Monte Carlo Theory as an Explanation of Bagging and Boosting

Roberto Esposito
Università di Torino
C.so Svizzera 185, Torino, Italy
esposito@di.unito.it

Lorenza Saitta
Università del Piemonte Orientale
Spalto Marengo 33, Alessandria, Italy
saitta@mfn.unipmn.it

Abstract

In this paper we propose the framework of Monte Carlo algorithms as a useful one to analyze ensemble learning. In particular, this framework allows one to guess when bagging will be useful, explains why increasing the margin improves performances, and suggests a new way of performing ensemble learning and error estimation.

1 Introduction

Ensemble Learning aims at combining learned hypotheses for classification tasks instead of choosing one. It has proved to be a valuable learning approach, and it is actively investigated in the Machine Learning community [Freund 1995; Freund & Schapire 1997; Breiman, 1996; Wolpert, 1992].

In addition to the advantages pointed out by Dietterich [2000], what really attracts the great interest to ensemble learning is its amazing effectiveness and robustness to overfitting. In fact, ensemble classifiers, and specifically AdaBoost, often show a drop in generalization error far beyond the point in which the training error reaches zero.

In this paper we concentrate on two approaches, namely *Bagging* [Breiman, 1996], and *Boosting,* as implemented in the *AdaBoost* algorithm [Freund & Schapire, 1997], and we propose an analysis of these two approaches by exploiting links with Monte Carlo theory. This parallel allows a deeper understanding of the basic process underlying hypothesis combination. In particular:

- it suggests the conditions under which Bagging is likely to be better than single hypothesis learning,
- it explains why reducing the margin [Schapire *et al.*, 1998] increases the performances,
- it allows some reasons for robustness to overfitting to be conjectured.

The article is organized as follows: in Section 2 we briefly review Bagging and AdaBoost algorithms, and in Section 3 we introduce basic terminology and results from Monte Carlo theory. Section 4 is the core of the paper: it introduces the link between Monte Carlo theory and Ensemble Learning algorithms, and analyzes the two algorithms in the ideal situation of having access to the whole example and hypothesis spaces. Section 5 relaxes the hypotheses of Section 4. In particular, it focuses on AdaBoost's case offering an intuitive accounting of the generalization performances of the algorithm. Finally, Section 6 concludes the paper and proposes future work.

2 Bagging and AdaBoost

Bagging [Breiman, 1996] is one among the most studied Ensemble Learning algorithms. It takes in input a learning set L_0, an integer number T, and a (possibly weak) induction algorithm $\mathcal{A}$. At each iteration step t the algorithm creates a bootstrap replicate L [Efron & Tibshirani, 1993] of the original learning set, and uses it in conjunction with $\mathcal{A}$ to form a new hypothesis h_t. When T hypotheses have been learned, they are combined using a simple majority voting rule.

Breiman [1996] found a significant increase of classification accuracy, when the hypotheses are bagged, on different datasets from the UCI repository, for both regression and classification tasks. Moreover, in the same paper, a theoretical analysis is carried on. The notion of *order correct* classifier is introduced, and it is shown that when $\mathcal{A}$ is order correct, then Bagging is expected to behave almost optimally (i.e., it behaves like the Bayes classifier). Throughout the paper it is claimed that "*bagging stable classifiers is not a good idea*". A "stable" classifier is qualitatively defined as one that does not show large variations in the learned hypotheses when the training set undergoes small variations. Experimental analysis of Bagging's performances is provided by Bauer and Kohavi [1999], Quinlan [1996], and Dietterich [2000].

The simplest version of AdaBoost [Freund & Schapire, 1997] deals with binary classification problems, and this is the one we refer to. AdaBoost maintains a weight distribution over the training examples, and adjusts it at each iteration. The adjusted weights are chosen so that

higher values are associated to previously misclassified examples. As a main effect, the weighting scheme forces A to focus on "more difficult" examples, ensuring thus that all the examples will be, sooner or later, taken into account by the weak learner. The hypothesis combination rule is a weighted majority vote that puts more weight on successful hypotheses (i.e., hypotheses that incur a low weighted training error).

Despite its apparent simplicity AdaBoost solves a difficult problem: choosing example and hypothesis weights in such a way to ensure an exponential decrease in the training error committed by the boosted committee. Moreover, the choice is made without any knowledge or assumption on the weak learner itself (except that the weak learner is always able to outperform random guessing). In contrast, Bagging only works when "benign" weak learners are used. On the other hand, AdaBoost seems to be more sensitive to noise than Bagging [Quinlan, 1996].

As mentioned earlier, AdaBoost shows the characteristic of being robust with respect to overfitting [Schapire et al., 1998; Drucker & Cortes, 1996; Quinlan, 1996; Breiman, 1998; Freund & Schapire, 1998]. A rather convincing account of this behavior imputes to AdaBoost the ability to be very successful in increasing the final classifier's margin over the training examples [Schapire et al., 1998].

In the effort of understanding and/or explaining AdaBoost's behavior, boosting has been related to other existing theories, such as Game Theory [Schapire, 2001], logistic regression [Friedman et al., 1998; Collins et al., 2000], optimization [Rätsch et al., 2002], and Brownian motion [Freund, 1995].

3 Monte Carlo Algorithms

In this section we recall some basic notions from Monte Carlo algorithms theory. In the literature, the name Monte Carlo often denotes a generic stochastic algorithm; in this paper we will refer to the more restrictive definition given by Brassard and Bratley [1988]. Let $\prod$ be a class of problems, Y a finite set of answers for the problems in $\prod$, and $c : \prod \to 2^Y$ be a function that maps each problem into the set of its correct answers.

Definition 1 – A stochastic algorithm A, mapping $\prod$ into Y, is said to be a *Monte Carlo* algorithm, if, when applied to an instance π of $\prod$, it always terminates, providing an answer to π, which may be occasionally incorrect.

Two properties of Monte Carlo algorithms turn out to be relevant:

Definition 2 – A Monte Carlo algorithm is *consistent* if it never outputs two distinct *correct* answers when run two times on the same problem instance π.

Definition 3 – A Monte Carlo algorithm is *p-correct* if the probability that it gives a correct answer to π is at least p, independently of the specific problem instance π considered.

The great interest in Monte Carlo algorithms resides in their *amplification* ability: given a problem instance π, and a Monte Carlo algorithm *MC*, if *MC* is run multiple times on π and the majority answer is taken as the result, then, under mild assumptions, the probability of a correct answer exponentially increases. The conditions are that *MC must be consistent for* π and that *p must be greater than random guess*. We note that, whenever $c(\pi)$ reduces to a mapping from $\prod$ to Y (instead of 2^Y), the algorithm is consistent. Let us denote by MC_T the algorithm obtained by combining with a majority vote the answers given by multiple runs of a Monte Carlo *MC*.

4 Complete Information Case

In this section we provide a theoretical account of the links between Monte Carlo theory and bagging or boosting, in the case we have complete information on the learning/classification problem. More precisely, let X be a finite set of examples[1], Φ a target concept (binary classification), A a (weak) learner. Let $|X| = N$ and w be a probability distribution over X. Let moreover Σ be the power set of X. Any intensional concept description, belonging to a given language $\mathcal{L}$, will be mapped to an element σ of Σ, i.e., σ is the set of examples classified by that description as belonging to Φ. In this way, we can reduce ourselves to a finite set of (extensional) hypotheses. Let $\Theta \subseteq \Sigma$ be any subset of Σ. Let $|\Theta| = K$. A probability distribution q is associated to the elements of Θ. The set Θ can be thought of as the collection of the hypotheses generated by a learning algorithm A applied to a number of training sets, or a set of hypotheses provided by an oracle. Learning can then be simulated by extracting with replacement elements from Θ according to q (each extraction corresponds to a run of A on some training set).

Let us represent the described situation by means of Table 1. Here, let $p_j(x_k) \in \{1,0\}$[2] be the probability that hypothesis θ_j correctly classifies example x_k. Let $p(x_k)$ be the average of such probabilities for x_k. Let r_j be the accuracy of hypothesis θ_j over the whole X.
By using the given probability distributions, we can compute the average accuracy for the whole table:

$$r = \sum_{k=1}^{N} w_k p(x_k) = \sum_{j=1}^{K} q_j r_j = \sum_{j=1}^{K} \sum_{k=1}^{N} q_j w_k p_j(x_k) \qquad (1)$$

[1] The finiteness of X is not essential; it is only assumed for the sake of simplicity.
[2] This assumption implies that the Bayes error is zero.

Table 1 – Theoretical setting

	Ω_1	….	Ω_j	….	Ω_K	
x_1	$p_1(x_1)$		$p_j(x_1)$		$p_K(x_1)$	$p(x_1)$
….						….
x_k	$p_1(x_k)$	….	$p_j(x_k)$	….	$p_K(x_k)$	$p(x_k)$
….						….
x_N	$p_1(x_N)$		$p_j(x_N)$		$p_K(x_N)$	$p(x_N)$
	r_1		r_j		r_K	r

Given a subset Z of X, let $||Z||$ the measure of Z w.r.t. $\mathbf{w}$, i.e.:

$$||Z|| = \sum_{x_k \in Z} w_k$$

Analogously, if $\Omega' \subseteq \Omega$, we define:

$$||\Omega'|| = \sum_{\Omega_j \in \Omega'} q_j$$

Clearly the measures $||Z||$ and $||\Omega'||$ reduce to $|Z|/N$ and $|\Omega'|/K$, respectively, if $\boldsymbol{q}$ and $\boldsymbol{w}$ are uniform distributions.

4.1 Monte Carlo and Bagging

We will now establish a link between Monte Carlo algorithms and Bagging. Let L_0 be a training set, and T an integer; let L be a generic bootstrap replicate of L_0. In this case, Ω is the set of all the different hypotheses that can be obtained by applying A to L_0's replicates. The frequencies with which each Ω_j is generated determine $\boldsymbol{q}$.

If we look at Table 1, Bagging proceeds, so to say, *by columns*, by extracting T columns from Ω and using them to obtain a majority vote on each x_k. Considering the global average accuracy r, or even the exact distribution of the r_j, it is not easy to say how many of the x_k will be correctly classified by the majority voting procedure, and, then, whether to perform Bagging will be rewarding. It is generally said [Breiman, 1996] that bagging is useful when A is "unstable". We will show later that this notion of "instability" can be made more precise.

Let us consider now a Monte Carlo interpretation of Table 1. To the contrary of Bagging, a Monte Carlo algorithm MC_T proceeds *by rows:* for each x_k (problem to be solved), it extracts T hypotheses Ω_j from Ω and classifies x_k by majority voting. This process is repeated for each row[3]. This procedure is necessary, because the components of the Monte Carlo voting must be independent, but the entries in each column are not. Monte Carlo theory tells us that the correctness of MC_T on x_k can be amplified if:

1. The T trials are independent
2. MC is consistent
3. $p(x_k) > 1/2$

Condition (1) is true by construction of the process (extraction with replacement of T hypotheses for each row). Condition (2) is true as long as the Bayes error is zero. Condition (3) may or may not be true, depending on the set Ω.

Let X_a be the set of examples for which $p(x_k) > 1/2$. According to the Monte Carlo theory, all these examples will be "amplified", i.e., they will be correctly classified in the limit (for $T \to \infty$). The application of MC_T to all the rows yields an asymptotic accuracy:

$$a = ||X_a|| \qquad (2)$$

By denoting by a_T the expected Monte Carlo accuracy when only a finite number T of extractions is performed, we obtain:

$$\lim_{T \to \infty} \Pr\{a_T = a\} = 1 \qquad (3)$$

Let now Ω_b be the set of hypotheses whose accuracy is greater than a. The value

$$b = ||\Omega_b|| \qquad (4)$$

is the probability that a hypothesis "better" than bagging is obtained in a single extraction. Then, in an infinite number of trials, such a hypothesis will occur with a frequency b. It is clear that, in the limit, Bagging is surely convenient only when when $\Omega_b = \varnothing$, because, in this case, there is no single hypothesis better than Bagging. When $\Omega_b \neq \varnothing$ a better hypothesis will surely appear, soon or later. The interesting case, however, occurs when only a finite number T of extractions is performed. Before approaching this case, let us make some considerations on what has been described so far.

Let us first notice that ensemble learning claims that amplification occurs when the r_j's (accuracy of each of the combined hypotheses) are greater than 1/2 (condition on the columns). Actually it may not be the case. What is really required is that, for each example x_k, more than 1/2 of the learned hypotheses are correct on it (condition on the rows). Then, Monte Carlo Theory suggests that Bagging would be successful when it is possible to find a subset of Ω such that $X_a = X$. Before looking with more details into the relationship between the r_j's and the $p(x_k)$'s, let us consider some extreme cases.

Let A be a weak learner that always returns the same hypothesis Ω, regardless of the training set. The lucky situation in which that single hypothesis Ω is quite accurate over the training set does not imply anything about the occurrence of Monte Carlo amplification. In fact, all the $p(x_k)$ are equal to 1 on those examples which are correctly classified by Ω, and 0 otherwise. In other words, whatever the accuracy of Ω, no amplification at all occurs. This is an example of a perfectly stable learner.

Let us consider a situation in which the global average r of the accuracy is a little more than 1/2. This value is at the same time the average of the last column and on the last row in Table 1, according to (1). For the sake of illustration, let $\boldsymbol{q}$ and $\boldsymbol{w}$ be uniform distributions. Let us consider the following three extreme situations (a de-

[3] Let us set apart, for the moment, the computational issues.

tailed analysis of several other interesting cases is provided by Esposito [2003]) :

(a) The $p(x_k)$'s in the last column are distributed in such a way that one half of the examples have $p(x_k)$ values close to one, and one half of the examples have $p(x_k)$ values close to zero. The r_j's in the last row, on the contrary, have all almost the same value around 1/2. (All hypotheses show the same performances and they cover almost the same examples)

(b) The $p(x_k)$'s in the last column have all almost the same value around 1/2, whereas the r_j's in the last row have, for one half, values close to one, and for the other half close to zero. (Some hypotheses are very good and other are very bad, but their coverage is uniformly distributed on the examples).

(c) Both distributions are uniformly valued around 1/2 (All the hypotheses are uncertain on all examples).

In case (a) half of the examples are amplifiable, but Bagging is not useful, because any single hypothesis classifies correctly half of the examples, so that nothing can be gained by the Bagging process.
In case (b), the number of amplifiable examples depends on how many of them have actually values a little over 1/2. But, even in the best case ($X_a = X$, i.e., $\gamma = 1$), Bagging is again not convenient. In fact, half of the single hypotheses have $r_j = 1$; then, in one over two extractions (on average) a perfect hypothesis is extracted.
In case (c), if the $p(x_k)$ values in the last columns are all a little greater than 1/2, then $\gamma = 1$. This is the case in which Bagging is convenient, because the single hypothesis have all accuracy close to 1/2. Actually, case (c) is the ideal case for Bagging, because amplification can be obtained with the minimum effort (with "bad" hypotheses we may obtain a large amplification).

The following theorem summarizes the above considerations and establishes a link between Bagging and Monte Carlo amplification (proof is provided by Esposito [2003]):

Theorem 1 – The amplified accuracy γ cannot be greater than $2r$, where r is the average accuracy of the bagged hypotheses. ❑

Then, even by bagging single hypotheses with accuracy less then 1/2 it is possible to obtain higher accuracies by Monte Carlo amplification. However, in order to reach $\gamma = 1$, the value of r must be at least 1/2. Even in this case, some of the combined hypotheses are allowed to have a accuracy less than 1/2, without hindering amplification, provided that $X_a = X$ and that r does not go below 1/2.

4.2 Monte Carlo and AdaBoost

Let us notice that, if the set Ω of hypotheses coincides with the power set 2^X (all possible hypotheses are considered), we are in the situation in which all $p(x_k) = 1/2$, $r = 1/2$, and $\gamma = 1/2$, so that no amplification is possible. This means that the learning procedure was not able to make any useful selection inside the hypothesis space. On the contrary, we would like that the weak learner has the nice property of focusing on the best hypotheses. In the previous section we showed that the basic Monte Carlo approach, implemented by Bagging, may fail due to a number of reasons; this happens, for instance, in the cases (a) and (b) of Section 4.1. Even if not explicitly stated, the underling problem is that the weak learner may not be suited enough for this kind of procedure. The question we are now trying to investigate is what to do when such a situation occurs, in particular when we do not want or can modify the weak learner. Then, is it possible to convert case (a) or (b) into case (c)? Or, said in other words, is it possible to convert a learning problem in which the Monte Carlo assumptions for amplification do not hold into one in which they do?

The solution to this problem can be found by thinking again of the role of the weak learner in the Monte Carlo process. As mentioned earlier, the learner has the duty to modify the distribution from which the hypotheses are drawn (i.e., the q_j of the columns) in such a way that "better" hypotheses are more likely to be drawn. Moreover, we can actually change the behavior of the weak learner through its input, i.e., by modifying the training set. In particular, we have to force the weak learner to extract hypotheses that make the minimum value of the $p(x_k)$'s grow larger than 1/2 (in order to satisfy the Monte Carlo conditions of amplifiability for all the examples). Since the weak learner is influenced by the training set, the solution is to modify the training set in such a way that examples with low $p(x_k)$ values are more represented, in the attempt to make the weak learner try harder to be successful on them. What we obtain with this reasoning is an AdaBoost-like algorithm. In other words, when the weak learner is unable to satisfy Monte Carlo conditions for amplification, AdaBoost forces it to satisfy them by weighting the examples. More precisely, AdaBost tries to increase the minimum $p(x_k)$ with the unconscious goal of extending X_a. In fact, when the weak learner satisfies relatively mild conditions, it has been shown by Schapire et al. [1998] that AdaBoost is actually effective in increasing the minimal *margin*:

Definition 4 – For any particular example x_k, the classification *margin* $\mu(x_k)$ for x_k is the difference between the weight assigned to the correct label and the maximal weight assigned to any single incorrect label.

In the case of binary classification, the margin is the difference between the weight assigned to the correct label and the weight assigned to the incorrect one.

Theorem 2 – In the case of binary classification, the following relation holds between the margin and the Monte Carlo $p(x_k)$-correctness value:

$$\mu(x_k) = \mathrm{p}(x_k) - (1 - p(x_k)) = 2p(x_k) - 1 \qquad (7)$$

The proof of Theorem 2 derives immediately from the definition of margin and $p(x_k)$. Formula (7) explains why increasing the margin improves the performances of boosting: in fact, increasing the minimum margin lets the cardinality of the set X_α increase, augmenting the number of examples for which the Monte Carlo conditions for amplifiability hold.

4.3 Non-Asymptotic Case

Let us consider a finite integer T. Let β_R denote the probability that a hypothesis "better" than Bagging is obtained in R extractions. In order to compare Bagging with hypothesis selection, let us consider a classification accuracy γ. Given Table 1, and given a number δ close to 0, let $T(\delta)$ be the minimum number of components of the bagged hypothesis such that:

$$\Pr\{\beta_{T(\delta)} \geq \gamma\} \geq 1-\delta \qquad (8)$$

If $\gamma > \varepsilon$, then probability (8) is zero.
Analogously, let Ω_γ be the set of hypotheses with $r_j > \gamma$ and $R(\delta)$ be the minimum number of hypothesis extractions such that:

$$\Pr\{\text{Extracting } \alpha_j \in \Omega_\gamma\} \geq 1-\delta \qquad (9)$$

Again, if $\Omega_\gamma = \varnothing$, probability (9) is zero.

We can draw a graph of T (ordinate) vs. R (abscissa), parameterized by γ. The graph is problem dependent. What we can say in general is that points in the graph above the diagonal correspond to situation for which selection is better than Bagging (for the same accuracy γ, Bagging requires more trials), points below the diagonal correspond to situation for which selection is worse than Bagging (for the same accuracy γ, Bagging requires less trials), and points on the diagonal correspond to situations of indifference.

5 Problems with Partial Information

The setting discussed in Section 4 is useful for understanding the functioning of ensemble learning, but we need to extend it to the case of real learning problems. In concrete situation, neither the set Ω nor the example space X is available *a priori*. What we have is a training set L_0 ($|L_0| = M$) and a test set T_0, both subsets of X, and a learning algorithm A for generating hypotheses. We learn hypotheses, possibly consistent with L_0, and estimate the generalization error on T_0[4].

If we consider the problem from the Monte Carlo perspective and we go back to Table 1, we can think to proceed as follows. At the beginning we have, on the rows, the training and the test examples, but the columns are empty. Let us consider the first example $x_1 \in L_0$. Let A run on T replicates of L_0 and let us compute $p_e(x_1)$, which is an unbiased estimate of the Bernoulli probability $p(x_1)$. Even though the T hypotheses generated are only used to classify x_1, and could then be discarded, nothing hinders from recording them in the first T (or less, if there is duplication) columns of Table 1. When we consider x_2, we repeat the whole procedure, running again A on T new replicates of L_0 and classifying x_2. Again we record the new generated hypotheses (or update the probability distribution $\boldsymbol{q}$ in case of repetitions). When we finish examining L_0, we have also filled the part of the table corresponding to the test set, which allows estimates $p_e(x_i)$ for the examples in the test set to be computed. Each $p_e(x_i)$ is based on $T\,M$ experimental values, which is the number of runs necessary to estimate the generalization accuracy using the leave-one-out method on a number M of training examples.

What is interesting is that, during learning, we also *learn* whether it is convenient or not to exploit Bagging for classifying future unseen examples. It also turns out, experimentally[5], that a much lower number of runs than $T \cdot M$ is very often sufficient to obtain a stable estimate of the sought probabilities.

5.1 AdaBoost's generalization behavior

In this section we discuss AdaBoost's generalization capabilities and their interpretation in the Monte Carlo framework. Having related the margin of an example x_k with the average probability that it is correctly classified by the hypotheses learned via the weak learner allows us to transfer results provided by Schapire et al. [1998] to the Monte Carlo framework. Interestingly enough, the link may be used to give a new, more intuitive, interpretation of the margin explanation of AdaBoost's generalization capabilities. Moreover a nice explanation of why the test error is often observed to decrease even after the training error reaches zero can be suggested.

To understand how generalization error can be interpreted in the Monte Carlo framework it is necessary to deepen a little the discussion about the link between ensemble learning and Monte Carlo algorithms. Let us choose the randomization procedure (whatever it is). Once this is done, the choice of the weak learner determines the distribution of the $p(x_k)$'s. In other words, when the two major parameters of the ensemble algorithm are fixed, we can think of $p(x_k)$ as a theoretical property of the examples.
Once the ensemble process starts, we compute the estimations $p_e(x_k)$. The accuracy of the estimates increases with the number of iterations performed. Whether the final rule will classify correctly or not any example, is a matter of two things: the value of the underlying exact

[4] In this paper we do not consider to use cross-validation or leave-one-out.

[5] The datasets used, as well as part of the experimental results, can be found at the URL:
http://www.di.unito.it/~esposito/mcandboost

probability $p(x_k)$, and how lucky we were in the choice of the rules used to perform the actual estimation.

The above argument might explain the finding of the test error decreasing, for AdaBoost, beyond the point in which training error reaches zero. In fact, as an effect of the learning bias introduced by the weak algorithm, it is likely that the $p(x_k)$'s corresponding to examples in the training set would be greater than the ones corresponding to the other examples. It is then likely that those examples would be correctly classified with high probability after a moderate number of iterations. On the contrary, other examples would require a greater number of iterations, and thus they may start to be correctly labelled well beyond the point where the training error reaches zero. In fact, even though any example with $p(x_k) > 1/2$ will eventually be correctly classified, the number of iterations necessary to see this effect to emerge strongly depends on T. The observed effect of the test error decreasing is hence due to those examples for which $p(x_k)$ is only slightly greater than 1/2. Initially, their $p(x_k)$ is possibly underestimated by the observed $p_e(x_k)$. For small values of T, $p_e(x_k)$ can be quite different from $p(x_k)$ and greater values of T are necessary to let the choice of hypotheses converge toward a set for which $p_e(x_k) \approx p(x_k)$ and, hence, to obtain a stable correct classification of x_k in the test set.

6 Conclusions and Further Work

In this paper we have proposed to analyze ensemble learning (in particular Bagging and, partly, AdaBoost) from the perspective of Monte Carlo algorithm theory. We have shown that this theory can shed light on some basic aspects of Bagging and boosting, and, also offers a new way to actually perform ensemble learning and error estimation. More research is nevertheless needed to deepen our understanding of the presented framework. In particular, it would be very interesting to investigate the relationships between the Monte Carlo explanation of Ensemble Learning and other well known ones (for instance, the bias-variance decomposition). Moreover, it would be very interesting to investigate how a particularly aggressive class of Monte Carlo algorithms, namely the one of biased Monte Carlos, relates to ensemble learning.

References

[Bauer and Kohavi, 1999]. E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning, 36:*105-139.

[Brassard and Bratley, 1988] G. Brassard and P. Bratley. *Algorithmics: theory and practice*. Prentice-Hall, Inc., 1988.

[Breiman, 1996] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[Breiman, 1998] L. Breiman. Arcing classifiers. The Annals of Statistics, 26(3):801– 849, 1998.

[Collins *et al.*, 2000] M. Collins, R. Schapire, and Y. Singer. Logistic regression, adaboost and bregman distances. In *Computational Learing Theory*, pages 158–169, 2000.

[Dietterich, 2000] T. Dietterich. Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857:1–15, 2000.

[Drucker and Cortes, 1996] Drucker and Cortes, Boosting decision trees. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems,Vol. 8,* (pp. 479–485), The MIT Press, 1996.

[Efron and Tibshirani, 1993] B. Efron and R. Tibshirani. *An introduction to the Bootstrap*. Chapman and Hall, 1993.

[Esposito, 2003] R. Esposito. *Analyzing Ensemble Learning in the Framework of Monte Carlo Theory*. Ph.D. Thesis, University of Torino, Dept. of Informatica, Italy.

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

[Freund and Shapire, 1997] Y. Freund and R. 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.

[Freund and Shapire, 1998] Y. Freund and R. Schapire. Discussion of the paper ``arcing classifiers'' by Leo Breiman. *The Annals of Statistics*, 26(3):824-832, 1998.

[Friedman *et al.*, 1998] J. Friedman, T. Hastie, and R. Tibshirani. *Additive logistic regression: a statistical view of boosting*. Technical report, Department of Statistics, Sequoia Hall, Stanford Univerity, July 1998.

[Quinlan, 1996] R. Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 725-730, 1996. AAAI Press / MIT Press

[Rätsch *et al.*, 2002] G. Rätsch, M. Warmuth, S. Mika, T. Onoda, S. Lemm_ and K. Müller. Barrier boosting. In *13th Conference on Computational Learning Theory, 2000*.

[Schapire, 2001] R. Schapire. Drifting games. *Machine Learning* 43(3): 265-291, 2001.

[Schapire *et al.*, 1998] R. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistic, 26(5):*1651–1686, 1998.

[Wolpert, 1992] D. Wolpert. Stacked Generalization, *Neural Networks*, 5, 241-259, 1992.