Under consideration for publication in Knowledge and Information Systems

Cost Complexity-based Pruning of Ensemble Classifiers

Andreas L. Prodromidis and Salvatore J. Stolfo

Department of Computer Science, Columbia University, New York, NY 10027

Abstract. In this paper we study methods that combine multiple classification models learned over separate data sets. Numerous studies posit that such approaches provide the means to efficiently scale learning to large datasets, while also boosting the accuracy of individual classifiers. These gains, however, come at the expense of an increased demand for run-time system resources. The final ensemble meta-classifier may consist of a large collection of base classifiers that require increased memory resources while also slowing down classification throughput. Here, we describe an algorithm for pruning (i.e. discarding a subset of the available base classifiers) the ensemble meta-classifier as a means to reduce its size while preserving its accuracy and we present a technique for measuring the tradeoff between predictive performance and available run time system resources. The algorithm is independent of the method used initially when computing the meta-classifier. It is based on decision tree pruning methods and relies on the mapping of an arbitrary ensemble meta-classifier to a decision tree model. Through an extensive empirical study on meta-classifiers computed over two real data sets, we illustrate our pruning algorithm to be a robust and competitive approach to discarding classification models without degrading the overall predictive performance of an ensemble computed over those that remain after pruning.

Keywords: distributed data mining, meta-learning, classifier evaluation, pruning, ensembles of classifiers, credit card fraud detection.

1. Introduction

Recently, there has been considerable interest in *meta-learning* techniques that combine or integrate an ensemble of models computed by the same or different learning algorithms over multiple data subsets (Chan, Stolfo and Wolpert, 1996; Dietterich, 1997). An advantage of such an approach is that it can improve both *efficiency* and *scalability* by executing the machine learning processes *in parallel* and on (possibly disjoint) subsets of the data (*a data reduction technique*). Moreover, it can produce a "higher quality" final classification model, also called *meta-classifier*, by combining classifiers with different inductive bias (Mitchell, 1982).

The JAM system (Java Agents for Meta-learning) (Stolfo, Prodromidis, Tselepis, Lee, Fan and Chan, 1997), for example, is a distributed agent-based data mining system that integrates multiple models. JAM takes full advantage of the inherent parallelism and distributed nature of meta-learning by providing a set of learning agents that compute models (classifier agents) over data stored locally at a site. JAM supports the remote dispatch and exchange of learning and classifier agents among the participating data base sites through a special distribution mechanism. It also provides a set of meta-learning agents that combine the computed models that were learned (perhaps) at different sites.

Several methods for integrating ensembles of models have been studied, including techniques that combine the set of models in some linear fashion (Ali and Pazzani, 1996; Breiman, 1994; Breiman, 1996; Freund and Schapire, 1995; Krogh and Vedelsby, 1995; Opitz and Shavlik, 1996; Perrone and Cooper, 1993; Schapire, 1990; Tresp and Taniguchi, 1995; LeBlanc and Tibshirani, 1993), techniques that employ referee functions to arbitrate among the predictions generated by the classifiers, (Jacobs, Jordan, Nowlan and Hinton, 1991; Jordan and Jacobs, 1994; Jordan and Xu, 1993; Kong and Dietterich, 1995; Ortega, Koppel and Argamon-Engelson, 1999; R. and J., 1994), methods that rely on principal components analysis (Merz, 1999; Merz and Pazzani, 1999) or methods that apply inductive learning techniques to learn the behavior and properties of the candidate classifiers (Chan and Stolfo, 1993; Wolpert, 1992).

Constructing ensembles of classifiers is not cheap and produces a final outcome that is expensive due to the increased complexity of the final meta-classifier. In general, meta-classifiers combine all their constituent base-classifiers. Hence, to classify unlabeled instances, predictions need to be generated from all base-classifiers before the meta-classifier can produce its final classification. This results in significant decrease in classification throughput (the speed with which an unknown datum can be classified) and increased demand for system resources (including memory to store base classifiers).

From experiments conducted on a Personal Computer with a 200MHz Pentium processor running Solaris 2.5.1 where base- and meta-classifiers were trained to detect credit card fraud, we measured a decrease of 50%, 70% and 80% credit card transaction processing throughput for meta-classifiers composed of 13, 20 and 30 base-classifiers, respectively. Meta-classifier throughput is crucial in real-time systems, such as e-commerce or intrusion detection systems. Memory constraints are equally important. For the same problem, a single ID3 decision tree (Quinlan, 1986) may require more than 850KBytes of main memory, while a C4.5 decision tree (Quinlan, 1993) may need 100KBytes. Given the phenomenal growth in the number and size of the databases and data warehouses, retaining and deploying an increasing number of base classifiers and meta-classifiers may not be practical nor feasible.

In this paper we consider pruning techniques that aim to reduce the complexity and cost of ensemble meta-classifiers and we describe a technique for studying the tradeoff between predictive performance and available resources. We introduce a *post-training pruning* algorithm for discarding the base classifiers that are redundant or less "relevant" to the meta-classifier. *Post-training pruning* is applied to a pre-existing meta-classifier.[1] The objective of pruning, here, is to

[1] Conversely, pre-training pruning (Prodromidis and Stolfo, 1998) refers to the filtering of base classifiers before they are included in an ensemble meta-classifier.

build a smaller and faster meta-classifier that achieves comparable performance (accuracy) to the unpruned meta-classifier.

Our post-training pruning algorithm is based on the pruning methods employed by decision-tree learning algorithms. In principle, it can be applied to any meta-classifier, regardless of the method used to integrate the base classifiers. In this study, we test it on meta-classifiers computed by the Naive Bayes (Minksy and Papert, 1969) and the Ripper (Cohen, 1995) learning algorithms (stacking), on meta-classifiers combined by the majority voting and weighted voting schemes and on meta-classifiers integrated by the SCANN (Merz, 1999) algorithm. The remainder of this paper is organized as follows. Section 2 provides a brief overview of these classifier-combining techniques and Section 3 introduces the post-training pruning algorithm. The effectiveness of the algorithm is empirically evaluated on classifiers and meta-classifiers computed over two credit card transaction data sets. Section 4 details these experiments and compares the results against the performance results of two simple pruning algorithms, namely, the Least Predictive algorithm and the Correlation-based algorithm. The Least Predictive algorithm selects and removes the worst predictive base classifiers while the Correlation-based algorithm determines and discards the base classifiers that are least correlated to the meta-classifier. The empirical evaluation illustrates the decision-tree-based post-training pruning method to be highly robust in reducing the size of the meta-classifiers while maintaining predictive performance. Finally, Section 5 concludes the paper.

2. Background

We consider three methods for combining the predictions of multiple base classifiers, voting, stacking and SCANN (based on Correspondence Analysis). Before we overview the three combining strategies, we introduce the following notation. Let C_i, $i = 1, 2, ..., K$, denote a base classifier computed over data subsets D_j, $j = 1, 2, ..., L$, of training data D, and M the number of possible classes. Let $\mathbf{x}$ be an instance whose classification we seek, and $C_1(\mathbf{x})$, $C_2(\mathbf{x})$,..., $C_K(\mathbf{x})$, $C_i(\mathbf{x}) \in \{y_1, y_2, ..., y_M\}$, be the predicted classifications of $\mathbf{x}$ from the K base classifiers. Finally, let V be a separate validation set of size n that is used to generate *meta-level* predictions of the K base classifiers.

2.1. Voting

Voting denotes the simplest method of combining predictions from multiple classifiers. In its simplest form, called *plurality* or *majority* voting, each classification model C_i, $i = 1, 2, ..., K$ contributes a single vote (its own classification). The collective prediction is decided by the majority of the votes, i.e. the class $y_j, j \in \{y_1, y_2, ..., y_M\}$ with the most votes is the final prediction. In *weighted* voting, on the other hand, the classifiers have varying degrees of influence on the collective predictions which is relative to their predictive accuracy. Each model is associated with a specific weight determined by its performance (e.g accuracy, cost model) on a validation set. The final prediction is decided by summing over all weighted votes and by choosing the class with the highest aggregate. For a binary classification problem, for example, where each classifier C_i with weight

w_i casts a 0 vote for class y_1 and a 1 vote for class y_2, the aggregate is given by:

$$S(\mathbf{x}) = \frac{\sum_{i=1}^{K} w_i C_i(\mathbf{x})}{\sum_{i=1}^{K} w_i} \tag{1}$$

If we choose 0.5 to be the threshold distinguishing classes y_1 and y_2, the weighted voting method classifies unlabeled instances $\mathbf{x}$ as y_1 if $S(\mathbf{x}) < 0.5$, as y_2 if $S(\mathbf{x}) > 0.5$ and randomly if $S(\mathbf{x}) = 0.5$.

This approach can be extended to non-binary classification problems by mapping the m-class problem into m binary classification problems and by associating each class j with a separate $S_j(\mathbf{x})$, $j \in 1, 2, ...m$. To classify an instance $\mathbf{x}$, each $S_j(\mathbf{x})$ generates a confidence value indicating the prospect of $\mathbf{x}$ being classified as j versus being classified as non-j. The final class selected corresponds to the $S_j(\mathbf{x})$, $j \in 1, 2, ...m$ with highest confidence value.

In this study, the weights w_i's are set according to the performance (with respect to a selected evaluation metric, e.g accuracy) of each classifier C_i on a separate validation set.

2.2. Stacking

The main difference between voting and *Stacking* (Wolpert, 1992) is that the latter combines base classifiers in a non-linear fashion. The combining task, called a *meta learner*, integrates the independently computed base classifiers into a higher level meta-classifier, by learning over a *meta-level training set*. This meta-level training set is composed by using the base classifiers' predictions $C_1(\mathbf{x})$, $C_2(\mathbf{x})$,..., $C_K(\mathbf{x})$, $C_i(\mathbf{x})$ on the validation set V as attribute values, and the true class as the target. From these predictions, the meta-learner learns the characteristics and performance of the base classifiers C_i, $i = 1, 2, ..., K$ and computes a meta-classifier which is a model of the "global" data set. To classify an unlabeled instance $\mathbf{x}$, the base classifiers present their own predictions to the meta-classifier which then makes the final classification.

2.3. SCANN

The third approach we consider is Merz's SCANN (Merz, 1999) (Stacking, Correspondence Analysis and Nearest Neighbor) algorithm for combining multiple models as a means to improve classification performance. As with stacking, the combining task integrates the independently computed base classifiers in a non-linear fashion. The meta-level training set is composed by using the base classifiers' predictions on the validation set as attribute values, and the true class as the target. In this case, however, the predictions and the correct class are demultiplexed and represented in a 0/1 form.[2] SCANN employs correspondence analysis (Greenacre, 1984) (similar to Principal Component Analysis) to geometrically explore the relationship between the validation examples, the models'

[2] The meta-level training set is a matrix of n rows (records) and $[(K + 1) \cdot M]$ columns (0/1 attributes), i.e. there are M columns assigned to each of the K classifiers and M columns for the correct class. If classifier $C_i(\mathbf{x}) = j$, then the j^{th} column of C_i will be assigned a 1 and the rest $(M - 1)$ columns a 0.

predictions and the true class. In doing so, it maps the true class labels and the predictions of the classifiers onto a new scaled space that clusters similar prediction behaviors. Then, the nearest neighbor learning algorithm is applied over this new space to meta-learn the transformed predictions of the individual classifiers. To classify unlabeled instances, SCANN maps them onto the new space and assigns them the class label corresponding to the closest class point.

SCANN is a sophisticated combining method that seeks to geometrically uncover the position of the classifiers relative to the true class labels. On the other hand, it relies on singular value decomposition techniques to compute the new scaled space and capture these relationships, which can be expensive, both in space and time, as the number of examples and hence as the number of the base classifiers increases (the overall time complexity of SCANN is $(O((M \cdot K)^3))$.

3. Post-training pruning

Post-training pruning refers to the pruning of a meta-classifier after it is constructed. In contrast with pre-training pruning (Prodromidis and Stolfo, 1998) which uses greedy forward-search methods to choose classifiers, post-training pruning is considered a backwards selection method. It starts with all available K classifiers (or with the classifiers pre-training pruning selected) and iteratively removes one at a time. The objective is to perform a search on the (possibly reduced) space of meta-classifiers and prune the meta-classifier without degrading predictive performance.

The deployment and effectiveness of post-training pruning methods depends highly upon the availability of unseen labeled (validation) data. Post-training pruning can be facilitated if there is an abundance of data, since a separate labeled subset can be used to estimate the effects of discarding specific base classifiers and thus guide the backwards elimination process. A hill climbing pruning algorithm, for example, would employ the separate validation data to evaluate and select the best (out of K possible) meta-classifier with $K-1$ base classifiers, then evaluate and select the best meta-classifier with $K-2$ (out of $K-1$ possible) and so on. In the event that additional data is not available, standard cross validation techniques can be used, instead, to estimate the performance of the pruned meta-classifier, at the expense of increased complexity.

A disadvantage of such a method, besides the need for a separate data set, and its vulnerability to the horizon effect, is the overhead of constructing and evaluating all the intermediate meta-classifiers ($O(K^2)$ in the example), which, depending on the combining methods (e.g the learning algorithm in stacking) and the data set size, can be prohibitive.

To our knowledge, Margineantu and Dietterich's "Pruning Adaptive Boosting" (Margineantu and Dietterich, 1997), constitutes the only other (besides our prior work on pre-training pruning methods) related research on this subject. In that paper, the authors study the problem of pruning the ensemble of classifiers obtained by the boosting algorithm AdaBoost (Freund and Schapire, 1996). Their research however, was restricted to computing all classifiers by applying the *same* learning algorithm on many different subsets of the *same* training set. Furthermore, their most effective pruning algorithm, "Reduce-Error Pruning with Backfitting" suffers from the same disadvantage of computing intermediate meta-classifiers as our hill climbing method.

Next, we describe an efficient post-training pruning algorithm that does not

require intermediate meta-classifiers or separate validation data. Instead it extracts information from the ensemble meta-classifier itself and employs the meta-classifier training set seeking to minimize (meta-level) training error. Furthermore, we consider the more general setting where the final classifications may be obtained be any meta-learning technique, not just voting (ADABOOST), and ensembles of classifiers may be generated by applying, possibly, different learning algorithms over (possibly) distinct databases,

3.1. Cost complexity pruning

The algorithm is based on the *minimal cost complexity pruning* method of the CART decision tree learning algorithm (Breiman, Friedman, Olshen and Stone, 1984). CART computes a large decision tree T_0 to fit the training data, by allowing the splitting process to continue until all terminal nodes are either small, or pure (i.e. all instances belong to the same class) or contain only instances with identical attribute-value vectors. Next, it applies the cost complexity pruning method to compute a set of consecutive nested subtrees $T_i, i \in \{1, 2, ..., R\}$ of decreasing size from the original large tree T_0 by progressively pruning upwards to its root node (T_R corresponds to the subtree that consists of the root node only).

To compute the set of these nested subtrees $T_i, i \in \{1, 2, ..., R\}$, the cost complexity pruning method associates a complexity measure $C(T)$ with the number of terminal nodes of decision tree T. The method prunes decision trees by seeking to minimize a cost complexity metric $R_\alpha(T)$ that combines two factors, the size (complexity) and the accuracy (or equivalently the error rate) of the tree. Specifically, $R_\alpha(T)$ is defined as

$$R_\alpha(T) = R(T) + \alpha \cdot C(T) \tag{2}$$

where $R(T)$ denotes the misclassification cost (error rate[3]) of the decision tree T and α represents a complexity parameter ($\alpha \geq 0$).

The degree of pruning of the initial tree T_0 can be controlled by adjusting the complexity parameter α, which, according to Equation 2 corresponds to the weight of the complexity factor $C(T)$. If α is small, the penalty for having a large number of terminal nodes is small; as the penalty α per terminal node increases, the pruning algorithm removes an increasing number of terminal nodes in an attempt to compensate and thus generates the nested subtrees $T_i, i \in \{1, 2, ..., R\}$. The *minimal cost complexity pruning* method guarantees to find the "best" (according to the misclassification cost) pruned decision tree T_r, $r \in \{1, 2, ..., R\}$, of the original tree T_0 of a specific size (as dictated by the complexity parameter). An alternative pruning method using Rissanen's minimum description length is described in (Quinlan and Rivest, 1989).

3.2. Pruning meta-classifiers

The post-training pruning algorithm employs the minimal cost complexity method as a means to reduce the size (number of base-classifiers) of the meta-classifiers.

[3] Estimated over a separate pruning subset of the training set or using cross validation methods.

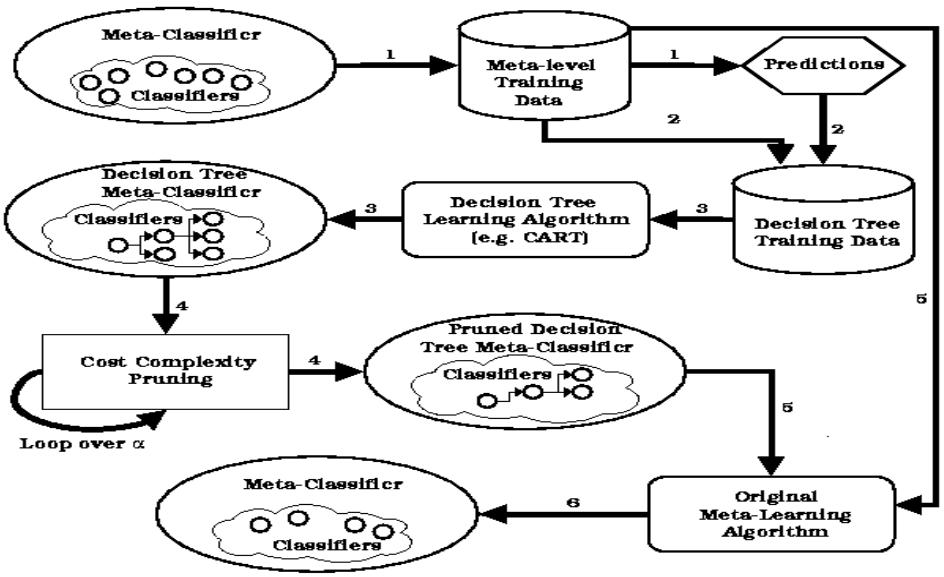

Fig. 1. The six steps of the Post-Training Pruning Algorithm

In cases where the meta-classifier is built via a decision tree learning algorithm, the use of the cost complexity pruning method is straightforward. The decision tree model constitutes the ensemble meta-classifier and each node corresponds to a single base-classifier. Thus, by determining which nodes to remove, the pruning algorithm discovers and discards the base classifiers that are least important.

To apply this method on meta-classifiers of arbitrary representation (i.e. non decision-tree), the post-training pruning algorithm maps the meta-classifier to its "decision-tree equivalent". In general, the algorithm can be considered as a multi-phase process. Initially, the algorithm seeks to model the arbitrary meta-classifier as an equivalent decision tree meta-classifier that imitates its behavior (a similar approach appeared in (Craven and Shavlik, 1996) in the context of neural networks). Then it removes as many base-classifiers as needed using the minimal cost complexity pruning method on the derived decision tree model, and finally it re-combines the remaining base classifiers using the original meta-learning algorithm. Hereinafter, the term "decision tree model" will refer to the decision tree trained to imitate the behavior of the initial arbitrary meta-classifier.

These phases are graphically illustrated in six steps in Figure 1. For completeness, a detailed description of the post-training pruning algorithm is provided in Table 1.

1. The meta-classifier is applied to its own (meta-learning) training set. (Recall that the attributes of the meta-level training set correspond to the predictions

Input: Set of base classifiers $\mathcal{C} = \{C_1, C_2, ..., C_K\}$, validation set $\mathcal{V} = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}\}$, meta-learning algorithm $\mathbf{A_{ML}}$, meta classifier $\mathbf{MC}$, decision tree algorithm $\mathbf{A_{DT}}$, throughput requirement (classifications/sec) T, convergence parameter $\delta(\delta > 0)$, stopping criteria $\epsilon(0 < \epsilon < \delta)$
Output: Pruned meta classifier $\mathbf{MC}^*$
Begin
$\mathcal{MLT} = \{< \vec{a}_1, \vec{a}_2, ..., \vec{a}_K, \vec{tc} >\}$, $\vec{a}_i[j] = C_i(\mathbf{x_j}) \forall i, \vec{tc}[j] = \text{TrueClass}(\mathbf{x_j})$, j=1, 2, ...,n, $x_j \in \mathcal{V}$;
$\mathcal{DTT} = \{< \vec{a}_1, \vec{a}_2, ...\vec{a}_K, \vec{pc} >\}$, predicted class $\vec{pc}[j] = \mathbf{MC}(\mathbf{x_j})$;
$\mathbf{DTM} = \mathbf{A_{DT}}(\mathcal{DDT})$, the decision tree model trained with $\alpha = 0$;
$K^* = K$; $\mathbf{DTM}^* = \mathbf{DTM}$; /* initialization of running variables */
$T^* = 1/\text{t}$, $t = max\{t_i | t_i = time(\mathbf{MC}(\mathbf{x_i}))$, i = 1, 2, ...,n $\}$; /* Throughput estimate of $\mathbf{MC}$ */
While $(T^* < T)$ **do**
$\delta^* = \delta$; $K^* = K^* - 1$; /* remove one classifier at a time */
L_1: **While** $\mathbf{DTM}^*$ has more than K^* classifiers **do**
$\alpha = \alpha + \delta^*$; /* increase α to remove classifiers */
$\mathbf{DTM}^* = \underset{i}{argmin}\{[R_\alpha(\mathbf{DTM_i}) + \alpha \cdot C(\mathbf{DTM_i})] | \mathbf{DTM_i} =$ subtree of $\mathbf{DTM}\}$;
end while
if ($\mathbf{DTM}^*$ has too many classifiers pruned (i.e. less than K^*) and $\delta^* > \epsilon$)
reset $\alpha = \alpha - \delta^*$; adjust $\delta^* = \delta^*/2$; $\mathbf{DTM}^* = \mathbf{DTM}$; **goto** L_1;
$\mathcal{C} = \mathcal{C} - \hat{C}$, $\hat{C}$ = the classifier that is not included in $\mathbf{DTM}^*$;
$\mathcal{MLT}^* = \{< \vec{a}_{j_1}, \vec{a}_{j_2}, ..., \vec{a}_{j_{K^*}}, \vec{tc} >\}$, $C_{j_1}, C_{j_2}, ..., C_{j_{K^*}} \in \mathcal{C}$ (retained classifiers);
$\mathbf{MC}^* = \mathbf{A_{ML}}(\mathcal{MLT}^*)$;
$T^* = 1/t$, $t = max\{t_i | t_i = time(\mathbf{MC}^*(\mathbf{x_i}))$, i = 1, 2, ...,n $\}$;
end while
$\mathcal{MLT}^* = \{< \vec{a}_{j_1}, \vec{a}_{j_2}, ..., \vec{a}_{j_{K^*}}, \vec{tc} >\}$, $C_{j_1}, C_{j_2}, ..., C_{j_{K^*}} \in \mathcal{C}$;
$\mathbf{MC}^* = \mathbf{A_{ML}}(\mathcal{MLT}^*)$;
End

Table 1. Post-Training Pruning Algorithm

of the base classifiers on the validation set and the true class labels correspond to the correct classes of the validation set.)

2. A new training set, called decision tree training set, is composed by using the meta-level training set (without the true class labels) as attributes, and the predictions of the meta-classifier on the meta-level training set as the true class target.
3. A decision-tree-based algorithm, such as CART, computes the "decision-tree equivalent" of the original meta-classifier by learning its input/output behavior recorded in the decision-tree training set. The resultant decision-tree meta-classifier is trained to imitate the behavior of the original meta-classifier and discover the manner in which it combines its constituent base classifiers. Furthermore, this decision tree reveals the base classifiers that do not participate in the splitting criteria and hence are irrelevant to the meta-classifier. Those base classifiers that are deemed irrelevant are pruned in order to meet the performance objective.
4. The next stage aims to further reduce the number of selected base classifiers, if necessary, according to the restrictions imposed by the available system resources and/or the runtime constraints. The post-training pruning algorithm applies the minimal cost complexity pruning method to reduce the size of the decision tree and thus prune away additional base classifiers. The degree of

pruning can be controlled by the complexity parameter α, as described in Section 3.1. (The loop in the figure corresponds to a bi-section search for the proper value of the α parameter). Since minimal cost complexity pruning eliminates first the branches of the tree with the least contribution to the decision tree's performance, the base classifiers pruned during this phase will also be the least "important" base classifiers.

5. A new meta-level training set is composed by using the predictions of the remaining base classifiers on the original validation set as attributes and the true class labels as target. This meta-level training set is the new input to the original meta-learning algorithm.
6. Finally, the original meta-learning algorithm is trained over this new meta-level training set to re-compute the pruned ensemble meta-classifier.

3.2.1. Remarks

The algorithm is guaranteed to terminate due to the finite structure of the decision tree and the bounded number of times ($\lceil \log_2(\delta/\epsilon) \rceil$) the inner loop will be executed. Moreover, Theorem 3.10 of (Breiman et al., 1984) proves the monotonic relation between the decreasing size of the decision tree as the complexity parameter α increases. Furthermore, the cost-complexity pruning algorithm is computationally rapid. The pruning steps require only a small fraction of the total tree construction time (Breiman et al., 1984).

The success of the post-training pruning algorithm depends on the degree the decision tree learning algorithm "overfits" the original meta-classifier. We choose to train the model of the meta-classifier on the very same data that was used to compute it in the first place. In some sense this is equivalent to generating test suites of test sets to exercise all program execution paths. By modeling the original meta-classifier based on its responses to its own training set we ensure that the decision tree learning algorithm has access to the most inclusive information regarding the meta-classifier's behavior. Furthermore, we do not require a separate *pruning* data set.

This post-training pruning method is algorithm and representation independent, i.e. it does not examine the internal structure and strategies of the learning algorithms that generate the base classifiers, nor the assumptions and characteristics of the underlying data or their schema definitions. Instead, it treats the base classifiers as black boxes and composes an independent training set (i.e., the decision tree training set) based on the input/output of each of these base classifiers.

Another benefit of the post-training pruning algorithm comes from the modeling of the original meta-classifier by a decision tree classifier. In general, an ensemble meta-classifier may have an internal representation that we cannot easily view or parse (except, of course, for its constituent base classifiers). By inducing a "decision tree equivalent" model, the algorithm generates an alternative and more declarative representation that we can inspect. Other related methods for describing and computing comprehensible models of ensemble meta-classifiers have been studied in the contexts of Knowledge Acquisition (Domingos, 1997), Knowledge Probing (Guo and Sutiwaraphun, 1998) and meta-classifier correlation-based visualization tools (Prodromidis, Stolfo and Chan, 1999).

Computing decision tree models as part of the post-training pruning algorithm is not only useful for pruning or for explaining the behavior of the meta-

classifier. These intermediate models are also meta-classifiers and hence can also be used to classify unlabeled instances. Recall, that they are also trained over the predictions of the available base classifiers, albeit for a different target class. As a result, their predictive accuracy may be inferior to that of the original meta-classifier. On the other hand, in a distributed data mining system, such as JAM, where classifiers and meta-classifiers can be exchanged and used as black boxes, it may not be possible to prune the imported meta-classifiers to adhere to local constraints (e.g if the original meta-learning algorithm is not known or not accessible). In this case, it may be preferable to trade some of their predictive accuracy for the more efficient pruned decision tree meta-classifiers.

The next section describes a comprehensive set of experiments that evaluates our post-training pruning algorithm and attempts to shed light on these issues. Specifically, we address the following questions:

- How accurately can a decision tree-based algorithm learn the behavior of a meta-classifier of arbitrary representation?
- What is the penalty of using the decision-tree meta-classifiers instead of the original meta-classifiers?
- How does the cost-complexity based pruning method compare to a more straightforward approach that evaluates and discards the worst base classifiers?
- How robust is post-training pruning? What is the tradeoff between the predictive accuracy and the classification throughput of the meta-classifier as we increase the degree of pruning?

4. Empirical Evaluation

We evaluated the post-training pruning algorithm on meta-classifiers trained to predict credit card transactions as legitimate or fraudulent. Results of the simple pruning methods as well as the pre-training pruning methods on the same task are reported in (Prodromidis and Stolfo, 1998).[4] We obtained two large databases from Chase and First Union banks, members of FSTC (Financial Services Technology Consortium) each with 500,000 records (69 MBytes of data) of sampled credit card transaction data spanning one year. The schemas of the databases were developed over years of experience and continuous analysis by bank personnel to capture important information for fraud detection. The records have a fixed length of 137 bytes each and about 30 numeric and categorical attributes including the binary class label (fraud/legitimate transaction). Chase bank data consisted of 20% fraud and 80% legitimate transactions, whereas First Union data consisted of 15% versus 85% of fraud/legitimate distribution.

To compute the set of base classifiers we employed five different inductive learning programs, Bayes, C4.5, ID3, CART and Ripper. Bayes, implements a naive Bayesian learning algorithm described in (Minksy and Papert, 1969), ID3 (Quinlan, 1986), its successor C4.5 (Quinlan, 1993), and CART (Breiman et al., 1984) are decision tree based algorithms, and Ripper (Cohen, 1995) is a rule induction algorithm.

[4] Additional multiple-model experiments with results on other tasks and data sets can be found in (Fan, Lee, Stolfo and Miller, 2000; Fan, 2000).

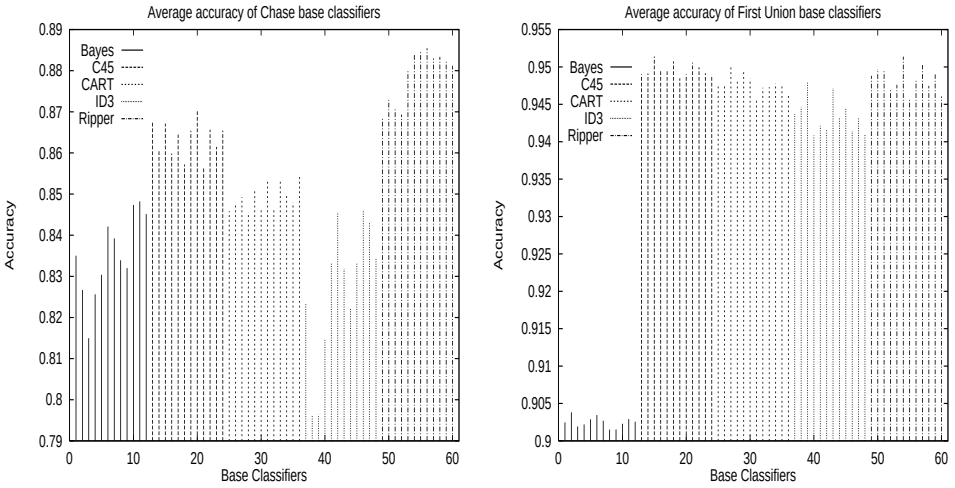

Fig. 2. Accuracy of base classifiers on Chase(left) and First Union (right) data.

We evaluated the various classification models from two perspectives, their predictive performance and their efficiency. To measure and compare predictive performance we used several different metrics including accuracy, TP-FP spread (TP and FP stand for True Positive, and False Positive respectively), and a cost model fit to the credit card fraud detection problem. Overall accuracy expresses the ability to give correct predictions, TP-FP denotes the ability to catch fraudulent transactions while minimizing false alarms and the cost model captures the performance of a classification model with respect to the goal of the target application (stop loss due to fraud). To compare the efficiencies of the various ensemble meta-classifiers we measured classification throughput. Throughput here denotes the rate at which a stream of data items can be piped through and labeled by a meta-classifier.

Due to space constraints, however, we limit this report to (mostly) accuracy and throughput results. The emphasis in this paper is not on presenting an effective fraud detection method, but rather on the evaluation of the post-training pruning algorithm as a general method for reducing the size of an ensemble meta-classifier. Detailed information on effective fraud detectors with extensive results (TP-FP spread and cost model) from the mining of these credit card data sets can be found in (Prodromidis and Stolfo, 1999*a*).

4.1. Computing Base Classifiers

The first step involves the training of the base classifiers. We split each data set in 12 subsets (each subset corresponding to one month's data) and distribute them across six different data sites (each site storing two subsets). Then we apply the 5 learning algorithms on each subset of data, therefore creating 60 classifiers (10 classifiers per data site). Next, each data site imports all "remote" base classifiers (50 in total) to test them against its "local" data. In essence, each classifier is evaluated on 5 different (and unseen) subsets.

Figure 2 presents the averaged accuracy results for the Chase (left plot) and

Bank data	Majority	Weighted	Bayes	Ripper	SCANN
Chase	89.58%	89.59%	88.65%	89.66%	89.74%
First Union	96.16%	96.19%	96.21%	96.53%	96.44%

Table 2. Performance of the meta-classifiers

First Union (right plot) credit card data, respectively. The x-axis represents the base classifiers, and each vertical bar corresponds to a single model (60 in total). The first set of 12 bars denotes Bayesian classifiers, the second set of 12 bars denotes C4.5 classifiers etc. Each bar in an adjacent group of 12 bars corresponds to a specific subset used to train the classifiers. For example, the first bar of the left plot represents the accuracy (83.5%) of the Bayesian classifier that is trained on the first data subset, the second bar represents the accuracy (82.6%) of the Bayesian classifier that is trained on the second data subset and the 13^{th} bar of the same plot (the first of the second group of 12 adjacent bars) represents the accuracy 86.8% of the C4.5 classifier that is trained on the first subset.

With the exception of two ID3 base classifiers on the Chase data set, all base classifiers demonstrate significantly better results comparing to the trivial mean predictor (80% for the Chase and 85% for the First Union data). Moreover, by combining these separately learned classifiers, it is possible to generate new models with improved fraud detection capabilities. Overall, the learning algorithms performed better on the First Union data set than on the Chase data set. To justify this result, one needs to take into consideration that there are fewer fraudulent transactions in the First Union data and this causes a higher baseline accuracy. Next, we combine these separately learned classifiers under the voting, stacking and SCANN strategies hoping to generate meta-classifiers with improved fraud detection accuracies.

4.2. Combining Base Classifiers

Although the sites are populated with 60 classification models, in meta-learning they combine only the 50 "imported" base classifiers. Since each site uses its local data to meta-learn (the first subset as the validation set and the second subset as the test set) it avoids using the 10 "local" base classifiers to ensure that no classifiers predict on their own training data. We applied 5 different meta-learning methods at each site: the 2 voting strategies (majority and weighted), the 2 stacking strategies corresponding to the Bayes and Ripper learning algorithms and the SCANN algorithm. Since all sites meta-learn the base classifiers independently, the setting of this experiment corresponds to a 6-fold cross validation with each fold executed in parallel.

The performance results of these meta-classifiers averaged over the 6 sites are reported in Table 4.2. A comparison to Figure 2 indicates that meta-classifiers outperform all base classifiers, and in most cases by a significant margin. Due to the size of the data sets, the empirical results exhibit a very low variance. In general, a 0.1% accuracy difference between two methods is a statistically significant result with 99% confidence according to the 2-tailed paired t test. On the other hand, the meta-classifiers are composed of 50 base classifiers suggesting a significant increase in resource requirements (e.g memory, CPU time) and substantial reduction in classification throughput. Post-training pruning seeks

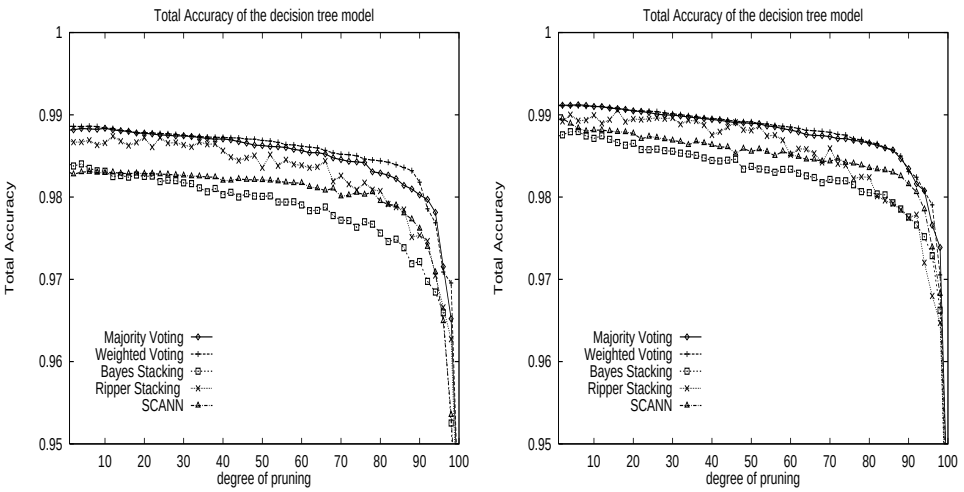

Fig. 3. Accuracy of decision tree models in emulating the behavior of Chase (left) and First Union (right) meta-classifiers

to address this issue. The objective is to remove a subset of the constituent base classifiers while preserving the accuracy gains.

4.3. Post-training pruning experiments

The first phases of the post-training pruning seek to learn and prune a decision tree model of the original meta-classifier. Initially, we applied the CART learning algorithm on the decision tree training set of each meta-classifier. The initial decision-tree models were generated with the complexity parameter α set to zero to allow the growth of trees that are as accurate and as "close" to the original meta-classifiers as possible. Then, by increasing the α parameter (i.e. the degree of pruning), we gradually computed trees with fewer nodes (thus pruning base classifiers). One can quantify the effectiveness of the derived models by comparing their predictions against the predictions of their corresponding meta-classifiers on the separate test sets.

The accuracy of the decision tree models and the impact of pruning on their performance is shown in Figure 3. Each plot (left for Chase, right for First Union) displays the average accuracy (y-axis) of the decision tree algorithm in imitating the behavior of the 5 different methods for constructing ensembles. The x-axis represents decision-tree models of progressively smaller sizes. The initial (unpruned) model corresponds to the left-most points of each curve (i.e. degree of pruning is 0%). Naturally, an accuracy result of 100% would signify a decision tree model that has perfectly learned to imitate its meta-classifier. According to this figure, learning the behavior of these meta-classifiers has been fairly successful, with decision tree models achieving a 98%-99% accuracy rate, even when pruning is as heavy as 80%.

The last phase of the post-training pruning algorithm aims to re-combine the remaining base classifiers (those included in the decision tree model) using the original meta-learning algorithm. To evaluate the effectiveness of the prun-

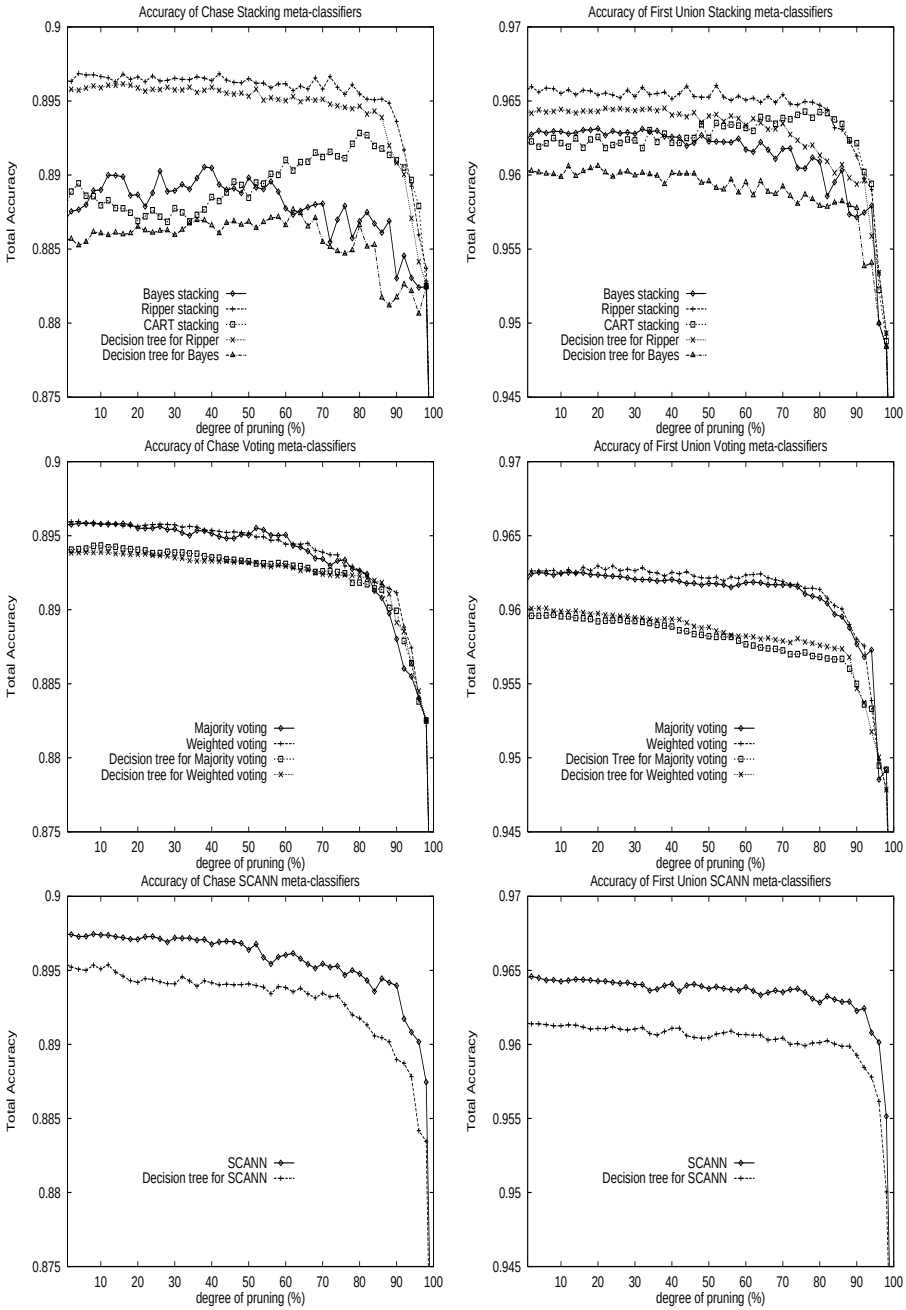

Fig. 4. Accuracy of the Stacking (top plots), Voting (middle plots) and SCANN (bottom plots) meta-classifiers and their decision tree models for Chase (left) and First Union (right) data

ing algorithm, we measured the accuracy of all the intermediate meta-classifiers obtained as we increased the degree of pruning from 0% (original unpruned meta-classifier) to 100% (all base classifiers are pruned, default prediction is the most frequent class, i.e. legitimate transaction).

In Figure 4 we present the accuracy results for the Chase (left side plots) and the First Union (right side plots) meta-classifiers. The plots at the top row correspond to the stacking meta-learning methods (Bayes, Ripper), the plots at the center row represent the voting (majority, weighted) meta-learning methods and the plots at the bottom row belong to the SCANN meta-learning method. These graphs demonstrate that post-training pruning was quite successful in all cases examined. The algorithm determined and pruned the redundant and/or the less "contributing" base classifiers, and generated substantially smaller meta-classifiers without significant performance penalties.

The graphs also depict the overall accuracy of the decision tree models of the meta-classifiers. Recall, that these are models trained to "imitate" the behavior of an ensemble, not to directly detect fraudulent transactions. As expected, their predictive performance is inferior to that of their corresponding (pruned) meta-classifier. On the other hand, it is interesting to note that it is possible to compute decision tree models that outperform other original meta-classifiers. In these experiments, for example, the decision tree model of Ripper (and those of the voting meta-classifiers) outperforms the original Bayesian meta-classifier.

To further explore the last observation, we also measured the accuracy of the original stacking CART meta-classifier as a function of the degree of pruning. In other words, we studied the direct performance of a decision tree learning algorithm as a method of combining base classifiers. The results are depicted by the curve denoted as "CART stacking". Surprisingly, CART appears to be more effective when learning to combine base classifiers indirectly (e.g. by observing the Ripper meta-classifier) than through direct learning of this target concept. This suggests that searching the hypothesis space as modeled by a previously computed meta-classifier may, in some cases, be easier than searching the original hypothesis space. It is not entirely clear why this may be so, and is a subject of further study. As a result, in cases where meta-classifiers are considered as black boxes, or meta-learning algorithms are not available, it may be beneficial to compute, prune and use their "decision tree equivalents" instead.

4.4. Cost Complexity vs. Least Predictive vs. Correlation Pruning

The Least Predictive and Correlation Pruning are two pruning algorithms that use simple and straightforward criteria to discard base classifiers. The first method associates each base classifier with a specific weight determined by its performance (e.g. accuracy, cost model) on a validation set. Then it prunes the meta-classifier by removing the base classifiers with the lowest weights.

The second method measures the degree of "similarity" between the meta-classifier and each base classifier. In particular, we define as *Correlation* between two classifiers, the ratio of the instances in a validation set where the two classifiers produce identical predictions. In the meta-learning context, the Correlation between a meta-classifier and a base classifier denotes the degree the meta-classifier agrees with the base classifier. The Correlation-based algorithm that we consider as part of our experiments, removes the base classifiers that are

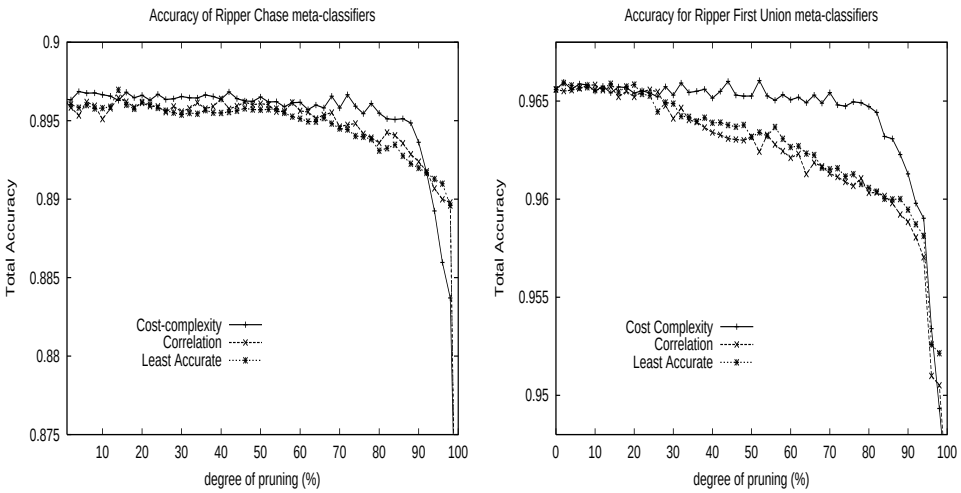

Fig. 5. Comparison of three post-training pruning algorithms on Chase (left) and First Union (right) credit card data.

least correlated to the meta-classifier. It begins by identifying and removing the base classifier that is least correlated to the initial meta-classifier, and continues iteratively, by computing a new reduced meta-classifier (with the remaining base classifiers), then it searches for the next least correlated base classifier and removes it. The method stops when enough base classifiers are pruned.

We compared these two algorithms and the cost complexity-based algorithm by applying them to the unpruned meta-classifiers. The accuracy results for the two data sets are presented in Figure 5, with the Chase graph at the left side, and the First Union graph at the right side. Note that the Least Predictive plot is marked as "Least Accurate" to reflect the fact that the weights of the base classifiers in the Least Predictive algorithm are calculated based on their accuracy.

According to these figures, when the pruning was light, the three methods exhibited comparable performance. This can be attributed to the fact that the pruned meta-classifiers were very "similar", i.e. they had many common base classifiers. But this changed as the degree of pruning increased and each method discarded different sets of base classifiers. The cost-complexity post-training pruning algorithm computed superior meta-classifiers compared to those of the Least Accurate and Correlation-based pruning methods. Overall, the cost-complexity algorithm is successful in pruning the majority of the base classifiers with less performance penalties than the other two methods. Furthermore, a closer inspection of the graphs shows that the two simpler methods performed better with Chase base classifiers that with First Union base classifiers.

The least accurate and the least correlated base classifiers are not always the "least important" base classifiers. Pruning them away may cause the search process to not consider "good" combinations. In general, the more diverse the base classifiers are, the more room a meta-classifier has to improve its performance. In these experiments, the Chase base classifiers were more diverse than the First

	Chase			First Union		
Algorithm	Accuracy	TP-FP	Savings	Accuracy	TP-FP	Savings
Best base classifier	88.50%	0.551	$ 812K	95.20%	0.749	$ 806K
Best meta classifier	89.74%	0.632	$ 903K	96.53%	0.848	$ 950K

Table 3. Performance of the meta-classifiers.

Union base classifiers. (This is evident from the graphs of Figure 2.)[5] As a result, in the Chase bank case, the Least Accurate pruning algorithm combined the most accurate base classifiers that were already more diverse. Hence it achieved better results than in the First Union case, where the best base classifiers were not as diverse.

The degree of pruning of a meta-classifier within a data site is dictated by the classification throughput requirements of the particular problem. In general, higher throughput requirements necessitate heavier pruning. The last set of experiments investigates the trade-off between throughput and predictive performance.

We measure predictive performance by using three different metrics: overall accuracy, $TP - FP$ spread,[6] and a realistic cost model fit to this problem.

Credit card companies have a fixed overhead that serves as a threshold value for challenging the legitimacy of a credit card transaction. If the transaction amount *amt*, is below this threshold, they choose to authorize the transaction automatically. Each transaction predicted as fraudulent requires an "overhead" referral fee for authorization personnel to decide the final disposition. This "overhead" cost is typically a "fixed fee" that we call X. Therefore, even if we could accurately predict and identify all fraudulent transactions, those whose *amt* is less than X would produce $\$(X - amt)$ in losses anyway. To calculate the savings each fraud detector contributes due to stopping fraudulent transactions, we use the following cost model for each transaction:

- If prediction is "legitimate" or $(amt \leq X)$, authorize the transaction $(savings = 0)$;
- Otherwise investigate the transaction:
  - If transaction is "fraudulent", $savings = amt - X$;
  - otherwise $savings = -X$;

According to this cost model the perfect fraud detector would save $1,470K per month for Chase and $1,085K per month for First Union. Table 6 summarizes the predictive performance of the best base classifiers and the meta-classifiers averaged over the 6 data sites. A comprehensive analysis of the performance of the various learning and meta-learning strategies can be found in (Prodromidis and Stolfo, 1999*b*).

[5] A study with detailed measurements on the diversity of these base classifiers can be found in (Prodromidis, 1999).

[6] In comparing the classifiers, one can replace the $TP-FP$ spread, which defines a certain family of curves in the ROC plot, with a different metric or even with a complete analysis (Provost and Fawcett, 1997; Provost and Fawcett, 1998) in the ROC space.

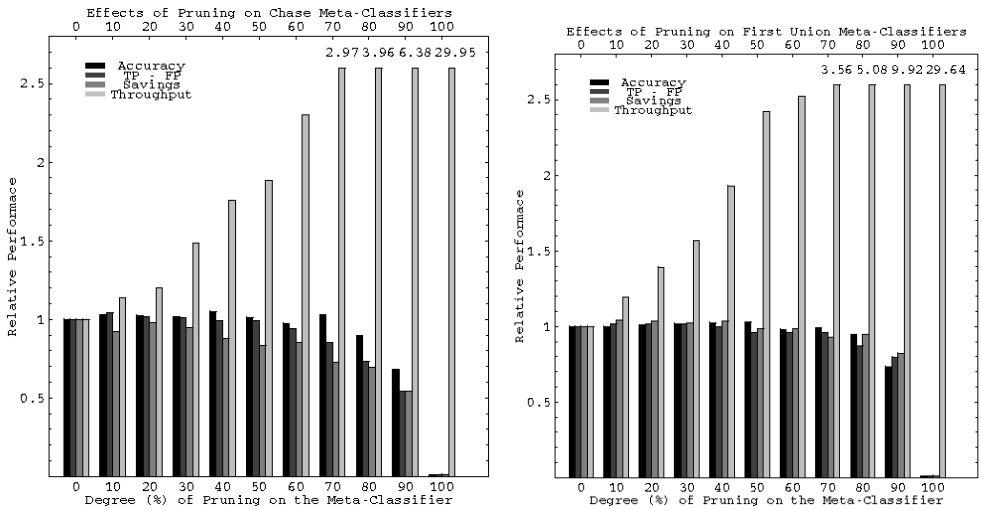

Fig. 6. Bar charts of the accuracy (black), TP-FP (dark gray), savings (light gray) and throughput (very light gray) of the Chase (right) and First Union (left) meta-classifiers as a function of the degree of pruning.

4.5. Predictive performance vs. throughput

To normalize the different evaluation metrics and better quantify the effects of pruning, we measured the ratio of the performance improvement of the pruned meta-classifier over the performance improvement of the complete (original) meta-classifiers. In other words, we measured the performance *gain* ratio: $G_{P=ACCURACY}, G_{P=TP-FP}$ and $G_{P=SAVINGS}$ where G is defined as:

$$G = \frac{P_{PRUNED} - P_{BASE}}{P_{COMPLETE} - P_{BASE}} \tag{3}$$

and P_{PRUNED}, $P_{COMPLETE}$ and P_{BASE} denote the performance (according to accuracy, TP-FP spread or cost savings) of the pruned meta-classifier, the complete meta-classifier and the best base classifier, respectively. Values of $G \simeq 1$ indicate pruned meta-classifiers that sustain the performance levels to that of the complete meta-classifier while values of $0 < G < 1$ indicate better pruned meta-classifier results than the best base classifier but with performance losses comparing to the complete meta-classifier. When only the best base classifier is used, there is no performance improvement and $G = 0$. Theoretically, a negative G would suggest a meta-classifier with worse performance than the base classifier.

Figure 6 demonstrates the algorithm's effectiveness on the Chase (left) and First Union (right) classifiers by displaying the performance gain ratio's and throughputs of the pruned Ripper stacking meta-classifiers as a function of the degree of pruning. The figure displays $G_{P=ACCURACY}, G_{P=TP-FP}$ and $G_{P=SAVINGS}$ of each pruned meta-classifier. Almost identical results have been obtained for the other meta-classifiers as well, but are not shown here due to space considerations. The black bars represent the accuracy gain ratios $G_{P=ACCURACY}$, the dark gray bars represent the TP-FP gain ratios $G_{P=TP-FP}$ and the light gray bars represent the savings gain ratios $G_{P=SAVINGS}$ of the pruned meta-

classifier. The very light gray bars correspond to the relative throughput of the pruned meta-classifier T_P to the throughput of the complete meta-classifier T_C. To estimate the throughput of the meta-classifiers, we measured the time needed for a meta-classifier to generate a prediction. This time includes the time required to obtain the predictions of the constituents base classifiers sequentially on an unseen credit card transaction, the time required to assemble these predictions into a single meta-level "prediction" vector and the time required for the meta-classifier to input the vector and generate the final prediction. The measurements were performed on a Personal Computer with a 200MHz Pentium processor running Solaris 2.5.1. These measurements show that cost-complexity pruning is successful in finding Chase meta-classifiers that retain their performance levels to 100% of the original even with as much as 60% of the base classifiers pruned or within 60% of the original with 90% pruning. At the same time, the pruned classifiers exhibit 230% and 638% higher throughput. For the First Union base classifiers, the results are even better. With 80% pruning, the pruned meta-classifiers have gain ratios $G \simeq 1$ and with 90% pruning they are within 80% of the original performance. The throughput improvement in this case is 5.08 and 9.92 times better, respectively.

5. Concluding remarks

Constructing ensembles of classifiers computed over many data sets is a scalable and effective solution to learning over large and possibly distributed databases. In this paper, we addressed a shortcoming of this approach that has largely been ignored, the increased demand for run-time system resources. The final ensemble meta-classifier may consist of a large collection of base classifiers that require significantly more memory resources, while substantially slowing down classification throughput. We described a pruning algorithm, called post-training pruning, that seeks to determine the base classifiers that are redundant or "less-important" to the classification process of the ensemble meta-classifier. Thus, given a set of system constraints and requirements, the algorithm computes a smaller and faster ensemble meta-classifier with predictive performance that is comparable to the original meta-classifier.

We evaluated the post-training pruning algorithm and compared it against two other pruning algorithms using real credit card transaction data sets provided by two separate institutions. Our results demonstrated that our algorithm is successful and highly robust in computing pruned meta-classifiers that preserve the predictive performance of the original unpruned meta-classifiers even when system constraints call for heavy pruning. This line of work, however, is far from being complete; future research plans include experiments on additional data sets as well as studying of alternative algorithms and methods for exploring (analytically, empirically and graphically) the relationships among base classifiers and meta-classifiers. To our knowledge, there have been few attempts to incorporate resource requirements in the process of forming ensemble classifiers, even though such constraints are common place for operational systems.

Acknowledgements. We are in debt to Chris Merz for providing us with his implementation of the SCANN algorithm. This research was supported by the Intrusion Detection Program (BAA9603) from DARPA (F30602-96-1-0311), NSF (IRI-96-32225 and CDA-96-25374) and NYSSTF (423115-445).

References

Ali, K. and Pazzani, M. (1996), 'Error reduction through learning multiple descriptions', *Machine Learning* **24**, 173–202.

Breiman, L. (1994), Heuristics of instability in model selection, Technical report, Department of Statistics, University of California at Berkeley.

Breiman, L. (1996), 'Stacked regressions', *Machine Learning* **24**, 41–48.

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

Chan, P. and Stolfo, S. (1993), Toward parallel and distributed learning by meta-learning, *in* 'Working Notes AAAI Work. Knowledge Discovery in Databases', pp. 227–240.

Chan, P., Stolfo, S. and Wolpert, D., eds (1996), *Working Notes for the AAAI-96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms*, Portland, OR.

Cohen, W. (1995), Fast effective rule induction, *in* 'Proc. 12th Intl. Conf. Machine Learning', Morgan Kaufmann, pp. 115–123.

Craven, M. and Shavlik, J. J. W. (1996), 'Extracting tree-structured representations of trained networks', *Advances in Neural Information Processing Systems* **8**.

Dietterich, T. (1997), 'Machine learning research: Four current directions', *AI Magazine* **18**(4), 97–136.

Domingos, P. (1997), Knowledge acquisition from examples via multiple models, *in* 'Proc. Fourteenth Intl. Conf. Machine Learning', pp. 98–106.

Fan, W. (2000), On the effective use of stacking, PhD thesis, Department of Computer Science, Columbia University, New York, NY.

Fan, W., Lee, W., Stolfo, S. and Miller, M. (2000), A multiple model cost-sensitive approach for intrusion detection, *in* 'Proc. Eleventh European Conference of Machine Learning', Barcelona Spain, pp. 148–156.

Freund, Y. and Schapire, R. (1995), A decision-theoretic generalization of on-line learning and an application to boosting, *in* 'Proceedings of the Second European Conference on Computational Learning Theory', Springer-Verlag, pp. 23–37.

Freund, Y. and Schapire, R. (1996), Experiments with a new boosting algorithm, *in* 'Proc. Thirteenth Conf. Machine Learning', pp. 148–156.

Greenacre, M. J. (1984), *Theory and Application of Correspondence Analysis*, Academic Press, London.

Guo, Y. and Sutiwaraphun, J. (1998), Knowledge probing in distributed data mining, *in* P. C. H. Kargupta, ed., 'Work. Notes KDD-98 Workshop on Distributed Data Mining', AAAI Press, pp. 61–69.

Jacobs, R., Jordan, M., Nowlan, S. J. and Hinton, G. E. (1991), 'Adaptive mixture of local experts', *Neural Computation* **3(1)**, 79–87.

Jordan, M. I. and Jacobs, R. A. (1994), 'Hierarchical mixtures of experts and the EM algorithm', *Neural Computation* **6**, 181–214.

Jordan, M. I. and Xu, L. (1993), Convergence results for the em approach to mixtures of experts architectures., *in* 'AI memo 1458'.

Kong, E. B. and Dietterich, T. (1995), Error-correcting output coding corrects bias and variance, *in* 'Proc. Twelfth Intl. Conf. Machine Learning', pp. 313–321.

Krogh, A. and Vedelsby, J. (1995), Neural network ensembles, cross validation, and active learning, *in* G. Tesauro, D. Touretzky and T. Leen, eds, 'Advances in Neural Info. Proc. Sys. 7', MIT Press, pp. 231–238.

LeBlanc, M. and Tibshirani, R. (1993), Combining estimates in regression and classification, Technical Report 9318, Department of Statistics, University of Toronto, Toronto, ON.

Margineantu, D. and Dietterich, T. (1997), Pruning adaptive boosting, *in* 'Proc. Fourteenth Intl. Conf. Machine Learning', pp. 211–218.

Merz, C. (1999), 'Using correspondence analysis to combine classifiers', *Machine Learning* **36**, 33–58.

Merz, C. and Pazzani, M. (1999), 'A principal components approach to combining regression estimates', *Machine Learning* **36**, 9–32.

Minksy, M. and Papert, S. (1969), *Perceptrons: An Introduction to Computation Geometry*, MIT Press, Cambridge, MA. (Expanded edition, 1988).

Mitchell, T. (1982), 'Generalization as search', *Artificial Intelligence* **18**, 203–226.

Opitz, D. W. and Shavlik, J. J. W. (1996), 'Generating accurate and diverse members of a neural-network ensemble', *Advances in Neural Information Processing Systems* **8**, 535–541.

Ortega, J., Koppel, M. and Argamon-Engelson, S. (1999), 'Arbitrating among competing classifiers using learned referees', *Machine Learning* . in press.

Perrone, M. P. and Cooper, L. N. (1993), 'When networks disagree: Ensemble methods for hybrid neural networks.', *Artificial Neural Networks for Speech and Vision* pp. 126–142.

Prodromidis, A. (1999), Management of Intelligent Learning Agents in Distributed Data Mining Systems, PhD thesis, Department of Computer Science, Columbia University, New York, NY.

Prodromidis, A. L. and Stolfo, S. J. (1998), Pruning meta-classifiers in a distributed data mining system, *in* 'Proc of the First National Conference on New Information Technologies', Athens, Greece, pp. 151–160. Extended version.

Prodromidis, A. L. and Stolfo, S. J. (1999*a*), Agent-based distributed learning applied to fraud detection. CUCS-014-99.

Prodromidis, A. L., Stolfo, S. J. and Chan, P. K. (1999), Effective and efficient pruning of meta-classifiers in a distributed data mining system, Technical report, Columbia Univ. CUCS-017-99.

Prodromidis, A. and Stolfo, S. (1999*b*), A comparative evaluation of meta-learning strategies over large and distributed data sets, *in* 'Workshop on Meta-learning, Sixteenth Intl. Conf. Machine Learning', Bled, Slovenia, pp. 18–27.

Provost, F. and Fawcett, T. (1997), Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions, *in* 'Proc. Third Intl. Conf. Knowledge Discovery and Data Mining', pp. 43–48.

Provost, F. and Fawcett, T. (1998), Robust classification systems for imprecise environments, *in* 'Proc. AAAI-98', AAAI Press.

Quinlan, J. R. (1986), 'Induction of decision trees', *Machine Learning* **1**, 81–106.

Quinlan, J. R. (1993), *C4.5: programs for machine learning*, Morgan Kaufmann, San Mateo, CA.

Quinlan, R. and Rivest, R. (1989), 'Inferring decision trees using the minimum description length princliple', *Information and Computation* **80**, 227–248.

R., W. S. and J., R. A. (1994), Classification using hierarchical mixtures of experts., *in* 'IEEE Workshop on Neural Networks for Signal Processing IV', pp. 177–186.

Schapire, R. (1990), 'The strength of weak learnability', *Machine Learning* **5**, 197–226.

Stolfo, S., Prodromidis, A., Tselepis, S., Lee, W., Fan, W. and Chan, P. (1997), JAM: Java agents for meta-learning over distributed databases, *in* 'Proc. 3rd Intl. Conf. Knowledge Discovery and Data Mining', pp. 74–81.

Tresp, V. and Taniguchi, M. (1995), 'Combining estimators using non-constant weighting functions', *Advances in Neural Information Processing Systems* **7**, 419–426.

Wolpert, D. (1992), 'Stacked generalization', *Neural Networks* **5**, 241–259.

Author Biographies

Andreas L. Prodromidis is the Director of Research and Development of iPrivacy LLC. He earned a Diploma in Electical Engineering at the National Technical University of Athens in 1993 and his Ph.D. in Computer Science at Columbia University in 1999. His primary research focus has been in distributed and scalable systems, data mining and machine learning. He has developed JAM a distributed data mining system for managing intelligent learning agents across multiple database sites. Dr. Prodromidis spent two years at the IBM T.J. Watson Research Center where he focused on resource management and distributed systems. More recently, he consulted as a senior scientist to Activium Inc. on issues of network security and intrusion detection.

Salvatore J. Stolfo is Professor of Computer Science at Columbia University. He received his Ph.D. from NYU Courant Institute in 1979 and has been on the faculty of Columbia ever since. He has published extensively in the areas of parallel computing, AI Knowledge-based systems, and most recently Data Mining. Dr. Stolfo co-developed the first Expert Database System in the early 1980's that was widely distributed to a large number of telephone wire centers around the nation. He has lead a project that developed the 1023-processor DADO parallel computer designed to accelerate knowledge-based and pattern directed inference systems. His most recent research has been devoted to distributed data mining systems with applications to fraud and intrusion detection in network information systems. Browse the URL http://www.cs.columbia.edu/ids for complete details. He is a member of two editorial boards and has recently co-chaired several workshops in the area of data mining, intrusion detection and the Digital Government and co-chaired the PC of the SIGKDD 2000 Conference. He presently co-directs the Digital Government Research Center, a joint research center between Columbia University and USC/ISI. He served as the Chairman of Computer Science and the Director of the Center for Advanced Technology at Columbia University. He was also an expert witness in the DOJ versus Microsoft "browser wars" case. Presently he is a member the Congressional Internet Caucus Advisory Committee, and Visa 3D Secure Authenticated Internet Payments Vendor Program as the Chief Scientific Advisor to iPrivacy LLC, a recently established start-up he co-founded. He has been awarded ten patents in the areas of parallel computing and database inference, and six patents are pending in the area of internet privacy.

Correspondence and offprint requests to: Andreas L. Prodromidis, iPrivacy LLC, 599 Lexington Ave., Suite 2300, New York, NY 10027, USA. Email: andreas@iprivacy.com