Pruning Classifiers in a Distributed Meta-Learning System*

Andreas L. Prodromidis †
Salvatore Stolfo
Department of Computer Science
Columbia University
1214 Amsterdam Ave. Mail Code 0401
New York, NY 10027
{andreas,sal}@cs.columbia.edu

Philip K. Chan
Computer Science
Florida Institute of Technology
Melbourne, FL 32901
pkc@cs.fit.edu

March 16, 1998

Abstract

JAM is a powerful and portable agent-based distributed data mining system that employs meta-learning techniques to integrate a number of independent classifiers (concepts) derived in parallel from independent and (possibly) inherently distributed databases. Although meta-learning promotes scalability and accuracy in a simple and straightforward manner, brute force meta-learning techniques can result in large, inefficient and some times inaccurate meta-classifier hierarchies. In this paper we explore several techniques for evaluating classifiers and we demonstrate that meta-learning combined with certain pruning methods can achieve similar or even better performance results in a much more cost effective manner.

Keywords: classifier evaluation, pruning, metrics, distributed mining, meta-learning.

*This research is 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).

†Supported in part by IBM

1 Introduction

Meta-learning [6] is a technique that addresses the scaling problem of machine learning, i.e. the problem of learning useful information from large and inherently distributed databases. The idea is to execute a number of concept learning processes on a number of data subsets in parallel, and then to combine their collective results through another phase of learning. Initially, each concept learning task, also called *base learner*, computes a concept or *base classifier* that models its underlying data subset or *training set*. Next, a separate concept learning task, called *meta learner*, combines these independently computed base classifiers into a higher level concept or classifier, called *meta classifier*, by learning from a *meta-level training set*. This meta-level training set is basically composed from the predictions of the individual base-classifiers when tested against a separate subset of the training data, also called *validation set*. From their predictions, the meta-learner detects the properties, the behavior and performance of the base-classifiers and computes a meta-classifier that models the "global" data set.

Meta-learning improves *efficiency* by executing *in parallel* the base-learning processes (implemented as distinct serial programs) on smaller (and possibly disjoint) subsets of the training data set (*a data reduction technique*). Furthermore, it improves *accuracy* by combining different learning systems each having different *inductive bias* (e.g representation, search heuristics, search space) [16]. By combining separately learned concepts, meta-learning is expected to derive a higher level learned model that explains a large database more accurately than any of the individual learners.

The JAM system [22] is designed to implement meta-learning, hence it can take full advantage of its inherent parallelism and distributed nature. Briefly, JAM is a distributed agent-based data mining system that provides a set of learning agents that compute models (concepts) over data stored locally at a site and a set of *meta-learning* agents for combining multiple models that were learned (perhaps) at different sites. The system employs a special distribution mechanism which allows the migration of the derived models or *classifier agents* to other remote sites.

Employing efficient distributed protocols at the system architecture level, however, addresses the scalability problem only partially. The analysis of the characteristics and dependencies of the classifiers and the management of the agents within the data sites constitutes the other half of the scalability problem. If not controlled properly, the size and arrangement of the meta-classifiers inside each data site may incur unnecessary and prohibitive overheads.

In the context of credit card fraud detection, for example, as additional classifiers are combined in a meta-classifier, the more time and resources are expected to classify incoming unknown transactions increasing overheads and lowering throughputs. Figure 1 plots the required time per transaction and throughput of a detection system as a function of the number of base classifiers integrated in five different meta-classifiers.

Memory constraints are equally important. For the same problem, a single ID3 decision tree ([19]) may require more than 650KBytes of main memory, while a C4.5 decision tree ([20]) may need 75KBytes. Retaining a large number of base classifiers and meta-classifiers may not be feasible.

Meta classifiers are defined recursively as collections of classifiers structured in multi-level trees [7] and determining the optimal set of classifiers is a combinatorial problem. *Pre-training pruning*[1] refers to the filtering of the classifiers before they are used in the training of a meta-classifier. Instead of treating classifiers as

[1] As opposed to *post-training pruning* [17] which denotes the evaluation and revision/pruning of the meta-classifier after it is computed.

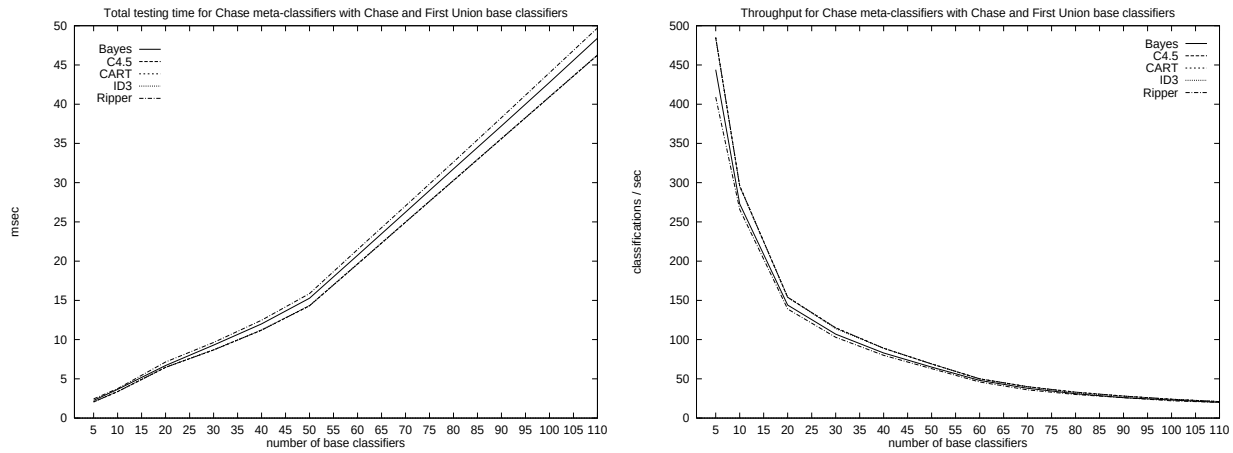

Figure 1: Left: Average Testing time of a credit card transaction. Right: Throughput of credit card transactions.

sealed entities combined in a brute force manner, with pre-training pruning we introduce a pre-meta-learning stage for analyzing the available classifiers and qualifying them for inclusion in a meta-classifier. Only those classifiers that appear (according to one or more predefined metrics) to be most "promising" will participate. The goal of pre-training pruning (referred to as pruning, for the rest of our discussion) is to build partially grown meta-classifiers (meta-classifiers with pruned subtrees) that are more efficient and scalable and at the same time achieve comparable or better performance (accuracy) results than fully grown meta-classifiers.

To take advantage of the benefits of pruning we need to address the following questions:

- Given a classifier, which properties can provide valuable information for selecting that classifier for inclusion in a meta-learning process?
- How do we compare classifiers?
- What qualifies a combination of classifiers as "good"?

In this paper we examine several different evaluation metrics and pruning algorithms, and we compare their performance in the credit card fraud detection domain. By way of summary, we find that pruned meta-classifiers can sustain accuracy levels, increase the $TP - FP$ spread and reduce losses due to fraud (with respect to a cost model fitted to the problem) at a substantially higher throughput, compared to the non-pruned meta-classifiers. The remainder of this paper is organized as follows. Section 2 reviews various classifier evaluation metrics that have been examined in the past and introduces the evaluation metrics and general methods used in this study. Section 3 describes the pruning algorithms while section 4 presents the experiments performed and the results collected. The paper concludes with section 5.

2 Evaluating and Selecting Classifiers

To analyze, compare and manage ensembles of classifiers, one can employ several different measures and methods. Before we present the metrics employed in this study, we summarize the previous and current research within the Machine Learning and KDD communities.

Provost and Fawcett [18] introduced the ROC convex hull method for its intuitiveness and flexibility. The method evaluates models for binary classification problems, by mapping them onto a *True Positive/False Negative* plane and by allowing comparisons under different metrics (true positive/false negative rates, accuracy,

cost, etc.). The focus of this paper, however, is on evaluation methods that are suitable for multiple class problems and on metrics that provide information about the interdependencies among the base classifiers and their potential when forming ensembles of classifiers [9, 13].

Margineantu and Dietterich [15] acknowledged the importance of pruning ensembles of classifiers and studied the problem of evaluating and pruning the set of hypothesis (classifiers) obtained by the boosting algorithm ADABOOST [11]. According to their findings, by examining the diversity and accuracy of the available classifiers, it is possible for a subset of classifiers to achieve similar levels of performance as the entire set. In this paper, we are considering the same problem, but with two additional dimensions; we study the more general setting where ensembles of classifiers can be obtained by training (possibly) different learning algorithms over (possibly) distinct databases. Furthermore, instead of voting (ADABOOST) over the predictions of classifiers for the final classification, we adopt meta-learning to combine predictions of the individual classifiers.

In this work, we focus on the *diversity* and *specialty* metrics. Apart from *accuracy* and these metrics, *correlation error* and *coverage* have also been used to analyze and explain the properties and performance of classifiers. Ali and Pazzani [1] define as correlation error the fraction of instances for which a pair of base classifiers make the same incorrect predictions and Brodley and Lane [3] measured coverage by computing the fraction of instances for which at least one of the base classifiers produces the correct prediction.

2.1 Diversity

Brodley [4] defines diversity by measuring the *classification overlap* of a pair of classifiers, i.e. the percentage of the instances classified the same way by two classifiers while Chan [5] associates it with the entropy in the predictions of the base classifiers. (When the predictions of the classifiers are distributed evenly across the possible classes, the entropy is higher and the set of classifiers more diverse.) Kwok and Carter [14] correlate the error rates of a set of decision trees to their syntactical diversity, while Ali and Pazzani [1] studied the impact of the number of gain ties [2] on the accuracy of an ensemble of classifiers.

Here, we approximate the diversity of a set of classifiers $\mathcal{S}$ by calculating the average diversity of all possible pairs of classifiers:

$$\mathcal{D} = \frac{\sum_{i=1}^{|\mathcal{S}|-1} \sum_{j=i+1}^{|\mathcal{S}|} \sum_{k=1}^{n} Dif\ (C_i(y_k),\ C_j(y_k))}{\frac{(|\mathcal{S}|-1)\cdot|\mathcal{S}|}{2} \cdot n}$$

where $C_j(y_i)$ denotes the classification of the y_i instance by the C_j classifier and $Dif(a, b)$ returns zero if a and b are equal, and one if they are different. The more diverse the set of classifiers is, the more room a meta classifier will have to improve performance.

2.2 Class specialty

The term *class specialty* defines a family of evaluation metrics that concentrate on the "bias" of a classifier towards certain classes. A classifier specializing in one class, should exhibit, for that class, both, a high True Positive (TP) and a low False Positive (FP) rate. The TP rate is a measure of how "often" the classifier predicts the class correctly, while FP is a measure of how often the classifier predicts the wrong class.

For concreteness, given a classifier C_j and a data set containing n examples, we construct a two dimensional contingency table T^j where each cell T^j_{kl} contains the number of examples x for which the true label $L(x) = c_k$ and $C_j(x) =$

[2] The information gain of an attribute captures the "ability" of that attribute to classify an arbitrary instance. The information gain measure favors the attribute whose addition as the next split-node in a decision tree (or as the next clause to the clause body of a rule) would result in a tree (rule) that would separate into the different classes as many examples as possible.

c_l. According to this definition, cell T^j_{kk} contains the number of examples classifier C_j classifies correctly as c_k. If the classifier C_j is capable of 100% accuracy on the given data set, then all non-zero counts appear along the diagonal. The sum of all the cells in T^j is n. Then, True Positive and False Positive rates are defined as:

$$TP(C_j, c_k) = \frac{T^j_{kk}}{\sum_{i=1}^{c} T^j_{ki}}$$

$$FP(C_j, c_k) = \frac{\sum_{i \neq k} T^j_{ik}}{\sum_{i \neq k} \sum_{l=1}^{c} T^j_{il}}$$

In essence, $FP(C_j, c_k)$ calculates the number of c_k examples that classifier C_j misclassified versus the total number of examples that belong in different classes (c is the number of classes). The *class specialty* metric quantifies the bias of a classifier towards a certain class. In particular, a classifier C_j is highly biased/specialized for class c_k when its $TP(C_j, c_k)$ is high and its $FP(C_j, c_k)$ is low.

One-sided class specialty metric Given the definitions of the TP and FP rates, this is the simplest and most straight forward metric of the class specialty family. The one-sided class specialty metric evaluates a classifier by inspecting the TP rate or the FP rate (but not both) of the classifier on a given class over the validation data set. A simple pruning algorithm would integrate in a meta-classifier the (base-) classifiers that exhibit high TP rates with the classifiers that exhibit low FP rates.

Two-sided class specialty metrics The problem with the one-sided class specialty metric is that it may qualify poor classifiers. Lets take for example the extreme case of a classifier that always predicts the class c_k. This classifier is highly biased and the algorithm will select it. So, we define two new metrics, the *positive combined specialty* $PCS(C_j, c_k)$ and the *negative combined specialty* $NCS(C_j, c_k)$ metrics, that take into account both the TP and FP rates of a classifier for a particular class. The former is biased towards TP rates, while the later is biased towards FP rates:

$$PCS(C_j, c_k) = \frac{TP(C_j, c_k) - FP(C_j, c_k)}{1 - TP(C_j, c_k)}$$

$$NCS(C_j, c_k) = \frac{TP(C_j, c_k) - FP(C_j, c_k)}{FP(C_j, c_k)}$$

Combined class specialty metric A third alternative is to define a metric that combines the TP and FP rates of a classifier for a particular class into a single formula. Such a metric has the advantage of distinguishing the single best classifier for each class with respect to some predefined criteria. The *combined class specialty* metric, or $CCS(C_j, c_k)$, is defined as:

$$CCS(C_j, c_k) = f_{TP}(c_k) \cdot TP(C_j, c_k) + f_{FP}(c_k) \cdot FP(C_j, c_k)$$

where $-1 \leq f_{TP}(c_k), f_{FP}(c_k) \leq 1, \forall c_k$. Coefficient functions f_{TP} and f_{FP} are single variable functions quantifying the importance of each class according to the needs of the problem and the distribution of each class in the entire data set.[3]

In many real world problems, e.g. medical data diagnosis, credit card fraud, etc, the plain $TP(C_j, c_k)$ and $FP(C_j, c_k)$ rates fail to capture the entire story. The distribution of the classes in the data set may not be balanced and maximizing the TP rate of one class may be more important than maximizing total accuracy. In the credit card fraud detection problem, for instance, catching expensive fraudulent transactions is more vital than eliminating the possibility for false alarms. The *combined class specialty metric* provides the means to associate a cost model with the performance of each classifier and evaluate the classifiers from an aggregate cost perspective.

[3] A more general and elaborate specialty metric may take into account the individual instances as well:

$$CCS(C_j, c_k, y_i) = f_{TP}(c_k, y_i) \cdot TP(C_j, c_k, y_i) + f_{FP}(c_k, y_i) \cdot FP(C_j, c_k, y_i)$$

In all versions, the pruning algorithm evaluates all available (base-) classifiers and then selects the ones with the highest specialty per class. Recall that high specialty for a class means high accuracy for that class, so, in essence, the algorithm chooses the (base-) classifiers with the most specialized and accurate view of each class. We expect that a meta-classifier trained on these (base-) classifiers will be able to uncover and learn their bias and take advantage of their properties.

Combining metrics Instead of relying just on one criterion to choose the (base-) classifiers the pruning algorithms can employ several metrics simultaneously. Different metrics capture different properties and qualify different classifiers as "best". By combining the various "best" classifiers into a meta-classifier we can presumably form meta-classifiers of higher accuracy and efficiency, without searching exhaustively the entire space of the possible meta-classifiers. For example, one possible approach would be to combine the (base-) classifiers with high *coverage* and low *correlation error*. A different strategy that combines CCS *rates* and *coverage* would be to iteratively select classifiers based on their CCS score on examples which the preceding classifiers failed to cover. In another study [21] concerning credit card fraud detection we employ evaluation formulas for selecting classifiers that are based on characteristics such as *diversity*, *coverage* and *correlated error* or their combinations, i.e. *True Positive rate* and *diversity*.

3 Pruning algorithms

Pruning refers to the evaluation and selection of classifiers before they are used for the training of the meta-classifier. A pruning algorithm is provided with a set of pre-computed classifiers $\mathcal{H}$ (obtained from one or more databases by one or more machine learning algorithms) and a validation set $\mathcal{V}$ (a separate subset of data, different from the training and test sets). The result is a set of classifiers $\mathcal{C} \subseteq \mathcal{H}$ to be combined in a higher level meta-classifier. To reduce the complexity of determining the optimal meta-classifier, we employed the diversity and the coverage/specialty metrics to guide the greedy search. We implemented, tested and compared a diversity-based pruning algorithm and three particular instances of a coverage/specialty-based pruning algorithm, described in Table 3.

Diversity-Based Pruning Algorithm The diversity-based algorithm works iteratively selecting one classifier each time starting with the most accurate base classifier. Initially it computes the diversity matrix d where each cell d_{ij} contains the number of instances of the validation set for which classifiers C_i and C_j give different predictions. In each round, the algorithm adds to the list of selected classifiers $\mathcal{C}$ the classifier C_k that is most diverse to the classifiers chosen so far, i.e. the C_k that maximizes $\mathcal{D}$ over $\mathcal{C} \cup C_k, \forall k$ in $1, 2, \ldots |\mathcal{H}|$. The selection process ends when the N most diverse classifiers are selected. N is a parameter that depends on factors such as minimum system throughput, memory constraints or diversity thresholds:[4] The algorithm is independent of the number of attributes of the data set and is bounded by an $O(n \cdot |\mathcal{H}|^2)$ (where n denotes the number of examples) complexity due to the computation of the diversity matrix. For all practical problems, however, $|\mathcal{H}|$ is much smaller than n and the overheads are minor.

Coverage/Specialty-Based Pruning Algorithm This algorithm combines the *coverage* metric and one of the instances of the *specialty metric* covered in section 2. Initially, the algorithm starts by choosing the base classifier with the best performance with respect to the

[4]The diversity $\mathcal{D}$ of a set of classifiers $\mathcal{C}$ decreases as the size of the set increases. By introducing a threshold, one can avoid including redundant classifiers in the final outcome.

Let $\mathcal{C} := \emptyset$, N := maximum number of classifiers
For i := 1, 2, ..., $|\mathcal{H}|$ - 1 **do**
  For j := i, i+1, ..., $|\mathcal{H}|$ **do**
    Let d_{ij} := the number of instances where C_i and C_j give different predictions
Let C' := the classifier with the highest accuracy
$\mathcal{C} := \mathcal{C} \cup C'$, $\mathcal{H} := \mathcal{H} - C'$
For i := 1, 2, ..., N **do**
  For j := 1, 2, ..., $|\mathcal{H}|$ **do**
    Let $D_j := \sum_{k=1}^{|\mathcal{C}|} d_{jk}$
  Let C' := the classifier from $\mathcal{H}$ with the highest D_j
  $\mathcal{C} := \mathcal{C} \cup C'$, $\mathcal{H} := \mathcal{H} - C'$

Let $\mathcal{C} := \emptyset$
For all target classes c_k, k = 1, 2, ..., c, **do**
  Let $\mathcal{E} := \mathcal{V}$
  Let $\mathcal{C}_k := \emptyset$, $\mathcal{H}_k := \emptyset$
  Until no other examples in $\mathcal{E}$ can be covered or $\mathcal{H}_k = \emptyset$
    Let C' := the classifier with the best *class specialty* on target class c_k for $\mathcal{E}$
    $\mathcal{C}_k := \mathcal{C}_k \cup C'$, $\mathcal{H}_k := \mathcal{H}_k - C'$
    $\mathcal{E}$:= $\mathcal{E}$ - examples covered by C'
  $\mathcal{C} := \mathcal{C} \cup \mathcal{C}_k$

Table 1: The Diversity-based (left) and Coverage/Specialty-based (right) pruning algorithms

specialty metric for a particular target class on the validation set. Then it continues by iteratively selecting classifiers based on their performance on the examples the previously chosen classifiers failed to cover. The cycle ends when there are no other examples to cover and it repeats the selection process for a different target class. The complexity of the algorithm is bounded by $O(n \cdot c \cdot |\mathcal{H}|)$. For each target class (c is the total number) it considers at most $|\mathcal{H}|$) classifiers and each time it compares each classifier with all remaining classifiers (bounded by $|\mathcal{H}|$)) on all misclassified examples (bounded by n).

Even though the algorithm performs a greedy search, it combines classifiers that are diverse (they classify correctly different subsets of data), accurate (with the best performance on the data set used for evaluation with respect to the class specialty) and with high coverage. The three instances of the pruning algorithm combine:

1. coverage and $PCS(C_j, c_k)$
2. coverage and $TP(C_j, c_k)$ - $FP(C_j, c_k)$
3. coverage and a cost model tailored to the credit card fraud detection problem.

4 Experiments and Results

Learning algorithms Five inductive learning algorithms are used in our experiments. ID3, its successor C4.5, and Cart [2] are decision tree based algorithms, Bayes, described in [10], is a naive Bayesian classifier and Ripper [8] is a rule induction algorithm.

Learning tasks Two data sets of real credit card transactions were used in our experiments provided by the Chase and First Union Banks, members of the FSTC (Financial Services Technology Consortium).

These two data sets contain credit card transactions labeled as fraudulent or legitimate. Each bank supplied .5 million records spanning one year. Chase bank data consisted, on average, of 42,000 sampled credit card transaction records per month with 20% fraud versus 80% non-fraud distribution, whereas First Union data were sampled in a non-uniform manner (many records from some months, very few from others) with 15% fraud versus 85% non-fraud.

To evaluate and compare the meta-classifiers constructed, we adopted three metrics: the overall accuracy, the $(TP - FP)$ spread and a cost model fitted to the credit card detection problem. Overall accuracy expresses the ability of a classifier to give correct predictions, $(TP-FP)$ denotes the ability of a classifier to catch fraudulent transactions while minimizing false alarms, and finally, the cost model captures the performance of a classifier with respect to the goal of the target application (stop loss due to fraud).

Credit card companies have a fixed overhead that serves as a threshold value for challenging the legitimacy of a credit card transaction. In other words, 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. With this overhead cost taken into account, we seek to produce classifiers and meta-classifiers that generate the maximum savings $S(C_i, X)$:

$$S(C_j, X) = \sum_{i=1}^{n} [F(C_j, y_i) \cdot (amt(y_i) - X) - L(C_j, y_i) \cdot X] \cdot I(y_i, X)$$

where $F(C_j, y_j)$ returns one when classifier C_j classifies correctly a fraudulent transaction y_j and $L(C_j, y_j)$ returns one when classifier C_j misclassifies a legitimate transaction y_j. $I(x_i, X)$ inspects the transaction amount *amt* of transaction x_i, and returns one if it greater than \$X and zero otherwise.

Experiments First, we prepared the set of candidate base classifiers, i.e. the original set of base classifiers the pre-training pruning algorithm is called to evaluate. We obtained these classifiers by dividing the original data sets into 12 non-overlapping subsets (by months) and by applying the 5 learning algorithms on each of these subsets. Overall we created 60 base classifiers for each data set. In our experiments we use a 6-fold cross validation approach, meaning that in each fold we use 10 subsets for training and we keep one subset for the meta-level training phase and one for the meta-level testing phase.

In each fold, there are 50 candidate base classifiers (10 subsets, 5 algorithms each), hence there are $(2^{50} - 50)$ different possible combinations of base classifiers from which the pre-training pruning algorithm had to choose one. In the meta-learning stage, we employ all five learning algorithms. The evaluation and selection of the base classifiers is based on the subset that is used as the meta-level training set. The overall performance of the pre-training pruning method is judged against the subset used as the meta-level testing set.

Results The results from these experiment are displayed in Figures 2, and 3. Figure 2 plots the overall accuracy, the $(TP - FP)$ spread and the savings (in dollars) for the Chase bank credit card data, and Figure 3 for the First Union data. Each figure contrasts the four (1 diversity-based and 3 coverage/specialty based) pruning methods and an additional classifier selection method denoted here as *arbitrary*. As the name indicates, *arbitrary* uses no intelligence to evaluate base classifiers; instead it combines them in a "random" order, i.e. when they become available. For brevity, we plotted only the best performing meta-learning algorithms, the accuracy of the Ripper meta-classifiers and the $TP - FP$ rates and savings of the Bayesian meta-classifiers.

The vertical lines in the figures denote the number of base classifiers integrated in the final meta-classifier as determined by the coverage/specialty algorithms. The final Chase meta-classifier for the coverage/TP-FP algorithm, for example, combines 33 base classifiers, while the final First Union meta-classifier consists of 26 base classifiers. In these graphs we have included the intermediate performance results (i.e. the accuracy, $TP - FP$ rates and savings of the partially built meta-classifiers) as well as the performance results of the redundant meta-classifiers would have had, had we used more base-classifiers or not introduced the pruning phase. The final size of the diversity-based pruning algorithm is determined as discussed in section 3.

The benefits from the pruning methods are clear. In all four cases the performance of

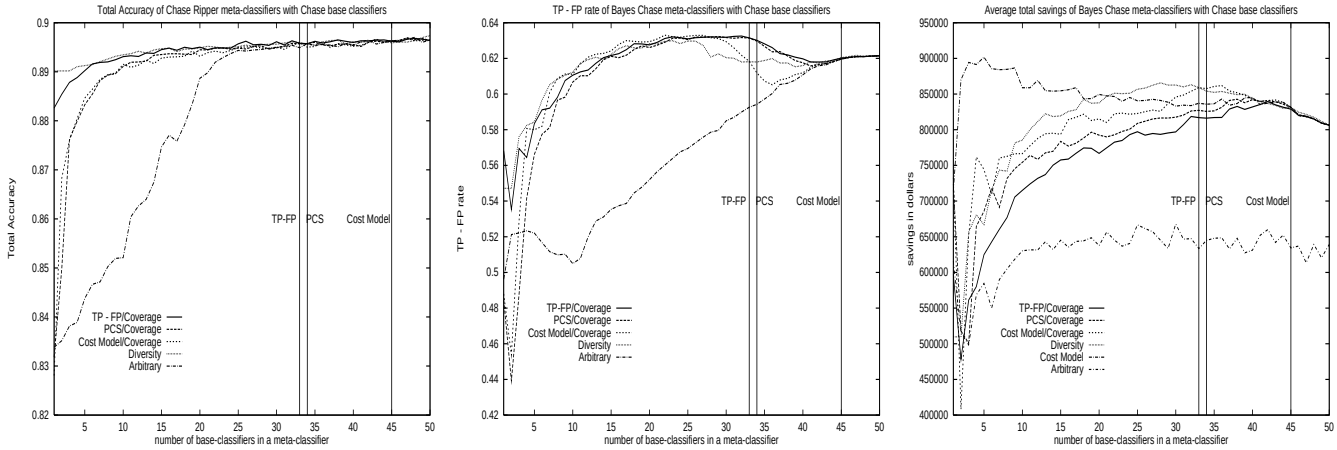

Figure 2: Pruning algorithms: Total accuracy (left), (TP-FP) spread (middle) and savings (right) on CHASE credit card data.

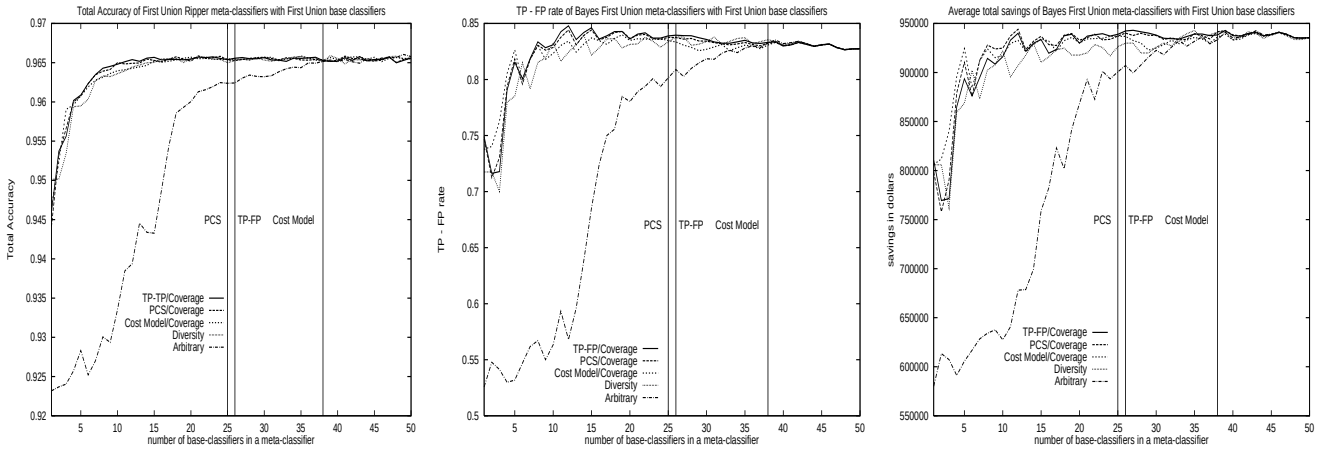

Figure 3: Pruning algorithms: Total accuracy (left), (TP-FP) spread (middle) and savings (right) on First Union credit card data.

the pruned meta-classifiers are superior to that of the complete meta-classifier[5] and certainly better than the "arbitrary" non-intelligent pruning method both with respect to effectiveness ($TP - FP$, overall accuracy, savings) as well as efficiency (less base-classifiers are retained); using more base-classifiers than selected (denoted by the vertical lines) has no positive impact on the performance of the meta-classifiers. Overall, the pruning methods composed meta-classifiers with 6.2% higher $(TP - FP)$ spread and \$180K/month additional savings over the best single classifier for the Chase data and 10% higher $(TP-FP)$ spread and \$140K/month additional savings over the best single classifier for the First Union data. These pruned meta-classifiers achieve 60% better throughput 1% higher $(TP - FP)$ spread and \$100K/month additional savings than the non-pruned meta-classifier for the Chase data and 100% better throughput 2% higher $(TP - FP)$ spread and \$10K/month additional savings for the First Union data.

Although all pruning algorithms have good performance, from the figures we can see that the diversity-based pruning algorithm performs

[5]Pruned meta-classifiers are superior to the best base-classifiers as well.

best with respect to the overall accuracy and the cost model in the Chase credit card data sets and nearly best with respect to the TP-FP rate. The differences among the four methods are more evident in the right plot of Figure 2 where we present the savings of the meta-classifiers on the Chase data. In order of performance, the methods are ranked as: diversity-based first, cost-model/coverage second, PCS/coverage-based third and TP-FP/coverage-based fourth. For the First Union data, the four pruning algorithms are comparably successful and their performance plots are less distinguishable. Indeed, a closer inspection on the classifiers composing the pruned sets (C) for the four different pruning algorithms, verified that, the sets of selected classifiers were more "similar" (there were more common members) for First Union than for Chase. As expected, all pruning methods tend to converge after a certain point. After all, as they add classifiers, their pool of selected classifiers is bound to become very similar.

Note that training classifiers to distinguish fraudulent transactions is not a direct approach to maximizing savings. In this case, the learning task is ill-defined. The base-classifiers are unaware of the adopted cost model and the actual value (in dollars) of the fraud/legitimate label. Similarly, the meta-classifiers are trained to maximize the overall accuracy not by examining the savings in dollars but by relying on the predictions of the base-classifiers. In fact, a more thorough investigation revealed that with only few exceptions, the Chase base-classifiers were inclined towards catching "cheap" fraudulent transactions and for this they exhibited low savings scores. Naturally, the meta-classifiers are trained to trust the wrong base-classifiers for the wrong reasons, i.e. they trust the most base-classifiers that are most accurate instead of the classifiers that accrue highest savings. To test this theory, we developed an additional pruning algorithm that evaluates the base classifiers with respect to the cost model and associates them with priorities according to their results. The algorithm forms meta-classifiers by selecting base-classifiers based on their priority. The performance of this algorithm is displayed in the right plot of Figure 2 under the name "cost model". The resulting meta-classifiers exhibited substantially improved performance. [6]

One way to rectify this ill-defined situation, is to tune the learning problem according to the adopted cost model. For example, we can transform the binary classification problem into a multi-class problem by multiplexing the binary class and the continuous amt attribute (properly quantized into several "bins"). The classifiers derived from the modified problem would presumably fit better to the specifications of the cost model and achieve better results.

5 Conclusion

Efficiency and scalability of a distributed data mining system, such as the JAM meta-learning system, has to be addressed at two levels, the system architecture level and the data site level. In this paper, we concentrated on the later; we delved inside the data sites to explore the types, characteristics and properties of the available classifiers and deploy only the most essential classifiers. The goal was to reduce complex, redundant and sizeable meta-learning hierarchies, while minimizing overheads. We introduced several evaluation metrics and selection algorithms and we adopted measures such as the overall accuracy, the TP-FP spread and a cost model to measure the usefulness and effectiveness of our pruning methods. The experiments suggest that base classifiers in meta-learning can achieve similar or better performance results than the brute-force assembled meta-classifiers in a much more cost effective way.

[6] The nearsightedness of the learning algorithms was not pronounced in the First Union data set: the majority of the First Union base-classifiers happened to catch the "expensive" fraudulent transactions anyway, so the pruning algorithms were able to produce meta-classifiers with the appropriate base classifiers.

References

[1] K. Ali and M. Pazzani. Error reduction through learning multiple descriptions. *Machine Learning*, 24:173–202, 1996.

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

[3] C. Brodley and T. Lane. Creating and exploiting coverage and diversity. In *Work. Notes AAAI-96 Workshop Integrating Multiple Learned Models*, pages 8–14, 1996.

[4] C.Brodley. Addressing the selective superiority problem: Automatic algorithm/model class selection. In *Proc. 10th Intl. Conf. Machine Learning*, pages 17–24. Morgan Kaufmann, 1993.

[5] P. Chan. *An Extensible Meta-Learning Approach for Scalable and Accurate Inductive Learning*. PhD thesis, Department of Computer Science, Columbia University, New York, NY, 1996.

[6] P. Chan and S. Stolfo. Meta-learning for multistrategy and parallel learning. In *Proc. Second Intl. Work. Multistrategy Learning*, pages 150–165, 1993.

[7] P. Chan and S. Stolfo. Sharing learned models among remote database partitions by local meta-learning. In *Proc. Second Intl. Conf. Knowledge Discovery and Data Mining*, pages 2–7, 1996.

[8] W. Cohen. Fast effective rule induction. In *Proc. 12th Intl. Conf. Machine Learning*, pages 115–123, 1995.

[9] T.G. Dietterich. Machine learning research: Four current directions. *AI Magazine*, 18(4):97–136, 1997.

[10] C. Elkan. Boosting and naive bayesian learning [http://www-cse.ucsd.edu/∼elkan/papers/bnb.ps]. Department of Computer Science and Engineering, Univ. of California, San Diego, CA, 1997.

[11] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proc. Thirteenth Conf. Machine Learning*, pages 148–156, 1996.

[12] J. Fürnkranz and G. Widmer. Incremental reduced error pruning. In *Proc. 11th Intl. Conf. Mach. Learning*. Morgan Kaufmann, 1994.

[13] L. Hansen and P. Salamon. Neural network ensembles. *IEEE Trans. Pattern Analysis and Mach. Itell.*, 12:993–1001, 1990.

[14] S. Kwok and C. Carter. Multiple decision trees. In *Uncertainty in Aritificial Intelligence 4*, pages 327–335, 1990.

[15] D. Margineantu and T. Dietterich. Pruning adaptive boosting. In *Proc. Fourteenth Intl. Conf. Machine Learning*, pages 211–218, 1997.

[16] T. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

[17] A. L. Prodromidis. On the management of distributed learning agents. Technical Report CUCS-032-97 (PhD Thesis proposal), Department of Computer Science, Columbia University, New York, NY, 1997.

[18] F. Provost and T. Fawcett. Anaylysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Proc. Third Intl. Conf. Knowledge Discovery and Data Mining*, pages 43–48, 1997.

[19] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[20] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[21] S. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. Chan. Credit card fraud detection using meta-learning: Issues and initial results. Working notes of AAAI Workshop on AI Approaches to Fraud Detection and Risk Management, 1997.

[22] S. Stolfo, A. Prodromidis, S. Tselepis, W. Lee, W. Fan, and P. Chan. JAM: Java agents for meta-learning over distributed databases. In *Proc. 3rd Intl. Conf. Knowledge Discovery and Data Mining*, pages 74–81, 1997.