Pruning Meta-Classifiers in a Distributed Data Mining System

Andreas L. Prodromidis * and Salvatore J. Stolfo
Department of Computer Science
Columbia University
New York, NY 10027
{andreas,sal}@cs.columbia.edu

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 (models) 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, redundant, inefficient and some times inaccurate meta-classifier hierarchies. In this paper we explore several methods for evaluating classifiers and composing meta-classifiers, we expose their limitations and we demonstrate that meta-learning combined with certain pruning methods has the potential to achieve similar or even better performance results in a much more cost effective manner.

1 Introduction

Machine learning constitutes a significant part in the overall *Knowledge Discovery and Data Mining* (KDD) process, the process of extracting useful knowledge from large databases. One means of mining useful information from large databases is to apply various machine learning algorithms to discover patterns exhibited in the data and compute descriptive representations (also called classifiers or models) that can be subsequently used for a variety of strategic and tactical purposes.

Machine learning or *Inductive learning* (or *learning from examples* [17]) aims to identify regularities in a given set of training examples with little or no knowledge about the domain from which the examples are drawn. Given a set of n training examples, i.e. $\{(\mathbf{x_1}, y_1), ..., (\mathbf{x_n}, y_n)\}$, for some unknown function $y = f(\mathbf{x})$, each interpreted as a set of attribute (feature) vectors $\mathbf{x}$ of the form $\{x_1, x_2, ..., x_m\}$ and a class label y associated with each vector, $y \in \{y^1, y^2, ..., y^k\}$, the task is to compute a *classifier* or *model* $\hat{f}$ that approximates f and correctly labels any feature vector drawn from the same source as the training set, for which the class label y is unknown.

However, one of the main challenges in KDD, is the development of inductive learning techniques that **scale up** to large and (physically) distributed data sets. Many organizations seeking added value from their data are already dealing with overwhelming amounts of information that in time will likely grow in size faster than available improvements in machine resources. Many of the current generation of learning algorithms are computationally complex and require all the data to be resident in main memory which is clearly untenable for many realistic problems and databases. Furthermore, the data may be inherently distributed and cannot be localized on any one machine for a variety of practical reasons. In such situations, it may not be possible to inspect all of the data at one processing site to compute one primary "global" classifier or model at a reasonable cost. We call the problem of learning useful new information from large and inherently distributed databases, the *scaling problem for machine learning*.

Meta-learning [7], a technique similar to *stacking* [26], was developed recently to deal with the scaling problem. The basic idea is to execute a number of machine learning processes on a number of data subsets in parallel, and

*Supported in part by an IBM fellowship
This research is supported by the Intrusion Detection Program (BAA9603) from DARPA (F30602-96-1-0311), NSF (IRI-96-32225, CDA-96-25374) and NYSSTF (423115-445).

then to combine their collective results (classifiers) through an additional phase of learning. Initially, each learning task, also called a *base learner*, computes a *base classifier*, i.e. a model of its underlying data subset or *training set*. Next, a separate learning task, called a *meta learner*, integrates these independently computed base classifiers into a higher level classifier, called a *meta classifier*, by learning over a *meta-level training set*. This meta-level training set is composed from the predictions of the individual base-classifiers when tested against a separate subset of the raw training data, also called a *validation set*. From these predictions, the meta-learner discovers the properties of the base-classifiers and computes a meta-classifier which models the "global" data set. To classify an unlabeled instance, the base-classifiers present their own predictions to the meta-classifier which then makes the final classification.

Meta-learning addresses the *scaling problem* for machine learning. It improves *efficiency* by executing *in parallel* the base-learning processes on (possibly disjoint) subsets of the training set (*a data reduction technique*). Meta-learning, is *scalable* because meta-classifiers can be similarly combined into higher level meta-classifiers in a distributed fashion and it improves *predictive performance* by combining classifiers with different *inductive bias* [19].

The JAM system (Java Agents for Meta-learning) [25] is a distributed agent-based data mining system that implements meta-learning. 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, by supporting the remote dispatch and exchange of learning and classifier agents among the participating data base sites through a special distribution mechanism, and by providing a set of meta-learning agents that combine the computed models that were learned (perhaps) at different sites.

Employing meta-learning, however, addresses the scalability problem only partially. As the number of data sites, the size of data subsets and the number of deployed learning algorithms increases, more base classifiers are made available at each site and meta-learning and meta-classifying are bound to strain the system resources. The analysis of the characteristics and dependencies of the classifier and the selection and use of the most appropriate agents within the data sites constitutes the other half of the problem. This problem is the topic of this paper. We investigate the effects of discarding certain base-classifiers on the performance of the meta-classifier. If not controlled properly, the size and arrangement of the meta-classifiers inside each data site may incur unnecessary and prohibitive overheads.

From actual 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 50%, 70% and 80% credit card transaction processing throughput drop for meta-classifiers composed of 13, 20 and 30 base-classifiers respectively. Memory constraints are equally important. For the same problem, a single ID3 decision tree ([22]) may require more than 750KBytes of main memory, while a C4.5 decision tree ([23]) may need 80KBytes. Retaining a large number of base classifiers and meta-classifiers may not be practical nor feasible.

Meta classifiers are defined recursively as collections of classifiers structured in multi-level trees [8], hence determining the optimal set of classifiers is a combinatorial problem. *Pre-training pruning* (as opposed to *post-training pruning* [20] which denotes the evaluation and revision/pruning of the meta-classifier after it is computed) refers to the filtering of the classifiers before they are used in the training of a meta-classifier. Instead of combining classifiers in a brute force manner, with pre-training pruning (hereafter referred to as pruning) we introduce a preliminary 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 pre-defined metrics) to be most "promising" participate. The objective of pruning is to build partially grown meta-classifiers (meta-classifiers with pruned subtrees) to achieve comparable or better performance (accuracy) results than fully grown meta-classifiers.

Here we examine several 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 $(TP - FP)$[1] spread, and reduce losses due to fraud (with respect to a cost model fit to the problem) at a substantially higher throughput, compared to the unpruned meta-classifiers. The remainder of this paper is organized as follows. In Section 2 we review various classifier evaluation metrics and approaches that have been examined in the past and we introduce the metrics used in this study. Sections 3 describes the pruning algorithms and Section 4 presents the experiments performed and the results collected. Finally, Section 5 concludes the paper.

2 Evaluating and Selecting Classifiers

One can employ several different measures and methods to analyze, compare and combine ensembles of classifiers. Leo Breiman [2] and LeBlanc and Tibshirani [14] acknowledge the value of using multiple predictive models to increase accuracy, but they view the problem from a different perspective. They rely on cross-validation data and analytical methods, (e.g. least squares regression), to estimate the best linear combination of the available hypotheses (models). Instead, our methods employ meta-learning to combine the available models, i.e. we apply arbitrary learning algorithms to discover the correlations among the available models. Meta-learning has the advantage of employing an arbitrary learning algorithm for computing non-linear relations among the classifiers (at the expense, perhaps, of generating less intuitive representations).

In a related study, Merz's $SCANN$ algorithm [16] employs correspondence analysis (similar to Principal Component Analysis) to map the predictions of the available classifiers onto a new scaled space that clusters similar prediction behaviors and then uses the nearest neighbor algorithm to meta-learn the transformed predictions of the individual classifiers. $SCANN$ is a sophisticated end effective combining algorithm, but too expensive, computationally, to combine classifiers and generate predictions within domains with many classifiers and large data sets. Furthermore, the pruning methods presented in this paper preceed the meta-learning phase and, as such, can be used in conjunction with $SCANN$ or any other algorithm if computational constraints pose no problem.

Provost and Fawcett [21] 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 Positive* plane and by allowing comparisons under different metrics (TP/FP rates, accuracy, cost, etc.). The intent of these algorithms is to discover and prune away the sub-optimal models, while in this work, we focus on evaluation metrics that provide information about the interdependencies among the (base) classifiers and their potential when forming ensembles of classifiers [10, 12]. In fact, the performance of sub-optimal yet diverse models can be substantially improved when combined together and even surpass that of the best single models.

In other related work, Margineantu and Dietterich [15] studied the problem of pruning the ensemble of classifiers (i.e. 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. Their research however, was restricted in computing all classifiers by applying the *same* learning algorithm on many different subsets of the *same* training set. In this paper, we consider the more general setting where ensembles of classifiers can be obtained by applying, 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.

Next, we focus on the *diversity* and *specialty* metrics. *Accuracy*, *correlation error* and *coverage* are other metrics metrics explored in the literature. For example, Ali and Pazzani [1] define correlation error as the fraction of instances for which a pair of classifiers make the same incorrect predictions and Brodley and Lane [4] measure coverage by computing the fraction of instances for which at least one of the classifiers produces the correct prediction.

[1] TP stands for True Positive, i.e. percentage of actual fraud that is caught, FP stands for False Positive, i.e. percentage of false alarms

2.1 Diversity

Brodley [5] 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 [6] 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 [13] 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 measure the diversity $\mathcal{D}$ within a set of classifiers $\mathcal{H}$ by calculating the average diversity of all possible pairs of classifiers in that set $\mathcal{H}$:

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

where $C_j(y_i)$ denotes the prediction of the y_i instance by the C_j classifier and $Dif(a, b)$ returns one if a and b are different and zero if they are equal. Intuitively, the more diverse the classifiers are, 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 $\mathbf{x}$ for which the true label $L(\mathbf{x}) = c_k$ and $C_j(\mathbf{x}) = c_l$. 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:

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

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. With the class specialty defined, we can evaluate the candidate (base-) classifiers and combine those with the highest "bias" per class, or in other words, those with the most specialized and accurate view of each class, in the hope that a meta-classifier trained on these "expert" (base-) classifiers will be able to uncover and learn their bias and take advantage of their properties.

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. A simple pruning algorithm can use this metric to evaluate and integrate 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. Consider, for example, the extreme case of a classifier always predicting the class c_k. This classifier is highly biased and the algorithm will select it. So, we define two new metrics that are more balanced, the *positive combined specialty* $PCS(C_j, c_k)$ and the *negative combined specialty* $NCS(C_j, c_k)$ metrics, that take into account

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

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:

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

With these metrics defined, we integrate in a meta-classifier the (base-) classifiers that exhibit high PCS rates with the classifiers that exhibit low NCS rates.

Combined class specialty metric A third alternative is to combine 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] Note that, the *accuracy* of a classifier C_j can be computed by the $\sum_{k=1}^{c} CCS(C_j, c_k)$, with each $f_{TP}(c_k)$ set to the distribution of the class c_k in the testing set and each $f_{FP}(c_k)$ set to zero.

In many real world problems, medical diagnosis, fraud detection, 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 uniform 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 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.

2.3 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*. In another study [24] concerning credit card fraud detection we employed evaluation formulas for selecting classifiers that are based on *diversity*, *coverage* and *correlated error* and their combinations.

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, disjoint from the training and test sets). The result is a set of classifiers $\mathcal{C} \subseteq \mathcal{H}$ to be combined in a meta-classifier. Determining the optimal meta-classifier is a combinatorial problem, so we employ the accuracy, diversity, coverage and specialty metrics to guide a greedy search. More specifically, we implemented a diversity-based pruning algorithm and three instances of the combined coverage/specialty-based pruning algorithm, described in Table 1.

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}

[3] A more general and elaborate specialty metric may aggregate on the individual instances as well:

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

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 ratio of the 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 $D_j := 2 \cdot (\sum_{k=1}^{|\mathcal{C}|} d_{jk})/|\mathcal{C}| \cdot (|\mathcal{C}| + 1)$
  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

contains the ration of the 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 from the classifiers chosen so far, i.e. the choose 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 its complexity is bounded by $O(n \cdot |\mathcal{H}|^2)$ (where n denotes the number of examples) due to the computation of the diversity matrix. For many 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 specialty metric for a particular target class on the validation set. Then, it iteratively selects 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. The algorithm repeats the selection process for a different target class. The complexity of the algorithm is bounded by $O(n \cdot c \cdot |\mathcal{H}|^2)$. For each target class (c is the total number) it considers at most $|\mathcal{H}|$ classifiers. 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 implemented instances of the pruning algorithm that we studied combine coverage with $PCS(C_j, c_k)$, coverage with a specific $CCS(C_j, c_k)$, and coverage with a combined class specialty metric that associates a specific cost model tailored to the credit card fraud detection problem.

4 Experiments and Results

Learning tasks Two data sets of real credit card transactions were used in our experiments provided by the Chase and First Union Banks. The two sets contain .5 million of credit card transactions labeled as fraudulent or legitimate 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 average distribution, whereas First Union data were sampled in a non-uniform manner (many records from some months, very few from others) with a total of 15% fraud versus 85% non-fraud.

To evaluate the meta-classifiers, we adopted three metrics: overall accuracy, $(TP - FP)$ spread and a cost model fitted to the credit card detection problem. Overall accuracy expresses the ability to give correct predictions, $(TP -$

[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.

FP)[5] denotes the ability 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 for challenging the legitimacy of a transaction. If the transaction amount *amt*, is below this threshold, the transaction is authorized 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 $\$Y$. Therefore, even if we could accurately predict and identify all fraudulent transactions, those whose *amt* is less than $\$Y$ would produce $(Y - amt)$ in losses anyway. With this overhead taken into account, we aim to produce meta-classifiers that maximize savings $S(C_i, Y)$:

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

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

Experiments First we distributed the data sets across six different data sites (each site getting two months of data) and we prepared the set of base classifiers the pruning algorithm is called to evaluate. We obtained these classifiers by applying 5 learning algorithms (three decision-tree algorithms, ID3, its successor C4.5, and Cart [3], a naive Bayesian algorithm [18], and the rule induction algorithm Ripper [9]) on each month of data, therefore creating 60 base classifiers (10 classifiers per data site). Next, we had each site import the "remote" base classifiers (50 in total), and used only these in the pruning and meta-learning phases thus ensuring that no classifier would be tested unfairly on known data. Specifically, we had each site use half of its local data (one month) to test, prune and meta-learn the base-classifiers and the other half to evaluate the performance of the pruned meta-classifier. Note that the above (month-dependent) data partitioning scheme, was applied only on the Chase data set. The very skewed nature of the First Union data forced us to equi-partition the entire data set randomly into 12 subsets and assign 2 subsets per site.

With 50 candidate classifiers per site, there are 2^{50} possible combinations from which the pruning algorithm has to choose one. In the meta-learning stage, we employed all five learning algorithms to learn the selected base classifiers. The plots reported in the next section, present the average performance results of pruning and meta-learning over all six sites. In essence, this experiment provides 6-fold cross validation.

Results The results from these experiments are displayed in Figure 1. The top part plots the overall accuracy, the $(TP - FP)$ spread and the savings (in dollars) for the Chase credit card data, and the bottom part for the First Union data. Each figure contrasts two specialty/coverage based, one diversity-based, a metric-specific pruning method and an additional classifier selection method denoted here as *arbitrary*. As the name indicates, *arbitrary* uses no particular strategy to evaluate base classifiers; instead it combines them in a "random" order, i.e. when they become available. The metric-specific pruning methods (which correspond to three different instances, accuracy, $(TP - FP)$,[6] and cost model, of the CCS specialty metric) evaluate, rank and select the base-classifiers according to their performance with respect to that metric. 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 specialty/coverage algorithms. The final Chase meta-classifier for the (TP-FP)/coverage algorithm,

[5]The $(TP - FP)$ spread is an ad-hoc, yet informative and simple metric characterizing the performance of the classifiers. 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 [21] in the ROC space.

[6]Fraud detection is a binary classification problem, hence we need not examine specialty on both target classes (Section 3). Then the expression: f_{TP}(fraud) $\cdot TP(C_j, fraud)$ + f_{FP}(fraud) $\cdot FP(C_j, fraud)$ is reduced to $TP(C_j, fraud) - FP(C_j, fraud)$ to match the evaluation metric.

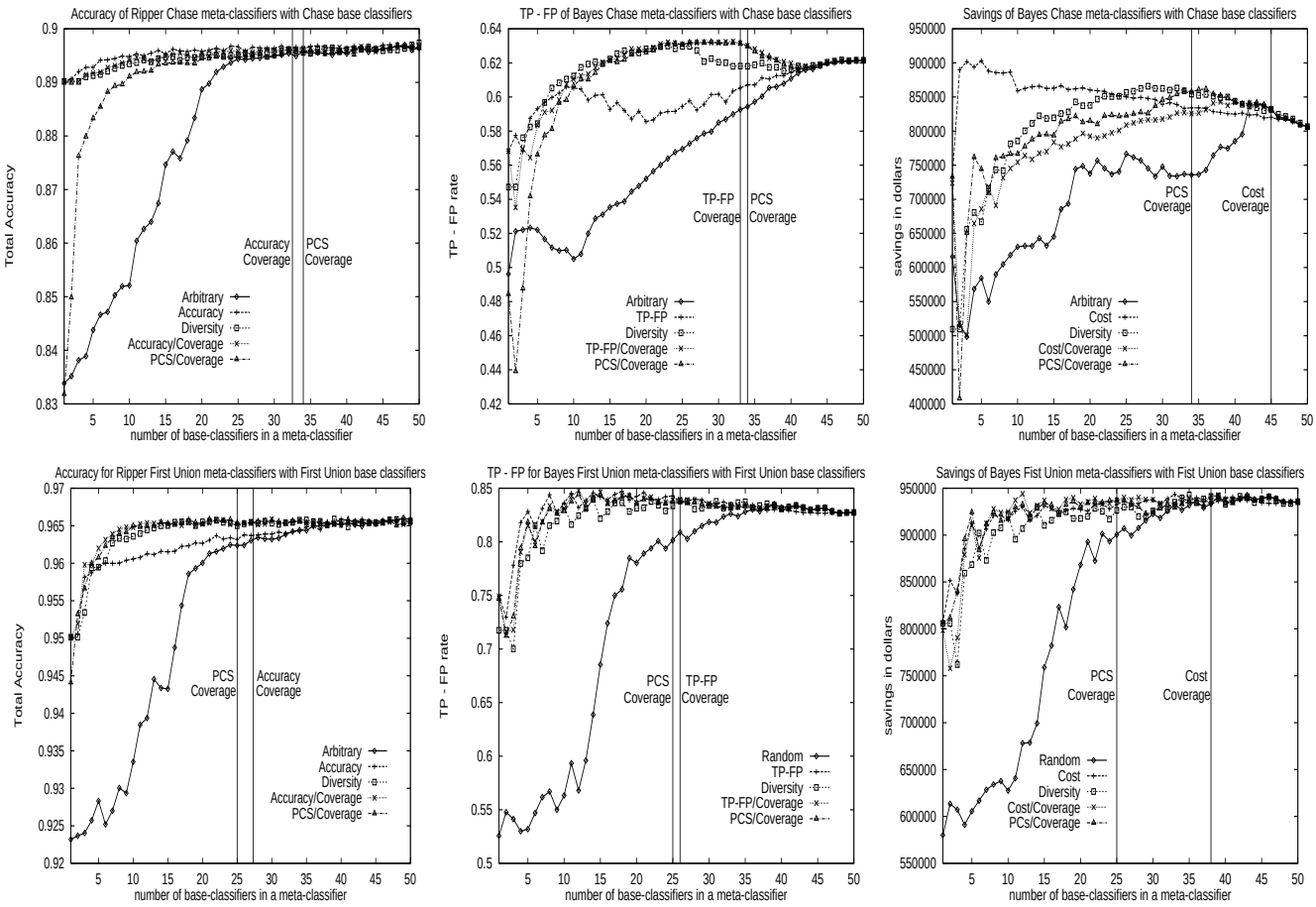

Figure 1: Accuracy (left), TP-FP (middle) and savings (right) of pruning algorithms on Chase (top) and First Union (bottom) data.

for example, combines 33 base classifiers (denoted by the (TP-FP)/coverage vertical line). 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 the redundant meta-classifiers would have had, had we used more base-classifiers or not introduced the pruning phase. Vertical lines for the diversity-based and metric-specific pruning algorithms are not shown since they depend on real-time constraints and available resources as discussed in Section 3.

The benefits from the pruning methods are clear. In all cases the performance of the pruned meta-classifiers are superior to that of the unpruned meta-classifier[7] and certainly better than the "arbitrary" pruning method both with respect to effectiveness (accuracy, $(TP-FP)$, savings) as well as efficiency (few 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 also achieve 60% better throughput, 1% higher $(TP - FP)$ spread and \$100K/month additional savings than the unpruned 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.

Analysis The graphs indicate that the metric-specific pruning methods choose better combinations of Chase classifiers, while the specialty/coverage- and diversity-based pruning methods perform better for the First Union classifiers. This observation is more pronounced in most of the plots (not shown here) generated from the meta-classifiers com-

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

puted by the other learning algorithms.

The performance of a meta-classifier, however, is directly related to the properties and characteristics of its constituent base-classifiers. As we have already noted, the more diverse the set of base-classifiers is, the more room for improvement the meta-classifier has. For example, to obtain diverse classifiers from a single learning program Freund and Schapire [11] introduced the boosting algorithm for re-sampling the data set to artificially generate diverse training subsets. In our experiments, the diversity of the base classifiers is attributed, first, to the use of disparate learning algorithms, and second, to the degree the training sets are different. Although the first factor is the same for both Chase and First Union data sets, this is not the case with the second. Recall that the First Union classifiers were trained on subsets of data of equal size and class distribution, while the Chase base-classifiers were trained on subsets of data defined according to the date of the credit card transaction, which led to variations in the size of the training sets and the class distributions. As a result, in the Chase bank case, the simple metric-specific pruning algorithm combines the best base-classifiers that are already diverse and hence achieves superior results while the specialty/coverage pruning algorithms combine diverse base-classifiers that are not necessarily the best. On the other hand, in the First Union case, the specialty/coverage pruning algorithms are more effective, since the best base-classifiers are not as diverse.

Further observe that in the First Union data the various pruning algorithms are comparably successful (their plots are less distinguishable). Indeed, a closer inspection on the classifiers composing the pruned sets, revealed that, the sets of selected classifiers were more "similar" (there were more common members) for First Union than for Chase. The inspection showed that for First Union, the specialty/coverage-based and diversity-based pruning algorithms tended to select mainly the ID3 base-classifiers for being more specialized/diverse[8] thus substantiating the conjecture that the primary source of diversity for these meta-classifiers is the use of different learning algorithms. If the training sets for First Union were more diverse, there would have been more diversity among the other base-classifiers and presumably more variety in the pruned set. All pruning methods tend to converge after a certain point. After all, as they add classifiers, their pool of selected classifiers is bound to converge to the same set.

Note that training classifiers to distinguish fraudulent transactions is not a direct approach to maximizing savings (or the $(TP - FP)$ spread). 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 a 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 base-classifiers that are most accurate instead of the classifiers that accrue highest savings.

The superior performance of the cost-specific pruning method confirms this hypothesis ("cost" plot, Figure 1, top-right graph). The cost-specific pruning method forms meta-classifiers by selecting base-classifiers based on their performance with respect to the cost model. While the ensemble consists of base classifiers with good cost model performance the meta-classifier exhibits substantially improved performance as well. The curves for the First Union data set were not as distinct since the majority of the First Union base-classifiers happened to catch the "expensive" fraudulent transactions anyway so the pruning algorithms were able to form meta-classifiers with the appropriate base classifiers. The same, but to a lesser degree, holds for the $TP - FP$ spread. In general, unless the learning algorithm's target function is aligned with the evaluation metric, the resulting base- and meta-classifiers will never be able to address the classification problem as best as possible.

One way to deal with this situation in the credit card fraud domain, is to tune the learning problem to the cost

[8] ID3 is known to overfit its training sets. Small changes in the learning sets can force ID3 to compute significantly different classifiers.

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 Conclusions

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. Although there are many intricate issues that need to be addressed and resolved, the experiments suggest that pruning 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, especially when the learning algorithm's target function is different from the evaluation metric and only a small number of classifiers is appropriate.

References

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

[2] L. Breiman. Stacked regressions. *Machine Learning*, 24:41–48, 1996.

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

[4] C. Brodley and T. Lane. Creating and exploiting coverage and diversity. *AAAI-96 Workshop on Integrating Multiple Learned Models*.

[5] C.Brodley. Addressing the selective superiority problem: Automatic algorithm/model class selection. *10th Intl. Conf. Mach. Learning*, 1993.

[6] P. Chan. *An Extensible Meta-Learning Approach for Scalable and Accurate Inductive Learning*. PhD thesis, Columbia Univ., 1996.

[7] P. Chan and S. Stolfo. Meta-learning for multistrategy and parallel learning. *2nd Intl. Work. Multistrategy Learning*, pages 150–165, 1993.

[8] P. Chan and S. Stolfo. Sharing learned models among remote database partitions by local meta-learning. In *Proc. KDD'96*, pages 2–7, 1996.

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

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

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

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

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

[14] M. LeBlanc and R. Tibshirani. Combining estimates in regression and classification. TR 9318, Dept. of Statistics, Univ. of Toronto, 1993.

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

[16] C. Merz. Using correspondence analysis to combine classifiers. *Machine Learning*, 1998. to appear.

[17] R. Michalski. A theory and methodology of inductive learning. *Machine Learning: An Artificial Intelligence Approach*, pages 83–134, 1983.

[18] M. Minksy and S. Papert. *Perceptrons: An Introduction to Computation Geometry*. MIT Press, (Expanded edition, 1988).

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

[20] A. L. Prodromidis. On the management of distributed learning agents. CUCS-032-97 (PhD Thesis proposal), CS. Dept. Columbia Univ.,1997.

[21] F. Provost and T. Fawcett. Analysis 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.

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

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

[24] 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.

[25] 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.

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