Arbitrating Among Competing Classifiers Using Learned Referees

Julio Ortega
julio@us.ibm.com, Phone: 1-972-406-5946
IBM Global Business Intelligence Solutions, 1503 LBJ Freeway
Dallas, TX 75234, USA

Moshe Koppel
koppel@cs.biu.ac.il, Phone: 972-3-531-8407
Dept. of Mathematics and Computer Science, Bar-Ilan University
Ramat Gan 52900, Israel

Shlomo Argamon (corresponding author)
argamon@mail.jct.ac.il, Phone: 972-2-675-1294
Dept. of Computer Science, Jerusalem College of Technology, Machon Lev
21 Havaad Haleumi St.
Jerusalem 91160,Israel

Abstract

The situation in which the results of several different classifiers and learning algorithms are obtainable for a single classification problem is common. In this paper, we propose a method that takes a collection of existing classifiers and learning algorithms, together with a set of available data, and creates a combined classifier that takes advantage of all of these sources of knowledge. The basic idea is that each classifier has a particular subdomain for which it is most reliable. Therefore, we induce a *referee* for each classifier, which describes its area of expertise. Given such a description, we *arbitrate* between the component classifiers by using the most reliable classifier for the examples in each subdomain. In experiments in several domains, we found such arbitration to be significantly more effective than various voting techniques which do not seek out subdomains of expertise. Our results further suggest that the more fine-grained the analysis of the areas of expertise of the competing classifiers, the more effectively they can be combined. In particular, we find that classification accuracy increases greatly when using intermediate subconcepts from the classifiers themselves as features for the induction of referees.

Keywords: Integrating multiple models, theory revision, bias selection, decision trees, stacked generalization, constructive induction

1. Introduction

In this paper, three currently active research programs in machine learning (integrating multiple classifiers [12] [5, 6] [49] [16] [50],theory revision [20] [36] [40] [2] [44] [51] [35] [28] [13] [3] and bias selection [33] [32] [23] [7] [45]) are viewed from a single perspective. The goal of integrating multiple classifiers is to improve the performance and scalability of learning algorithms by generating multiple classifiers, running them on distributed systems, and combining their results. Theory revision systems make use of two sources of knowledge: an existing imperfect model of a domain and a set of available data. Through a combination of analytical and empirical learning, theory revision systems create a new model of improved performance, as measured by expected accuracy on unseen data. Bias selection systems, on the other hand, make use of available data for a domain and several empirical learning algorithms. The various implicit biases of these algorithms may result in different behavior (and accuracies) over a domain or portion thereof. The objective of bias selection systems is either to select a single algorithm or to combine the results of several learning algorithms in order to maximize the predictive accuracy of empirical learning on unseen instances of the domain. Both theory revision and bias selection can address the problem of combining several classifiers with the help of a set of data. Some of the classifiers may be pre-existing classifiers of the domain (as the imperfect classifier of theory revision systems) while others may be constructed by different empirical learning algorithms (as in bias selection systems). We propose a method that takes such a collection of existing classifiers and learning algorithms, together with a set of available data, and creates a combined classifier that takes advantage of all of these sources of knowledge.

The situation in which the results of many classifiers and learning algorithms are obtainable for a single problem is common. This tends to occur for complex problems of high importance. As an example, many competing classifiers have been developed for predicting the glucose levels of diabetic patients. In fact, special journal issues [10] and symposiums [8] are dedicated to presenting a variety of glucose-level prediction classifiers. Also, several learning algorithms have been suggested for the same problem [26] [1]. Similarly, a variety of mathematical models as well as learning algorithms exist for predicting returns of financial transactions. Prediction of stock option pricing using the Black-Scholes mathematical model and neural networks are compared and discussed, for example, by Malliaris and Salchenberger [30]. Although our techniques can be extended to numerical prediction problems such as those in the diabetes and financial domains, this paper addresses only classification problems.

1.1 Our approach

The approach of this paper, reported independently by Ortega [38] and Koppel and Engelson [27], is to build a *referee* predictor for each of the component classifiers. As we discuss later in this section, each referee predictor will tell us whether its corresponding component classifier can be trusted for particular unseen instances. We assume that each classifier has a particular subdomain for which it is most reliable, and so we learn a description of this area of expertise. Given such a description, we *arbitrate* between the component classifiers

1. A special set of diabetes data provided for this event was also donated to the UCI Machine Learning Repository [37]

by using the most reliable expert for diagnosis of problems in each subdomain. That is, the final classification is that returned by the component classifier whose correctness can be trusted the most, according to a confidence level provided by the referees.

The intuition behind our approach is that every imperfect classifier (whether hand-crafted or learned by induction over a set of data) is only able to make correct predictions over a subset of the entire domain space. Consider, for example, a classifier for diagnosing chronic ulcer disease in which the intermediate concept "intestinal cramps" is too generally defined, but the theory is otherwise correct. Then, patients whose symptoms do not satisfy "intestinal cramps" *as defined in the classifier* will indeed be correctly diagnosed regarding chronic ulcer disease. Of the patients that do satisfy "intestinal cramps" some will be correctly classified while others will not and we could say that the classifier is not reliable for this group of patients. This illustration suggests that we might use a set of training examples to determine the areas of a particular classifier's 'expertise', represented in terms of the classifier's own ontology.

In general, for classifiers constructed by induction over a set of data, the portion of the domain space in which they can correctly predict depends on the specific set of data available for training (i.e., the more representative it is of the whole domain space, the better the learning algorithm will perform) and also on the implicit biases of the learning algorithm used for induction. Given the same data, several learning algorithms will give different classifiers only due to their different biases. For hand-crafted classifiers, correctness and coverage also depend, implicitly, on the data originally available to a researcher for developing the classifier and the researcher's own biases.

1.2 Other approaches

Our method can be viewed as an application of Wolpert's stacked generalization [52] to classification problems where some of the generalizers are not necessarily created through empirical learning but are instead manually constructed. From a very general perspective, stacked generalization can be equated to meta-learning, as it refers to the induction of classifiers (i.e., *generalizers*) over inputs that are, in turn, the predictions of other classifiers induced over the original input space. The arbitrating mechanism in our approach can be viewed as a generalizer. Its inputs, referees and component classifiers, consist of the predictions of classifiers induced from the original input space using either automated induction techniques, for the referees and the learning component classifiers, or ad-hoc induction by a human expert, for the hand-crafted component classifiers. The idea of using auxiliary learners for combining multiple classifiers was also exploited in the parallel meta-learning methods of Chan and Stolfo [12]. Our method differs from the above in that it is *modular*, i.e. a referee is learned for each individual component model allowing new component models to be added without relearning a combining classifier. By contrast, the approach of Chan and Stolfo learns a combination classifier based on the inputs and outputs of all the component models, and thus adding a new component model necessitates a new meta-learning episode.

The advantage of combining multiple learned classifiers to obtain more accurate classification was also demonstrated by Ali and Pazzani [1]. Separating the training data into subsets where classifiers either succeed or fail to make correct predictions was used

in Schapire's Boosting algorithm [46]. Experimental evidence by Drucker et al. [17] indicated that boosting was superior to a single learner and to a voting committee of learners. Our method can also be viewed as multiple model probabilistic evidence combination, an issue that has been explored both in the decision tree literature (Buntime [9], Kwok and Carter [29]) and in the neural networks literature (i.e., the "Mixture of Experts" approach of Jordan et al. [25, 24]). Our approach utilizes a winner-take-all method of evidence combination: each referee assigns a "confidence", i.e., a probability of being correct, to its corresponding classifier. The classifier that is assigned the highest confidence is trusted to make the final prediction for each specific example. The experiments described below also examine another evidence combination method (simple voting) with less remarkable results.

Our arbitration approach is also closely related to Merz's SCANN [33]. Roughly, SCANN uses a nearest-neighbor method to determine which component classifier to use to classify each test example. The representation used by SCANN is produced by singular-value decomposition of the results of applying the component classifiers to the training data. SCANN can be viewed as a nearest-neighbor equivalent to the symbolic learning approach given in this paper, and so although it does not construct explicit representations of referees, SCANN's final results may be similar to those given by our arbitration method.

We will compare our arbitration method with two simpler methods, SAM (Select All Majority) and CVM (Cross-Validation Majority), discussed by Merz [32]. In the SAM approach, the prediction of each component classifier is an equally weighted vote for that particular prediction. The prediction with most votes is selected. Ties are broken arbitrarily. CVM is a variation of an algorithm suggested by Schaffer [45]. In CVM, the cross-validation accuracy for each classifier is estimated with the training data, and the classifier of highest accuracy is selected for use with all of the test data. In the case of ties (i.e., the same best accuracy is obtained by several classifiers), the majority class predicted by the set of classifiers of best accuracy is returned. Hence, if all classifiers have the same estimated accuracy, then CVM's behavior is the same as SAM's. Although Schaffer uses his algorithm to select among learning algorithms, there is no difficulty in extending his approach to non-learning classifiers: their accuracy is simply evaluated on the training data and multiple cross-validation runs are unnecessary. (A similar approach, Bagging [5, 6], generates component classifiers from data much like CVM, except that it uses bootstrap samples [18] instead of cross-validation runs, and combines their result by majority voting as in SAM.)

We will show that arbitration often produces better results than these simpler methods, because it is better able to identify and exploit the strengths of each component classifier.

2. Referees

In this section, we will consider how to use a set of classified examples in order to find the strengths and weaknesses of a given classifier. The idea is to determine for what subsets of examples the classifier's result can be taken to be reliable, and to what extent. This procedure constitutes the first stage in the arbitration mechanism for multiple classifiers, to be described in the next section.

2.1 Referee induction

To be precise, an *example* E is an assignment of values to a set of *primitive attributes* $\{a_i\}$. The task is to assign each example to one of a set of classes $\{c_i\}$. A *classifier* Γ is a function from examples to classes. Given a training set $\mathcal{E}$ of examples together with their correct classifications, we want to determine Γ's probable accuracy for different subsets of possible new examples.

The basic idea is to use a decision tree induction algorithm (such as C4.5 [43]) to induce a decision tree for distinguishing between cases where the classifier is correct and those where it is incorrect. Such a decision tree divides the example space up into a number of disjoint subsets, each corresponding to a leaf of the tree. For examples in each subset we can estimate the accuracy of Γ based on the fraction of examples in that subset that Γ classified correctly. In order that these estimates be useful, the decision tree must be pruned sufficiently to ensure that the subsets at the leaves are large enough to give us good estimates of the classifier's reliability. For example, given a large set of training examples, a leaf for which 80 out of 100 training examples are correct for the classifier gives a better estimate of the classifier's reliability than one for which 8 out of 10 are correct. Reliability estimates should therefore take into account the sample size over which they are computed.

A further point is the choice of features for induction. It is not always the case that those regions of the example space where Γ is reliable can be simply described in terms of primitive attributes; more complicated derived attributes may need to be used to attain efficient and effective induction. We discuss this issue in more detail below, here we just assume that some set of features is provided for inducing referees.

The algorithm for inducing a referee is therefore as follows:

LearnReferee(Γ,$\mathcal{E}$):

1. Partition $\mathcal{E}$ into $\mathcal{E}_C$ and $\mathcal{E}_I$, the subsets of $\mathcal{E}$ correctly and incorrectly classified by Γ, respectively;

2. Select a set of features as a basis for concept induction including computed features (these should always include the primitive attributes defining the examples and the example's class in Γ);

3. Build a pruned decision tree T (we use C4.5 for this purpose) using the features selected in the previous step;

4. For each leaf L in T:

(a) Compute the *reliability* [2] of L based on C and I, the numbers of examples in $\mathcal{E}_C$ and in $\mathcal{E}_I$, respectively, classified to L; we use the formula[3]

$$\frac{\max(C, I)}{C + I + \frac{1}{2}}$$

(b) If $C > I$ then set L's *correctness* is 'correct', otherwise to 'incorrect'.

5. Return T.

In other words, we use induction to distinguish between those examples the classifier classifies correctly and those it classifies incorrectly, also estimating the decisiveness of such determinations.

2.2 Choosing features for referee induction

As mentioned above, the nature of the referees that are learned depends on which features are used for induction. In particular, it may not be the case that those regions of the example space where Γ is reliable can be simply described in terms of primitive attributes. Therefore, it may be preferable to use intermediate subconcepts in the classifier Γ itself as features for induction. In a logical theory, for example, these subconcepts are the various propositions and clauses in the theory, in a decision tree these subconcepts are the conditions at the various decision nodes in the tree, and in a neural net these might be the activation values of internal nodes. These features are computed for each training example and added to its description when performing the induction.

As we will see, much of the power of our method comes from the use of internal propositions of a flawed theory as features for induction. The use of such internal propositions as features for use in induction is not new, going back to the work on constructive induction of Drastal and Raatz [15], and more recently to the work of Ortega and Fisher [39] and Donoho and Rendell [14]. Our approach differs, however, in that rather than learn a decision tree directly for the target concept, we learn a decision tree which distinguishes when the given theory is correct and when it is incorrect. Due to the differences in the target concept, the trees learned for the correct/incorrect concept may differ greatly from those learned directly for the target concept. In particular, when flaws are focused in one subconcept of the theory, the tree constructed for the correct/incorrect concept may be more natural, in that it will simply rule out classifying on the basis of that subconcept.

The fact that we learn decision trees for the 'theory correct/incorrect' concept also sheds some light on the utility of using internal nodes. As we will see below, internal propositions are particularly useful in diagnosing examples for which the theory is unreliable. This is because it is precisely truth or falsehood of internal nodes in the *flawed* theory which will predict well whether particular flaws affect the classification of the given example, and so

2. Note that in this paper, we use the term *reliable* or *reliability* to refer to the estimates provided by referee predictors as to how often a component classifier will make correct predictions. This usage of the term may not correspond with its meaning in the statistics or COLT literature.
3. The addition of $\frac{1}{2}$ to the denominator is a quick-and-dirty application of the m-estimate method [11] of smoothing to compensate for small sample sizes, with $m = \frac{1}{2}$ and prior probability 0. Other approaches are possible.

will give a better estimate of the theory's reliability. Hence we expect the internal features of a theory, in combination with the root of that theory, to be better features for induction of referees than either the primitive attributes or other kinds of synthetic features.

2.3 Referees for classifiers learned from data

The **LearnReferee** algorithm can build referees for classifiers learned from data with machine learning algorithms as well as for pre-existing classifiers. All that is required is that the examples in the training data be divided into two subsets: a subset where the classifier is expected to make *correct* classifications and a subset where the classifier is expected to make *incorrect* classifications.

Using a learning algorithm as a component classifier is particularly useful if all pre-existing classifiers are unreliable for some examples. In such a case, we can train a new classifier on these examples, incorporating a learning classifier into our ensemble. Arbitration will then automatically classify according to the learned classifier whenever no pre-existing classifier is considered sufficiently reliable. This should result in overall accuracies higher than those of any of the component classifiers alone.

2.4 Example: Chess endgame

To illustrate the operation of **LearnReferee**, consider the flawed chess *endgame* theory (from the UCI Machine Learning Repository [37]) depicted in Table 1. It should be noted that examples are of the form $(a = 1, b = 4, \ldots)$, and that propositions such as `a=b` are defined explicitly in the actual theory, (although the definitions are not shown). The errors introduced here are that a new clause is added for `king-attack-king`, that the antecedent `adj-bf` was moved from `king-attack-king` to the first `illegal` clause, and one of the clauses for `rook-attack-king` is deleted.

Our goal is to find a referee which characterizes the set of examples for which the flawed theory classifies incorrectly, that is, those examples whose classifications are affected by the flawed literals and clauses. This goal can only be realized, however, when the representation of the data is adequate for learning with a given learning algorithm, i.e., when it is possible to learn referees that correctly predict whether each of the component classifiers should be trusted with probability better than random. As noted above (Section 2.2), internal features of the theory are likely to be useful for this purpose. (In any event, for the *endgame* domain, effective learning is not even possible with the original representation of the data, where the primitives are the numerical positions of the pieces on the board [41].) Therefore, we include internal features for learning, so that we can learn a useful referee.

To evaluate **LearnReferee**, we used C4.5 with pruning, using internal features of the theory for performing the induction. We used a set of 1000 examples, 315 of which were positive and 685 of which were negative, where 749 were correctly classified by the theory. We ran **LearnReferee** on all 1000 examples in the training set; Table 1 shows the referee thus constructed.

The critical point is that when the flawed clauses and literals in the theory are not used in classifying an example, its classification by the theory will be reliable, and when they are used, the example's classification will be unreliable. To properly appreciate this point, let us consider in detail two of the leaves of the tree.

Table 1: A flawed version of the *endgame* theory, shown in propositional form. For simplicity we do not show the computation of low-level nodes such as `between-cae`. Added antecedents and added clauses are shown in boldface, while deleted components (not in the flawed theory) are shown in italics.

illegal	:-	same-loc-ab-cd, **adj-bf**
illegal	:-	same-loc-ab-ef
illegal	:-	same-loc-cd-ef
illegal	:-	king-attack-king
illegal	:-	rook-attack-king
king-attack-king	:-	adj-ae, adj-bf
king-attack-king	:-	adj-ae, b=f
king-attack-king	:-	a=e, *adj-bf*
king-attack-king	**:-**	**knight-move-ab-ef**
rook-attack-king	*:-*	*c=e, king-not-b-file*
rook-attack-king	:-	d=f, king-not-b-rank
king-not-b-rank	:-	¬b=d
king-not-b-rank	:-	b=d, ¬between-cae

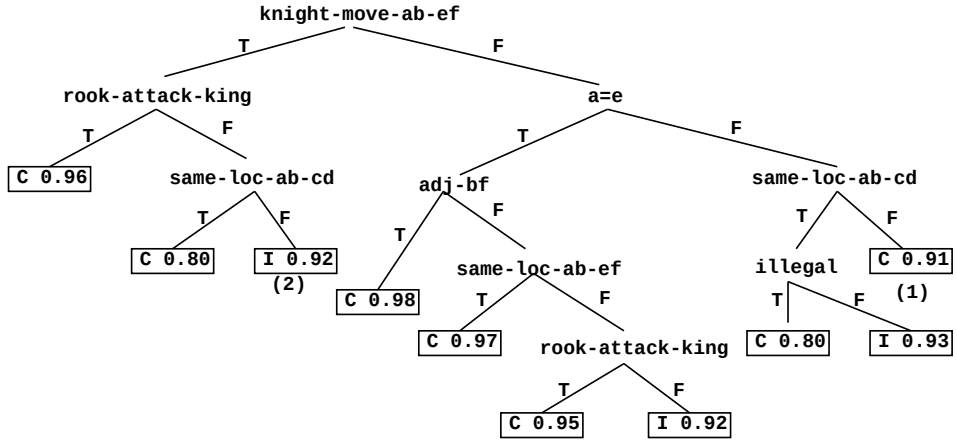
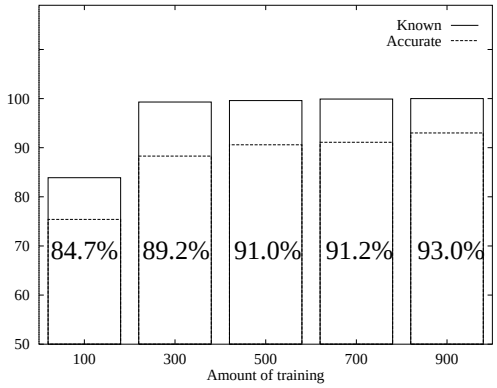
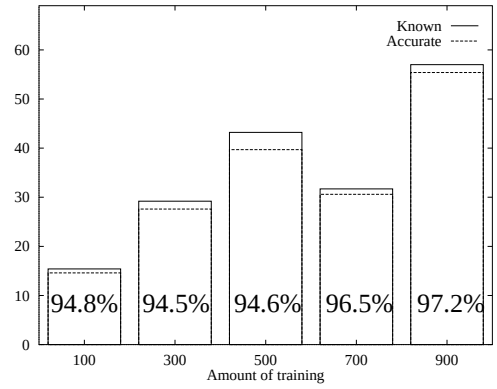


Figure 1: Referee for the flawed theory shown in Table 1, learned from 1000 training examples. Leaves are labeled with correctness (`C` = correct, `I` = incorrect) and reliability.



(a)



(b)

Figure 2: Using the referee for selective classification with the *endgame* theory for reliability thresholds of (a) 0.8 and (b) 0.9. Plotted are average fraction of known examples (those assigned a class) and classification accuracy for known examples, over ten disjoint 100-example test sets.

Consider first the leaf marked (1), where `knightmove-ab-ef` is false, `a=e` is false, and `same-loc-ab-cd` is false, to which 766 of the examples are classified. For examples classified to this leaf, three of the theory's flaws have no effect: the clauses affected by the change of `adj-bf` are false regardless of the value of `adj-bf`, and when `knight-move-ab-ef` is false. The theory thus classifies examples at this leaf correctly with high reliability. This shows how using a classifier's internal features in the referee affords easy characterization of the examples for which the classifier is reliable.

Consider now the leaf marked (2), where `knightmove-ab-ef` is true, but `rook-attack-king` and `same-loc-ab-cd` are false. For examples classified to this leaf, the theory is considered *incorrect* with high reliability. This demonstrates how inductive methods can find useful information about a theory; we can discover both when the theory is reliably correct, and when it is reliably incorrect. In the latter case, for theories such as this one where the root is two-valued, the right thing to do is to classify opposite to the theory [19] (perhaps requiring higher reliability for determining incorrectness than for determining correctness).

Figure 2 shows the results of using the theory to classify only those examples whose reliability score is above a certain threshold, marking all others as "unknown" (assigning the class opposite to that given by the theory when it is reliably incorrect for an example). We used 10-fold cross-validation, withholding varying amounts of nested training data in order to create a learning curve.

As can be seen, the number of examples assigned a class by this procedure tends to increase with the amount of training, as the leaves become more decisive. At the same time, classification accuracy increases, and becomes considerably higher than classification by the theory alone (74.9%). This is due to reversing reliably incorrect classifications by the theory, as well as rejecting unreliably classified examples.

3. Arbitration

Now consider the case where we are given several possibly flawed classifiers for a particular domain. Rather than try to combine the results of the different classifiers, we seek to choose the *right* classifier to use to classify each new input example. By using each classifier just on that subdomain for which it is most reliable, our overall results should be considerably better than those of the best individual classifier. If different classifiers are experts in different subdomains, then the more experts we have, the greater the area that is covered reliably [1].

Given a set of classifiers $\Gamma_1,\ldots,\Gamma_n$ for a binary concept and a set of classified examples $\mathcal{E}$, we use the following arbitration algorithm to classify new input examples by using the most reliable expert for each one.

Arbitrate($\Gamma_1,\ldots,\Gamma_n,\mathcal{E},E$):

1. Construct a referee for each input classifier Γ_i, T_i = **LearnReferee**($\Gamma_i,\mathcal{E}$);
2. For the new input example E:
   (a) Evaluate the correctness c_i and reliability r_i of each classifier Γ_i for E, using T_i;
   (b) Let Γ_{i_*} be the classifier with highest reliability r_{i_*};

(c) If c_{i_*} ='correct', assign E the class $\Gamma_{i_*}(E)$;

(d) Else if c_{i_*} ='incorrect', assign E the class $\neg\Gamma_{i_*}(E)$;

Put simply, the algorithm classifies each new example E according to the classifier that has proven most reliable in classifying examples in the training set similar to E. Of course, this classification may be the opposite of that given by one of the given classifiers, if that classifier is incorrect with higher reliability than that of any other classifier being correct (or incorrect). If the concept being learned is not binary, we extend the algorithm to simply use the most reliably correct (or least reliably incorrect) classifier on each example.

4. Results

Experiments were conducted using both pre-existing classifiers and several variations of the C4.5 learning algorithm as component classifiers. Besides the *endgame* theory described previously, we also used Ourston's [40] version of Michalski and Chilausky's soybean expert theory [34]. In addition, we constructed a set of artificial classifiers of different quality based on the audiology dataset from the UCI (University of California at Irvine) Machine Learning repository [37], named *audio*, *audio1*, *audio2*, *audio3*, and *audio4*. These classifiers were created following Mooney's approach [36] for generating classifiers of varying degrees of imperfection. A perfect classifier, named *audio*, was constructed by running C4.5 with all pruning disabled on the complete data set of 226 audiology examples. The result is a classifier that contains 86 rules with an average of 7.79 conditions per rule. The *audio1*, *audio2*, *audio3*, and *audio4* classifiers were each created by randomly adding and then randomly deleting a percentage (respectively 5, 10, 25, 50) of all conditions from the rules of the perfect classifier, resulting in some overly specific rules and some overly general rules. The respective accuracies of these distorted theories are 91%, 65%, 47%, 22%. As learning classifiers we use the standard C4.5 learning algorithm in default mode and C4.5 with all pruning disabled.

For the experiments of sections 4.1 and 4.2, 30 trials were conducted using approximately 2/3 of the available data for training and the remaining 1/3 for testing. The figures and tables in these two sections show the average accuracy, computed on the test data, over the 30 trials. In sections 4.3 and 4.4, accuracy is computed by 10-fold cross-validation, or as specifically indicated in each one of the figures.

4.1 Arbitration versus voting and cross-validation methods

Figure 3 illustrates the different behavior of SAM, CVM, and **Arbitrate** using the *audio2*, *audio3*, and *audio4* as components. Since the component classifiers in this experiment are simply given, SAM's accuracy is independent of the number of examples available for training. SAM's accuracy (53%) is higher than that of the two worst component classifiers, *audio3* (47%) and *audio4* (22%), but lower than that of the best component classifier, *audio2* (65%). SAM's accuracy appears to be an average of the accuracy of the non-learning component classifiers. With only a few component classifiers of very different quality, as is the case in this experiment, SAM's majority voting strategy results in values lower than those of the best component classifier, although some papers [1, 22, 31] have

reported experiments in which SAM achieves better accuracy than even the best component classifier.

CVM, on the other hand, always chooses the best single classifier, as determined by the accuracy obtained by each component classifier on the training data. With non-learning classifiers, the optimal choice for CVM is the best available component classifier, *audio2* in this experiment. As a result, the accuracy of the best component classifier is an upper-bound on the accuracy that CVM can obtain. In the experiment corresponding to Figure 3, the upper-bound on CVM's accuracy is 65%, the accuracy of the *audio2* classifier. However, this optimal choice is made consistently by CVM only when the training set is large enough to permit reliable estimates of the accuracy of the component classifiers. In our experiment, this occurs with training sets containing 55 or more examples. With training sets containing fewer examples, CVM's accuracy is lower than that of the best classifier.

In contrast to CVM and SAM, the accuracy of **Arbitrate** in this experiment is not bounded by that of the best component classifier. As shown in Figure 3 **Arbitrate**'s accuracy is 71%, higher than the 65% accuracy of *audio2*, the best of the component classifiers.

Figure 4 shows the average accuracies obtained with SAM, CVM, and **Arbitrate** when their component classifiers include three non-learning classifiers (*audio2*, *audio3*, and *audio4*) and two learning classifiers (*C4.5 with default pruning* and *C4.5 with no pruning*). As before, CVM's accuracy tracks closely that of the best component classifier for each given training set size, and SAM's accuracy appears to be an average of the accuracy of the component classifiers. For training sets larger than 10 examples, **Arbitrate** obtains better accuracies than SAM, CVM, or any of the component classifiers. With 150 examples, **Arbitrate** obtains mean accuracies of 84%, larger than the accuracy of both the best non-learning classifier, 65% of *audio2*, and the best learning classifier, 77% of *C4.5 with no pruning*. As before, the accuracy results are an average over 30 trials. In each trial, 1/3 of the data (76 examples) is reserved for testing and is not presented to the learning algorithms. Progressively larger samples of the remaining 2/3 of the data (150 examples) are used to obtain the different points of the learning curves shown in the experiments.

The behavior of SAM, CVM, and **Arbitrate** using the soybean expert theory and C4.5, illustrated by the learning curves of Figure 5, is similar to that observed with the *audio2* artificial audiology classifier described in the previous section. SAM's accuracy is always between that of the hand-crafted classifier and that of C4.5. CVM's accuracy is never too different from that of the best accuracy of either the hand-crafted classifier or C4.5. With training sets of 40 and 100 examples, **Arbitrate**'s accuracy is higher than that of the best of both the hand-crafted classifier and C4.5. However, with 150 examples, **Arbitrate**'s accuracy is the same as that of CVM and C4.5 alone. Apparently, in this domain 150 examples are sufficient to overwhelm the effect of the prior knowledge given in the form of the expert classifier. More generally, Ting et al [48] have observed that the benefits of model combination for a specific problem diminish as the steepness of the learning curve decreases, something that typically occurs with larger number of training examples.

4.2 Arbitration versus theory and induction

Knowledge about the quality of the hand-crafted classifier can help determine the most appropriate approach for combining such a classifier with induction from data. Table 2

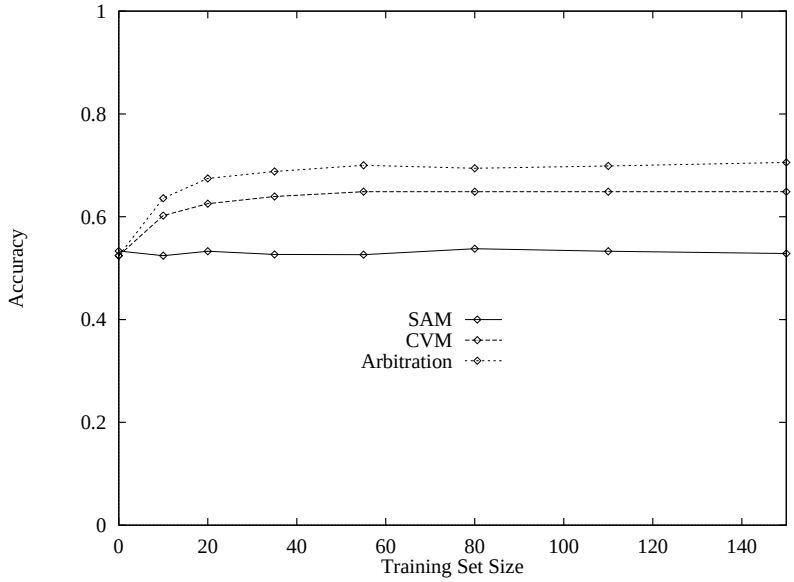


Figure 3: Average accuracy of SAM, CVM, and Arbitrate using *audio2*, *audio3*, and *audio4* as component classifiers.

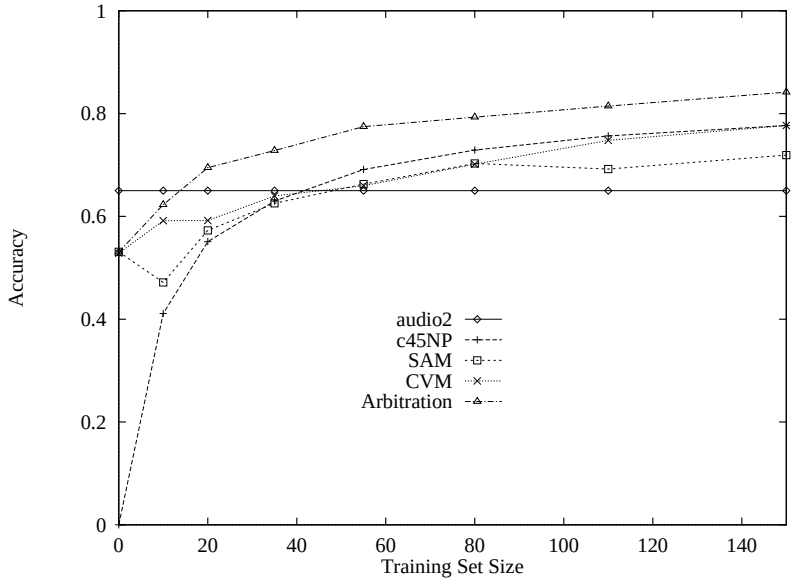


Figure 4: Average accuracy of SAM, CVM, and Arbitrate using *audio2*, *audio3*, *audio4*, *C4.5*, and *C4.5 with no pruning* as component classifiers.

shows the difference between the average accuracy obtained by CVM, using an artificial audiology classifier and C4.5 with the audiology data set, and the best of the average accuracies obtained by either the classifier alone or C4.5 alone. These results are based on training sets containing 150 examples. Table 3 does the same for our **Arbitrate** method. From these tables one can conclude that if no knowledge about the quality of the hand-crafted classifier exists, **Arbitrate** may be a better choice, since it obtains positive accuracy improvements for more than half of the combination of classifiers and training set sizes we examine. In contrast, CVM never produces any accuracy increase over that of the best component classifier. With training sets of less than 150 examples the same conclusions become even more evident.

Even though **Arbitrate**'s potential for improving accuracy is better than CVM's, so are the risks. With a good quality classifier such as *audio1*, the potential loss in accuracy with respect to the best component classifier is higher in **Arbitrate** (5%) than in CVM (2%).

Tables for SAM are not included because its accuracy for this data set always stays below that of CVM and Arbitration. SAM's accuracy appears to be an average of the accuracies of the component classifiers. As with SAM, CVM's accuracy is always a value between the accuracy of the component classifiers, but is usually very close to that obtained by the best of the component classifiers.

In contrast to CVM, **Arbitrate** can obtain accuracies that are higher than those of its component classifiers. The increase in accuracy can be attributed to the use of referees to indicate which classifier can be trusted the most in specific instances.

4.3 Arbitration with internal features

As noted earlier, the benefits of **Arbitrate** can only be realized when the representation of the data is adequate for learning with a given learning algorithm, i.e., when it is possible to learn referees that correctly predict whether each of the component classifiers should be trusted with probability better than random. In the above experiments with the audiology and soybeans domains the concept representation appeared adequate. However, as noted, learning is not possible with the original representation of the data in the *endgame* domain, so we must include internal features.

We now consider the use of arbitration with internal features, using three flawed versions of the *endgame* theory, each with six random flaws (each the insertion or deletion of an antecedent or a clause). We used a set of 1000 preclassified examples in 10-fold cross-validation experiments.

We compare accuracy against a voting approach (SAM) as well as C4.5 learning with different feature sets. 'Primitive' features are the lowest level logical propositions of the *endgame* theory (shown in Table 1), for example `adj-ae` (which is actually defined arithmetically as numerical features `a` and `e` being adjacent). Using these primitive features gives C4.5 a chance to learn a good classifier. We also use a feature set including all primitive features as well as the classes given by each of the three component classifiers. This strategy, which is equivalent to Chan and Stolfo's *class-attribute-combiner* [12], amounts to treating each component classifier as a black box, whose internal structure cannot be accessed. Finally, we examine C4.5 learning using all the internal features of all the com-

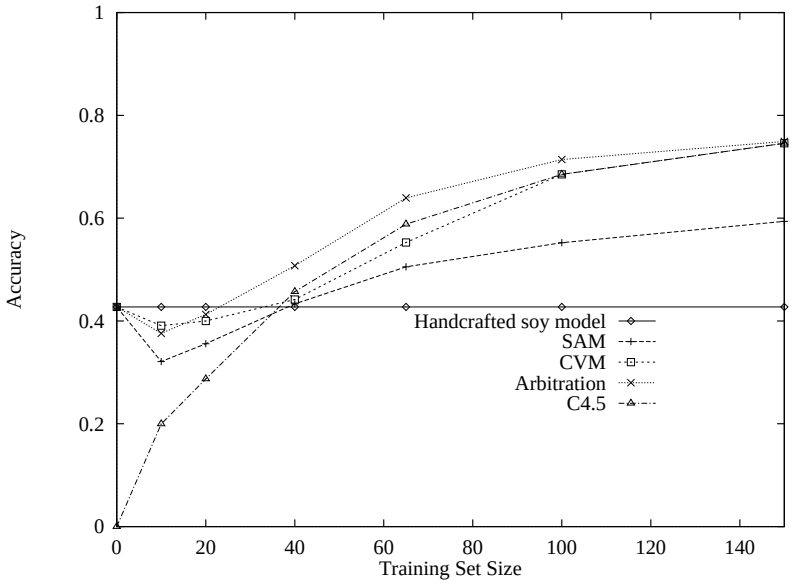


Figure 5: Average accuracy of SAM, CVM, and Arbitrate on the soybean domain.

Table 2: Difference between the accuracy obtained by CVM and the best of the accuracies obtained by either the classifier alone or C4.5 alone.

Training Set Sizes	Accuracy Improvement with Classifier ... audio	audio1	audio2	audio3	audio4
0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	–.01	–.01	–.06
20	0.0	–.02	–.03	–.02	–.03
35	0.0	0.0	–.01	–.01	–.01
55	0.0	0.0	–.02	–.01	0.0
80	0.0	0.0	–.04	–.01	0.0
110	0.0	0.0	–.02	0.0	0.0
150	0.0	0.0	0.0	0.0	0.0

ponent classifiers. Arbitration can be thought of as a restricted version of this approach in which learning is performed separately for the internal features of each component classifier. This restriction has the primary advantage of *modularity*, as depicted in Figure 6. By learning a referee for each component classifier separately, component classifiers can be added and removed from the system without requiring learning to start from scratch. Second, C4.5 with all internal features is more likely to suffer from the problem of *overfitting*, due to the abundance of features. In the *endgame* domain, we find arbitration to perform as well as C4.5 with all internals, despite the restriction to learning only from each classifier individually. This brand of modularity, in turn, is conducive to efficient implementations in distributed computers.

The accuracies of the three theories over all 1000 examples were all near 70%, as shown in Figure 7. Using C4.5 to learn the target concept directly on primitive attributes gives similarly low accuracy. By including the classifications given by all three theories as a feature for building C4.5's decision tree, however, accuracy improves noticeably. The best results shown are obtained by using C4.5 with all internal features, with an accuracy of 93.2% and by the **Arbitrate** algorithm, with an accuracy of 93.4% (both using 10-fold cross-validation). This shows how by properly combining the information in the theories with the information in the training set, high classification accuracy can be obtained.

It is still possible, however, that the gain in accuracy actually arises from the building of the referees for each theory, and that arbitration itself contributes little. To test the hypothesis that selecting the correct theory is important, we generated the learning curves shown in Figure 8. Five learning curves are compared: C4.5 using primitives and theories' classifications (as above), for each of the three theories, classifying according to the theory's referee (either using or reversing the theory's classification), and arbitrating between the three theories. As is clear from the figure, arbitration performs substantially better than each of the three individual theories, even with their associated decision trees. This reflects the fact that **Arbitrate** uses different theories for different types of examples.

4.4 Increasing the number of component classifiers

We now give results for a more extensive experiment testing the hypothesis that arbitration is capable of exploiting the reliable portion of each component theory. To do this we consider arbitration among different numbers of component theories. If the different theories are being properly exploited, we expect classification error to decrease geometrically as we increase the number of component theories, as noted by Ali and Pazzani [1].

For each experimental trial we generated five flawed theories, each by inserting twelve random errors into the base theory. We randomly ordered the theories and recorded the average accuracy produced by running **Arbitrate** on the first theory, the first two theories, and so forth, doing 10-fold cross-validation on 1000 examples. We ran 8 such trials and averaged the results for each number of theories used in arbitration.

The results of this experiment are summarized in Figure 9. As is apparent, the error rate decreases greatly with the number of theories used. In fact, the error appears to decrease

Table 3: Difference between the accuracy obtained by **Arbitrate** and the best of the accuracies obtained by either the classifier alone or induction alone.

Training Set Sizes	Accuracy Improvement with Classifier ... audio	audio1	audio2	audio3	audio4
0	0.0	0.0	0.0	0.0	0.0
10	0.0	–.05	–.02	+.01	–.02
20	0.0	–.05	+.01	+.08	+.05
35	0.0	–.05	+.06	+.06	+.04
55	0.0	–.03	+.09	+.05	+.03
80	0.0	–.01	+.08	+.04	+.03
110	0.0	+.01	+.08	+.04	+.07
150	0.0	+.02	+.07	+.04	+.03

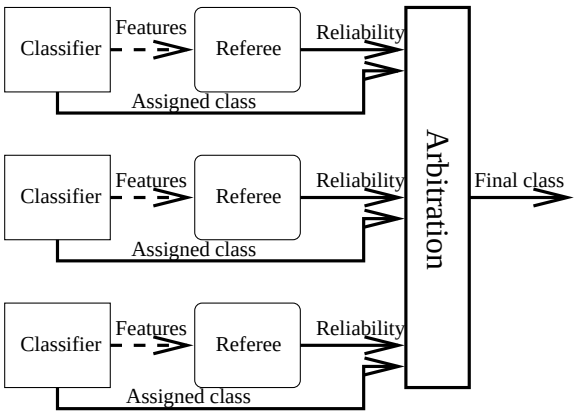


Figure 6: The modular architecture of the arbitration scheme. Each referee depends only on a single component classifier.

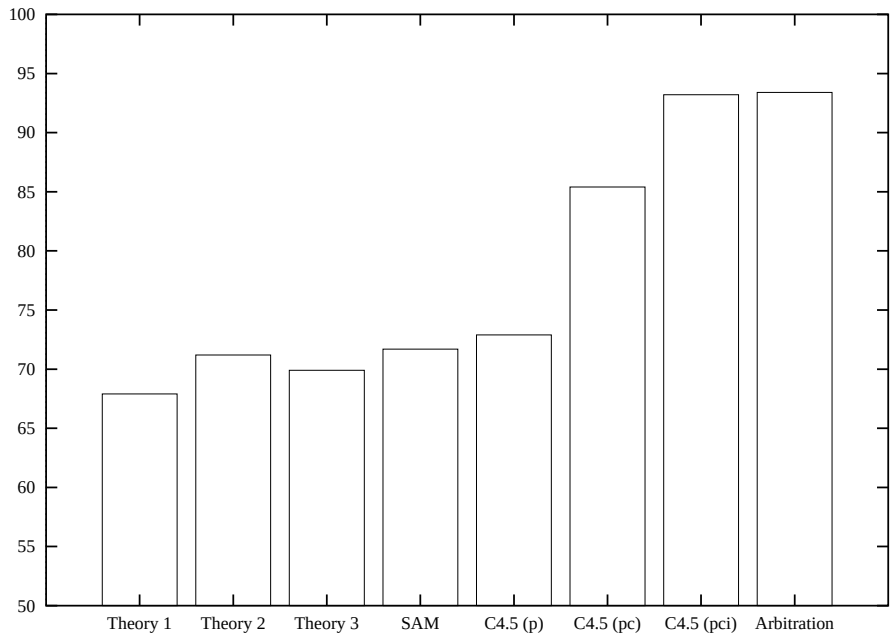


Figure 7: The accuracy of the three flawed *endgame* theories on 1000 training examples; the accuracy of C4.5 with primitive attributes (C4.5 (p)), C4.5 using primitive features and theory classifications (C4.5 (pc)), C4.5 using all internal features (C4.5 (pci)), and arbitration between the theories, all learning results using 10-fold cross-validation.

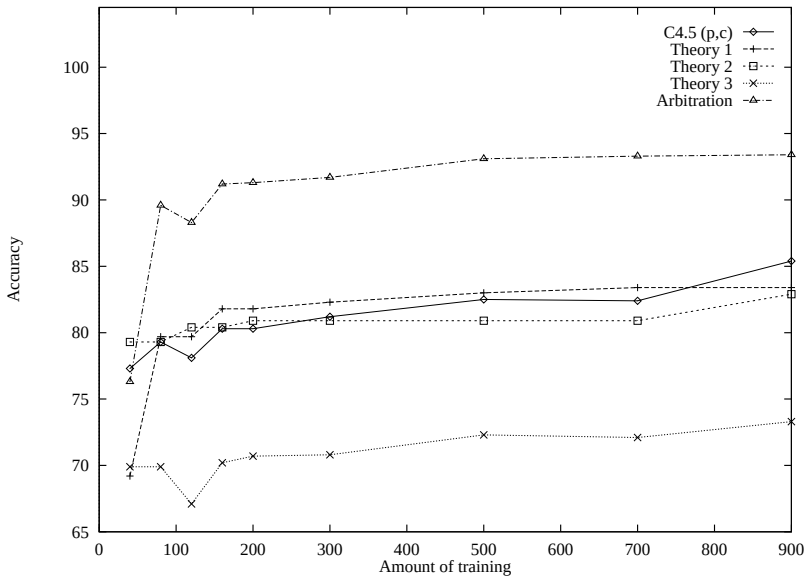


Figure 8: Learning curves for classification accuracy of: C4.5 with primitives and theory classifications, three flawed theories combined with their referees (using internal nodes), and **Arbitrate** for the three theories. Each curve was generated using 10-fold cross-validation on 1000 examples, by withholding different numbers of examples from each 900-example training set to create a learning curve.

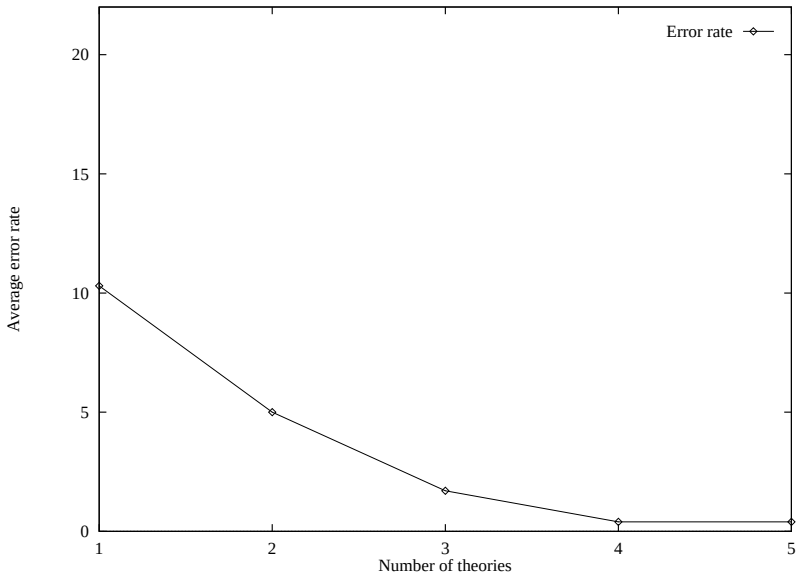


Figure 9: Average error rate in 10-fold cross-validation as the number of theories arbitrated between increases. See text for full explanation.

geometrically[4], which is what we would expect from nearly optimally taking advantage of differently flawed theories [1].

5. Conclusions

We have seen how different classifiers can be usefully combined by exploiting a set of training examples to discover for which subdomain each classifier is most reliable. In particular, we have seen that such methods are more effective than various voting techniques which do not seek subdomains of expertise. Moreover, our results suggest the general rule that the more fine-grained the analysis of the areas of expertise of the competing classifiers', the more effectively they can be combined. In particular, we have seen that we can classify far more effectively by using intermediate subconcepts which appear in the classifiers themselves as features for induction. This sort of constructive induction is effective in general for bolstering learning algorithms like C4.5. It is, however, particularly useful for arbitration because it is precisely the intermediate concepts of a given flawed classifier which can be used to isolate the subdomains of expertise of that classifier.

We have also seen that for arbitration to be effective, the component classifiers must cover different portions of the data descriptions space and the representation of the data must contain features that are relevant to the task of distinguishing between those different portions of the data description space. If these favorable conditions are met, arbitration can produce results considerably superior to those of any individual component classifier. If not, however, a performance penalty may result. In such cases, a more conservative approach may be indicated.

We believe that the approach described in this paper can be extended to address the problem of theory revision when two or more pre-existing classifiers or learning algorithms exist for a domain. Our **Arbitrate** approach results in a collection of decision trees, each starting at the root of a referee's decision tree. The leaves of a referee's decision tree predicting *Correct* will contain the corresponding model (either an existing propositional classifier, or a learned decision tree). Using the techniques like those described by Quinlan in [42], this collection of trees could be pruned and refined into a single propositional classifier. This classifier could be presented to human experts for review and could also become a pre-existing theory with which to start a new iteration of theory revision when a new batch of fresh data is obtained. The theory in such an iterated theory revision process can become the memory that allows transfer of expertise in a lifelong-learning setting [47].

Although in this paper we only address classification problems, the ideas we have discussed can also be applied to numerical prediction problems for which multiple competing models exist, such as predicting glucose levels in diabetic patients [10]. In general, C4.5 or a numerical prediction method similar to C4.5 (such as CART [4] and MARS [21]) could be used, analogously to C4.5's use described in this paper, to estimate the reliability and likely error of each component predictor.

4. When very low error rates are reached, though, it seems the decrease is reduced. In our experiment, the error rates for four theories and five theories are both 0.4%, corresponding to classifying just 4 examples out of 1000 incorrectly.

References

[1] K. Ali and M. Pazzani. Error reduction through learning multiple descriptions. *Machine Learning*, 24(3), 1996.

[2] P. T. Baffes and R. J. Mooney. Symbolic revision of theories with M-of-N rules. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, Chambery, France, 1993.

[3] F. Bergadano and A. Giordana. A knowledge intensive approach to concept induction. In *Proceedings of the Fifth International Machine Learning Conference*, 1988.

[4] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, Belmont, CA, 1984.

[5] Leo Breiman. Stacked regressions. *Machine Learning*, 24:49–64, 1996.

[6] Leo Breiman. Technical note: Some properties of splitting criteria. *Machine Learning*, 24:41–47, 1996.

[7] C. E. Brodley. Addressing the selective superiority problem: Automatic algorithm/model class selection. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 17–24, Amherst, MA, 1993.

[8] P. Brunetti and W. K. Waldhaust, editors. *International Symposium on Advanced Models for Therapy of Insulin-Dependent Diabetes.* Raven Press, New York, NY, 1987.

[9] W. Buntime. Classifiers: A theoretical and empirical study. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 638–644, Sydney, Australia, 1991.

[10] E. R. Carson, U. Fischer, and editors E. Salzsieder. Special issue: Models and computers in diabetes research and diabetes care. *Computer methods and programs in biomedicine*, 32, 1990.

[11] B. Cestnik. Estimating probabilities: A crucial task in machine learning. In *Proc. Ninth European Conference on Artificial Intelligence*, pages 147–149, London, 1990. Pitman.

[12] 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.

[13] W. W. Cohen. Abductive explanation-based learning: A solution to the multiple inconsistent explanation problem. *Machine Learning*, 8(2):167–219, 1992.

[14] S. K. Donoho and L. A. Rendell. Rerepresenting and restructuring domain theories: A constructive induction approach. *Journal of Artificial Intelligence Research*, 2:411–446, 1995.

[15] G. Drastal and S. Raatz. Empirical results on learning in an abstraction space. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1989.

[16] H. Drucker. Improving regressors using boosting techniques. In *Proc. 14th International Conference on Machine Learning*, pages 107–115. Morgan Kaufmann, 1997.

[17] 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.

[18] B. Efron and G. Gong. A leisurely look at the bootstrap, the jackknife, and cross-validation. *The American statistician*, 37(1):36–48, 1983.

[19] S. P. Engelson and M. Koppel. Distilling reliable information from unreliable theories. In *Proceedings of the International Conference on Machine Learning*, 1995.

[20] N. S. Flann and T.G. Dietterich. A study of explanation-based methods for inductive learning. *Machine Learning*, 4(2):187–226, 1989.

[21] J. H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–141, 1991.

[22] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-12(10):993–1001, October 1990.

[23] T. K. Ho, J. J. Hull, and S. N. Srihari. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):66–75, 1994.

[24] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.

[25] M. I. Jordan, R. A. Jacobs, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[26] I. Kohane and S. Uckun, editors. *AAAI 1994 Spring Symposium on Artificial Intelligence in Medicine: Interpreting Clinical Data*. AAAI Press, Stanford, CA, 1994.

[27] M. Koppel and S. P. Engelson. Integrating multiple classifiers by finding their areas of expertise. In *Proceedings of the AAAI-96 Workshop On Integrating Multiple Learned Models*, 1996.

[28] M. Koppel, R. Feldman, and A. Segre. Bias-driven revision of logical domain theories. *Journal of Artificial Intelligence Research*, 1:159–208, 1994.

[29] S. W. Kwok and C. Carter. Multiple decision trees. In R. D. Shachter, T. S. Levitt, L. N. Kanal, and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence 4*, pages 327–335. Elsevier Science Publishers B. V., North Holland, 1990.

[30] M. Malliaris and L. Salchenberger. A neural network model for estimating option prices. *Journal of Applied Intelligence*, 3(3):193–206, 1993.

[31] Ofer Matan. Ensembles for supervised classification learning. Thesi CS-TR-97-1587, Stanford University, Department of Computer Science, March 1997.

[32] C. J. Merz. Dynamic learning bias selection. In *Preliminary Papers of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 386–393, Ft. Lauderdale, FL, 1995.

[33] C. J. Merz. Combining classifiers using correspondence analysis. In *Advances in Neural Information Processing*, 1997.

[34] R. S. Michalski and R. L. Chilausky. Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *International Journal of Policy Analysis and Information Systems*, 4(2):125–161, 1980.

[35] R. S. Michalski, editor. Special issue on multistrategy learning. *Machine Learning*, 11:109–261, 1993.

[36] R. J. Mooney. Induction over the unexplained: Using overly-general theories to aid concept learning. *Machine Learning*, 10:79–110, 1993.

[37] P. M. Murphy and D. W. Aha. *UCI Repository of Machine Learning Databases*. Department of Information and Computer Science, University of California at Irvine, Irvine, CA, 1992.

[38] J. Ortega. *Making The Most Of What You've Got: Using Models and Data To Improve Prediction Accuracy*. PhD thesis, Vanderbilt University, May 1996.

[39] J. Ortega and D. Fisher. Flexibly exploiting prior knowledge in empirical learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1041–1049, 1995.

[40] D. Ourston. *Using Explanation-Based and Empirical Methods in Theory Revision*. PhD thesis, University of Texas, Austin, TX, 1991.

[41] M. Pazzani and D. Kibler. The utility of knowledge in inductive learning. *Machine Learning*, 9(1):57–94, 1992.

[42] J. R. Quinlan. Simplifying decision trees. *International Journal of Man-machine Studies*, 1988.

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

[44] B. L. Richards and R. J. Mooney. Refinement of first-order horn-clause domain theories. *Machine Learning*, 19(2):95–131, 1995.

[45] C. Schaffer. Selecting a classification method by cross-validation. In *Preliminary Papers of the Fourth International Workshop on Artificial Intelligence and Statistics*, pages 15–25, Ft. Lauderdale, FL, 1993.

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

[47] T. Thrun and T. Mitchell. Learning one more thing. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, San Mateo, CA, August 1995. Morgan Kaufmann.

[48] K. M. Ting, B. T. Low, and I. H. Witten. Learning from batched data: Model combination vs data combination. *Knowledge and Information Systems: An International Journal*, In Press.

[49] Kai Ming Ting. The characterisation of predictive accuracy and decision combination. In *Proc. 13th International Conference on Machine Learning*, pages 498–506. Morgan Kaufmann, 1996.

[50] Kai Ming Ting and Ian H. Witten. Stacked generalization: when does it work? In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 866–873, San Francisco, August 23–29 1997. Morgan Kaufmann Publishers.

[51] G. G. Towell, J. W. Shavlik, and M. O. Noordewier. Refinement of approximate domain theories by knowledge-based neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 861–866, Boston, MA, 1990.

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