Mixture of Expert Agents for Handling Imbalanced Data Sets

S. B. Kotsiantis and P. E. Pintelas
Educational Software Development Laboratory
Department of Mathematics
University of Patras, Hellas
{sotos, pintelas}@math.upatras.gr

***Abstract*— Many real-world data sets exhibit skewed class distributions in which almost all cases are allotted to a class and far fewer cases to a smaller, usually more interesting class. A classifier induced from an imbalanced data set has, typically, a low error rate for the majority class and an unacceptable error rate for the minority class. This paper firstly provides a systematic study on the various methodologies that have tried to handle this problem. Finally, it presents an experimental study of these methodologies with a proposed mixture of expert agents and it concludes that such a framework can be a more effective solution to the problem. Our method seems to allow improved identification of difficult small classes in predictive analysis, while keeping the classification ability of the other classes in an acceptable level.**

***Index Terms*— supervised machine learning, imbalanced data sets, expert agents.**

I. INTRODUCTION

TYpically classifiers are expected to be able to generalize over unseen instances of any class with equal accuracy. For example, in a two-class domain of positive and negative instances, the classifier will perform on an unseen set of instances with equal accuracy on both the positive and negative classes. This of course is the ideal situation. In many applications classifiers are faced with imbalanced data sets, which can cause the classifier to be biased towards one class. This bias is the result of one class being heavily under represented in the training data compared to the other classes. It can be attributed to the way in which classifiers are designed. Inductive classifiers are typically designed to minimize errors over the training instances. Learning algorithms, because of the fact that the cost of performing well on the over-represented class outweighs the cost of poor performance on the smaller class, can ignore classes containing few instances. Moreover, the difficulty to distinguish between the rare cases (i.e., true exceptions) and noise is also responsible for poor performance on the minority class.

For a number of application domains, a huge disproportion in the number of cases belonging to each class is common. For instance, in detection of fraud in telephone calls [9] and credit card transactions [25], the number of legitimate transactions is much higher than the number of fraudulent transactions. Moreover, in direct marketing (Ling and Li, 1998), it is common to have a small response rate (about 1%) for most marketing campaigns. Other examples of domains with intrinsic imbalance can be found in the literature such as rare medical diagnoses [27] and oil spills in satellite images [16]. Thus, learning with skewed class distributions is an important issue in supervised learning.

The machine learning community has mainly addressed the issue of class imbalance in two ways. One is to assign distinct costs to training instances [7]. The other is to re-sample the original dataset, either by oversampling the minority class and/or under-sampling the majority class [15]; [12]. Although many methods for coping with imbalanced data sets have been proposed, still remain open questions. One open question is whether simply changing the distribution skew can improve predictive performance systematically. Another question is whether we can tailor learning algorithms to this special learning environment so that the accuracy for the extreme class values can be improved.

To handle the problem, we developed an Agent-based Knowledge Discovery (ABKD) method that combines the two fields of Distributed Artificial Intelligence and Machine Learning. In ABKD, an agent is a software entity that can 1) interoperate with its data source and/or other agents, 2) receive/gather raw data, 3) process and learn from the data source or from other sources, and 4) coordinate with other agents to produce relevant and useful information. Adopting agent technology provides parallelism, improves the speed and the reliability of learning and assists developers in designing distributed learning systems. The effectiveness of our approach is evaluated over eight imbalanced datasets using the C4.5 [22], 5NN [1], Naive Bayes [6] as classifiers and the geometric mean of accuracies as performance measure [16]. In the following section we briefly describe the

ISSN 1109-9305

used machine learning techniques and we explain the reasons for their poor performance in imbalance data sets. Section 3 reviews the attempts for handling imbalanced data sets, while section 4 presents the details of our approach. Section 5 presents experimental results comparing our approach to other approaches. Finally, section 6 discusses the results and suggests directions for future work.

II. Learning techniques and algorithms

A small imbalance in the class distribution is not serious, but when some classes are heavily under-represented, many machine-learning methods are likely to run into problems. Cases belonging to small classes are lost among the more frequent cases during learning, and, consequently, classifiers such as decision trees, Bayesian networks and instance-based classifiers are unable to classify correctly new unseen cases from the minority classes. In the following subsections we briefly describe decision trees, Bayesian networks and instance based classifiers and we refer to the reasons for their poor performance in minority class of imbalance data sets.

A. Decision trees

Murthy [20] provides a recent overview of existing work in decision trees and a taste of their usefulness to the newcomers in the field of machine learning. Decision trees are trees that classify instances by sorting them based on feature values. Each node in a decision tree represents a feature in an instance to be classified, and each branch represents a value that the node can take. Instances are classified starting at the root node and sorting them based on their feature values.

The feature that best divides the training data would be the root node of the tree. The same process is then repeated on each partition of the divided data, creating sub trees until the training data are divided into subsets of the same class. At each level in the partitioning process a statistical property known as information gain is usually used to determine which feature best divides the training instances [22].

A decision tree, or any learned hypothesis h, is said to overfit training data if there exists another hypothesis h' that has a larger error than h when tested on the training data, but a smaller error than h when tested on the entire data set. There are two common approaches that decision tree induction algorithms can use to avoid overfitting training data: 1) Stop the training algorithm before it reaches a point in which it perfectly fits the training data, 2) Prune the induced decision tree.

For the scope of our study the most well-known decision tree algorithm - C4.5 [22] - was used. One of the latest researches that compare decision trees and other learning algorithms is made by Tjen-Sien Lim et al. [26] and shows that the mean error rates of most algorithms are sufficiently similar and that their differences are statistically insignificant. But, unlike error rates, there are huge differences between the training times of the algorithms. C4.5 has one of the best combinations of error rate and speed.

The reason of poor performance of decision tree classifiers in minority class is that most of the classifiers employ a post-pruning method. Any node can be removed and assigned the most common class of the training instances that are sorted to the node in question. Thus, if a class is rare, decision tree algorithms often prune the tree down to a single node that classifies all instances as members of the common class leading to poor accuracy on the instances of minority class.

B. Naive Bayes

Probabilistic classifiers and, in particular, the naive Bayes classifier, are among the most popular classifiers in the machine learning community and they are used increasingly in many applications. Naive Bayes (NB) classifier is the simplest form of Bayesian network (Jensen, 1996) since it captures the assumption that every feature (a_1, a_2, ... a_i) is independent of the rest of the features, given the state of the class feature V.

The formula used by the Naive Bayes classifier is:

$$v_{\max} = \max_{v_j \in V} P(v_j) \prod_i P(a_i|v_j)$$

where V is the target output of the classifier and $P(a_i|v_j)$ and $P(v_i)$ can be calculated based on their frequency in the training data.

For numerical features, one can model the component marginal distributions in a wide variety of ways. The simplest would be to adopt some parametric form e.g. marginal Gaussian estimators. The assumption of independence is clearly almost always wrong. However, Friedman [10] explains why simple naive Bayes method remains competitive, even though it provides very poor estimates of the true underlying probabilities. Good probability estimates are not necessary for good classification; similarly, low classification error does not imply that the corresponding class probabilities are being estimated (even remotely) accurately. In addition, [6] performed a large-scale comparison of Naive Bayes classifier with state-of-the-art algorithms for decision tree induction and instance-based learning on standard benchmark datasets, and found it sometimes to be superior to each of the other learning schemes even on datasets with substantial feature dependencies.

The extreme skewness in class distribution is problematic for Naïve Bayes because the prior probability

of the majority class overshadows the differences in the attribute conditional probability terms.

C. Instance-based learning

Instance-based learning algorithms belong to the category of lazy-learning algorithms [18], as they delay the induction until classification is performed. One of the most straightforward instance-based learning algorithms is the nearest neighbour algorithm [1]. K-Nearest Neighbour (kNN) is based on the principle that the value of the label of an unclassified instance can be determined by observing the class of its nearest neighbours.

In general, instances can be considered as points within an n-dimensional instance space where each of the n-dimensions corresponds to one of the n-features that are used to describe an instance. The absolute position of the instances within this space is not as significant as the relative distance between instances. This relative distance is determined by using a distance metric. Ideally, the distance metric must minimize the distance between two similarly classified instances, while maximizing the distance between instances of different classes [1]. In our study, we made use of the well known 5-NN algorithm using the Euclidean distance as distance metric.

In imbalanced data sets as the number of the instances of the majority class grows, so does the likelihood that the nearest neighbour of any instance will belong to the majority class. This leads to the problem that many instances of the minority class will be misclassified.

III. Review of existing techniques for handling imbalanced data sets

A classifier induced from an imbalanced data set has, typically, a low error rate for the majority class and an unacceptable error rate for the minority class. The problem arises when the misclassification cost for the minority class is much higher than the misclassification cost for the majority class. In this situation, it is important to accurately classify the minority class in order to reduce the overall cost.

A simple method that can be used to imbalanced data sets is to reweigh training instances according to the total cost assigned to each class [5]. The idea is to change the class distributions in the training set towards the most costly class. Suppose that the instances of the positive class are four times more than the instances of the negative class. If the number of negative instances are artificially increased by a factor of four, then the learning system, aiming to reduce the number of classification errors, will come up with a classifier that is skewed towards the avoidance of error in the negative class, since any such errors are penalised four times more.

Japkowicz [11] discussed the effect of imbalance in a dataset. She mainly evaluated two strategies: under-sampling and resampling. Two resampling methods were considered. Random resampling consisted of re-sampling the smaller class at random until it consisted of as many samples as the majority class and "focused resampling" consisted of resampling only those minority instances that occurred on the boundary between the minority and majority classes. Random under-sampling was also considered, which involved under-sampling the majority class samples at random until their numbers matched the number of minority class samples; focused under-sampling involved under-sampling the majority class samples lying further away. She noted that both the sampling approaches were effective, and she also observed that using the sophisticated sampling techniques did not give any clear advantage in the domain considered.

Kubat and Matwin [15] also selectively under-sampled the majority class while keeping the original population of the minority class with satisfied results. Batista et al. [2] used a more sophisticated under-sampling technique in order to minimize the amount of potentially useful data. The majority class instances are classified as "safe", "borderline" and "noise" instances. Borderline and noisy cases are detected using Tomek links, and are removed from the data set. Only safe majority class instances and all minority class instances are used for training the learning system. A Tomek link can be defined as follows: given two instances x and y belonging to different classes, and be d(x, y) the distance between x and y then a (x, y) pair is called a Tomek link if there is not a case z, such that d(x, z) $<$ d(x, y) or d(y, z) $<$ d(y, x).

Both, under-sampling and over-sampling, have known drawbacks. Undersampling can throw away potentially useful data, and over-sampling can increase the likelihood of occurring overfitting, since most of over-sampling methods make exact copies of the minority class instances. In this way, a symbolic classifier, for instance, might construct rules that are apparently accurate, but actually, cover one replicated instance.

Another approach is that of Ling and Li [17]. They combined over-sampling of the minority class with under-sampling of the majority class. However, the over-sampling and under-sampling combination did not provide significant improvement. Chawla et al. [4] propose an over-sampling approach in which the minority class is over-sampled by creating "synthetic" instances rather than by over-sampling with replacement with better results.

Changing the class distribution is not the only way

to improve classifier performance when learning from imbalanced data sets. A different approach to incorporating costs in decision-making is to define fixed and unequal misclassification costs between classes. Cost model takes the form of a cost matrix, where the cost of classifying a sample from a true class j to class i corresponds to the matrix entry λ_{ij}. This matrix is usually expressed in terms of average misclassification costs for the problem. The diagonal elements are usually set to zero, meaning correct classification has no cost. We define conditional risk for making a decision α_i as:

$$R(a_i|x) = \sum_i \lambda_{ij} P(v_j|x)$$

The equation states that the risk of choosing class i is defined by fixed misclassification costs and the uncertainty of our knowledge about the true class of x expressed by the posterior probabilities. The goal in cost-sensitive classification is to minimize the cost of misclassification, which can be realized by choosing the class (v_j) with the minimum conditional risk.

An alternative to balancing the classes is to develop a learning algorithm that is intrinsically insensitive to class distribution in the training set. An example of this kind of algorithm is the SHRINK algorithm [16] that finds only rules that best summarize the positive instances (of the small class), but makes use of the information from the negative instances.

MetaCost [7] is another recently proposed method for making a classifier cost-sensitive. The procedure begins to learn an internal cost-sensitive model by applying a cost-sensitive procedure, which employs a base learning algorithm. Then, MetaCost procedure estimates class probabilities using bagging and then re-labels the training instances with their minimum expected cost classes, and finally relearns a model using the modified training set.

Furthermore, Schapire et al. [23] gave different weights for false positives and false negatives to apply AdaBoost than bagging in text-filtering. AdaBoost uses a base classifier to induce multiple individual classifiers in sequential trials, and a weight is assigned to each training instance. At the end of each trial, the vector of weights is adjusted to reflect the importance of each training instance for the next induction trial. This adjustment effectively increases the weights of misclassified instances and decreases the weights of the correctly classified instances. Fan et al. [8] also proposed a similar technique. Their intuition was that in addition to assigning high initial weights to costly instances, the weight-updating rule should also take cost into account and increase the weights of costly misclassification more but decrease the weights of costly correct classification less.

IV. Proposed Technique

Agent Based Knowledge Discovery (ABKD) can contribute to machine learning and data mining in a number of ways. First of all, adopting ABKD provides parallelism, improves the speed and the efficiency of machine learning. Second, agent concepts assist developers in designing distributed learning systems. The encapsulation of variables and methods in the object-oriented paradigm leads to the idea of encapsulating learning techniques, and thus developers can reuse agent objects that contain existing techniques. After defining the agent objects, the developers can design how the agent objects interact with one another to generate the correct results.

Many areas of research employ agent technology, and thus the definition of an agent varies according to the focus of the research. For example, research in multi-agent systems (MAS) commonly characterizes agents as autonomous and able to plan and coordinate within an organization for solving a problem. In Agent Based Knowledge Discovery, an agent is a software entity that can 1) interoperate with its data source and/or other agents, 2) receive/gather raw data, 3) process and learn from the data source or from other sources, and 4) coordinate with other agents to produce relevant and useful information. Based on this characterization, many aspects of research in ABKD, such as planning, coordination, and communication, overlap with other fields of agent research. This paper, however, limited the description of agent technology to the context of machine learning and knowledge discovery.

Our approach is to use the three agents (the first learns using Naïve Bayes, the second using C4.5 and the third using 5NN) on a filtered version of training data and combine their predictions according to a voting scheme. This technique attempts to achieve diversity in the errors of the learned models by using different learning algorithms. The intuition is that the models generated using different learning biases are more likely to make errors in different ways. We also used feature selection of the training data because in small data sets the amount of class imbalance affects more the induction and thus feature selection makes the problem less difficult.

It might be difficult or impossible to find a single classifier that performs as well as a good ensemble of classifiers. For a group of abstract level classifiers each of which outputs only a class label for each input instance, the means of obtaining a combined decision is bound to be limited to some sort of voting scheme, with or without taking prior performance into consideration. Among the combination methods for abstract-level classifiers, majority vote is the simplest to implement, since it requires no prior training [13].

The goal of feature selection is to reduce the dimensionality of feature space by selecting a subset of the features and discard the useless. In a given database, many features can be totally irrelevant or redundant in terms of predicting the target. Ideally, if we can identify these features and eliminate them from the training data, the learning performance would be improved. Mladenic and Grobelnik [19] proposed a feature subset selection approach to deal with imbalanced class distribution in the Information Retrieval domain with good results.

The well-known filter that computes the empirical mutual information between features and the class [24], and discards low-valued features was modified for our method. At this point, it must be mentioned that mutual information is based on entropy. If the training set $\boldsymbol{S}$ is partitioned into V subset $\boldsymbol{S}_1, \ldots, \boldsymbol{S}_V$ according to the V different values of a feature X, the *mutual information* between feature X and class attribute Y is defined as:

$$Gain(X) \equiv info(\mathbf{S}) + \sum_{v=1}^{V} \frac{|\mathbf{S}_v|}{|\mathbf{S}|} info(\mathbf{S}_v)$$

However, the mutual information gain criterion has a strong bias in favor of features with many different values, thus we rectify this bias by a kind of normalization – gain ratio that sometimes is used from decision tree algorithms [22]. In detail, the bias is rectified by normalization:

$$split\ info(\mathbf{S}) = -\sum_{v=1}^{V} \left(\frac{|\mathbf{S}_v|}{|\mathbf{S}|} \right) \log_2 \left(\frac{|\mathbf{S}_v|}{|\mathbf{S}|} \right)$$

which represents the potential information generated by dividing $\boldsymbol{S}$ into V partitions, whereas

$$Gain\ Ratio(X) \equiv Gain(X)/split\ info(X)$$

expresses the proportion of information generated by the partition.

Features are then selected by keeping those for which gain ratio exceeds a fixed threshold e. In order to have a robust selection, we set e to 0.02 of the gain ratio filter, in an attempt to discard only features with negligible impact on predictions. However, such a low threshold can discard many features.

The proposed mixture of expert agents consists of a Facilitator agent that filters the features of the data set and passes a copy of the instances in the learning agents. Then, each learning agent resamples data sets (the relationship between false negative and false positive costs is the inverse of the imbalanced priors) and returns prediction for each instance back to the Facilitator. Finally, the Facilitator makes the final prediction according to majority voting. The pseudocode of our approach is presented in Table I.

Moreover, our approach is schematically represented in Figure 2.

TABLE I

REPRESENTATION OF OUR AGENT-BASED APPROACH

1. The Agent Facilitator filters the features of the data set.
2. The Facilitator passes a copy of the Instances to Agent 1
3. The Facilitator passes a copy of the Instances to Agent 2
4. The Facilitator passes a copy of the Instances to Agent 3
5. Start three threads
- *Thread 1: Agent 1 resamples data and returns prediction for each instance back to the Facilitator.*
- *Thread 2: Agent 2 resamples data and returns prediction for each instance back to the Facilitator.*
- *Thread 3: Agent 3 resamples data and returns prediction for each instance back to the Facilitator.*

6. Synchronize the threads
7. The Facilitator makes the final prediction according to majority voting.

It must be mentioned that JAM (Java Agents for Meta-Learning over Distributed Databases) was the first attempt that combined machine learning techniques with agent technology [25]. JAM scope was to handle more efficiently very large data sets.

In the following section, we empirically evaluate the performance of our approach with the other well known techniques for handling imbalanced data sets on eight data sets. We demonstrate that it is promising and advantageous to avoid commitment to a single best classifier during system construction. Instead, our approach can be used to build classification systems from the available classifiers that will perform best under any class distributions.

V. EXPERIMENTS

In Table II, there is a brief description of the data sets that we used for our experiments. Except for the "eap" data set, all were drawn from the UC Irvine Repository [3]. Eap data is from Hellenic Open University and was used in order to determine whether a student is about to drop-out or not [14]. The data sets from UC Irvine Repository are from domains of: image recognition (ionosphere), medical diagnosis (breast-cancer, diabetes, haberman, hepatitis, sick) and commodity trading (credit-g).

The performance of machine learning algorithms is typically evaluated using predictive accuracy. However, this is not appropriate when the data is imbalanced. A simple but effective strategy for classification would be to simply assign the majority class to all unknown instances. Although this approach would achieve high classification accuracy, it may not be desirable especially for applications that are more interested in detecting the minority class than the majority class. Thus, when comparing the performance of different classifiers in imbalanced data sets, accuracy as a measure is not enough.

Fig. 1. Representation of our agent-based approach

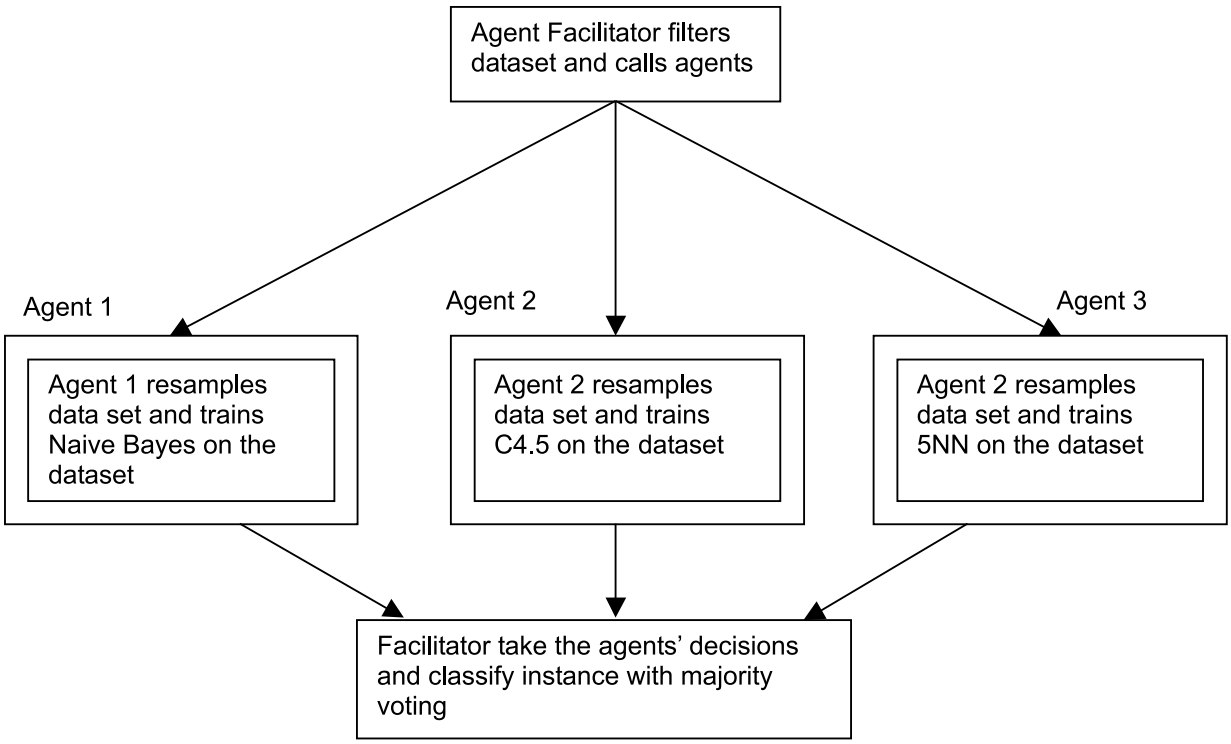

TABLE II
DESCRIPTION OF THE DATA SETS

Data sets	Instances	Categorical features	Numerical features	Instances of minority class	Classes
breast-cancer	286	9	0	85	2
credit-g	1000	13	7	300	2
Diabetes	768	0	8	268	2
Haberman	306	0	3	81	2
Hepatitis	155	13	6	32	2
Ionosphere	351	34	0	126	2
Eap	344	11	0	122	2
Sick	3772	22	7	231	2

A classifier's performance of two class problems can be separately calculated for its performance over the positive instances (denoted as α^+) and over the negative instances (denoted as α^-). The true positive rate (α^+) or sensitivity is the fraction of positive instances predicted correctly by the model. Similarly, the true negative rate (α^-) or specificity is the fraction of negative instances predicted correctly by the classifier.

Kubat et al. [16] propose the geometric mean of the accuracies: $g = \sqrt{a^+ \times a^-}$ for imbalanced data sets. The basic idea behind this measure is to maximize the accuracy on both classes. Moreover, ROC curves (Receiving Operator Characteristic) provide a visual representation of the trade off between true positives (α^+) and false positives (α^-). These are plots of the percentage of correctly classified positive instances α^+ with respect to the percentage of incorrectly classified negative instances α^- [21]. If the model is perfect, then its area under the ROC curve would equal to 1. If the model corresponds to random guessing, then its area under ROC curve would be equal to 0.5. Anything less than 0.5 would be worse than random guessing.

The most popular method for plotting a ROC curve is threshold variation [27]: given a set of test instances and a classifier, the numeric output for each test instance is computed, and the instances are ordered according to the corresponding numeric prediction. Then, for each instance, a (1-α^+, α^+) point is obtained, that is, considering that instances before it are classified as positive and instances after it are classified as negative. Subsequent (1-α^+, α^+) points are linked. The method for plotting a ROC curve is closely related to a method for

making algorithms cost-sensitive, that we call Threshold method [27]. This method uses a threshold so as to maximize the given performance measure in the curve.

The problem of determining which proportion of positive/negative examples is the best for learning is an open problem of learning from imbalanced data sets. In order to make the experiment more realistic, parameters of the cost models were not optimized for each data set, the relationship between false negative and false positive costs was chosen to be the inverse of the assumed prior to compensate for the imbalanced priors.

Classification ability of the learning methods in our experiments was measured with geometric mean of the accuracies. In the following Tables, win (v) indicates that the specific method along with the learning algorithm performed statistically better than the single classifier according to t-test with $p<0.05$. Loss (*) indicates that the specific method along with the learning algorithm performed statistically worse than the single classifier according to t-test with $p<0.05$. In all the other cases, there is no significant statistical difference between the results.

In Table III, one can see the comparisons of the proposed mixture of expert agents with other attempts that have tried to obtain the best performance of a given imbalance data set using Naive Bayes (NB) as base classifier. Five well-known algorithms were used for the comparison: Threshold method [27], Reweighing and Cost Sensitive method [5], Adaboost cost sensitive method [23] and Metacost algorithm [7]. We also present the accuracy of the simple Bayes algorithm as borderline. It must be mentioned that we used the free available source code for these methods by Witten and Frank [27] for our experiments.

In Table III, except for geometric mean, we also present the true-positive rate, and true-negative rate. It must be mentioned that positive class for our experiments is the majority class. In the last row of the Table 2, the mean value of the geometric means is also calculated in all data sets.

In general, all the tested techniques give better results than the single Naive Bayes. The most remarkable improvement is from our technique (ABKD), even though the Threshold method gives, on average, the best accuracy in the minority class. The Metacost cannot improve the results of the NB as his author suspects. It must be mentioned that for Naïve Bayes classifier, modifying the decision boundary (Cost Sensitive method) is equivalent to reweighing training instances so as the relationship between false negative and false positive costs to be the inverse of the imbalanced priors. Moreover, Adaboost cost sensitive method cannot give better results than Cost Sensitive and reweighing method, even though it is a more time consuming technique.

In Table IV, one can see the comparisons of the proposed mixture of expert agents with other attempts that have tried to obtain the best performance of a given imbalance data sets using C4.5 as base classifier. The same five well-known techniques for handling imbalanced data sets were also used for this comparison.

Likewise with the previous experiment, our method has better performance than the other techniques. However, Metacost has really better performance with C4.5 than NB. It must also be mentioned that Threshold method gives worst performance than single C4.5. For C4.5 classifier, modifying the decision boundary (Cost Sensitive method) is less efficient than reweighing training instances so as the relationship between false negative and false positive costs to be the inverse of the imbalanced priors. The reason may be that the pruned decision trees cannot estimate very well the probability of class prediction. Adaboost cost sensitive method, as in the previous experiment, cannot give better results than reweighing method even though it uses more time for training.

Similarly to our results, on several experiments performed in [21], decision tree classifiers generated from balanced distributions obtained results that were, frequently, better than those obtained from the naturally occurring distributions. Especially, these experiments were conducted with no pruning. Many of the results in both papers can be explained by understanding the role of small disjuncts in learning. Decision tree algorithms tend to form large disjuncts to cover general cases and small disjuncts to cover rare cases. Concepts with many rare cases are harder to learn than those with few, since general cases can be accurately sampled with less training data.

In Table V, one can see the comparisons of the proposed mixture of expert agents with other attempts that have tried to obtain the best performance of a given imbalance data sets using 5NN as base classifier. The same five well-known techniques for handling imbalanced data sets were used for this comparison, too.

All the techniques gave better results than single 5NN algorithm. However, our method has the best geometric mean of accuracies in this experiment, too. It must be mentioned that Adaboost cost sensitive method and Metacost algorithm are extremely time consuming techniques if they are combined with lazy algorithm 5NN without offering spectacular improvement in the performance. For 5NN classifier, modifying the decision boundary (Cost Sensitive method) is similarly efficient to reweighing training instances to the inverse of the imbalanced priors. Threshold method gives, on average, the least improvement in the performance of 5NN.

TABLE III

ACCURACY ON MAJORITY CLASS (α^+), ACCURACY ON MINORITY CLASS (α^-) AND GEOMETRIC MEAN (G) WITH NB AS BASE CLASSIFIER

Data sets		ABKD	ThresNB	ReWNB	CostNB	AdabcosNB	MetacostNB	NB
breast-cancer	g	0.62	0.63v	0.66v	0.66v	0.63v	0.65v	0.6
	α^+	0.72	0.62	0.74	0.74	0.72	0.79	0.85
	α^-	0.53	0.65	0.58	0.58	0.56	0.54	0.43
credit-g	g	0.67	0.71v	0.72v	0.72v	0.71v	0.66	0.65
	α^+	0.7	0.69	0.75	0.75	0.75	0.77	0.86
	α^-	0.65	0.74	0.69	0.69	0.67	0.57	0.49
diabetes	g	0.7	0.72	0.73	0.73	0.73	0.70	0.71
	α^+	0.72	0.65	0.78	0.78	0.77	0.75	0.84
	α^-	0.69	0.8	0.68	0.68	0.69	0.66	0.6
haberman	g	0.61v	0.59v	0.56v	0.56v	0.56v	0.57v	0.44
	α^+	0.77	0.64	0.89	0.89	0.88	0.87	0.94
	α^-	0.49	0.55	0.35	0.35	0.36	0.38	0.21
hepatitis	g	0.79	0.76	0.8	0.8	0.78	0.81v	0.78
	α^+	0.84	0.87	0.83	0.83	0.86	0.79	0.87
	α^-	0.75	0.67	0.78	0.78	0.71	0.84	0.7
ionosphere	g	0.9v	0.88v	0.82	0.82	0.91v	0.77*	0.83
	α^+	0.96	0.93	0.78	0.78	0.93	0.68	0.8
	α^-	0.84	0.81	0.87	0.87	0.9	0.88	0.86
eap	g	0.83	0.83	0.85	0.85	0.83	0.85	0.84
	α^+	0.87	0.86	0.87	0.87	0.85	0.88	0.9
	α^-	0.8	0.81	0.83	0.83	0.82	0.83	0.78
sick	g	0.93v	0.76*	0.86	0.86	0.87	0.8*	0.86
	α^+	0.95	0.98	0.82	0.82	0.88	0.73	0.94
	α^-	0.91	0.59	0.9	0.9	0.86	0.87	0.78
MEAN	g	0.76	0.74	0.75	0.75	0.75	0.73	0.71

TABLE IV

ACCURACY ON MAJORITY CLASS (α^+), ACCURACY ON MINORITY CLASS (α^-) AND GEOMETRIC MEAN (G) WITH C4.5 AS BASE CLASSIFIER

Data sets		ABKD	ThresC4.5	ReWC4.5	CostC4.5	Adabcos C4.5	Metacost C4.5	C4.5
breast-cancer	g	0.62v	0.45*	0.57v	0.5	0.56v	0.55v	0.5
	α^+	0.72	0.8	0.72	0.85	0.77	0.84	0.95
	α^-	0.53	0.25	0.45	0.3	0.41	0.36	0.26
credit-g	g	0.67v	0.64v	0.66v	0.61v	0.62v	0.64v	0.58
	α^+	0.7	0.7	0.67	0.82	0.81	0.76	0.85
	α^-	0.65	0.58	0.65	0.46	0.47	0.54	0.4
diabetes	g	0.7	0.7	0.72	0.72	0.67*	0.73v	0.7
	α^+	0.72	0.69	0.72	0.78	0.79	0.78	0.82
	α^-	0.69	0.71	0.73	0.67	0.57	0.67	0.6
haberman	g	0.61v	0.56v	0.63v	0.58v	0.57v	0.62v	0.52
	α^+	0.77	0.61	0.68	0.66	0.76	0.76	0.85
	α^-	0.49	0.51	0.58	0.51	0.43	0.52	0.32
hepatitis	g	0.79v	0.62v	0.73v	0.64v	0.7v	0.68v	0.58
	α^+	0.84	0.78	0.62	0.86	0.9	0.83	0.9
	α^-	0.75	0.49	0.85	0.48	0.55	0.56	0.37
ionosphere	g	0.9v	0.88	0.89	0.88	0.90	0.9	0.88
	α^+	0.96	0.95	0.94	0.94	0.94	0.98	0.94
	α^-	0.84	0.81	0.85	0.82	0.86	0.82	0.82
eap	g	0.83	0.69*	0.81	0.83	0.79*	0.82	0.83
	α^+	0.87	0.91	0.86	0.94	0.85	0.89	0.94
	$\alpha-$	0.8	0.53	0.77	0.74	0.74	0.76	0.74
sick	g	0.93	0.92	0.97v	0.96v	0.95	0.96v	0.93
	α^+	0.95	0.99	0.99	0.99	1	0.98	0.99
	α^-	0.91	0.85	0.95	0.92	0.9	0.95	0.87
MEAN	g	0.76	0.68	0.75	0.72	0.72	0.74	0.69

VI. CONCLUSION

Several aspects may influence the performance achieved by a classifier created by a Machine Learning

TABLE V

ACCURACY ON MAJORITY CLASS (α^+), ACCURACY ON MINORITY CLASS (α^-) AND GEOMETRIC MEAN (G) WITH 5NN AS BASE CLASSIFIER

Data sets		ABKD	Thres 5NN	ReW 5NN	Cost 5NN	Adabcos 5NN	Metacost 5NN	5NN
breast-cancer	g	0.62v	0.6v	0.62v	0.61v	0.61v	0.51v	0.45
	α^+	0.72	0.57	0.73	0.72	0.7	0.86	0.96
	α^-	0.53	0.63	0.52	0.52	0.53	0.3	0.21
credit-g	g	0.67v	0.59	0.66v	0.66v	0.63v	0.63v	0.57
	α^+	0.7	0.84	0.69	0.69	0.7	0.73	0.89
	α^-	0.65	0.42	0.63	0.63	0.56	0.55	0.37
diabetes	g	0.7	0.69	0.71v	0.71v	0.66	0.71v	0.68
	α^+	0.72	0.79	0.69	0.69	0.71	0.75	0.83
	α^-	0.69	0.61	0.74	0.74	0.62	0.68	0.56
haberman	g	0.61v	0.58v	0.57v	0.57v	0.53v	0.59v	0.39
	α^+	0.77	0.52	0.68	0.68	0.68	0.66	0.9
	α^-	0.49	0.65	0.47	0.47	0.41	0.52	0.17
hepatitis	g	0.79v	0.68	0.69v	0.73v	0.58*	0.8v	0.66
	α^+	0.84	0.91	0.79	0.85	0.8	0.84	0.94
	α^-	0.75	0.51	0.6	0.62	0.42	0.76	0.46
ionosphere	g	0.9v	0.82v	0.83v	0.83v	0.83v	0.79	0.78
	α^+	0.96	0.97	0.97	0.97	0.95	0.98	0.98
	α^-	0.84	0.7	0.71	0.71	0.72	0.63	0.62
eap	g	0.83v	0.79	0.8	0.8	0.78	0.77	0.78
	α^+	0.87	0.83	0.84	0.84	0.79	0.87	0.9
	α^-	0.8	0.75	0.76	0.76	0.77	0.69	0.68
sick	g	0.93v	0.62	0.84v	0.84v	0.87v	0.79v	0.61
	α^+	0.95	0.99	0.89	0.89	0.98	0.9	0.99
	α^-	0.91	0.39	0.79	0.79	0.77	0.7	0.37
MEAN	g	0.76	0.67	0.72	0.72	0.69	0.7	0.62

system. One of these aspects is related to the difference between the numbers of instances belonging to each class. When this difference is large, the learning system may have difficulties to learn the concept related to the minority class.

The problem of imbalanced data sets arises frequently. It is a problem in medical diagnosis, robotics, industrial production processes, communication network troubleshooting, machinery diagnosis, automated testing of electronic equipment, and many other areas. In this work, we survey some methods proposed by the Machine Learning community to solve the problem, we discuss some limitations of these methods and we propose a mixture of expert agents as a more effective solution to problem.

Our method allows improved identification of difficult small classes in predictive analysis, while keeping the classification ability of the other classes in an acceptable level. Furthermore, it is important to note that this method is not particularly computationally intensive. In particular, its computation costs are comparable to those of commonly used combination methods, such as Metacost. Thus, we demonstrate that it is possible and desirable to avoid complete commitment to a single best classifier during system construction. Instead, our approach can be used to build from the available classifiers a classification system that will perform best under any class distributions.

One of the most promising research lines refers to creating ensembles of classifiers by distributing the training set to reach balance in each of the resulting training samples. This involves a great variety of possibilities that we will cover in the near future. We will also examine the efficiency of our approach in larger and multi-class data sets.

REFERENCES

[1] Aha, D. (1997). Lazy Learning. Dordrecht: Kluwer Academic Publishers.

[2] Batista G., Carvalho A., Monard M. C. (2000), Applying One-sided Selection to Unbalanced Datasets. In O. Cairo, L. E. Sucar, and F. J. Cantu, editors, Proceedings of the Mexican International Conference on Artificial Intelligence – MICAI 2000, pages 315–325. Springer-Verlag.

[3] Blake, C., Keogh, E. & Merz, C.J. (1998). UCI Repository of machine learning databases `http://www.ics.uci.edu/~mlearn/MLRepository.html`. Irvine, CA: University of California.

[4] Chawla N., Bowyer K., Hall L., Kegelmeyer W. (2002), SMOTE: Synthetic Minority Over-sampling Technique, Journal of Artificial Intelligence Research 16, 321 - 357.

[5] Domingos P. (1998), How to get a free lunch: A simple cost model for machine learning applications. Proc. AAAI-98/ICML98, Workshop on the Methodology of Applying Machine Learning, pp1-7.

[6] Domingos P. & Pazzani M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29, 103-130.

[7] Domingos, P. (1999). MetaCost: A General Method for Making Classifiers Cost-Sensitive. Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining, 155-164. ACM Press.
[8] Fan, W., Stolfo, S.J., Zhang, J. & Chan, P.K. (1999). AdaCost: Misclassification costsensitive boosting. Proceedings of the Sixteenth International Conference on Machine Learning, 97-105. San Francisco: Morgan Kaufmann.
[9] Fawcett T. and Provost F. (1997), Adaptive Fraud Detection. Data Mining and Knowledge Discovery, 1(3):291–316.
[10] Friedman J. H. (1997), On bias, variance, 0/1-loss and curse-of-dimensionality. Data Mining and Knowledge Discovery, 1: 55-77.
[11] Japkowicz N. (2000), The class imbalance problem: Significance and strategies. In Proceedings of the International Conference on Artificial Intelligence, Las Vegas.
[12] Japkowicz N. and Stephen, S. (2002), The Class Imbalance Problem: A Systematic Study Intelligent Data Analysis, Volume 6, Number 5.
[13] Ji C. & Ma S. (1997), Combinations of weak classifiers. IEEE Transaction on Neural Networks, 8(1):32–42.
[14] Kotsiantis S., Pierrakeas C., Pintelas P. (2003), Preventing student dropout in distance learning systems using machine learning techniques, AI Techniques In Web-Based Educational Systems at Seventh International Conference on Knowledge-Based Intelligent Information & Engineering Systems, 3-5 September 2003
[15] Kubat, M. and Matwin, S. (1997), 'Addressing the Curse of Imbalanced Data Sets: One Sided Sampling', in the Proceedings of the Fourteenth International Conference on Machine Learning, pp. 179-186.
[16] Kubat, M., Holte, R. and Matwin, S. (1998), 'Machine Learning for the Detection of Oil Spills in Radar Images', *Machine Learning*, 30:195-215.
[17] Ling, C., & Li, C. (1998). Data Mining for Direct Marketing Problems and Solutions. In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98) New York, NY. AAAI Press.
[18] Mitchell, T. (1997) Machine Learning. McGraw Hill.
[19] Mladenic, D., & Grobelnik, M. (1999). Feature Selection for Unbalanced Class Distribution and Naive Bayes. In Proceedings of the 16th International Conference on Machine Learning., pp. 258–267. Morgan Kaufmann.
[20] Murthy (1998), Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey, Data Mining and Knowledge Discovery, 2, 345–389 (1998), Kluwer Academic Publishers.
[21] Provost, F. and Fawcett, T. (2001). Robust Classification for Imprecise Environments", Machine Learning, 42, 203–231.
[22] Quinlan J.R. (1993), C4.5: Programs for machine learning. Morgan Kaufmann, San Francisco.
[23] Schapire R., Singer Y. and Singhal A. (1998). Boosting and Rochhio applied to text filtering. In SIGIR'98.
[24] Singh M. and Provan G. (1996), Efficient learning of selective Bayesian network classifiers. In Proceedings of the 13th International Conference on Machine Learning (1996): 453-461, Bari, Italy.
[25] Chan Ph., Fan W., Prodromidis A., Stolfo, S. (1999), Distributed Data Mining in Credit Card Fraud Detection, IEEE Intelligent Systems on Data Mining, December 1999.
[26] Tjen-Sien Lim, Wei-Yin Loh, Yu-Shan Shih (2000), A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-Three Old and New Classification Algorithms. *Machine Learning*, 40, 203–228, 2000, Kluwer Academic Publishers.
[27] Witten, I. and Frank E. (2000) Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Mateo, CA, 2000.