Technical Reports on Mathematical and Computing Sciences: TR-C132
title: From computational learning theory to discovery science
author: Osamu Watanabe
affiliation:
Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology
Meguro-ku Ookayama, Tokyo 152-8552.
email: watanabe@is.titech.ac.jp
acknowledgments to financial supports:
Supported in part by the Ministry of Education, Science, Sports and Culture,
Grant-in-Aid for Scientific Research on Priority Areas (Discovery Science), 1999.

Abstract. Machine learning has been one of the important subjects of AI that is motivated by many real world applications. In theoretical computer science, researchers also have introduced mathematical frameworks for investigating machine learning, and in these frameworks, many interesting results have been obtained. Now we are proceeding to a new stage to study how to apply these fruitful theoretical results to real problems. We point out in this paper that "adaptivity" is one of the important issues when we consider applications of learning techniques, and we propose one learning algorithm with this feature.

1 Introduction

Discovery science[1] is a new area of computer science that aims at (i) developing efficient computational methods which enable automatic discoveries of scientific knowledge and decision making rules and (ii) understanding all the issues concerned with this goal. Of course, discovery science involves many areas, from practical to theoretical, of computer science. For example, *computational learning theory* should play a key role[2] in discovery science by providing various algorithms for knowledge discovery. It is, however, also true that many techniques developed in computational learning theory do not always scale well in a straightforward way and can not just simply be applied to the cases where a huge amount of data need to be handled. In a nutshell, we need to pay more and more attention to "practicality".

[1] This new key word — discovery science — is the name of a project accepted by Japanese government as one of the priority research projects. It is a three-year project on the area described above started from April 1998 (Chair: Prof. Arikawa, Kyushu Univ.). The name "Discovery Science" has been invented by Prof. Arikawa and Prof. Miyano when they were planning this project.

[2] In fact, computational learning theory researchers form the core of the Discovery Science project.

There are several issues that we should consider for pursuing the practicality. First of all, as often pointed out, we should consider constant factors seriously when discussing time and space efficiency of algorithms. Note that for some cases, program size of implemented algorithms may also become important. Secondly, we had better put more emphasis on average-case efficiency than worst-case one. For example, in the PAC learning framework, many concept classes have been proven to be hard to learn efficiently; nevertheless, some of these concept classes may still have learning algorithms that run reasonably fast on average, and this may be enough for some practical applications. On this issue, refer, for example, an average-case learning framework proposed by Zeugmann [Zeu98, RZ99].

In this paper, we point out that "adaptivity" is another important issue for designing practical learning algorithms. Researchers usually design algorithms considering the worst-case, which is in some sense reasonable because algorithms should robustly work under any circumstances. It is, however, often the case that a given situation is much better than the worst one. An *adaptive algorithm* is designed so that it can take advantage of a better situation (if so) to achieve better performance. More precisely, an adaptive algorithm adapts to a given situation (whether it is favorable or not) and change its strategy to get better performance in the current situation. In practice, when using some software, it may be hard to know *a priori* the circumstances where the software is used. Adaptive algorithms work well even in such cases.

The importance of the adaptivity has been mentioned in various fields. In statistics, for example, adaptive methods in statistical tests have been used since the Second World War [Wal47]. In computational learning theory, we have an adaptive boosting algorithm [FS97]. Shawe-Taylor et al. also proposed the notion of "luckiness" for discussing a way for exploiting simplicity in observed data (see, e.g., [SBWA98]). In this paper, we will show one fully adaptive learning algorithm that is constructed based on an adaptive selection technique and an adaptive boosting technique. With this learning algorithm, we would like to demonstrate the meaning and importance of these adaptive techniques.

A *boosting technique* or a *boosting algorithm* is a way to make a "strong" learning algorithm based on a "weak" learning algorithm. The first boosting algorithm was proposed by Schapire [Sch90] for answering *theoretically* to the question of whether a weak learning algorithm can be "boosted" to a strong one [KV94a]. Later Freund [Fre95] gave a simpler and more efficient boosting algorithm that provides much simpler final hypotheses than Schapire's algorithm. From this point, researchers have started considering boosting techniques as practical tools for designing learning algorithms. Then more recently, Freund and Schapire [FS97] developed an even more simple boosting algorithm — **AdaBoost** — that has several nice features for practical use including adaptivity. Due to these features,

AdaBoost has been used and experimented for various practical applications (see, e.g., [FS96, Qui96, Sch99]).

A boosting algorithm makes use of a weak learning algorithm (a *weak learner*, in short) that produces a not-so-accurate but simple hypothesis (which we call a *weak hypothesis*). Usually a weak learner is unspecified when discussing boosting algorithms. But if a weak learner is also adaptive, we may be able to obtain a fully adaptive learning algorithm. This is what we will try to do in this paper; that is, we will construct a learning algorithm by combining an adaptive boosting algorithm with an adaptive weak learning algorithm.

In this paper, for the weak learning we consider the simplest learning, *hypothesis selection*. That is, the learning task is simply to select a hypothesis from a given set H of hypotheses that has the best accuracy. Thanks to boosting algorithms, we can now use simple weak hypotheses; hence, the set H of all weak hypotheses could be within reasonable size, and it may still be feasible to search the best hypothesis exhaustively in H. On the other hand, a given dataset X, i.e., a set X of example instances, may be huge, and it is just impossible to check with all instances in X. A standard strategy that we would use in such a case is random sampling. That is, we do the selection on some set S of instances randomly chosen from X. Then, of course, an obtained hypothesis h that performs best on S may not be the best, but intuitively, if S is large enough, then the obtained hypothesis h must be close to the best. In fact, several statistical bounds tell us how big S should be. Here is one problem. In order to calculate the size of S by using any of these statistical bounds, we need to know how well the best hypothesis performs on X, but this is usually hard to know in advance. Fortunately, though, there is an *adaptive* way to do this sampling. Recently, we developed [DGW99] a general method to determine sample size based on the performance of hypotheses on observed examples, and by using this method, we can build an adaptive random sampling algorithm — **AdaSelect** — for hypothesis selection. We use this algorithm for our weak learner.

We cannot simply combine **AdaSelect** with **AdaBoost**. There is one more problem we must solve. The original **AdaBoost** does boosting by subsampling. That is, the algorithm first selects a set S of enough number of examples, and this S is used as a "training set" of its weak learner during the whole boosting process. Therefore, even if we have an adaptive weak learner, the boosting algorithm cannot make use of its adaptivity. This problem does not occur if we use a boosting by filtering algorithm. In this paper, we will propose a modified version of **AdaBoost** — **AdaBoost:F** — that is more appropriate for the boosting by filtering framework. Then we will combine our **AdaSelect** with this **AdaBoost:F**.

In this paper, we will analyze the performance of the proposed learning algorithm. But our analysis is rough and we consider only some simple case. More precise analysis

as well as analysis in more general cases is left to our future work.

Basic Notations

We define basic notations used throughout the paper. For any set A, let $\|A\|$ denote the number of elements in A. On the other hand, for any integer x, we use $|x|$ to denote its absolute value. Let $\ln x$ and $\log x$ denote the natural logarithm and the binary logarithm respectively.

For our learning model, we consider the distribution-free PAC learning of Valiant [Val84], and we discuss about learning Boolean concepts. Throughout this paper, we use X to denote some domain of instances and use F to denote some set of Boolean functions on X from which a target Boolean concept is chosen. On the other hand, we use H to denote a set of weak hypotheses and use symbol h to denote an element of H.

We use $f_* \in F$ to denote a given target. That is, our learning task is to obtain an hypothesis $f\colon X \to \{0,1\}$ that is close to f_*. More specifically, the "closeness" is defined as the probability that f coincides with f_* on randomly generated $x \in X$ under a given distribution D. That is, the closeness of f to f_* is measured by the following function.

$$\mathrm{cor}_D(f, f_*) \stackrel{\text{def}}{=} \Pr_{x:D}\{\, f(x) = f_*(x) \,\},$$

where we use $\Pr_{x:D}\{\cdots\}$ to denote the probability that $\cdots$ occurs under D. When f_* is clear from the context, we simply write $\mathrm{cor}_D(f, f_*)$ as $\mathrm{cor}_D(f)$. Here we may be able to use an *example generator* $\mathbf{EX}_{D,f_*}$ that generates, at each call, a pair $(x, f_*(x))$ of an instance x and its classification, where x is generated under the distribution D. We use $D(x)$ to denote the probability that x is generated, and in general, by $D(A)$ we mean that the probability that some instance in A is generated.

2 Adaptive Hypothesis Selection

Suppose that we are given a *huge* set X of instances and a large (but still *of manageable size*) set H of hypotheses and that we are asked to select the best (i.e., the closest) or a nearly the best hypothesis from H for some unknown target concept f_* on X. The simplest way is to test each $h \in H$ for all instances of X and select h_* in H that is closest to f_*. Clearly this selection is not feasible since X is huge. Thus, we usually do this selection on some sample set S of instances randomly chosen from X. Then, of course, an obtained hypothesis h that performs best on S may not be the best, but intuitively, if the sample set S is large enough, then the obtained hypothesis h must be close to the best. In fact, several statistical bounds tell us how big S should be. There is, however, one problem. To calculate sufficient sample size by using any of these statistical bounds,

we need to know how well the best hypothesis performs on X, but it is usually hard to know in advance. Thus, we would usually use some "safe" lower bound p for this. On the other hand, if the best performance is higher than this lower bound, which may often occurs, then we would do the selection on an unnecessarily large sample set. (Note that smaller p requires larger sample size.)

For solving this problem, researchers have proposed several techniques that are called *sequential analysis* or *adaptive sampling*; see, e.g., [Wal47, LNSS93]. An idea common in these techniques is as follows: (i) We do not determine *a priori* sample size. (ii) Instead, we do sampling *on-line* and determine when to stop sampling based on the currently estimated value. This idea often helps us reducing sample size greatly. Our adaptive hypothesis selection — **AdaSelect** — is one of these techniques that fits very well for selecting a nearly the best hypothesis.

Let us specify our problem more clearly. A hypothesis h is considered as the best if $\mathrm{cor}_D(h, f_*)$ is the largest. We use h_* to denote the best hypothesis. Intuitively, the least informative hypothesis is a *random hypothesis* h_{rnd}, a hypothesis that gives 0 or 1 uniformly at random for any instance. Note that even for this random hypothesis h_{rnd}, we have $\mathrm{cor}_D(h_{\mathrm{rnd}}, f_*) = 1/2$. Hence, intuitively, $\mathrm{cor}_D(h, f_*)$ must be larger than $1/2$ for any reasonable hypothesis h. Thus, it is more natural to measure the performance of a hypothesis h by its *advantage* over the random hypothesis, that is, by using the following function.

$$\mathrm{adv}_D(h, f_*) \stackrel{\mathrm{def}}{=} \Pr_{x:D}\{\, h(x) = f_*(x) \,\} - 1/2.$$

Again when f_* is clear from the context, we simply write $\mathrm{adv}_D(h, f_*)$ as $\mathrm{adv}_D(h)$.

Now our problem is stated precisely as follows.

<u>Hypothesis Selection</u>
Given: X, H, and $\mathbf{EX}_{D,f_*}$.
Input: σ and δ, where $0 < \sigma < 1$ and $0 < \delta < 1$.
Goal: With probability $1 - \delta$, find $h \in H$ such that $\mathrm{adv}_D(h) \geq (1 - \sigma)\mathrm{adv}_D(h_*)$.

In later discussion, for the simplicity, we will fix σ to $1/2$.

For solving this problem, we propose an algorithm **AdaSelect** described in Figure 1. Here for any set of examples, we use $\mathrm{adv}_S(h)$ to denote the advantage of h on S; that is,

$$\mathrm{adv}_S(h) \stackrel{\mathrm{def}}{=} \frac{\|\{\, (x, b) \in S \,:\, h(x) = b \,\}\|}{\|S\|} - \frac{1}{2}.$$

For this algorithm, we can prove the following properties, which are derived immediately from [DGW99, Theorem 3 and Theorem 4].

Algorithm AdaSelect
Given: X, H, and $\mathbf{EX}_{D,f_*}$. % Let $N = \|H\|$.
Input: σ, and δ, where $0 < \sigma < 1$ and $0 < \delta < 1$.
begin
  repeat % Initially set $t = 0$ and $S = \emptyset$.
    $t \leftarrow t + 1$;
    $(x, b) \leftarrow \mathbf{EX}_{D,f_*}$;
    $S \leftarrow S \cup \{(x, b)\}$;
    $a_t \leftarrow b\sqrt{\ln(2Nt(t+1)/\delta)/(2t)}$;
  until $\exists h \in H$ $[\mathrm{adv}_S(h) \geq a_t(2/\sigma - 1)]$;
  output $h \in H$ with the largest $\mathrm{adv}_S(h)$;
end.
% Where b is some constant that we may consider $b \approx 1$ (see [DGW99]).

Figure 1: The adaptive hypothesis selection algorithm.

Theorem 1. For given inputs σ and δ, with probability $1 - \delta$, **AdaSelect** halts within t_0 steps and yields a hypothesis $h \in H$ such that $\mathrm{adv}_D(h) \geq (1 - \sigma)\mathrm{adv}_D(h_*)$. Letting $\gamma_* = \mathrm{adv}_D(h_*)$ and $N = \|H\|$, the time bound t_0 is estimated as follows.

$$t_0 \;=\; 4\left(\frac{1}{\sigma\gamma_*}\right)^2 \ln\left(\frac{2N}{\sigma\gamma_*\delta}\right).$$

Remark. We measure the running time of the algorithm by regarding one repeat-loop iteration as a basic step of the algorithm.

Intuitive Justification. Here instead of giving a proof of this theorem, we explain the idea behind this algorithm, which gives an intuitive justification to the theorem. In the following, parameters without definition are the same as those in the theorem.

The key ingredient of our algorithm is random sampling. Intuitively, our sample size is justified by the following lemma, which is derived from the Hoeffding bound (see, e.g., [KV94b]).

Lemma 1. Let S be a set of size t examples obtained by using $\mathbf{EX}_{D,f_*}$ t times. For any ϵ and δ', if $t \geq (1/2\epsilon^2)\ln(N/\delta')$, then the following holds with probability $1 - \delta'$.

$$\forall h \in H \;\; [\; \mathrm{cor}_S(h) - \epsilon \;\leq\; \mathrm{cor}_D(h) \;\leq\; \mathrm{cor}_S(h) + \epsilon \;].$$

Now let us solve our hypothesis selection problem with this lemma. Here for the simplicity, we let $\sigma = 1/2$, and we assume that δ' is set to some appropriate constant.

First we start with $\epsilon_1 = 1/8$ and collect enough number of examples to S. Suppose that h_1 is the best on S and $\mathrm{cor}_S(h_1) = 7/8$ (or more). This means by the lemma that $\mathrm{cor}_D(h_1) \geq 6/8$, i.e., $\mathrm{adv}_D(h_1) \geq 1/4$. On the other hand, since h_1 is the best, we have $\mathrm{cor}_S(h_*) \leq 7/8$, which implies again by the lemma that $\mathrm{cor}_S(h_*) \leq 1$, i.e., $\mathrm{adv}_D(h_*) \leq 1/2$. Hence, $\mathrm{adv}_D(h_1) \geq (1/2)\mathrm{adv}_D(h_*)$; that is, we can conclude that h_1 has desired advantage. On the other hand, if $\mathrm{cor}_S(h) < 7/8$ for all $h \in H$, then we are not sure whether the current best one meets the condition of the problem. In this case we try with $\epsilon_2 = 1/16$. For this ϵ_2, we can conclude that the current best hypothesis h_2 is a desired one if $\mathrm{cor}_S(h_2) \geq 11/16$. In this way, we try random sampling while decreasing ϵ (which increases necessary sample size), and as soon as ϵ becomes small enough, we can get a desired hypothesis. Roughly speaking, this is the idea of **AdaSelect**. In **AdaSelect**, the iteration phase is naturally incorporated, and thus, it has better sample complexity. □ **End Justification.**

Since the idea of sampling adaptively is natural, various adaptive sampling techniques have been used since the Second World War [Wal47]. However, while there are many techniques proposed for estimating, e.g., cor_D, as far as the author knows, no technique seems applicable for estimating adv_D.

3 Adaptive Boosting

Here we propose a modification of **AdaBoost** that seems more suitable for the boosting by filtering framework.

Let us begin by recalling some basic notions. In the PAC learning model, a learning algorithm can generate examples by using $\mathbf{EX}_{D,f_*}$, and it is expected to obtain some f that is close to f_*. More precisely, an algorithm A is called a *PAC learning algorithm* (in the strong sense) if for any D, and for any input ϵ and δ, $0 < \epsilon, \delta < 1$, by using $\mathbf{EX}_{D,f_*}$, A yields some hypothesis f such that $\mathrm{cor}_D(f, f_*)$ ($\stackrel{\mathrm{def}}{=} \Pr_{x:D}\{\, f(x) = f_*(x) \,\}$) $\geq 1 - \epsilon$ with probability at least $1 - \delta$. On the other hand, an algorithm yielding a hypothesis that is better than the random hypothesis, i.e., the hypothesis predicting 0 or 1 uniformly at random, is called a weak PAC learning algorithm. More precisely, for any D and any input δ, $0 < \delta < 1$, a *weak PAC learning algorithm* uses $\mathbf{EX}_{D,f_*}$ and obtains, with probability at least $1 - \delta$, a hypothesis h with advantage $\mathrm{adv}_D(h, f_*) = \gamma > 0$, i.e., $\mathrm{cor}_D(h, f_*) = 1/2 + \gamma$. A hypothesis generated by a weak learning algorithm is called a *weak hypothesis*. Usually, weak hypotheses are much simpler than target functions. We use h to denote a weak hypothesis and H to denote the set of weak hypotheses.

In this paper, for simplifying our discussion, we measure the efficiency of learning algorithms in terms of $1/\epsilon$ and $1/\gamma$. That is, we ignore instance size and we assume that

δ is fixed to some appropriate constant.

We explain the outline of boosting techniques. Suppose that we are given some weak learning algorithm **WeakLearn**. A boosting algorithm runs this **WeakLearn** several times, say t_0 times, under distributions $D_1, ..., D_{t_0}$ that are slightly modified from the given distribution D and collects weak hypotheses $h_1, ..., h_{t_0}$. A final hypothesis is built by combining these weak hypotheses. Here the key idea is to put more weight, when making a new weak hypothesis, to "problematic instances" for which the previous weak hypotheses perform poorly. That is, at the point when $h_1, ..., h_{t-1}$ have been obtained, the boosting algorithm computes a new distribution D_t that puts more weight on those instances that have been misclassified by most of $h_1, ..., h_{t-1}$. Then a new hypothesis h_t produced by **WeakLearn** on this distribution D_t should be strong on those problematic instances, thereby improving the performance of the combined hypothesis built from $h_1, ..., h_t$.

Boosting techniques are classified into two types depending on the way to execute **WeakLearn** on modified distributions:- boosting by subsampling and boosting by filtering [Fre95]. In the boosting by subsampling framework, a boosting algorithm first obtains, by using $\mathbf{EX}_{D,f_*}$, a set S of enough number of examples as a "training set". Then it runs **WeakLearn** on this S by changing weight of each examples. The goal is to obtain a hypothesis that explains the training set well. On the other hand, in the boosting by filtering framework, a boosting algorithm selects examples from the original domain for each time **WeakLearn** is executed. This selection procedure is regarded as a "filter" between $\mathbf{EX}_{D,f_*}$ and **WeakLearn**. That is, it observes each example generated by $\mathbf{EX}_{D,f_*}$ and either "rejects" it and throws it away or "accepts" it and passes it on to **WeakLearn**; by this process, the boosting algorithm runs **WeakLearn** on modified distributions.

Now we can explain technically the meaning of the "adaptivity" of **AdaBoost**. For using any of the previous boosting algorithms [Sch90, Fre95], we need to specify a lower bound γ of the advantage of weak hypotheses that one can expect from the weak learner **WeakLearn**. This γ, with other parameters, determines an actual boosting strategy, which works well so long as the weak learner keeps producing weak hypotheses with advantage $\geq \gamma$. But since this boosting strategy is fixed during the whole boosting process, even if some produced weak hypotheses have much better advantage the boosting algorithm cannot exploit such better hypotheses. More specifically, the speed of boosting depends on the advantage of hypotheses and larger advantage results in faster boosting. But since some fixed γ, which should be a "safe" lower bound of the expected advantage, is used throughout the boosting process, the boosting speed is fixed to a certain level that is unnecessarily slow if some of the obtained weak hypotheses have much larger advantage. **AdaBoost** solves this problem! It *adapts* to the accuracies of the obtained

weak hypotheses and depending on these accuracies it changes its strategy *on-line*, thereby taking advantage of better hypotheses.

Like the other boosting algorithms, **AdaBoost** can use any weak learning algorithm. Here we would like to use the adaptive selection algorithm **AdaSelect** for the weak learning algorithm of **AdaBoost**, thereby developing a fully adaptive learning algorithm that needs no prior knowledge on the circumstances where it is used. Unfortunately, however, **AdaBoost** is not appropriate for this purpose because it is designed in the boosting by subsampling framework. That is, **AdaBoost** first prepares a set S of examples that is used as a training set of **WeakLearn** during the boosting process; therefore, even if **WeakLearn** is adaptive, its adaptivity cannot be used in **AdaBoost**. This problem does not occur if the algorithm is designed in the boosting by filtering framework.

The original **AdaBoost** does not seem suitable for the boosting by filtering framework, at least in a straightforward way. **AdaBoost** also runs **WeakLearn** several times under modified distributions, but if we use the above mentioned filter for generating examples, then at some step, the probability $\widehat{W}_t^{\mathrm{AD}}$ that an example generated by $\mathbf{EX}_{D,f_*}$ is accepted and passed to **WeakLearn** may become so small that a huge number of examples need to be generated by $\mathbf{EX}_{D,f_*}$ for getting one example for **WeakLearn**. Intuitively, this means that the error probability ρ_t of the currently obtained combined hypothesis is small. Unfortunately, however, the probability $\widehat{W}_t^{\mathrm{AD}}$ could be very small compared with ρ_t, and hence it may occur that $\widehat{W}_t^{\mathrm{AD}}$ becomes very small while the error probability ρ_t is still not sufficiently small. For solving this problem, we propose an alternative way to define distributions.

We define some notations and then describe our algorithm. For any $t \geq 1$, assume that we have already obtained hypotheses $h_1, ..., h_{t-1}$, where each h_i is a weak hypothesis of f_* on some distribution D_i. Let $\varepsilon_1, ..., \varepsilon_{t-1}$ and $\gamma_1, ..., \gamma_{t-1}$ denote the error probability and the advantage of these hypotheses; that is, $\varepsilon_i \stackrel{\mathrm{def}}{=} 1 - \mathrm{cor}_{D_i}(h_i)$ and $\gamma_i \stackrel{\mathrm{def}}{=} \mathrm{adv}_{D_i}(h_i)$ $(= 1/2 - \varepsilon_i)$. We assume that the boosting algorithm somehow can get these γ_i's or their approximations. As in the original **AdaBoost**, we use parameters $\beta_1, ..., \beta_{t-1}$ that are defined as $\beta_i \stackrel{\mathrm{def}}{=} \sqrt{\varepsilon_i/(1-\varepsilon_i)}$ $(= \sqrt{(1-2\gamma_i)/(1+2\gamma_i)})$ for each i, $1 \leq i \leq t-1$. (**Remark.** Here for the simplicity, we define β_i as above, which is the square root of β_i used in [FS97].) Also a combined hypothesis f_{t-1} of $h_1, ..., h_{t-1}$ is their weighted majority vote that is defined in the same way as **AdaBoost**; that is, f_{t-1} is defined by

$$f_{t-1}(x) \stackrel{\mathrm{def}}{=} \begin{cases} 1, & \text{if } \prod_{i:h_i(x)=1} \beta_i \geq \prod_{i:h_i(x)=0} \beta_i, \\ 0, & \text{otherwise.} \end{cases}$$

On the other hand, we propose a new weight function. For any hypothesis h and any $x \in X$, define $\mathrm{cons}(h,x) \stackrel{\mathrm{def}}{=} 1$ (resp., -1) if $h(x) = f_*(x)$ (resp., $h(x) \neq f_*(x)$). Then for each instance $x \in X$, its weight $w_{t-1}(x)$ after the $(t-1)$th step is defined as follows.

Algorithm AdaBoost:F % ":F" is for *F*iltering
Given: X, $\mathbf{EX}_{D,f_*}$, and **WeakLearn**.
Input: ϵ and δ, where $0 < \epsilon < 1$ and $0 < \delta < 1$.
begin
  repeat % Initially set $t = 1$.
    call **WeakLearn** where examples are generated by $\mathbf{FiltEX}_{D_t}$;
    if $\mathbf{FiltEX}_{D_t}$ claims "accurate enough" during the above execution **then** exit;
    estimate γ_t; $\beta_t \leftarrow \sqrt{(1-2\gamma_t)/(1+2\gamma_t)}$;
    $t \leftarrow t+1$;
  end-repeat;
  output the current weighted majority vote hypothesis f_{t-1};
end.
% For the simplicity, we assume that **WeakLearn** always returns a weak hypothesis
% with desired accuracy.
procedure $\mathbf{FiltEX}_{D_t}$
begin
  repeat
    use $\mathbf{EX}_{D,f_*}$ to generate an example (x,b);
    $p \leftarrow \min\left(\prod_{1\le i\le t-1} \beta_i^{\mathrm{cons}(h_i,x)},\ 1\right)$;
    accept (x,b) with probability p;
  until some accepted example (x,b) is obtained;
  returns (x,b);
  % During the above execution, if it can be determined that $W_{t-1} < \epsilon$ with
  % confidence $1-\delta$, then terminates the execution by claiming "accurate enough".
end-procedure.

Figure 2: The adaptive boosting algorithm.

$$w_{t-1}(x) \stackrel{\mathrm{def}}{=} \begin{cases} D(x) \times \prod_{1\le i\le t-1} \beta_i^{\mathrm{cons}(h_i,x)}, & \text{if } \prod_{1\le i\le t-1} \beta_i^{\mathrm{cons}(h_i,x)} < 1, \text{ and} \\ D(x), & \text{otherwise.} \end{cases}$$

Let $W_{t-1} \stackrel{\mathrm{def}}{=} \sum_{x\in X} w_{t-1}(x)$. Finally define the next distribution D_t so that $D_t(x) \stackrel{\mathrm{def}}{=} w_{t-1}(x)/W_{t-1}$ for all $x \in X$. Note that D_1 is D.

Our modified version of the adaptive boosting — **AdaBoost:F** — is stated as Figure 2. Note first that $W_{t-1} \stackrel{\mathrm{def}}{=} \sum_{x\in X} w_{t-1}(x)$ bounds the error probability ρ_{t-1} of the weighted majority vote hypothesis f_{t-1}, that is, the following relation holds.

$$\rho_{t-1} \stackrel{\text{def}}{=} \Pr_{x:D}\{\, f_{t-1}(x) \neq f_*(x) \,\} \;\leq\; W_{t-1}.$$

On the other hand, it is clear from the design of $\mathbf{FiltEX}_{D_t}$ that W_{t-1} is the probability that some accepted example is obtained at one repeat-loop iteration in the execution of $\mathbf{FiltEX}_{D_t}$. Thus, at some step, if it is hard to get enough number of accepted examples by $\mathbf{FiltEX}_{D_t}$, which means that W_{t-1} is small, then $\mathbf{FiltEX}_{D_t}$ can stop by claiming that the current f_{t-1} is "accurate enough". (In Figure 2 we omit specifying the way to estimate γ_t and the way to determine $W_{t-1} < \epsilon$; they will be discussed in the next section.)

In the original **AdaBoost**, the above relation does not hold, and this is the reason why we need to modify the way to compute weights. The weight function of **AdaBoost** is $w^{\mathrm{AD}}_{t-1}(x) \stackrel{\text{def}}{=} D(x) \times \prod_{1\leq i\leq t-1} \beta_i^{\mathrm{cons}(h_i,x)}$. (*Cf.* $w_{t-1}(x)$ is defined by using the same formula if $D(x) \times \prod_{1\leq i\leq t-1} \beta_i^{\mathrm{cons}(h_i,x)} \leq D(x)$.) However, by one call of $\mathbf{EX}_{D,f_*}$, we cannot generate an example (x,b) with probability larger than $D(x)$; hence, if we want to generate each example (x,b) proportional to $w^{\mathrm{AD}}_{t-1}(x)$, then the probability of generating (x,b) by one call of $\mathbf{EX}_{D,f_*}$ should be $\widehat{w}^{\mathrm{AD}}_{t-1}(x) \stackrel{\text{def}}{=} w^{\mathrm{AD}}_{t-1}(x) \times \prod_{1\leq i\leq t-1} \beta_i$. Then the probability that some accepted instance is obtained at one repeat-loop iteration of the filter becomes $\widehat{W}^{\mathrm{AD}}_{t-1} \stackrel{\text{def}}{=} W^{\mathrm{AD}}_{t-1} \times \prod_{1\leq i\leq t-1} \beta_i$, where W^{AD}_{t-1} is the sum of all $w^{\mathrm{AD}}_{t-1}(x)$. Note that $\widehat{W}^{\mathrm{AD}}_{t-1}$ is much smaller than W^{AD}_{t-1}. Thus, although W^{AD}_{t-1} bounds the error probability ρ_{t-1}, it may occur that $\widehat{W}^{\mathrm{AD}}_{t-1}$ is very small but the error probability ρ_{t-1} is not small enough.

As explained above, we can overcome this problem by our modification. Since the correctness of **AdaBoost:F** is thus clear, what we need to analyze is its efficiency. For this, we discuss how fast W_t decreases. Here we consider the simplest case[3] where the advantage γ_i of each hypothesis h_i is at least γ and $\beta_i = \beta \stackrel{\text{def}}{=} \sqrt{(1-2\gamma)/(1+2\gamma)}$.

Theorem 2. Suppose that the advantage of weak hypotheses provided by **WeakLearn** is always at least γ during the execution of **AdaBoost:F**. Then for any ϵ, we have $W_t < \epsilon$ for some $t \leq 1/(2\gamma^2\epsilon)$.

Remark. In the following analysis, we approximate $\sqrt{1-4\gamma^2}$ by $1-2\gamma^2$. (See the proof below for the detail.)

Proof Outline. For any $t \geq 1$, let us consider the situation when the $(t-1)$th repeat-loop iteration has finished and the tth repeat-loop is about to start in the main procedure of **AdaBoost:F**, which we simply call "at the tth step". Thus, **WeakLearn** has been called for $t-1$ times, and let $h_1, ..., h_{t-1}$ be the sequence of obtained weak hypotheses.

[3] To tell the truth, I do not know at this moment how to analyze the general case.

Below we assume that the advantage of the next weak hypothesis h_t is at least γ and discuss how much W_t gets decreased from W_{t-1}.

We introduce some notations. For any instance x, the *advantage* (of correct hypotheses over incorrect ones) on x is the number of correct hypotheses minus that of incorrect hypotheses on x. Throughout the proof, a symbol k is used to denote this advantage. Hence, at the tth step, the value of k ranges from $-(t-1)$ to $t-1$. For any k, let C_k denote the set of instances x with advantage k, and define $\overline{w}_k$ by

$$\overline{w}_k \stackrel{\text{def}}{=} \begin{cases} D(C_k)\beta^k, & \text{if } k > 0, \text{ and} \\ D(C_k), & \text{otherwise.} \end{cases}$$

Note that $w_{t-1}(x) = D(x)\beta^k$ for all $x \in C_k$ and that $\overline{w}_k$ is the sum of weight $w_{t-1}(x)$ for all $x \in C_k$. Hence, we have $D_{t-1}(C_k) = \overline{w}_k / W_{t-1}$. We also define $W_{t-1}^+ \stackrel{\text{def}}{=} \sum_{1 \le k \le t-1} \overline{w}_k$ and $W_{t-1}^- \stackrel{\text{def}}{=} \sum_{-(t-1) \le k \le 0} \overline{w}_k$. That is, W_{t-1}^+ (resp., W_{t-1}^-) is the total weight of instances that are correctly (resp., incorrectly) classified by f_{t-1}. Note that $W_{t-1} = W_{t-1}^+ + W_{t-1}^-$.

Roughly speaking, the weight of instances in $C^- \stackrel{\text{def}}{=} \cup_{-(t-1) \le k \le 0} C_k$ remains the same and only the weight of instances in $C^+ \stackrel{\text{def}}{=} \cup_{1 \le k \le t-1} C_k$ gets decreased. But since $W_{t-1}^+ = w_{t-1}(C^+)$ gets decreased, some instances in C^- should move to C^+ because every weak hypothesis has advantage at least γ. Intuitively speaking, the C^+ part works like a pump absorbing instances from the C^- part.

For analyzing this phenomenon, we introduce another weight $\widetilde{w}_k$ for $k \le 0$, which is defined as follows. (A parameter α will be defined later.)

$$\widetilde{w}_k \stackrel{\text{def}}{=} (|k|\alpha + 1)\overline{w}_k.$$

Define $\widetilde{W}_{t-1}^- \stackrel{\text{def}}{=} \sum_{-(t-1) \le k \le 0} \widetilde{w}_k$ and $\widetilde{W}_{t-1} \stackrel{\text{def}}{=} W_{t-1}^+ + \widetilde{W}_{t-1}^-$. It turns out that the analysis becomes much simpler if we analyze $\widetilde{W}_t$ instead of W_t. Notice that this new weight is not used at all in the algorithm, it is used only for our analysis.

For any k, let p_k and q_k denote $\Pr_{x:D_{t-1}}\{\, h_t(x) = f_*(x) \wedge x \in C_k \,\}$ and $\Pr_{x:D_{t-1}}\{\, h_t(x) \neq f_*(x) \wedge x \in C_k \,\}$ respectively. Also define

$$\begin{array}{llllll} P^+ \stackrel{\text{def}}{=} \displaystyle\sum_{1 \le k \le t-1} p_k, & P^- \stackrel{\text{def}}{=} \displaystyle\sum_{-(t-1) \le k \le 0} p_k, & P^{+0} \stackrel{\text{def}}{=} P^+ + p_0, \\ Q^+ \stackrel{\text{def}}{=} \displaystyle\sum_{1 \le k \le t-1} q_k, & Q^- \stackrel{\text{def}}{=} \displaystyle\sum_{-(t-1) \le k \le 0} q_k, & P^{-0} \stackrel{\text{def}}{=} P^- - p_0. \end{array}$$

Then we have

$$\begin{aligned} p_k + q_k &= D_{t-1}(C_k) = \frac{\overline{w}_k}{W_{t-1}}, \\ P^+ + P^- &= P^{+0} + P^{-0} = \Pr_{x:D_t}\{\, h_t(x) = f_*(x) \,\}, \quad \text{and} \\ Q^+ + Q^- &= \Pr_{x:D_t}\{\, h_t(x) \neq f_*(x) \,\}. \end{aligned}$$

Furthermore, the following relation holds.

Claim 1. For any k, we have

$$\begin{aligned}\Pr_{x:D}\{\, h_t(x) = f_*(x) \wedge x \in C_k \,\} &= p_k \cdot W_{t-1} \cdot \beta^{\min(-k,0)}, \quad \text{and} \\ \Pr_{x:D}\{\, h_t(x) \neq f_*(x) \wedge x \in C_k \,\} &= q_k \cdot W_{t-1} \cdot \beta^{\min(-k,0)}.\end{aligned}$$

Now we estimate how $\widetilde{W}_t$ decreases from $\widetilde{W}_{t-1}$. For this, we derive formulas for computing W_t and $\widetilde{W}_t$ inductively. (Below we use, e.g., $\overline{w}_k^{(t)}$ to denote $\overline{w}_k$ just after the tth step. Recall, on the other hand, $\overline{w}_k$ is the value at (the beginning of) the tth step, i.e., just before the tth call of **WeakLearn**.)

For the case $k \geq 1$, we can show the following relation by using Claim 1.

$$\overline{w}_k^{(t)} = W_{t-1}(\beta^{-1} q_{k+1} + \beta p_{k-1}),$$

where $q_t = q_{t+1} = 0$. Summing up $\overline{w}_k^{(t)}$ for $k \geq 1$, we have

$$W_t^+ = W_{t-1} \times \left(\sum_{k=1}^{t} \beta^{-1} q_{k+1} + \beta p_{k-1} \right) = W_{t-1} \times (\beta^{-1}(Q^+ - q_1) + \beta(P^- + p_0)).$$

Note that $W_{t-1}^+ = (P^+ + Q^+) W_{t-1}$. Hence,

$$\Delta_t^+ = W_t^+ - W_{t-1}^+ = W_{t-1} \times \{\, (\beta^{-1} - 1) Q^+ - (1 - \beta) P^+ - \beta^{-1} q_1 + \beta p_0 \,\}.$$

For the case $k \leq 0$, we can prove the following again by using Claim 1.

$$\begin{aligned}\widetilde{W}_t^- &= \overline{w}_0^{(t)} + (1+\alpha)\overline{w}_{-1}^{(t)} + (1+2\alpha)\overline{w}_{-2}^{(t)} + \cdots + (1+t\alpha)\overline{w}_{-t}^{(t)} \\ &= W_t \times \{\, (\beta^{-1} q_1 + p_{-1}) + (1+\alpha)(q_0 + p_{-2}) + (1+2\alpha)(q_{-1} + p_{-3}) \\ &\qquad + \cdots + (1+t\alpha)(q_{-(t-1)} + p_{-(t+1)}) \,\},\end{aligned}$$

where $p_{-t} = p_{-(t+1)} = 0$. On the other hand, by definition we have

$$\begin{aligned}\widetilde{W}_{t-1}^- &= \overline{w}_0 + (1+\alpha)\overline{w}_{-1} + (1+2\alpha)\overline{w}_{-2} + \cdots + (1+(t-1)\alpha)\overline{w}_{-(t-1)} \\ &= W_t \times \{\, (q_0 + p_0) + (1+\alpha)(q_{-1} + p_{-1}) + (1+2\alpha)(q_{-2} + p_{-2}) \\ &\qquad + \cdots + (1+(t-1)\alpha)(q_{-(t-1)} + p_{-(t-1)}) \,\}.\end{aligned}$$

Thus,

$$\Delta_t^- = W_t^- - W_{t-1}^- = W_{t-1} \times (\alpha Q^- - \alpha P^{-0} + \beta^{-1} q_1 - p_0).$$

Therefore,

$$\Delta_t = \Delta_t^+ + \Delta_t^- = W_{t-1} \times \{\, (\beta^{-1} - 1) Q^+ + \alpha Q^- - (1 - \beta) P^{+0} - \alpha P^{-0} \,\}.$$

Recall that $Q^+ + Q^- = \Pr_{x:D_t}\{\, h_t(x) \neq f_*(x) \,\}$ and $P^{+0} + P^{-0} = \Pr_{x:D_t}\{\, h_t(x) = f_*(x) \,\}$. Thus, if the advantage of h_t is at least γ, then we have $Q^+ + Q^- \leq 1/2 - \gamma$ and

$P^{+0} + P^{-0} \geq 1/2 + \gamma$. Here we may assume that $\beta^{-1} - 1 \geq \alpha \geq 1 - \beta$ since $\beta < 1$. (For example, we can set $\alpha = 1 - \beta$.) Then Δ_t is maximized (even though it may be negative) if $Q^+ = 1/2 - \gamma$ and $P^{+0} = 1/2 + \gamma$. That is,

$$\begin{aligned} \Delta_t &\leq W_{t-1} \times ((\beta^{-1} - 1)(1/2 - \gamma) - (1 - \beta)(1/2 + \gamma)) \\ &= W_{t-1} \times \sqrt{1 - 4\gamma^2} - W_{t-1} \approx -2\gamma^2 \cdot W_{t-1}. \end{aligned}$$

Now suppose that $W_{t-1} \geq \epsilon$. Then $\widetilde{W}_t$ decreases from $\widetilde{W}_{t-1}$ by at least $2\gamma^2\epsilon$. On the other hand, $\widetilde{W}_0 = W_0 = 1$. Hence, either $W_i < \epsilon$ for some i, $1 \leq i \leq 1/(2\gamma^2\epsilon)$; otherwise, $\widetilde{W}_i < \epsilon$ for some i, $1 \leq i \leq 1/(2\gamma^2\epsilon)$, but $W_i \leq \widetilde{W}_i$, a contradiction. $\square$

The above theorem shows that **AdaBoost:F** needs to call **WeakLearn** $O(1/(\gamma^2\epsilon))$ times. Unfortunately, this is rather big compared with the previous boosting algorithms. For example, the boosting algorithm $\mathbf{B}_{\text{Filt}}$ of [Fre95] needs to call **WeakLearn** $O((1/\gamma^2)\ln(1/\epsilon))$ times. We believe that our analysis is still rough and we may be able to get a better bound. More specifically, we had Δ_t $(= \widetilde{W}_t - \widetilde{W}_{t-1}) \leq W_{t-1}\sqrt{1 - 4\gamma^2} - W_{t-1}$; but if $\widetilde{W}_{t-1} \approx W_{t-1}$, then we could get $\widetilde{W}_t \leq \widetilde{W}_{t-1}\sqrt{1 - 4\gamma^2}$, which gives $O((1/\gamma^2)\ln(1/\epsilon))$ bound. Note that the above bound is obtained by assuming $Q^- = P^- = 0$, in which case we certainly have $\widetilde{W}_{t-1} = W_{t-1}$. Or we may be able to use $\widetilde{w}_{t-1}(x)$ as an actual weight that is used to determine the next distribution. In this case we can also prove that $\widetilde{W}_t \leq \widetilde{W}_{t-1}\sqrt{1 - 4\gamma^2}$.

4 Fully Adaptive Learning

Now that we have obtained an adaptive boosting algorithm using filtering, it is easy to combine it with our adaptive hypothesis selection algorithm. We can simply use **AdaSelect** for **WeakLearn** of **AdaBoost:F**. Then we get a fully adaptive learning algorithm that requires no prior knowledge on the circumstances where it is used.

Let us estimate, though roughly, the efficiency of this learning algorithm. Here again we consider the simplest case. That is, we assume that for a given target function f_*, and for any distribution D' (used in the algorithm) there exists a hypothesis $h \in H$ that has advantage at least γ under D'. We also ignore the size parameter N (i.e., the number of possible weak hypotheses) and the confidence parameter δ (required by both **AdaSelect** and **AdaBoost:F**) since these parameters affect the efficiency only logarithmically. Here we measure running time by counting the number of calls of the generator $\mathbf{EX}_{D,f_*}$. That is, we estimate the number of generator calls executed by our learning algorithm until it obtains an ϵ-close hypothesis for a given target f_*.

First consider each call of **AdaSelect**. Here we simply let $\sigma = 1/2$. Then it follows from Theorem 1 that the number of examples for selecting a nearly the best weak hy-

pothesis is $O((1/\gamma^2)\ln(1/\gamma))$. Note that for getting one example by filtering, we need to generate, on average, $1/W_t$ examples by $\mathbf{EX}_{D,f_*}$; hence, (while $W_t \geq \epsilon$) the number of generator calls is $O(1/\epsilon)$ on average for each example required by **AdaSelect**. Thus, for each call of **AdaSelect**, the generator is called at most $O((1/\epsilon\gamma^2)\ln(1/\gamma))$ times.

The condition $W_{t-1} < \epsilon$ should be tested at some point. For this we can use random sampling. According to the Hoeffding bound, $O(1/\epsilon^2)$ examples is sufficient for this test. Thus, for example, if $1/\gamma^2 > 1/\epsilon$, then the number of generator calls per one call of **AdaSelect**, i.e., $O((1/\epsilon\gamma^2)\ln(1/\gamma))$, is large enough, and this test is achieved during the execution of **AdaSelect**. (We may postpone this test unless W_{t-1} becomes sufficiently small and $\mathbf{FiltEX}_{D_t}$ needs to call $\mathbf{EX}_{D,f_*}$ many times to generate one example.)

Next estimate how many times **AdaSelect** is called in **AdaBoost:F**. Here we consider another point that has been left in the previous section; that is, the estimation of γ_t. Note that a reasonable approximation of the advantage γ_t of an obtained weak hypothesis h_t is available through random sampling. Here again we can use the adaptive sampling, the idea behind our **AdaSelect**. In fact, we can use $\mathrm{adv}_S(h_t)$ computed in **AdaSelect**. It is not hard to show (by using analysis similar to the one in the proof of Theorem 1) that $\mathrm{adv}_S(h_t) \lesssim 3\mathrm{adv}_{D_t}(h_t)/2$. Thus, for γ_t, we may safely use $2\mathrm{adv}_S(h_t)/3$, which is larger than $\mathrm{adv}_{D_t}(h_*)/3 \geq \gamma/3$, where h_* is the optimal hypothesis. Thus, it follows from Theorem 2 that **AdaBoost:F** calls **AdaSelect** at most $O(1/(\epsilon\gamma^2))$ times.

In summary, the total number of generator calls is $O((1/\epsilon\gamma^2)\cdot\max(\,(1/\epsilon\gamma^2)\ln(1/\gamma), 1/\epsilon^2\,))$, which is roughly $O((1/\epsilon^2\gamma^4)\ln(1/\gamma))$. Let us compare this number with existing algorithms. For our example, consider here the combination of the boosting algorithm $\mathbf{B}_{\mathrm{Filt}}$ of [Fre95] and the simple batch type random hypothesis selection using the Hoeffding bound. For a given lower bound γ, the simple batch selection needs $O(1/\gamma^2)$ examples, and thus, in total $\mathbf{B}_{\mathrm{Filt}}$ needs to call the generator $\mathbf{EX}_{D,f_*}$ $O((1/\epsilon\gamma^4)\ln(1/\epsilon))$ times. Roughly speaking the additional factors $\ln(1/\gamma)$ and $1/\epsilon$ are what we have to pay for the adaptivity in the hypothesis selection and in the boosting respectively.

Clearly, our obtained bound is not satisfactory. Also we only consider the simplest situation, and for showing the adaptivity of this proposed algorithm, we need to evaluate its efficiency in more general situations where the accuracy of weak hypothesis varies during the boosting. On the other hand, for showing its practical applicability, we need to do some computer experiments on various kinds of datasets. These works are left for the future study.

Acknowledgment

A part of this work, in particular, the adaptive selection part came from a series of joint works [DGW98, DGW99] with Carlos Domingo and Ricard Gavaldà. I have been enjoying working with these talented researchers, and I particularly thank to them for giving me various useful comments on earlier versions of this manuscript. This work is supported in part by Grant-in-Aid for Scientific Research on Priority Areas (Discovery Science), 1999, the Ministry of Education, Science, Sports and Culture.

References

[DGW98] C. Domingo, R. Gavaldà, and O. Watanabe, Practical algorithms for on-line selection, in *Proc. of the First Int'l Conference on Discovery Science, DS'98*, Lecture Notes in Artificial Intelligence 1532:150–161, 1998.

[DGW99] C. Domingo, R. Gavaldà, and O. Watanabe, Adaptive sampling methods for scaling up knowledge discovery algorithms, Technical Report C-131, Dept. of Math. and Computing Sciences, Tokyo Institute of Technology, 1999.

[Fre95] Y. Freund, Boosting a weak learning algorithm by majority, *Information and Computation*, 121(2):256–285, 1995.

[FS96] Y. Freund and R.E. Schapire, Experiments with a new boosting algorithm, in *Machine Learning: Proc. of the 13th Int'l Conference*, 148–156, 1996.

[FS97] Y. Freund and R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.

[KV94a] M.J. Kearns and L.G. Valiant, Cryptographic limitations on learning boolean formulae and finite automata, *J. Assoc. Comput. Mach.*, 41(1):67–95, 1994.

[KV94b] M.J. Kearns and U.V. Vazirani, *An Introduction to Computational Learning Theory*, Cambridge University Press, 1994.

[LN95] Richard J. Lipton and Jeffrey F. Naughton, Query size estimation by adaptive sampling, *Journal of Computer and System Science*, 51:18–25, 1995.

[LNSS93] R.J. Lipton, J.F. Naughton, D.A. Schneider, and S. Seshadri, Efficient sampling strategies for relational database operations, *Theoretical Computer Science*, 116:195–226, 1993.

[Qui96] J.R. Quinlan, Bagging, boosting, and C4.5, in *Proc. of the 13th National Conference on Artificial Intelligence*, 725–730, 1996.

[RZ99] R. Reischuk and T. Zeugmann, A complete and tight average-case analysis of learning monomial, in *Proc. 16th Int'l Sympos. on Theoretical Aspects of Computer Science, STACS'99*, 1999, to appear.

[SBWA98] J. Shawe-Taylor, P.L. Bartlett, R.C. Williamson, and M. Anthony, Structural risk minimization over data-dependent hierarchies, *IEEE Trans. Information Theory*, 44(5):1926–1940, 1998.

[Sch90] R.E. Schapire, The strength of weak learnability, *Machine Learning*, 5(2):197–227, 1990.

[Sch99] R.E. Schapire, Theoretical views of boosting, in *Computational Learning Theory: Proc. of the 4th European Conference, EuroCOLT'99*, 1999, to appear.

[Val84] L. Valiant, A theory of the learnable, *Communications of the ACM*, 27(11):1134–1142, 1984.

[Wal47] A. Wald, *Sequential Analysis*, Wiley Mathematical, Statistics Series, 1947.

[Zeu98] T. Zeugmann, Lange and Wiehagen's pattern language learning algorithm: an average-case analysis with respect to its total learning time, *Annals of Math. and Artificial Intelligence*, 23(1-2):117–145, 1998.