Sifting informative examples from a random source.

Yoav Freund
AT&T Bell Labs,
600 Mountain Ave.
Murray Hill, NJ, 07974

Abstract

We discuss two types of algorithms for selecting relevant examples that have been developed in the context of computation learning theory. The examples are selected out of a stream of examples that are generated independently at random. The first two algorithms are the so-called "boosting" algorithms of Schapire [Schapire, 1990] and Freund [Freund, 1990], and the Query-by-Committee algorithm of Seung [Seung *et al.*, 1992]. We describe the algorithms and some of their proven properties, point to some of their commonalities, and suggest some possible future implications.

Introduction

One of the most studied frameworks for machine learning is that of learning concepts from random examples. A *concept* c is a mapping from some *instance* space X to a binary *label*. For example, an instance can be the results of some medical tests, and the label can be whether or not the diagnosed patient has some particular disease. The concept is a mapping from test results to diagnosis, as provided by a medical expert. The goal of concept learning is to find an approximation of the concept, given a set of labeled instances and some prior knowledge about properties of the concept. The learning algorithm observes a set of labeled instances, also called the *training set* and generates a *hypothesis*, which is an approximation of the concept.

In the PAC[1] framework,[Vapnik, 1982; Valiant, 1984] the instances are assumed to be drawn independently at random according to some fixed distribution $\mathcal{D}$ over the instance space. These instances are then labeled according to a concept c selected from a *concept class* C. We assume that the labeled instances, which are also called *examples*, are presented to the learner, one by one, and an updated hypothesis is output by the learning algorithm after each new example. The error of a hypothesis h is defined as

$$\mathrm{err}(h) = \Pr_{x \in \mathcal{D}} \left(c(x) \neq h(x) \right) ,$$

which is the probability that the hypothesis and the concept disagree on the label of an instance randomly drawn according to the distribution $\mathcal{D}$.

One of the goals of studies in computational learning theory is to characterize the number of examples that are required for generating a hypothesis with small error when learning a given concept class. It has been shown in various contexts [Blumer *et al.*, 1989; Ehrenfeucht *et al.*, 1989] that at least d/ϵ examples are needed for generating a hypothesis whose error is smaller than ϵ, where d is a parameter (such as the VC dimension) that characterizes the complexity of the concept class. In other words, if m_0 examples are needed to generate a hypothesis h_0 whose error is 10%, then an additional set of m_0 examples is typically needed for generating a hypothesis h_1 whose error is 5%. On the other hand, notice that the hypothesis h_0 can predict correctly the labels of 90% of the additional examples. In other words, later examples are less informative to the learning process than earlier ones. This observation opens up the possibility that not all of the later examples are as essential to the learning algorithm as the earlier ones. One can hope to find selection methods for sifting through the additional m_0 examples and filtering out a small set of *relevant* examples that contain the essential information. In this way we might reduce the amount of memory and computation time that is needed by the learning algorithm.

In this paper we present two types of filtering methods for which some rigorous mathematical analysis has been done. This analysis shows that the methods are guaranteed to work in some well defined scenarios. The formal definition of these scenarios is beyond the scope of this paper and can be found in the referenced papers. In this short abstract we describe the example selection rules used in the different algorithms, discussing the intuitions behind them, and summarizing their analysis.

The first method is based on the concept for *boosting*, which was invented by Schapire [Schapire, 1990]. An improved boosting algorithm was later given by the author [Freund, 1990; Freund, 1992]. Using boosting one can guarantee that in the PAC learning framework any learning algorithm can be used to select a small

[1] Probably Approximately Correct

subset of the training set which contains all of the information relevant for learning the concept. We give a sketch of the two boosting methods known today in Section .

The second method is "Query by Committee" (QBC) invented by Seung et al. [Seung *et al.*, 1992]. The goal here is to select the relevant instances *before* knowing their label. This is important, for example, when the price of labeling is high. In Section we describe the QBC algorithm and sketch its main properties.

In Section we highlight the properties that are common to the two methods and suggest some possible applications.

Boosting

Suppose that we are provided with a learning algorithm that generates hypotheses whose error is guaranteed to be smaller than $1/2-\gamma$ for some fixed $\gamma > 0$ and for any distribution $\mathcal{D}$ over the instance space. This type of learning algorithm is called a *weak* PAC learning algorithm because the error of its hypothesis is only slightly smaller than that of a random guess. We denote a generic weak learning algorithm by **WeakLearn** and call the hypotheses that it generates weak hypotheses.

The concept of a boosting algorithm was introduced by Schapire [Schapire, 1990] in order to prove that any weak PAC learning algorithm can be transformed into a "strong" PAC learning algorithm, i.e., one which can generate a hypothesis of arbitrary accuracy.

A boosting algorithm is a learning algorithm which uses **WeakLearn** as a subroutine. It calls **WeakLearn** several times, each time presenting it with a different distribution of examples. It then combines the resulting hypotheses into a single accurate hypothesis which is its output. The boosting algorithm generates the different distributions of examples by selecting them from a stream of random examples using selection rules that are based on the correct label of the instance and on the labels assigned to it by previously generated hypotheses. It is these selection rules which interest us in the context of this paper.

Schapire's boosting algorithm is based on a method for combining three weak hypotheses. The outcome of the weak hypotheses is combined using a majority vote. This combined rule is guaranteed to have error significantly smaller than the error of the combined weak hypotheses. In order to reach an arbitrary accuracy, this scheme is repeated recursively. The three hypotheses are generated in order by **WeakLearn**. In order to generate each one, **WeakLearn** is presented with examples that are selected out of a stream of random independent instances using the rules described below. Let us denote by x a random instance, by $c(x)$ the label assigned to x by the concept c and by $h(x)$ the label assigned to x by the hypothesis h. The three weak hypotheses h_1, h_2, h_3 are generated one by one as follows.

The rules for choosing the examples:

1. **In order to generate h_1, no selection is performed, i.e., the original distribution of examples is used.**
2. **In order to generate h_2, the following randomized rule is used:**
   with probability $1/2$, an example x such that $h_1(x) = c(x)$ is selected, and with probability $1/2$ an example such that $h_1(x) \neq c(x)$ is selected.
3. **In order to generate h_2, all examples such that $h_1(x) \neq h_2(x)$ are selected.**

Intuitively, this method for filtering examples emphasizes the examples on which the weak hypotheses are wrong.[2] Step 2 concentrates on the mistakes of h_1 exactly to the point that h_1 loses all of its edge over a random guess. Step 3 concentrates on those examples on which the weak hypotheses disagree, which implies that one of them is incorrect.

A different boosting algorithm was given by the author [Freund, 1990]. This boosting algorithm does not employ a recursive construction, but rather combines k weak hypotheses by a single majority vote rule. Similar to Schapire's method, the centerpiece of the algorithm is a rule for selecting examples. At a typical step during the boosting algorithm the hypotheses $h_1, \ldots, h_i$ have already been generated and the boosting algorithm selects the examples that it provides to **WeakLearn** in order to generate the hypothesis h_{i+1}. The filtering rule is defined as follows: for each example it counts the number r of hypotheses in $h_1, \ldots, h_i$ that produce the correct label of the example. It then accepts the example with probability that is a function of i and r, which is defined as follows:

$$\alpha_r^i = \begin{cases} \text{if } i - \frac{k}{2} < r \leq \frac{k}{2} \text{ then} \\ \binom{k-i-1}{\lfloor \frac{k}{2} \rfloor - r} \left(\frac{1}{2}+\gamma\right)^{\lfloor \frac{k}{2} \rfloor - r} \left(\frac{1}{2}-\gamma\right)^{\lceil \frac{k}{2} \rceil - i - 1 + r} \\ \text{otherwise } 0 \end{cases}, \tag{1}$$

As defined above, $\gamma > 0$ is a small constant such that **WeakLearn** is guaranteed to generate a hypothesis whose error is smaller than $1/2 - \gamma$ for any instance distribution $\mathcal{D}$. In Figure 1 we graph α_r^i against r for a typical value of i.

It can be seen from Figure 1 that the examples that have the highest probability of being accepted by the selection rule are those on which the number of (weak) hypotheses that vote for each of the two labels is approximately equal but the incorrect label has a slightly larger number of votes. Examples that have all the

[2] One might be tempted to select *only* the examples on which $h_1(x) \neq c(x)$. However, note that in this case **WeakLearn** can output $h_2 = \neg h_1$ as a hypothesis which has no error, but this clearly does not provide the boosting algorithm with a useful new hypothesis.

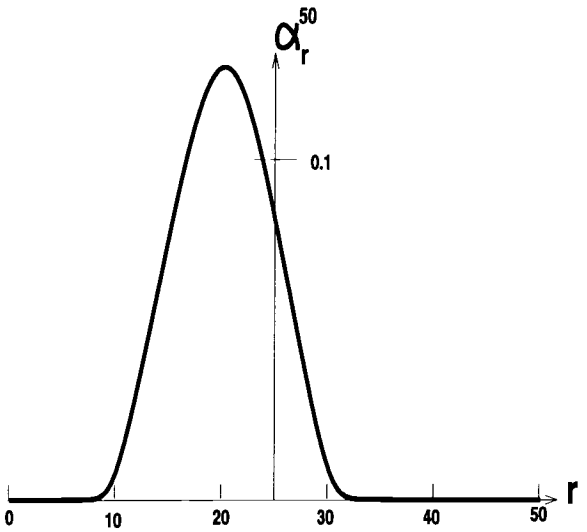

Figure 1: A figure of the value of α_r^i, the probability of accepting a random example, as a function of r, the number of correct hypotheses, for $i = 50$. The other parameters are fixed as $k = 100$, and $\gamma = 0.1$. The horizontal axis is placed at $r = 25$ which corresponds to the case in which the number of correct hypotheses is equal to the number of incorrect hypotheses.

weak hypotheses in agreement have very small probability of acceptance whether or not the agreed label is the correct one. The intuition behind this choice is as follows. If all the hypotheses agree and are all correct, then the label of the instance has been securely identified, and so it is a waste to spend more resources on labeling it. On the other hand, if all the hypotheses agree and they are all *incorrect* then the example is an *atypical* example which is very hard to classify correctly. It is thus better for the overall accuracy to treat it as noise and "give up" on labeling it correctly.

Both of these boosting algorithms have been proved to have the capability of transforming any weak learning algorithm to a strong learning algorithm. Moreover, the analysis of the algorithms shows that the number of times that **WeakLearn** is called during the boosting process is very small. Specifically, if $\epsilon > 0$ is the desired error of the final strong hypothesis, then **WeakLearn** is called $O(\log 1/\epsilon)$ times. Thus, if the number of examples that are required by **WeakLearn** does not depend on the distribution of the instances, the total number of examples selected is also $O(\log 1/\epsilon)$. On the other hand, the number of examples that the learning algorithm has to test in order to select the required number of examples[3] is[4] $\tilde{O}(1/\epsilon)$.

It thus seems that boosting is a promising method for selecting relevant examples out of a random source. More experiments on using boosting methods for real world problems are needed in order test this potential. Some encouraging results have been reached in experiments by Drucker [Drucker *et al.*, 1993; Drucker *et al.*,], however, these experiments tested the potential of boosting as a method for improving the performance of a learning algorithm and not as a method for reducing the size of the training set.

One assumption that is probably not completely realistic is that the performance of the learning algorithm **WeakLearn** does not depend on the distribution of the instances. Potentially, the learning algorithm might require more (selected) examples in order to reach the desired performance with respect to the filtered distributions. Experiments are needed to know the degree of this effect.

Query by Committee

Boosting provides some promising methods for selecting relevant examples. However, these methods are based on selection rules which depend on the labels of the examples. In some situations, obtaining the correct label of instances is expensive. For example, consider a voice recognition scenario: while it is usually easy to record speech, labeling the sound wave with its exact meaning (in terms of phonemes, for example) is a labor intensive task. In such situations, we would like to select the relevant instances before knowing their correct label. We would then find the label associated with each selected instance by making a so-called "query" to a teacher and add the example to the training set. The goal here is to minimize the number of queries made to the teacher. This framework has also been studied by Cohn [Cohn *et al.*, 1990] and is related to the problem of sequential experimental design [Lindley, 1956; Fedorov, 1972; Atkinson and Donev, 1992].

Seung et al. [Seung *et al.*, 1992] suggest a general instance selection method, which they call "Query by Committee" (QBC). Their method can be applied when there exists a distribution measure over the concept class. This learning framework is usually referred to as the Bayes learning framework [Haussler *et al.*, 1994].

In this Bayes prediction framework we assume that the concept to be learned is chosen according to a known distribution over the concept class, usually referred to as the "prior" distribution. We assume that this distribution is known to the learner in the sense

[3] It is know from other work that the dependence of the sample size (which is the total number of examples observed) on ϵ is $\Omega(1/\epsilon)$. [Blumer *et al.*, 1989; Ehrenfeucht *et al.*, 1989]

[4] The notation $\tilde{O}(\cdot)$ denotes the fact that we are ignoring log factors.

that the learner can generate a random concept from the prior distribution at will. While it is often unreasonable to assume knowledge of the exact prior distribution, the results described here hold even if a rough approximation of the prior distribution is used.

A simple version of the QBC algorithm, which was analyzed by Freund et al. [Freund *et al.*, 1992] can be described as follows. We say that a concept is "consistent" with a training set if it labels all of the examples in the set correctly. The final hypothesis generated by the QBC algorithm is a random consistent concept, chosen according to the prior distribution restricted to the consistent concepts. The selection of relevant instances is done as follows.

Given a random instance x,

1. **Select two consistent hypotheses, h_1 and h_2, independently at random according to the prior distribution over the concept class.**
2. **Accept x as a relevant instance if $h_1(x) \neq h_2(x)$.**

Notice the similarity of this rule to the selection rules used in the boosting algorithms. Mainly, all of these rules tend to select examples on which different hypotheses that are based on past experience predict different labels. While the frameworks in which these algorithms are used and analyzed are very different, this commonality seems clear and appealing on an intuitive level.

The analysis of this algorithm by Freund et al. shows that, in some natural cases, this selection rule has similar properties to that of the boosting algorithm. Specifically, if the desired expected error of the final hypothesis is $\epsilon > 0$, then the number of instances that have to be tested is $\tilde{O}(1/\epsilon)$, and the number of them that are accepted and presented as queries to the teacher is $O(\log 1/\epsilon)$.

Applying QBC to real-world problems is a promising research direction. While in most cases it is not easy to generate a random consistent hypothesis, the following approximation can sometimes be used.[5] Many learning algorithms, such as the Back-Propagation algorithm in neural networks, have a single hypothesis stored in memory, and continually change this hypothesis to improve its fit to the data. There are various random effects on this hypothesis, which include the choice of the initial hypothesis and random choices that are made during the learning process. It is a comparatively simple matter to duplicate the learning algorithm and maintain two hypotheses which have small random differences. Selecting instances according to the difference between these two hypotheses will tend to concentrate on those instances whose label cannot be determined by generalizing from the other examples. While the exact analysis of this algorithm is beyond the reach of current theory, experiments might shed light on their behavior.

[5] In some special cases efficient algorithms for selecting a random consistent hypothesis are known. For example, if the hypothesis class is the set of linear separators, then selecting a random consistent hypothesis amounts to selecting a random point from a convex polytope. Polynomial time algorithms for this problem have been found by Dyer, Frieze and Kannan [Dyer *et al.*, 1991] and had a series of improvements, last of which was given by Lovasz and Simonovits [Lovasz and Simonovits, 1993].

Summary and some possible applications

We have briefly presented two learning algorithms that suggest methods for selecting the relevant examples out of a stream of random examples. The boosting algorithms have been designed towards a different goal than the QBC algorithm, and their rigorous mathematical analysis is performed in different frameworks. However, there are two properties shared by all three algorithms:

1. All algorithms select examples on which previous hypotheses tend to disagree.
2. Under the proper assumptions, the dependence of the number of selected examples on the desired accuracy $\epsilon > 0$ is $O(\log 1/\epsilon)$, while the number of examples that have to be sifted is $O(1/\epsilon)$.

The main goal of boosting algorithms is to improve the performance of a weak learning algorithm, and the main goal of Query selection algorithms is to reduce the number of queries. We end our summary by suggesting a few new applications for this type of algorithms.

- **Reducing memory storage** Memory based learning algorithms, such as nearest neighbor classifiers, require a lot of memory. The methods suggested might be applied to save large amounts of space while causing only minimal degradation in classification.
- **Selective Labeling:** Today there is a lot of effort put into projects for labeling large amounts of random data, such as the NIST character database, and the TIMIT speech database. Selective choice of queries can improve the utilization of the great amounts of human effort put into these projects.
- **Parallel learning:** All of the algorithms for selecting examples have efficient parallel versions. Most of the computation time of these algorithms is spent on selecting the relevant examples, because (after some amount of training) most of the examples are judged to be irrelevant and filtered out. As the selection criteria depends only on a single example, it can be applied, in parallel, to many examples. If a sufficient number of parallel processors is available, the dependence of the running time on the desired accuracy $\epsilon > 0$ is (ignoring log factors) $\tilde{O}(\log 1/\epsilon)$.

- **Information Sharing:** One interesting case of parallel processing in learning is when several copies of a learning algorithm are run in different sites, and on different data. Suppose, for example, that a learning algorithm receives some initial training in a central development center, and then many clones of this algorithm are placed in different but similar environments to perform their work. Assume that these initial copies are good enough to perform useful work, but that additional, "on site" training can further improve their performance. Moreover, sharing the experience of such a network of algorithms can be used to increase the accuracy of all of them. Communicating examples that are judged to be relevant is a very simple and self-explanatory way of sharing experience. Moreover, as our algorithms select only very few examples as relevant, we expect this method of cooperation to require relatively little communication.

Acknowledgments

The author would like to thank Rob Schapire, Nick Littlestone and Avrim Blum for their helpful suggestions.

References

Atkinson, A. C. and Donev, A. N. 1992. *Optimum Experimental Designs.* Oxford science publications.

Blumer, Anselm; Ehrenfeucht, Andrzej; Haussler, David; and Warmuth, Manfred K. 1989. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery* 36(4):929–965.

Cohn, David; Atlas, Les; and Ladner, Richard 1990. Training connectionist networks with queries and selective sampling. In Touretzky, D., editor 1990, *Advances in Neural Information Processing Systems 2*, San Mateo, CA. Morgan Kaufmann.

Drucker, Harris; Schapire, Robert; and Simard, Patrice up . Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence* 7(4):705–719.

Drucker, Harris; Schapire, Robert; and Simard, Patrice 1993. Improving performance in neural networks using a boosting algorithm. In *Advances in Neural Informations Processing Systems 5*, San Mateo, CA. Morgan Kaufmann. 42–49.

Dyer, M.; Frieze, A.; and Kannan, R. 1991. A random polynomial time algorithm for approximating the volume of convex bodies. *Journal of the ACM, JACM* 38(1):1–17.

Ehrenfeucht, Andrzej; Haussler, David; Kearns, Michael; and Valiant, Leslie 1989. A general lower bound on the number of examples needed for learning. *Information and Computation* 82:247–261.

Fedorov, V. V. 1972. *Theory of Optimal Experiments.* Academic Press, New York.

Freund, Y.; Seung, H.S.; Shamir, E.; and Tishby, N. 1992. Information, prediction, and query by committee. In *Advances in Neural Informations Processing Systems 5*, San Mateo, CA. Morgan Kaufmann. 483–490.

Freund, Y. 1990. Boosting a weak learning algorithm by majority. In *Proceedings of the Third Workshop on Computational Learning Theory*, San Mateo, CA. Morgan Kaufmann. 202–216.

Freund, Y. 1992. An improved boosting algorithm and its implications on learning complexity. In *Proceedings of the Fifth Workshop on Computational Learning Theory*, San Mateo, CA. Morgan Kaufmann. 391–398.

Haussler, David; Kearns, Michael; and Schapire, Robert 1994. Bounds on the sample complexity of Bayesian learning using information theory and the VC dimension. *Machine Learning* 14(1):83–113.

Lindley, D. V. 1956. On a measure of the information provided by an experiment. *Ann. Math. Statist.* 27:986–1005.

Lovasz, and Simonovits, 1993. Random walks in a convex body and an improved volume algorithm. *Random Structures & Algorithms* 4.

Schapire, Robert E. 1990. The strength of weak learnability. *Machine Learning* 5(2):197–226.

Seung, H.S; Opper, M.; and Sompolinsky, H. 1992. Query by committee. In *Proceedings of the Fifth Workshop on Computational Learning Theory*, San Mateo, CA. Morgan Kaufmann. 287–294.

Valiant, L. G. 1984. A theory of the learnable. *Comm. ACM* 27:1134–1142.

Vapnik, V. N. 1982. *Estimation of Dependences Based on Empirical Data.* Springer-Verlag, New York.