International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 23, No. 4 (2009) 687–719

World Scientific
www.worldscientific.com

CLASSIFICATION OF IMBALANCED DATA: A REVIEW

YANMIN SUN
Pattern Discovery Technologies Inc., 554 Parkside Drive
Waterloo, Canada N2L 5Z4

ANDREW K. C. WONG
Systems Design Department, University of Waterloo
200 University Avenue West, Waterloo, Canada N2L 3G1

MOHAMED S. KAMEL
Electrical and Computer Engineering Department
University of Waterloo, 200 University Avenue West
Waterloo, Canada N2L 3G1

Classification of data with imbalanced class distribution has encountered a significant drawback of the performance attainable by most standard classifier learning algorithms which assume a relatively balanced class distribution and equal misclassification costs. This paper provides a review of the classification of imbalanced data regarding: the application domains; the nature of the problem; the learning difficulties with standard classifier learning algorithms; the learning objectives and evaluation measures; the reported research solutions; and the class imbalance problem in the presence of multiple classes.

Keywords: Classification; class imbalance problem.

1. Introduction

Classification is an important task of pattern recognition. A range of classification learning algorithms, such as decision tree, backpropagation neural network, Bayesian network, nearest neighbor, support vector machines, and the newly reported associative classification, have been well developed and successfully applied to many application domains. However, imbalanced class distribution of a data set has encountered a serious difficulty to most classifier learning algorithms which assume a relatively balanced distribution.[15,28,46,66] The imbalanced data is characterized as having many more instances of certain classes than others. As rare instances occur infrequently, classification rules that predict the small classes tend to be rare, undiscovered or ignored; consequently, test samples belonging to the small classes are misclassified more often than those belonging to the prevalent classes. In certain applications, the correct classification of samples in the small classes often has a greater value than the contrary case. For example, in a disease

diagnostic problem where the disease cases are usually quite rare as compared with normal populations, the recognition goal is to detect people with diseases. Hence, a favorable classification model is one that provides a higher identification rate on the disease category. Imbalanced or skewed class distribution problem is therefore also referred to as small or rare class learning problem.

Research on the class imbalance problem is critical in data mining and machine learning. Two observations account for this point: (1) the class imbalance problem is pervasive in a large number of domains of great importance in data mining community. Reported applications include medical diagnosis,[58] detection of oil spills in satellite radar images,[46] the detection of fraudulent calls,[28] risk management,[26] modern manufacturing plants,[66] text classification,[10] etc.; and (2) most popular classification learning systems are reported to be inadequate when encountering the class imbalance problem. These classification systems involve decision trees,[4,15,40,84] support vector machines,[40,65,87] backpropagation neural networks,[40] Bayesian network,[26] nearest neighbor[4,91] and the newly-reported associative classification approaches.[54,86] The significant difficulty of the class imbalance problem and its frequent occurrence in practical applications of machine learning and data mining have attracted a lot of research interest. Two workshops on this problem were held in 2000[37] and 2003[16] at the AAAI and ICML conferences, respectively. A number of research papers dedicated to this problem can be found in Ref. 17 and other publications. Research efforts are addressed on three aspects of the class imbalance problem: (1) the nature of the class imbalance problem (i.e. "In what domains do class imbalances most hinder the performance of a standard classifier?"[40]); (2) the possible solutions in tackling the class imbalance problem; and (3) the proper measures for evaluating classification performance in the presence of the class imbalance problem.

This paper provides an overview on the classification of imbalanced data. Each of the following sections studies one aspect of the class imbalance problem — encompassing the nature of the problem, the learning difficulties with standard classifier learning algorithms, the application domains, the learning objectives and evaluation measures, the reported research solutions, the class imbalance problem in the presence of multiple classes, and, finally, the conclusions and some recommendations for future research.

2. Examples of Application Domains

The class imbalance problem is pervasive in a large number of domains of great importance to the data mining community. This problem is intrinsic to some application domains. In some cases, it happens when the data collection process is limited due to certain reasons.[15] The following examples illustrate such cases.

- **Fraud Detection.** Fraud, such as credit card fraud and cellular fraud, is a costly problem for many business organizations. In the United States, cellular fraud costs the telecommunications industry hundreds of millions of dollars per

year.[70,80] Companies attempt to detect fraud by analyzing different consuming patterns in their transaction databases. However, in their transaction collections, there are many more legitimate users than fraudulent examples.

- **Medical Diagnosis.** Clinical databases store large amounts of information about patients and their medical conditions. Data mining techniques applied on these databases attempt to discover relationships and patterns among clinical and pathological data to understand the progression and features of certain diseases. The discovered knowledge can be used for early diagnosis. In the clinical databases, disease cases are fairly rare as compared with the normal populations.
- **Network Intrusion Detection.** As network-based computer systems play increasingly vital roles in modern society, attacks on computer systems and computer networks grow more commonplace. Learning prediction rules from network data is an effective anomaly detection approach to automate and simplify the manual development of intrusion signatures. It is found that different types of network attacks are present — some overwhelming, others rare — in the collection of network connection records. For example, the KDD-CUP'99 contest data contains four categories of network attacks: denial-of-service (dos), surveillance (probe), remote-to-local (r2l), and user-to-root (u2r). Among these four types of attacks, the u2r and r2l categories are intrinsically rare.
- **Detection of oil spills from radar images of the ocean surface.[46]** Only about 10% of oil spills originate from natural sources, such as leakage from sea beds. Much more prevalent is pollution caused intentionally by ships that want to dispose of oil residue in their tanks. Radar images from satellites provide an opportunity for monitoring coastal waters. Since oil slicks are less reflective of radar than the average ocean surface, they appear dark in an image. An oil spill detection system based on satellite images could be an effective early warning system, and possibly a deterrent to illegal dumping, and could have significant environmental impact. Although satellites are continually producing images, images containing oil spills are much fewer than those without oil spills.
- **Modern Manufacturing Plants.** In a modern manufacturing plant such as a Boeing assembly line,[66] more and more processes are being handled by automated or semi-automated cells, each with a computer as its controller that renders automatic alarm when flaw patterns are detected. In constructing an alarm system through supervised learning, the number of available defective cases is significantly fewer than that of ordinary procedures.

In addition to these examples, other reported applications involve text classification[10] and direct marketing.[51] Some of these applications, such as fraud detection, intrusion detection, medical diagnosis, etc., are also recognized as *anomaly detection* problems. In anomaly detection, the goal is to find objects that are different from most other objects.[74] Because anomalous and normal objects can be viewed as defining two distinct classes, a considerable subset of anomaly detection systems perceive anomaly detection as a dichotomous data partitioning

problem in which data samples are categorized as either abnormal or normal. As anomalies are commonly rare as compared with normal observations, the class imbalance problem is thus intrinsic to the anomaly detection applications.

3. Nature of the Problem

In a data set with the class imbalance problem, the most obvious characteristic is the skewed data distribution between classes. However, theoretical and experimental studies presented in Refs. 4, 38, 40, 41, 84 and 85 indicate that the skewed data distribution is not the only parameter that influences the modeling of a capable classifier in identifying rare events. Other influential facts include small sample size, separability and the existence of within-class subconcepts.

3.1. *Imbalanced class distribution*

For a biclass problem, the imbalance degree of a class distribution can be denoted by the ratio of the sample size of the small class to that of the prevalent class. In practical applications, the ratio can be as drastic as 1:100, 1:1000, or even larger.[15] In Ref. 85, research was conducted to explore the relationship between the class distribution of a training data set and the classification performances of decision trees. Their study indicates that a relatively balanced distribution usually attains a better result. However, at what imbalance degree the class distribution would deteriorate the classification performance cannot be stated explicitly, since other factors such as sample size and separability also affect the performance. In some applications, a ratio as low as $1 : 35$ can make some methods inadequate for building a good model, while in some other cases, $1 : 10$ is tough to deal with.[41]

3.2. *Small sample size*

In the class imbalance problem, when an imbalance degree is fixed, the sample size plays a crucial role in determining the "goodness" of a classification model. In the case that the sample size is limited, uncovering regularities inherent in small class is unreliable. Experimental observations reported in Ref. 40 indicate that as the size of the training set increases, the large error rate caused by the imbalanced class distribution decreases. This observation is quite understandable. When more data can be used, relatively more information about the small class benefits the classification modeling, which is then able to distinguish rare samples from the majority. Hence, the authors of Ref. 40 suggest that the imbalanced class distribution may not be a hindrance to classification if a large enough data set is provided, assuming that the data set is available and the learning time required for a sizeable data set is acceptable.

3.3. *Class separability*

The difficulty in separating the small class from the prevalent class is the key issue of the small class problem. If there exist highly discriminative patterns among each class, then not very sophisticated rules are required to distinguish class objects.

However, if patterns among each class are overlapping at different levels in some feature space, discriminative rules are hard to induce. Experiments conducted in Ref. 60 varying the degree of overlap between classes concluded that "The class imbalance distribution, by itself, does not seem to be a problem, but when allied to highly overlapped classes, it can significantly decrease the number of minority (small) class examples correctly classified." A similar claim based on experiments reported in Ref. 40 is that "Linearly separable domains are not sensitive to any amount of imbalance. As a matter of fact, as the degree of concept complexity increases, so does the system's sensitivity to imbalance."

3.4. *Within-class concepts*

In many classification problems, a single class is composed of various subclasses, or subconcepts. Usually, samples of a class are collected from different subconcepts. These subclasses or subconcepts do not always contain the same number of examples. This phenomena is referred to as *within-class imbalance*, corresponding to the imbalanced class distribution among subclasses.[38] The presence of within-class subconcepts worsens the imbalance distribution problem (no matter between or within class) in two aspects: (1) the presence of within-class subconcepts increases the learning concept complexity on the data set; and (2) the presence of within-class subconcepts is implicit in most cases.

4. Learning Difficulties with Standard Classifier Modeling Algorithms

In this section, a subset of well-developed classifier learning algorithms is reviewed and discussed. To be appreciated by less knowledgable readers to these learning algorithms, we have included a brief introduction on each classifier's learning methods and yielded an insight into the deficiency of each learning method when encountering the small classes. Nevertheless, due to space limitations, we have not been able to introduce all the concepts completely. At these places, we have provided related references for interested readers.

4.1. *Decision trees*

Decision trees use simple knowledge representation to classify examples into a finite number of classes. In a typical setting, the tree nodes represent the attributes, the edges represent the possible values for a particular attribute, and the leaves are assigned with class labels. Classifying a test sample is straightforward once a decision tree has been constructed. An object is classified by following paths from the root node through the tree to a leaf, taking the edges corresponding to the values of attributes. Some popular tree algorithms include ID3,[64] C4.5[63] and CART.[7,9]

A decision tree classifier is modeled in two phases: *tree building* and *tree pruning*. In tree building, the decision tree model is built by recursively splitting the training data set based on a locally optimal criterion until all or most of the records belonging

to each of the partitions bear the same class label. After building the decision tree, a tree pruning step is performed to overcome the overfitting of the training samples. Pruning attempts to improve the generalization capability of a decision tree by trimming the branches of the initial tree. Various pruning techniques are reported.[63] The general criterion for pruning a decision tree is the classification error: if replacement of this subtree with a leaf, or with its most frequently used branch, would lead to a lower predicted error rate, then prune the tree accordingly.

When building decision trees, the class label associated with a leaf is found by examining the training cases covered by the leaf and choosing the most frequent class. In the presence of the class imbalance problem, decision trees may need to create many tests to distinguish the small classes from the large classes. In some learning processes, the split action may be terminated before the branches for predicting small classes are detected. In other learning processes, the branches for predicting the small classes may be pruned as being susceptible to overfitting. The cause behind is that correctly predicting a small number of samples from the small classes contributes too little success to deduce the error rate significantly, as compared with the error rate increased by overfitting. Since the pruning is mainly based on the predicting error, there is a high probability that some branches that predict the small classes are removed and the new leaf node is labeled with a dominant class.

4.2. *Backpropagation neural networks*

The multi-layer perceptron (MLP), trained by the backpropagation (BP) algorithm,[35] is one of the most widely used neural models for classification problems. A BP neural network is composed of one input layer, one output layer, and one or more hidden layers. Each layer is composed of one ore more "neurons". Each neuron in a layer is connected to each neuron of the prior and the next layers. There are no activation functions in the neurons in input layer, but every neuron in hidden layer or output layer has its activation function. The activation function makes the output of the neuron nonlinear, scaled and differentiable (continuous). The connection between two neurons is parameterized by a weight. At the beginning, the weights are initialized to random numbers ranging between -1 and 1. Then the BP network is trained by iteratively adjusting the weights such that the difference between the computed output of the network and the targeted output is minimized. Standard backpropagation is a gradient descent algorithm, in which the network weights are moved along the negative of the gradient of the performance function. It requires many iterations over the training process to reach to the convergence of net error. For classification problems, a BP network approximates the posterior class probabilities given the feature vectors of the samples to be classified.

Empirical studies on training imbalanced biclass data sets in Ref. 3 observed that the net error for samples in the prevalent class was reduced rapidly in the first few iterations but the net error for small class increased considerably instead. Theoretical analysis indicated that this phenomenon occurred because the gradient

vector computed by the standard backpropogation algorithm was in a downhill direction for the prevalent class but in a uphill direction for the small class. It has been proved that the expected lengths of the gradient vectors with respect to different classes are proportional to the class sizes. This implies the length of the gradient vector of the prevalent class will be much larger than that of the small class when the class distribution is very askew. Therefore, the gradient vector is dominated by that of the dominant class. Consequently, if weights are recalculated in the direction of the gradient vector, the net error of the prevalent class will decrease significantly and that of the small class will increase significantly.[3] The empirical studies also reported that the subsequent rate of decrease of net error for the small class was very low. It needed thousands of iterations to reach an acceptable solution. Usually, the training process was terminated before the net error for the small class could be decreased. The deficient performance of BP neural networks with imbalance data sets are also reported in Refs. 11 and 40.

4.3. *Bayesian classification*

Bayesian classification is based on the inferences of probabilistic graphic models which specify the probabilistic dependencies underlying a particular model using a graph structure.[59] In its simplest form, a probabilistic graphical model is a graph in which nodes represent random variables, and the arcs represent conditional dependence assumptions. Hence it provides a compact representation of joint probability distributions. Under such formulation, an undirected graphical model is called as a Markov network, while a directed graphical model is called as a Bayesian network or a Belief Network.[34] To explore the probabilistic dependencies which underlie a particular model, learning a Bayesian network from data can be subdivided into parameter learning and structural learning, the latter being the more difficult concept. Recently, there has been significant work on methods whereby both the structures and the parameters of the graphic models can be learned directly from databases.[34] Once a probabilistic network is built, one can derive the probability of an event, conditioned by a set of observations for classification.

The problem of learning a probabilistic model is to find a network that best matches the given training data set. To exhaustively explore the dependencies among attributes, a complete graph, where every attribute is connected to every other attribute, is favorable. However, such networks do not provide any useful representation of the independence assertions in the learned distributions and overfit the training data.[31] Hence, the networks are learned according to certain scoring functions to approximate those dependency patterns which dominate the data. Obviously, for a given imbalanced data set, dependency patterns inherent in the small classes are usually not significant and hard to be adequately encoded in the networks. When the learned networks are inferred for classification, the samples of the small classes are most likely misclassified. Experimental results in Ref. 44 reported this observation.

4.4. *Support vector machines*

Support Vector Machines(SVMs) are one of the binary classifiers based on maximum margin strategy introduced by Vapnik.[79] Originally, SVMs were for linear two-class classification with margin, where margin means the minimal distance from the separating hyperplane to the closest data points. SVMs seek an optimal separating hyperplane, where the margin is maximal. The solution is based only on those data points at the margin. These points are called as support vectors. The linear SVMs have been extended to nonlinear examples when the nonlinear separated problem is transformed into a high dimensional feature space using a set of nonlinear basis functions. However, the SVMs are not necessary to implement this transformation to determine the separating hyperplane in the possibly high dimensional feature space. Instead, a kernel representation can be used, where the solution is written as a weighted sum of the values of a certain kernel function evaluated at the support vectors. When perfect separation is not possible, *slack* variables are introduced for sample vectors to balance the tradeoff between maximizing the width of the margin and minimizing the associated error.

SVMs are believed to be less prone to the class imbalance problem than other classification learning algorithms,[40] since boundaries between classes are calculated with respect to only a few support vectors and the class sizes may not affect the class boundary too much. Nevertheless, research works in Refs. 2 and 87 still indicate that SVMs can be ineffective in determining the class boundary when the class distribution is too askew. Experiments were conducted on SVMs in Ref. 87 to draw boundaries for two data sets: the first data set with the ratio of the number of the large class instances to the number of the small class instances of 10:1, and the second data set with the ratio of 10000:1. It turned out that the boundary of the second data set was much more skewed towards the small class than the boundary for the first data set, and thus caused a higher incidence of classifying test instances to the prevalent class. The underlining reason for this phenomenon is that as the training data gets more imbalanced, the support vector ratio between the prevalent class and the small class also becomes more imbalanced. The small amount of cumulative error on the small class instances count for very little in the tradeoff between maximizing the width of the margin and minimizing the training error. SVMs simply learn to classify everything as the prevalent class in order to make the margin the largest and the error the minimum.

4.5. *Associative classifiers*

Associative classification is a new classification approach integrating association mining and classification into a single system.[21,48,49,53,81,83,88] Association mining, or pattern discovery, aims to discover descriptive knowledge from a database, while classification focuses on building a classification model for categorizing new data. By and large, both association pattern discovery and classification rule mining are essential to practical data mining applications. Considerable efforts have been made

to integrate these two techniques into one system. A typical associative classification system is constructed in two stages: (1) discovering all the event associations (in which the frequency of occurrences is significant according to some tests); and (2) generating classification rules from the association patterns to build a classifier. In the first stage, the learning target is to discover the association patterns inherent in a database (also referred to as knowledge discovery). In the second stage, the task is to select a small set of relevant association patterns to construct a classifier given the predicting attribute. Associative rules for classification are derived from the discovered association patterns, which are selected based upon certain statistical tests.

For imbalanced data, association patterns describing the small classes are unlikely to be found since the combination of items characterizing the small classes occur too seldom to pass certain significancance tests for detecting association patterns. Consequently, classification rules obtained from the discovered association patterns for predicting the small classes are therefore rare and weak. This observation is also discussed in Refs. 55, 82 and 84.

4.6. *K-nearest neighbor*

K-Nearest Neighbor (KNN) is an instance-based classifier learning algorithm, which uses specific training instances to make predictions without having to maintain an abstraction (or model) derived from data. Initial theoretical results can be found in Ref. 19 and an extensive overview can be found in Ref. 22 The conceptual idea of the K-Nearest Neighbor algorithm is simple and intuitive. Given a test sample, the algorithm computes the distance (or similarity) between the test sample and all of the training samples to determine its k-nearest neighbors. The class of the test sample is decided by the most abundant class within the k-nearest neighbor samples.

In the presence of the imbalanced training data, samples of the small classes occur sparsely in the data space. Given a test sample, the calculated k-nearest neighbors bear higher probabilities of samples from the prevalent classes. Hence, test cases from the small classes are prone to being incorrectly classified. Research works in Refs. 4 and 91 reported this observation.

4.7. *Summary*

Generally, in constructing a classification model, a learning algorithm reveals the underlying relationship between the attribute set and class label, and identifies a model that best fits the training data. The task of the constructed classifier is to predict the class labels for any unseen input objects. Therefore, the learning objective is a classification model with good generalization capability (i.e. a model that accurately predicts the class labels of previously unknown records). For the best generalization, the model should fit the training data accordingly. If the model fits the training data poorly (i.e. the model underfits the data), both the training error and generalization error are high. In such a case, by learning the training set

better, both the training error and generalization error decrease. If the model fits the training data too well, the performance on the training examples still increases while the generalization error becomes worse. This phenomenon is known as model overfitting.

In order to avoid overfitting, additional techniques are introduced to limit the exhaustive learning on the training set. Such techniques search for the most common regularities in the data for improved generalization performance. For example, the Minimum Description Length (MDL) principle[67] is stated as: given a limited set of observed data, the best explanation (i.e. model with good generalization performance) is one that permits the greatest compression of the data. That is, the more we are able to compress the data, the better we learn about the underlying regularities that generated the data. Such a process generates an unavoidable *maximum-generality* bias favoring the discovery of more general rules (i.e. the larger disjuncts).[36,76] The maximum-generality learning for a decision tree is to prune the tree size considerably; for a BP neural network is to limit the networks complexity properly; for a Bayesian graphical model is to explore those dominant dependency patterns according to certain scoring functions; and for an associative classifier is to mine those significant association patterns by setting up a strict test threshold.

However, such an inductive bias of maximum-generality encountered a serious challenge with the classification of imbalanced data. Since interested instances occur infrequently, models that describe the rare class have to be highly specialized and cannot be easily simplified into more general rules with broader data coverage. Classification rules that predict the small class tend to be rare, undiscovered or ignored. Hence, the maximum-generality bias works well for the large class but not for the small class. Table 1 summarizes the learning strategies and learning difficulties with the small class of each classification learning algorithm.

Noisy data may also make it difficult to learn the rare cases. In the real world, data contains various types of errors, either random or systematic. Random errors are often referred to as noise. Given a sufficiently high level of background noise, a learner may not be able to distinguish between rare cases and noise-induced cases.[84] Most of the so-called noise-tolerant techniques that try to minimize the overall impact of noise usually perform at the expense of the rare cases, as they tend to remove noisy data and the rare cases as well.

5. Learning Objectives and Evaluation Measures

Evaluation measures play a crucial role in both assessing the classification performance and guiding the classifier modeling. Traditionally, accuracy is the most commonly used measure for these purposes. However, for classification with the class imbalance problem, accuracy is no longer a proper measure since the rare class has very little impact on the accuracy as compared to that of the prevalent class.[42,84] For example, in a problem where a rare class is represented by only 1% of the training data, a simple strategy can be one that predicts the prevalent class label for

Table 1. Classification methods and learning difficulties with the imbalanced data.

Method	Learning Strategies	Learning Difficulties with the Small Class	Comments
Decision Trees	• Recursively splitting the training data and assigning a class label to leaf by the most frequent class • Pruning a subtree with a leaf or a branch if lower training error obtained	• Needs many splits to distinguish the small class • Branches or leaves predicting the small class are prone to be pruned	• *Most research efforts on the class imbalance problem focus on C4.5*
BP Neural Networks (BP NNs)	• Iteratively adjusting the weights connecting the neurons to minimize training error by the gradient descent algorithm	• The gradient descent direction is dominated by the prevalent class • Training error is minimized only for the prevalent class	
Bayesian Classification	• Exploring the dominant dependency patterns among attributes	• Dependency patterns inherent in the small class are hard to be encoded	
Support Vector Machines (SVMs)	• Seeking an optimal separating hyperplane to maximize the margin and minimize the training error only based on data points at the margin	• Data points of the small class are rare at the margin • Decision boundary are skewed toward the small class	• *SVMs are reported to be less affected by the class imbalance problem*
Association Classification	• Deriving classification rules from association patterns	• Patterns of the small class are unlikely found as they occur seldomly	
K-Nearest Neighbor (K-NN)	• Deciding the class label of a test sample by the most abundant class within the k nearest neighbors	• The k nearest neighbors bear higher probabilites of samples from the prevalent class as samples of the small class occur sparsely	

Table 2. Confusion matrix.

	Predicted as Positive	Predicted as Negative
Actually Positive	True Positives (TP)	False Negatives (FN)
Actually Negative	False Positive (FP)	True Negatives (TN)

every example. It can achieve a high accuracy of 99%. However, this measurement is meaningless to some applications where the learning concern is the identification of the rare cases.

In the bi-class scenario, one class with very few training samples but high identification importance is referred to as the positive class; the other as the negative class. Samples can be categorized into four groups after a classification process as denoted in the confusion matrix given in Table 2.

Several measures can be derived using the confusion matrix:

- True Positive Rate: $TP_{\text{rate}} = \frac{TP}{TP+FN}$
- True Negative Rate: $TN_{\text{rate}} = \frac{TN}{TN+FP}$
- False Positive Rate: $FP_{\text{rate}} = \frac{FP}{TN+FP}$
- False Negative Rate: $FN_{\text{rate}} = \frac{FN}{TP+FN}$
- Positive Predictive Value: $PP_{\text{value}} = \frac{TP}{TP+FP}$
- Negative Predictive Value: $NP_{\text{value}} = \frac{TN}{TN+FN}$

Given a data set with imbalanced class distribution, the identification performance on the small class is usually unsatisfactory. To remedy this, the learning objective can be: (1) to balance the identification abilities between the two classes; and/or (2) to improve the recognition success on the small class. With respect to the different learning objectives, the classification performance should be evaluated by different measures. Clearly neither of these measures are adequate by themselves. For different evaluation criteria, several measures are devised.

5.1. *F-measure*

If only the performance of the positive class is considered, two measures are important: True Positive Rate (TP_{rate}) and Positive Predictive Value (PP_{value}). In information retrieval, True Positive Rate is defined as *recall* (R) denoting the percentage of retrieved objects that are relevant:

$$R = TP_{\text{rate}} = \frac{TP}{TP + FN} \tag{1}$$

Positive Predictive Value is defined as *precision* (P) denoting the percentage of relevant objects that are identified for retrieval:

$$P = PP_{\text{value}} = \frac{TP}{TP + FP} \tag{2}$$

F-measure (F) is suggested in Ref. 47 to integrate these two measures as an average

$$F\text{-measure} = \frac{2RP}{R+P}. \tag{3}$$

In principle, F-measure represents a harmonic mean between recall and precision[74]:

$$F\text{-measure} = \frac{2}{\frac{1}{R}+\frac{1}{P}}. \tag{4}$$

The harmonic mean of two numbers tends to be closer to the smaller of the two. Hence, a high F-measure value ensures that both recall and precision are reasonably high.

5.2. *G-mean*

When the performance of both classes is concerned, both True Positive Rate (TP_{rate}) and True Negative Rate (TN_{rate}) are expected to be high simultaneously. Kubat *et al.*,[46] suggested the G-mean defined as

$$G\text{-mean} = \sqrt{TP_{\text{rate}} \cdot TN_{\text{rate}}} \tag{5}$$

G-mean measures the balanced performance of a learning algorithm between these two classes. The comparison among harmonic, geometric, and arithmetic means are illustrated in Ref. 74 by way of an example. Suppose that there are two positive numbers 1 and 5. Their arithmetic mean is 3, their geometric mean is 2.236, and their harmonic mean is 1.667. The harmonic mean is the closest to the smaller value and the geometric mean is closer than the arithmetic mean to the smaller number.

5.3. *ROC analysis*

Some classifiers, such as Bayesian Network inference or some Neural Networks, assign a probabilistic score to its prediction. Class prediction can be changed by varying the score threshold. Each threshold value generates a pair of measurements of (FP_{rate}, TP_{rate}). By linking these measurements with the False Positive Rate (FP_{rate}) on the X axis and the True Positive Rate (TP_{rate}) on the Y axis, a Receiver Operating Characteristics (ROC) graph is plotted in Fig. 1.

The ideal model is one that obtains 1 True Positive Rate and 0 False Positive Rate (i.e. $TP_{\text{rate}} = 1$ and $FP_{\text{rate}} = 0$). Therefore, a good classification model should be located as close as possible to the upper left corner of the diagram, while a model that makes a random guess should reside along the main diagonal, connecting the points ($TP_{\text{rate}} = 0,\ FP_{\text{rate}} = 0$), where every instance is predicted as a negative class, and ($TP_{\text{rate}} = 1,\ FP_{\text{rate}} = 1$), where every instance is predicted as a positive class. A ROC graph depicts relative trade-offs between benefits (true positives) and costs (false positives) across a range of thresholds of a classification model. A ROC curve gives a good summary of the performance of a classification model. To compare several classification models by comparing ROC curves, it is hard to claim

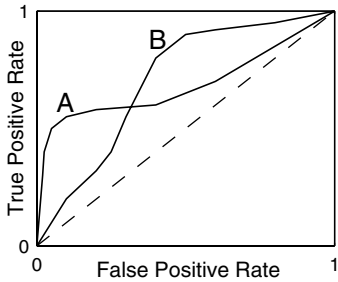

Fig. 1. ROC curves for two different classifiers.

a winner unless one curve clearly dominates the others over the entire space.[61] The area under a ROC curve (AUC) provides a single measure of a classifier's performance for evaluating which model is better on average. It has been shown that there is a clear similarity between AUC and well-known Wilcoxon statistics.[33]

6. Research Solutions

Reported solutions are developed at both data and algorithm levels. At the data level, the objective is to rebalance the class distribution by resampling the data space.[12,15,25] At the algorithm level, solutions try to adapt existing classifier learning algorithms to strengthen learning with regards to the small class.[62,89] Cost-sensitive learning solutions incorporating both the data and algorithm level approaches assume higher misclassification costs with samples in the rare class and seek to minimize the high cost errors.[52,66,90] Several boosting algorithms are also reported as meta-techniques which are applicable to most classifier learning algorithms.[27,73,77] The general idea of a boosting approach is to introduce cost items into the learning framework of AdaBoost[29,69] in an effort to boost more weights on samples associated with higher identification importance, and eventually bias the learning towards them. These boosting algorithms are referred to as cost-sensitive boosting algorithms as they employ cost items. They are regarded as solutions at the data level since they update the data space by weighting samples.

6.1. *Data-level approaches*

Solutions at the data-level include many different forms of resampling, such as randomly oversampling the small class, randomly undersampling the prevalent class, informatively oversampling the small class (in which no new samples are created, but the choice of samples to resample is targeted rather than random), informatively undersampling the prevalent class (the choice of samples to eliminate is targeted), oversampling the small class by generating new synthetic data, and combinations of the above techniques.[12,15,25,92]

Even though resampling is an often-used method in dealing with the class imbalance problem, the matter at issue is what is or how to decide the optimal class

distribution given a data set. A thorough experimental study on the effect of a training set's class distribution on a classifier's performance is conducted in Ref. 85. Their experimental results show that, with respect to the classification performance evaluated by AUC, a balanced class distribution (class size ratio is 1:1) performs relatively well but is not necessarily optimal. Optimal class distributions differ from data to data. A wrapper framework is proposed in Refs. 14 and 13 for empirically discovering the sampling parameters to optimize certain evaluation functions.

In addition to the class distribution issue, how to effectively resample the training data is another issue. Random sampling is simple but not sufficient in many cases. In a biclass application, the small class and prevalent class are also referred to as minority class and majority class respectively in literature. If the class imbalance problem of a data set is represented by within-class concepts, random oversampling may overduplicate samples on some parts and less so on others. A more favorable resampling process should, first, detect the subconcepts constituting the class; then, oversample each concept respectively to balance the overall distribution. However, such an informative resampling process increases the cost for data analysis. Informatively undersampling the prevalent class attempting to make the selective samples more representative poses another problem: what is the criterion in selecting samples? For example, if samples are measured by some distance measurements, those majority class samples which are relatively far away from the minority class samples may represent more the majority class features, while those which are relatively close to the minority class samples may be crucial in deciding the class boundary by some classifier learning algorithms. Which part should be more focused on when selecting quality samples? These issues cannot be settled systematically. A number of techniques are reported, but each of them may be effective if applied in a certain context.

6.2. *Algorithm-level approaches*

Generally, a common strategy to deal with the class imbalance problem is to choose an appropriate inductive bias. For decision trees, one approach is to adjust the probabilistic estimate at the tree leaf[62,89]; another approach is to develop new pruning techniques.[89] For SVMs, proposals such as using different penalty constants for different classes,[50] or adjusting the class boundary based on kernel-alignment ideal,[87] are reported. For association rule mining, multiple minimum supports for different classes are specified to reflect their varied frequencies in the database.[55] To develop an algorithmic solution, one needs the knowledge of both the corresponding classifier learning algorithm and the application domain, especially a thorough comprehension on why the learning algorithm fails when the class distribution of available data is uneven.

In recognition-based one-class learning, a system is modeled with only examples of the target class in the absence of the counter examples. This approach does not try to partition the hypothesis space with boundaries that separate positive and

negative examples, but it attempts to make boundaries which surround the target concept. For classification purposes, it measures the amount of similarity between a query object and the target class, where a threshold on the similarity value is introduced. Two classifier learning algorithms are studied in the context of the one-class learning approach: neural network training[39] and SVMs.[56] Under certain conditions such as multimodal domains, the one-class approach is reported to be superior to discriminative (two-class learning) approaches.[39] The threshold in this approach represents the boundary between the two classes. A too strict threshold means that positive data will be sifted, while a too loose threshold will include considerable negative samples. Hence, to set up an effective threshold is crucial with this approach. Moreover, many machine learning algorithms, such as decision trees, do not function unless the training data includes examples from different classes.

6.3. *Cost-sensitive learning*

Cost-sensitive classification considers the varying costs of different misclassification types.[57] A cost matrix encodes the penalty of classifying samples from one class as another. Let $C(i, j)$ denote the cost of predicting an instance from class i as class j. With this notation, $C(+, -)$ is the cost of misclassifying a positive (rare class) instance as the negative (prevalent class) instance and $C(-, +)$ is the cost of the contrary case. In dealing with the class imbalance problem, the recognition importance of positive instances is higher than that of negative instances. Hence, the cost of misclassifying a positive instance outweighs the cost of misclassifying a negative one (i.e. $C(+, -) > C(-, +)$); making a correct classification usually presents 0 penalty (i.e. $C(+, +) = C(-, -) = 0$). The cost-sensitive learning process then seeks to minimize the number of high cost errors and the total misclassification cost.

A cost-sensitive classification technique takes the cost matrix into consideration during model building and generates a model that has the lowest cost. Reported works in cost-sensitive learning fall into three main categories:

- **Weighting the data space.** The distribution of the training set is modified with regards to misclassification costs, such that the modified distribution is biased towards the costly classes. This approach can be explained by the *Translation Theorem* derived in Ref. 90. Against the normal space without considering the cost item, let us call a data space with domain $X \times Y \times C$ as the *cost-space*, where X is the input space, Y is the output space and C is the cost associated with mislabeling that example. If we have examples drawn from a distribution D in the cost-space, then we can have another distribution $\hat{D}$ in the normal space that

$$\hat{D}(X, Y) \equiv \frac{C}{E_{X,Y,C\sim D}[C]} D(X, Y, C). \tag{6}$$

  Here $E_{X,Y,C\sim D}[C]$ is the expectation of cost values. According to the translation theorem, those optimal error rate classifiers for $\hat{D}$ will be optimal cost minimizers

for D. Hence, when we update sample weights integrating the cost items, choosing a hypothesis to minimize the rate of errors under $\hat{D}$ is equivalent to choosing the hypothesis to minimize the expected cost under D.

- **Making a specific classifier learning algorithm cost-sensitive.** For example, in the context of decision tree induction, the tree-building strategies are adapted to minimize the misclassification costs. The cost information is used to: (1) choose the best attribute to split the data[52,66]; and (2) determine whether a subtree should be pruned.[5]
- **Using Bayes risk theory to assign each sample to its lowest risk class.** For example, a typical decision tree for a binary classification problem assigns the class label of a leaf node depending on the majority class of the training samples that reach the node. A cost-sensitive algorithm assigns the class label to the node that minimizes the classification cost.[20,89]

Converting sample-dependent costs into sample weights, methods in the first group, is also known as *cost-sensitive learning by example weighting*.[1] The weighted training samples are then applied to standard learning algorithms. This approach is at the data-level without changing the underlying learning algorithms. Methods in the second and third groups, adapting the existing learning algorithms, are at the algorithm-level. Cost-sensitive learning assumes that a cost-matrix is known for different types of errors or samples. Given a data set, however, the cost matrix is often unavailable.

6.4. *Boosting approaches*

6.4.1. *Ensemble learning and AdaBoost*

The basic idea of classifier ensemble learning is to construct multiple classifiers from the original data and then aggregate their predictions when classifying unknown samples. The main motivation for combining classifiers in redundant ensembles is to improve their generalization ability: each component classifier is known to make errors with the assumption that it has been trained on a limited set of data; the patterns that are misclassified by the different classifiers, however, are not necessarily the same.[45] The effect of combining redundant ensembles is also studied in terms of the statistical concepts of bias and variance. Given a classifier, bias-variance decomposition distinguishes among the *bias error*, the *variance error* and the *intrinsic error*. The bias can be characterized as a measure of its ability to generalize correctly to a test set, while the variance can be similarly characterized as a measure of the extent to which the classifier's prediction is sensitive to the data on which it was trained. The variance is then associated with overfitting: if a method overfits the data, the predictions for a single instance will vary between samples.[43] The improvement in performance arising from ensemble combinations is usually the result of a reduction in variance. This occurs because the usual effect of ensemble averaging is to reduce the variance of a set of classifiers. Bagging,[6]

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$
Initialize $D^1(i) = 1/m$.
For $t = 1, \ldots, T$:

(1) Train base learner $h_t \to Y$ using distribution D^t
(2) Choose weight updating parameter α_t:

$$\alpha_t = \frac{1}{2} \log \left(\frac{\sum_{i, y_i = h_t(x_i)} D^t(i)}{\sum_{i, y_i \neq h_t(x_i)} D^t(i)} \right) \tag{7}$$

(3) Update and normalize sample weights:

$$D^{t+1}(i) = \frac{D^t(i) \exp(-\alpha_t h_t(x_i) y_i)}{Z_t} \tag{8}$$

where, Z_t is a normalization factor.

Output the final classifier:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right) \tag{9}$$

Fig. 2. AdaBoost algorithm.

Random forest[8] and AdaBoost[30,68] are ensemble learning methods reported to be successful in variance reduction.

AdaBoost is also observed to be capable of bias reduction. As *stumps* (single-split trees with only two terminal nodes typically have low variance but high bias) are used as the base learner, bagging performs very poorly and AdaBoost improves the base classification significantly.[30,68] With an imbalanced data set, standard learning methods perform poorly on the rare class due to the introduced bias error. The ability of AdaBoost in reducing learning bias is crucial in dealing with the class imbalance problem. The Pseudocode for AdaBoost is given in Fig. 2.

The AdaBoost algorithm weighs each sample reflecting its importance and places the most weights on those samples which are most often misclassified by the preceding classifiers. The sample weighting strategy of AdaBoost is equivalent to resampling the data space combining both up-sampling and down-sampling. It hence belongs to data-level solutions, which are applicable to most classification systems without changing their learning methods. These positive features make the boosting approach an attractive technique in tackling the class imbalance problem. As AdaBoost is accuracy-oriented, its weighting strategy biases towards the prevalent class which contributes more to the overall classification accuracy. The learning issue becomes how to adapt the AdaBoost algorithm to incline its boosting strategy towards the interested class.

6.4.2. *Cost-sensitive boosting*

Cost-sensitive boosting algorithms are developed keeping the general learning framework of AdaBoost, but introducing the cost items into the weight update formula of AdaBoost (Eq. (8)) to adapt its weighting strategy. There are generally three ways to introduce cost items into the weight update formula of AdaBoost: inside the exponent, outside the exponent, and both inside and outside the exponent. Based on these three modifications, three cost-sensitive boosting algorithms, AdaC1, AdaC2, and AdaC3, are implemented in Refs. 72 and 73 with weight update parameter α_t recalculated on each round taking cost items into consideration:

- **AdaC1**

$$D^{t+1}(i) = \frac{D^t(i)\exp(-\alpha_t C_i h_t(x_i) y_i)}{Z_t} \tag{10}$$

α_t is selected as

$$\alpha_t = \frac{1}{2}\log\frac{1 + \sum_{i,y_i=h_t(x_i)} C_i D^t(i) - \sum_{i,y_i\neq h_t(x_i)} C_i D^t(i)}{1 - \sum_{i,y_i=h_t(x_i)} C_i D^t(i) + \sum_{i,y_i\neq h_t(x_i)} C_i D^t(i)} \tag{11}$$

- **AdaC2**

$$D^{t+1}(i) = \frac{C_i D^t(i)\exp(-\alpha_t h_t(x_i) y_i)}{Z_t} \tag{12}$$

α_t is selected as

$$\alpha_t = \frac{1}{2}\log\frac{\sum_{i,y_i=h_t(x_i)} C_i D^t(i)}{\sum_{i,y_i\neq h_t(x_i)} C_i D^t(i)} \tag{13}$$

- **AdaC3**

$$D^{t+1}(i) = \frac{C_i D^t(i)\exp(-\alpha_t C_i h_t(x_i) y_i)}{Z_t} \tag{14}$$

α_t is selected as

$$\alpha_t = \frac{1}{2}\log\frac{\sum_i C_i D^t(i) + \sum_{i,y_i=h_t(x_i)} C_i^2 D^t(i) - \sum_{i,y_i\neq h_t(x_i)} C_i^2 D^t(i)}{\sum_i C_i D^t(i) - \sum_{i,y_i=h_t(x_i)} C_i^2 D^t(i) + \sum_{i,y_i\neq h_t(x_i)} C_i^2 D^t(i)} \tag{15}$$

AdaCost[27] is a variant of AdaC1 by introducing a *cost adjustment function* β instead of cost items.

- **AdaCost**

$$D^{t+1}(i) = D^t(i) \cdot \exp(-\alpha_t y_i h_t(\underline{x}_i)\beta_{\mathrm{sgn}(h_t(\underline{x}_i), y_i)}) \tag{16}$$

α_t is selected as

$$\alpha_t = \frac{1}{2}\log\frac{1 + r_t}{1 - r_t} \tag{17}$$

where

$$r_t = \sum_i D^t(i)\exp(-\alpha_t y_i h_t(\underline{x}_i)\beta_{\mathrm{sgn}(h_t(\underline{x}_i), y_i)}) \tag{18}$$

The requirement for the cost adjust function is: for an instance with a higher cost factor, the function increases its weight "more" if the instance is misclassified, but decreases its weight "less" otherwise.

CSB2 is another cost-sensitive boosting algorithm reported in Ref. 77.

- **CSB2**

$$D^{t+1}(i) = \frac{D^t(i)C_{\mathrm{sgn}(h_t(\underline{x}_i),y_i)}\exp(-\alpha_t y_i h_t(\underline{x}_i))}{Z_t} \tag{19}$$

where $\mathrm{sgn}(h_t(\underline{x}_i), y_i)$ denotes "+" if $h_t(\underline{x}_i)$ equals to y_i ($\underline{x}_i$ is correctly classified), "–" otherwise. The parameters C_+ and C_- are set as $C_+ = 1$ and $C_- = \mathrm{cost}(y_i, h_t(\underline{x}_i)) \geq 1$, where $\mathrm{cost}(i, j)$ is the cost of misclassifying a sample of class i to class j. CSB2 uses the same α_t as computed by AdaBoost.

Referring to the confusion matrix in Table 2, a desirable weight updating rule is to increase the weights of False Negatives more than those of False Positives, but decrease the weights of True Positives more conservatively than those of True Negatives.[27] The resampling effects of these cost-sensitive boosting algorithms are summarized in Table 3. The weighting strategy of AdaBoost distinguishes samples on their classification outputs: correctly classified or misclassified. However, it treats samples of different types (classes) equally: weights of misclassified samples from different classes are increased by an identical ratio; and weights of correctly classified samples from different classes are decreased by another identical ratio. All cost-sensitive boosting algorithms increase the weights of False Negatives more than those of False Positives. On the true predictions (True Positive and True Negative), their weighting strategies are different. AdaC2 and AdaCost are preferable by preserving more weights on the True Positives; AdaC1 and CSB2 are on the opposite; and AdaC3 is uncertain as it is a combinational result of AdaC1 and AdaC2. A thorough study of these cost-sensitive algorithms can be found in Ref. 72. The general conclusion is that AdaC2 is superior to its rivals. The advantages of AdaC2 is illustrated in terms of its sample weighting strategy, its stagewise additive modeling, its sensitively to cost setups, and its generalization performance with respect to the small class.

Table 3. Resampling effects.

	True Predictions	False Predicitions
An ideal weighting strategy	Decreases the weights of True Positives more conservatively than those of True Negatives	Increases the weights of False Negatives more than those of False Positives
AdaBoost	×	×
AdaC1	×	∨
AdaC2	∨	∨
AdaC3	Uncertain	∨
AdaCost	∨	∨
CSB2	×	∨

6.4.3. *Other boosting algorithms*

RareBoost[42] scales False Positive examples in proportion to how well they are distinguished from True Positive examples, and scales False Positive examples in proportion to how well they are distinguished from True Negative examples (refer to Table 2). In their algorithm, the weight update factor α_t^p for positive predictions at the tth iteration is calculated as

$$\alpha_t^p = \frac{1}{2} \ln \frac{TP_t}{FP_t} \tag{20}$$

where TP_t and FP_t denote the weight summation over all True Positive examples and False Positive examples, respectively. The weight update factor α_t^n for negative predictions at the tth iteration is calculated as

$$\alpha_t^n = \frac{1}{2} \ln \frac{TN_t}{FN_t} \tag{21}$$

where TN_t and FN_t denote the weight summation over all True Negative examples and False Negative examples, respectively. Weights are then updated separately using different factors respecting positive predictions and negative predictions.

This weighting strategy decreases the weights of True Positives (TP) and True Negatives (TN), and increases the weights of False Positives (FP) and False Negatives (FN) only if $TP > FP$ and $TN > FN$. The constraint of $TP > FP$ is equivalent to that the precision measure [Eq. 2] of the positive class should be greater than 0.5. In the presence of the class imbalance problem, the small class is always associated with both poor recall and precision values. Hence, such a constraint is a strong condition. Without this condition being satisfied, the algorithm will collapse.

There are some other boosting algorithms, such as SMOTEBoost[18], and DataBoost-IM[32] are developed as a combination of a data synthesis algorithm and the boosting procedure. These algorithms introduce synthetic samples in each round of boosting to balance the training knowledge of different classes. Synthesizing data is application-dependent and hence involves extra learning cost.

6.5. *Summary*

Figure 3 summarizes the reported research solutions to the problem of classification of imbalanced data. These solutions can be categorized as solutions at data level and solutions at algorithm level. Especially, solutions by cost-sensitive learning include methods of both operating data space and adapting learning algorithms.

For cost-sensitive learning, the cost items are used to characterize the identification costs of different samples. The cost value of a sample may depend on the nature of the particular case. For example, in detection of frauds, the cost of missing a particular case of frauds will depend on the amount of money involved in that particular case.[28] Similarly, the cost of a certain kind of mistaken medical diagnosis may be conditional on the particular patient who is misdiagnosed.[78] In these cases,

the costs are "real" values associated with the particular case. These cost values can be used to adapt certain learning processes to minimize cost errors rather than to minimize classification errors. With the class imbalance problem, higher misclassification costs are associated with samples in the small class. The intention to minimize the cost errors leads to increase in the classification performance on the small class. However, in most applications, these cost values are unknown. To bias the learning according to certain parameters, "artificial" cost values are generated or searched to optimize some evaluation functions such as F-measure, G-mean, ROC analysis (Sec. 5). These artificial cost values represent "learning importance" or "weights" of samples, which can be integrated to construct a more satisfactory classification system. For example, when applied to cost-sensitive boosting and/or cost-sensitive weighting approaches, these values are used to update the data space in an informative way which expands the sample sizes of the small classes intentionally in order to bias the learning towards them. Obviously, exploring effective cost setups introduces extra learning cost. Briefly, when real cost values corresponding to a certain application domain are available, the cost information can be used to update either the standard learning algorithms or the learning data space in an effort to minimize cost errors. These solutions can be therefore at both algorithm level and data level. When real cost values are not available, for the purpose of improved performance on the small class, artificial cost values are introduced as a

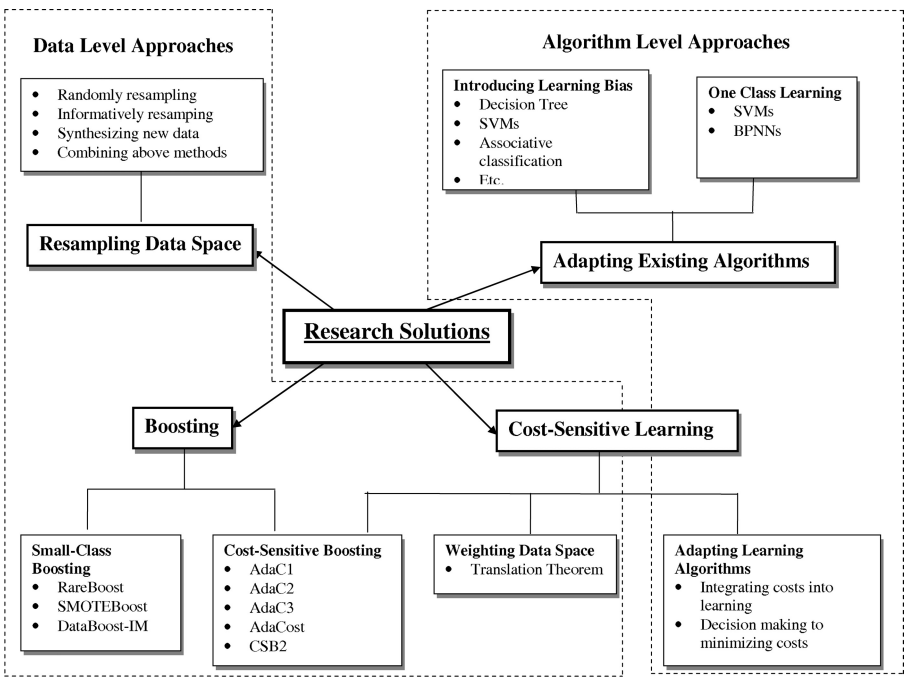

Fig. 3. Research solutions to the problem of classification of imbalanced data.

Table 4. Features of research solutions.

Solutions	Advantages	Constraints/Disadvantages
Data Level Approaches	Applicable to any classification learning systems	
Resampling Data Space	• Straightforward • Often-used	• Unknown optimal class distribution • Uncertain criterion in selecting samples • Risk of information loss when under-sampling the prevalent class • Risk of overfitting when over-sampling the small class
Cost-Sensitive Boosting	• Automatically resampling without implicit factors involved • Combining multiple classifiers decreases the risk of overfitting	• Extra learning cost for exploring effective cost setups when real cost values are not available
Cost-Sensitive Weighting Data Space	• Simple • Easy to apply	• Extra learning cost for exploring effective cost setups when real cost values are not available • Risk of model overfitting
Algorithm-Level Approaches	Effective in a certain context	Needs of extensive knowledge about the specific learning methods and the application domains
One-Class Learning		• Insufficiency for learning algorithm requiring samples from different classes
Cost-Sensitive Classification		• Deficiencies when real cost value are unavailable

way to bias the learning. This kind of cost values is usually used to manipulate the data space yielding solutions at data level.

Table 4 outlines features of some research solutions. Solutions at algorithm level are usually specific to certain classifier learning algorithms and/or application problems. They are hence effective in certain context. To develop an algorithm-level solution, extensive knowledge about the learning algorithm and the application domain is indispensable. Solutions at data level are therefore more preferable in practice as they are applicable to any classification learning systems without changing their underlying learning strategies. From the view of scientific research, however, some implicit factors, such as the optimal class distribution, and the criterion in selecting representative samples, pertain to some data-level solutions such as those often-used resampling methods. Cost-sensitive boosting, as another solution approach at data level, resamples the data space automatically eliminating the need for exploring the optimal class distribution and the representative

samples. Cost-sensitive boosting is hence a promising approach, even though extra tests are unavoidable to explore the effective cost setups when real cost values are unavailable.

7. Classification of Multi-Class Imbalanced Data

7.1. *Multi-class imbalance problem*

In practice, some applications have more than two classes where the unbalanced class distributions hinder the classification performance. An example is taken from the network intrusion detection problem provided as part of the KDD-CUP'99 contest as mentioned in Sec. 2. Each record in this data set represents either an intrusion or a normal connection. There are four kinds of attacks, where two kinds of attacks, remote-to-local (r2l) and user-to-root (u2r), are represented with only 0.23% and 0.01% of the training samples and 5.20% and 0.07% of the test samples, respectively. The winning entry identified only 8.4% and 13.2% attacks of these two categories as compared with 83.3% and 97.1% identification rates of the other two kinds of attacks, surveillance (probe) and denial-of-service (dos).[24] Obviously, the identification rates of the rare classes lag far behind those of prevalent classes. However, correct identification of each kind of attack is equally important for further security actions.

Due to the complicated situations when multiple classes are present, those methods tackling the class imbalance problem of binary applications are not directly applicable. For binary-class applications, solutions at data level typically change the class size ratio of the two classes, either by oversampling the smaller class or downsampling the prevalent class, and run the learning algorithm many times in search of the "optimal" distribution. When multiple classes are present, these solutions are not practical anymore due to the increased search space. Solutions at algorithm level try to adapt the learning algorithms to bias towards the smaller class. When several smaller classes exist, the situation becomes complicated in adapting the learning algorithm.There are several approaches for extending the binary classifiers to handle multiclass problems.[75] Let $Y = \{c_1, c_2, \ldots, c_k\}$ be the set of classes of the input data. When it is assumed that the classifiers output a binary decision, there are two basic approaches. In the first approach, a classifier between one class and the $k-1$ other classes is trained (in total k classifiers). This is called the $1-r$ (one-against-rest) approach. In the second approach, a classifier is trained between each pair of classes (in total $k(k-1)/2$ classifiers). This classifier is used to distinguish between a pair of classes (c_i, c_j). Instances that do not belong to either c_i or c_j are ignored when constructing the binary classifier for (c_i, c_j). This is called the $1-1$ (one-against-one) approach. In both $1-r$ and $1-1$ approaches, a test sample is classified by combining a set of decisions. One strategy is to use voting, where the object is labeled to the class with the highest number of votes. The obvious problems with these approaches are that both cases increase the learning cost and may lead to ties or contradictory voting[23] among the different classes. To avoid these

ties and inconsequential labels, other efforts are necessary. In the presence of data with imbalanced class distributions, another crucial problem especially with the $1-r$ approach is that one class versus the other classes will worsen the imbalanced distribution even more for the small classes.

7.2. *AdaC2.M1 algorithm*

To advance the classification of multiclass imbalanced data, a cost-sensitive boosting algorithm AdaC2.M1 is reported in Ref. 71. AdaBoost was originally designed for binary classification problems. A straightforward extension of AdaBoost to multiclass problems is AdaBoost.M1.[29] AdaBoost.M1 differs slightly from AdaBoost. The main differences are the replacements of the weight update formula of Eq. (8) by:

$$D^{t+1}(i) = \frac{D^t(i)\exp(-\alpha_t I[h_t(x_i) = y_i])}{Z_t} \tag{22}$$

where

$$I[h_t(x_i) = y_i] = \begin{cases} +1 & \text{if } h_t(x_i) = y_i \\ -1 & \text{if } h_t(x_i) \neq y_i \end{cases} \tag{23}$$

and the final hypothesis of Eq. 9 by:

$$H(x) = \arg\max_{c_i} \left(\sum_{t=1}^{T} \alpha_t [h_t(x) = c_i] \right). \tag{24}$$

AdaC2.M1 is developed by introducing cost items into the learning framework of AdaBoost.M1 by way of AdaC2. The weight update formula of AdaC2.M1 is

$$D^{t+1}(i) = \frac{C_i D^t(i)\exp(-\alpha_t I[h_t(x_i) = y_i])}{Z_t} \tag{25}$$

where α_t is selected the same as Eq. (13). C_i denotes the cost value of the ith sample. The cost value is set with an identical number for samples in the same class. The weighting process of AdaC2.M1 can be interpreted in two steps. At the first step, each sample is first weighted by its cost item (which equals to the misclassification cost of the class that the sample belongs to). Samples of the classes with larger cost values will obtain more sample weights; while samples of the classes with smaller cost values will find their size diminishes. Consequently, the class with the largest cost value will always enlarge its class size at this phase. The second step is the weighting procedure of AdaBoost.M1. That is, weights of false predictions are increased and those of true predictions are decreased. This weighting strategy will always increase class size of class associated with the highest cost value. For those classes with poor performances, relatively higher cost values can be associated such that the learning can bias towards them.

One difficulty in applying the AdaC2.M1 is that the cost matrix is often unavailable for a given problem domain. Empirical methods are not workable anymore since the search space is increased exponentially as the class number increases. Search

algorithms can be applied for setting up effective cost values. However, given the large search space, knowledge-poor search algorithms may be too costly. Heuristic-based methods, such as the Genetic Algorithms (GA), Hill-Climbing and Simulated Annealing, provide a promising alternative, as they are operationally effective methods that direct a search in a problem space within practical time and space limits. Empirical study reported in Ref. 71 shows that AdaC2.M1 is effective in biasing the learning directed by the cost setups generated by GA, and eventually improves the identification performance on those rare instances.

8. Conclusion and Future Research

This review provides a comprehensive study of state-of-the-art research developments on the classification of imbalanced data. This paper, starting from several examples of application domains that the class imbalance problem disturbs, discusses the nature of the class imbalance problem; reviews most standard classifier learning algorithms, such as decision tree, backpropagation neural network, Bayesian network, nearest neighbor, support vector machines, and associative classification, to gain insights into their learning diculties when encountering imbalanced data; presents a thorough survey on reported research solutions to this problem and investigates both their advantages and constraints in an effort to instigate advanced research ideas in future.

Most reported research efforts are exploring solutions to the binary class imbalance problems. There is very few reported work in literature addressing the multiple class imbalance problem. This happens due to: (1) the fact that the biclass imbalanced data is pervasive in a large number of practical projects of great importance, such as anomaly detection; and (2) the even more complicated situation of the data imbalance problem when multiple classes are considered. For an in-depth analysis of data, there are often multiple classes rather than only binary classes. For example, when the anomalous situation becomes more varied, the goal for anomaly detection expects to categorize the types of anomalies rather than only to recognize samples as anomalies or not. Therefore, it is essential to promote more research efforts on the multiple class imbalance problem.

Another interesting research issue open for future investigation is the classification of imbalanced data with multiple class labels. In classic pattern recognition problems, classes are mutually exclusive by definition. Classification errors occur when the classes overlap in the feature space. There is a different situation where the classes by definition are not mutually exclusive. Thereby, a single example may belong to any number of classes. With some application domains of this kind, the class imbalance problem is present. Combined with the multiple class label issue, the class imbalance problem assumes an even more complex situation. Such a situation challenges the classic pattern recognition paradigm and demands a different treatment. Due to the intriguing topics and tremendous potential applications, the classification of imbalanced data will continue to receive more and more attention in both the scientific and the industrial worlds.

References

1. N. Abe, B. Zadrozny and J. Langford, An iterative method for multi-class cost-sensitive learning, *Proc. Tenth ACN SIGKDD Int. Conf. Knowledge Discovery and Data Min.*, Seattle, WA (August 2004), pp. 3–11.
2. R. Akbani, S. Kwek and N. Jakowicz, Applying support vector machines to imbalanced datasets, *Proc. Eur. Conf. Mach. Learn.*, Pisa, Italy (September 2004), pp. 39–50.
3. R. Anand, K. G. Mehrotra, C. K. Mohan and S. Ranka, An improved algorithm for neural network classification of imbalanced training sets, *IEEE Trans. Neural Networks* **4**(6) (1993) 962–969.
4. G. E. A. P. A. Batista, R. C. Prati and M. C. Monard, A study of the behavior of several methods for balancing machine learning training data, *SIGKDD Explorations Special Issue on Learning from Imbalanced Datasets* **6**(1) (2004) 20–29.
5. J. Bradford, C. Kunz, R. Kohavi, C. Brunk and C. E. Brodley, Pruning decision trees with misclassification costs, *Proc. Tenth Eur. Conf. Mach. Learn. (ECML-98)*, Chemnitz, Germany (April 1998), pp. 131–136.
6. L. Breiman, Bagging predictiors, *Mach. Learn.* **24**(2) (1996) 123–140.
7. L. Breiman, Arcing classifier, *Ann. Statist.* **26**(3) (1998) 801–849.
8. L. Breiman, Random forests, *Mach. Learn.* **45** (2001) 5–32.
9. L. Breiman, J. H Freidman, R. A. Olshen and C. J. Stone, *Classification and Regression Trees* (Wadsworth Belmont, 1984).
10. C. Cardie and N. Howe, Improving minority class predication using case-specific feature weights, *Proc. Fourteenth Int. Conf. Mach. Learn.*, Nashville, TN (July 1997), pp. 57–65.
11. K. Carvajal, M. Chacón, D. Mery and G. Acuna, Neural network method for failure detection with skewed class distribution, *INSIGHT, J. British Institute of Non-Destructive Testing* **46**(7) (2004) 399–402.
12. N. V. Chawla, K. Bowyer, L. Hall and W. P. Kegelmeyer, SMOTE: Synthetic minority over-sampling technique, *J. Artif. Intell. Res.* **16** (2002) 321–357.
13. N. V. Chawla, D. A. Cieslak, L. O. Hall and A. Joshi, Automatically countering imbalance and its empirical relationship to cost, *Data Mining and Knowledge Discovery*, http://www.springerlink.com/content/g406864763788315/.
14. N. Chawla, L. Hall and A. Joshi, Wrapper-based computation and evaluation of sampling methods for imbalanced datasets, *Workshop on Utility-Based Data Mining Held in Conjunction with the 11th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Chicago, IL (2005).
15. N. V. Chawla, N. Japkowicz and A. Kolcz, Editorial: special issue on learning from imbalanced data sets, *SIGKDD Explorations Special Issue on Learning from Imbalanced Datasets* **6**(1) (2004) 1–6.
16. N. V. Chawla, N. Japkowicz and A. Kotcz, *Proc. ICML'2003 Workshop on Learning from Imbalanced Data Sets*, ICML (2003).
17. N. V. Chawla, N. Japkowicz and A. Kotcz, *SIGKDD Explorations, Special Issue on Class Imbalances* **6**(1) (ACM, June 2004).
18. N. V. Chawla, A. Lazarevic, L. O. Hall and K. W. Bowyer, SMOTEBoost: improving prediction of the minority class in boosting, *Proc. Seventh Eur. Conf. Principles and Practice of Knowledge Discovery in Database*, Dubrovnik, Croatia (2003), pp. 107–119.
19. T. Cover and P. Hart, Nearest neighbor pattern classification, *IEEE Trans. Inform. Th.* **13** (1967) 21–27.

20. P. Domingos and P. Metacost, Metacost: a general method for making classifiers cost sensitive, *Advances in Neural Networks. Int. J. Patt. Recogn. Artif. Intell.* San Diego, CA (1999), pp. 155–164.
21. G. Dong, X. Zhang, L. Wong and J. Li, CAEP: classification by aggregating emerging patterns, *Proc. Second Int. Conf. Discovery Sci. (DS'99)*, Japan (December 1999), pp. 43–55.
22. R. Duda and P. Hart, *Pattern Classification and Scene Analysis* (John Wiley & Sons, 1973).
23. R. Duda, P. Hart and D. Stork, *Pattern Classification* (Wiley, 2001).
24. C. Elkan, Results of the kdd'99 classifier learning contest, http://www-cse.ucsd.edu/users/elkan/clresults.html.
25. A. Estabrooks, A combination scheme for inductive learning from imbalanced data sets, Master's thesis, Faculty of Computer Science, Dalhousie University, Halifax, Nova Scotia, Canada (2000).
26. K. Ezawa, M. Singh and S. W. Norton, Learning goal oriented bayesian networks for telecommunications risk management, *Proc. Thirteenth Int. Conf. Mach. Learn.*, Bari, Italy (1996), pp. 139–147.
27. W. Fan, S. J. Stolfo, J. Zhang and P. K. Chan, Adacost: misclasification cost-sensitive boosting, *Proc. Sixth Int. Conf. Mach. Learn. (ICML-99)*, Bled, Slovenia (1999), pp. 97–105.
28. T. E. Fawcett and F. Provost, Adaptive fraud detection, *Data Mining and Knowledge Discovery* **1**(3) (1997) 291–316.
29. Y. Freund and R. E. Schapire, A decision-theoretic generalization of on-line learning and an aplication to boosting, *J. Comput. Syst. Sci.* **55**(1) (1997) 119–139.
30. J. Friedman, T. Hastie and R. Tibshirani, Additive logistic regression: a statistical view of boosting, *Ann. Statist.* **28**(2) (2000) 337–374.
31. N. Friedman, D. Geiger and M. Goldszmidt, Bayesian network classifiers, *Mach. Learn.* **29**(2/3) (1997) 131–163.
32. H. Guo and H. L. Viktor, Learning from imbalanced data sets with boosting and data generation: the databoost-IM approach, *SIGKDD Explorations Special Issue on Learning from Imbalanced Datasets* **6**(1) (2004) 30–39.
33. J. A. Hanley and B. J. McNeil, The meaning and use of the area under a receiver operating characteristic (ROC) curve, *Intell. Data Anal. J.* **143** (1982) 29–36.
34. D. Hecherman, Bayesian networks for knowledge discovery, *Advance in Knowledge Discovery and Data Mining*, Chapter 11, (AAAI Press/The MIT Press, 1996), pp. 273–305.
35. J. Hertz, A. Krogh and R. G. Palmer, *Introduction to the Theory of Neural Computation* (Addison Wesley, 1991).
36. R. C. Holte, L. E. Acker and B. W. Porter, Concept learning and the problem of small disjuncts, *Proc. Eleventh Int. Joint Conf. Artif. Intell.*, Detroit, Michigan (August 1989), pp. 813–818.
37. N. Japkowicz, *Proc. AAAI'2000 Workshop on Learning from Imbalanced Data Sets, AAAI Tech Report WS-00-05* (AAAI, 2000).
38. N. Japkowicz, Concept-learning in the presence of between-class and within-class imbalances, *Proc. Fourteenth Conf. Canadian Soc. for Computational Studies of Intelligence*, Ottawa, Canada (June 2001), pp. 67–77.
39. N. Japkowicz, Supervised versus unsupervised binary-learning by feedforward neural networks, *Mach. Learn.* **41**(1) (2001).
40. N. Japkowicz and S. Stephen, The class imbalance problem: a systematic study, *Intell. Data Anal. J.* **6**(5) (2002) 429–450.

41. M. V. Joshi, *Learning Classifier Models for Predicting Rare Phonemena*, PhD thesis, University of Minnesota, Twin Cites, Minnesota, USA (2002).
42. M. V. Joshi, V. Kumar and R. C. Agarwal, Evaluating boosting algorithms to classify rare classes: comparison and improvements, *Proc. First IEEE Int. Conf. Data Min. (ICDM'01)* (2001).
43. M. S. Kamel and N. Wanas, Data dependence in combining classifiers, *Proc. Fourth Int. Workshop on Multiple Classifiers Systems*, Surrey, UK (June 2003).
44. H. Kück, Bayesian formulations of multiple instance learning with applications to general object recognition, Master's thesis, University of British Columbia, Vancouver, BC, Canada (2004).
45. J. Kittler, M. Katef, R. Duin and J. Matas, On combining classifiers, *IEEE Trans. Patt. Anal. Mach. Intell.* **20**(3) (1998).
46. M. Kubat, R. Holte and S. Matwin, Machine learning for the detection of oil spills in satellite radar images, *Mach. Learn.* **30** (1998) 195–215.
47. D. Lewis and W. Gale, Training text classifiers by uncertainty sampling, *Proc. Seventeenth Annual Int. ACM SIGIR Conf. Research and Development in Information*, New York, NY (August 1998), pp. 73–79.
48. J. Li, G. Dong, K. Ramamohanarao and L. Wong, DeEPs: a new instance-based lazy discovery and classification system, *Mach. Learn.* **54**(2) (2004) 99–124.
49. W. Li, J. Han and J. Pei, CMAR: accurate and efficient classification based on multiple class-association rules, *Proc. 2001 IEEE Int. Conf. Data Min. (ICDM'01)*, San Jose, CA (November 2001), pp. 369–376.
50. Y. Lin, Y. Lee and G. Wahba, Support vector machines for classification in nonstandard situations, *Mach. Learn.* **46** (2002) 191–202.
51. C. X. Ling and C. Li, Data mining for direct marketing: problems and solutions, *Proc. Fourth ACM SIGKDD Int. Conf. Knowledge Discovery and Data Min.*, New York, NY (August 1998), pp. 73–79.
52. C. X. Ling and C. Li, Decision trees with minimal costs, *Proc. 21st Int. Conf. Machine Learn.*, Banff, Canada (July 2004).
53. B. Liu, W. Hsu and Y. Ma, Integrating classification and association rule mining, *Proc. Fourth ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, New York, NY (August 1998), pp. 80–86.
54. B. Liu, W. Hsu and Y. Ma, Mining association rules with multiple minimum supports, *Proc. Fifth ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, San Diego, CA (August 1999), pp. 337–341.
55. B. Liu, Y. Ma and C. K. Wong, Improving an association rule based classifier, *Proc. 4th Eur. Conf. Principles of Data Mining and Knowledge Discovery*, Lyon, France (September 2000), pp. 504–509.
56. L. M. Manevitz and M. Yousef, One-class svms for document classification, *J. Mach. Learn. Res.* **2** (2001) 139–154.
57. D. Margineantu, Class probability estimation and cost-sensitive classification decisions, *Proc. 13th Eur. Conf. Mach. Learn.*, Helsinki, Finland (August 2002), pp. 270–281.
58. P. M. Murph and D. W. Aha, *UCI Repository of Machine Learning Databases*, Department of Information and Computer Science, University of California: Irvine (1991).
59. J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Infernence* (Morgan Kaufmann, 1988).
60. R. C. Prati and G. E. A. P. A. Batista, Class imbalances versus class overlapping: an analysis of a learning system behavior, *Proc. Mexican Int. Conf. Artif. Intell. (MICAI)*, Mexico City, Mexico (April 2004), pp. 312–321.

61. F. Provost and T. Fawcett, Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions, *Proc. Third Int. Conf. Knowledge Discovery and Data Mining (KDD-97)*, Newportbeach, CA (August 1997), pp. 43–48.
62. J. R. Quinlan, Improved estimates for the accuracy of small disjuncts, *Mach. Learn.* **6** (1991) 93–98.
63. J. R. Quinlan, *C4.5: Programs For Machine Learning* (Morgan Kaufmann Publishers, 1993).
64. J. R. Quinlan, Induction of decision trees, *Mach. Learn.* **1**(1) (1986) 81–106.
65. B. Raskutti and A. Kowalczyk, Extreme rebalancing for SVMs: a case study, *Proc. Eur. Conf. Mach. Learn.*, Pisa, Italy (September 2004), pp. 60–69.
66. P. Riddle, R. Segal and O. Etzioni, Representation design and brute-force induction in a Boeing manufactoring domain, *Appl. Artif. Intell.* **8** (1991) 125–147.
67. J. Rissanen, Modeling by the shortest data description, *Automatica* **14** (1978) 465–471.
68. R. E. Schapire, The boosting approach to machine learning — An overview, *MSRI Workshop on Nonlinear Estimation and Classification*, Berkeley, CA (March 2002), pp. 149–172.
69. R. E. Schapire and Y. Singer, Improved boosting algorithms using confidence-rated predictions, *Mach. Learn.* **37**(3) (1999) 297–336.
70. S. Steward, Lighting the way in '97, *Cellular Business* (1997), p. 23.
71. Y. Sun, M. S. Kamel and Y. Wang, Boosting for learning multiple classes with imbalanced class distribution, *Proc. Sixth IEEE Int. Conf. Data Min.*, HongKong, China (December 2006), pp. 592–602.
72. Y. Sun, M. S. Kamel, A. K. C. Wong and Y. Wang, Cost-sensitive boosting for classification of imbalanced data, *J. Patt. Recogn.* **40**(12) (2007) 3358–3378.
73. Y. Sun, A. K. C. Wong and Y. Wang, Parameter inference of cost-sensitive boosting algorithms, *Fourth Int. Conf. Machine Learning and Data Mining (MLDM)*, Leipzig, Germany (July 2005), pp. 21–30.
74. P. Tan, M. Steinbach and V. Kumar, *Introduction to Data Mining* (Addison Wesley, 2006).
75. D. M. J. Tax and R. P. W. Duin, Using two-class classifiers for multiclass classification, *Int. Conf. Patt. Recogn.*, Quebec, Canada (2002).
76. K. M. Ting, The problem of small disjuncts: its remedy in decision trees, *Proc. Tenth Canadian Conf. Artif. Intell.*, Banff, Alberta (May 1994), pp. 91–97.
77. K. M. Ting, A comparative study of cost-sensitive boosting algorithms, *Proc. 17th Int. Conf. Mach. Learn.*, Stanford University, CA (2000), pp. 983–990.
78. P. Turney, Types of cost in inductive concept learning, *Proc. Workshop on Cost-Sensitive Learning at the 17th Int. Conf. Mach. Learn.*, Stanford University, CA (2000), pp. 15–21.
79. V. Vapnik and A. Lerner, Pattern recognition using generalized portrait method, *Automat. Remont Contr.* **24** (1963) 774–780.
80. D. Walters and W. Wilkinson, Wireless fraud, now and in the future: a view of the problem and some solutions, *Mobile Phone News* (1994), pp. 4–7.
81. K. Wang, S. Zhou and Y. He, Growing decision tree on support-less association rules, *Proc. Sixth ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'00)*, Boston, MA (August 2000), pp. 265–269.
82. Y. Wang, *High-Order Pattern Discovery and Analysis of Discrete-Valued Data Sets*, PhD thesis, University of Waterloo, Waterloo, Ontario, Canada (1997).
83. Y. Wang and A. K. C. Wong, From association to classification: Inference using weight of evidence, *IEEE Trans. Knowl. Data Engin.* **15**(3) (2003) 764–767.

84. G. Weiss, Mining with rarity: a unifying framework, *SIGKDD Explorations Special Issue on Learning from Imbalanced Datasets* **6**(1) (2004) 7–19.
85. G. Weiss and F. Provost, Learning when training data are costly: the effect of class distribution on tree induction, *J. Artif. Intell. Res.* **19** (2003) 315–354.
86. A. K. C. Wong and Y. Wang, High order pattern discovery from discrete-valued data, *IEEE Trans. Knowl. Data Engin.* **9**(6) (1997) 877–893.
87. G. Wu and E. Y. Chang, Class-boundary alignment for imbalanced dataset learning, *Proc. ICML'03 Workshop on Learning from Imbalanced Data Sets*, Washington, DC (August 2003).
88. X. Yin and J. Han, CPAR: Classification based on predictive association rules, *Proc. 2003 SIAM Int. Conf. Data Min. (SDM'03)*, San Francisco, CA (May 2003), pp. 331–335.
89. B. Zadrozny and C. Elkan, Learning and making decisiions when costs and probabilities are both unknown, *Proc. Seventh Int. Conf. Knowledge Discovery and Data Mining*, San Francisco, CA (August 2001), pp. 204–213.
90. B. Zadrozny, J. Langford and N. Abe, Cost-sensitive learning by cost-proportionate example weighting, *Proc. Third IEEE Int. Conf. Data Min.*, Melbourne, Florida (November 2003), pp. 435–442.
91. J. Zhang and I. Mani, KNN approach to unbalanced data distributions: a case study involving information extraction, *Proc. ICML'03 Workshop on Learning from Imbalanced Data Sets*, Washington, DC (August 2003).
92. Z. H. Zhou and X. Y. Liu, Training cost-sensitive neural networks with methods addressing the class imbalance problem, *IEEE Trans. Knowl. Data Engin.* **18**(1) (2006) 63–77.

Yanmin Sun received her PhD from the University of Waterloo in 2007 and joined Pattern Discovery Technology Ltd, Waterloo, Canada, as a research scientist in 2008. She currently holds an industrial post-doctoral fellowship sponsored by NSERC of Canada. Between 2000 and 2001, Dr. Sun was a visiting scientist in the Department of Computer Sciences at the University of Waterloo, Canada. Before coming to Canada, Dr. Sun was an associated professor at Wuhan University of Technology, Wuhan, China.

Her research interests include data mining, machine learning and statistical analysis.

Andrew K. C. Wong received his PhD from the Carnegie Mellon University, Pittsburgh, Pennsylvania in 1968, and taught there for several years thereafter. From 1980 to 2002, he was a professor of Systems Design Engineering and the Founding Director of the PAMI Lab, at the University of Waterloo (UW), Waterloo, Ontario, Canada. From 2000 to 2003, he served as a Distinguished Chair Professor of the Computing Department at Hong Kong Polytechnic University, Hong Kong. In 2002, he was conferred Distinguished Professor Emeritus at UW. Since then, he has served at UW as an Adjunct Professor in Systems Design Engineering, Electrical and Computer Engineering (01-08), and the School of Computer Sciences (97-08). He has published extensively and has been invited as a keynote/plenary speaker for IEEE related conferences. He was on the IEEE Distinguished Speaker Program and served as a General Chair of ISTED Conference of Robotics and Systems in Hawaii, 1996 and IEEE/RSJ International Conference in Victoria, Canada, 1998. He is a Fellow of the IEEE (for his contribution in machine intelligence, computer vision, and intelligent robotics).

Dr. Wong is a founder, and retired director (93-03) of Virtek, a publicly traded company and a leader in laser vision technology. He was for years its president (1986 to 1993) and Chairman (1993 to 1997). In 1997, he co-founded Pattern Discovery Software Technology Ltd. and has served as its Chairman ever since. The technologies he developed have been applied to space robotics, manufacturing, business, education, text mining, petro industry and biomedical industry.

His current research interests include knowledge discovery, data mining, machine learning, data analysis, pattern post processing and analysis and their applications.

Mohamed S. Kamel received the B.Sc. (Hons) EE (Alexandria University), M.A.Sc (McMaster University), Ph.D (University of Toronto).

He joined the University of Waterloo, Canada in 1985 where he is at present Professor and Director of the Pattern Analysis and Machine Intelligence Laboratory at the Department of Electrical and Computer Engineering and holds a University Research Chair. Professor Kamel held Canada Research Chair in Cooperative Intelligent Systems from 2001 to 2008.

He has authored and co-authored over 400 papers in journals and conference proceedings, 11 edited volumes, 16 chapters in books, 2 patents and numerous technical and industrial project reports. Under his supervision, 81 Ph.D and M.A.Sc students have completed their degrees.

He is the Editor-in-Chief of the *Int. J. Robotics and Automation*, Associate Editor of the *IEEE SMC, Part A*, *Pattern Recognition Letters*, *Cognitive Neurodynamics J.* and *Pattern Recognition J.* He is also member of the editorial advisory board of the *Int. J. Image and Graphics* and the *Intelligent Automation and Soft Computing Journal*. He also served as Associate Editor of *Simulation*, *the Journal of the Society for Computer Simulation*.

Dr. Kamel is member of ACM, PEO, Fellow of IEEE, Fellow of the Engineering Institute of Canada (EIC), Fellow of the Canadian Academy of Engineering (CAE) and Fellow of the International Association of Pattern Recognition (IAPR). He served as consultant for General Motors, NCR, IBM, Northern Telecom and Spar Aerospace. He is co-founder of Virtek Vision Inc. of Waterloo (now part of Gerber Technology Co) and chair of its Technology Advisory Group. He served as member of the board from 1992–2008 and VP research and development from 1987 to 1992.

His research interests are in computational intelligence, pattern recognition, machine learning and cooperative intelligent systems.