RACOG and wRACOG: Two Probabilistic Oversampling Techniques

Barnan Das, Narayanan C. Krishnan and Diane J. Cook, *Fellow, IEEE*

Abstract—As machine learning techniques mature and are used to tackle complex scientific problems, challenges arise such as the imbalanced class distribution problem, where one of the target class labels is under-represented in comparison with other classes. Existing oversampling approaches for addressing this problem typically do not consider the probability distribution of the minority class while synthetically generating new samples. As a result, the minority class is not represented well which leads to high misclassification error. We introduce two probabilistic oversampling approaches, namely RACOG and wRACOG, to synthetically generating and strategically selecting new minority class samples. The proposed approaches use the joint probability distribution of data attributes and Gibbs sampling to generate new minority class samples. While RACOG selects samples produced by the Gibbs sampler based on a predefined *lag*, wRACOG selects those samples that have the highest probability of being misclassified by the existing learning model. We validate our approach using nine UCI datasets that were carefully modified to exhibit class imbalance and one new application domain dataset with inherent extreme class imbalance. In addition, we compare the classification performance of the proposed methods with three other existing resampling techniques.

Index Terms—Imbalanced class distribution, probabilistic oversampling, approximating joint probability distribution, Gibbs sampling.

1 Introduction

With a wide spectrum of industries drilling down into their data stores to collect previously unused data, data are being treated as the "new oil". Social media, healthcare, and e-commerce companies are refining and analyzing data with the goal of creating new products and/or adding value to existing ones. As machine learning techniques mature and are used to tackle scientific problems in the current deluge of data, a variety of challenges arise due to the nature of the underlying data. One well-recognized challenge is the imbalanced class distribution problem, where one of the target classes (the *minority* class) is under-represented in comparison with the other classes (the *majority* class or classes).

While many datasets do not contain a perfectly uniform distribution among class labels, some distributions contain less than 10% minority class data samples, which is considered imbalanced. Because the goal of supervised learning algorithms or *classifiers* is to optimize prediction accuracy for the entire data set, most approaches ignore performance on the individual class labels. Therefore, a *random classifier* that labels all data samples from an imbalanced class dataset as members of the majority class would become the highest performing algorithm despite incorrectly classifying all minority class samples. However, in many problem domains such as cancerous cell identification [1], oil-spill detection [2], fraud detection [3], keyword extraction [4], and text classification [5], identifying members of the minority class is critical, sometimes more so than achieving optimal overall accuracy for the majority class.

The imbalanced class distribution problem has vexed researchers for over a decade and has thus received focused attention. The common techniques that have been investigated for addressing this problem include: *Resampling*, which balances class priors of training data by either increasing the number of minority class data samples (*oversampling*) or decreasing the number of majority class data samples (*undersampling*); *cost-sensitive learning*, which assigns higher misclassfication cost for minority class samples than majority class; and *kernel-based learning* methods, which make the minority class samples more separable from the majority class by mapping the data to a high dimensional feature space. Among these techniques, resampling methods remain at the forefront due to their ease of implementation. Liu et al. [6] articulate a number of reasons to prefer resampling to other methods. First, because resampling occurs during preprocessing, the approach can be combined with others such as cost-sensitive learning, without changing the algorithmic anatomy [7]. Second, theoretical connections between resampling and cost-sensitive learning indicate that resampling can alter the misclassification costs of data points [8]. Third, empirical evidence demonstrates nearly identical performance between resampling and cost-

• *B. Das and D. J. Cook are with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164. N. C. Krishnan is with Indian Institute of Technology, Ropar, India*
E-mails: {barnandas, djcook}@wsu.edu, ckn@iitrpr.ac.in

sensitive learning techniques [9], [10]. Although both under- and over-sampling techniques have been improving over the years, we focus our attention on oversampling because it is well suited for the application that motivates our investigation. Because many class imbalance problems involve an absolute rarity of minority class data samples [11], undersampling of majority class examples is not advisable.

Most of the oversampling approaches [12], [13] that add synthetic data to alleviate class imbalance problem typically rely on spatial location of minority class samples in the Euclidean feature space. These approaches harvest *local* information of minority class samples to generate synthetic samples that are also assumed to belong to the minority class. Although this approach may be acceptable for data sets where a crisp decision boundary exists between the classes, spatial location-based synthetic oversampling is not suitable for data sets that have overlap between the minority and majority classes. Therefore, a better idea is to exploit *global* information of minority class samples, which can be done by considering the probability distribution of minority class while synthetically generating new minority class samples.

In this paper, we exploit the probability distribution of the minority class to introduce two oversampling approaches, namely, RACOG and wRACOG, to generating and strategically selecting new minority class data points. Specifically, the proposed algorithms optimally approximate n-dimensional discrete probability distribution of the minority class using Chow-Liu's dependence tree algorithm, and generates samples from this distribution using Gibbs sampling. Gibbs sampling generates a Markov chain of new minority class samples which undergo a strategic selection procedure. While RACOG selects samples from the Markov chain generated by the Gibbs sampler using a predefined *lag*, wRACOG selects those samples that have the highest probability of being misclassified by the existing learning model.

We validate our approach using nine UCI datasets, carefully modified to exhibit class imbalance, and one new application domain dataset[2] with inherent extreme class imbalance. The proposed methods, RACOG and wRACOG are compared with two existing oversampling techniques, SMOTE [12] and SMOTEBoost [13] and an undersampling technique RUSBoost [14]. We evaluate the alternative approaches using the performance measures Sensitivity, G-mean, and Area Under ROC Curve (AUC-ROC).

2 Related Work

Learning from class imbalanced datasets is a niche, yet critical area in supervised machine learning due to its increased prevalence in real world problem applications [15], [16], [17], [18]. Due to the pervasive nature of the imbalanced class problem, a wide spectrum of related techniques has been proposed. One of the most common solutions that has been investigated is cost sensitive learning (CSL). CSL methods counter the underlying assumption that all errors are equal by introducing customized costs for misclassifying data points. By assigning a sufficiently high cost to minority sample points, the algorithm may devote sufficient attention to these points to learn an effective class boundary. The effectiveness of CSL methods has been validated theoretically [8], [10] and empirically [9], [19], although other studies indicate that there is no clear winner between CSL and other methods such as resampling [20]. In addition, CSL concepts have been coupled with existing learning methods to boost their performance [21], [22], [23]. However, CSL approaches have drawbacks that limit their application. First, the misclassification costs are often unknown or need to be painstakingly determined for each application. Second, not all learning algorithms have cost sensitive implementation.

A second direction is to adapt the underlying classification algorithm to consider imbalanced classes, typically using kernel-based learning methods. Since kernel-based methods provide state-of-the-art techniques for many machine learning applications, using them to understand the imbalanced learning problem has attracted increased attention. The kernel classifier construction algorithm proposed by Hong et al. [24] is based on orthogonal forward selection and a regularized orthogonal weighted least squares (ROWLSs) estimator. Wu et al. [25] propose a kernel-boundary alignment (KBA) algorithm for adjusting the SVM class boundary. KBA is based on the idea of modifying the kernel matrix generated by a kernel function according to the imbalanced data distribution. Another interesting kernel modification technique is the k-category proximal support vector machine (PSVM) [26] proposed by Fung et al. This method transforms the soft-margin maximization paradigm into a simple system of k-linear equations for either linear or nonlinear classifiers.

Probably the most common approach, however, is to resample, or modify the dataset in a way that balances the class distribution. Determining the ideal class distribution is an open problem [21] and in most cases it is handled empirically. Naive resampling methods include oversampling the minority class by duplicating existing data points and undersampling the majority class by removing chosen data points. However, random over-sampling and under-sampling increases the possibility of overfitting and discarding useful information from the data, respectively.

An intelligent way of oversampling is to synthetically generate new minority class samples. Synthetic minority class oversampling technique, or SMOTE [12], has shown a great deal of success in various application domains. SMOTE oversamples the minority

2. http://ailab.wsu.edu/casas/datasets/prompting.zip

class by taking each minority class data point and introducing synthetic examples along the line segments joining any or all of the k-minority class nearest neighbors. In addition, adaptive synthetic sampling techniques have been introduced that take a more strategic approach to selecting the set of those minority class samples on which synthetic oversampling should be performed. For example, Borderline-SMOTE [27] generates synthetic samples only for those minority class examples that are "closer" to the decision boundary between the two classes. ADASYN [28], on the other hand, uses the density distribution of the minority class samples as a criterion to automatically decide the number of synthetic samples that need to be generated for each minority example by adaptively changing the weights of different minority examples to compensate for the skewed distribution. Furthermore, class imbalance problems associated with *intra-class* imbalanced distribution of data in addition to *inter-class* imbalance can be handled by cluster-based oversampling (CBO) proposed by Jo et al. [29].

The information loss incurred by random undersampling can be overcome by methods to strategically remove minority class samples. Liu et al. [30] proposed two informed undersampling techniques that use ensembles to learn from majority class subsets. For data sets that have overlapping minority and majority classes [31], data cleansing is used to minimize unwanted overlapping between classes by removing pairs of minimally distanced nearest neighbors of opposite classes, popularly known as Tomek links [32], [33], [34]. Because removing Tomek links changes the distribution of the data, it might not be beneficial in situations when there is a dense overlap between the classes, because this approach might lead to overfitting. There are additionally ensemble learning approaches [35] that combine the powers of resampling and ensemble learning. These approaches include, but are not restricted to, SMOTEBoost [13], RUSBoost [14], IVotes [36] and SMOTE-Bagging [37].

Empirical studies [20] have shown that approaches such as cost sensitive learning or kernel-based learning are not quite suitable for class imbalanced data sets that have "rare" minority class samples. Also, any form of under-sampling witnesses the same problem. We are interested in applying supervised learning techniques to a dataset that includes rare minority class samples. While oversampling is a natural solution to this problem, existing oversampling techniques such as SMOTE, Borderline-SMOTE and SMOTEBoost generate new data samples that are spatially close to existing minority class examples in the Euclidean space. Ideally new data samples should be representative of the entire minority class and not just be drawn from local information. Therefore, in this paper we focus on satisfying the criteria of *global* representation of the minority class by generating multivariate samples from the joint probability distribution of the underlying random variables or attributes.

Although not very popular, exploiting the probability distribution of the minority class for oversampling has been used in the past. Liu et. al. [6] proposed a generative oversampling technique that assumes a probability distribution for the minority class whose parameters are learned from the training samples. Artificial data points are added to the resampled data set by generating points from the learned probability distribution until the desired number of minority class points in the training set has been reached. The application domain under consideration is text classification and therefore the authors assume a multinomial distribution of the minority class. This approach would probably work well when the probability distribution is known or can at least be guessed from the data. However our approach is effective when the user is agnostic of the underlying distribution. Moreover, the authors report experimental results for SVM alone and thus it does not give a clear idea of whether their approach is effective for other classifiers or not. On the other hand, Gao et. al. [38] proposed an oversampling approach based on Parzen Windows or kernel density estimation of the minority class data samples. According to the estimated probability density function, synthetic instances are generated as the additional training data. Oversampling is done by treating a randomly-drawn minority class sample as the mean of a Gaussian distribution, and adding the product of a pseudo-random vector, smoothing parameter of the kernel function and Cholesky decomposition of the covariance matrix of the minority class, to the randomly-drawn minority class sample. Like Liu et. al's approach, this oversampling method is also highly tuned to a specific classifier, Radial Basis Function-based classifier in this case.

Thus, for most of the probabilistic oversampling techniques, we observe that specific classifiers are tuned to perform best with the oversampling approach. In contrast, we propose a purely preprocessing approach to probabilistic oversampling. With extensive experiments we validate that our approaches work well with four commonly used classifiers.

3 Application in Pervasive Computing

We are motivated to pursue this challenge by a problem in pervasive computing. Specifically, we are designing smart environments that perform health monitoring and assistance. Studies have shown that smart environment technologies can detect errors in activity completion and might be utilized to extend independent living in one's own home without compromising safety [39], [40]. One type of intervention that is valuable for individuals with cognitive impairment is automated prompts that aid with activity initiation and completion.

TABLE 1
Human Annotated Sensor Events

Date	Time	SensorID	Status	Annotation	Prompt
2009-05-11	14:59:54	D010	CLOSE	cook-3	*prompt*
2009-05-11	14:59:55	M017	ON	cook-4	*no-prompt*
2009-05-11	15:00:02	M017	OFF	none	*no-prompt*
2009-05-11	15:00:17	M017	ON	cook-8	*no-prompt*
2009-05-11	15:00:34	M018	ON	cook-8	*no-prompt*
2009-05-11	15:00:35	M051	ON	cook-8	*no-prompt*
2009-05-11	15:00:43	M016	ON	cook-8	*no-prompt*
2009-05-11	15:00:43	M015	ON	cook-9	*no-prompt*
2009-05-11	15:00:45	M014	ON	cook-9	*no-prompt*

Rule-based approaches to prompting individuals for activity initiation or completion have been developed by gerontechnology researchers [41], [42]. However, cognitive rehability theory indicates that combining prompting technology with knowledge of activity and cognitive context are valuable for effective health promotion [41], [43]. We postulate that prompt timing can be automated by incorporating contextual information provided by a smart home.

To determine the ability of a machine learning algorithm to generate appropriate activity-aware prompts, we performed a study in our smart home with 300 volunteer participants, aged 50+, who are healthy older adults or individuals with mild cognitive impairment. The smart home is a two-story apartment equipped with sensors that monitor motion, door open/shut status, and usage of water, burner, and specific items throughout the apartment.

Clinically-trained psychologists watch over a web camera as the participants perform 8 different activities. The psychology experimenters remotely issue a prompt when they determine that the individual is having difficulty initiating or completing an activity. Sensor events, denoted by the event date, time, sensor identifier, and message, are collected continuously, along with the prompt timings. A human annotator annotates the sensor events with corresponding activity and a sub-step that act as the ground truth. Table 1 shows a snippet of annotated events. On the basis of ground truth information, the feature vector for every activity sub-step present in the database is generated that consists of temporal, spatial and contextual features. We can view automated prompting as a supervised learning problem in which each activity step is mapped to a "prompt" or "no-prompt" class label. Thus, automated prompting emulates natural interventions provided by a caregiver.

The *prompting* dataset, as we would like to call it, has 3980 examples with 17 features. Out of the 17 features, 4 are categorical and the rest are numeric. The difficulty that is faced for the prompting problem is that the majority of activity steps are "no-prompt" cases and standard machine learning algorithms will likely map all data points to this class, which defeats the purpose of the intervention. Our goal is to design solutions to the class imbalance problem that improve sensitivity for this prompting application.

We evaluate the applicability of our algorithm for other datasets by evaluating them on nine additional real-world datasets from the UCI repository that exhibit the class imbalance problem: *abalone*, *car*, *nursery*, *letter*, *connect-4*, *ecoli1*, *haberman*, *yeast4* and *yeast6*. Characteristics of these datasets are summarized in Table 2. The real-valued attributes were transformed into discrete sets using equal-width binning. The multi-class datasets were converted into binary class by choosing a particular class label as *minority* class and the rest of the class labels together as *majority* class.

4 Probabilistic Approaches for Oversampling

In this paper, the proposed approach is based on the idea of using the probability distribution of the minority class to generate new minority class training samples. This way we can avoid the possibility of the synthetically generated training samples actually belonging to any other class in case of class overlap. In order to sample from the probability distribution of the minority class, we first need to estimate the distribution and then employ a mechanism to generate samples from that distribution. In the following, we describe how the probability distribution is estimated and new samples are generated.

4.1 Approximating Discrete Multivariate Probability Distribution using Dependence Tree

A central problem in many application domains, including ours, is to estimate the underlying n-dimensional probability distribution from a finite number of samples. Limitations on the available samples often require the distribution to be approximated by some simple assumptions, such as mutual independence or normality of the n-random variables under consideration. Chow-Liu proposed a notion of tree dependence to approximate the probability distribution [44]. Specifically, the proposed dependence tree approach optimally approximates an n-dimensional discrete probability distribution by a product of second order distributions, or the distribution of the first-order tree dependence. In other words, the objective is to find an optimal set of $n-1$ first order dependence relationship among the n variables.

If $P(x)$ is a joint probability distribution of n discrete variables $x_1, x_2, \ldots, x_n$, x denoting the n vector $(x_1, x_2, \ldots, x_n)$, a product approximation in which only the second order distributions are used can be represented as follows:

$$P(x) = \prod_{i}^{n} P(x_i|x_{j(i)}), 0 \leq j(i) < 1 \qquad (1)$$

Each variable in Equation 1 may be conditioned upon one of the other variables and thus $P(x_i|x_0)$ is

TABLE 2
Description of selected datasets

Dataset	Size	Dim	% Min. Class	Description
prompting	3,980	17	3.7437	Description in Section 3.
abalone	4,177	8	6.2006	Predicting age of large sea snails, abalone, from its physical measurements.
car	1,728	6	3.9931	Evaluating car acceptability based on price, technology and comfort.
nursery	12,960	8	2.5463	Nursery school application evaluation based on financial standing, parents' education and social health.
letter	20,000	16	3.7902	Classifying English alphabets using image frame features.
connect-4	5000	42	10.00	Predicting first player's outcome in connect-4 game given 8-ply positions information.
ecoli1	336	7	22.92	Classification of E.coli from other bacteria.
haberman	306	3	26.46	Survival of patients who had undergone surgery for breast cancer..
yeast4	1,484	8	3.44	Modified *yeast* dataset with ME2 as minority class and combination of other classes a majority class.
yeast6	1,484	8	2.36	Modified *yeast* dataset with EXC as minority class and combination of other classes a majority class.

by definition equal to $P(x_i)$. A probability distribution that can be represented as in Equation 1 is called a probability distribution of first-order tree dependence.

Thus, the objective of Chow-Liu's approximation method is as follows: given an n^{th}-order probability distribution $P(x_1, x_2, \ldots, x_n)$, x_i being discrete, find a distribution of tree dependence $P_\tau(x_1, x_2, \ldots, x_n)$ such that $I(P, P_\tau) \leq I(P, P_t)$ for all $t \in T_n$ where T_n is the set of all possible first-order dependence trees. The solution τ is called the optimal first-order dependence tree. $I(P, P')$ is a metric to measure the closeness of two probability distributions $P(x)$ and $P'(x)$ of n discrete variables, popularly known as the Kullback-Leibler divergence and is represented as follows:

$$I(P, P') = \sum_x P(x) log \frac{P(x)}{P'(x)} \tag{2}$$

As there could be n^{n-2} trees with n vertices, there will be an enormous number of dependence trees from which to identify the optimal tree structure. As a solution to this optimization problem, the optimal dependence tree is constructed using mutual information and a maximum-weighted tree. Specifically, the mutual information $I(x_i, x_j)$ between two variables x_i and x_j is assigned as the weight of the $(x_i \leftarrow x_j)$ branch in the dependence tree. Additionally, a maximum-weighted dependence tree is a dependence tree t such that for all t' in T_n

$$\sum_{i=1}^{n} I(x_i, x_{j(i)}) \geq \sum_{i=1}^{n} I(x_i, x_{j'(i)}) \tag{3}$$

A probability distribution of tree dependence $P_t(x)$ is an optimal approximation to $P(x)$ if and only if its dependence tree t has maximum weight. The mathematical proof behind this hypothesis is avoided due to space constraints. Thus the problem of finding the optimal first-order dependence tree is transformed to that of maximizing the total branch weight of a dependence tree. The algorithm for Chow-Liu's dependence tree construction is given in Figure 1.

Algorithm 1: Chow-Liu Dependence Tree Construction

1: Take a set of samples from the distribution that is being approximated as input.
2: On the basis of the input samples, compute all $n(n-1)/2$ pairwise mutual information $I(x_i, x_j), i = 1, 2, 3, ..., (n-1)$, $j = 2, 3, ..., n$, and $i < j$. $I(x_i, x_j)$ is the weight of the branch $(x_i \leftarrow x_j)$ in the dependence tree.
3: Use Kruskal's algorithm to construct a maximum weight dependence tree.

Fig. 1. Chow-Liu Dependence Tree Construction

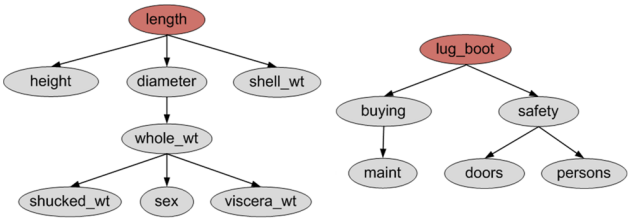

Fig. 2. Dependence Trees for (*left*) abalone and (*right*) car datasets

In this paper, we use Chow-Liu's algorithm to approximate the joint probability distribution of the minority class. This algorithm runs in $O(n^2 log(n))$ time where n is the dimension of the data. Sample dependence trees of *abalone* and *car* datasets are shown in Figure 2.

In order to generate new minority class samples from the approximated probability distribution, we need to use a sampling technique. In the current approach, we use Gibbs sampling as it works well with multivariate distributions.

4.2 Gibbs Sampling

Gibbs sampling is a Markov chain Monte Carlo (MCMC) method which originated with the Metropolis algorithm [45], [46]. It creates a Markov chain of random variables that converges to a target probability distribution. At each sampling step, the algorithm considers univariate conditional distribution of each dimension, where the value of a dimension depends

on the values of the other $n-1$ dimensions and the previous value of the same dimension. Such conditional distributions are easier to model than the full joint distribution. Figure 3 shows the algorithm for the standard Gibbs sampler. The univariate conditional distribution of each dimension represented by $P\left(X_i|x_1^{(t+1)},...,x_{i-1}^{(t+1)},x_{i+1}^{(t)},...,x_n^{(t)}\right)$ is used to choose attribute values that form a new synthetically generated sample.

Algorithm 1: Gibbs Sampler

1: $X^{(0)} =< x_1^{(0)},...,x_n^{(0)} >$
2: **for** $t = 1$ to T
3: **for** $i = 1$ to n
4: $x_i^{(t+1)} \sim P\left(X_i|x_1^{(t+1)},...,x_{i-1}^{(t+1)},x_{i+1}^{(t)},...,x_n^{(t)}\right)$

Fig. 3. Gibbs Sampling

The conditional distribution in Step 4 of the Gibbs sampling algorithm (Figure 3) is essentially sampling from a joint probability distribution. As discussed in Section 4.1, we approximate the joint probability distribution of the minority class using Chow-Liu's dependence tree represented in Equation 1. To generate a new minority class sample, the value of an attribute x_i represented by $x_i^{(t+1)}$ is determined by randomly sampling from the distribution of the state space (all possible values) of attribute x_i. Implementation of Gibbs sampling depends on two factors: ***burn-in*** and ***lag***. Burn-in is the number of sample generation iterations that are needed for the samples to reach a stationary distribution. Lag is the number of consecutive samples that are discarded from the Markov chain following each accepted sample to avoid autocorrelation between consecutive samples.

4.3 The RACOG Algorithm

We first describe the RACOG algorithm and eventually show what improvements have been incorporated in wRACOG. The **RA**pidy **CO**nverging **G**ibbs sampler (RACOG) uses Gibbs sampling to generate new minority class samples from the probability distribution of the minority class approximated using Chow-Liu algorithm. RACOG enhances standard Gibbs sampling by offering an alternative mechanism for choosing the initial values of the random variable X (denoted by $X^{(0)}$). Conventionally, the initial values of the random variable to "ignite" the Gibbs sampler are randomly chosen from the state space of the attributes. On the other hand, in RACOG, we choose the minority class data points as the set of initial samples and run the Gibbs sampler for every minority class sample. The total number of iterations for the Gibbs sampler is restricted by the desired *minority*:*majority* class distribution. Thus, RACOG produces multiple Markov chains, each starting with a different minority class sample, instead of one very long chain as done in conventional Gibbs sampling. As the initial samples

Algorithm 3: RACOG

1: **function** RACOG (*minority*, D, n, β, α, T)
Input: *minority* = minority class data points; D = size of *minority*; n = *minority* dimensions; β = burn-in period; α = lag; T = total number of iterations
Output: *new_samples* = new minority class samples
2: Construct Dependence tree DT using Chow-Liu algorithm.
3: **for** $d = 1$ to D **do**
4: $X^{(0)}$ = *minority(d)*
5: **for** $t = 1$ to T **do**
6: **for** $i = 1$ to n **do**
7: Simplify $P\left(X_i|x_1^{(t+1)},...,x_{i-1}^{(t+1)},x_{i+1}^{(t)},...,x_n^{(t)}\right)$ using DT
8: $x_i^{(t+1)} \sim P(S_i)$ where S_i is the state space of attribute x_i
9: **if** $t > \beta$ **AND** t **mod** $(\alpha) = 0$
10: *new_samples* = *new_samples* + $X^{(t)}$
11: **return** *new_samples*

Fig. 4. The RACOG Algorithm

of RACOG are chosen directly from the minority class samples, it helps in achieving faster convergence of the generated samples with the minority class distribution.

There are arguments in the literature about the pros and cons of a single long Markov chain versus multiple shorter chain approaches. Geyer [47] argues that a single long chain is a better approach because if long burn-in periods are required or if the chains have high autocorrelations, using a number of shorter chains may result in chains that are too short to adequately represent the minority class. However, a single long chain requires a very large number of iterations to converge with the target distribution. Our experiments show that the argument made by Geyer does not hold when multiple chains are generated using the minority class samples as the initial samples of the Gibbs sampler.

Figure 4 summarizes the RACOG algorithm that runs in $O(n^2log(n) + D.T.n)$ time where $n =$ dimension of data, $D =$ cardinality of minority class, and $T =$ predetermined number of iterations.

While the RACOG algorithm enhances the traditional Gibbs sampler by making it suitable for class imbalanced data, this approach does not take into account the usefulness of the generated samples. As a result, RACOG might add minority class samples that are redundant and have no contribution towards constructing a better hypothesis. In order to address this issue, we propose wRACOG, an enhancement to the RACOG algorithm.

4.4 The wRACOG algorithm

The enhanced RACOG algorithm, named wRACOG, is a wrapper-based technique over RACOG utilizing Gibbs sampling as the core data sampler. The purpose of introducing wRACOG is to get rid of *burn-in*, *lag*

and predefined number of iterations associated with sample selection in Gibbs sampling. By performing iterative training on the dataset with newly generated samples, wRACOG selects those samples from the Markov chain that have the highest probability of being misclassified by the learning model generated from the previous version of the dataset.

While the RACOG algorithm generates minority class samples for a fixed (predefined) number of iterations, the wRACOG algorithm keeps on fine tuning its hypothesis at every iteration by adding new samples until there is no further improvement with respect to a chosen performance measure. As our goal is to improve the performance of classifiers on the minority class, wRACOG keeps on adding new samples until there is no further improvement in *sensitivity* (true positive rate) of the *wrapper* classifier (the core classifier that retrains at every iteration) of wRACOG. This process acts as the "stopping criterion" for the wRACOG algorithm. Please note that *sensitivity* is not an invariant choice for the stopping criteria. Other performance measures, such as *precision*, *F-measure*, etc. can also be used and the choice is entirely application dependent.

At each iteration of wRACOG, new minority class samples are generated by the Gibbs sampler. The model learned by the *wrapper* classifier on the enhanced set of samples produced in the previous iteration is used to make predictions on the newly generated set of samples. During classification, the class labels of the new samples are assumed to be the same as the minority class because the Gibbs sampler produces new samples from the probability distribution of the minority class. Those samples that are misclassified by the model are added to the existing set of data samples and a new model is trained using the *wrapper* classifier. At each iteration, the trained model performs prediction on a held out *validation* set and the sensitivity of the model is recorded. Generation of new samples stops once the standard deviation of sensitivities over the past iterations falls below a threshold. As the wRACOG algorithm might end up running many iterations, the standard deviation of sensitivities is calculated over a fixed number of most recent iterations (*slide_win*). We use the values *slide_win=10* and *threshold=0.02* for our experiments, determined by performing an empirical study on the datasets described in this paper. The wRACOG algorithm is summarized in Figure 5. wRACOG runs in $O(n^2log(n) + B.D.n + B.W)$ time where, $n =$ dimension of data, $B =$ number of sample batches generated before stopping criteria is satisfied, $D =$ cardinality of minority class, and $W =$ time complexity of *wrapper* classifier.

Although wRACOG is similar to existing boosting techniques (such as AdaBoost) in the way misclassified samples are assigned higher weights to ensure their selection during random sampling in the next boosting iteration, there are a number of major differences. Firstly, while in traditional boosting, both training and prediction are performed on the same set of data samples, wRACOG trains the current hypothesis on the data samples from the previous iteration and performs prediction only on the newly generated samples from the Gibbs sampler. Secondly, there is no concept of changing weights of the samples before resampling, as the newly generated samples are directly added to the existing set of samples. Thirdly, wRACOG does not use weighted voting of multiple hypotheses learned at every iteration. Instead, it employs multiple iterations to fine tune a single hypothesis. We hypothesize that by applying this approach we can reduce the generation of redundant samples to converge more closely to the true distribution of the minority class, and also reduce the overhead of generating multiple hypotheses as is employed by traditional boosting techniques.

Algorithm 4: wRACOG

1: **function** wRACOG (*train*, *validation*, *wrapper*, *slide_win*, *threshold*, *slide_win*)
Input: *train* = training dataset enhanced at each iteration with new samples; *validation* = validation set on which trained model is tested at every iteration to track improvements; *wrapper* = classifier that is retrained on the enhanced dataset at every iteration; *slide_win* = sensitivities of previous iterations; *threshold* = threshold of standard deviation of sensitivities over *slide_win*
Output: *new_train* = final hypothesis encoded in the oversampled training set
2: Build *model* by training *wrapper* on *train*
3: Run Gibbs sampler on all minority class samples simultaneously
4: **do**
5: Perform prediction on newly generated samples using *model*
6: Add misclassified samples to form *new_train*
7: Train *model* on *new_train* using *wrapper*
8: Perform prediction on *validation* set using trained *model* and add sensitivity to *slide_win*
9: **while** ($\sigma(slide\_win) \geq threshold$)
10: **return** *new_train*

Fig. 5. The wRACOG Algorithm

5 Experimental Setup

We hypothesize that approximating the joint probability distribution of the minority class and using Gibbs sampling to synthetically generate minority class samples will yield improved results over existing resampling methods for problems that exhibit class imbalance. We compare the performance of the classifiers on datasets preprocessed by wRACOG, RACOG, and three well-known sampling techniques (SMOTE, SMOTEBoost, RUSBoost), against the datasets with no preprocessing (henceforth named *Baseline*). In the following we provide brief description of three well-known sampling techniques that have been used in our experiments.

SMOTE: SMOTE [12] oversamples the minority class by creating "synthetic" samples based on spatial location of the samples in the Euclidean space. Oversampling is performed by considering each minority class sample and introducing synthetic examples along the line segments joining any or all of the k-minority class nearest neighbors. The k-nearest neighbors are randomly chosen depending upon the amount of oversampling that is required. Synthetic samples are generated in the following way: First, the difference between the minority class sample under consideration and its nearest neighbor is computed. This difference is multiplied by a random number between 0 and 1, and it is added to the sample under consideration to form a new sample. Consequently, by adding diversity to the minority class, this approach forces the decision boundary between the two regions to be crisper. However, as SMOTE does not rely on the probability distribution of the minority class as a whole, there is no guarantee that the generated samples would belong to the minority class, especially when the samples from the majority and minority classes overlap [31]. We use SMOTE to produce a 50:50 distribution of minority and majority class samples, which is considered as near optimum [48].

SMOTEBoost: By using a combination of SMOTE and a standard boosting procedure, SMOTEBoost [13] models the minority class by providing the learner with misclassified minority class samples from the previous boosting iteration and a broader representation of those samples achieved by SMOTE. The inherent skewness in the updated distribution is rectified by introducing SMOTE to increase the number of minority class samples according to the new distribution. SMOTEBoost maximizes the margin of the skewed class dataset and increases the diversity among the classifiers in the ensemble by producing a unique set of synthetic samples at every iteration. Although iterative learning of the weak learner by the boosting procedure attempts to form a hypothesis which improves the classification of minority class samples, the quality of the generated samples is still dependent on the spatial location of minority class samples in the Euclidean space, as is done by SMOTE. Moreover, if SMOTEBoost executes SMOTE k times (for k boosting iterations), generating $k \times (\#majority\ class\ samples - \#minority\ class\ samples)$ samples is computationally expensive.

RUSBoost: RUSBoost [14] is very similar to SMOTEBoost, but claims to achieve better classification performance on the minority class samples by randomly under-sampling (RUS) the majority class. Although this method results in a simpler algorithm with a faster model training time, it fails to achieve desired performance (explained later in Section 6) as claimed by Seiffert et al., especially when the datasets have an absolute rarity of minority class samples. As most of the data sets under consideration have an absolute rarity of minority class samples, the class imbalance ratio has been set to 35:65 (minority:majority). This choice of class distribution is based on the empirical investigations performed by Khoshgoftaar et al. [49] which verify that a 35:65 class distribution would result in better classification performance than a 50:50 class distribution when samples from one class are extremely rare as compared to others when RUSBoost is used.

wRACOG and its predecessor RACOG are compared with the aforementioned alternative sampling techniques. RACOG oversamples the minority class to achieve a 50:50 class distribution. Therefore, the total number of iterations is fixed and is determined on the basis of $(\#majority\ class\ samples - \#minority\ class\ samples)$, *burn-in* and *lag*. A *burn-in* period of 100 and a *lag* of 20 iterations are chosen as the convention in the literature [50] to avoid autocorrelation among the samples generated by the Gibbs sampler. wRACOG, on the other hand, adds samples to the minority class until the standard deviation of sensitivity (true positive rate) over the 10 recent iterations falls below an empirically determined threshold. The classifiers presented in Table 3 are used as *wrapper* classifiers and are trained on an enhanced dataset at every iteration of wRACOG.

We choose four of the best-known classifiers in machine learning to evaluate the performance of the proposed method and other sampling approaches: decision tree, SVM, k-nearest neighbor and logistic regression. We performed parameter tuning of the chosen classifiers on the baseline data before preprocessing. The parameter values of the classifiers that performed best on the baseline datasets are used in conducting experiments with the existing and proposed sampling approaches. Table 3 lists parameter values for the corresponding classifiers.

TABLE 3
Classifiers and parameter values

Classifier	Parameter Values
C4.5 Decision Tree	Confidence factor = 2, Minimum # instances per leaf = 2
SVM	Kernel = RBF, RBF kernel $\gamma = 0.01$
k-Nearest Neighbor	k =5, Distance measure = Euclidean
Logistic Regression	Log likelihood ridge value = $1 \times 10^{(-8)}$

All experiments are performed using 5-fold cross validation where prediction is performed on unsampled held-out data. We report the following performance measures in this paper: Sensitivity (*true positive rate*), G-mean ($\sqrt{true\ positive\ rate \times true\ negative\ rate}$), and Area Under ROC Curve (AUC-ROC).

6 Results and Discussion

The primary limitation of classifiers that model imbalanced class datasets is in achieving desirable prediction accuracy for minority class instances. That is, the

sensitivity is typically low assuming that the minority class is represented as the positive class. The proposed approaches place emphasis on boosting the sensitivity of the classifiers while maintaining a strong prediction performance for both of the classes, which is measured by G-mean. However, we understand that the choice of performance measure that needs to be boosted when dealing with class imbalanced dataset is tightly coupled with the application domain. Moreover, finding a trade-off between improving classifier performance on the minority class in isolation and on the overall dataset should ideally be left at the discretion of the domain expert.

Due to space constraints, we represent the performance for only the C4.5 decision tree graphically. Performance of other classifiers are represented in Tables 5, 6 and 7. However, our discussion gives a generic idea of the performance of all classifiers considered in the current experiments.

We compare the sensitivity of the C4.5 decision tree on all six approaches in Figure 6. From the figure it is quite evident that both RACOG and wRACOG perform better than the other methods. Additionally, there is not much performance variability for RACOG and wRACOG over the 5-fold cross validation, except for the *connect-4* and *haberman* datasets. RUSBoost fails by performing nowhere close to the oversampling techniques. The poor performance of RUSBoost can be attributed to the rarity of minority class samples in the datasets under consideration. When the minority class samples are inherently rare, random under-sampling of a majority class to achieve a 35:65 class distribution (as done by RUSBoost) at each iteration of RUSBoost, makes the majority class samples rare as well. The poor performance of RUSBoost shows that it adds minor to no value over Baseline. Both SMOTE and SMOTEBoost are good contenders, although there is no clear winner between them. This might be because, as an implementation decision, we keep the number of boosting iterations of SMOTEBoost to be constant at 10. Seiffert et. al. suggest [14] that there is no significant improvement between 10 and 50 boosting iterations. We might get a better picture if the performance of SMOTEBoost is compared against variable number of boosting iterations. This analysis is not included in the current paper, but would be part of our future work. From a *sensitivity* standpoint, wRACOG performs better than RACOG on a majority of the datasets. For the rest of the datasets, the performance of wRACOG is at par with RACOG. We verify the statistical significance of the improvements using *Student's t-test*. RACOG and wRACOG exhibit significant ($p < 0.05$) performance improvement over SMOTE and SMOTEBoost. These improvements have been reported in **bold** in Tables 5, 6 and 7.

Because the sampling techniques boost the minority class, there is always a tendency for the false positive rate to increase. This trend is observed for most of the sampling techniques. However, the increase in false positive rate is not very significant and therefore it does not affect the G-mean scores. Figure 7 reports the G-mean scores of C4.5 decision tree on all the methods when tested with the ten datasets. Clearly, RACOG and wRACOG result in a superior performance over SMOTE and SMOTEBoost. An interesting thing to note is that the G-means of the decision tree due to wRACOG is 0 for the *haberman* dataset, implying that the false positive rate is increased to 1 when *haberman*

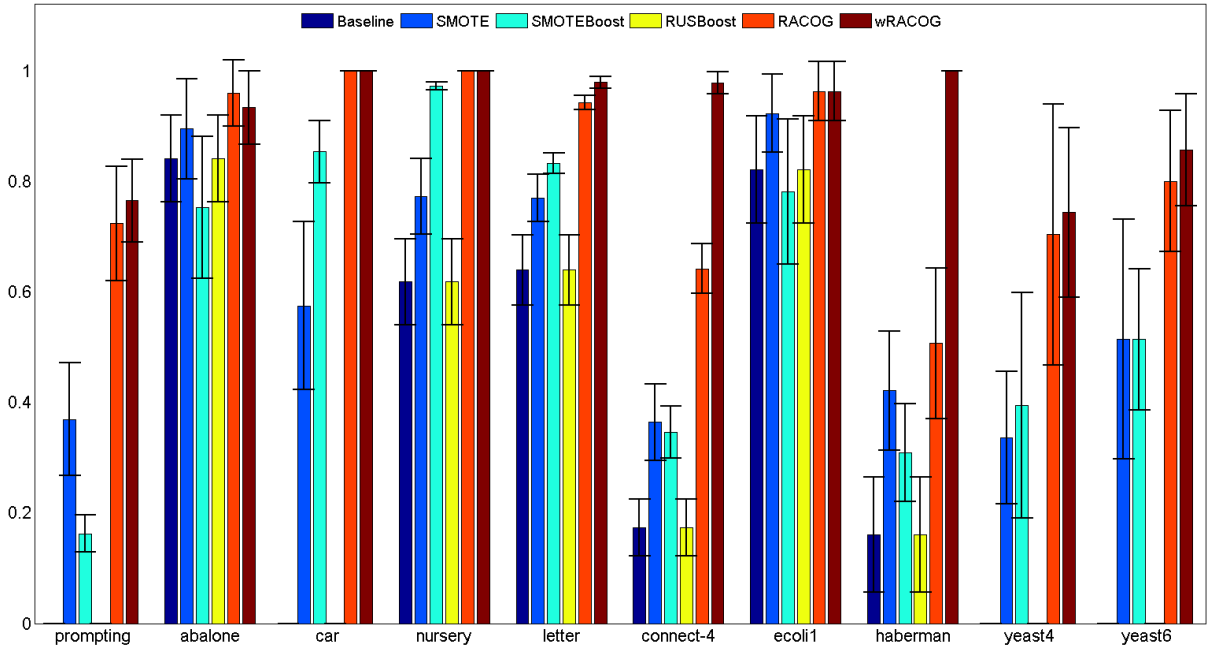

Fig. 6. Sensitivity for C4.5 Decision Tree

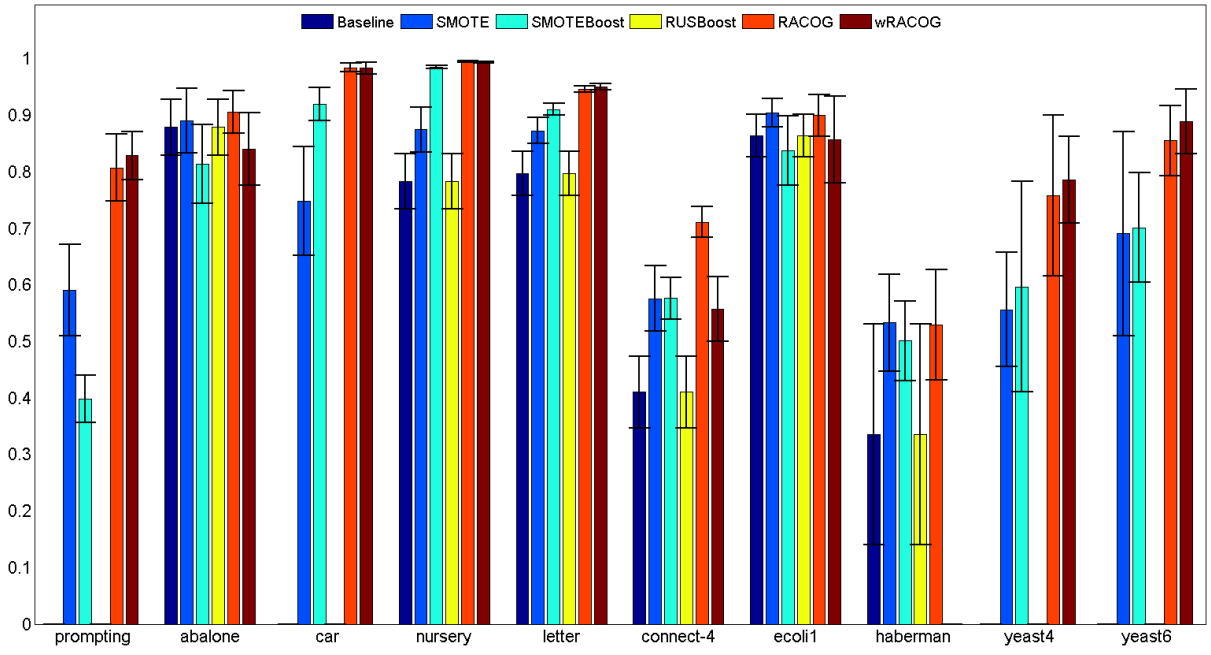

Fig. 7. G-mean for C4.5 Decision Tree

is preprocessed with wRACOG. This is likely due to the fact that *haberman* has only three attributes, which prevents correct approximation of the joint probability distribution.

Table 4 reports the AUC-ROC produced by different approaches on each of the 10 datasets when evaluated with a C4.5 decision tree. For the *prompting*, *car*, *yeast4* and *yeast6* datasets, Baseline and RUSBoost do not perform any better than random prediction. For the *letter* and *connect-4* datasets, the AUC-ROC for SMOTEBoost is higher than RACOG and wRACOG. However, no statistically significant improvement of RACOG and wRACOG was found over SMOTEBoost based on AUC-ROC. As there is no clear winner between RACOG and wRACOG on any of the performance measures, we do not conduct any statistical significance test between them.

The number of samples added to the baseline datasets by the different oversampling algorithms for achieving the reported performance is also an important parameter to analyze. Ideally, we would want to obtain a high performance by adding the least number of samples possible. Figure 8 illustrates the number of samples that are added by the different approaches. As the number of samples added by SMOTEBoost is far higher than other methods, we present log_{10} values of the *number of added samples* so that the comparison could be better represented in the plot. SMOTE and RACOG try to achieve a 50:50 class distribution and thus add samples accordingly. SMOTEBoost requires ten boosting iterations to produce ten hypotheses on which weighted voting is performed while predicting. The hypothesis learned at each iteration of SMOTEBoost has the form: $h_t : X \times Y \longrightarrow [0, 1]$, and therefore

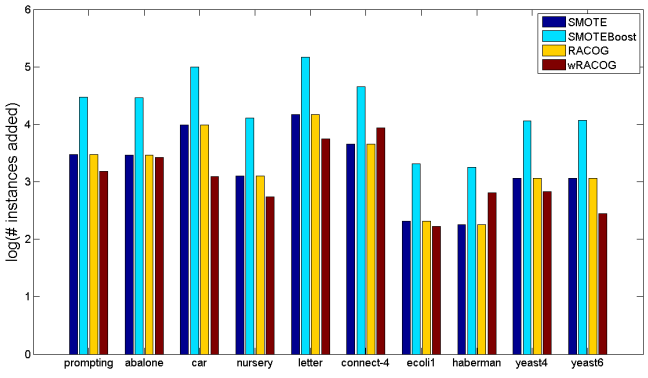

Fig. 8. Comparison of *log*(number of instances added) by different methods

stores the instances generated by SMOTE at every iteration. Hence, SMOTEBoost adds ten times the number of samples added by SMOTE. On the other hand, wRACOG requires a fraction ($\sim 56\%$) of the number of samples generated by SMOTE and RACOG, and ($\sim 5.6\%$) of the number of samples generated by SMOTEBoost, to obtain superior performance. We attribute this behavior to wRACOG's sample selection methodology which ensures diversity in the samples that are added.

7 Conclusion

In this paper, we propose two probabilistic oversampling algorithms for generating new minority class samples for class imbalanced datasets. The unknown probability distribution of the minority class is approximated using Chow-Liu's dependence tree algorithm. Although learning a Bayesian network on the

TABLE 4
AUC-ROC for C4.5 Decision Tree

Datasets	**Baseline**	**SMOTE**	**SMOTEBoost**	**RUSBoost**	**RACOG**	**wRACOG**
prompting	0.5 ± 0.00	0.8511 ± 0.05	0.8676 ± 0.02	0.5 ± 0.00	0.8707 ± 0.04	0.868 ± 0.03
abalone	0.9282 ± 0.05	0.9068 ± 0.05	0.923 ± 0.03	0.8802 ± 0.05	0.9057 ± 0.06	0.861 ± 0.03
car	0.5 ± 0.00	0.9526 ± 0.06	0.9944 ± 0.01	0.5 ± 0.00	0.9884 ± 0.01	0.9885 ± 0.01
nursery	0.9908 ± 0.01	0.9773 ± 0.02	0.9999 ± 0.00	0.807 ± 0.04	0.9976 ± 0.00	0.9967 ± 0.00
letter	0.8894 ± 0.03	0.9535 ± 0.01	0.9883 ± 0.01	0.8179 ± 0.03	0.9714 ± 0.00	0.9708 ± 0.01
connect-4	0.6651 ± 0.04	0.6836 ± 0.03	0.8196 ± 0.03	0.5801 ± 0.02	0.7544 ± 0.02	0.7104 ± 0.03
ecoli1	0.9337 ± 0.03	0.9318 ± 0.03	0.9384 ± 0.02	0.8677 ± 0.03	0.8983 ± 0.05	0.8798 ± 0.03
haberman	0.5211 ± 0.04	0.533 ± 0.10	0.5994 ± 0.06	0.5402 ± 0.03	0.5641 ± 0.09	0.5 ± 0.00
yeast4	0.5 ± 0.00	0.7309 ± 0.11	0.7962 ± 0.06	0.5 ± 0.00	0.7598 ± 0.18	0.7636 ± 0.13
yeast6	0.5 ± 0.00	0.8124 ± 0.11	0.8932 ± 0.06	0.5 ± 0.00	0.8519 ± 0.08	0.8741 ± 0.08

TABLE 5
Results for SVM

Sensitivity

Dataset	**Baseline**	**SMOTE**	**SMOTEBoost**	**RUSBoost**	**RACOG**	**wRACOG**
prompting	0.0268 ± 0.00	0.5971 ± 0.10	0.5196 ± 0.17	0.0268 ± 0.00	$\mathbf{0.693 \pm 0.08}$	$\mathbf{0.6826 \pm 0.08}$
abalone	0.00 ± 0.00	0.9482 ± 0.06	0.7812 ± 0.13	0.00 ± 0.00	0.9231 ± 0.07	$\mathbf{0.9733 \pm 0.06}$
car	0.00 ± 0.00	1.00 ± 0.00	0.9258 ± 0.13	0.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
nursery	0.00 ± 0.00	1.00 ± 0.00	0.9818 ± 0.03	0.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
letter	0.7336 ± 0.06	0.9248 ± 0.04	0.8747 ± 0.02	0.7336 ± 0.06	0.9221 ± 0.01	0.9143 ± 0.01
connect-4	0.00 ± 0.00	0.62 ± 0.07	0.562 ± 0.07	0.00 ± 0.00	$\mathbf{0.762 \pm 0.04}$	$\mathbf{0.986 \pm 0.02}$
ecoli1	0.00 ± 0.00	0.9365 ± 0.06	0.7529 ± 0.13	0.00 ± 0.00	0.9098 ± 0.06	$\mathbf{0.9882 \pm 0.03}$
haberman	0.5211 ± 0.00	0.533 ± 0.08	0.5994 ± 0.04	0.5402 ± 0.00	0.5641 ± 0.23	0.5 ± 0.00
yeast4	0.00 ± 0.00	0.5873 ± 0.18	0.4909 ± 0.19	0.00 ± 0.00	$\mathbf{0.7655 \pm 0.11}$	$\mathbf{0.7655 \pm 0.11}$
yeast6	0.00 ± 0.00	0.8 ± 0.16	0.6 ± 0.19	0.00 ± 0.00	$\mathbf{0.8571 \pm 0.14}$	$\mathbf{0.8571 \pm 0.14}$

G-mean

Dataset	**Baseline**	**SMOTE**	**SMOTEBoost**	**RUSBoost**	**RACOG**	**wRACOG**
prompting	0.1461 ± 0.00	0.7317 ± 0.05	0.686 ± 0.14	0.1452 ± 0.00	$\mathbf{0.7919 \pm 0.05}$	$\mathbf{0.7913 \pm 0.05}$
abalone	0.00 ± 0.00	0.8968 ± 0.04	0.8425 ± 0.06	0.00 ± 0.00	0.8844 ± 0.04	$\mathbf{0.8621 \pm 0.05}$
car	0.00 ± 0.00	0.9622 ± 0.01	0.9422 ± 0.07	0.00 ± 0.00	0.9622 ± 0.01	0.9622 ± 0.01
nursery	0.00 ± 0.00	0.976 ± 0.00	0.9788 ± 0.02	0.00 ± 0.00	0.9672 ± 0.00	0.9775 ± 0.00
letter	0.8553 ± 0.04	0.9424 ± 0.02	0.9241 ± 0.01	0.8553 ± 0.04	0.9364 ± 0.01	0.9394 ± 0.01
connect-4	0.00 ± 0.00	0.7372 ± 0.04	0.7063 ± 0.05	0.00 ± 0.00	$\mathbf{0.7877 \pm 0.01}$	0.585 ± 0.03
ecoli1	0.00 ± 0.00	0.8888 ± 0.04	0.8211 ± 0.07	0.00 ± 0.00	0.875 ± 0.05	0.8378 ± 0.10
haberman	0.00 ± 0.00	0.5032 ± 0.07	0.471 ± 0.03	0.00 ± 0.00	0.5586 ± 0.11	0.00 ± 0.00
yeast4	0.00 ± 0.00	0.7082 ± 0.13	0.6672 ± 0.12	0.00 ± 0.00	$\mathbf{0.8082 \pm 0.07}$	$\mathbf{0.8106 \pm 0.06}$
yeast6	0.00 ± 0.00	0.8721 ± 0.09	0.7568 ± 0.12	0.00 ± 0.00	$\mathbf{0.8853 \pm 0.07}$	$\mathbf{0.8999 \pm 0.08}$

AUC-ROC

Dataset	**Baseline**	**SMOTE**	**SMOTEBoost**	**RUSBoost**	**RACOG**	**wRACOG**
prompting	0.8651 ± 0.06	0.8585 ± 0.06	0.8323 ± 0.00	0.5126 ± 0.04	$\mathbf{0.8896 \pm 0.04}$	$\mathbf{0.8857 \pm 0.04}$
abalone	0.5 ± 0.00	0.8989 ± 0.04	0.9307 ± 0.05	0.5 ± 0.00	0.8864 ± 0.04	0.8707 ± 0.05
car	0.5 ± 0.00	0.963 ± 0.01	0.9862 ± 0.01	0.5 ± 0.00	0.963 ± 0.01	0.963 ± 0.01
nursery	0.5 ± 0.00	0.9763 ± 0.00	0.9948 ± 0.00	0.5 ± 0.00	0.9677 ± 0.00	0.9777 ± 0.00
letter	0.9878 ± 0.03	0.9861 ± 0.01	0.9825 ± 0.01	0.8654 ± 0.03	0.9856 ± 0.00	0.9861 ± 0.01
connect-4	0.5 ± 0.00	0.7497 ± 0.03	0.8253 ± 0.03	0.5 ± 0.00	$\mathbf{0.7884 \pm 0.01}$	0.6672 ± 0.02
ecoli1	0.5 ± 0.00	0.8911 ± 0.03	0.9153 ± 0.02	0.5 ± 0.00	0.8759 ± 0.05	0.8534 ± 0.08
haberman	0.5 ± 0.00	0.5852 ± 0.04	0.5684 ± 0.10	0.5 ± 0.00	0.5997 ± 0.06	0.5 ± 0.00
yeast4	0.5 ± 0.00	0.7291 ± 0.10	0.7937 ± 0.07	0.5 ± 0.00	$\mathbf{0.8109 \pm 0.06}$	$\mathbf{0.814 \pm 0.06}$
yeast6	0.5 ± 0.00	0.8796 ± 0.08	0.9272 ± 0.05	0.5 ± 0.00	0.8885 ± 0.07	0.9037 ± 0.07

minority class training examples is an alternative approach to approximate the probability distribution, we choose the dependence tree approach as it is less computationally expensive than learning a Bayesian network. Gibbs sampling is used as a sampling technique to synthetically generate new minority class samples from the approximated probability distribution. The proposed approaches, RACOG and wRACOG, differ in the sample selection strategy that is used to select samples from the Markov chain generated by the Gibbs sampler. RACOG runs the Gibbs sampler for a predetermined number of iterations and selects samples from the Markov chain using predefined *burn-in* and *lag*. On the other hand, wRACOG selects samples that have the highest probability of being misclassified by the existing learning model. It keeps on adding samples unless there is no further improvement in sensitivity over a predefined number of most recent iterations.

Experiments with RACOG and wRACOG on a wide variety of datasets and classifiers indicate that the algorithm is able to attain higher sensitivity than other methods, while maintaining higher G-mean. This supports our hypotheses that generating new

TABLE 6
Results for k-Nearest Neighbor

Sensitivity

Dataset	Baseline	SMOTE	SMOTEBoost	RUSBoost	RACOG	wRACOG
prompting	0.0268 ± 0.02	0.5971 ± 0.07	0.5256 ± 0.12	0.0268 ± 0.02	$\mathbf{0.693 \pm 0.14}$	$\mathbf{0.6826 \pm 0.11}$
abalone	0.7631 ± 0.11	0.9365 ± 0.04	0.8447 ± 0.07	0.7631 ± 0.11	0.9482 ± 0.06	0.9482 ± 0.06
car	0.0579 ± 0.06	0.7991 ± 0.09	0.8724 ± 0.07	0.0579 ± 0.06	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
nursery	0.303 ± 0.02	0.9061 ± 0.02	0.9576 ± 0.02	0.303 ± 0.02	1.00 ± 0.00	1.00 ± 0.00
letter	0.7336 ± 0.02	0.9248 ± 0.01	0.8747 ± 0.03	0.7336 ± 0.02	0.9221 ± 0.02	0.9143 ± 0.02
connect-4	0.036 ± 0.01	0.722 ± 0.03	0.484 ± 0.07	0.036 ± 0.01	0.622 ± 0.05	$\mathbf{0.904 \pm 0.04}$
ecoli1	0.6847 ± 0.12	0.8965 ± 0.06	0.7945 ± 0.14	0.6847 ± 0.12	$\mathbf{0.9216 \pm 0.06}$	$\mathbf{0.9482 \pm 0.06}$
haberman	0.061 ± 0.04	0.4199 ± 0.17	0.2956 ± 0.08	0.061 ± 0.04	0.3699 ± 0.08	$\mathbf{1.00 \pm 0.00}$
yeast4	0.0382 ± 0.05	0.5091 ± 0.12	0.5855 ± 0.12	0.0382 ± 0.05	$\mathbf{0.7255 \pm 0.11}$	$\mathbf{0.7618 \pm 0.14}$
yeast6	0.0857 ± 0.08	0.8286 ± 0.12	0.6 ± 0.19	0.0857 ± 0.08	$\mathbf{0.9143 \pm 0.13}$	0.8571 ± 0.10

G-mean

Dataset	Baseline	SMOTE	SMOTEBoost	RUSBoost	RACOG	wRACOG
prompting	0.1461 ± 0.08	0.7317 ± 0.04	0.6895 ± 0.08	0.1461 ± 0.08	$\mathbf{0.7919 \pm 0.08}$	$\mathbf{0.7907 \pm 0.07}$
abalone	0.8389 ± 0.05	0.8957 ± 0.03	0.8592 ± 0.04	0.8389 ± 0.05	0.8955 ± 0.05	0.8536 ± 0.06
car	0.1824 ± 0.18	0.8237 ± 0.05	0.898 ± 0.04	0.1824 ± 0.18	0.8742 ± 0.01	$\mathbf{0.9458 \pm 0.01}$
nursery	0.5503 ± 0.02	0.9157 ± 0.01	0.9592 ± 0.01	0.5503 ± 0.02	0.9178 ± 0.00	0.9828 ± 0.00
letter	0.8553 ± 0.01	0.9424 ± 0.01	0.9241 ± 0.02	0.8553 ± 0.01	0.9364 ± 0.01	0.9394 ± 0.01
connect-4	0.1885 ± 0.02	0.7118 ± 0.01	0.6534 ± 0.04	0.1885 ± 0.02	0.7209 ± 0.03	0.6739 ± 0.03
ecoli1	0.7926 ± 0.07	0.8719 ± 0.03	0.8251 ± 0.08	0.7926 ± 0.07	0.8803 ± 0.05	0.8684 ± 0.05
haberman	0.2178 ± 0.13	0.5289 ± 0.09	0.482 ± 0.07	0.2178 ± 0.13	0.4831 ± 0.07	0.1671 ± 0.22
yeast4	0.1232 ± 0.17	0.6514 ± 0.08	0.737 ± 0.08	0.1232 ± 0.17	0.7653 ± 0.05	$\mathbf{0.807 \pm 0.07}$
yeast6	0.2265 ± 0.21	0.879 ± 0.06	0.7434 ± 0.12	0.2265 ± 0.21	$\mathbf{0.8932 \pm 0.06}$	$\mathbf{0.8923 \pm 0.06}$

AUC-ROC

Dataset	Baseline	SMOTE	SMOTEBoost	RUSBoost	RACOG	wRACOG
prompting	0.8651 ± 0.04	0.8585 ± 0.04	0.8323 ± 0.05	0.5126 ± 0.01	$\mathbf{0.8854 \pm 0.04}$	$\mathbf{0.8843 \pm 0.05}$
abalone	0.9361 ± 0.03	0.928 ± 0.03	0.903 ± 0.04	0.8449 ± 0.05	0.9295 ± 0.04	0.9188 ± 0.04
car	0.9859 ± 0.01	0.9523 ± 0.02	0.9698 ± 0.01	0.529 ± 0.03	0.9951 ± 0.00	0.993 ± 0.01
nursery	0.9999 ± 0.00	0.9848 ± 0.00	0.9944 ± 0.00	0.6515 ± 0.01	0.9995 ± 0.00	0.9992 ± 0.00
letter	0.9878 ± 0.01	0.9861 ± 0.00	0.9825 ± 0.01	0.8654 ± 0.01	0.9856 ± 0.00	0.9861 ± 0.00
connect-4	0.7895 ± 0.02	0.7893 ± 0.01	0.7849 ± 0.04	0.5173 ± 0.00	0.8187 ± 0.02	0.8098 ± 0.04
ecoli1	0.9236 ± 0.02	0.9206 ± 0.02	0.9063 ± 0.03	0.8037 ± 0.06	0.9128 ± 0.03	0.9151 ± 0.05
haberman	0.5634 ± 0.09	0.5774 ± 0.07	0.572 ± 0.08	0.5261 ± 0.02	0.5678 ± 0.09	$\mathbf{0.6046 \pm 0.11}$
yeast4	0.8287 ± 0.08	0.8038 ± 0.07	0.8286 ± 0.07	0.5174 ± 0.03	$\mathbf{0.8517 \pm 0.06}$	$\mathbf{0.8658 \pm 0.06}$
yeast6	0.8932 ± 0.09	0.8931 ± 0.09	0.8838 ± 0.08	0.5422 ± 0.04	$\mathbf{0.9517 \pm 0.03}$	$\mathbf{0.954 \pm 0.03}$

samples by considering the global distribution of minority class points is a good approach for dealing with class imbalance. The motivation to focus mainly on improving sensitivity comes from the our application domain in pervasive computing. However, we ensure that the performance of the classifiers on both the classes is not hampered.

An open question that still remains is to determine how the proposed approaches and alternative sampling techniques compare with random oversampling and undersampling. Counter-intuitive to the widely accepted notion that random oversampling and undersampling perform worse than sophisticated sampling methods for imbalanced class distribution, our brief experiments show that these approaches perform at par, and sometimes even better than other sampling approaches discussed in this paper. However, random oversampling and undersampling are inherently different from the sampling approaches discussed in this paper because they duplicate training samples rather than creating synthetic samples, which makes the comparison difficult. In our future work, we will investigate this issue and for which we might need a more extensive experimental setup that includes more datasets with high dimensions and very high cardinality. In addition, we will also examine the sampling approaches for handling continuous-valued variables.

ACKNOWLEDGMENTS

We would like to thank Maureen Schmitter-Edgecombe and members of the Clinical Psychology department for their help in gathering *prompting* data for our experiments. This material is based upon work supported by the National Science Foundation under grant 1064628 and by the National Institutes of Health under application R01EB009675.

REFERENCES

[1] K. Woods, C. Doss, K. Bowyer, J. Solka, C. Priebe, and K. W., "Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 6, pp. 1417–1436, 1993.

[2] M. Kubat, R. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Machine learning*, vol. 30, no. 2, pp. 195–215, 1998.

[3] C. Phua, D. Alahakoon, and V. Lee, "Minority report in fraud detection: classification of skewed data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 50–59, 2004.

TABLE 7
Results for Logistic Regression

Sensitivity

Dataset	Baseline	SMOTE	SMOTEBoost	RUSBoost	RACOG	wRACOG
prompting	0.088 ± 0.04	0.5488 ± 0.10	0.3183 ± 0.15	0.088 ± 0.04	0.4738 ± 0.09	0.4867 ± 0.10
abalone	0.8314 ± 0.06	0.8565 ± 0.09	0.7898 ± 0.09	0.8314 ± 0.06	0.8565 ± 0.12	$\mathbf{0.9082 \pm 0.08}$
car	0.2896 ± 0.06	0.9692 ± 0.07	0.8335 ± 0.10	0.2896 ± 0.06	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
nursery	0.7576 ± 0.07	0.9606 ± 0.02	0.9849 ± 0.02	0.7576 ± 0.07	1.00 ± 0.00	1.00 ± 0.00
letter	0.6437 ± 0.03	0.8943 ± 0.03	0.8983 ± 0.02	0.6437 ± 0.03	0.922 ± 0.03	$\mathbf{0.9485 \pm 0.02}$
connect-4	0.32 ± 0.06	0.62 ± 0.05	0.586 ± 0.05	0.32 ± 0.06	$\mathbf{0.81 \pm 0.02}$	$\mathbf{0.988 \pm 0.01}$
ecoli1	0.7882 ± 0.13	0.8267 ± 0.15	0.7529 ± 0.17	0.7882 ± 0.13	0.84 ± 0.12	$\mathbf{0.9333 \pm 0.05}$
haberman	0.2221 ± 0.07	0.4081 ± 0.07	0.3963 ± 0.08	0.2221 ± 0.07	$\mathbf{0.4706 \pm 0.08}$	$\mathbf{0.9375 \pm 0.06}$
yeast4	0.1764 ± 0.15	0.5291 ± 0.16	0.4109 ± 0.12	0.1764 ± 0.15	$\mathbf{0.7273 \pm 0.19}$	$\mathbf{0.7473 \pm 0.20}$
yeast6	0.4 ± 0.19	0.5714 ± 0.23	0.5714 ± 0.14	0.4 ± 0.19	$\mathbf{0.7714 \pm 0.16}$	$\mathbf{0.7714 \pm 0.19}$

G-mean

Dataset	Baseline	SMOTE	SMOTEBoost	RUSBoost	RACOG	wRACOG
prompting	0.288 ± 0.07	0.7062 ± 0.07	0.5393 ± 0.13	0.288 ± 0.07	0.6649 ± 0.06	0.6714 ± 0.06
abalone	0.8659 ± 0.03	0.8691 ± 0.05	0.849 ± 0.05	0.8659 ± 0.03	0.8638 ± 0.06	0.8404 ± 0.05
car	0.5314 ± 0.06	0.9566 ± 0.03	0.8898 ± 0.05	0.5314 ± 0.06	$\mathbf{0.9713 \pm 0.00}$	$\mathbf{0.971 \pm 0.00}$
nursery	0.8677 ± 0.04	0.9696 ± 0.01	0.9814 ± 0.01	0.8677 ± 0.04	0.9872 ± 0.00	0.989 ± 0.00
letter	0.7996 ± 0.02	0.9219 ± 0.01	0.9245 ± 0.01	0.7996 ± 0.02	0.9287 ± 0.01	0.933 ± 0.01
connect-4	0.5571 ± 0.05	0.7363 ± 0.04	0.7202 ± 0.03	0.5571 ± 0.05	$\mathbf{0.8014 \pm 0.02}$	0.6316 ± 0.06
ecoli1	0.8521 ± 0.08	0.8578 ± 0.09	0.8176 ± 0.10	0.8521 ± 0.08	$\mathbf{0.8658 \pm 0.08}$	$\mathbf{0.8672 \pm 0.03}$
haberman	0.4446 ± 0.07	0.53 ± 0.06	0.5306 ± 0.07	0.4446 ± 0.07	0.547 ± 0.06	0.1725 ± 0.17
yeast4	0.3633 ± 0.23	0.6879 ± 0.10	0.6178 ± 0.09	0.3633 ± 0.23	$\mathbf{0.7798 \pm 0.10}$	$\mathbf{0.7969 \pm 0.09}$
yeast6	0.6185 ± 0.13	0.7364 ± 0.15	0.7431 ± 0.09	0.6185 ± 0.13	$\mathbf{0.8421 \pm 0.08}$	$\mathbf{0.8453 \pm 0.10}$

AUC-ROC

Dataset	Baseline	SMOTE	SMOTEBoost	RUSBoost	RACOG	wRACOG
prompting	0.846 ± 0.01	0.8413 ± 0.03	0.8479 ± 0.01	0.5396 ± 0.02	0.8264 ± 0.02	0.835 ± 0.02
abalone	0.9213 ± 0.03	0.9303 ± 0.03	0.9252 ± 0.03	0.8672 ± 0.03	0.9198 ± 0.04	0.8924 ± 0.04
car	0.9746 ± 0.01	0.9759 ± 0.00	0.9705 ± 0.01	0.637 ± 0.03	0.9751 ± 0.01	0.9778 ± 0.00
nursery	0.9962 ± 0.00	0.9955 ± 0.00	0.9942 ± 0.00	0.8765 ± 0.03	0.9959 ± 0.00	0.9958 ± 0.00
letter	0.9814 ± 0.00	0.9772 ± 0.00	0.9726 ± 0.00	0.8187 ± 0.02	0.9804 ± 0.01	0.9792 ± 0.00
connect-4	0.8829 ± 0.02	0.862 ± 0.01	0.8293 ± 0.01	0.6482 ± 0.03	0.8799 ± 0.02	$\mathbf{0.8778 \pm 0.02}$
ecoli1	0.931 ± 0.04	0.9072 ± 0.05	0.9221 ± 0.03	0.857 ± 0.08	0.9227 ± 0.05	0.9203 ± 0.03
haberman	0.6045 ± 0.10	0.5801 ± 0.08	0.564 ± 0.10	0.5666 ± 0.04	0.5961 ± 0.07	0.5507 ± 0.06
yeast4	0.7984 ± 0.12	0.7384 ± 0.09	0.7818 ± 0.07	0.5851 ± 0.07	0.8047 ± 0.12	0.8053 ± 0.12
yeast6	0.9097 ± 0.04	0.9007 ± 0.05	0.8349 ± 0.09	0.6969 ± 0.09	0.9037 ± 0.06	0.8969 ± 0.07

[4] P. Turney *et al.*, "Learning algorithms for keyphrase extraction," *Information Retrieval*, vol. 2, no. 4, pp. 303–336, 2000.

[5] D. Lewis and W. Gale, "A sequential algorithm for training text classifiers," in *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Springer-Verlag New York, Inc., 1994, pp. 3–12.

[6] A. Liu, J. Ghosh, and C. Martin, "Generative oversampling for mining imbalanced datasets," in *Proceedings of the 2007 International Conference on Data Mining, DMIN*, 2007, pp. 25–28.

[7] B. Zadrozny, J. Langford, and N. Abe, "Cost-sensitive learning by cost-proportionate example weighting," in *Third IEEE International Conference on Data Mining, 2003. ICDM 2003.* IEEE, 2003, pp. 435–442.

[8] C. Elkan, "The foundations of cost-sensitive learning," in *International Joint Conference on Artificial Intelligence*, vol. 17, no. 1. Lawrence Erlbaum Associates Ltd., 2001, pp. 973–978.

[9] K. McCarthy, B. Zabar, and G. Weiss, "Does cost-sensitive learning beat sampling for classifying rare classes?" in *Proceedings of the 1st international workshop on Utility-based data mining*. ACM, 2005, pp. 69–77.

[10] M. Maloof, "Learning when data sets are imbalanced and when costs are unequal and unknown," in *ICML-2003 Workshop on Learning from Imbalanced Data Sets II*, 2003.

[11] G. Weiss, "Mining with rarity: a unifying framework," *SigKDD Explorations*, vol. 6, no. 1, pp. 7–19, 2004.

[12] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, June 2002.

[13] N. Chawla, A. Lazarevic, L. Hall, and K. Bowyer, "SMOTEBoost: Improving prediction of the minority class in boosting," *Knowledge Discovery in Databases: PKDD 2003*, pp. 107–119, 2003.

[14] C. Seiffert, T. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSBoost: A hybrid approach to alleviating class imbalance," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 40, no. 1, pp. 185–197, 2010.

[15] N. Chawla, N. Japkowicz, and A. Kotcz, "Editorial: special issue on learning from imbalanced data sets," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 1–6, 2004.

[16] H. He and E. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[17] Y. Sun, A. Wong, and M. Kamel, "Classification of imbalanced data: A review," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 4, p. 687, 2009.

[18] N. Chawla, "Data mining for imbalanced datasets: An overview," *Data Mining and Knowledge Discovery Handbook*, pp. 875–886, 2010.

[19] X. Liu and Z. Zhou, "The influence of class imbalance on cost-sensitive learning: an empirical study," in *Data Mining, 2006. ICDM'06. Sixth International Conference on*. IEEE, 2006, pp. 970–974.

[20] G. Weiss, K. McCarthy, and B. Zabar, "Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs," in *International Conference on Data Mining*, 2007, pp. 35–41.

[21] F. Provost, T. Fawcett, and R. Kohavi, "The case against accuracy estimation for comparing induction algorithms," in *Proceedings of the Fifteenth International Conference on Machine Learning*, vol. 445, 1998.

[22] C. Drummond and R. Holte, "Exploiting the cost (in) sensitivity of decision tree splitting criteria," in *Machine Learning-International Workshop then Conference*, 2000, pp. 239–246.

[23] M. Kukar and I. Kononenko, "Cost-sensitive learning with neural networks," in *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)*. Citeseer, 1998, pp. 445–449.
[24] X. Hong, S. Chen, and C. Harris, "A kernel-based two-class classifier for imbalanced data sets," *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 28–41, 2007.
[25] G. Wu and E. Chang, "Kba: Kernel boundary alignment considering imbalanced data distribution," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 786–795, 2005.
[26] G. Fung and O. Mangasarian, "Multicategory proximal support vector machine classifiers," *Machine Learning*, vol. 59, no. 1, pp. 77–97, 2005.
[27] H. Han, W. Wang, and B. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," *Advances in Intelligent Computing*, pp. 878–887, 2005.
[28] H. He, Y. Bai, E. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *IEEE International Joint Conference on Neural Networks, 2008 (IEEE World Congress on Computational Intelligence)*. IEEE, 2008, pp. 1322–1328.
[29] T. Jo and N. Japkowicz, "Class imbalances versus small disjuncts," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 40–49, 2004.
[30] X. Liu, J. Wu, and Z. Zhou, "Exploratory undersampling for class-imbalance learning," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 39, no. 2, pp. 539–550, 2009.
[31] V. García, R. Mollineda, and J. Sánchez, "On the k-nn performance in a challenging scenario of imbalance and overlapping," *Pattern Analysis & Applications*, vol. 11, no. 3, pp. 269–280, 2008.
[32] I. Tomek, "Two modifications of cnn," *IEEE Transaction on Systems, Man and Cybernetics*, vol. 6, pp. 769–772, 1976.
[33] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: one-sided selection," in *Machine Learning-International Workshop then Conference*. Morgan Kaufmann Publishers, Inc., 1997, pp. 179–186.
[34] G. Batista, R. Prati, and M. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, 2004.
[35] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man and Cybernetics-Part : Applications and Reviews*, vol. 42, no. 4, pp. 463–484, July 2012.
[36] J. Błaszczyński, M. Deckert, J. Stefanowski, and S. Wilk, "Integrating selective pre-processing of imbalanced data with ivotes ensemble," in *Rough Sets and Current Trends in Computing*. Springer, 2010, pp. 148–157.
[37] S. Wang and X. Yao, "Diversity analysis on imbalanced data sets by using ensemble models," in *Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on*. IEEE, 2009, pp. 324–331.
[38] M. Gao, X. Hong, S. Chen, and C. J. Harris, "Probability density function estimation based over-sampling for imbalanced two-class problems," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*. IEEE, 2012, pp. 1–8.
[39] D. Cook, M. Schmitter-Edgecombe *et al.*, "Assessing the quality of activities in a smart environment," *Methods of Information in Medicine*, vol. 48, no. 5, p. 480, 2010.
[40] F. Doctor, H. Hagras, and V. Callaghan, "A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 35, no. 1, pp. 55–65, 2005.
[41] M. Lawton and L. Nahemow, *Environment and aging theory: A focus on aging*, R. Scheidt and P. Windley, Eds. Greenwood Press, 1998.
[42] A. Mihailidis, G. Fernie, and J. Barbenel, "The use of artificial intelligence in the design of an intelligent cognitive orthosis for people with dementia," *Assistive Technology*, vol. 13, no. 1, pp. 23–39, 2001.
[43] A. M. Seelye, S.-E. M., B. Das, and D. J. Cook, "Using cognitive rehabilitation theory to inform the development of smart prompting technologies," *Reviews in Biomedical Engineering*, 2012, to appear.
[44] C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees," *Information Theory, IEEE Transactions on*, vol. 14, no. 3, pp. 462–467, 1968.
[45] N. Meteopolis and S. Ulam, "The monte carlo method," *Journal of the American statistical association*, vol. 44, no. 247, pp. 335–341, 1949.
[46] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, p. 1087, 1953.
[47] C. Geyer, "Practical markov chain monte carlo," *Statistical Science*, vol. 7, no. 4, pp. 473–483, 1992.
[48] G. Weiss and F. Provost, "Learning when training data are costly: The effect of class distribution on tree induction," *Journal of Artificial Intelligence Research (JAIR)*, vol. 19, pp. 315–354, 2003.
[49] T. Khoshgoftaar, C. Seiffert, J. Van Hulse, A. Napolitano, and A. Folleco, "Learning with limited minority class data," in *Machine Learning and Applications, 2007. ICMLA 2007. Sixth International Conference on*. IEEE, 2007, pp. 348–353.
[50] A. Raftery and S. Lewis, "How many iterations in the gibbs sampler," *Bayesian statistics*, vol. 4, no. 2, pp. 763–773, 1992.

Barnan Das is a PhD candidate in Computer Science at Washington State University. He received his Bachelor of Technology degree in Computer Science and Engineering from West Bengal University of Technology, Kolkata, India in 2009. Mr. Das's research interest is in addressing machine learning and data mining challenges in pervasive and mobile healthcare domains. Including IEEE, Mr. Das is also Student Member with ACM SIGKDD and SIAM.

Narayanan C. Krishnan is a visiting faculty at Indian Institute of Technology Ropar. He received his B.Sc and M.Sc degrees in Mathematics and M. Tech degree in Computer Science from Sri Sathya Sai Institute of Higher Learning in 2000, 2002 and 2004 respectively, a Ph.D. degree in Computer Science from Arizona State University in 2010. Prior to his moving to IIT Ropar, Narayanan was associated with the Center for Advanced Studies in Adaptive Systems at Washington State University, where he designed and developed algorithms for on-line activity recognition. His primary research interests include human behavior modeling from pervasive sensors and other data mining problems in pervasive computing applications.

Diane J. Cook is a Huie-Rogers Chair Professor in the School of Electrical Engineering and Computer Science at Washington State Univeristy. Dr. Cook received a B.S. degree in Math/Computer Science from Wheaton College in 1985, a M.S. degree in Computer Science from the University of Illinois in 1987, and a Ph.D. degree in Computer Science from the University of Illinois in 1990. Her research interests include artificial intelligence, machine learning, graph-based relational data mining, smart environments, and robotics. Dr. Cook is an IEEE Fellow.