Explaining Bagging with Monte Carlo Theory

Roberto Esposito¹ and Lorenza Saitta²

¹ Università di Torino, Dipartimento di Informatica,
Corso Svizzera 185, 10149 Torino, Italy
`esposito@di.unito.it`

² Università del Piemonte Orientale, Dipartimento di Informatica,
Spalto Marengo 33, 15100 Alessandria, Italy
`saitta@mf.n.unipmn.it`

Abstract. In this paper we propose the use of the framework of Monte Carlo stochastic algorithms to analyze ensemble learning, specifically, *bagging*. In particular, this framework allows one to explain bagging's behavior and also why increasing the margin improves performances. Experimental results support the theoretical analysis.

1 Introduction

Ensemble learning has been brought to the attention of the Machine Learning (ML) community by Schapire [17], who suggested that combining several learned hypotheses instead of choosing one could be a winning strategy in classification/regression tasks. Since then, ensemble learning has been actively investigated [11, 12, 3, 19].

Advantages of ensemble learning have been discussed by Dietterich [5] in a recent paper. Among others, the amazing effectiveness and robustness to overfitting are responsible for attracting a great deal of interest. In fact, it often happens that the generalization error still decreases after the training error reaches zero [16, 6]. In the attempt to explain this appealing behavior, many theoretical and empirical works have tried to relate ensemble learning to results from other fields, such as Game theory [10, 12, 18], Logistic regression [13, 4], and Convex optimization [15].

In this paper we limit ourselves to a specific ensemble learning algorithm, i.e., *Bagging* [3], and we propose to analyze its properties from the point of view of Monte Carlo algorithms theory. This new parallel sheds some light on the foundations of bagging. In particular:

- it suggests the conditions under which bagging is likely to be better than single hypothesis learning,
- it explains why margin reduction increases performances, and
- it allows some reasons for robustness to overfitting to be conjectured.

Bagging [3] is one among the most studied ensemble learning algorithms. It takes in input a learning set $\mathbf{L}_0$, an integer number T , and a (possibly weak) induction algorithm $\mathcal{A}$. At each iteration t the algorithm creates a bootstrap [7]

replicate $\mathbf{L}$ of the original learning set, and uses it as input to $\mathcal{A}$ to form a new hypothesis h_t . When T hypotheses have been learned, they are combined using a simple majority voting rule.

Experimental analysis of bagging’s performances is provided by Bauer and Kohavi [1], Quinlan [14] and Dietterich [5].

Breiman [3] found a significant increase of the classification accuracy, when the hypotheses are bagged, on different datasets from the UCI repository, for both regression and classification. 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.

2 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 [2].

Definition 1. *Let Π be a class of problems, $\mathcal{Y}$ a finite set of answers for the problems in Π , and $c : \Pi \rightarrow 2^{\mathcal{Y}}$ be a function that maps each problem into the set of its correct answers. A stochastic algorithm $\mathcal{A}$ mapping Π into $\mathcal{Y}$ is said to be a Monte Carlo algorithm if it always terminates, when applied to an instance $x \in \Pi$, providing an answer to x , which may occasionally be incorrect.*

The two following properties of Monte Carlo algorithms turn out to be relevant to the goals of this paper.

Definition 2. *A Monte Carlo algorithm is consistent if it never outputs two distinct correct answers for the same problem x .*

We may notice that, whenever $c(x)$ reduces to a mapping from Π to $\mathcal{Y}$ (instead of $2^{\mathcal{Y}}$), the algorithm is consistent.

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

Definition 4. *Given a p -correct Monte Carlo algorithm, the difference $\gamma = p - 1/2$ is called the advantage of the algorithm.*

The great interest in Monte Carlo algorithms resides in their *amplification* ability.

Theorem 1. (Brassard and Bratley [2]) *Let Π be a class of problems, and let MC be a Monte Carlo algorithm both consistent and p -correct over Π . By running MC T times on any instance $x \in \Pi$ and taking the majority answer as result, the probability that x is correctly classified increases exponentially with T , provided that p is greater than $1/2$.*

In the next section we will apply the notions introduced so far to learning.

3 A Monte Carlo Ensemble Learner

For the sake of simplicity, we only consider, in this paper, binary classification problems, i.e., $\mathcal{Y} = \{+1, -1\}$. Let $\mathcal{X}$ be a finite set of examples³, c a target concept, and $\mathcal{A}$ a (weak) learner. Elements of $\mathcal{X}$ labelled $+1$ are the positive examples of c , whereas elements labelled -1 are the negative examples. Let $|\mathcal{X}| = N$ and let $\mathbf{w}$ be a probability distribution over $\mathcal{X}$. Let moreover $2^{\mathcal{X}}$ be the power set of $\mathcal{X}$. The learner $\mathcal{A}$, using a learning set $\mathbf{L}$, acquires an intensional description λ of c , expressed in a given *hypothesis language*. Each description λ classifies the elements of $\mathcal{X}$ partly as positive and partly as negative. Let φ be the *extension* of λ , i.e., the set of examples that it classifies as positive. Obviously $\varphi \in 2^{\mathcal{X}}$. Let us now consider a set $\mathcal{L}$ of training sets. We can define $\mathcal{A}$ as a function from training sets to (extensional) hypotheses:

$$\mathcal{A} : \mathcal{L} \rightarrow 2^{\mathcal{X}} \quad (1)$$

Expression (1) represents the composition of two functions: the first one maps training sets into formulas of the hypothesis language, whereas the second one maps formulas to their extensions. As several descriptions in the hypothesis language may share the same extension, function (1) induces over the elements of $2^{\mathcal{X}}$ a probability distribution $\mathbf{q}$, which depends upon the elements in $\mathcal{L}$, and upon their probability distribution. We notice that, even though the description language may be infinite, the codomain of $\mathcal{A}$ in (1) is always finite, because $|2^{\mathcal{X}}| = 2^N$. Finally, given $\mathcal{L}$, it is possible that not all the elements of $2^{\mathcal{X}}$ be reachable, i.e., it is possible that $\mathcal{A}$ can only generate a proper subset Φ of $2^{\mathcal{X}}$. Then, we may think of $\mathbf{q}$ as a probability distribution over Φ , being the probabilities associated to the elements of $2^{\mathcal{X}} - \Phi$ equal to zero.

As we are neither interested in the specific working of $\mathcal{A}$ nor in the description language, let us abstract away, for the moment, the details of the learning process, and consider as given by an oracle the set Φ of hypotheses and the associated probability distribution $\mathbf{q}$. Let $|\Phi| = R$. Then, learning can be simulated by extracting with replacement from Φ hypotheses according to $\mathbf{q}$. In the following we will also consider as given the whole set $\mathcal{X}$ of examples, together with its distribution $\mathbf{w}$. This is clearly an ideal case in which complete information is available.

Let us represent the described situation by means of the matrix $\mathcal{M}$ reported in Table 1. Given a hypothesis φ_j and an example x_k , the classification

³ The finiteness of $\mathcal{X}$ is not essential; it is only assumed for the sake of simplicity.

$\varphi_j(x_k) \in \mathcal{Y} = \{+1, -1\}$ assigned by φ_j to x_k may be either correct or incorrect. Let $p_j(x_k) \in \{0, 1\}$ ⁴ be the probability that hypothesis φ_j correctly classifies example x_k , i.e., that $\varphi_j(x_k) = c(x_k)$. Let $p(x_k)$ be the average of such probabilities for x_k , and r_j the accuracy of hypothesis φ_j over the whole $\mathcal{X}$ (r_j is the true error rate of φ_j). We have:

$$p(x_k) = \sum_{j=1}^R q_j p_j(x_k) \quad r_j = \sum_{k=1}^N w_k p_j(x_k) \quad (2)$$

Table 1. Matrix $\mathcal{M}$, describing the learning problem setting

	φ_1		φ_j		φ_R	
x_1	$p_1(x_1)$		$p_j(x_1)$		$p_R(x_1)$	$p(x_1)$
....						
x_k	$p_1(x_k)$		$p_j(x_k)$		$p_R(x_k)$	$p(x_k)$
....						
x_N	$p_1(x_N)$		$p_j(x_N)$		$p_R(x_N)$	$p(x_N)$
	r_1		r_j		r_R	r

We can compute the average accuracy for the whole table:

$$r = \sum_{k=1}^N w_k p(x_k) = \sum_{j=1}^R q_j r_j = \sum_{j=1}^R \sum_{k=1}^N q_j w_k p_j(x_k) \quad (3)$$

In order to draw a parallel between Monte Carlo algorithms and ensemble learning, we note that the set Π coincides with $\mathcal{X}$; in fact, each example $x_k \in \mathcal{X}$ is a problem to solve. Moreover, $\mathcal{Y} = \{+1, -1\}$ is the set of answers. In order to facilitate the analysis, let us permute the rows and the columns of $\mathcal{M}$ in such a way that the $p(x_k)$ values decrease from top to bottom and the r_j values decrease from left to right. This simply amounts to rename both the examples and the hypotheses. Let us define the following Monte Carlo algorithm $\mathbf{MC}(x_k|\Phi, \mathbf{q})$, applied to a problem x_k :

```

MC( $x_k|\Phi, \mathbf{q}$ )
  Extract  $\varphi_j$  from  $\Phi$  according to  $\mathbf{q}$ 
  Classify  $x_k$  with  $\varphi_j$ 
  Return  $\varphi_j(x_k)$ 

```

⁴ This assumption implies that the Bayes error is zero.

As $p(x_k)$ is the probability of extracting a hypothesis φ_j that correctly classifies x_k , **MC** is $p(x_k)$ -correct. Let $\gamma_k = p(x_k) - 1/2$ be the advantage of **MC**. If we make T independent calls to **MC** and take the majority answer, we can let the advantage γ_k become as close to $1/2$ as desired, provided that:

- **MC** is consistent
- the calls to **MC** are independent from each other,
- $p(x_k) > 1/2$.

The first condition is true as long as the Bayes error is zero (in this case x_k has only one correct label). The second condition is more tricky: the calls to **MC** are independent by definition, but nothing assures us that the φ_j 's themselves are not related each other. This depends on the way $\mathcal{A}$ exploits the learning set to generate the hypotheses. We can expect that the more independent the φ_j 's are, the better the repeated application of **MC** will provide results close to the theoretical predictions.

Let now $\mathcal{X}_a = \{x_k | p(x_k) > 1/2\}$ be the subset of $\mathcal{X}$ containing the *amplifiable* examples, i.e., those that will be correctly classified in the long run.

If we define by $\pi_t(x_k)$ the probability that x_k is correctly classified by T calls to **MC**, we obtain:

$$\pi_T(x_k) = \sum_{t=\lfloor \frac{T}{2} \rfloor + 1}^T \binom{T}{t} p(x_k)^t (1 - p(x_k))^{T-t} \quad (4)$$

Then

$$\forall x_k \in \mathcal{X}_a : \lim_{T \rightarrow \infty} \pi_T(x_k) = 1 \quad (5)$$

In Figures 1,2, and 3, some graphs of $\pi_T(x_k)$ versus T are reported. From these figures one can see that $p(x_k)$ has a strong effect on the number of **MC** calls needed to let $\pi_T(x_k)$ increase.

From (5) it is clear that the probability measure

$$\|\mathcal{X}_a\| = \sum_{x_k \in \mathcal{X}_a} w_k =_{\text{def}} \rho_\infty$$

is the upper bound of the asymptotic accuracy. This accuracy will be 1 when $\mathcal{X}_a = \mathcal{X}$.

By using the notions introduced so far, we can define the following Monte Carlo ensemble learning algorithm, **AmpMC**:

```

AmpMC( $\Phi, \mathcal{X}, T$ )
  Extract with replacement  $x_k$  from  $\mathcal{X}$  according to  $\mathbf{w}$ 
   $H_T = 0$ 
  for  $t = 1, T$  do
     $h_t = \mathbf{MC}(x_k | \Phi, \mathbf{q}) = \varphi_t(x_k)$ 
     $H_T = H_T + h_t$ 
  end
  if  $H_T > 0$  then  $y(x_k) = 1$  else  $y(x_k) = -1$ 

```

Fig. 1. Graph of $\pi_T(x_k)$ vs. T for $p(x_k) = 0.7$.

Fig. 2. Graph of $\pi_T(x_k)$ vs. T for $p(x_k) = 0.5$.

Fig. 3. Graph of $\pi_T(x_k)$ vs. T for $p(x_k) = 0.3$.

It is important to point out that algorithm **MC** runs T times over each example x_k , and the results of the runs on previous examples are forgotten. In a sense, **AmpMC** works “by rows” on $\mathcal{M}$, and the classification accuracy of the base learner is not taken into consideration explicitly, but only indirectly through (3). On the contrary, bagging works, so to say, “by columns”, by extracting T columns from Φ and using them to obtain a majority vote on each x_k . Considering the global average accuracy r , or even the single r_j ’s, it is not immediate to say how many of the x_k ’s will be correctly classified by the majority voting procedure.

By establishing a link between bagging’s and **AmpMC**’s behaviors, we want to see whether this latter can explain (or better, predicts) the former. In this paper we limit ourselves to the explanation aspect. If we extract T columns φ_t ($1 \leq t \leq T$) from Φ , independently and without replacement, we see that, for each $x_k \in \mathcal{X}$ the conditions 1) and 2) required by the Monte Carlo theory are still verified, but condition 3) is not guaranteed, not even in the case that all r_t are greater than $1/2$. In fact, as the r_t ’s and the $p(x_k)$ ’s are only loosely coupled by (3), it is possible that $\mathcal{X}_a$ be a proper subset of $\mathcal{X}$ and, then, not all examples can be amplified. It is easy to prove that ρ_∞ can reach at most the value $2r$ [8].

Thus, even with single hypotheses with accuracy less than $1/2$ it is possible to obtain higher accuracies by Monte Carlo amplification. However, in order to reach $\rho_\infty = 1$, the value of r must be at least $1/2$. Even in this case, some of the component hypotheses are allowed to have an accuracy less than $1/2$, without

hindering amplification, provided that $\mathcal{X}_a = \mathcal{X}$ and that r does not go below $1/2$.

4 Predicting Bagging's behavior

In previous sections we claimed that Bagging can be viewed as the application of the Monte Carlo algorithms theory to Machine Learning. Here we use the Monte Carlo theory to estimate the empirical error of the bagged classifier at time t , which is important in two different ways. First, experiments show that the estimate is indeed *very* accurate, offering, thus, an empirical support to the claim we made. Second, it has a significant theoretical meaning, since it makes clear what process is going on as learning proceeds. The estimate assumes that $\mathbf{p} = (p(\mathbf{x}_1), p(\mathbf{x}_2), \dots, p(\mathbf{x}_m))$ is known. Actually, in practice $\mathbf{p}$ is, in turn, estimated empirically.

Let us denote with $\mathcal{D}(p)$ the probability distribution of $p(\mathbf{x})$. Monte Carlo theory explains Bagging's behavior in terms of $\mathcal{D}(p)$. In particular, when the weak learning algorithm has been chosen, the distribution of $p(\mathbf{x})$ can be considered as a “property” of the examples themselves and Bagging as a way to “discover” such a property. As a consequence, the error committed by the combined classifier when the number of combined hypotheses is T can be decomposed into three parts (let us denote with $\hat{p}(\mathbf{x})$ the observed $p(\mathbf{x})$ after T iterations of the weak learning algorithm):

- the final accuracy ρ_∞ of the Monte Carlo process, i.e., the accuracy of the combined classifier when $T \rightarrow \infty$. In terms of $\mathcal{D}(p)$, this term can be written as:

$$\rho_\infty = \int_{p > 0.5} \mathcal{D}(p) dp$$

- A correction ρ_T^- to the final accuracy, given by the fraction of examples for which $p(\mathbf{x}) \leq 0.5$, but which, with T hypotheses, are likely to be overestimated so that the apparent $\hat{p}(\mathbf{x})$ is greater than 0.5.
- A correction ρ_T^+ to the final accuracy, given by the fraction of examples for which $p(\mathbf{x}) > 0.5$, but which, with T hypotheses, are likely to be underestimated so that the apparent $\hat{p}(\mathbf{x})$ is less than 0.5.

By summarizing, the expected accuracy of the Bagging algorithm with T hypotheses is:

$$\rho_T = \rho_\infty - \rho_T^+ + \rho_T^- \quad (6)$$

Since the value ρ_∞ is fixed in advance and does not change, as more hypotheses are combined, the drop in the error of the bagged hypothesis is explained because the other two terms tend to zero. Actually, it is not so hard to reduce ρ_T^+ and ρ_T^- to something that is simple to compute, but we omit the details here for the sake of simplicity.

The above argumentation builds upon the assumption that nothing more than a Monte Carlo process is going on during the learning process. A preliminary experimentation has been performed in order to test the above conjecture.

The experiment consists of two parts. In the first one, we acquire $T = 1000$ hypotheses using the Bagging algorithm. The training error of the bagged hypothesis is measured during the learning process for each $1 \leq T \leq 1000$. At the end of the process, for $T = 1000$, we use the collected hypotheses to estimate the distribution $\mathcal{D}(p)$ of $p(\mathbf{x})$.

In the second phase, we perform a straightforward amplification of a Monte Carlo algorithm for which the probability to solve a particular problem follows $\mathcal{D}(p)$, i.e., the distribution that we estimated in the first part of the experiment. In particular, we built a Monte Carlo procedure $\mathbf{MC}(p)$ which simply extracts at random a number in $[0..1]$ and pretends to be correct whenever that number is less than p . The amplified version of the algorithm performs the experiment T times and is correct whenever more than $T/2$ experiments were successful. As we did for Bagging, we measure the accuracy of the algorithm for $1 \leq T \leq 1000$.

Figures 4 and 5 show the errors of the two algorithms plotted against increasing T values. The two experiments were performed using the “ionosphere” dataset from Irvine’s repository and a two dimensional, noise free, dataset that we call “Asia”. In both cases the weak learning algorithm is a simple CART-like algorithm⁵.

In [8] and [9] the correspondence between Bagging and Monte Carlo theory has been exploited also to estimate Bagging’s generalization error, and a much larger experimentation is presented. Here, we want to emphasize the meaning of the presented experiments. In particular, it is important to note that while the first experiment has access to all learning parameters (such as the weak learner, the hypotheses it generates, the training set, etc.), the second one can access only few of them, namely the training set size, and the distribution $\mathcal{D}(p)$. Moreover, we note that $\mathcal{D}(p)$ is related to the final bagged hypothesis only. In fact, it reports, for each $p \in [0..1]$, the fraction of examples for which the observed frequencies at $T = 1000$ is p . While such information allows one to immediately evaluate the accuracy of the final bagged classifier, its error when $T \neq 1000$ cannot be evaluated (as far as we know) as a function of $\mathcal{D}(p)$.

5 Monte Carlo interpretation of examples’ margin

In Ensemble Learning the notion of margin has been introduced for the first time by Schapire et al. [16] as an explanation of the effectiveness of their ADABOOST algorithm. They define the weight assigned to a label as the (weighted) fraction of hypotheses that output that label.

Definition 5. (Schapire et al. [16]) *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.*

⁵ In the implementation used, the algorithm stops to grow the tree as soon as eighty percent of the examples are covered.

Fig. 4. Plot of the errors of a simulated Monte Carlo algorithm and the errors of Bagging on the “Ionosphere” dataset.

Fig. 5. Plot of the errors of a simulated Monte Carlo algorithm and the errors of Bagging on an artificial dataset.

Theorem 2. *In the case of binary classification, the following relation holds between the margin and the Monte Carlo $p(\mathbf{x})$ value:*

$$\mu(\mathbf{x}) = p(\mathbf{x}) - (1 - p(\mathbf{x})) = 2p(\mathbf{x}) - 1$$

The proof of Theorem 2 derives immediately from the definition of margin and $p(\mathbf{x})$. The theorem directly explains why increasing the margin improves the performances of boosting: in fact, increasing the minimum margin lets the cardinality of the set $\mathcal{X}_a$ increase, augmenting the number of examples for which the Monte Carlo conditions for amplifiability hold.

Interestingly, formula (6) implies that Bagging does not increase the margin at all. More precisely, it “uncovers” the margin of each example. This is rather different from ADABOOST approach, which directly maximizes the minimal margin over the examples.

6 Conclusion

In this paper we have proposed to analyze ensemble learning (in particular Bagging) from the perspective of Monte Carlo algorithms theory. In particular, after introducing basic Monte Carlo theory notions, we showed that these notions, interpreted from a Machine Learning perspective, directly lead to the Bagging algorithm. The paper then focused on the implications of the link. In particular, it emphasized that the real important parameter is the probability with which each example is correctly classified, and explained why reducing the margin increases the algorithm performances. Finally, the paper offered a preliminary experimentation supporting the claim that Bagging is indeed nothing more than a Monte Carlo algorithm.

References

1. Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139, 1999.
2. Gilles Brassard and Paul Bratley. *Algorithmics: theory and practice*. Prentice-Hall, Inc., 1988.
3. Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
4. Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, adaboost and bregman distances. In *Computational Learning Theory*, pages 158–169, 2000.
5. Thomas G. Dietterich. Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857:1–15, 2000.
6. Harris Drucker and Corinna Cortes. Boosting decision trees. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 479–485. The MIT Press, 1996.
7. B. Efron and R. Tibshirani. *An introduction to the Bootstrap*. Chapman and Hall, 1993.

8. Roberto Esposito. *Analyzing Ensemble Learning in the Framework of Monte Carlo Theory*. PhD thesis, Dipartimento di Informatica, Università di Torino, C.so Svizzera 185, 10149 Torino, Italy, 2003.
9. Roberto Esposito and Lorenza Saitta. Monte Carlo Theory as an Explanation of Bagging and Boosting. In *Proceeding of the Eighteenth International Joint Conference on Artificial Intelligence (to appear)*, 2003.
10. Y. Freund and R. Schapire. Game theory, on-line prediction and boosting. In *Proceedings, 9th Annual Conference on Computational Learning Theory*, pages 325–332, 1996.
11. Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pages 148–146. Morgan Kaufmann, 1996.
12. Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 1(55):119–139, 1997.
13. J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. Technical report, Department of Statistics, Sequoia Hall, Stanford University, 1998.
14. J. R. Quinlan. Bagging, boosting and c4.5. In *Proc. of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 725–730. AAAI Press / MIT Press, 1996.
15. Gunnar Rätsch, Manfred Warmuth, Sebastian Mika, Takashi Onoda, Steven Lemm, and Klaus-Robert Müller. Barrier boosting. In *Thirteenth Annual Conference on Computational Learning Theory*, 2000.
16. R. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26(5):1651–1686, 1998.
17. Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.
18. Robert E. Schapire. Drifting games. In *Computational Learning Theory*, pages 114–124, 1999.
19. David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.