $\exists M \in \mathbb{R}$ such that $(\forall u \in U)\,[V(u,\beta_n) < M]$, it follows that $\left\|\beta_n - \tilde{\beta}_n\right\|_{\infty}$ is bounded.

Proof. Note that B is a closed set due to the continuity of $V(u,\beta)$ and therefore $\tilde{\beta}_n$ is well defined. To derive a contradiction, suppose that $\left\|\beta_n - \tilde{\beta}_n\right\|_{\infty}$ is not bounded. Therefore, by looking only at a sub-sequence, it can be assumed that $\left\|\beta_n - \tilde{\beta}_n\right\|_{\infty} \to \infty$. Now, define the following set of linear transformations, for every $l \in \{2\ldots L\}$:

$$T^l(\beta) = \left(g^{l-1}\left(u_1,\beta^{l-1}\right) - g^l\left(u_1,\beta^l\right), \ldots, g^{l-1}\left(u_{n_u},\beta^{l-1}\right) - g^l\left(u_{n_u},\beta^l\right)\right).$$

Note that $\beta \in Ker(T^l)$ if and only if the resulting classifiers g^{l-1} and g^l agree on the unlabeled examples: $(\forall u \in U)\left[g^{l-1}\left(u,\beta^{l-1}\right) = g^l\left(u,\beta^l\right)\right]$. Further define $T(\beta)$ to be the linear transformation resulting from ordering all $T^l(\beta)$ in an $(n_u(L-1))$-tuple.

For the combined transformation T, $\beta \in Ker(T)$ if and only if all resulting classifiers agree on all unlabeled examples. Hence, it follows that $Ker(T) = B$.

Since $Ker(T)$ is a vector subspace and since $\tilde{\beta}_n$ is the closest vector in it to β_n, $\beta_n - \tilde{\beta}_n$ is the projection of β_n onto the orthogonal complement of $Ker(T)$. By Lemma 19, it follows that

$$\|T(\beta_n)\|_2 = \left\|T(\beta_n - \tilde{\beta}_n)\right\|_2 \longrightarrow \infty.$$

Therefore the difference between at least two classifiers on one of the unlabeled examples grows to infinity. This contradicts the assumption that $(\forall u \in U)\,[V(u,\beta_n) < M]$. □

Theorem 21. *Under the assumptions of Theorem 18 with the additional assumption that $er(x)$ is strictly monotonic and that all underlying hypothesis spaces are able to correctly classify the data using finite ensemble classifiers from the intersection of the hypothesis spaces, both the error and the disagreement of the ensemble classifiers constructed by Algorithm 3 converge to 0.*

Proof. Denote the correct classifiers as $\tilde{g}^l = \sum \tilde{\beta}^l_i f^l_j$ and by $\tilde{\beta}$ the corresponding coefficient vector $\left(\tilde{\beta}^1_1, \ldots, \tilde{\beta}^L_{|H^L|}\right)$. Since these classifiers come from the intersection of the hypothesis spaces $(\forall u, l_1, l_2)\,\left[\tilde{g}^{l_1}(u) = \tilde{g}^{l_2}(u)\right]$ and therefore $(\forall \alpha)\left[V\left(\beta + \alpha\tilde{\beta}, u\right) = V(\beta, u)\right]$. This implies the following:

$$\frac{\partial F}{\partial \tilde{\beta}} = \left\langle \nabla F, \tilde{\beta} \right\rangle = \left\langle \nabla E, \tilde{\beta} \right\rangle + \left\langle \eta \nabla D, \tilde{\beta} \right\rangle = \left\langle \nabla E, \tilde{\beta} \right\rangle = \frac{\partial E}{\partial \tilde{\beta}}.$$

From Theorem 18, it follows that $\left|\frac{\partial E}{\partial \tilde{\beta}}(\beta_n)\right| \longrightarrow 0$. Since by assumption all the $\tilde{g}^l$ classifiers are correct $(\forall x\,.\,y_x \tilde{g}^l(x) > 0)$ and $er(x)$ is an increasing monotonic function,

$$\frac{\partial E}{\partial \tilde{\beta}} = \sum_{l,i} \sum_{x_j \in S} er'\left(-y_j g^l(x_j)\right) y_j \tilde{\beta}^l_i f^l_i(x_j) = \sum_{l} \sum_{x_j \in S} er'\left(-y_j g^l(x_j)\right) y_j \tilde{g}^l(x_j) > 0.$$

Therefore, $\left|\frac{\partial E}{\partial \tilde{\beta}}(\beta_n)\right| \to 0$ implies that $(\forall x_i \in S)\,\left[er'\left(-y_j g^l(x_j)\right) \to 0\right]$ and hence $y_j g^l(x_j) \to \infty$, giving the necessary error convergence.

Furthermore $er'\left(-y_j g^l(x_j)\right) \to 0$ implies that $(\forall j, l)\left[\frac{\partial E}{\partial \beta^l_j}(\beta_n) \to 0\right]$ and therefore

$$(\forall j, l)\left[\eta \frac{\partial D}{\partial \beta^l_j}(\beta_n) = \frac{\partial F}{\partial \beta^l_j}(\beta_n) - \frac{\partial E}{\partial \beta^l_j}(\beta_n) \to 0\right] \tag{4.6}$$

or $\lim_{n\to\infty} \|\nabla D(\beta_n)\|_\infty = 0$.

Now, assume in contradiction that $\exists \hat{u} \in U$ such that $\lim_{n\to\infty} V(\hat{u}) \neq 0$. Therefore

$$\lim_{n\to\infty} |D(\beta_n) - L\,|U|\,er(0)| \neq 0. \tag{4.7}$$

As in Lemma 20, denote by $\tilde{\beta}_n$ the closest element in $B = \{\beta : (\forall u \in U)\,[V(u,\beta) = 0]\}$. Since by definition the resulting ensembles 'agree' on all examples it follows that

$$(\forall n)\left[D\left(\tilde{\beta}_n\right) = L\,|U|\,er(0)\right].$$

Using equation 4.3 to apply Lemma 20, there exists some M such that $\left\|\beta_n - \tilde{\beta}_n\right\|_2 < M$. Furthermore since the tangent to a convex function is always an under estimator, it holds that:

$$D(\beta_n) + \left\langle \nabla D(\beta_n), \tilde{\beta}_n - \beta_n \right\rangle \left\|\beta_n - \tilde{\beta}_n\right\|_2 \leq D\left(\tilde{\beta}_n\right)$$

or

$$\begin{aligned} |D(\beta_n) - L|U|er(0)| = \left|D(\beta_n) - D\left(\tilde{\beta}_n\right)\right| &< \left|\left\langle \nabla D(\beta_n), \beta_n - \tilde{\beta}_n \right\rangle\right| M \\ &\leq \|\nabla D(\beta_n)\|_2 M^2. \end{aligned}$$

Which, using Equation 4.6, gives the contradiction to equation 4.7. □

Chapter 5

Experiments

In this chapter a few experiments are presented, testing the algorithm (and theory) presented in the previous chapters. The algorithm was tested on two problems: the first is an artificially constructed problem where all theoretical assumptions hold and the second a ‘real life’ test case. The algorithm was applied to the same web pages classification problem that was used by Blum and Mitchel in [5].

In these experiments, e^x was used as the loss function $er(x)$. This gives an algorithm which is very similar to AdaBoost, with the additional agreement requirement. In order to have a reference point, the proposed AgreementBoost algorithm is compared to AdaBoost, which is run separately on each of the underlying learning algorithms.

In all experiments done, the η parameter is set using the following formula: $\eta = \frac{n_s}{n_u}c$, where c is some constant. This keeps the relative influence of the disagreement and training error terms in Equation 4.1 roughly constant within a single series of experiments. This compensates for the fact that the number of labeled and unlabeled examples changes. More details about the effects of the η parameter and about the reason for these settings are presented in Section 5.3.1.

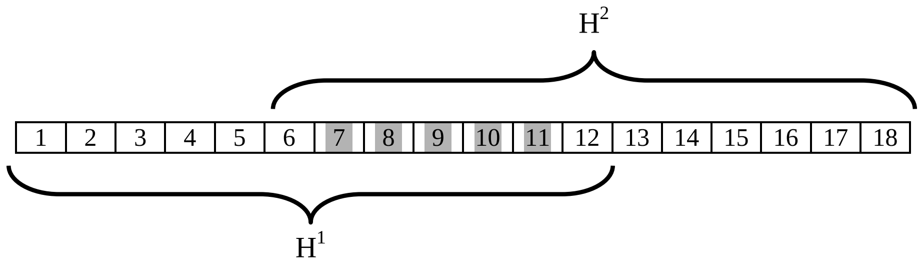
5.1 A toy problem

In order to test the algorithm in a clean and noiseless environment, it was first applied to the following artificial test problem: classification of randomly generated vectors in $\mathbb{R}^n$.

Denoting the coordinates of a vector $x \in \mathbb{R}^n$ by $x = (x_1, \ldots, x_n)$, the hypothesis spaces available to the algorithm consist of classifiers of the form $Pos_j = (x_j > 0)$ or $Neg_j = (x_j \leq 0)$ for $j = 1 \ldots n$, i.e., the underlying hypotheses classify a vector by choosing one of its coordinates and looking at its sign. It then could classify all vectors with a positive j^{th} coordinate as belonging to the $+1$ class (these classifiers are denoted by Pos_j) or as -1 (denoted by Neg_j).

The algorithm is then run using two underlying learners, each being able to choose only part of the possible hypotheses. Learner no. 1 has access to all hypotheses inspecting the first two thirds of the coordinates ($H^1 = \bigcup_{j \leq \frac{2}{3}n} \{Pos_j, Neg_j\}$) while learner no. 2 can access only the last two thirds of the hypotheses ($H^2 = \bigcup_{j \geq \frac{1}{3}n} \{Pos_j, Neg_j\}$). When presented with a query, both algorithms perform an exhaustive search on their

Figure 5.1: An illustration of the hypothesis spaces in the toy problem



$n = 18$, target function coordinates for $r = 5$ are colored in grey

respective hypothesis space and return the best hypothesis found. Note that the intersection of the both hypothesis spaces $H^1 \cap H^2 = \bigcup\limits_{\frac{1}{3}n \le j \le \frac{2}{3}n} \left\{Pos_j, Neg_j\right\}$ is available to both learners.

As the example space $\mathcal{X}$, the n-dimensional cube $= [-1,+1]^n$ is used, along with a uniform distribution over $\mathcal{X}$. After being drawn, each example is labeled by a target function:

$$f_t = \sum_{i=0}^{r-1} \frac{1}{r} Pos_{\lceil \frac{n-r}{2} \rceil + i} \text{ for some radius parameter } r < \frac{1}{3}n.$$

Note that $f_t \in convH^1 \cap convH^2$ and therefore this problem complies to all assumptions needed for the theoretical analysis to hold. Figure 5.1 shows an illustration of the division of hypothesis spaces with respect to the target function.

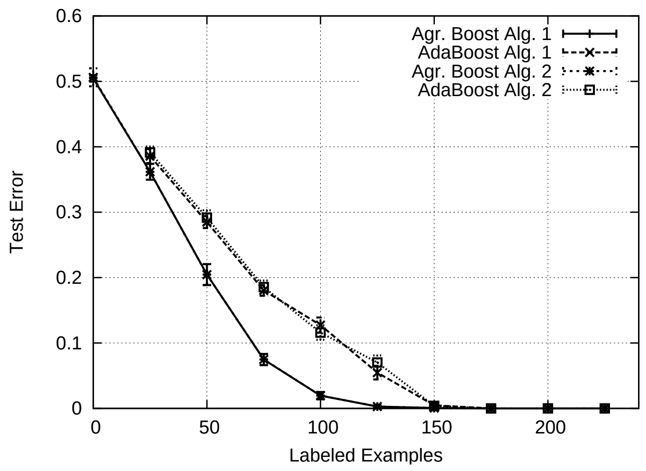
5.1.1 Results

The full results of the comparison on this toy problem are presented in Table 5.1. The boosting algorithms were run on different sample sizes from $n_s = 0$ to $n_s = 225$ examples, where all resulting classifiers had no test error. The experiment was repeated 20 times for each sample size, using freshly drawn examples at each time. All runs used 2000 unlabeled examples and 560 labeled examples as a test group ($\approx$ 25% of 2225, the total number of examples in the 'training set'). Table 5.1 presents the averages of the different runs along with their standard deviation σ. Recall that $\hat{V}\left(g^1, g^2\right)$ is the variance of g^1 and g^2 on the unlabeled examples. Figure 5.2 presents the average test error[1] of the various classifiers in more visual way.

AgreementBoost shows a clear advantage over AdaBoost, at least for this artificial problem where all assumptions hold. The foreseen advantage can be seen both in its precision form and in terms of saving on labeled examples. For example, using $n_s = 100$ labeled examples, AgreementBoost produces classifiers with an average test error of 0.019 and 0.02 while the classifiers boosted by AdaBoost have an error of only 0.127 and 0.116. In terms of saving on labeled examples, AgreementBoost saves $\approx$ 40% of labeled examples if one wants to achieve an error of $\approx$ 0.07. For an error of $\approx$ 0.02 a saving of $\approx$ 30% is achieved.

[1]The ratio between the misclassified examples and the total number of test examples.

Figure 5.2: AgreementBoost applied to toy problem



AgreementBoost compared to AdaBoost.
$n = 200$, $r = 5$, $n_u = 2000$, $\eta = \frac{n_s}{250}$, 1000 boosting iterations

Table 5.1: AgreementBoost applied to toy problem

$n = 200$, $r = 5$, $n_u = 2000$, $\eta = \frac{n_s}{250}$, 1000 boosting iterations

	Agreement Boost				**AdaBoost**	
Labeled	**Learner 1**		**Disagreement**		**Learner 1**	
Examples	Test Err.	σ	$\hat{V}(g^1, g^2)$	σ	Test Err.	σ
0	0.505	±0.022	0	±0.0		
25	0.362	±0.054	4e-6	±3e-6	0.386	±0.047
50	0.205	±0.071	11e-6	±6e-6	0.285	±0.041
75	0.075	±0.038	14e-6	±5e-6	0.180	±0.037
100	0.019	±0.025	10e-6	±5e-6	0.127	±0.052
125	0.003	±0.007	9e-6	±3e-6	0.055	±0.047
150	5.3e-4	±1.7e-3	7e-6	±4e-6	0.002	±0.005
175	0.000	±0.000	4e-6	±2e-6	0.000	±0.000

	Agreement Boost				**AdaBoost**	
Labeled	**Learner 2**		**Disagreement**		**Learner 2**	
Examples	Test Err.	σ	$\hat{V}(g^1, g^2)$	σ	Test Err.	σ
0	0.506	±0.061	0.0	±0.0		
25	0.362	±0.054	4e-6	±3e-6	0.391	±0.040
50	0.205	±0.071	11e-6	±6e-6	0.292	±0.051
75	0.075	±0.038	14e-6	±5e-6	0.186	±0.045
100	0.020	±0.025	10e-6	±5e-6	0.116	±0.050
125	0.003	±0.007	9e-6	±3e-6	0.070	±0.048
150	5.3e-4	±1.7e-3	7e-6	±4e-6	0.004	±0.017
175	0.000	±0.000	4e-6	±2e-6	0.000	±0.000

5.2 Classifying web pages

In this section, we return to the problem of classifying web pages from the WebKb database presented in [5]. The WebKb database [8] contains 1051 web pages, collected from the websites of computer science faculties of four different universities. For each web page, the database contains both the words contained in the page itself (referred to as View 1 in [5]) and words appearing in links referring to that web pages (View 2). The web pages are split into two classes: homepages of courses (230) and non-course pages (821). The goal of the learning algorithms presented in this section is to correctly classify web pages into these two classes.

In order to determine the quality of the resulting classifiers, 25% of the examples in the database were randomly selected in each experiment and held out as a test group. The experiments were repeated 20 times for each parameter set (number of labeled examples, unlabeled examples etc.). The tables and graphs in this section show the average result and standard deviation of these 20 experiments.

5.2.1 Naive Bayes for Text Classification

Naive Bayes is a well-known classifier based on the probabilistic Bayes Rule. The version of it used here, adapted to the problem of text classification, is taken from [7]. For simplicity, we present a simpler model than the one discussed in [7], but this simplified version results in the same practical algorithm. In this model, each example, or document d_i, is represented as a set of words taken from some finite vocabulary set $\mathcal{V} = \left\{w_1, w_2, \ldots, w_{|\mathcal{V}|}\right\}$. Further, let $w_{d_{i,k}}$ denote the word in position k of document d_i.

Enumerating the possible document classes by $\{c_j\}$, it is assumed that the documents are 'generated' in the following way: first, a class c_j is chosen at random according to some distribution $P(c)$ over the possible classes. Then, the length of the document $|d_i|$ is chosen, independently of the class c_j. Having the desired length, the document words are then drawn independently according to some distribution on $\mathcal{V}$, $P(w|c_j)$, which is based on the class c_j. Putting these together, the probability of a specific document d_i, given its class c_j is given by

$$P(d_i|c_j) = P(|d_i|) \prod_{k=1}^{|d_i|} P\left(w_{d_{i,k}}|c_j\right).$$

The probability of a specific document to be drawn, regardless of the chosen class is then given by its marginal probability: $P(d_i) = \sum_j P(c_j) P(d_i|c_j)$. Using Bayes rule, it is now possible to give the probability of a class c_j, given a specific document d_i:

$$P(c_j|d_i) = \frac{P(c_j) P(d_i|c_j)}{P(d_i)} = \frac{P(c_j) \prod_{k=1}^{|d_i|} P\left(w_{d_{i,k}}|c_j\right)}{\sum_{\hat{j}} P\left(c_{\hat{j}}\right) \prod_{k=1}^{|d_i|} P\left(w_{d_{i,k}}|c_{\hat{j}}\right)}.$$

To apply this model to a real classification problem, the necessary distributions are extracted from the training set. For every class c_j, $P(c_j)$ is taken to be the relative frequency of that class within the training set, a set of documents $\mathcal{D}$ and the class assignment:

$$P(c_j) = \frac{\sum_{\hat{i}=1}^{|\mathcal{D}|} P(c_j|d_{\hat{i}})}{|\mathcal{D}|}$$

where $P(c_j|d_{\hat{i}})$ is 1 if document $d_{\hat{i}}$ is classified to class c_j and 0 otherwise.

The probability of the words given a specific class is calculated in a similar fashion while using a Laplacean prior[2]:

$$P(w_t|c_j) = \frac{1 + \sum_{\hat{i}=1}^{|\mathcal{D}|} N(w_t, d_{\hat{i}}) P(c_j|d_{\hat{i}})}{|\mathcal{V}| + \sum_{s=1}^{|\mathcal{V}|} \sum_{\hat{i}=1}^{|\mathcal{D}|} N(w_s, d_{\hat{i}}) P(c_j|d_{\hat{i}})}$$

where $N(w_t, d_{\hat{i}})$ is the number of occurrences of word w_t in document $d_{\hat{i}}$. After extracting these probabilities, a new document d_i is classified by selecting the class with the highest posterior probability: $\arg\max_j P(c_j|d_i)$.

This concludes the description of a general Naive Bayes algorithm. However, this algorithm cannot be used as is in the context of boosting. At each iteration the Naive Bayes algorithm is not only given a set of documents $\mathcal{D}$, but also a new weighing function $w : \mathcal{D} \to [0,1]$. The algorithm should respond as if w represents the true distribution of documents. To accommodate this demand, w is incorporated into the estimates of probabilities, replacing the previous uniform like treatment:

$$P(c_j) = \sum_{\hat{i}=1}^{|\mathcal{D}|} w(d_{\hat{i}}) P(c_j|d_{\hat{i}}) \tag{5.1}$$

$$P(w_t|c_j) = \frac{1 + \sum_{\hat{i}=1}^{|\mathcal{D}|} w(d_{\hat{i}}) N(w_t, d_{\hat{i}}) P(c_j|d_{\hat{i}})}{|\mathcal{V}| + \sum_{s=1}^{|\mathcal{V}|} \sum_{\hat{i}=1}^{|\mathcal{D}|} w(d_{\hat{i}}) N(w_s, d_{\hat{i}}) P(c_j|d_{\hat{i}})} \tag{5.2}$$

5.2.2 Agreeing with the Village Fool...

The first set of experiments on the WebKb database mimics the experiments performed in [5]. The Naive Bayes algorithm is used as a single underlying learning algorithm, applied to each of the so called views: page content and words on incoming links. This is done in a similar fashion to the toy problem, where AgreementBoost is run using the same learning algorithm on two different aspects of an example. AgreementBoost was allowed to run for 1000 iterations, using 525 unlabeled examples and setting $\eta = \frac{n_s}{264}$.

As can be seen in Figure 5.3 and Table 5.2, the classifiers built by AgreementBoost are roughly as good as the better of the two AdaBoost classifiers. Both AgreementBoost classifiers perform roughly the same as AdaBoost applied to the web pages content using Naive Bayes.

One of the main assumptions used in Chapter 2 was that the underlying learners are all capable to produce a good classifier. However, as Figure 5.3(b) and Table 5.3 show, this is not the case in this experiment. While learning the links pointing to the pages produces a classifier with very low training error, it highly over-fits the data and has a

[2]This prevents words from having 0 probability in any class.

Table 5.2: WebKb database, Naive Bayes applied to content and links

$\eta = 1 * \frac{n_s}{264}$, $n_u = 525$

Labeled Examples	**Agreement Boost** **View 1 - Content** Test Err.	σ	**Disagreement** $\hat{V}(g^1, g^2)$	σ	**AdaBoost** **View 1 - Content** Test Err.	σ
52	0.149	±0.041	2.4e-3	±6.1e-3	0.126	±0.071
105	0.101	±0.028	4.5e-3	±7.4e-3	0.092	±0.043
158	0.081	±0.027	6.6e-3	±8.6e-3	0.101	±0.060
184	0.082	±0.029	8.3e-3	±9.8e-3	0.091	±0.064
211	0.084	±0.038	7.3e-3	±8.5e-3	0.087	±0.051
264	0.071	±0.027	9.5e-3	±0.01	0.099	±0.058
Labeled Examples	**Agreement Boost** **View 2 - Links** Test Err.	σ	**Disagreement** $\hat{V}(g^1, g^2)$	σ	**AdaBoost** **View 2 - Links** Test Err.	σ
52	0.173	±0.081	2.4e-3	±6.1e-3	0.263	±0.214
105	0.124	±0.041	4.5e-3	±7.4e-3	0.221	±0.197
158	0.099	±0.028	6.6e-3	±8.6e-3	0.248	±0.187
184	0.097	±0.025	8.3e-3	±9.8e-3	0.251	±0.186
211	0.099	±0.031	7.3e-3	±8.5e-3	0.198	±0.154
264	0.093	±0.027	9.5e-3	±0.01	0.233	±0.168

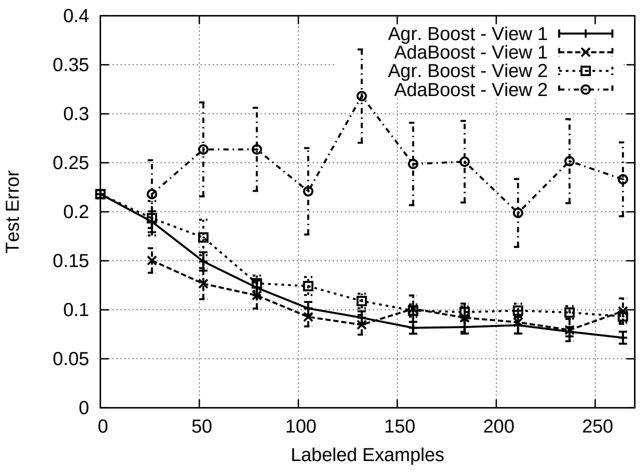
Table 5.3: WebKb database, Naive Bayes applied to content and links

Training Error vs. Test Error (AdaBoost,$n_s = 264$)

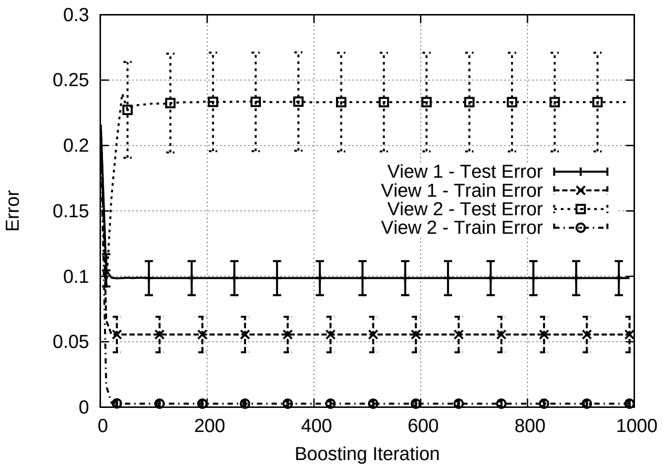
Boosting Iteration	**View 1 - Content** Training Error	σ	Test Error	σ
11	0.066	±0.054	0.104	±0.055
51	0.055	±0.060	0.099	±0.058
101	0.055	±0.060	0.099	±0.058
201	0.055	±0.060	0.099	±0.058
991	0.055	±0.060	0.099	±0.058
Boosting Iteration	**View 2 - Links** Training Error	σ	Test Error	σ
11	0.016	±0.011	0.090	±0.019
51	2.6e-3	±7.5e-3	0.227	±0.163
101	2.6e-3	±7.5e-3	0.232	±0.169
201	2.6e-3	±7.5e-3	0.233	±0.168
991	2.6e-3	±7.5e-3	0.233	±0.168

Figure 5.3: WebKb database, Naive Bayes applied to content and links

$$\eta = 1 * \frac{n_s}{264}\ ,\ n_u = 525$$



(a) Test error: Agreement Boost vs AdaBoost



(b) Overfitting, AdaBoost, $n_s = 264$

very large test error. It is therefore not surprising that such a classifier has nothing to contribute.

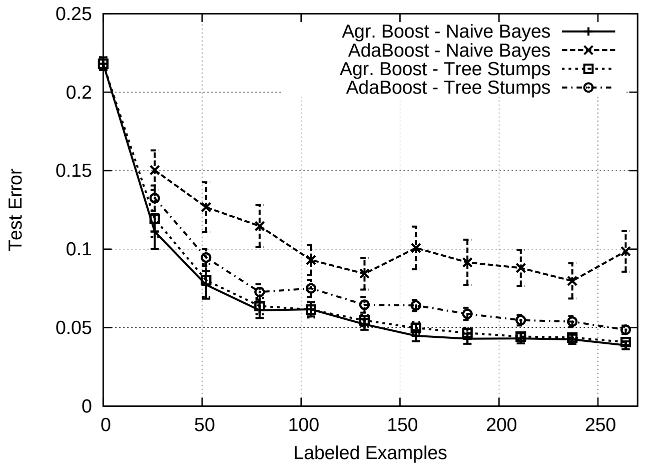
Nevertheless, AgreementBoost does seem to be able to 'choose' the better classifier. Despite of the fact that the two classifiers are forced to agree, the resulting consensus is as good as the better independent classifier.

5.2.3 Using a better learner

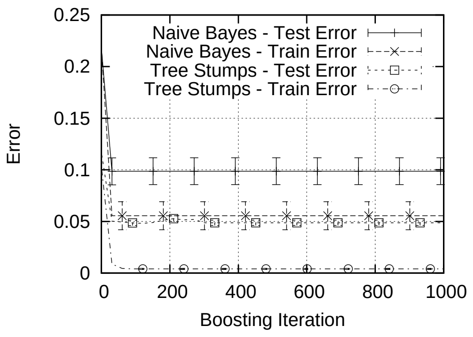
In light of the performance of the underlying links-based algorithm, it was replaced by a another learning algorithm which learns the web pages' content. This new underlying

Figure 5.4: WebKb database, using Naive Bayes and Tree Stumps

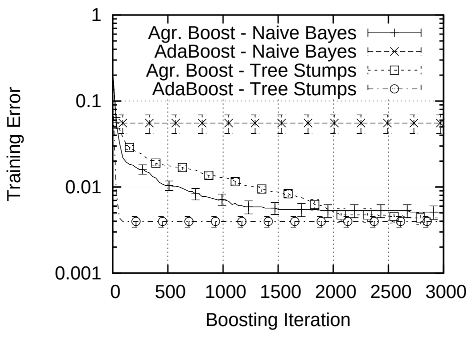
$$\eta = 1 * \frac{n_s}{264},\ n_u = 525$$



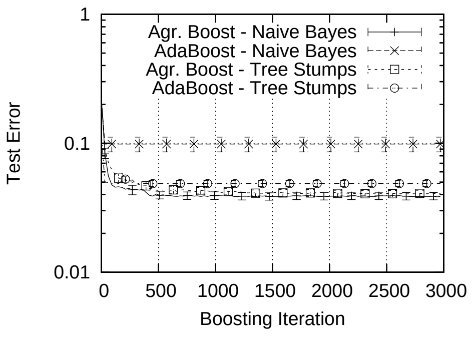
(a) Test error: Agreement Boost vs AdaBoost



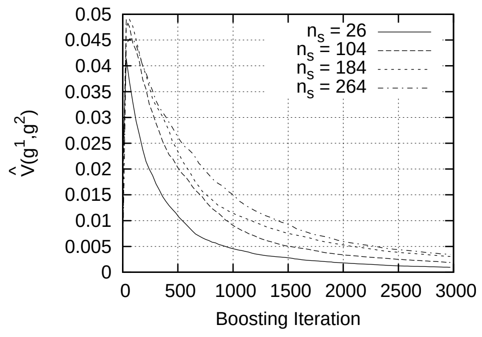
(b) Test vs. Training Error, AdaBoost, $n_s = 264$



(c) Training Error, Agreement Boost, $n_s = 264$



(d) Test Error, Agreement Boost, $n_s = 264$



(e) Disagreement ($\hat{V}(g^1, g^2)$)

Table 5.4: WebKb database, using Naive and Tree Stumps

$\eta = 1 * \frac{n_s}{264}$, $n_u = 525$

Labeled Examples	**Agreement Boost** **Naive Bayes** Test Err.	σ	**Disagreement** $\hat{V}(g^1, g^2)$	σ	**AdaBoost Naive Bayes** Test Err.	σ
52	0.077	±0.039	1.2e-3	±9.2e-4	0.126	±0.071
105	0.061	±0.020	1.8e-3	±1.0e-3	0.093	±0.042
158	0.045	±0.015	2.7e-3	±2.8e-3	0.101	±0.061
184	0.043	±0.015	3.0e-3	±2.6e-3	0.091	±0.064
211	0.043	±0.014	3.0e-3	±2.1e-3	0.088	±0.050
264	0.038	±0.011	3.3e-3	±2.8e-3	0.098	±0.058

Labeled Examples	**Agreement Boost** **Tree Stumps** Test Err.	σ	**Disagreement** $\hat{V}(g^1, g^2)$	σ	**AdaBoost Tree Stumps** Test Err.	σ
52	0.080	±0.046	1.2e-3	±9.2e-4	0.094	±0.024
105	0.061	±0.022	1.8e-3	±1.0e-3	0.075	±0.024
158	0.050	±0.016	2.7e-3	±2.8e-3	0.064	±0.015
184	0.046	±0.014	3.0e-3	±2.6e-3	0.059	±0.017
211	0.044	±9.9e-3	3.0e-3	±2.1e-3	0.054	±0.015
264	0.040	±0.010	3.3e-3	±2.8e-3	0.049	±0.011

Table 5.5: WebKb database, using Naive Bayes and Tree Stumps

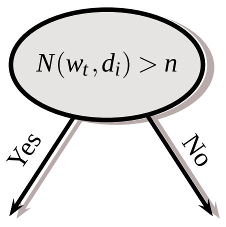
Training Error and Test Error per iteration
$\eta = 1 * \frac{n_s}{264}$, $n_u = 525$

Boosting Iteration	**Agreement Boost** **Naive Bayes** Trn. Err.	σ	**Tree Stumps** Trn. Err.	σ	**AdaBoost Tree Stumps** Trn. Err.	σ
31	55.8e-3	±3.7e-2	66.8e-3	±1.6e-2	8.7e-3	±4.6e-3
151	18.3e-3	±8.8e-3	28.7e-3	±7.4e-3	3.9e-3	±2.5e-3
301	15.1e-3	±8.5e-3	21.9e-3	±6.5e-3	3.9e-3	±2.5e-3
1021	7.0e-3	±4.9e-3	12.8e-3	±3.8e-3		
2011	5.3e-3	±4.0e-3	5.1e-3	±3.8e-3		
2971	5.1e-3	±4.0e-3	4.3e-3	±3.0e-3		

Boosting Iteration	**Agreement Boost** **Naive Bayes** Test Err.	σ	**Tree Stumps** Test Err.	σ	**AdaBoost Tree Stumps** Test Err.	σ
31	0.083	±0.034	0.088	±0.022	0.050	±0.012
151	0.046	±0.017	0.053	±0.019	0.049	±0.011
301	0.043	±0.015	0.048	±0.017	0.049	±0.011
1021	0.039	±0.012	0.042	±0.011		
2011	0.038	±0.011	0.041	±0.011		
2971	0.038	±0.011	0.040	±0.010		

learner is based on a degenerate version of decision trees called tree stumps. Tree stumps consist of only one decision node, classifying an example only according to a single test. In these experiments, the web pages are classified by testing the number of instances of a single word within them. If the word has more instances then a given threshold, the web page is classified to one class and otherwise to the other. An example of such a tree stump is presented in Figure 5.5.

Figure 5.5: A Tree Stump



Before moving on to the results of using AgreementBoost with the new Tree Stumps algorithm, note that it does indeed perform better. This can be seen in Figure 5.4(b) and Table 5.5, showing the results using 264 labeled examples. When boosted with AdaBoost, the resulting tree stumps ensemble had a very low training (3.9×10^{-3} average) and test error (0.049 average). For comparison, the Naive Bayes ensemble had an average training error of 0.055 and an average test error of 0.099.

The full summary of experiments performed with the Tree Stumps algorithm is presented in Figure 5.4 and Tables 5.4 & 5.5. In all experiments shown, 526 examples were used as unlabeled examples, allowing for up to 264 labeled examples. To perform a fair competition and to avoid over-fitting, the AdaBoost was run for only 300 iterations[3]. As can be seen, AgreementBoost produces substantially better classifiers. On average, using the full 264 labeled example set, the tree stumps ensemble produced by AgreementBoost had 0.04 error on the test set. The naive Bayes classifier performed even better with a 0.038 test error. In comparison, the tree stumps ensemble constructed by AdaBoost, which was better than the corresponding naive Bayes classifier, had a test error of 0.049.

In terms of labeled examples reduction, AgreementBoost has also produced good results. The final test error achieved by AdaBoost using the full labeled exampled set (264 examples), was already achieved by AgreementBoost's classifiers using 158 labeled examples, a reduction of 40%.

Figures 5.4(c),(d) and (e) give some insight into the algorithm and the construction of the classifiers. Unlike AdaBoost, which reaches its final training error within 100 iterations, AgreementBoost is slower to converge. However, the agreement between the classifiers results in a considerably lower training error for the naive Bayes classifier. While the AdaBoost-run quickly gets stuck with a training error of 0.055 , the agreement with the tree stumps classifier 'drags' the naive Bayes well below that level. In the AgreementBoost run, the naive Bayes classifier continues to descend, leveling out at around 1500 iterations with a training error of $\approx 5.3 \times 10^{-3}$. Surprisingly, the naive Bayes descends faster then the better tree stumps classifier.

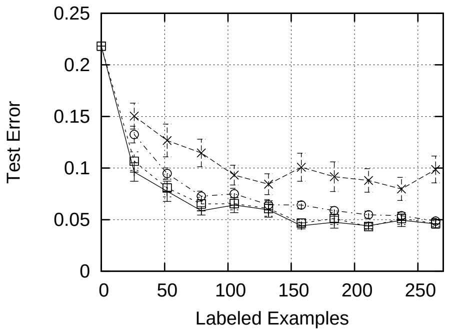
Since this apparent improvement seems to be the result of the extra disagreement term, it is interesting to see the dynamics between it and the training error. This can be achieved by comparing Figure 5.4(c) and Figure 5.4(d). In the beginning of the run, roughly corresponding to the period where AdaBoost reaches its minimum training error, the two learning algorithms focus mainly on the reduction of training error: i.e., the labeled examples. As a result, there is an initial sharp increase in the average disagreement between the two classifiers. After 91 iterations, disagreement is in its peak

[3]'Early stopping' is one of the first techniques suggest to avoid over-fitting.

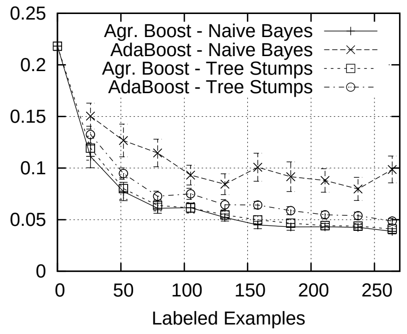
Figure 5.6: Setting the η parameter Stumps

WebKb database, using Naive Bayes and Tree

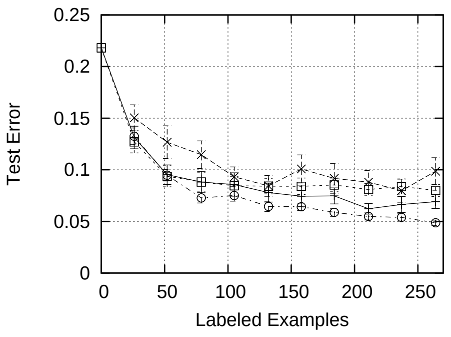
$n_u = 525$



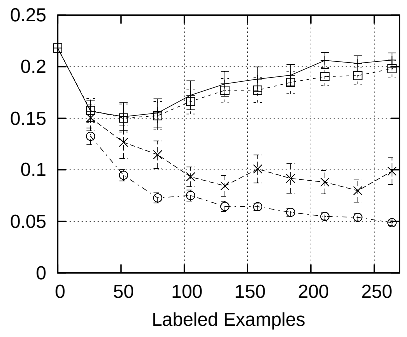
(a) $\eta = 0.1 * \frac{n_s}{264}$



(b) $\eta = 1.0 * \frac{n_s}{264}$



(c) $\eta = 10 * \frac{n_s}{264}$



(d) $\eta = 50 * \frac{n_s}{264}$

average value of 45.2×10^{-3}, while starting from the perfect agreement of the initial 0-classifiers. After this initial strive for correctness, the focus shifts and both learners pay a more equal attention to both labeled and unlabeled examples. The disagreement slowly reduces while still improving the training error.

5.3 Agreeing is important, but how much?

5.3.1 Setting the η parameter

The convergence proof presented in Chapter 4 ensures that if all assumptions are met, the resulting classifiers will fully agree (albeit in the limit) regardless of what value is given to the parameter η. However, this does not give any guarantee as to how fast that will be achieved nor to what happens when the hypothesis spaces cannot fully agree or classify the data. Since the algorithm advances by iteratively minimizing a weighted sum of error and disagreement, one can expect that the weight given to disagreement

Table 5.6: Setting the η parameter

WebKb database, using Naive Bayes and Tree Stumps

$n_u = 525$

Labeled Examples	Naive Bayes Test Err.	Naive Bayes σ	Tree Stumps Test Err.	Tree Stumps σ	Disagreement $\hat{V}(g^1, g^2)$	Disagreement σ
$\eta = 0.1 * n_s/264$						
105	0.064	±0.032	0.065	±0.027	5.3e-3	±7.5e-3
184	0.047	±0.025	0.051	±0.026	4.6e-3	±5.1e-3
264	0.046	±0.019	0.046	±0.012	11.1e-3	±1.9e-2
$\eta = 10 * n_s/264$						
105	0.085	±0.050	0.084	±0.040	1.7e-3	±8.3e-4
184	0.074	±0.034	0.085	±0.029	2.0e-3	±7.1e-4
264	0.069	±0.029	0.080	±0.025	2.2e-3	±4.8e-4
$\eta = 50 * n_s/264$						
105	0.172	±0.062	0.166	±0.054	3.4e-4	±1.4e-4
184	0.191	±0.046	0.184	±0.048	2.9e-4	±1.5e-4
264	0.206	±0.031	0.198	±0.036	2.3e-4	±1.3e-4

For $\eta = 1 * 264/n_s$ and AdaBoost statistics see Table 5.4

will have effects on the algorithm behavior.

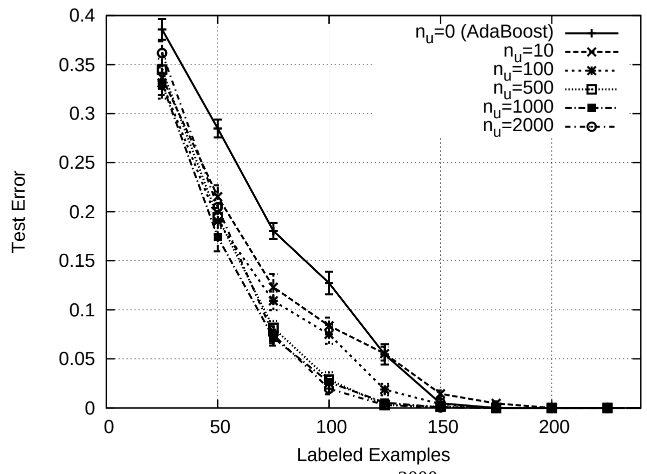
A series of experiments aimed at studying these effects, are presented in Figure 5.6 and Table 5.6. The figures and table display the results of setting η to different values, both lower and higher then the one used so far. For comparison, Figure 5.6(b) shows the results presented already in the previous section for $\eta = 1 * \frac{n_s}{264}$. Setting η to the lower value of $0.1 * \frac{n_s}{264}$ results in relatively good classifiers although the results seem less robust. The gap between the classifiers constructed by AgreementBoost and the ones of AdaBoost is not as stable as when using $\eta = 1 * \frac{n_s}{264}$.

Unlike lower values, setting η to higher values resulted in inferior results. For $\eta = 10 * \frac{n_s}{264}$ the AgreementBoost classifiers lie in between the classifiers produced by AdaBoost and for $\eta = 50 * \frac{n_s}{264}$ the AgreementBoost-classifiers are hardly better than random guessing.

This behavior is caused by the iterative nature of AgreementBoost. At every iteration, only one of the underlying learning algorithm is allowed to advance. At every step only one classifier is changed while the others stay the same. As a result, making disagreement very important restricts the step size (α_t^l) that will be selected in step 2.a.ii. Using a lower value of η allows the algorithm to make bigger steps and converge faster. Therefore, for the high value of $\eta = 50$ the algorithm is effectively stuck.

For this reason, η was set throughout the experiments to be proportional to the ratio $\frac{n_s}{n_u}$. When the ratio $\frac{n_s}{n_u}$ is small, namely there are many unlabeled examples, the sum in the disagreement term consists of more monomials than the error term. Therefore, if η is not changed accordingly and is left constant throughout the experiment, the disagreement term would dominate (see equation 4.2). As a result, in early experiments

Figure 5.7: Effects of unlabeled examples, toy problem



Displaying Alg. 1, $n = 200$, $r = 5$, $\eta = 1 * \frac{2000*n_s}{n_u*250}$,1000 boosting iterations

and using a constant η, the algorithm sometimes performed poorly . Setting η to be proportional to the ratio $\frac{n_s}{n_u}$, compensates for this effect and reduces the importance of the disagreement term accordingly.

5.3.2 The effects of the number of unlabeled examples.

The experiments presented in this section are designed to test the effects of another aspect of the disagreement term: the number of unlabeled examples (n_u). Striving for 'global' agreement, the previous experiments were performed using relatively a lot of unlabeled examples. However, it may very well be that only a few (or fewer) unlabeled examples are sufficient for good results.

Figure 5.7 and Table 5.7 display the results of using various numbers of unlabeled

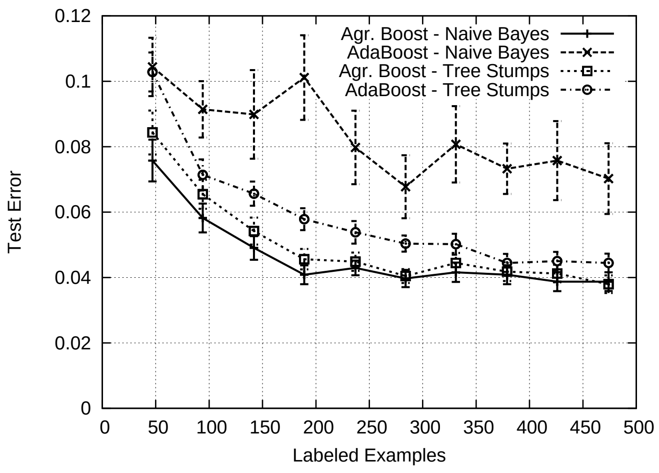
Table 5.7: Effects of unlabeled examples, artificial toy problem

Displaying Alg. 1,$n = 200$, $r = 5$, $\eta = 1 * \frac{2000*n_s}{n_u*250}$,1000 boosting iterations

Unlabeled Examples	**Labeled Examples** $n_s = 75$ Test Err.	σ	$n_s = 125$ Test Err.	σ	$n_s = 175$ Test Err.	σ
0	0.180	±0.036	0.054	±0.046	0.000	±0.000
10	0.123	±0.061	0.055	±0.032	4.6e-3	±9.8e-3
100	0.109	±0.042	0.018	±0.026	0.000	±0.000
500	0.081	±0.031	2.9e-3	±6.4e-3	0.000	±0.000
1000	0.081	±0.031	2.9e-3	±6.4e-3	0.000	±0.000
2000	0.074	±0.038	2.9e-3	±7.1e-3	0.000	±0.000

Figure 5.8: WebKb database, using Naive Bayes and Tree Stumps

Using less unlabeled examples:$\eta = 1 * \frac{n_s}{474}$, $n_u = 315$



examples in the toy problem. In order to keep the magnitude of the disagreement term roughly constant, η was set to $\frac{2000*n_s}{n_u*250}$. Due to the symmetry in the problem, only the results of first classifier are presented.

The results suggest that using only relatively few unlabeled examples ($n_u = 10$, for $n_s > 125$), distracts the algorithm and produces classifiers that are poorer than those generated without any unlabeled examples. However, using $n_u = 100$ unlabeled examples already produces a classifier which is better. For $n_u \geq 500$, there is hardly any difference between the resulting classifiers.

Since the experiments on the toy problem suggest that one can use a relatively small number of unlabeled examples, the WebKb experiment (Section 5.2.3) was repeated. This time, the algorithm was run using only 30% of the database as unlabeled examples (instead of 50% in the original experiment). Since the size of the database is limited, this allowed testing the behavior of AgreementBoost on a larger set of labeled examples. The results are shown in Figure 5.8 and Table 5.8.

The performance of AgreementBoost in this experiment are just as good, if not better, than the previous experiment. As before, AgreementBoost produces better classifiers than both ensembles constructed by AdaBoost. Using roughly 284 labeled examples and 315 unlabeled examples, AgreementBoost reduced the average error from 5% to 3.9%. For comparison, in the previous experiment, using 264 labeled examples and 525 unlabeled examples the results were very similar: 4.9% versus 3.8%. However, the AgreementBoost-ed naive Bayes achieves an almost maximal precision already for 189 labeled examples, having an average training error of 4.1%. From this point, the AgreementBoost classifiers seem to have reached their limit and do no improve significantly with the addition of labeled examples.

The best achieved average test error of AdaBoost (4.5%, by the tree stumps classifier, using 379 labeled examples) is already achieved by AgreementBoost's naive Bayes using only ≈ 150 labeled examples and by the tree stumps classifier using 189. This

Table 5.8: WebKb database, using Naive Bayes and Tree Stumps

Using less unlabeled examples:$\eta = 1 * \frac{n_s}{474}$, $n_u = 315$

	Agreement Boost				**AdaBoost**	
Labeled	**Naive Bayes**		**Disagreement**		**Naive Bayes**	
Examples	Test Err.	σ	$\hat{V}(g^1, g^2)$	σ	Test Err.	σ
94	0.058	±0.019	4.3e-4	±3.1e-4	0.09	±0.038
189	0.041	±0.013	5.7e-4	±3.0e-4	0.10	±0.057
284	0.039	±0.011	7.3e-4	±2.9e-4	0.067	±0.043
379	0.040	±0.013	9.2e-4	±3.5e-4	0.073	±0.034
475	0.038	±0.012	9.9e-4	±3.1e-4	0.070	±0.048
	Agreement Boost				**AdaBoost**	
Labeled	**Tree Stumps**		**Disagreement**		**Tree Stumps**	
Examples	Test Err.	σ	$\hat{V}(g^1, g^2)$	σ	Test Err.	σ
94	0.065	±0.019	4.3e-4	±3.1e-4	0.071	±0.021
189	0.045	±0.013	5.7e-4	±3.0e-4	0.058	±0.015
284	0.040	±9.4e-3	7.3e-4	±2.9e-4	0.050	±0.011
379	0.041	±0.012	9.2e-4	±3.5e-4	0.045	±0.012
475	0.038	±0.012	9.9e-4	±3.1e-4	0.045	±0.013

amounts to reduction of 50% for the tree stumps classifier - the lesser of the two.

Another interesting point to notice is the switch in performance between the two classifiers. When boosted by AdaBoost, the relatively simple tree stumps classifier outperforms the more power full naive Bayes algorithm. However, when using AgreementBoost, their roles reverse: Naive Bayes performs slightly better than the tree stumps ensemble. Some hints for this behavior could already be seen in the previous experiment (Figure 5.4(a)), but here it is much more pronounced.

Chapter 6

Summary and Conclusions

In the first chapter of this thesis, we have formulated and made concrete the potential advantage in making several learning algorithms agree. In order to ensure and validate the learners agreement, an extra set of unlabelled examples is used along side the usual labeled exampled of the supervised training model.

Drawing on the general intuition, a new generalization bound was derived where the penalty relating to the hypothesis space complexity is reduced due the use of unlabelled examples. As a complexity measure, we have used the Rademacher complexity due its technical advantage. However, the same general scheme can be used to improve other generalization bounds that involve other complexity measures. The suggested scheme will work as long as the complexity measure is monotonic with respect to inclusion. However, we expect any proper complexity measure to have this property.

Motivated by the theoretical framework, we have presented a new boosting algorithm named AgreementBoost along with a proof of its convergence (in the limit). The algorithm was applied to a test toy problem and a web-page classification problem (as defined by Mitchel and Blum [5]). In both problems, AgreementBoost reduced the number of labeled examples necessary in order to achieve a desired error by up to 40% (compared to AdaBoost).

While agreement successfully reduces the number of labeled examples, it is not without a price. Increasing the importance assigned to the learners' agreement causes a reduction in the algorithm's convergence speed. Since AgreementBoost constructs its ensembles iteratively, this results in larger and computationally more expensive classifiers. The exact trade-off between agreement weight and convergence speed is yet to be established.

When designing AgreementBoost, we have opted for simplicity and thus avoided using many of the possible improvements and modifications, many of which are non-trivial and justify new research projects. We name a few:

1. Many of the improvements of AdaBoost suggested in the literature can adapted for AgreementBoost. Modifications like regularization terms for the hypotheses weights and soft margins will probability improve that algorithm's performance.

2. For simplicity, we have kept the agreement weight (η) constant along the run. However, we suspect that changing it during the algorithm's run might lead to superior results.

3. Following previous work, we have performed all experiments using only two

underlying learners. However, the theoretical framework is quite more general, allowing for an arbitrary number of underlying learners. Further experimental study involving more learners is required.

Acknowledgement. I'm utterly grateful to Leen Torenvliet, Peter Grunwald, Pieter Adriaans and Peter van Emde Boas for their time, comments and helpful insights, leading to the final version of this work.

Bibliography

[1] R.Meir and G. Ratsch: An Introduction to Boosting and Leveraging. *Advanced lectures on machine learning*. Pages: 118 - 183. ISBN:3-540-00529-3.

[2] P.L. Bartlett adn S. Mendelson: Rademacher and Gaussian Complexities: Risk bounds and Structural Results. *The Journal of Machine Learning Research*, Vol 3. 2003. Pages 463-482.

[3] V. Koltchinskii and D. Panchenko: Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistic*s, 30(1), February 2002.

[4] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee: Boosting the margin: A new explanation for the effectiveness of voting methods. In *Machine Learning: Proceedings of the Fourteenth International Conference*, 1997.

[5] A. Blum and T. Mitchell: Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*. ACM, 1998.

[6] Anselm Blumer and A. Ehrenfeucht and David Haussler and Manfred K. Warmuth: Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM* Vol. 36, issue 4, 1989. Pages 929-965.

[7] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell: Learning to classify text from labeled and unlabeled documents. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. AAAI Press, 1998.

[8] A subset of the 4 Universities dataset containing web pages and hyperlink data. Used in [5, 7] and available on-line:http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-51/www/co-training/data/

[9] D. Luenberger: Introduction to Linear and Nonlinear Programming. Addison-Wesley publishing company. 1973. ISBN 0-201-04347-5.

[10] T. Zhang and F. Oles, A probability analysis on the value of unlabeled data for classification problems. In *Proceedings of the International Conference on Machine Learning (ICML),* pp. 1191–1198, 2000.

[11] Seong-Bae Park and Byoung-Tak Zhang: Co-trained support vector machines for large scale unstructured document classification using unlabeled data and syntactic information. In *Information Processing and Management: an International Journal* Volume 40, Issue 3, Jan 2004.

[12] A. Levin, P. Viola and Y. Freund: Unsupervised Improvement of Visual Detectors using Co-Training. *International Conference on Computer Vision* (ICCV), Oct 2003, Nice, France

[13] Kamal Nigam and Rayid Ghani: Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management* 2000.

[14] Sally Goldman and Yan Zhou: Enhancing supervised learning with unlabeled data. In *International Joint Conference on Machine Learning*, 2000.

[15] R. Hwa, M. Osborne, A. Sarkar, M. Steedman: Corrected Co-training for Statistical Parsers. In the *Proceedings of the Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, International Conference of Machine Learning, Washington D.C., 2003.

[16] D. Pierce and C. Cardie: Limitations of Co-Training for Natural Language Learning from Large Datasets. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* 2001.

[17] Michael Collins and Yoram Singer: Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.

[18] S. Dasgupta, Michael L. Littman, David A. McAllester: PAC Generalization Bounds for Co-training. In *Advances in Neural Information Processing Systems 14* (2001).

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

[20] R. Schapire, Y. Freund, P. Bartlett, and W. Sun Lee: Boosting the margin: A new explanation for the effectiveness of voting methods. In *Machine Learning: Proceedings of the Fourteenth International Conference*, 1997.

[21] L. Mason, J. Baxter, P. L. Bartlett, and M. Frean. Boosting algorithms as gradient descent in function space. *Technical report, RSISE*, Australian National University, 1999.

[22] A.J. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National Conference on Artifical Intelligenc*e, 1998.

[23] K. P. Bennett and A. Demiriz and R. Maclin: Exploiting unlabeled data in ensemble methods. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining,* 2002.

[24] G. Rätsch, S. Mika, and M.K. Warmuth. On the convergence of leveraging. *NeuroCOLT2 Technical Report 98*, Royal Holloway College, London, August 2001.