Toward Scalable Learning with Non-uniform Class and Cost Distributions: A Case Study in Credit Card Fraud Detection

Article · July 1998

Source: CiteSeer

CITATIONS
357

READS
474

2 authors:

Philip K. Chan
Florida Institute of Technology
110 PUBLICATIONS **9,681** CITATIONS

SEE PROFILE

Salvatore J. Stolfo
Columbia University
315 PUBLICATIONS **21,088** CITATIONS

SEE PROFILE

Toward Scalable Learning with Non-uniform Class and Cost Distributions: A Case Study in Credit Card Fraud Detection *

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

Salvatore J. Stolfo
Department of Computer Science
Columbia University
New York, NY 10027
sal@cs.columbia.edu

March 16, 1998

Abstract

Very large databases with skewed class distributions and non-uniform cost per error are not uncommon in real-world data mining tasks. One such task is credit card fraud detection: the number of fraudulent transactions is small compared to legitimate ones, the amount of financial loss for each fraudulent transaction depends on the transaction amount and other factors, and millions of transactions occur each day. We devised a multi-classifier meta-learning approach to address these three issues. Our empirical results indicate that the approach can significantly reduce loss due to illegitimate transactions.

Keywords: skewed class distributions, non-uniform error cost, large amounts of data, credit card fraud detection.

*This work was partially funded by grants from DARPA (F30602-96-1-0311), NSF (IRI-96-32225 & CDA-96-25374), NYSSTF (423115-445).

1 Introduction

Very large databases with skewed class distributions and non-uniform cost per error are not uncommon in real-world data mining tasks. One such task is credit card fraud detection: the number of fraudulent transactions is small compared to legitimate ones, the amount of financial loss for each fraudulent transaction depends on the amount of transaction and other factors, and millions of transactions occur each day. A similar task is cellular phone fraud detection [12]. Each of these three issues has not been widely studied in the machine learning research community.

In Dec. 96 Fawcett [11] summarized the responses to his inquiry on learning with skewed class distributions. The number of responses was amazingly few given skewed distributions are not rare in the real world. Kubat and Matwin [14] acknowledged the performance degradation effects of skewed class distributions and investigated techniques for removing unnecessary instances from the majority class. Instances that are in the borderline region, noisy, or redundant are candidates for removal. Cardie and Howie [3] stated that skewed class distributions are "the norm for learning problems in natural language processing (NLP)." In a case-based learning framework, they studied techniques to extract relevant features from previously built decision trees and customize local feature weights for each case retrieval.

Error rate (or accuracy) is commonly used in evaluating learning algorithms; cost-sensitive learning has not been widely investigated. Assuming the errors can be grouped into a few types and each type incurs the same cost, some studies (for example, [15]) proposed algorithms that aim to reduce the total cost. Another line of cost-sensitive research tries to reduce the cost in using a classifier. For instance, some sensing devices are costlier in the robotics domain [19] and some medical tests are more expensive in the medical diagnosis domain. Fawcett and Provost [12] considered non-uniform cost per error in their cellular phone fraud detection task. However, they only considered non-uniform cost in evaluating different algorithms—the learning process was not cost-sensitive.

Until recently, researchers in machine learning have been focused on small data sets. Efficiently learning from large amounts of data has been gaining attention due to the fast growing field of data mining, where data are abundant. Sampling (e.g., [4]) and parallelism (e.g., [13, 17]) are the two main directions in scalable learning. Much of the parallelism work focuses on parallelizing a particular algorithm on a particular parallel architecture. That is, a new algorithm or architecture requires substantial amount of parallel programming work.

We devised a multi-classifier meta-learning approach to address these three issues. Our approach is based on creating data subsets with the appropriate class distribution, applying learning algorithms to the subsets independently and in parallel, and integrating to optimize cost performance of the classifiers by learning (meta-learning [5]) from their classification behavior. That is, our method utilizes all available training examples and does not change the underlying learning algorithms. It also handles non-uniform cost per error and is cost-sensitive during the learning process. Although our architecture and algorithm-independent approach is not as efficient as the fine-grained parallelization approaches, it allows different "off the shelf" learning programs to be "plugged" into a parallel and distributed environment with relative ease. Our empirical results for the credit card fraud problem indicate that our approach can significantly reduce loss due to illegitimate transactions.

We describe the credit card fraud detection task in Section 2. Section 3 examines the

effects of training class distributions on the performance. Section 4 discusses our multi-classifier meta-learning approach. For completeness, part of the exposition in this article also appears in a companion paper [8]. Section 5 summaries our results and directions.

2 Credit Card Fraud Detection

When banks lose money because of credit card fraud, card holders partially (possibly entirely) pay for the loss through higher interest rates, higher membership fees, and reduced benefits. Hence, it is both the banks' and card holders' interest to reduce illegitimate use of credit cards, particularly when plastic is prevalent in today's increasingly electronic society. The Chase Manhattan Bank provided us with a data set that contains 500,000 transactions from 10/95 to 9/96, about 20% of which are fraudulent (the real distribution is much more skewed (*fortunately*)—the 20:80 distribution is what we were given).

Due to the different dollar amount of each credit card transaction and other factors, the cost of failing to detect different fraudulent transactions is not the same. Hence we define:

$$AverageAggregateCost = \frac{1}{n}\sum_{i}^{n} Cost(i) \tag{1}$$

where $Cost(i)$ is the cost associated with instance i and n is the total number of instances.

After consulting with a bank representative, we settled on a simplified cost model (the cost model used by the bank is more complex and is still evolving). Since it takes time and personnel to investigate a potential fraudulent transaction, an *overhead* is incurred for each investigation. That is, if the amount of a transaction is smaller than the overhead, it is not worthwhile to investigate the transaction even if it is suspicious. For example, if it takes ten dollars to investigate a potential loss of one dollar, it is more economical not to investigate. Therefore, assuming a fixed *overhead* and considering:

[Number of instances]	Actual Positive (fraudulent)	Actual Negative (legitimate)
Predicted Positive (fraudulent)	True Positive (*Hit*) [a]	False Positive (*False Alarm*) [b]
Predicted Negative (legitimate)	False Negative (*Miss*) [c]	True Negative (*Normal*) [d]

we devised the following cost model for each transaction:

$Cost(FN)$	$tranamt$
$Cost(FP)$	$overhead$ if $tranamt > overhead$ or 0 if $tranamt \leq overhead$
$Cost(TP)$	$overhead$ if $tranamt > overhead$ or $tranamt$ if $tranamt \leq overhead$
$Cost(TN)$	0

where $tranamt$ is the amount of a credit card transaction. Furthermore, we define the false-negative rate as $\frac{c}{a+c}$ and the false-positive rate as $\frac{b}{b+d}$. Based on this cost model, we next study the effects of training class distributions on performance.

3 Effects of Training Distributions

Experiments were performed to study the effects of training class distributions on the credit card cost model. We use data from the first 10 months (10/95 - 7/96) for training and

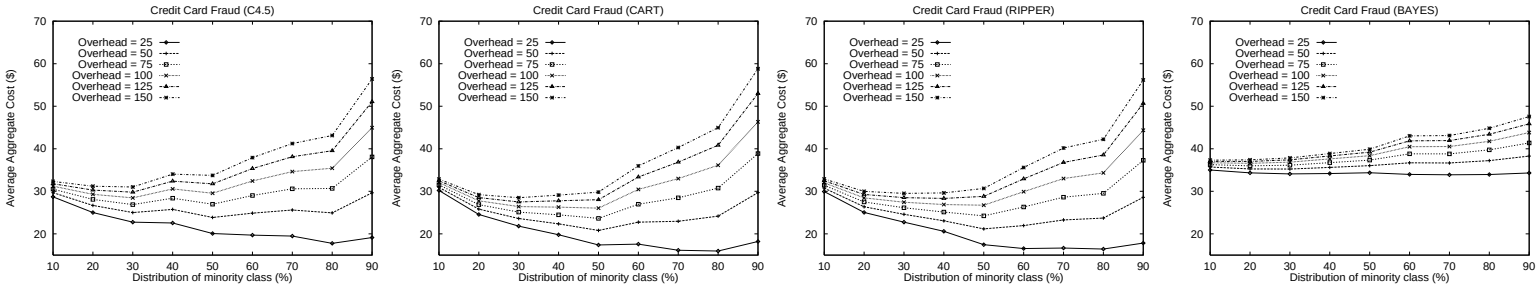

Figure 1: Training distribution vs. the credit card fraud cost model

Table 1: Cost and saving in the credit card fraud domain

Learning Alg.	Overhead = $50				Overhead = $75				Overhead = $100			
	Cost	fraud%	%saved	$saved	Cost	fraud%	%saved	$saved	Cost	fraud%	%saved	$saved
C4.5	23.85	50%	35%	525K	26.88	30%	27%	404K	28.46	30%	23%	341K
CART	20.80	50%	44%	647K	23.64	50%	36%	534K	26.05	50%	30%	437K
RIPPER	21.16	50%	43%	632K	24.23	50%	34%	510K	26.73	50%	28%	409K
BAYES	35.23	30%	5%	69K	35.99	20%	3%	39K	36.58	20%	1%	15K

the 12th month (9/96) for testing. In order to vary the fraud distribution from 10% to 90% for each month, we limit the size of the training sets to 6,400 transactions, which are sampled randomly without replacement. Four learning algorithms (C4.5, CART, RIPPER, and BAYES) were used in our experiments. We obtained C4.5 [18] and CART [1] as part of the IND package [2] from NASA Ames Research Center; both algorithms compute decision trees. RIPPER [10] is a rule learning algorithm and was obtained from W. Cohen. BAYES is a naive Bayesian learning algorithm that is based on computing conditional probabilities using the Bayes Rule as described in [9].

The results are plotted in Figure 1. Each data point is an average of 10 classifiers, each of which is generated from a separate month. Each curve represents a different amount of overhead. Fraud is the minority class. As expected, the larger overhead leads to higher cost. More importantly, we observe that when the overhead is smaller, the cost minimizes at a larger percentage of fraudulent transactions (minority class) in the training set. When the overhead is smaller, the bank can afford to send larger number of transactions for investigation. That is, the bank can tolerate more false-alarms (a higher false-positive rate) and aim for fewer misses (a lower false-negative rate), which can be achieved by a larger percentage of fraudulent transactions (positive's) [8]. Conversely, if the overhead is larger, the bank should aim for fewer false-alarms (a lower FP rate) and tolerate more misses (a higher FN rate), which can be obtained by a smaller percentage of positive's. (Note that, at some point, the overhead can be large enough making fraud detection economically unattractive.)

The test set (from 9/96) has 40,038 transactions and 17.5% of them are fraudulent. If fraud detection is not available, on the average, $36.96 is lost per transaction. Table 1 shows the maximum savings of each algorithm with the most effective fraud percentage. *Cost* is the dollars lost per transaction; *fraud%* denotes the most effective fraud percentage for training; *%saved* represents the percentage of savings from the average loss of $36.96; *$saved* shows the total dollars saved for the month (9/96).

BAYES performed relatively poor (we suspect the way we are treating attributes with real values in BAYES is not appropriate for the fraud domain) and is excluded from the

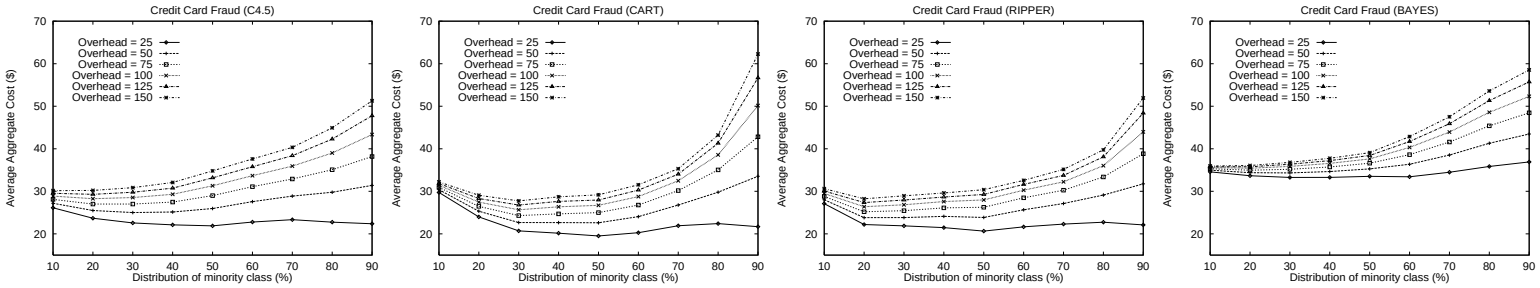

Figure 2: Training distribution vs. cost with cost-based sampling

Table 2: Cost and saving using cost-based sampling

Learning Alg.	Overhead = $50				Overhead = $75				Overhead = $100			
	Cost	fraud%	%saved	$saved	Cost	fraud%	%saved	$saved	Cost	fraud%	%saved	$saved
C4.5	25.01	30%	32%	478K	27.00	30%	27%	399K	28.26	20%	24%	348K
CART	22.61	50%	39%	574K	24.30	30%	34%	507K	25.66	30%	31%	452K
RIPPER	23.82	20%	36%	526K	26.38	20%	32%	471K	26.38	20%	29%	424K
BAYES	34.32	30%	7%	106K	34.95	20%	5%	80K	35.43	20%	4%	61K

following discussion. Considering the amount of overhead ranged from $50 to $100, the learning algorithms we used generally achieved at least 20% in savings or at least $300K. With an overhead of $75 or less, at least half a million dollars in savings can be attained. More importantly, maximum savings were achieved 7 out of 9 times when the fraud percentage used in training is 50%, which is not the "natural" 20% in the original data set.

3.1 Cost-based sampling

Since random sampling in our previous experiments reflects the natural distribution of transactions in the data set and smaller transactions are more abundant, smaller transactions are sampled more often than the larger ones. We hypothesize that if we sample proportional to the transaction amount, the learned classifiers are trained from transactions with larger amounts and more apt to catch larger fraud. In this set of experiments, again, we varied the fraud percentage from 10% to 90%. If the number of available fraudulent (or normal) transactions is larger than the desired ones, the transactions are sampled without replacement according to:

$$P(fraud_i) = \frac{tranamt(fraud_i)}{\sum_i tranamt(fraud_i)}$$

In Figure 2 each data point is an average of 10 classifiers from the first 10 months. Table 2 shows the cost and savings with the same format as Table 1. The cost is minimized at a lower fraud percentage, around 20-30%, but the performance is not much different from random sampling in the previous set of experiments (Table 1). (Cost-proportional sampling *with* replacement did not yield significantly different results either.)

3.2 Stratification

Since in our previous experiments each classifier was generated from data from the same month, we next attempt to study if stratifying the data would have any impact on the

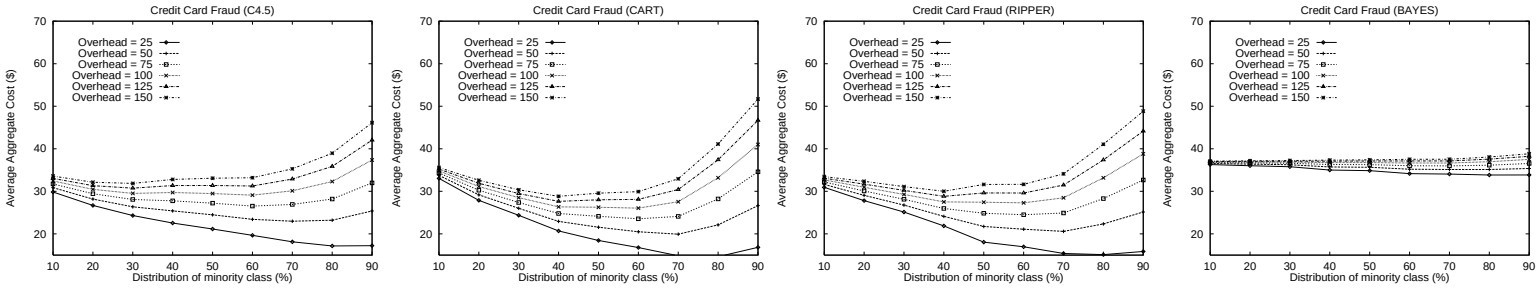

Figure 3: Training distribution vs. cost using stratification

Table 3: Cost and saving using stratification

Learning Alg.	Overhead = $50				Overhead = $75				Overhead = $100			
	Cost	fraud%	%saved	$saved	Cost	fraud%	%saved	$saved	Cost	fraud%	%saved	$saved
C4.5	22.96	70%	38%	560K	26.52	60%	28%	418K	29.09	60%	21%	315K
CART	19.92	70%	46%	682K	23.54	60%	36%	537K	26.05	60%	30%	437K
RIPPER	20.58	70%	44%	656K	24.50	60%	34%	499K	27.29	60%	26%	387K
BAYES	35.12	70%	5%	74K	35.97	70%	3%	40K	36.64	70%	1%	13K

results. In *stratification* a classifier is generated from a training set that contains 1/10 of the data from each month and 10 classifiers are produced. Again, in this set of experiments, the fraud percentage varies from 10% to 90% and the average of 10 classifiers are plotted in Figure 3. Table 3 shows the cost and savings using the same format as the previous tables. The cost minimizes at a higher fraud percentage at around 60-70%, but the performance does not significantly differ from not using stratification (Table 1).

In summary, based on our empirical results in this study, we observe that varying the training class distribution has larger effects on cost performance than cost-based sampling or stratification. Since the "natural" distribution is 20:80, one way to achieve the "desired" 50:50 distribution is to ignore 75% of the legitimate transactions (or 60% of all the transactions), as we did in the experiments above. The following section discusses an approach that utilizes all the data and improves performance.

4 A Multi-classifier and Meta-learning Approach to Skewed Distributions

As we discussed earlier, using the natural class distribution might not yield the most effective classifiers (particularly when the distribution is highly skewed). Given a skewed distribution, we would like to generate the desired distribution without removing any data. Our approach is to create data subsets with the desired distribution, generate classifiers from the subsets, and integrate them by learning (meta-learning) from their classification behavior. In our fraud domain, the natural skewed distribution is 20:80 and the desired distribution is 50:50. We randomly divide the majority instances into 4 partitions and 4 data subsets are formed by merging the minority instances with each of the 4 partitions containing majority instances. That is, the minority instances are replicated across 4 data subsets to generate the desired 50:50 distribution. Figure 4 depicts this process.

Formally, let n be the size of the data set with a distribution of $x : y$ (x is the percentage

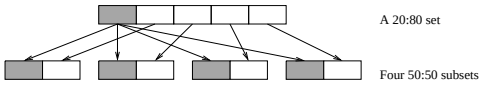

Figure 4: Generating four 50:50 data subsets from a 20:80 data set

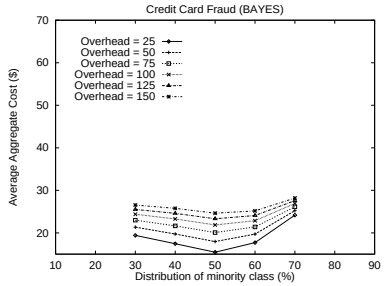

Figure 5: Training distribution vs. cost using meta-learning

of the minority class) and $u : v$ be the desired distribution. The number of minority instances is $n \times x$ and the desired number of majority instances in a subset is $nx \times \frac{v}{u}$. The number of subsets is the number of majority instances ($n \times y$) divided by the number of desired majority instances in each subset, which is $\frac{ny}{\frac{nxv}{u}}$ or $\frac{y}{x} \times \frac{u}{v}$. (When it is not a whole number, we take the ceiling ($\lceil \frac{y}{x} \times \frac{u}{v} \rceil$) and replicate some majority instances to ensure all of the majority instances are in the subsets.) That is, we have $\frac{y}{x} \times \frac{u}{v}$ subsets, each of which has nx minority instances and $\frac{nxv}{u}$ majority instances.

The next step is to apply a learning algorithm(s) to each of the subsets. Since the subsets are independent, the learning process for each subset can be run in parallel on different processors. For massive amounts of data, substantial improvement in speed can be achieved for super-linear-time learning algorithms.

The generated classifiers are combined by learning (meta-learning) from their classification behavior. Several meta-learning strategies are described in [5]. To simplify our discussion, we only describe the *class-combiner* (or *stacking* [20]) strategy. In this strategy a meta-level training set is composed by using the (base) classifiers' predictions on a validation set as attribute values and the actual classification as the class label. This training set is then used to train a meta-classifier. For integrating subsets, class-combiner can be more effective than the voting-based techniques [6].

4.1 Experiments and Results

To evaluate our multi-classifier meta-learning approach to skewed class distributions, we used transactions from the first 8 months (10/95 - 5/96) for training, the ninth month (6/96) for validating, and the twelfth month (9/96) for testing (the two-month gap is chosen according to the amount of time needed to completely determine the legitimacy of transactions). Based on the empirical results in Section 3, the desired distribution is 50:50. Since the natural distribution is 20:80, four subsets are generated from each month for a total of 32 subsets. We applied four learning algorithms (C4.5, CART, RIPPER, and BAYES) to each subset and generated 128 base classifiers. BAYES was used to train the meta-classifier.

Results from different amounts of overhead are plotted in Figure 5. Each data point is the average of ten runs using different random seeds. To demonstrate that 50:50 is indeed

Table 4: Cost and savings using meta-learning

Method	Overhead = $50				Overhead = $75				Overhead = $100			
	Cost	fraud%	%saved	$saved	Cost	fraud%	%saved	$saved	Cost	fraud%	%saved	$saved
Class combiner	17.96	50%	51%	761K	20.07	50%	46%	676K	21.87	50%	41%	604K
Single CART	20.80	50%	44%	647K	23.64	50%	36%	534K	26.05	50%	30%	437K
Class combiner	22.61	natural	39%	575K	23.99	natural	35%	519K	25.20	natural	32%	471K
Avg. single classifier	27.97	natural	24%	360K	29.08	natural	21%	315K	30.02	natural	19%	278K

the desired distribution, we also performed experiments on other distributions and plotted the results in the figure. As expected, the cost is minimized when the fraud percentage is 50%. Surprisingly, 50% is the desired distribution for any of the overhead amount. This is different from the results obtained from previous experiments when meta-learning was not used.

Furthermore, to investigate if our approach is indeed fruitful, we ran experiments on the class-combiner strategy directly applied to the original data sets from the first 8 months (i.e., they have the natural 20:80 distribution). We also evaluate how individual classifiers generated from each month perform without class-combining.

Table 4 shows the cost and savings from class-combiner using the 50:50 distribution (128 base classifiers), the average of individual CART classifiers generated using the desired distribution (from Table 1), class-combiner using the natural distribution (32 base classifiers—8 months $\times$ 4 learning algorithms), and the average of individual classifiers using the natural distribution (the average of 32 classifiers). Compared to the other three methods, class-combining on subsets with a 50:50 fraud distribution clearly achieves a significant increase in savings. At least $600K were saved for the month (6/96). When the overhead is $50, more than half of the losses were prevented. Surprisingly, we also observe that when the overhead is $50, a classifier ("single CART") trained from one's month data with the desired 50:50 distribution (generated by throwing away some data) achieved significantly more savings than combining classifiers trained from all eight months' data with the natural distribution. This reaffirms the importance of employing the appropriate training class distribution in this domain.

4.2 Class distribution in the validation set

Thus far we have concentrated on the class distributions in training the base classifiers. We hypothesize that the class distribution in the validation set (and hence the meta-level training set) affects the overall performance of meta-learning. To investigate that hypothesis, in this set of experiments, we fixed the training distribution of the base classifiers to 50:50 and varied the distribution of the validation set from 30:70 to 70:30 (as in the previous set of experiments). The different distributions are generated by keeping all the fraudulent transactions and randomly selecting the corresponding number of legitimate transactions (i.e., some legitimate records are thrown away).

Each data point in Figure 6 is an average of 10 runs using different random seeds (though the same set as in pervious experiments). To prevent amplifying small differences unnecessarily, we use the same scale in Figure 6 as in the other similar figures. Unexpectedly, the curves are quite flat. That is, changing the class distribution in the validation set does not

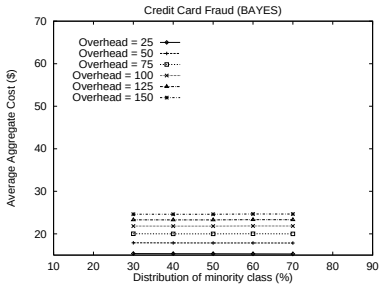

Figure 6: Distribution in validation set vs. cost using meta-learning

Table 5: Cost and savings using different validation distributions

	Overhead = $50				Overhead = $75				Overhead = $100			
Method	Cost	Val. fraud%	%saved	$saved	Cost	Val. fraud%	%saved	$saved	Cost	Val. fraud%	%saved	$saved
Class combiner	17.96	natural	51%	761K	20.07	natural	46%	676K	21.87	natural	41%	604K
Class combiner	17.84	70%	52%	766K	19.99	50%	46%	679K	21.81	50%	41%	607K

seem to affect the cost performance (a subject of further investigation).

Table 5 has the same format as the previous cost and savings tables, but the *fraud%* columns are replaced by *Val. fraud%*, which denotes the fraud percentage in the validation set. The first row shows the results from class-combiner using the natural distribution in the validation set (from Table 4). The second row has the lowest cost performance and the corresponding fraud percentage in the validation set. The cost is minimized at around 50% (the difference is quite small among different distributions (Figure 6)), but it is not much different from the performance obtained from the natural distribution. Note that a 50:50 distribution was obtained by throwing away 60% of the data from the natural 20:80 distribution in the validation set. That is, comparable performance can be achieved with less data in the validation set, which is particularly beneficial in this domain that has large amounts of data. We next investigate another means of reducing the training time.

4.3 Excluding transactions below the overhead

Since transactions with amounts smaller than the overhead are generally automatically approved, they might contribute little to evaluating transactions with amounts larger than the overhead. Excluding the small transactions during training reduces the training time. This is particularly beneficial in our domain because many credit card transactions are relatively small—in our data 52% of the transactions are smaller than or equal to $100. However, an important question remains: how does this approach affect the cost performance?

As before we focused on three overheads: $50, $75, and $100. After removing transactions with amounts below an overhead, the data set has a distribution close to 15:85 for each of the three overheads (this is different from the original 20:80). In this set of experiments, the class distribution is 50:50 for training the base classifiers. Table 6 shows results obtained from using the natural distribution (first row) and 50:50 distribution (second row) in the validation set. The results produced where we did not exclude any transactions are presented in the third row (from Table 4). The outcome indicates that removing data below the overhead substantially degrades the cost performance. Moreover, using a 50:50 distribution in the

Table 6: Cost and savings with only transactions > overhead

Method	Overhead = $50				Overhead = $75				Overhead = $100			
	Cost	Val. fraud%	%saved	$saved	Cost	Val. fraud%	%saved	$saved	Cost	Val. fraud%	%saved	$saved
Class comb. ($> OH$)	23.02	natural	38%	558K	24.37.	natural	34%	504K	25.21	natural	32%	470K
Class comb. ($> OH$)	22.79	50%	38%	567K	24.20	50%	35%	511K	25.09	50%	32%	475K
Class combiner	17.96	natural	51%	761K	20.07	natural	46%	676K	21.87	natural	41%	604K

validation set does not help reduce the degradation. This conforms with the results from the previous section on varying the validation distribution (Figure 6).

5 Concluding Remarks

This study demonstrates that the training class distribution affects the performance of the learned classifiers and the natural distribution can be different from the desired training distribution to maximize performance. Also, in this domain, cost-based sampling and stratification have smaller effects than training class distributions. Removing transactions with amounts lower than the overhead can shorten training time, but not without sacrificing some cost performance.

Moreover, our empirical results indicate that our multi-classifier meta-learning approach using a 50:50 distribution in the data subsets for training can significantly reduce the amount of loss due to illegitimate transactions. The subsets are independent and can be processed in parallel. Training time can further be reduced by also using a 50:50 distribution in the validation set without degrading the cost performance. That is, this approach provides a means for efficiently handling learning tasks with skewed class distributions, non-uniform cost per error, and large amounts of data. Not only is our method efficient, it is also scalable to larger amounts of data.

One limitation of our approach is the need to run preliminary experiments to determine the desired distribution based on a defined cost model. This process can be automated but it is unavoidable since the desired distribution is highly dependent on the cost model and the learning algorithm.

Using four learning algorithms, our approach generates 128 classifiers from a 50:50 class distribution and eight months of data. We might not need to keep all 128 classifiers since some of them could be highly correlated and hence redundant. Also, more classifiers are generated when the data set is larger or additional learning algorithms are incorporated. Metrics for analyzing an ensemble of classifiers (e.g., diversity, correlated error, and coverage) can be used in pruning unnecessary classifiers [16]. More importantly, since thieves also learn and fraud patterns evolve over time, some classifiers are more relevant than others at a particular time. Therefore, an adaptive classifier selection method is essential. Unlike a monolithic approach of learning one classifier using incremental learning, our modular multi-classifier approach facilitates adaptation over time and removal of out-of-date knowledge. Although banks do not share credit card data for fear of losing valuable customers to competitors or violating the customers' privacy, a bank can import "black-box" classifiers from other banks to improve its local performance [7].

References

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

[2] W. Buntine and R. Caruana. *Introduction to IND and Recursive Partitioning.* NASA Ames Research Center, 1991.

[3] C. Cardie and N. Howe. Improving minority class prediction using case-specific feature weights. In *Proc. 14th Intl. Conf. Mach. Learning*, pages 57–65, 1997.

[4] J. Catlett. Megainduction: A test flight. In *Proc. Eighth Intl. Work. Machine Learning*, pages 596–599, 1991.

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

[6] P. Chan and S. Stolfo. A comparative evaluation of voting and meta-learning on partitioned data. In *Proc. Twelfth Intl. Conf. Machine Learning*, pages 90–98, 1995.

[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] P. Chan and S. Stolfo. The effects of training class distributions on performance using cost models. Submitted to 15th Intl. Conf. Machine Learning, 1998.

[9] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–285, 1989.

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

[11] T. Fawcett. Learning with skewed class distributions–summary of responses. Machine Learning List, Vol. 8, No. 20, Dec 1996.

[12] T. Fawcett and F. Provost. Combining data mining and machine learning for effective user profiling. In *Proc. 2nd Intl. Conf. Knowledge Discovery and Data Mining*, pages 8–13, 1996.

[13] E. Han, G. Karypis, and V. Kumar. Scalable parallel data mining for association rules. In *Proc ACM-SIGMOD-97*, 1997.

[14] M. Kubat and S. Matwin. Addressing the curse of imbalanaced training sets: One sided selection. In *Proc. 14th Intl. Conf. Machine Learning*, pages 179–186, 1997.

[15] M. Pazzani, C. Merz, T. Hume P. Murphy, K. Ali, and C. Brunk. Reducing misclassification costs. In *Proc. 11th Intl. Conf. Machine Learning*, pages 217–225, 1994.

[16] A. Prodromidis, S. Stolfo, and P. Chan. Pruning classifiers in a distributed meta-learning system. Submitted to 4th Intl. Conf. Knowledge Discovery and Data Mining, 1998.

[17] F. Provost and J. Aronis. Scaling up inductive learning with massive parallelism. *Machine Learning*, 23:33–46, 1996.

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

[19] M. Tan. Cost-sensitive learning of classification knowledge and its applications in robotics. *Machine Learning*, 13:7, 1993.

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