

A Framework for Scalable Cost-sensitive Learning Based on Combining Probabilities and Benefits

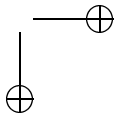
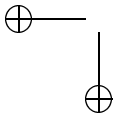
Wei Fan, Haixun Wang, Philip Yu, and Sal Stolfo†*

Abstract

We present a general framework for scalable cost-sensitive learning based on ensembles of classifiers. To compute ensembles for scalable learning, base models are computed from data subsets and their predictions are combined. Previous research mainly focus on combining *class labels* for *cost-insensitive* problems. In our study, we concentrate on how to combine the *probability* and *benefit* outputs (such as charity donation amount) for scalable *cost-sensitive* learning. We identify a few properties of both averaging and cost-sensitive optimal decision-making. Due to these properties, averaging has great potential for higher benefit as well as its obvious advantage of scalability. Experiments have shown that the averaging methods achieve a benefit level as good as or even better than the respective single classifier trained from the same dataset as a whole. However, the averaging methods exhibit linear speedup and scaled speedup that are independent of learning algorithms. Besides the averaging methods, we have experimented with tree-structured methods using variations of meta-learning and regression. Nonetheless, these approaches do not lead to higher benefits but consume higher overhead, hence are less scalable. Compared with tree-structured methods, the averaging methods are more efficient to classify examples, and easier to implement and verify.

*IBM T. J. Watson Research, email addresses {weifan, haixun, psyu}@us.ibm.com

†Department of Computer Science, Columbia University, email address sal@cs.columbia.edu

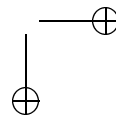
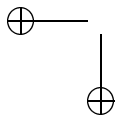


1 Introduction

Lately, cost-sensitive learning has been an area of extensive research interests [5]. In many areas of application where different examples carry different benefits, it is not enough to maximize the accuracy based on the simplified assumption that each example has the same benefit and there is no penalty for misclassification. For example, credit card fraud detection seeks to maximize the total transaction amount of correctly detected frauds minus the cost to investigate all (correctly and incorrectly) detected frauds. Undetected fraud causes a loss of the whole transaction amount; it is far more profitable to detect frauds with high transaction amount than those whose amount is not even higher than the cost to investigate. Cost-sensitive learning will be more challenging, when the databases are very large, sometimes, too large to fit into main memory. In reality, terabyte dataset is common. A typical credit card company has millions of new transactions on a daily basis. Training on large datasets will be very slow since most algorithms require all data to be held in main memory. The training complexity of many algorithms is worse than linear even if the data can be held in main memory. In this paper, we are interested in proposing new approaches to solve cost-sensitive scalable problems.

In this paper, we develop a framework of scalable methods to perform cost-sensitive learning using ensemble of classifiers, while maintaining similar or even better benefit than a single classifier. The original dataset is partitioned into subsets; base classifiers are trained from data subsets and their probability and benefit outputs are combined to produce the final prediction. Previous work focuses on combining *class labels* for *cost-insensitive* problems. We study how to combine *class probabilities* and *benefits* for *cost-sensitive* problems. We extensively explore a wide range of combining methods that include averaging, meta-learning [1] and regression. We identify a few properties of both averaging and cost-sensitive optimal decision-making. Due to these properties, averaging has good potential of higher benefit in addition to its obvious advantage of high scalability. We propose two combining methods that use averages of probability and benefit, and don't require any additional training. Our study has shown that the averaging of probabilities and benefits of base classifiers gain excellent benefit while avoiding the overhead to train meta-level classifier to combine probabilities and benefits. Both theoretical analysis and empirical study have shown that the averaging ensemble achieves linear *speedup* and *scaled speedup*, and superlinear improvement (defined in Section 3, and it is different from speedup) over the single model. At the same time, it maintains the same level of or even higher benefit than the single model trained on the entire dataset. Our experiments show that this holds for very large number of partitions. Tree-structured combining methods or *meta-learning* have been shown by Chan [1] to be effective to combine class labels. We experiment to use variations of meta-learning and also tree-structured regression to combine probabilities and benefits. However, our studies have found that these more complicated methods yield neither better scalability nor higher benefit.

Difference from previous work There has been much research in ensembles of classifiers in the past decade. Most of the existing approaches focus on *cost-insensitive* problems and combine *class label* outputs. However we are interested



in *cost-sensitive* problems and combining *probability and benefit* outputs. In [6, 4], Dietterich and Fan review different existing methods on how to generate and combine base classifiers. Nonetheless, there is limited research to use ensembles on cost-sensitive datasets, especially, to combine probabilities and benefits to solve cost-sensitive problems. Our approach addresses this problem and achieves satisfactory efficiency and benefit. In [3], Chan and Stolfo generated base classifiers with different percentage of positive examples to detect credit card fraud. Each base classifier outputs class labels which are combined using meta-learning [1]. Chan’s meta-learning [1] builds up a tree of classifiers to combine *class labels* to scale up *cost-insensitive* learning. Neither probability nor benefit is used in making a prediction. As discussed in Section 6, our approaches not only apply to *cost-sensitive* problems, but are also more scalable than meta-learning, i.e. linear speedup that is independent of the algorithm versus sublinear speedup that is dependent on the algorithm. Moreover, we also explored to use variations of meta-learning to combine probabilities and benefits, but neither benefit nor scalability is as good as the simpler averaging methods.

The rest of the paper proceeds as follows. In Section 2, we discuss the optimal decision-making policy and two of its important properties that are used to design the combining mechanisms. We also review the probability and benefit calculation methods. We discuss the details of the new combining approaches in Section 3. In particular, we formally analyze the scalability of the combining methods in Section 3. The experimental design and results are presented in Section 4. In Section 6, we discuss related works. The paper ends with a concluding remark in Section 7.

2 Cost-sensitive Decision Making

We start with an example of cost-sensitive learning. Suppose that the cost to investigate a fraud for a credit card transaction x is \$90 and the amount of transaction for x is $Y(x)$. In this case, the *optimal decision-making policy* is to predict x being fraud if and only if $(R(x) = P(\text{fraud}|x) \cdot Y(x)) > 90$, where $P(\text{fraud}|x)$ is the estimated probability that x is a fraud. $R(x)$ is called the *risk* to solicit x .

This policy has a few interesting properties. For simplicity, we use $P(x)$ instead of $P(\text{fraud}|x)$ to denote estimated probabilities. First, the exact value of estimated probability $P(x)$ is not important as long as it does not switch from above to below (or vice versa) a *decision threshold*. The decision threshold $T(x)$ is the threshold to predict x being *positive* or fraud in this example. For credit card fraud detection, $T(x) = \frac{90}{Y(x)}$. Re-writing the optimal decision policy using decision threshold, if and only if $P(x) > T(x)$, the optimal decision is to predict fraud; otherwise, the decision is nonfraud. It means that the exact value of $P(x)$ is not crucial; if $T(x)$ is 0.2, both $P(x) = 0.4$ and 0.5 produce the same predictions. We call this property *error-tolerance*. It makes probability estimate resilient to small errors. The second useful property is that the decision-making policy is biased towards predicting expensive examples (those with high $Y(x)$) to be positive. Since $T(x) \propto \frac{1}{Y(x)}$, $T(x)$ is small for expensive instances. It is more likely for $P(x)$ of expensive examples to be higher than $T(x)$. We call this property *expensive example bias*. Both properties help us design effective combining mechanisms (Section 3).

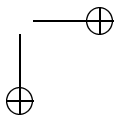
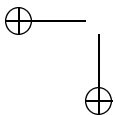
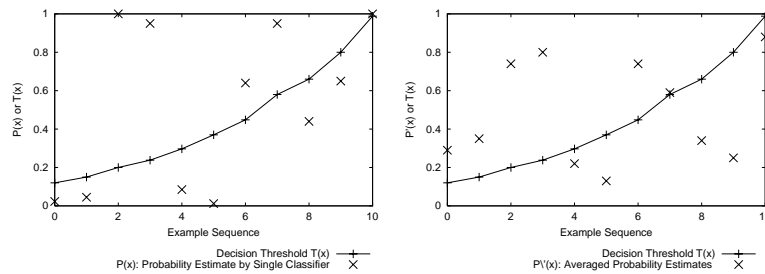


Figure 1. Cost-sensitive decision plots. The left plot is conjectured for a single classifier, while the right plot is conjectured for the averaged probability of multiple classifiers.



Next we discuss several approaches to compute $P(x)$, and also $Y(x)$ if not given.

Recently, Zadrozny and Elkan [16, 17] have proposed several methods to compute probabilities from decision trees and naive Bayesian classifiers. For decision trees, suppose that t is the number of examples and p is the number of positive examples associated with a leaf. *Raw probability* is defined as $\frac{p}{t}$. *Smoothed probability* or *smoothing* is defined as $\frac{p+m \cdot b}{t+m}$, where b is the *base rate* of positive examples in the training set and m is chosen to be 100 as suggested by Zadrozny and Elkan [17]. To compute *curtailed probability* or *curtailment*, we stop searching down the tree if the current node has less than v examples and use the current node to compute raw probabilities. The exact value v is not critical if v is small enough. We have used $v=100$. *Curtailment plus smoothing* computes smoothed probability at the same node of *curtailment*.

For some problems, such as donation and merchandise purchase, the benefit (donation amount and purchase amount) is not known in advance. There are a few methods to estimate this benefit. In this paper, we employ the multiple linear regression method. The dataset to train the benefit estimator uses different feature sets from the one to compute probabilities; otherwise, probability and benefit outputs would be highly correlated.

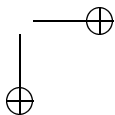
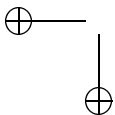
3 Combining Methods

We first discuss a few desiderata of the combining approaches, and then present the details of the methods and scalability analysis.

Desiderata One simple method is to use averages of probabilities and estimates. The obvious advantage is its scalability since no meta-level classifiers are trained. The benefit is also potentially higher as explained below.

The base models trained from disjoint data subsets make uncorrelated noisy errors to estimate probabilities. It is known and studied by Tumer and Ghosh [13] that uncorrelated errors are reduced by averaging. The averaged probability may still be different from the single classifier, but the difference may not make a difference to final prediction due to the error tolerance property (Section 2).

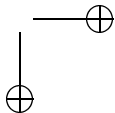
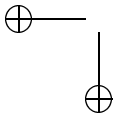
The averaged probability is very likely to have higher benefits because of



its “evening effect” and stronger bias towards predicting expensive examples to be positive. To explain this effect, we use the *cost-sensitive decision plot*. For each data point x , we plot its decision threshold $T(x)$ (Section 2) and probability estimate $P(x)$ in the same figure. The sequence of examples on the x -axis is ordered increasingly by their $T(x)$ value. Figure 1 illustrates two exemplary plots. The left plot is conjectured for single classifier, while the right plot is conjectured for averaged probability of multiple classifiers. All data points above the $T(x)$ line (with $P(x) > T(x)$) are predicted positive. Using this plot, we explain the *evening effect*. Since probability estimates by multiple classifiers are uncorrelated, it is very unlikely for all of them to be close to either 1 or 0 (the extremities) and their resultant average will likely spread more “evenly” between 1 and 0. This is visually illustrated in Figure 1 by comparing the right plot to the left plot. The evening effect favors more towards predicting expensive examples to be positive. $T(x)$ of expensive examples are low; these examples are in the left portion of the decision plots. If the estimated probability by single classifier $P(x)$ is close to 0, it is very likely for the averaged probability $P'(x)$ to be bigger than $P(x)$, and consequently bigger than $T(x)$ of expensive examples and predict them to be positive. The two expensive data points in the bottom left corner of the decision plots are predicted to be negative by the single classifier, however predicted to be positive by the multiple model. Due to the evening effect, averaging of multiple probabilities biases more towards expensive examples than the single classifier. This is a desirable property since expensive examples contribute greatly towards total benefit. Cheaper examples have higher $T(x)$, and they are shown in the right portion of both plots in Figure 1. If single classifier $P(x)$ for a cheap example is close to 1, it is more likely for the averaged probability $P'(x)$ to be lower than $P(x)$, and consequently lower than $T(x)$, and for cheaper examples to be predicted as negative (the example on the upper right corner of the plots). However, cheap examples carry much less benefit than expensive examples; the bias towards expensive examples of averaged probabilities still have potentially higher total benefits than single probability. It is better to catch a fraud of \$10,000 than one with \$100. Although not explicitly explained by the authors, *smoothed probability* ([16] and Section 2) increases the total benefit by the same reason. Based on the above conjectured analysis, we think that averaging will actually increase total benefit.

Tree-structured combining methods have been shown by Chan [1] with the name of *meta-learning* to be effective to combine class labels. We explore the possibility of tree-structured methods to combine probabilities and benefits.

Methods Assume that the dataset is partitioned into k disjoint subsets S_i , and a classifier $C_i(x)$ is trained from S_i . The classifier outputs a member probability for each prediction, $P_i(x)$. For problems for which the benefit of x is not known in advance (such as charitable donation), a separate dataset is used to compute a model (such as linear regression) to estimate this benefit. Similarly, this dataset is partitioned, and a model $Y_i(x)$ is built from each partition. $R_i(x) = P_i(x) \cdot Y_i(x)$ is the individual risk of corresponding base models. We use $\bar{P}(x)$, $\bar{Y}(x)$ and $\bar{R}(x)$ to denote *combined probability*, *combined benefit estimate*, and *combined risk* respectively.



One straightforward approach is to simply average individual risks to compute combined risk,

$$\text{Simple Averaging : } \bar{R}(x) = \frac{\sum R_i(x)}{k} = \frac{\sum P_i(x) \cdot Y_i(x)}{k} \quad (1)$$

Alternatively, we can combine probabilities and estimates separately by average and regression, and then multiply them together. Using simple averages, averaged probabilities and averaged estimates are $\bar{P}_{avg}(x) = \frac{\sum P_i(x)}{k}$ and $\bar{Y}_{avg}(x) = \frac{\sum Y_i(x)}{k}$ respectively. Using regression, $P_i(x)$'s are used as independent variables and true probabilities (1 for positives and 0 for negatives) are dependent variables to compute a regression model, $\bar{P}_{reg}(x) : (P_1(x), \dots, P_k(x)) \rightarrow \{0, 1\}$, where x is taken from the complete training set. Combined estimate is computed similarly using regression, $\bar{Y}_{reg}(x) : (Y_1(x), \dots, Y_k(x)) \rightarrow y(x)$. When k is big, training a regression model with k independent variables from n training instances can be very inefficient. Instead, we generate a binary tree of regression models that resembles a meta-learning tree [1]. Two base models (P_i and P_{i+1}) are combined using regression. A random subset of examples (size $\frac{n}{k}$) is selected to train this regression model. There will be a total of $\frac{k}{2}$ regression models at the first level. These models are combined similarly and this process is repeated to form a tree of regression combining models. Details on meta-learning tree can be found in [1].

Depending on how $\bar{P}(x)$ and $\bar{Y}(x)$ are computed, there are four combinations to compute $\bar{R}(x)$

$$\text{Multiplex Averaging : } \bar{R}(x) = \bar{P}_{avg}(x) \cdot \bar{Y}_{avg}(x) \quad (2)$$

$$\text{Averaging Regression : } \bar{R}(x) = \bar{P}_{avg}(x) \cdot \bar{Y}_{reg}(x) \quad (3)$$

$$\text{Regression Averaging : } \bar{R}(x) = \bar{P}_{reg}(x) \cdot \bar{Y}_{avg}(x) \quad (4)$$

$$\text{Regression Regression : } \bar{R}(x) = \bar{P}_{reg}(x) \cdot \bar{Y}_{reg}(x) \quad (5)$$

We use *averaging* to refer to both *simple averaging* and *multiplex averaging* when the meaning is clear from the contexts. And we use *averaging ensemble* to refer to the ensemble combined using the averaging method. Multiplex averaging (formula (2)) is different from simple averaging (formula (1)) in that each $P_i(x)$ is multiplied with every $Y_j(x)$, then divided by k^2 . The chance for the combined result being dominated by a big $P_i(x) \cdot Y_i(x)$ is much lower than simple averaging.

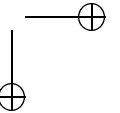
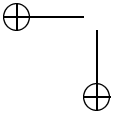
In addition, we use variations of meta-learning [2] to compute probabilities. One of the most effective meta-learning method is *combiner* or *stacking* [15]. It learns the correlation of predicted labels of base classifiers to the true class label. A combiner tree is trained to combine the class label outputs of base classifiers. We manipulate the combiner at the root of the tree to output probabilities, $\bar{P}_{combiner}(x)$.

Depending on how the estimators are combined, we have

$$\text{Combiner Averaging : } \bar{R}(x) = \bar{P}_{combiner}(x) \cdot \bar{Y}_{avg}(x) \quad (6)$$

$$\text{Combiner Regression : } \bar{R}(x) = \bar{P}_{combiner}(x) \cdot \bar{Y}_{reg}(x) \quad (7)$$

If the benefit is given, we don't need to combine estimated benefits, thus there



are three unique methods to combine probabilities, *Averaging*: $\bar{P}_{avg}(x)$, *Regression*: $\bar{P}_{reg}(x)$ and *Combiner*: $\bar{P}_{combiner}(x)$.

Averaging (simple averaging and multiplex averaging) doesn't train any meta-level classifier to combine base models, therefore it is efficient and scalable. It also provides natural incremental learning; when new data increment becomes available, we simply learn a new model and remove the oldest model in the ensemble. The training overhead and speedup to learn a tree of classifiers (regression and combiner tree model) is analyzed in [1]. Next we analyze the speedup and scalability of the averaging methods.

Speedup and Scalability Analyses The ensembles are designed to be utilized in a parallel and distributed processing environment. Here, we analyze and evaluate the speedup and scalability when the ensembles are trained either sequentially or concurrently and averaging is used to combine the ensemble. Following the analysis of meta-learning by Chan [1], we use the following notations.

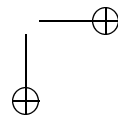
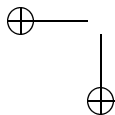
- n is input size or the number of examples.
- k is the number of data partitions which is the same as the number of processors if the ensemble is trained in parallel.
- T_b is the execution time of the pure sequential algorithm to train the single classifier on the entire dataset. In our case, it is the training time to learn on the complete dataset, denoted by $c \cdot O(f(n))$. We assume that the original dataset can fit into the main memory; otherwise the training time will be much longer. For simplicity of analysis, we drop the constant c and just use $O(f(n))$.
- T_p is the parallel execution time of the scalable algorithm to compute the ensemble of classifiers. It is the time to train all k base models from k partitions parallelly on k processors, which is $O(f(\frac{n}{k}))$.
- T_s is the serial execution time of the scalable algorithm. In our case, it is the training time to train each base models individually on the same computer, which is $O(k \times f(\frac{n}{k}))$.
- W is the problem size (work) [8], which measures the total number of computational units needed for serial execution. That is, $T_s = W \times t_u$, where t_u is the time spent for a unit of computation.

There are two metrics that many people use to measure the improvement of the scalable algorithm to the pure serial algorithm, or the improvement of the averaging ensemble over learning a monolithic classifier from the complete dataset.

- *Parallel Improvement*: it is the number of times that parallel training of the averaging ensemble is faster than training a single model on the complete dataset, therefore defined as

$$Parallel\ Improvement = \frac{T_b}{T_p} = O\left(\frac{f(n)}{f(\frac{n}{k})}\right) \quad (8)$$

- *Serial Improvement*: it is the number of times serial training of the same averaging ensemble is faster over learning a single model on the complete



dataset. Therefore, it is defined as

$$\text{Serial Improvement} = \frac{T_b}{T_s} = O\left(\frac{f(n)}{k \times f(\frac{n}{k})}\right) \quad (9)$$

To analyze the scalability of the scalable algorithm, it is customary to use the *speedup* metric.

- *Speedup* is the number of times parallel execution is faster over serial execution with a fixed problem size of the same scalable algorithm. For the averaging ensemble approach, it is

$$\text{Speedup} = \frac{T_s}{T_p} = O\left(\frac{k \times f(\frac{n}{k})}{f(\frac{n}{k})}\right) = O(k). \quad (10)$$

A linear speedup of $O(k)$ regardless of the learning algorithm and input size is the best speedup achievable.

- *Scaled speedup* [7, 9] provides a metric for scalability. It measures the speedup of a parallel system when the problem size increases linearly with the number of processors. This metric describes the dependence of speedup on the problem size.

$$\text{Scaled Speedup} = \frac{T_s(W \times k)}{T_p(W \times k)} \quad (11)$$

where T_s and T_p are expressed as functions of problem size. The averaging ensemble always divides the dataset into k equal parts and the speedup is always $O(k)$ that is independent of the size of the training data or problem size. Therefore, the scaled speedup is the same as speedup, which is $O(k)$. A linear scaled speedup is the best scalability possible. Parallel systems with linear or near-linear scaled speedup with respect to k is considered *scalable*.

From formulas (8) and (9), both parallel and serial improvements depend on the learning algorithm. Without losing generality, we choose a popular decision tree algorithm C4.5 to analyze. The complexity of C4.5 is roughly $O(n \cdot \log(n))$ for continuous feature sets and $O(n)$ for discrete feature sets. The extra $\log(n)$ for continuous feature sets is due to sorting. By replacing $f(n)$ with $O(n \cdot \log(n))$ in formula (8), parallel improvement becomes,

$$\text{C4.5 Parallel Improvement} = O\left(k \times \frac{\log(n)}{\log(n) - \log(k)}\right) \quad (12)$$

This is superlinear and close to linear for relative small numbers of k . Practically, k is much smaller than n . By replacing $f(n)$ in formula (9), serial improvement becomes

$$\text{C4.5 Serial Improvement} = O\left(\frac{\log(n)}{\log(n) - \log(k)}\right) \quad (13)$$

It is sublinear and close to linear for realistic numbers of k . In Figure 2, we plot the above two improvements using $n=1000$ and $1 \leq k \leq 200$.

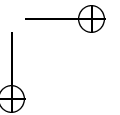
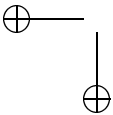
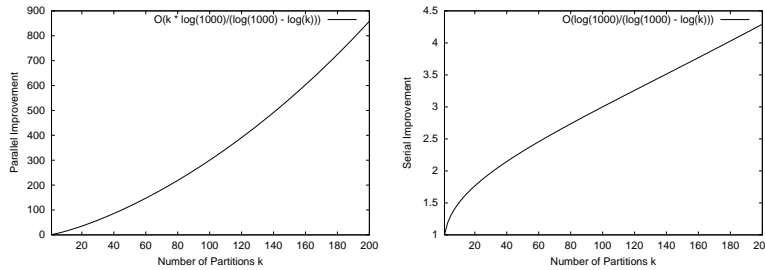


Figure 2. Plot of theoretical parallel and serial improvements of *C4.5* using averaging ensemble over training a single model on the complete dataset



Based on the above analysis, the averaging ensemble is indeed very scalable in theory. In the next Section, we will empirically evaluate its effectiveness to gain total benefit and scale up learning. We follow the empirical study with a discussion why ensemble might work.

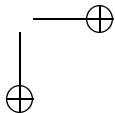
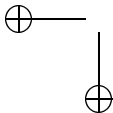
4 Experimental Setup

We have evaluated the ensemble approaches on three types of cost-sensitive problems. They differ in how the profit of x is obtained and if there is any penalty for misclassification besides losing the profit. We discuss each dataset and different measurements in our experiments. This is followed by an analysis of the experimental results.

Dataset In the first problem (donation), neither the probability $P(x)$ nor the benefit $y(x)$ is known. Additionally, only positive example carries profit and there is a penalty for false positives. We use the donation dataset that first appeared in KDD'98 competition. Suppose that the cost of requesting a charitable donation from an individual x is 0.68, and the best estimate of the amount that x will donate is $Y(x)$ (the estimated benefit of x). In this case, the optimal decision is to solicit x if and only if $(R(x) = P(\text{donate}|x)Y(x)) > 0.68$. Detailed description of this dataset can be found in [16]. To estimate the donation amount, we use the multiple linear regression method. The base donation rate is $b = 0.05$, which is used for smoothing.

In the second problem (credit card fraud), the benefit is known and it is *per instance*. Only positives carry benefit and there is also a penalty for false positives. We use a credit card fraud detection dataset as explained in Section 2. The data was sampled from a one year period and contains a total of 5M transaction records. The features record the time of the transaction, merchant type, merchant location, and past payment and transaction history summary. We use data of the last month as test data (40038 examples) and data of previous months as training data (406009 examples). The base rate of positives in this dataset is $b = 0.2$.

In the third problem (adult), the benefit is known and it is *per class* (as opposed to *per instance*). Unlike the first two problems, both positives and negatives carry benefits. Additionally, besides losing the benefit, there is no additional penalty



for misclassification as opposed to the previous two cases. We use the adult dataset from UCI repository and artificially associate a benefit of 2 to positive class and a benefit of 1 to negative class for correct prediction. The best decision is to predict *positive* when $(P(\text{positive}|x) \cdot 2) > ((1 - P(\text{positive}|x)) \cdot 1)$. We use the natural split of training and test sets, so the results can be easily duplicated. The training set contains 32561 entries and the test set contains 16281 records with base positive rate of around $b = 0.24$.

Experimental Design We use total benefit to compare different methods. It is the total profit obtained to solicit x being positive according to the optimal decision policy. For the donation dataset, this is the total donation amount minus the cost of campaign letters. For the credit card fraud detection dataset, it is the total correctly detected fraudulent transaction amount minus the overhead of investigation. And for the adult dataset, this is the total profits of correct predictions.

One important aspect of our experiments is to test that the scalable methods' total benefit is independent of the number of partitions k ; this important feature will enable us to run on arbitrarily large datasets using arbitrary number of partitions k to fit each partition into memory. We have chosen to use the following number of partitions, $k = 8, 16, 32, 64, 128$ and 256 .

We used the C4.5 decision tree learning algorithm without pruning. We obtained the source code of release 8 and modified it to output probabilities. We measured the training time of C4.5 on each partition and the complete dataset to measure the empirical scalability of the ensemble.

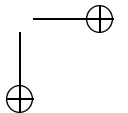
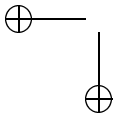
5 Experimental Result

We first study how the four probability approaches work for three datasets. The total benefit serves as baseline to compare with the ensembles. We then find out if the various combining methods work and how the benefits are influenced by the number of partitions. Finally, we find out the empirical efficiency to train the ensemble.

Dataset	raw	smoothing	curtailment	curtailment plus smoothing
Donation	12489.6	14291.2	13292.7	14424.9
Credit Card Fraud	552400	765009	733980	782545
Adult	16255	<u>15392</u>	16443	<u>15925</u>

Table 1. *Baseline total benefits attained by various probability computation methods. The best method for each dataset is highlighted on bold font. If a result is worse than raw probability, it will be underscored*

Total Benefits The baseline results are shown in Table 1. The donation and credit card fraud datasets verify the discoveries of Zadrozny and Elkan in that smoothing, curtailment and curtailment plus smoothing are better than raw C4.5 score, and curtailment plus smoothing is the most effective method. However, in adult dataset, both smoothing and curtailment plus smoothing are much worse than raw probability. Smoothing squashes the probability towards base probability ($b = 0.24$ for adult). For some node with small number of examples and a raw



Donation				
	raw	smoothing	curtailment	curtailment plus smoothing
Simple Averaging	13842.6	<u>12974.7</u>	14643.3	<u>13742.1</u>
Multiplex Averaging	13875.8	<u>13000.4</u>	14702.9	<u>13752.2</u>
Averaging Regression	13790.7	<u>12879.6</u>	14597.5	<u>13680.3</u>
Regression Averaging	13640.4	<u>12780.7</u>	14620.5	<u>13605.4</u>
Regression Regression	13545.4	<u>12604.7</u>	14512.2	<u>13534.4</u>
Combiner Averaging	13420.4	<u>12442.4</u>	14310.3	<u>13355.6</u>
Combiner Regression	13200.7	<u>12454.7</u>	14207.5	<u>13271.2</u>

Credit Card Fraud					Adult			
	rw	smth	ctl	ctlsmt h	rw	smth	ctl	ctlsmt h
Averaging	808438	<u>712112</u>	804964	<u>784009</u>	16235	<u>14022</u>	16435	<u>15145</u>
Regression	729384	<u>681254</u>	784310	<u>779353</u>	15910	<u>13932</u>	16021	<u>14992</u>
Combiner	738294	<u>661725</u>	789845	<u>768784</u>	15845	<u>13864</u>	15945	<u>14745</u>

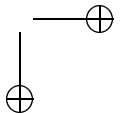
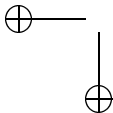
Table 2. Average total benefits of ensemble methods over different numbers of partitions. If the result is worse than the respective baseline in Table 1, it will be underscored. Results in bold font is the best result for that dataset for both the single classifier and ensemble approaches

probability > 0.24 , the smoothed probability will be very close to 0.24. For example, suppose a node has a raw probability of 0.4 and smoothed probability of 0.24. At a raw probability of 0.4, all the examples covered by this node will be predicted being positive since $2 \times 0.4 > 0.6$, which is the optimal solution. However, after smoothing, all the examples will be predicted negative instead since $2 \times 0.24 < 0.6$. This problem won't be serious if each example has its per instance benefit factor or the benefit ratio from positives over negatives is very big.

In order to compare different combining methods, we calculate the average of total benefits of each combining methods over different numbers of partitions. The results are shown in Table 2. Since the benefit $y(x)$ is known for both credit card fraud and adult datasets, there is no need to estimate it, thus we only need to combine probabilities using *Averaging*, *Regression* and *Combiner*.

First we have found that the different combining methods all perform reasonably well and there is no big difference for different combining methods. The averaging methods consistently beat the more complicated combiner and regression methods across three datasets and with all four different ways to compute probability. This means that although remarkably simple, averaging (both simple and multiplex averaging) is a very accurate method to combine classifiers, and using more complicated and expensive methods don't bring extra benefits. In Section 3, we explain why the averaging ensemble works well.

We have also observed that smoothing is consistently worse than the baseline smoothing result (single model trained on all the dataset), even curtailment plus smoothing is worse than curtailment by itself. The reason is that each data partition is much smaller than the original dataset, consequently, the average number of examples covered by a leaf node in the base classifier is significantly less than that of the single classifier. Using the same parameters (m and b), smoothing squashes



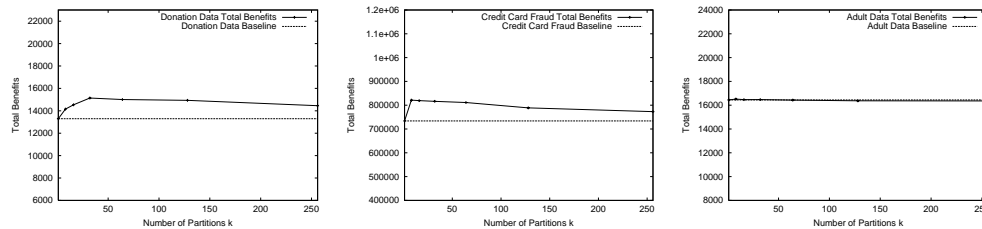
the probabilities of base classifiers more towards b . In effect, it over-estimates or under-estimates the probabilities. Nonetheless, the best total benefits attained for both single classifier and different combining methods are all by averaging methods as highlighted in bold font in Table 2. In general, curtailment averaging is the most effective approach.

In Table 3 and Figure 3, we plot changes of total benefits (using curtailment averaging¹) with growing number of partitions k using the curtailed probabilities. We can clearly see the total benefits by ensemble methods are all higher than the baseline for all chosen k for both the donation and credit card fraud dataset. For the donation dataset, the results for $k = 32$ and $k = 64$, 15141 and 15004 respectively, are better than the winning entry of KDD98 cup, which was 14712. For the adult dataset, the ensemble method's total benefit is higher than that of the baseline with $k < 64$, and slightly worse than but still close enough to the baseline when $k \geq 64$. When k increase, there is a very slow decrease in total benefits, but the tendency is very slow. At $k = 256$, the total benefit still remains higher than that of the single classifier for donation and credit card fraud datasets, and close enough to the baseline for adult (16359 vs. 16443), but the training cost is much less (explained in the next section). It is important to mention that the training set size of adult (32561) is 3 times smaller than donation dataset (95412) and 13 times smaller than credit card fraud dataset (406009).

	baseline	k=8	16	32	64	128	256
Donation	13292.70	14151.20	14534.90	15141.20	15004.00	14932.30	14453.50
Credit Card	733980	821311	819155	816244	811282	788795	772995
Adult	16443	16513	16470	16471	16428	16370	16359

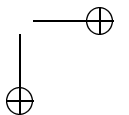
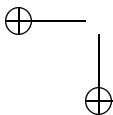
Table 3. Changes in total benefits relative to the number of partitions k using averaged curtailed probabilities

Figure 3. Plot of changes in total benefits as shown in Table 3



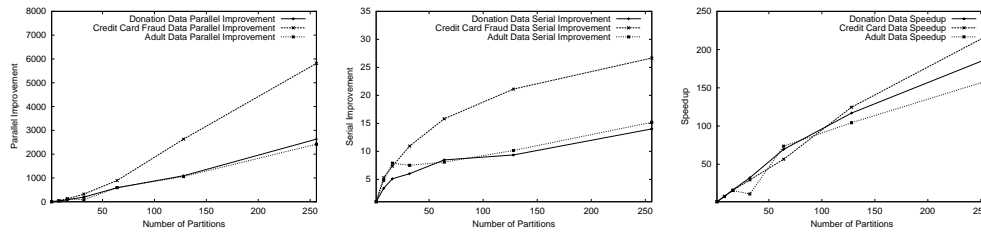
Training Efficiency In Figure 4, we plot serial improvement (formula (9)), parallel improvement (formula(8)) and speedup (formula(10)) in relation to the number of

¹The results for curtailment simple averaging and multiplex averaging for donation dataset are very close for the donation dataset, therefore, we choose to present multiplex averaging result.



partitions k . All training time to compute these measures is just to generate the unpruned C4.5 decision tree, all data partitions and original data can be held in main memory and there is no other jobs running at the same time. As shown in the first plot of Figure 4, the parallel improvement for all three datasets are very similar in shape. This is due to the fact that all three datasets have continuous features and the complexity of C4.5 becomes $O(n \cdot \log(n))$. Comparing these empirical parallel improvement plots with the theoretical parallel improvement plot (the left one in Figure 2), they are very close in shape, verifying that theoretical analysis and experimental results match very well. For credit card fraud dataset, when the number of partitions is 256, the parallel improvement is as high as 5800. This means that by using 255 more computing resources, we can in return get an improvement of 5800 in training time. The second plot of Figure 4 shows the serial improvement. Like the parallel improvements, the shape of all three datasets are similar and closely match the theoretical analysis plot (right plot of Figure 2) . For credit card fraud dataset, simply by dividing the data set into 256 partitions and learning models one by one on the same computer, we will have an improvement in running time by as much as 26 times. The last plot of Figure 4 displays speedup in our experiments. All three curves are close to the theoretical result, $O(k)$. The disturbances in the adult curve is due to its relatively small training set size as compared to donation and credit card fraud datasets. A linear speedup of $O(k)$ means that by using k times more computer to run the same ensemble approach concurrently, we will expect a speedup of roughly k not only in theory but also in practice.

Figure 4. *Serial improvement, parallel improvement and speedup relative to the number of partitions k*



We also measured *scaled speedup* by varying the input size. The detailed results (not shown) have verified that it is linear and independent of the input size.

Explanation We explained the reason in Section 3 why averaging may have potentially higher benefits. The empirical results verified our conjecture. As an example, in Figure 5, we plot decision plots (as defined in Section 3) for the credit card fraud dataset. We choose raw probability estimates and $k = 256$ for the averaging ensemble. The number on each plot shows the number of examples² whose $P(x) > T(x)$

²To show these numbers clearly on the plot, we don't plot the surrounding data points around the text area

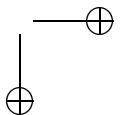
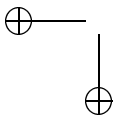


Figure 5. *Decision threshold and probability output by single model and 256-classifier ensemble for credit card dataset*

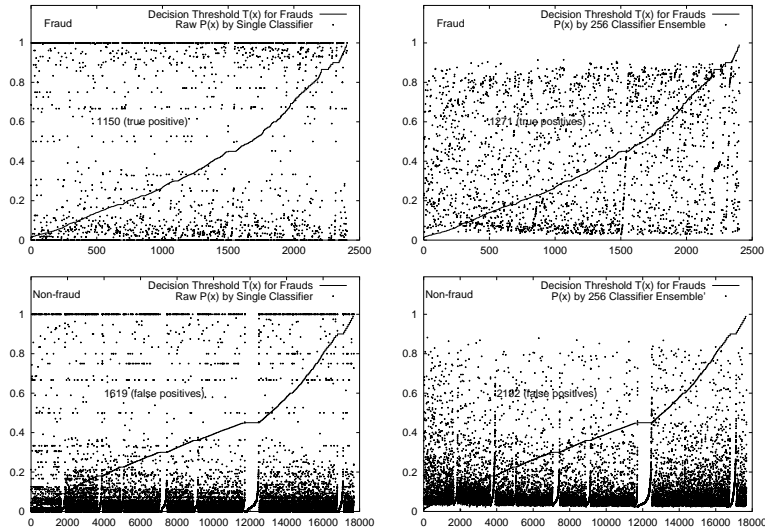
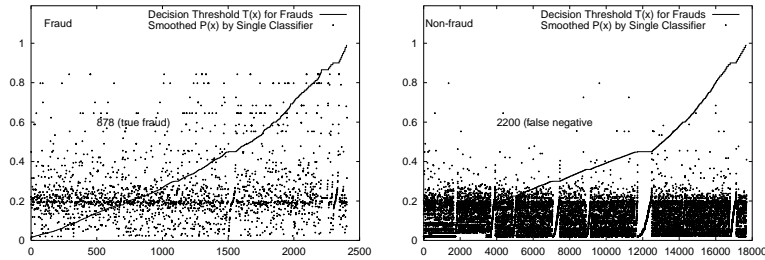
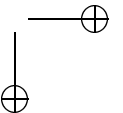
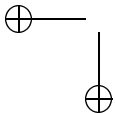


Figure 6. *Decision threshold and smoothed probability output by single model on credit card dataset*



(predicted as frauds). The top two plots are fraudulent transactions and those bottom plots are non-fraud. The overall effect of the averaging ensemble increases the number of true positives from 1150 to 1271 and the number of false positives from 1619 to 2192. However the average transaction amount of the “extra number” of detected frauds by the ensemble ($121=1271 - 1150$) is around \$2400, which greatly overcomes the cost of extra false alarm (\$90 per false alarm). For problems like credit card fraud, donation and catalog mailing where positive examples have var-



ied profits and negative examples have low or fixed cost, the ensemble methods tend to beat the single model.

We applied the same analysis to compare *raw* and *smoothing* (proposed in [16]) trained from the entire dataset as single classifier and could clearly observe the same phenomenon to explain why smoothing works (as shown in Figure 6). Smoothing moves the probability more aggressively to the base rate $b = 0.2$ (Section 2). The effect is that more expensive transactions (frauds and non-frauds) are detected as frauds.

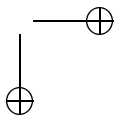
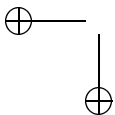
6 Related Work

The most noteworthy work to scale up learning to combine class labels is meta-learning [1]. Meta-learning has been implemented as JAM [10]. It is the first known system to use ensemble of classifiers to combine class labels to scale up learning over large and distributed datasets. Other incomplete list of significant contributions to scalable learning (mostly are on learning class labels or cost-insensitive learning) are scalable decision tree learner SPRINT [12], rule-based algorithm for multi-processor learning DRL [11], incremental decision tree learner ID5 [14] among many others.

7 Conclusion

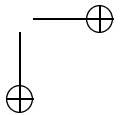
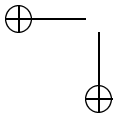
We have proposed and evaluated several ensemble approaches to combine probabilities and benefits for cost-sensitive scalable learning. We identify some properties of both averaging and optimal decision-making. Due to these properties, averaging offers higher benefit in addition to its obvious advantage of scalability. We propose two methods that use averages of probabilities and benefits. We analyzed their speedup and scalability. Since tree-structured combining methods have been shown previously to be effective to combine class labels, we explore tree-structured regression and variations of meta-learning to combine probabilities and benefits. We have chosen three different types of cost-sensitive problems for empirical study.

We have found that the averaging and tree-structured methods achieve total benefit that is as good as or even better than the single classifier trained on the entire dataset. The most accurate approaches are the two averaging methods, whose total benefit top among all the ensemble approaches and even consistently beat the single classifier. They also exhibit linear speedup and scaled speedup relative to the number of partitions k and independent of the learning algorithm. When using C4.5, the parallel improvement of averaging over the single classifier is superlinear relative to k . When training the same ensemble serially, the improvement in training time is sublinear but close to linear relative to k . The improvement in training time will be even more if the entire dataset cannot fit into main memory. Our study have also shown that more complicated tree-structured methods offer neither higher benefit nor better scalability. Compared with tree-structured methods, the averaging methods are more efficient to classify examples due to its single level of base classifiers, and they are straightforward and easy to implement and verify. Based on our extensive experimental and analytical study of many candidate combining methods, we find that the averaging approaches are an effective framework for scalable cost-sensitive learning.



Bibliography

- [1] P Chan. *An Extensible Meta-learning Approach for Scalable and Accurate Inductive Learning*. PhD thesis, Columbia University, Oct 1996.
- [2] P Chan and S Stolfo. Sharing learned models among remote database partitions by meta-learning. In *Proceedings of Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, Portland, Oregon, 1996.
- [3] P Chan and S Stolfo. Towards scalable learning with non-uniform class and cost distributions: a case study in credit card fraud detection. In *Proceedings of Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 164–168, 1998.
- [4] T Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning*, 40(3):139–159, 2000.
- [5] T Dietterich, D Margineatu, F Provost, and P Turney, editors. *Cost-Sensitive Learning Workshop (ICML-00)*, 2000.
- [6] W Fan. *Cost-sensitive, Scalable and Adaptive Learning Using Ensemble-based Methods*. PhD thesis, Columbia University, Dec 2000.
- [7] J. Gustafson. Reevaluating Amdahl’s law. *Comm. ACM*, 31(5):532–533, 1988.
- [8] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to parallel computing: Design and analysis of algorithms*. Benjamin-Cummings, Redwood City, CA, 1994.
- [9] V. Kumar and A. Gupta. Analyzing scalability of parallel algorithms and architectures. *J. Parallel & Distributed Computing*, 22:379–391, 1994.
- [10] Andreas Prodromidis. *Management of Intelligent Learning Agents in Distributed Data Mining Systems*. PhD thesis, Columbia University, Oct 1999.
- [11] F J Provost and D Hennesy. Scaling up: Distributed machine learning with cooperation. In *Proceedings of Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, 1996.



- [12] J Shafer, Ramesh Agrawl, and M Mehta. SPRINT: A scalable parallel classifier for data mining. In *Proceedings of Twenty-second International Conference on Very Large Databases (VLDB-96)*, pages 544–555, San Francisco, California, 1996. Morgan Kaufmann.
- [13] K Tumer and J Ghosh. Theoretical foundations of linear and order statistics combiner for neural pattern classifiers. Technical Report TR-95-02-98, The Computer and Vision Research Center, The University of Texas at Austin, 1995.
- [14] Paul Utgoff. An improved algorithm for incremental induction of decision. In *Proceedings of Eleventh International Conference on Machine Learning (ICML-94)*, pages 318–325, 1994.
- [15] D Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.
- [16] B Zadrozny and C Elkan. Learning and making decisions when costs and probabilities are both unknown. In *Proceedings of the Seventh ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD01)*, 2001.
- [17] B Zadrozny and C Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proceedings of Eighteenth International Conference on Machine Learning (ICML'2001)*, 2001.

