Knowledge Discovery Via Multiple Models

Pedro Domingos
Sec. Sistemas, Dept. Eng. Mecânica
Instituto Superior Técnico
Av. Rovisco Pais
Lisbon 1096, Portugal
E-mail: pedrod@gia.ist.utl.pt

Abstract

If it is to qualify as knowledge, a learner's output should be accurate, stable and comprehensible. Learning multiple models can improve significantly on the accuracy and stability of single models, but at the cost of losing their comprehensibility (when they possess it, as do, for example, simple decision trees and rule sets). This article proposes and evaluates CMM, a meta-learner that seeks to retain most of the accuracy gains of multiple model approaches, while still producing a single comprehensible model. CMM is based on reapplying the base learner to recover the frontiers implicit in the multiple model ensemble. This is done by giving the base learner a new training set, composed of a large number of examples generated and classified according to the ensemble, plus the original examples. CMM is evaluated using C4.5RULES as the base learner, and bagging as the multiple-model methodology. On 26 benchmark datasets, CMM retains on average 60% of the accuracy gains obtained by bagging relative to a single run of C4.5RULES, while producing a rule set whose complexity is typically a small multiple (2–6) of C4.5RULES's, and also improving stability. Further studies show that accuracy and complexity can be traded off by varying the number of artificial examples generated.

Keywords: Classification, multiple models, meta-learning, comprehensible models, stability.

1 Introduction

Data mining seeks to extract useful knowledge from databases. Because of this concern with knowledge, rather than simply accurate prediction, research in this area places a high value on algorithms that produce comprehensible output. There are also practical reasons for this. In many (if not most) applications, it is not enough for a learned model to be accurate; it also needs to be understood by its human users, if they are to trust it and deem it acceptable. Also, users often wish to gain insight into a domain, rather than simply obtain an accurate classifier for it, and this is possible only if they are able to make sense of the learner's output. Even when predictive accuracy is the sole goal, comprehensibility is an important asset for a learner, because it facilitates the process of interactive refinement that is at the heart of most successful applications.

Because machine learning seeks to capture a broad spectrum of knowledge, from the commonsense to the expert-level, it has also tended to focus on representations—like decision trees, rule sets and belief networks—that are more powerful than those traditionally found in statistics and pattern recognition. This flexibility, while essential to the field's goals, has the disadvantage that it allows learners to be overly responsive to the training data, producing models that can change dramatically with small changes in the data. This instability undermines their claim to producing knowledge. As Turney and coauthors found when working on industrial applications of decision tree learning, "the engineers are disturbed when different batches of data from the same process result in radically different decision trees. The engineers lose confidence in the decision trees, even when we can demonstrate that the trees have high predictive accuracy." [32]. Other researchers have also noted the negative impact of instability on learners' ability to produce knowledge [12].

Recently, an approach that mitigates this problem has been the object of much research (see, for example, [6]). It consists of learning several (say, fifty) different models by means of variations in the learner or the data, and then combining these models in some way to make predictions. Different forms of this "multiple models" approach include bagging [3], boosting [18], stacking [34], Bayesian averaging [4], error-correcting output coding [22], combiner trees [7], and others. This approach has been found to be quite effective in practice [14, 29], and also has substantial theoretical foundations [23, 19]. However, the focus of this line of research has been on reducing instability as a means to improving accuracy; from the point of view of knowledge acquisition, it in fact represents a setback, because it gives up the essential goal of output comprehensibility. While (for example) a single decision tree can easily be understood by a human as long as it is not too large, fifty such trees, even if individually simple, exceed the capacity of even the most patient user; and this is aggravated by the fact that, to understand the ensemble's prediction, it is necessary in addition to understand (and keep track of) how the trees will interact at performance time.

Thus, while significant progress has been made towards separately achieving each of the three goals (accuracy, stability and comprehensibility), the overarching one of attaining all three simultaneously remains elusive. This article aims to move closer to this ideal—or at least to further explore the space of trade-offs among the three subgoals—by proposing a learning method that combines some of the accuracy and stability gains of multiple models with the comprehensibility of a single model. The proposed approach is described in the next section. It is then evaluated on a large number of benchmark datasets. The article concludes with a short review of related work, and an outline of directions for future research.

2 The CMM Algorithm

2.1 Basic Framework

In classification problems, a model is (implicitly or explicitly) a division of the instance space into regions, each of which is assigned to one of the possible classes. A learner is accurate to the extent that the division into class regions it produces, given a set of training examples, coincides with the "real" one; it is stable to the extent that it produces the same division into regions given two different training sets from the same domain; and its output is comprehensible insofar as it states clearly (to a human user) what the class regions are.

Suppose L is a classification learner that produces models in a comprehensible representation (e.g., rule sets), but is also unstable. Then it may be possible to improve its accuracy and stability by applying it $m \gg 1$ times, either to m different training sets[1] or with m different versions of L itself[2], and combining the resulting m models in some form. This combination can be carried out at either learning or performance time, but in either case it effectively results in assigning a class label to each possible instance, and thus to once again partitioning the instance space into class regions. In other words, once combined, the multiple models constitute again a single model. However, the class regions in this new model are, in general, a function of all the individual models and of how they are combined, and the clear division that might have been present in each one of them is lost.

This lost comprehensibility can potentially be recovered in the following way. Just as L can be applied to model the "true" partitioning of the instance space into class regions, by learning from previously-collected training examples, it can be applied to model the partitioning produced by the combined models, by learning from a set of randomly generated examples, whose classes are those predicted by the combined models. Because L produces models in a comprehensible representation, the resulting "meta-learned" model will also be comprehensible, as long as it is not too complex. Since the partitioning produced by combining m models is likely to be more complex than that produced by each of the component models, the meta-learned model is also likely to be more complex than them; but not necessarily to such an extent that it is rendered incomprehensible, especially since it is only an approximation to the exact partitioning produced by the combined models. On the other hand, because of this fact, the meta-learned model is also likely to be less accurate than the combined one, especially if the representational power of the language of combined models is greater than that of the component model language. However, and crucially, because the accuracy and stability of learned models tend to increase with training set size (due to decreasing variance [21]), and the training set size for the meta-learning step can be made as large as desired, subject to computational resource constraints, it should be possible to obtain a meta-learned model that is more accurate and stable than the base models.

[1]Where two different training sets can be composed of the same instances, but with different weights, as in boosting [18].

[2]Which, in the limit, can be m different learners, as is typically done in stacking [34].

Table 1: The CMM meta-learning algorithm.

```
Inputs:
    $S$ is the training set,
    $L$ is a learning algorithm,
    $C$ is a procedure for combining models,
    $m$ is the no. of component models to generate,
    $n$ is the no. of new examples to generate.

Procedure CMM $(S, L, C, m, n)$

For $i = 1$ to $m$
    Let $S_i$ be a variation of $S$.
    Let $M_i$ = Model produced by applying $L$ to $S_i$.

For $j = 1$ to $n$
    Let $\vec{x}$ be a randomly generated example.
    Let $c$ be the class assigned to $\vec{x}$ by $C_{M_1,\ldots,M_m}(\vec{x})$.
    Let $S = S \cup \{(\vec{x}, c)\}$.

Let $M$ = Model produced by applying $L$ to $S$.

Return $M$.
```

The procedure just described, which will be called CMM (for "Combined Multiple Models"), is shown in pseudo-code in Table 1.[3] Two significant points should be noted. One is that, to the extent possible, the examples for the meta-learning phase should be generated from the same distribution as the "true" training examples, since many learners are sensitive to this distribution, and could be misled by (for example) uniform sampling. More precisely, if x is an unclassified example and c its class, since the "true" probability distribution $Pr(x)$ is usually unknown, it should be estimated as closely as possible. While many general methods exist for approximately solving this problem (e.g., Parzen windows [15]), in the implementation described below the approach followed is one of replicating the way the learner L implicitly models $Pr(x)$. This avoids a mismatch between the bias of L and that of the probability estimation procedure.

The other point to be noted is that, in CMM, the training set for meta-learning is composed of the artificially generated examples *and* the original ones. This recognizes the fact that the combined model is itself only an approximation to the "true" partitioning of the instance space into class regions, and thus the meta-learning phase may benefit from taking into consideration the original examples.

[3]Instead of variations of S, variations of L may be used. Note that this procedure is quite different from stacking (Wolpert, 1992), where the final output is a two-level classifier that explicitly includes all the models produced, plus a meta-classifier to combine their predictions at performance time.

2.2 An Implementation

In order to test and apply the ideas above, an implementation of CMM was carried out using a specific learner L and a specific multiple-model methodology. C4.5RULES [28] release 8 was used as the base learner. C4.5RULES produces propositional rule sets, which we believe to be the most easily understood of all representations currently in use. C4.5RULES also has the advantage of being widely used, and thus constituting a good standard for empirical comparisons. In this system, rules are extracted from a previously-learned decision tree, and are ordered (i.e., if more than one rule applies, the one appearing first in the ordering prevails). Bagging [3] was used as the multiple-model methodology, on account of being perhaps the simplest one available, and of its effectiveness with decision trees being well established [3, 29]. In the bagging procedure, given a training set of size s, a "bootstrap" replicate of it is constructed by taking s samples *with replacement* from the training set. Thus a new training set of the same size is produced, where each of the original examples may appear once, more than once, or not. On average, for sufficiently large s, 63% of the original examples will appear in the bootstrap sample. The learner L is then applied to this training set. This procedure is repeated m times, and the resulting m models are aggregated by uniform voting (i.e., when a test example is presented, the class that is predicted by the greatest number of models is predicted; if a tie occurs, the lowest-ordered class is chosen, as in [3]).[4]

Examples for meta-learning are generated using the probability distribution implicit in the rule sets produced by C4.5RULES. Effectively, the number of training examples classified by each rule, as a fraction of the training set size, is an estimate of the probability of finding an example in the region covered by that rule, and not covered by any preceding rules. Given that the m component models thus represent m approximations to the "true" example distribution, all with equal weight, and that n new examples are required, CMM generates n/m of these examples from each component rule set. For each individual rule, if it classified r of the s examples in the bootstrap sample it was induced from, $(r/s) \cdot (n/m)$ examples covered by it will be generated. For each example, this is done by ensuring that it satisfies the rule's preconditions, and beyond that by setting the values of its attributes according to a uniform distribution (thus leading to a piecewise uniform approximation to the example distribution). Before being accepted, an example is matched with the previous rules in the rule set, and if it is covered by any it is regenerated, to ensure that the current rule does not become under-represented, and the previous rule that matched it over-represented. This is done up to a maximum number of times, 100 by default.[5] Missing values for each attribute are generated in similar proportions to those found in the original training set.

[4]Note that, although learning m bootstrap models is m times more computationally expensive than learning a single model, this does not imply that bagging is infeasible for large databases, since it can be trivially parallelized with almost 100% efficiency by learning each of the m models on a different processor.

[5]The exact value of this parameter is unimportant. It suffices that it make the number of examples incorrectly generated from a rule smaller than the quantization error implicit in the computation of $(r/s) \cdot (n/m)$.

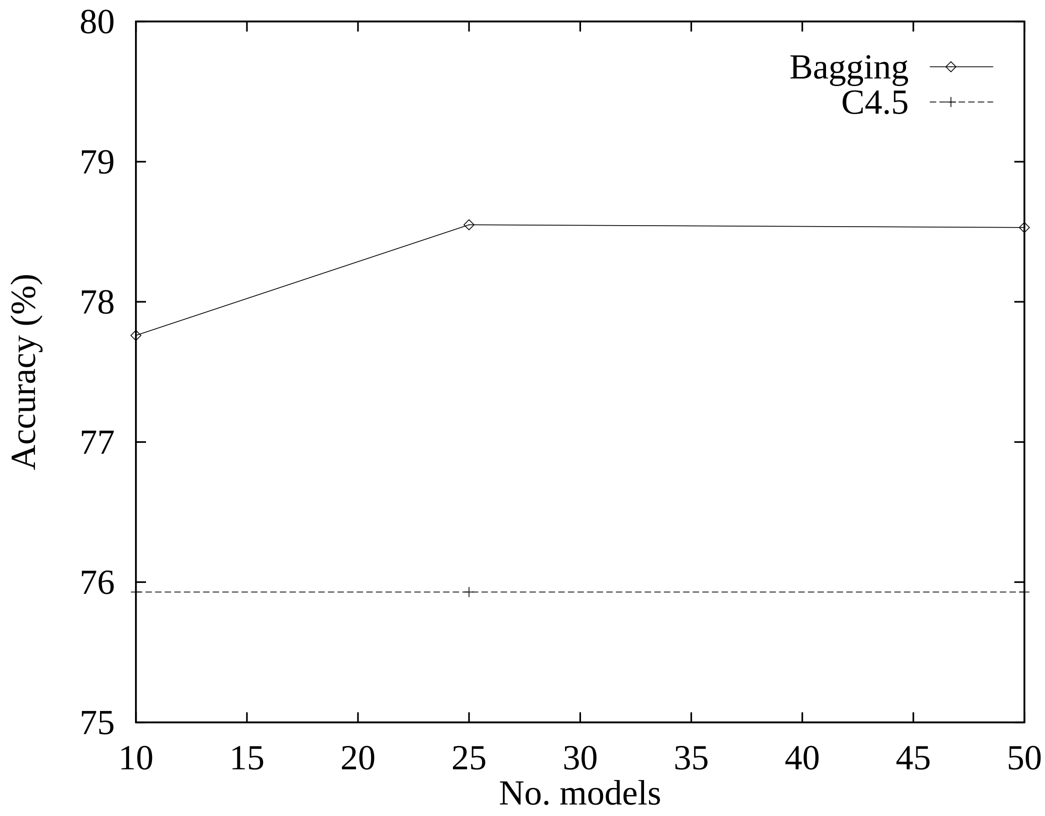

Figure 1: Average accuracy of bagging as a function of the number of component models.

3 Empirical Evaluation

The question of whether the approach just outlined will indeed, in any given domain, retain a significant fraction of bagging's accuracy and stability gains over a single run of C4.5RULES, without producing an overly complex model, is one to be answered empirically. To this end, experiments were carried out using a varied and representative sample of 26 datasets from the Irvine repository [24].

A value for m (the number of component models) was determined by comparing the accuracies obtained with $m = 10$, 25 and 50 on a subset of the datasets. The results are shown in Figure 1, as averages over all datasets. $m = 50$ was found to produce no significant gains over $m = 25$, but $m = 10$ led to poor results. This suggests some atypicality in Breiman's [3] results on the (artificial) waveform domain, where he studied the effect of m and found most of the gain in accuracy to be obtained with 10 replicates, and also suggests that Quinlan [29] might have obtained even better results for bagging if more replicates had been used. $m = 25$ was used throughout the studies reported below.

The number n of examples generated randomly for meta-learning was set to 1000. This value reflects the knowledge that C4.5RULES tends to produce rule sets whose size grows approximately linearly with training set size [13, 25], and thus that using a very large n is likely to lead to unnecessarily complex models. Within this constraint, n was chosen to be larger than any of the dataset sizes present. Given that the randomly generated examples are added to the original ones, this implies that the training set size for meta-learning will always be at least twice the size of the original training set.

An experimental methodology similar to that of [3] was followed, with 20 runs instead of 100, due to the large number of datasets used. In each run, 90% of the examples in the dataset were randomly chosen for training, and the remainder were used for testing. Table 2 shows the resulting average accuracies and standard deviations. The results for a single model are those obtained by running C4.5RULES on the entire unchanged training set. C4.5/C4.5RULES was applied with its default settings throughout.

Bagging improves accuracy relative to a single run of C4.5RULES in 22 out of the 26 datasets, by 3.5 percentage points on average (2.6% in all 26 datasets). These 22 datasets, where bagging improves on C4.5RULES, are those where it would make sense to apply CMM. CMM improves on C4.5RULES in all but four of them. Its average accuracy is 2.1% higher than C4.5RULES's, i.e., CMM retains on average 60% of bagging's accuracy gains, and similarly in all 26 datasets. CMM is more accurate than C4.5RULES with a confidence of 99.7% according to a sign test, and 99.9% according to a Wilcoxon test (96% and 99.2%, respectively, on all 26 datasets). In six of the 22 datasets, CMM is more accurate than bagging.

Output comprehensibility is more difficult to measure than accuracy, since it is ultimately subjective. However, an oft-used operational measure of it is output size, counting one unit for each antecedent and each consequent of each rule (including the default rule, with 0 antecedents and 1 consequent). While this measure is necessarily imperfect, its meaningful use here is facilitated by the fact that two of the outputs being compared (single model and CMM) are interpreted in exactly the same way, being produced by the same learner.

The significance of changes in output size is different from that of changes in accuracy, where in general each percentage point of improvement counts. In this respect, output size behaves similarly to running time, another complexity measure: its exact value is arguably of little significance, as long as it stays within given acceptable bounds. Thus a greater increase in complexity is more acceptable when starting from a lower basis, since the resulting final complexity will be correspondingly lower.

The output sizes obtained for C4.5RULES, bagging and CMM are shown in Table 3, along with the ratio between CMM's size and C4.5RULES's. The datasets are listed in increasing order of size (i.e., increasing number of examples). Bagging's output size, computed as the sum of the component model sizes, obviously underestimates the difficulty in comprehending the bagged ensemble, and is given here only for indicative purposes. CMM's complexity is typically a small multiple of C4.5RULES's (2–6), with the smaller multiples occurring for the larger datasets. This is as desired, and leads to CMM's complexity staying below 250 for almost all domains. Thus, by this measure, CMM can be considered to produce comprehensible output in almost all cases, assuming C4.5RULES does. As a further comparison, another widely-used rule learner, CN2 [8], was run on these datasets. While being on average less accurate than C4.5RULES and CMM, CN2 has an output complexity that is often greater than CMM's. This is further evidence that CMM is broadly competitive with single-model rule learners in its ability to produce comprehensible results.

The output sizes shown for CMM in these datasets may be unnecessarily large, due to the tendency of C4.5RULES's output size to grow approximately in step with input size, and to the fact that 1000 artificial examples are being generated for all domains, leading to a ratio between the meta-learning and original training set sizes that is quite large for the smaller datasets (those with tens of examples). It may thus be possible to substantially optimize CMM's complexity without seriously affecting accuracy by choosing

Table 2: Empirical results: average accuracies and their standard deviations.

Dataset	**CMM**	**Bagging**	**Single**
Lenses	75.0±6.8	75.0±6.8	62.5±7.1
Lung cancer	40.0±7.5	36.7±7.2	31.7±7.0
Soybean (small)	97.0±1.6	97.0±1.6	98.0±1.4
Labor	85.8±2.5	90.0±1.9	81.7±2.7
Post-operative	68.9±3.0	66.1±3.3	71.7±3.1
LED	61.5±3.2	61.5±3.2	60.0±2.5
Zoology	94.0±1.8	93.5±1.8	92.0±1.7
Promoters	87.7±1.7	88.6±1.5	87.3±2.5
Echocardiogram	67.7±3.2	71.9±3.5	65.8±2.5
Lymphography	76.7±2.3	80.0±2.6	75.3±2.8
Iris	94.3±1.1	94.3±1.1	94.7±1.1
Hepatitis	76.2±2.5	79.4±2.7	77.5±2.6
Wine	94.2±1.0	95.8±1.0	94.7±1.6
Audiology	77.8±1.5	78.8±1.5	75.2±1.3
Sonar	71.7±2.2	77.4±1.6	76.9±1.6
Glass	73.1±1.5	77.4±1.5	71.7±1.7
Breast cancer	70.2±1.8	69.7±1.9	68.4±1.9
Horse colic	86.3±1.6	86.0±1.4	83.8±1.4
Heart disease	81.8±1.5	80.2±1.7	78.0±1.7
Solar flare	69.2±1.7	70.0±1.6	71.1±1.7
Primary tumor	43.4±2.1	46.9±2.3	42.4±2.4
Liver disease	66.0±1.8	69.6±1.9	63.7±1.4
Voting records	95.5±0.7	96.6±0.6	96.4±0.7
Credit	87.5±1.1	88.3±0.8	86.2±1.1
Pima diabetes	75.5±0.9	76.4±0.9	73.6±0.8
Annealing	96.2±0.5	95.2±0.6	93.9±0.6
Average	77.4	78.6	75.9

Table 3: Empirical results: Output sizes and stabilities. "C/S" is the ratio between CMM's output size and the single model's.

Dataset	**Output size**				**Stability**		
	CMM	Bagging	Single	C/S	CMM	Bagging	Single
Lenses	17.5	255.2	9.7	1.8	94.1	94.1	90.2
Lung cancer	198.8	354.6	14.8	13.5	55.0	58.5	52.9
Soybean (small)	160.1	268.5	10.2	15.7	83.9	85.0	78.2
Labor	39.7	271.0	9.9	4.0	89.9	93.3	75.1
Post-operative	82.4	688.8	9.1	9.0	85.7	90.7	83.3
LED	202.1	1211.6	47.1	4.3	73.2	73.4	63.5
Zoology	140.8	623.7	29.4	4.8	92.1	92.5	93.4
Promoters	143.0	538.3	21.8	6.6	84.6	87.8	83.7
Echocardiogram	85.1	729.8	13.1	6.5	71.7	79.2	64.4
Lymphography	213.2	877.5	34.2	6.2	51.8	65.2	62.0
Iris	17.6	303.5	11.0	1.6	96.0	96.9	99.2
Hepatitis	158.2	630.9	29.0	5.4	71.8	78.5	71.6
Wine	69.4	399.1	15.0	4.6	71.1	79.4	74.6
Audiology	440.3	1973.3	74.2	5.9	45.3	53.8	52.4
Sonar	90.6	816.6	35.2	2.6	62.2	83.4	75.4
Glass	214.6	1740.5	61.8	3.5	67.1	70.5	69.3
Breast cancer	174.1	1933.1	16.8	10.4	75.4	76.0	68.1
Horse colic	52.8	972.5	22.6	2.3	88.7	89.6	82.7
Heart disease	177.7	1396.9	48.5	3.7	80.2	88.8	75.1
Solar flare	250.1	2070.6	52.8	4.7	83.8	87.7	84.7
Primary tumor	579.0	5029.8	90.3	6.4	52.9	55.9	50.9
Liver disease	116.1	2329.9	53.0	2.2	82.1	85.0	70.6
Voting records	74.2	597.1	21.9	3.4	90.4	93.2	94.2
Credit	104.9	2181.2	49.4	2.1	89.7	90.6	81.4
Pima diabetes	82.4	3476.6	38.9	2.1	83.1	85.4	76.3
Annealing	90.2	1680.7	64.0	1.4	88.4	90.8	87.3
Average	156.6	1364.2	35.8	4.9	77.6	82.2	75.7

n as a function of dataset size. This is a matter for future research.

Stability was measured following the ideas contained in [32], and taking advantage of the models produced in the train-test runs carried out. The stability of a system is defined as the (estimated) probability that models generated by the system from different training sets will agree on an arbitrary example. For each domain, $ne = 1000$ unclassified examples were generated using a uniform distribution in the instance space. For each system, stability was then computed as:

$$Stab = 100\% \times \frac{1}{ne} \sum_{e=1}^{ne} \left[\frac{2}{nr\,(nr-1)} \sum_{i=1}^{nr} \sum_{j=1}^{i-1} agree_{eij} \right]$$

where nr is the number of train-test runs carried out (and therefore the number of models produced by the system), and $agree_{eij}$ is 1 if models M_i and M_j predicted the same class for example e, and 0 otherwise. The use of artificial examples obviates the need to set apart examples from the original dataset for stability testing, further dwindling the number available for training, and/or leading to unreliable estimates of stability. The use of a uniform distribution reflects a deeper notion of stability than that implicit in using some estimate of the dataset's distribution [32]. Note that, because the different models are not generated from independent training sets, the empirical measure above will tend to overestimate stability.

The results are shown in Table 3. On average, bagging is 6% more stable than a single run of C4.5RULES. CMM is 2% more stable than C4.5RULES, and thus retains a third of bagging's stability gains. CMM improves stability in all but five of the 20 datasets where bagging is more accurate and stable than C4.5RULES. On all 26 datasets, CMM is more stable than C4.5RULES with a confidence of 91% according to a sign test, and 94% according to a Wilcoxon test. These results are remarkable, in view of the fact that CMM, like C4.5RULES, produces a single rule set. Taken together with the accuracy and complexity results, they show that CMM can indeed retain a large part of bagging's accuracy and stability gains while still outputting comprehensible rule sets, and thus brings us closer to the goal of producing models that qualify as knowledge.

The running time of the bagging procedure is proportional to the number of models being generated, while the time needed to produce the combined model is a function of the total number of examples. With C4.5RULES and the numbers of models and examples used in our experiments, the running time of the meta-learning procedure (not including calls to the learning procedure L) was always lower than that of bagging by more than an order of magnitude, and therefore not a significant factor.

A summary lesion study was conducted by (1) excluding the original examples from the meta-learning training set,[6] and (2) replacing the probability estimation procedure described in the previous section with a uniform distribution of the examples in instance space. The results are shown in Table 4. As expected, both changes had a significant negative impact on accuracy, confirming the utility of the procedures used. CMM was more accurate than the "all new examples" version in 18 datasets and less in 4, by 2.6% on average, with a confidence greater than 99% according to a Wilcoxon test. CMM was more accurate than the "uniform probability" version in 15 datasets and less in 5, by 1.1% on average, with a confidence greater than 99% according to a Wilcoxon test.

[6]More precisely, replacing the original examples with an equal number of artificial ones, in order to maintain a constant training set size.

Estimating example probabilities from the frequencies in the leaves of the underlying (pruned) decision trees also decreased accuracy, lending support to the notion that a good match between the learner's bias and that of the probability estimation procedure is important for good results. Disabling the generation of missing values had a large negative impact in the annealing dataset, where very large numbers of missing values are present, and a less discernible one in the datasets where fewer such values occur.

C4.5RULES's pruning parameters during the meta-learning phase can be used to trade off the accuracy and complexity of CMM's output. For example, using a confidence level of 10% instead of the default, or applying the optional Fisher's exact test with a 10% confidence level, both resulted in output size reductions in the 10–30% range for almost all domains, while decreasing average accuracy by only fractions of a percentage point.[7] Another parameter that can potentially be used to trade off accuracy and complexity is n, the number of synthetic examples generated. To investigate this, the experiments were repeated with $n = 500$ and $n = 2500$. The results are shown in Figures 2 to 4, as averages across all datasets. As expected, complexity increases steadily with training set size, and average accuracy improves significantly from $n = 500$ to $n = 1000$, but not from $n = 1000$ to $n = 2500$.[8] Since, in general, varying n is only useful up to the point where accuracy asymptotes, the best results may be obtained by empirically estimating this point for each dataset.[9] More conveniently, CMM may be used with a base learner whose output size is insensitive to training set size, once the accuracy asymptote is reached (or at least less sensitive than C4.5RULES). Experiments with such systems (e.g.,[13, 20]) are an area for future research.

4 Related Work

The CMM algorithm bears interesting relationships to many pieces of previous research in inductive learning. Apart from its effect on accuracy, pruning of decision trees and rule sets can be viewed as an attempt to extract a simpler, more comprehensible model from an overly complex one. Work by Catlett [5] (Chapter 5), Quinlan [28] (Chapter 5), Evans and Fisher [16], and Fayyad, Djorgovski and Weir [17] has this flavor. Quinlan [27] briefly describes merging all branches from multiple decision trees into a single rule set and extracting the "best" rules, with promising results. Buntine [4] considers the extraction of a single "good" tree from an option tree (a compact representation of multiple trees) to be an important problem for future research. Kong and Dietterich [22] make a similar statement for error-correcting output coding and other multiple-model schemes. Shannon and Banks [30] have recently proposed a method for combining multiple decision trees into one, based on measuring distances between them and finding the "median" tree. Their approach is considerably more complex than CMM, and has not yet been implemented, or shown to improve on the accuracy and stability of single trees. The high instability of tree learners may make it very difficult for this method to produce good results. Utans

[7]In order to ensure a fair comparison with C4.5RULES, these changes were also applied when learning the single model; while they also produced some reductions in output size at little cost in accuracy, these reductions were generally smaller than those obtained with CMM.

[8]Interestingly, the stability of CMM continues to improve even after its accuracy asymptotes.

[9]However, stability will often continue to improve after accuracy asymptotes, and this may make it worthwhile to use larger training sets. For example, in the experiments described, with $n = 2500$ CMM's stability gain averaged over all datasets was nearly half of bagging's, compared to a third for $n = 1000$.

Table 4: Lesion study results: average accuracies and their standard deviations. "All new" shows the result of ignoring the original examples. "Uniform" shows the result of using a uniform example distribution.

Dataset	**CMM**	**All new**	**Uniform**
Lenses	75.0	75.0±5.9	75.0±5.9
Lung cancer	40.0	33.3±6.4	35.0±5.8
Soybean (small)	97.0	97.0±1.4	97.0±1.4
Labor	85.8	79.2±2.8	85.8±2.4
Post-operative	68.9	71.1±2.8	69.4±3.0
LED	61.5	61.5±2.8	61.0±2.7
Zoology	94.0	92.0±1.6	94.0±1.6
Promoters	87.7	84.5±1.6	83.6±1.4
Echocardiogram	67.7	67.3±3.0	66.5±2.2
Lymphography	76.7	75.0±2.4	77.7±2.5
Iris	94.3	96.3±0.9	94.3±1.0
Hepatitis	76.2	75.6±2.1	76.2±2.6
Wine	94.2	94.2±0.9	94.7±1.2
Audiology	77.8	67.3±2.1	75.8±1.3
Sonar	71.7	65.7±1.8	67.1±1.1
Glass	73.1	61.2±2.5	70.2±1.7
Breast cancer	70.2	66.4±2.1	70.5±1.3
Horse colic	86.3	85.8±1.2	84.7±1.2
Heart disease	81.8	79.0±1.7	80.0±1.6
Solar flare	69.2	69.1±1.4	69.4±1.4
Primary tumor	43.4	40.1±2.0	39.3±2.2
Liver disease	66.0	67.3±1.9	64.9±1.3
Voting records	95.5	95.6±0.7	95.1±0.7
Credit	87.5	86.2±0.8	86.7±0.8
Pima diabetes	75.5	75.2±0.7	74.5±0.9
Annealing	96.2	85.6±0.9	96.1±0.5
Average	77.4	74.9	76.3

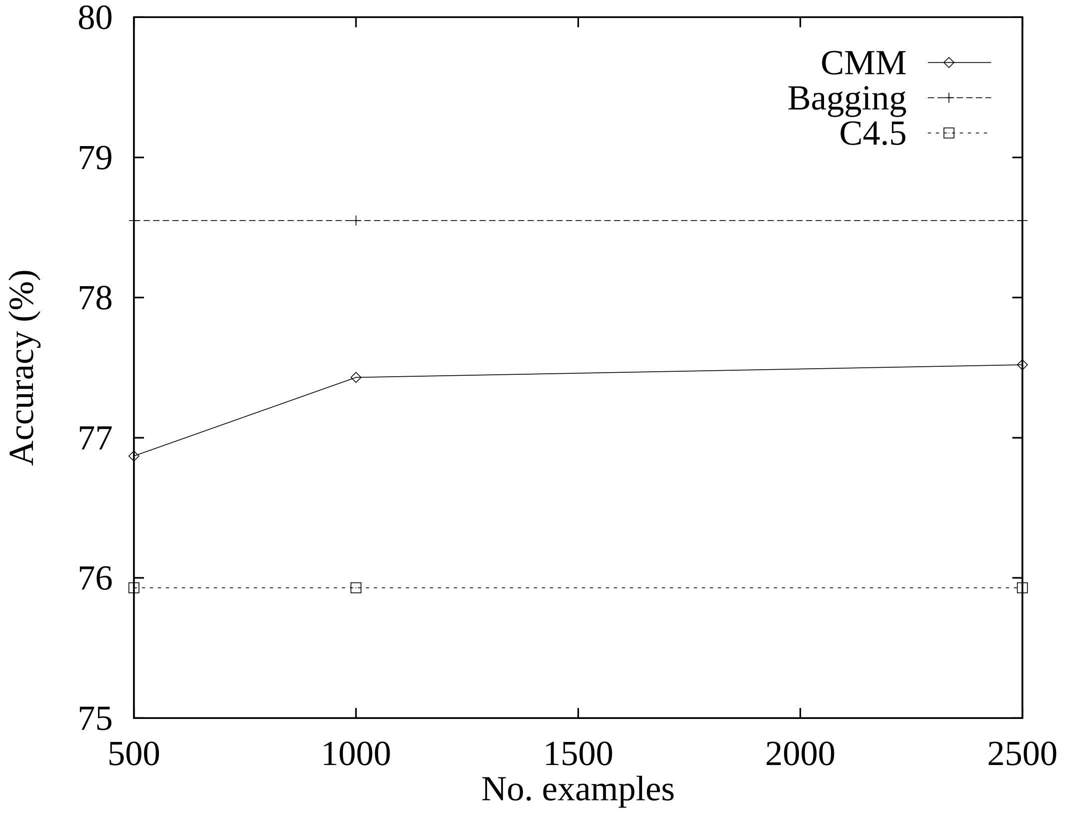

Figure 2: Average accuracy as a function of the number of artificial examples.

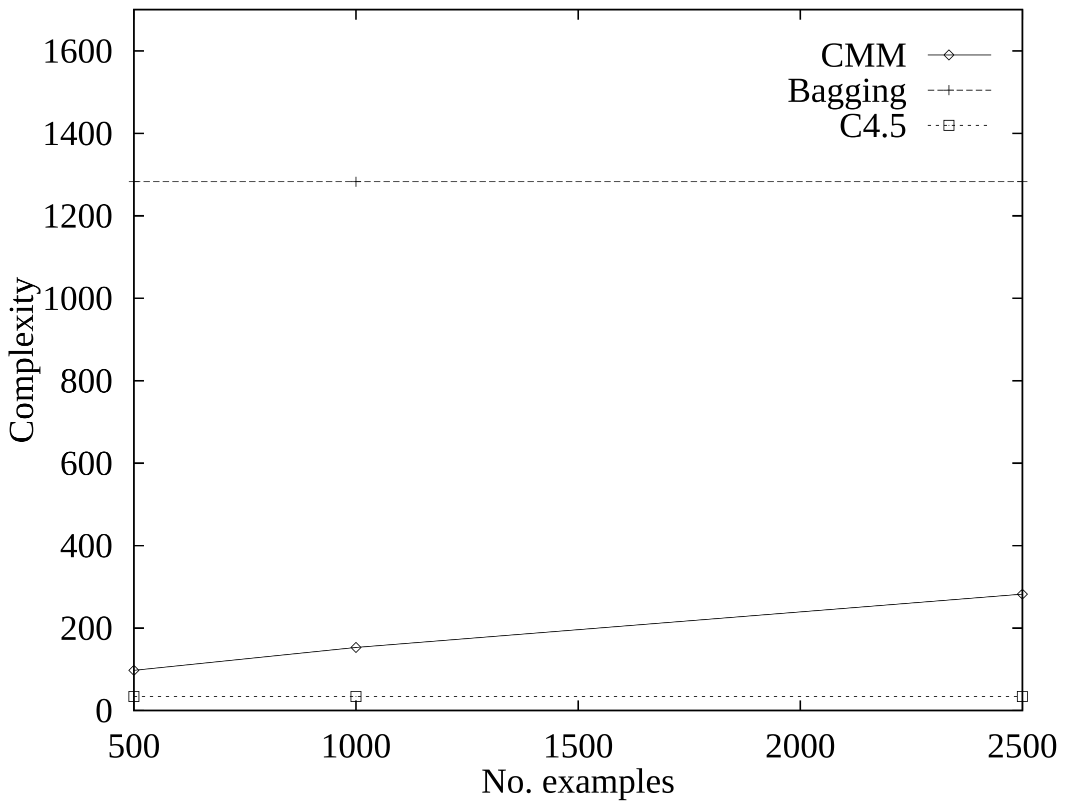

Figure 3: Average complexity as a function of the number of artificial examples.

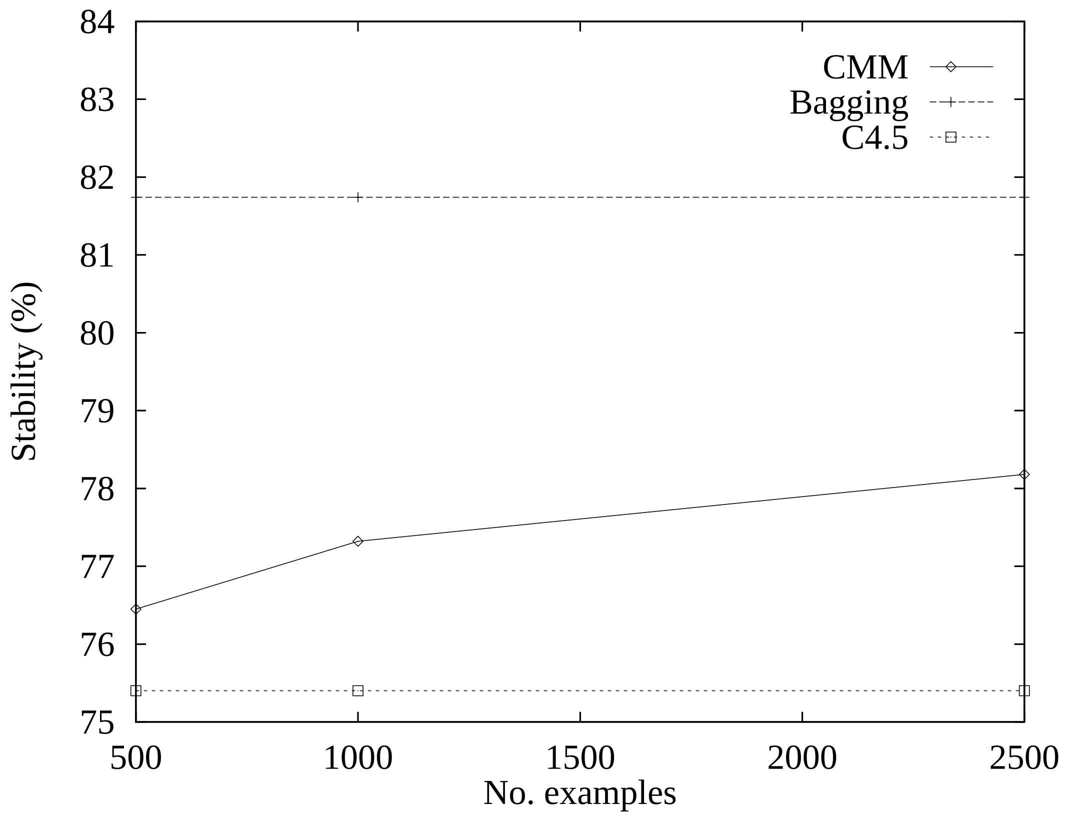

Figure 4: Average stability as a function of the number of artificial examples.

[33] combined multiple neural networks into one by averaging parameters, with the goal of achieving performance-time computational savings; the output still suffers from the opaqueness of a neural network. Datta and Kibler [11] have sought to develop single-model classifiers that are more stable than standard machine learning algorithms, but have so far not attempted to evaluate the comprehensibility of the prototype-based representation they use.

CMM is an example of a method for extracting comprehensible output from a learned model (in this case, a bagged ensemble of models). Substantial research has been carried out for the case where the model is a neural network (e.g., [31, 1]). Algorithms based on queries to an oracle are also relevant to this problem, and have been the object of much study in the theoretical community (e.g., [2]). Although oracle-based algorithms are generally of limited usefulness when learning directly from real data, they can be applied more easily in the meta-learning phase, using the previously-learned model (or model ensemble) as the oracle (e.g., [10]). More generally, many forms of active learning [9], where the learner has some degree of control over the information it obtains from the environment, are potentially applicable to this problem.

5 Future Work

Several directions for future research are readily apparent, apart from those already mentioned in previous sections. One is to use different algorithms as the base learner and the meta-learner. To be justified, this should result in better performance than either algorithm used alone. An intriguing possibility is that, when models in language $\mathcal{L}$ are combined, the result may be best expressed in a different language $\mathcal{L}_M$ (presumably with $\mathcal{L} \subset \mathcal{L}_M$). If general conditions under which this is and is not the case can be derived, they may then be applied to the choice of meta-learner given a base learner.

Ideally, the meta-learning phase would result in models that are as stable and accurate as the bagged ensemble, while being of similar complexity to the base models. Replacing the random sampling of instance space currently used with query-based/active learning techniques may bring this limit closer, while keeping the computational complexity of the meta-learning phase within acceptable bounds. An alternative (and perhaps less promising) approach is to explicitly combine the component rule sets, for example by applying induction to the problem of determining what surface features they have in common.

Another direction for future research is investigating the conditions under which CMM can be expected to work (i.e., improve accuracy and stability relative to the base learner, while losing little or no comprehensibility). Intuitively, if the problem is "easy" (i.e., the base learner can find the "best" model or a close one in one pass), there should be little gain in using multiple models and CMM. On the other hand, if the base learner's representation is inappropriate to the domain, it may be impossible to improve accuracy without corresponding increases in complexity (e.g., if the representation used is one of rules with tests on single attribute values, and the true frontier is not axis-parallel). Thus, the greatest gains from using CMM may be obtained in domains where the base learner's representation is adequate, but the learner has difficulty finding the "right" model, for example because of its greedy search procedure or limited choice of rule construction operators. Careful empirical study should be able to shed some light on this matter.

A limitation of the work described here is that it uses a somewhat naive notion of comprehensibility, equating it with simplicity. In reality, many other factors are involved, and will vary from one user group to another (e.g., [26]). Although arriving at a better definition of comprehensibility is a difficult task due to the subjective component involved, much progress should result from seeking a deeper understanding of what makes a model comprehensible, and using the results to guide algorithms like CMM.

A limitation of the implementation used in this article is that it is only applicable to domains where there is no relationship between succeeding examples. In datasets that represent time-varying processes, the random subsampling carried out by bootstrapping would destroy important information. The study of implementations that avoid this problem, for example by randomizing the learning algorithm instead of the training set, is another area for future research.

The meta-learning method described here can be used to (attempt to) render comprehensible the output of any learner, by applying rule induction to the models it produces. For example, it could be used to extract rules from a neural network. It would be interesting to see how it (or variations of it) fare at this task.

6 Conclusion

In the quest to automate knowledge discovery, current learners still have a considerable way to go, if learned models are viewed as constituting knowledge only if they are simultaneously accurate, stable and comprehensible. This article described a method for taking an unstable learner of comprehensible representations and obtaining one that is more stable and accurate, while still producing comprehensible output. The proposed method, called CMM, is based on learning several models, combining them, and reapplying the learner to make explicit the mapping thus produced. Experimental tests have shown CMM to be a promising approach.

Acknowledgements

This article describes research conducted at the University of California, Irvine, and partly supported by a NATO scholarship. The author is grateful to all those who provided the datasets used in the experiments. Please see the documentation in the UCI Repository for detailed information.

References

[1] R. Andrews and J. Diederich, editors. *Proceedings of the NIPS-96 Workshop on Rule Extraction from Trained Artificial Neural Networks.* NIPS Foundation, Snowmass, CO, 1996.

[2] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

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

[4] W. L. Buntine. *A Theory of Learning Classification Rules.* PhD thesis, School of Computing Science, University of Technology, Sydney, Australia, 1990.

[5] J. Catlett. *Megainduction: Machine Learning on Very Large Databases.* PhD thesis, Basser Department of Computer Science, University of Sydney, Sydney, Australia, 1991.

[6] P. Chan, S. Stolfo, and D. Wolpert, editors. *Proceedings of the AAAI-96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms.* AAAI Press, Portland, OR, 1996.

[7] P. K. Chan and S. J. Stolfo. A comparative evaluation of voting and meta-learning on partitioned data. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 90–98, Tahoe City, CA, 1995. Morgan Kaufmann.

[8] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–283, 1989.

[9] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15:201–221, 1994.

[10] M. W. Craven. *Extracting Comprehensible Models from Trained Neural Networks.* PhD thesis, Department of Computer Sciences, University of Wisconsin – Madison, Madison, WI, 1996.

[11] P. Datta and D. Kibler. Learning prototypical concept descriptions. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 158–166, Tahoe City, CA, 1995. Morgan Kaufmann.

[12] T. G. Dietterich. Editorial. *Machine Learning*, 24:91–93, 1996.

[13] P. Domingos. Linear-time rule induction. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 96–101, Portland, OR, 1996. AAAI Press.

[14] H. Drucker, C. Cortes, L. D. Jackel, Y. LeCun, and V. Vapnik. Boosting and other machine learning algorithms. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 53–61, New Brunswick, NJ, 1994. Morgan Kaufmann.

[15] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, NY, 1973.

[16] B. Evans and D. Fisher. Process delay analysis using decision tree induction. *IEEE Expert*, 9(1):60–66, 1994.

[17] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1–34. AAAI Press, Menlo Park, CA, 1996.

[18] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, Bari, Italy, 1996. Morgan Kaufmann.

[19] J. H. Friedman. On bias, variance, 0/1 - loss, and the curse-of-dimensionality. Technical report, Department of Statistics and Stanford Linear Accelerator Center, Stanford University, Stanford, CA, 1996. ftp://playfair.stanford.edu/pub/friedman/kdd.ps.Z.

[20] D. Jensen. Adjusting for multiple testing in decision tree pruning. In *Preliminary Papers of the Sixth International Workshop on Artificial Intelligence and Statistics*, pages 295–302, Fort Lauderdale, FL, 1997. Society for Artificial Intelligence and Statistics.

[21] R. Kohavi and D. H. Wolpert. Bias plus variance decomposition for zero-one loss functions. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 275–283, Bari, Italy, 1996. Morgan Kaufmann.

[22] E. B. Kong and T. G. Dietterich. Error-correcting output coding corrects bias and variance. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 313–321, Tahoe City, CA, 1995. Morgan Kaufmann.

[23] D. Madigan, A. E. Raftery, C. T. Volinsky, and J. A. Hoeting. Bayesian model averaging. In *Proceedings of the AAAI-96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms*, pages 77–83, Portland, OR, 1996. AAAI Press.

[24] C. J. Merz, P. M. Murphy, and D. W. Aha. UCI repository of machine learning databases. Machine-readable data repository, Department of Information and Computer Science, University of California at Irvine, Irvine, CA, 1997.

[25] T. Oates and D. Jensen. The effects of training set size on decision tree complexity. In *Preliminary Papers of the Sixth International Workshop on Artificial Intelligence and Statistics*, pages 379–390, Fort Lauderdale, FL, 1997. Society for Artificial Intelligence and Statistics.

[26] M. Pazzani, S. Mani, and W. R. Shankle. Comprehensible knowledge discovery in databases. In *Proceedings of the Ninenteenth Annual Conference of the Cognitive Science Society*, Stanford, CA, 1997. Erlbaum.

[27] J. R. Quinlan. Generating production rules from decision trees. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 304–307, Milan, Italy, 1987. Morgan Kaufmann.

[28] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[29] J. R. Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730, Portland, OR, 1996. AAAI Press.

[30] W. D. Shannon and D. Banks. A distance metric for classification trees. In *Preliminary Papers of the Sixth International Workshop on Artificial Intelligence and Statistics*, pages 457–464, Fort Lauderdale, FL, 1997. Society for Artificial Intelligence and Statistics.

[31] G. G. Towell and J. W. Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine Learning*, 13:71–101, 1993.

[32] P. Turney. Bias and the quantification of stability. *Machine Learning*, 20:23–33, 1995.

[33] J. Utans. Weight averaging for neural networks and local resampling schemes. In *Proceedings of the AAAI-96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms*, pages 133–138, Portland, OR, 1996. AAAI Press.

[34] D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.