

# A Fully Distributed Framework for Cost-Sensitive Data Mining

Wei Fan<sup>1</sup>, Haixun Wang<sup>1</sup>, Philip S. Yu<sup>1</sup>, and Salvatore J. Stolfo<sup>2</sup>

<sup>1</sup>IBM T.J.Watson Research Center, 30 Saw Mill River Road, Hawthorne, NY 10532

{weifan, haixun, psyu}@us.ibm.com

<sup>2</sup>Department of Computer Science, Columbia University, New York, NY 10027

sal@cs.columbia.edu

## Abstract

Data mining systems aim to discover patterns and extract useful information from facts recorded in databases. A widely adopted approach is to apply machine learning algorithms to compute descriptive models or classifiers from the available data. Two of the main challenges in this area are that i) databases are large and possibly physically distributed, and ii) data are cost-sensitive, or examples in the databases usually have different prices or benefits (such as charity donation amount) that require an effective model to be more accurate towards examples with higher benefits. Here, we explore the development of techniques that address both issues to scale up cost-sensitive data mining. One naive approach for distributed data mining is a centralized system that ships all available data from different sites onto a single site to learn a global model. Besides its obvious communication overhead, this approach is ineffective due to many practical concerns. The second approach is a partially distributed system that coordinates learning among multiple sites by exchanging both temporarily learned models and limited amount of data. Their control mechanisms are usually complicated, which reduces its scalability, ease of use and verifiability. In this paper, we propose a fully distributed system where each site computes its models locally without interacting with any other sites. The probability outputs of different local models are coalesced together by heuristics to produce the final output. Our framework also addresses other important issues such as heterogenous platforms, multiple schemas, multiple learning algorithms, incrementability, security, and fault-tolerance. We evaluate our framework through extensive empirical studies. Our results have shown that the fully distributed learning approach achieves higher accuracy than both the centralized and partially distributed learning methods, however, it incurs much less training time, neither communication nor computation overhead.

**Keywords:** cost-sensitive, distributed, ensemble, probability

**Area:** Distributed Algorithm, Distributed Databases

# 1 Introduction

During the last two decades, our ability to collect and store data has significantly out-paced our ability to analyze, summarize and extract “knowledge” from the continuous stream of input. A short list of examples is probably enough to place the current situation into perspective:

- A typical credit card company processes millions of new transactions on a daily basis.
- NASA’s Earth Observing System (EOS) of orbiting satellites and other space-borne instruments send one terabyte of data to receiving stations every day.
- A typical affiliate marketing company records nearly 1 billion click-through data on a normal business day.

Traditional data analysis methods that require humans to process large datasets are completely inadequate and to quote John Naisbett, “We are drowning in information but starving for knowledge.”

*Data Mining* is the complex process of identifying valid, novel, potentially useful and ultimately understandable patterns in data. In a relational database context, a typical data mining task is to explain and predict the value of some attribute given a collection of tuples with known attribute values. One means of performing such a task is to employ various machine learning algorithms. Given a set of training examples of the form,  $\mathcal{S} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ , for some unknown function  $y = f(\mathbf{x})$ , the learning task is to compute a *classifier* that approximates the true function to correctly label any examples drawn from the same source as the training set. We denote classifiers by  $C_1, \dots, C_L$ . The  $\mathbf{x}_i$  values are feature vectors whose components are discrete- or real-values, for example, the amount of purchase, merchant category, location of the merchant, its distance from the billing address, payment history information, frequency of purchase, etc.  $y_i$  is called the *class label*, and it is drawn from a finite set, such as  $\{fraud, nonfraud\}$ .

One of the main challenges in machine learning and data mining is the development of inductive learning techniques that scale up to large and possibly physically distributed datasets. Many organizations seeking

added values from their data are already dealing with overwhelming amounts of information. The number and size of their databases and data warehouses grows at phenomenal rates, faster than the corresponding improvements in machine resources and inductive learning techniques. Most of the current generation of learning algorithms are computationally complex and require all data to be resident in main memory which is clearly untenable for many realistic problems and databases. Furthermore, in many cases, data may be inherently distributed and cannot be localized on any one machine (even by a trusted third party) for a variety of practical reasons including security and fault tolerant distribution of data and services, competitive (business) reasons, statutory constraints imposed by law as well as physically dispersed databases or mobile platforms like an armada of ships. In such situation, it may not be possible, nor feasible, to inspect all of the data at one centralized processing site to compute one primary “global” classifier.

Lately, *cost-sensitive* learning has been an area of extensive research interests [2]. In many areas of application where different examples carry different benefits, it is not enough to maximize the accuracy based on the simplified *cost-insensitive* 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. Charity donation seeks to maximize the total benefit made up of the sum of all donated charities minus any costs to send campaign letters to potential donors. Similarly, catalog mailing merchants maximize the total benefit comprised of the profit minus the cost of mailing catalogs to all potential buyers.

In this paper, we are interested in studying accurate and efficient frameworks for distributed cost-sensitive learning.

## 1.1 Our contribution

We develop a framework to perform distributed cost-sensitive learning using ensemble of classifiers, and maintain similar or even better benefit than the centralized system that trains a single “global” model by shipping all data from each site onto a single site. Base classifiers or models are trained from each distributed data subset. To classify an unknown instance, the probability and benefit outputs of multiple base classifiers are combined using various heuristics to produce the final prediction. We propose and evaluate two types of distributed learning systems, *fully distributed* and *partially distributed*, using two different ways of combining.

In the fully distributed approach, each distributed site computes its model locally without any interaction (neither model nor data exchange) with any other sites; therefore, the method has “zero” communication and computation overhead. To classify an unknown datum, the probabilistic prediction by multiple models are combined together using variations of *averaging*. We identify a few properties of both averaging and cost-sensitive *optimal decision-making*. Due to these properties, fully distributed averaging ensemble has good potential of higher benefit in addition to its obvious advantage of high scalability and zero communication and computation overhead. Our study has shown that the fully distributed averaging ensemble (of probabilities and benefits of base classifiers) gain excellent benefit while avoiding both computation and communication overhead.

In the partially distributed approach, learning among multiple sites are coordinated and synchronized. Both models and limited amount of data are exchanged among participating sites. Tree-structured methods or *meta-learning* [1] have been demonstrated to be accurate to combine class labels (i.e., *fraud* and *nonfraud*) for *cost-insensitive* distributed learning, or each example has the same cost and there is no additional penalty for misclassification. It learns the correlation of base models’ predicted class labels to the true class label. The tree is built bottom up; it recursively combines a few models at a time until a tree is constructed. Both models and data are exchanged among multiple sites, and all participating sites have to be coordinated and

synchronized. One form of tree-structured methods have quadratic communication overhead. We propose partially distributed learning methods that use variations of meta-learning. Our studies have found that these more complicated methods yield neither higher benefit (or accuracy) nor less overhead than the simpler fully distributed averaging ensemble. *The conclusion of our study is that the fully distributed approaches based on averaging is an accurate framework for distributed and cost-sensitive data mining and it has neither communication nor computation overhead.*

The high accuracy and zero overhead of the fully distributed approach via averaging ensemble will solve a lot of practical problems. The intrinsic parallelism of multiple sites can be fully used. Since there is no communication overhead, the total training time is only bounded by the slowest site. For applications where reliable connection is impossible (such as military and mobile computing), participating parties cannot be interconnected due to security and statutory reasons (such as different banks), the fully distributed averaging ensemble is appropriate. Since the number of classifiers to coalesce is not predefined and can be changed at any time without any further computation, it provides both robustness and fault-tolerance. If one site is down or decides not to participate, the other sites won't be influenced and the fully distributed learning proceeds without interruption.

## **1.2 Difference from previous work**

In previous work [3], we have evaluated similar methods for scalable learning of very large dataset on a single site. The focus of this paper is on distributed cost-sensitive learning, where the data physically reside on different machines across the network. There are several distinctive concerns in distributed data mining; these include communication overhead, computation overhead, unbalanced dataset size, fault-tolerance, security, heterogeneous platforms, and incremental learning.

### 1.3 Organization

The rest of the paper proceeds as follows. In Section 2, we discuss the optional decision-making policy and two of its important properties that are used to design the distributed learning algorithms. We also review the probability and benefit calculation methods. We discuss the details of the new methods in Section 3. We also analyze the overhead of these approaches in Section 3.2. The experimental design and results are presented in Section 4. In Section 5, we discuss some important issues of distributed cost-sensitive learning. Related work are reviewed in Section 6. The paper ends with a concluding remark and future work in Section 7.

## 2 Background on Cost-sensitive Decision Making

We discuss some principles of cost-sensitive learning that are necessary to design an effective distributed cost-sensitive learning algorithm. 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  $P(x)$  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 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 3.3).

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

## 2.1 Computing Probabilities and Benefits

We discuss how to compute probabilities from *decision trees*, one of the most commonly used inductive learning algorithms. Decision tree algorithms build a tree of *feature tests*, such as “if gender=male” and “if salary < 90”. Each feature test at a node of the tree splits the data going through the node into two subsets according to the result of the test. A prediction is made when the data at a node has only one single class, such as *fraud*, or cannot be distinguished further in which case the majority class will be predicted. Two of the most effective methods to compute probabilities as proposed by Zadrosny and Elkan [12] include *raw probability* and *curtailed probability*. Suppose that  $t$  is the number of examples and  $p$  is the number of positive examples associated with a node. *Raw probability* is defined as  $\frac{p}{t}$ . 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  $v=100$ . Similar probability computing methods can be applied to Naive Bayes Classifiers and Decision Rules.

For some problems, the benefit for each record  $x$  is given. For credit card fraud detection, the transaction amount of frauds is the benefit for correct prediction. For other 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.

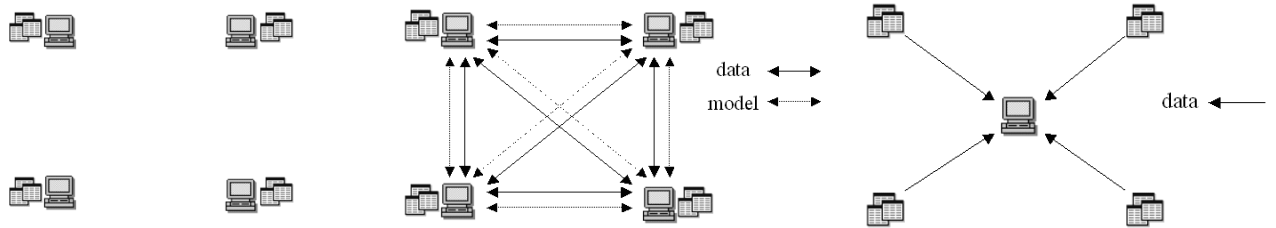


Figure 1: (a) Fully Distributed (b) Partially Distributed (c) Centralized.

### 3 Distributed Learning Systems

We first discuss fully distributed and partially distributed approaches, their communication overhead and explain why they may work.

#### 3.1 Methods

Assume that there are  $k$  participating distributed sites and the data subset at each site is denoted as  $S_i$ , and a classifier  $C_i$  is trained from  $S_i$ . The classifier outputs a class label  $C_i$  (e.g. *fraud* and *nonfraud*) and a member probability  $P_i(x) (\in [0, 1])$  for each testing example  $x$ . 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. In this section, we propose several methods on how to combine the individual probabilities, risks or class labels to compute a global or combined risk. We use  $\bar{P}(x)$ ,  $\bar{Y}(x)$  and  $\bar{R}(x)$  to donate *combined probability*, *combined benefit estimate*, and *combined risk* respectively.

**Fully Distributed** As shown in Figure 1(a), each site computes its model locally without interaction with any other sites, i.e., neither data nor model exchange. Consequently, it incurs neither communication nor computation overhead.

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 averaging 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.

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

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.

**Partially Distributed Learning** In partially distributed learning as shown in Figure 1(b), learning is distributed, coordinated and synchronized among several sites. Both models and limited amount of data are exchanged among multiple sites. Typically, it proceeds in the following steps.

1. Each site computes its model locally.
2. Each site broadcasts its local model to all other sites.
3. Each site predicts with all models on a subset of its local data.
4. Subset of the predictions and true values are broadcast to either all or a few sites.
5. Either all or a few sites computes new models locally.
6. Iterate until a stopping condition is met.

Partially distributed algorithm usually incurs some amount of both communication and computation overhead. Previously, tree-structured method or *meta-learning* [1] has been demonstrated to be effective to

combine *class labels* (such as *fraud* and *nonfraud*) for *cost-insensitive* problems. Here, we borrow similar ideas for *cost-sensitive* problems. Meta-learning constructs a tree of classifiers. Let us use a binary tree as an example. In the first level of the tree, each base classifier will be randomly paired with another base classifier. For  $k$  distributed sites, there will be  $\frac{k}{2}$  pairs in total. Each pair of base classifiers will be combined using some method and a combining classifier will be computed to combine a pair of base classifiers. In order to produce the data to train each combining classifier, all base classifiers are exchanged among  $k$  sites, and a subset of predicted labels and true labels from all training data on all distributed sites is chosen and transferred to one single site<sup>1</sup>. The random selection of examples is to avoid possibly un-uniform distribution of data among  $k$  sites. After generating the first level of combining classifiers, the resultant  $\frac{k}{2}$  combining classifiers are combined in the same manner. This process proceeds recursively until a tree of combining classifiers with depth  $\log_2(k)$  is constructed.

One of the most effective meta-learning methods is *combiner* or *stacking* [11]. It learns the correlation of predicted labels (such as *fraud* and *nonfraud*) of base classifiers to the true class label of the same example. The training set to generate a combiner contains the predicted labels by two base classifiers and the true label of the same example  $x$ , such as  $((C_1(x), C_2(x)), \text{label}(x))$  or  $((\text{fraud}, \text{nonfraud}), \text{fraud})$  for one particular example  $x$ . The original proposal of meta-learning outputs *class labels* (*fraud* and *nonfraud*) at the root of the tree. We manipulate the root of the combiner tree to output probabilities  $\bar{P}_{combiner}(x) (\in [0, 1])$  using the method described in Section 2.1.

The second method is to use regression to directly combine probabilities instead of class labels.  $P_i(x)$ 's (the probability output of base classifiers) are interpreted as independent variables and true probabilities (1 for positives and 0 for negatives) of the same examples are used as dependent variables to compute a regression model,  $\bar{P}_{reg}(x) : (P_1(x), \dots, P_{k'}(x)) \rightarrow \{0, 1\}$ . The examples  $x$  are chosen from the complete training set. Benefit estimates by base models can be combined 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 can be very inefficient.

---

<sup>1</sup>This is the most accurate and simplest way of meta-learning. In his thesis [1], Chan discusses more sophisticated methods.

Instead, we generate a binary tree ( $k' = 2$ ) of regression models that resembles a meta-learning tree as just described.

Depending on how the estimators are combined, we have

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

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

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

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

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

If the benefit is given, we don't need to combine estimated benefits, thus there are three unique methods to combine probabilities, *Averaging*:  $\bar{P}_{\text{avg}}(x)$ , *Regression*:  $\bar{P}_{\text{reg}}(x)$  and *Combiner*:  $\bar{P}_{\text{combiner}}(x)$ .

**Centralized Algorithm** As a comparison, we also describe the centralized algorithm in Figure 1(c). In centralized algorithm, the data from each participating sites are first sent to a single site. A global model is computed at that single site. Obviously, it suffers from heavy communication overhead to send all the data. In addition, most of the learning algorithms' complexity are worse than linear and request data to be resident in main memory. For example, decision tree learner has a complexity of  $O(n \cdot \log(n))$ . If the data is too big or cannot fit into main memory in the worst case, the data will be swapped in and out, and training will be too slow.

### 3.2 Overhead Analysis

The total time to compute a model across a number of distributed sites consists of the training time of the slowest site, computation and communication overhead.

$$\text{Total Training Time} = \text{slowest site} + \text{communication overhead} + \text{computation overhead} \quad (8)$$

	Fully Distributed	Partially Distributed	Centralized
Communication Overhead	0	$2k^2$ models + $3n$ numbers or labels	$n$ original data points
Computation Overhead	0	$c_1 \cdot l \cdot O(f(\frac{n}{2}, 2)) + c_3 \cdot (2k - 1) \cdot n$	0
Slowest site	$c_2 \cdot O(g(n_{max}))$	$c_2 \cdot O(g(n_{max}))$	$c_2 \cdot O(g(n))$

Table 1: Overhead analysis.  $k$  is the number of distributed site.  $l = \log_2(k)$ .  $n$  is the number of examples across all  $k$  sites.  $O(f(N, K))$  is the complexity of the algorithm to train the combiner or regressor, which depends on both the number of examples and number of features.  $n_{max}$  is the size of the largest dataset at one of the  $k$  sites and  $n_{max} \ll n$ .  $(2k - 1) \cdot n$  is the complexity to test all on  $n$  training examples using all  $2k - 1 = k + k - 1$  models in the tree.  $O(g(N))$  is the complexity of the learning algorithm to train on the training set.  $c_1, c_2$  and  $c_3$  are constants.

We assume that there are  $k$  distributed sites,  $k = 2^l$  and the total number of training examples from all  $k$  sites is  $n$ .

The fully distributed approach via averaging ensemble incurs neither computation nor communication costs. Since each site computes its model locally and independently, the intrinsic parallelism of multiple systems is taken full advantage of. The total training time is only bounded by the slowest site, that is of  $c_2 \cdot O(g(n_{max}))$  where  $O(g(N))$  is the complexity of the inductive learner and  $n_{max}$  is the size of the largest dataset. As shown in previous work [3], averaging ensemble exhibits both linear speedup and scaled speedup when applied to scale up learning a very large dataset on a single site. Linear speedup and scaled speedup are the best possible.

On the contrary, both meta-learning and regression-based partially distributed methods incur communication and computation overhead. For simplicity, we assume that 2 models are combined at a time, and a random selection of  $\frac{n}{k}$  examples are used to train either the combiner or regressor (or called combining model for short) in the tree. The tree has a total of  $l$  (or  $\log_2(k)$ ) levels and  $k - 1$  combining models. At each level, all newly learned models are sent to every site to choose the random set of  $\frac{n}{k}$  examples. Totally, we transfer  $\sum_1^l k \times ((k - 1) + (\frac{k}{2} - 1) + \dots + 1) \simeq 2k^2$  models for all  $l$  levels. To compute each combining model in the tree,  $\frac{2n}{k}$  predictions and  $\frac{n}{k}$  true values are randomly chosen and sent to a single site. Considering all  $k - 1$  non-leaf combining models in the tree, there are  $(k - 1) \cdot \frac{2n}{k} \simeq 2n$  predictions and

$(k - 1) \cdot \frac{n}{k} \simeq n$  true values, or  $3n$  numbers or labels to be transferred. Besides transferring data and models, tree-structured approach consumes an additional computational overhead of  $c_1 \cdot (k - 1) \cdot O(f(\frac{n}{k}, 2))$  to compute  $k - 1$  combining models in serial or  $c_1 \cdot l \cdot O(f(\frac{n}{k}, 2))$  in parallel.  $O(f(N, K))$  is the complexity of the algorithm to train either the combiner or regressor, which depends on both the number of examples  $N$  and the number of feature  $k$ . To produce the training data for the combining models, all  $2k - 1$  ( $k$  base classifiers and  $k - 1$  combining models) predict on all  $n$  original training data, which brings an additional overhead of  $c_3 \cdot (2k - 1) \cdot n$ .

As a comparison, the centralized system has to transfer all data to one single site for a total  $n$  original data points. The total training time on  $n$  data points is in the scale of  $O(g(n))$ , which is significantly slower than  $O(g(n_{max}))$  when  $n_{max} \ll n$ , especially if  $n$  original training examples cannot fit into the main memory and memory swapping takes place.

As a summary, we list the overhead of all three methods in Table 1 as a comparison. The fully distributed averaging ensemble has obviously the lowest “zero” overhead, and the training time  $O(g(n_{max}))$  is the quickest as well. It remains to be seen if its accuracy is as good as the other more complicated tree-structured methods, and all distributed approaches will have accuracy similar or better than the single “global” model of centralized learning.

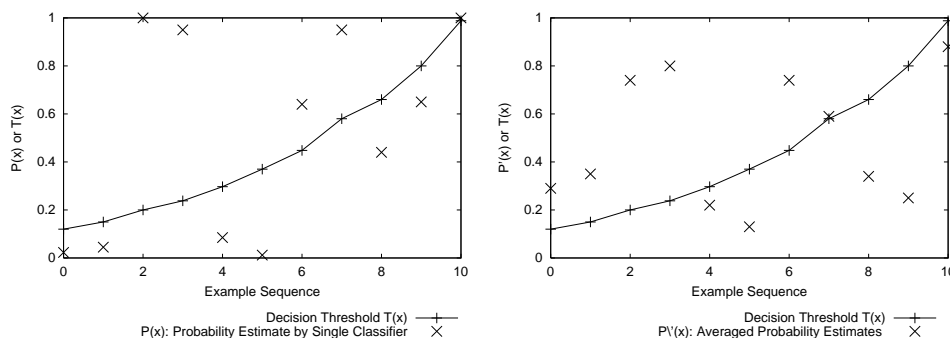
### 3.3 Desiderata

The total benefit or accuracy of the fully distributed averaging ensemble 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 [9] that uncorrelated errors are reduced by averaging. The averaged probability may still be different from the global classifier of the centralized system, but the difference may not make a difference to final prediction due to the error tolerance property (Section 2).

The averaged probability for the fully distributed learning is very likely to have higher benefits than the

Figure 2: Cost-sensitive decision plots. The left plot is conjectured for a global classifier of the centralized system, while the right plot is conjectured for the averaged probability of multiple classifiers. Due to the *even-ing effect*, two expensive examples in the bottom left corner are predicted as positive by the multiple model.



“global” classifier because of its “even-ing 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 2 illustrates two exemplary plots. The left plot is conjectured for global 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 *even-ing 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. (that’s where “even-ing effect” comes from.) This is visually illustrated in Figure 2 by comparing the right plot to the left plot. The even-ing 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 global 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 global classifier, however predicted to be

positive by the multiple model. Due to the even-ing effect, averaging of multiple probabilities biases more towards expensive examples than the global classifier. This is a desirable property since expensive examples contribute greatly towards total benefit. Based on the above conjectured analysis, we think that averaging will actually increase total benefit.

We employ tree-structured method to design partially distributed learning. 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.

## 4 Experiment

We have evaluated the distributed learning system 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.

### 4.1 Dataset

In the first problem, 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  $(R(x) = P(\text{donate}|x)Y(x)) > 0.68$ . The data has already been divided into a training set and a test set. The training set consists of 95412 records for which it is known whether or not the person made a donation and how much the donation was if the person donated. The test set contains 96367 records for which similar donation information was not published until after the KDD'98 competition. We used the standard training/test set splits to compare with previous results. The

feature subsets were based on the KDD'98 winning submission of Georges and Milley [4]. To estimate the donation amount, we use the multiple linear regression method.

In the second problem, 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 dataset was provided by an international bank to Columbia University for research in fraud and intrusion detection. It 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).

In the third problem, 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.

## 4.2 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 sent to both donors and non-donors. For the credit card fraud detection dataset, it is the total correctly detected fraudulent transaction amount minus the overhead of investigation of both true positives (*fraud* predicted as *fraud*) and false positives (*nonfraud* predicted as *fraud*). And for the adult dataset, this is the total profits of correct predictions.

Since we don't have truly distributed datasets, we simulate distributed learning by partitioning a single

dataset into  $k$  subsets as if there were  $k$  distributed sites. Communication overhead records the total number of bytes to be transferred if these sites were indeed interconnected across a network. Computation overhead is the total extra training time to train the combiner or regressor. This is a widely adopted approach to evaluate distributed data mining algorithms. One important aspect of our experiments is to test that the scalable methods’ total benefit is independent of the number of sites  $k$ , or the total benefit of distributed learning across  $k$  sites is very close to the total benefit of the “global” single classifier of centralized learning system. We have chosen to use the following number of partitions,  $k = 8, 16, 32, 64, 128$  and  $256$ . To generate these data subset, the original dataset is “sequentially” partitioned into smaller pieces.

We used the C4.5 decision tree learning algorithm [7] 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.

### 4.3 Experimental Result

The total benefit serves as baseline to compare with the distributed approaches. We then find out how well the various distributed methods work and how the benefits are influenced by the number of sites. Finally, we evaluate the training overhead of each approach.

Dataset	raw	curtailment
Donation	12489.6	13292.7
Credit Card Fraud	552400	733980
Adult	16255	16443

Table 2: The total benefits attained by centralized learning method of the global classifier. This is the baseline to evaluate distributed methods.

#### 4.3.1 Total Benefits

The results by the centralized learning are shown in Table 2, which serve as the baseline to evaluate distributed methods.

Learning Methods		raw	curtailment
Fully Distributed	Simple Averaging	13842.6	14643.3
	Multiplex Averaging	13875.8	<b>14702.9</b>
Partially Distributed	Averaging Regression	13790.7	14597.5
	Regression Averaging	13640.4	14620.5
	Regression Regression	13545.4	14512.2
	Combiner Averaging	13420.4	14310.3
	Combiner Regression	13200.7	14207.5

Learning Method		raw	curtailment
Fully Distributed	Averaging	<b>808438</b>	804964
Partially Distributed	Regression	729384	784310
	Combiner	738294	789845
raw	curtailment		
16235	<b>16435</b>		
15910	16021		
15845	15945		

Table 3: Average total benefits of ensemble methods over different numbers of partitions. Results in bold font is the best result for that dataset for centralized, partially and fully distributed methods. Comparing with the baseline results in Table 2. The results by distributed methods are all better.

In order to compare distributed learning methods, we calculate the average of total benefits of each method over different numbers of sites  $k$ . The results are shown in Table 3. 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 distributed methods all perform reasonably well, and they all have higher benefits than the baseline centralized learning method. The fully distributed learning via averaging ensemble consistently beat the more complicated partially distributed learning via combiner and regression tree across three datasets. This means that although remarkably simple, fully distributed learning (both simple and multiplex averaging) is a very accurate method, and using more complicated and expensive methods don't bring extra benefits.

In Table 4 and Figure 3, we plot changes of total benefits (using curtailment averaging<sup>2</sup>) with growing number of sites  $k$  using the fully distributed approach. We can clearly see the total benefits are all higher than

<sup>2</sup>The results for curtailment simple averaging and multiplex averaging are very close for the donation dataset, therefore, we choose to present multiplex averaging result.

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. 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).

#### 4.4 Communication and Computation Overhead

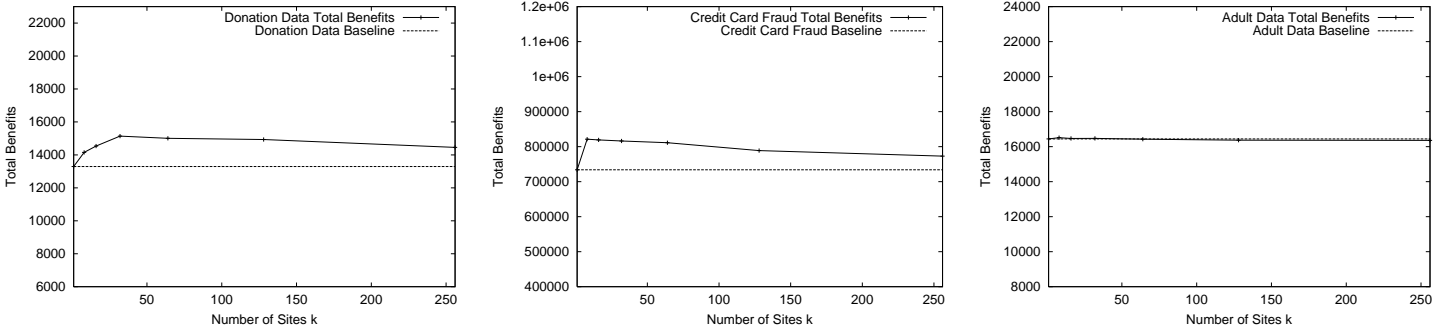
Both the communication and computation overhead are obviously 0 for the fully distributed approach. For the partially distributed approach, we use credit card dataset with  $k = 256$  as an example. The total number of extra models (combiner or regressor) are 255. Every regressor have 3 parameters ( $y = a_0 + a_1x_1 + a_2x_2$ ) and each parameter is a real number stored as 8 bytes. The total number of bytes is 24 bytes per regressor or  $256 \times 511 \times 24$  bytes = 1Mbytes to transfer all the models. The decision tree generated for each combiner is around 100 bytes, so the total bytes to transfer all combiners is approximately 4MBytes. The additional data (predictions and true values) transfered for both regressor and combiner tree is about 9MBytes ( $406009 \times 3 \times 8$ ). The extra time to train the regressors is around 40 minutes in serial training and 9 minutes in parallel; the extra time to train the combiners is about 15 minutes in serial and 5 minutes in parallel. Besides computing the combining models, both methods spend an additional 30 minutes to produce the training data to compute the combining model.

In our previous work, we have shown that the total training time to compute the fully distributed averaging ensemble with  $k = 256$  in parallel is about 5814 times faster than computing a single global classifier from all data (without counting the extra time to transfer all data to a single site.)

	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 Fraud	733980	821311	819155	816244	811282	788795	772995
Adult	16443	16513	16470	16471	16428	16370	16359

Table 4: Changes in total benefits relative to the number of sites  $k$  using curtailed probabilities

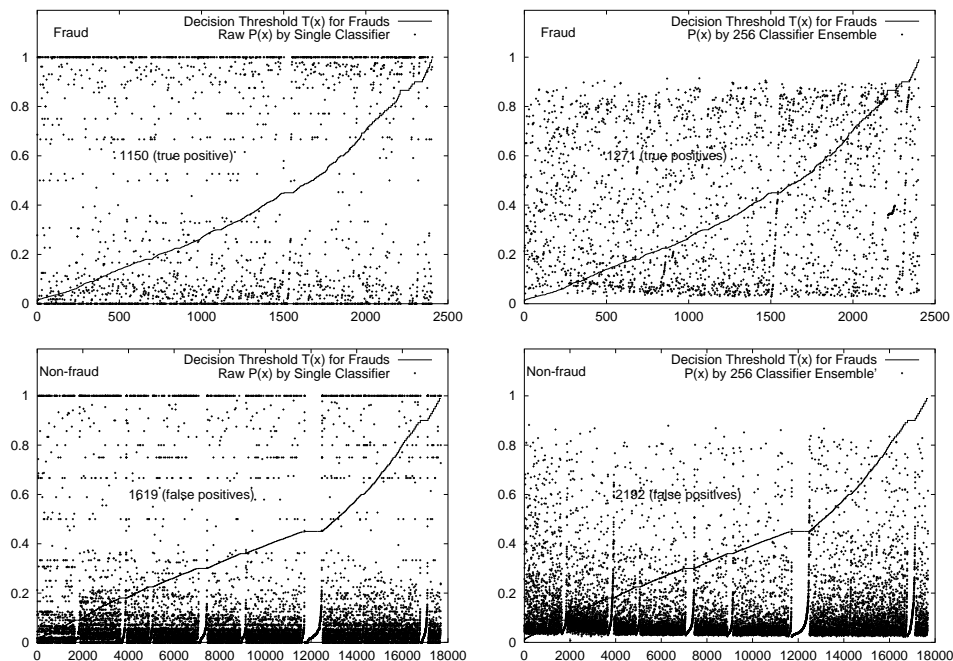
Figure 3: Plot of changes in total benefits as shown in Table 4



#### 4.5 Additional Experiment on Very Large Dataset and Different Learners

We have tested the fully distributed approach on a huge catalog mailing dataset on a completely different learning system. Due to agreement with the merchant, we cannot reveal more details about the dataset beyond what is discussed in this paper. The dataset contains millions of records of ordering history of the merchant’s customers. Each record is described by hundreds of features that profile the customer’s geographic, demographic, lifestyle and past ordering information. The merchant need to choose the subset of customers to solicit catalog (about \$5 per copy) in order to maximize profits. The learning task is very challenging due to the large size of the dataset and huge number of features and cost-sensitive nature of the problem. Due to proprietary reasons, the dataset has to be run on designated learning system which uses variations of Naive Bayes and Linear Regression. We have randomly partitioned training set into 5 subsets. All five base classifiers were computed concurrently on the same computer that the centralized global model was trained (on the 5 subsets as a whole). The total profit by the global model for the centralized learning

Figure 4: Decision threshold and probability output by centralized single model and 256-classifier fully distributed averaging ensemble for credit card dataset



is \$14160.5 while the fully distributed averaging ensemble has achieved a profit of \$15009.5, which is an increase of 7% over the single model.

We have also tested the donation dataset on this learning system, which uses variations of Naive Bayes and Regression that are different from C4.5, and obtained higher profit and less training time with the fully distributed averaging ensemble approach.

#### 4.6 Explanation

We explained the reason in Section 3.3 why averaging may have potentially higher benefits. The empirical results verified our conjecture. As an example, in Figure 4, we plot decision plots (as defined in Section 3.3) 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 examples<sup>3</sup> whose  $P(x) > T(x)$  (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 varied profits and negative examples have low or fixed cost, the ensemble methods tend to beat the single model.

## 5 Discussion

Compared with both centralized and partially distributed systems, the fully distributed averaging ensemble has several advantages besides its zero communication and computation overhead. Since base models are computed independently on the local sites, it takes full advantage of the intrinsic parallelism of multiple systems. As shown in our previous work [3], the fully distributed averaging ensemble has both linear speedup and scaled speedup. The method is simple and straightforward, and doesn't require complicated control mechanism, if any. Since each site is computed individually without any communication, they don't have to be inter-connected, and can be completely detached in the extreme case. For applications where *security* is a major concern, averaging ensemble provides a satisfactory solution; there is no information exchange during learning, consequently, little opportunity to leak and steal information. It also provides a natural solution for *heterogeneous systems, multiple schemas* and *multiple learning algorithms*; each site can totally use its own code, own system and data format (even legacy system) to compute its local models. As long as the format of the models are interchangeable, the local models can be shared and exchanges to form the averaging ensemble with ease. Since each learning procedure is totally independent, the failure

---

<sup>3</sup>To show these numbers clearly on the plot, we don't plot the surrounding data points around the text area

of one site will not influence the other sites at all, which provide natural *fault-tolerance*. When more sites decide to participate in the ensemble, their models can be joined at any time; when some sites have new data available, their model can be upgraded at any time and replace the older model. This provide an easy solution for *incrementability*.

Unbalanced dataset size may be an important issue for some applications of distributed data mining; the size of the dataset at different site may be very different in size. Big dataset may dominate both training time and accuracy. Since the accuracy of averaging ensemble doesn't depend on  $k$ , the number of sites, we can always make data size at different site similar. Actually, when the dataset at one site is partitioned into data subsets, the learning on this site alone (sequential or parallel) also improves.

## 6 Related Work

There has been extensive research on distributed data mining. However, they mostly focus on *cost-insensitive* problems where each example has the same price and there is no penalty for misclassification. 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 combined or used in making a prediction. Meta-learning has been implemented as Java based Meta-learning System or JAM [5]. Incremental learning, such as ID5 [10], assimilates one or a few example at a time. The temporary model from previous learned site can be transfered to other sites to complete learning; however, the intrinsic parallelism of multiple systems are not taken advantage of, and the modification to make each algorithm incremental is algorithm specific. SPRINT [8] utilizes data structure to scale up decision tree learning on very large dataset; but it doesn't handle distributed learning among multiple sites. Some researchers parallelize specific algorithm, such as DRL rule-based algorithm for multi-processor learning [6], to solve distributed learning problems. Like incremental learning, each parallel modification is algorithm-specific.

## 7 Conclusion and Future Work

We have proposed and evaluated several ensemble approaches to combine probabilities and benefits for distributed cost-sensitive 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 for fully distributed learning. 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 for partially distributed learning. We have chosen three different types of cost-sensitive problems for empirical study.

We have found that both fully and partially distributed methods achieve total benefit that is as good as or even better than the global classifier trained on the all available data from every site. The most accurate approaches are the two fully distributed averaging methods, whose total benefit top among all the ensemble approaches and even consistently beat the centralized global classifier. They also exhibit zero communication and computation overhead. Our study have also shown that more complicated partially distributed tree-structured methods offer neither lower communication nor computation overhead.

Based on our extensive experimental and analytical study of many candidate combining methods, we find that the fully distributed averaging approaches are an effective framework for distributed cost-sensitive data mining.

**Future Work** One problem faces all ensemble approaches is the low efficiency of model deployment since multiple models have to be used to predict an example. We have proposed a few pruning methods to reduce the number of models. The initial results have shown that with less than 10% of the original number of models, we can still achieve the same level of benefit as the original averaging ensemble with all the classifiers. When  $k$ , the number of site, becomes too big, the averaging ensemble may not work well. We have used several sampling techniques to randomly choose  $k'$  out of  $k$  data subsets ( $k' \ll k$ ) to train  $k'$

models to estimate the performance of  $k$  models. Our initial experiments show that we accurately estimate the total benefits within 95% of confidence.

## References

- [1] P Chan. *An Extensible Meta-learning Approach for Scalable and Accurate Inductive Learning*. PhD thesis, Columbia University, Oct 1996.
- [2] T Dietterich, D Margineatu, F Provost, and P Turney, editors. *Cost-Sensitive Learning Workshop (ICML-00)*, 2000.
- [3] Wei Fan, Haixun Wang, Philip Yu, and Salvatore Stolfo. A framework for scalable cost-sensitive learning based on combining probabilities and benefits. In *submitted*. 2001.
- [4] J. Georges and A. H. Milley. In *KDD'99 competition report*, 1999. <http://www-cse.ucsd.edu/users/elkan/kdresults/html>.
- [5] Andreas Prodromidis. *Management of Intelligent Learning Agents in Distributed Data Mining Systems*. PhD thesis, Columbia University, Oct 1999.
- [6] Foster Provost and D Hennessy. Scaling up distributed machine learning via cooperation. In *Proceedings of Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, 1996.
- [7] J Ross Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufman Publishers, 1992.
- [8] 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.
- [9] 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.
- [10] Paul Utgoff. An improved algorithm for incremental induction of decision tree. In *Proceedings of Eleventh International Conference on Machine Learning (ICML-94)*, pages 318–325, 1994.
- [11] D Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.
- [12] 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.