# On Identifying Critical Nuggets of Information during Classification Tasks

David Sathiaraj and Evangelos Triantaphyllou

**Abstract**—In large databases, there may exist critical nuggets—small collections of records or instances that contain domain-specific important information. This information can be used for future decision making such as labeling of critical, unlabeled data records and improving classification results by reducing false positive and false negative errors. This work introduces the idea of critical nuggets, proposes an innovative domain-independent method to measure criticality, suggests a heuristic to reduce the search space for finding critical nuggets, and isolates and validates critical nuggets from some real-world data sets. It seems that only a few subsets may qualify to be critical nuggets, underlying the importance of finding them. The proposed methodology can detect them. This work also identifies certain properties of critical nuggets and provides experimental validation of the properties. Experimental results also helped validate that critical nuggets can assist in improving classification accuracies in real-world data sets.

**Index Terms**—Data mining, classification, critical nuggets, outliers, classification accuracy, class boundary, duality

## 1 Introduction

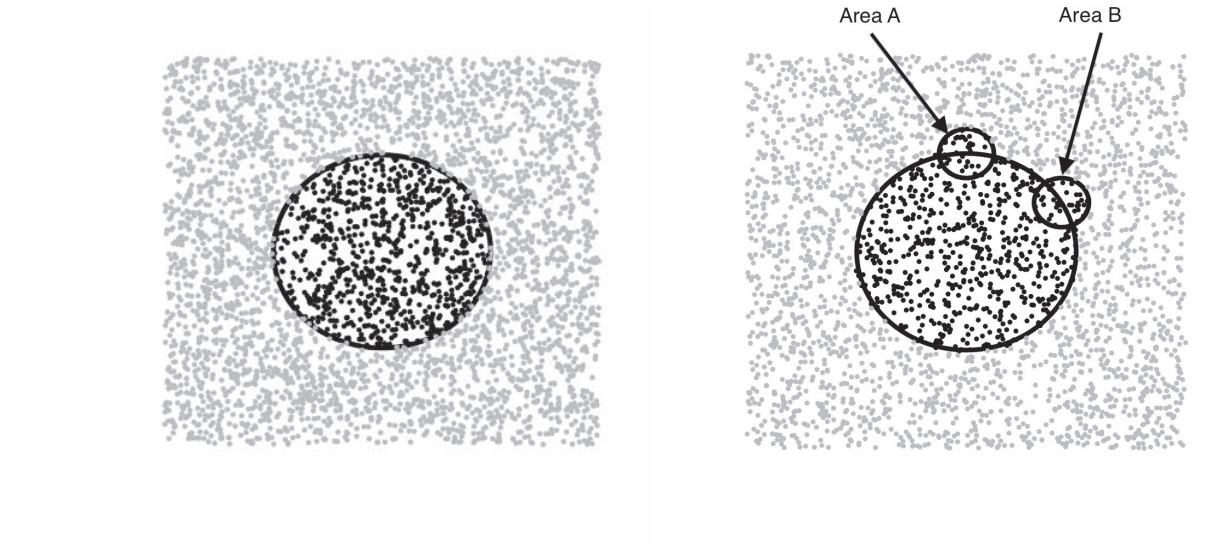
In recent times, detecting patterns and outliers has emerged as an important area of work in the field of data mining. It has several applications including detecting fraud in business transactional data [1], identifying network intrusions [1], isolating abnormal trends in time-series data [2], and picking out suspicious criminal activity [3]. A lot of work in data mining has been devoted to finding interesting patterns or rules in data sets [4], [5], and [6]. In [7], research was extended to the mining of outliers and the concept of distance-based outliers was proposed to identify records that are different from the rest of the data set. A good definition of an outlier is that of [8], *an outlier is an observation that deviates so much from other observations as to arouse suspicions that it was caused by a different mechanism*. Distance-based measures as in [9], [10], and [1] have been used in algorithms to delineate outliers or abnormal records from normal records. However, not much work has focused on finding critical nuggets of information that may be hidden in data sets. These nuggets of information may not always be detected by pattern mining methods or by distance-based outlier detection methods as nuggets may not conform to a specific pattern and may not be outliers.

A simple visual example is outlined in Fig. 1, where the data set with protrusions around the circular region (Fig. 1b) might be considered more interesting than the simpler circular region (Fig. 1a). The protrusions serve as critical nuggets of information that are more interesting as these areas can be studied further for improved classification results. In real life, one such example is if one were asked to identify benign tumors that are very close to becoming malignant. Such data records, if they were to exist in a data set, would not "deviate so much" from both benign and malignant observations, but instead would lie extremely close to the class boundary separating the benign and malignant classes. They may not necessarily "deviate enough" to be captured by distance-based outlier detection methods. In tight elections, the undecided voters are crucial in deciding the outcome. The problem of identifying the undecided voters and the attributes that can tilt them to the opposite side is valuable information. Another example is to predict cases from bank loan data that are very close to bankruptcy. In this setting, the important task is to identify cases before they become bankrupt. In many applications, the problem is not of finding individual outliers, but instead, of finding critical nuggets (subsets of data) that provide valuable information, which in turn can be used for improved classification results and a better understanding of false positive and false negative errors.

This paper considers the notion of identifying subsets of critical data instances in data sets. Critical nuggets of information can take the following form during classification tasks: small subsets of data instances that lie very close to the class boundary and are sensitive to small changes in attribute values, such that these small changes result in the switching of classes. Such critical nuggets have an intrinsic worth that far outweighs other subsets of the same data set. In classification tasks, consider a data set that conforms to a certain representation or a classification model. If one were to perturb a few data instances by making small changes to some of their attribute values, the original classification model representing the data set changes. Also, if one were to remove those data instances, the original model could change significantly. The magnitude of changes to the original model provides clues to the criticality of such data instances, as more critical data

- *D. Sathiaraj is with the NOAA Southern Regional Climate Center and the Department of Computer Science, Louisiana State University, E328 Howe Russell, Baton Rouge, LA 70803. E-mail: davids@srcc.lsu.edu.*
- *E. Triantaphyllou is with the Department of Computer Science, Louisiana State University, 298 Coates Hall, Baton Rouge, LA 70803. E-mail: trianta@lsu.edu.*

*Manuscript received 5 May 2011; revised 14 Jan. 2012; accepted 1 Apr. 2012; published online 23 May 2012.*
*Recommended for acceptance by Y. Chen.*
*For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2011-05-0249.*
*Digital Object Identifier no. 10.1109/TKDE.2012.112.*




(a) An uninteresting circular region. (b) Interesting protrusions around the circular region.

Fig. 1. Gray dots indicate points of one class while black dots indicate points of another class.

instances tend to impact the model more significantly than data instances that are comparatively noncritical. This idea is exploited in this paper to introduce the notion of critical nuggets, to define a metric for criticality and for the eventual mining of critical nuggets.[1]

This paper is organized as follows: Section 2 provides a discussion on some related work from the literature. A motivation behind finding critical nuggets is also provided. In Section 3, a criticality metric, $CR_{score}$ is defined. This metric is used to calculate the criticality of a subset of data instances. In this section, algorithms have been provided to help in calculating the metric and to reduce the search space by isolating approximate class boundaries in training data. In Section 4, a set of experiments are presented. The experiments were conducted on some real-world data sets. For each of the data sets, critical nuggets were identified and properties such as the ones related to some duality relations were validated. Experimental results on how critical nuggets can help in improving classification accuracy have been provided. Finally, Section 5 presents some concluding remarks and some ideas for future work.

## 2 Related Work and Problem Motivation

In classification problems, the main goal is to derive an accurate representative data model that can correctly classify new test data instances. The accuracy of the classification model can be affected by the presence of outliers in a data set and the inability to correctly classify data records near the boundary.

Considering the first case of outliers are critical nuggets different from outliers and can existing approaches in outlier detection help in finding critical nuggets? Critical nuggets in certain cases may involve outliers, but this may not always be true. In the example of the previous section, cells in tumors may not show anomalous behavior on an individual basis but collectively, such cells may contain critical pieces of information. In [11], the authors note that the performance of a distance-based outlier detection method *"greatly relies on a distance measure, defined between a pair of data instances, which can effectively distinguish between normal and anomalous instances. Defining distance measures between instances can be challenging when the data are complex."* Moreover, critical nuggets that belong to a data set may not be at a great "distance" from the other "normal" points, and may end up being classified as "normal." For a comprehensive survey on outlier detection methods, please refer to the extensive survey in [11]. In the field of distance-based outlier detection, researchers have focused on proposing algorithms that reduce the time complexity $O(n^2)$ of calculating distances [12], [9], [10], and [13]. Work has also been done on density-based outlier detection such as [14] where outliers are defined as objects that show anomalous trends with respect to their local neighborhoods and tend to lie in a less dense area with respect to a more dense local neighborhood. In [15], the concept of density-based detection is extended to cluster-based outlier detection where the approach does not only find single point outliers but instead clusters of outliers. Intuitively, cluster-based outlier methods may not necessarily lead to identifying critical areas such as the protrusions in Fig. 1b, which do not lie at a great "distance" from the rest of the points.

With this differentiation between critical nuggets and outliers, can critical nuggets be found among data records near the boundary? One can utilize an intuition that is motivated by two commonly occurring scenarios in classification algorithms:

1. *Points near the boundary, in general, are critical.* The deciding factor for most classification algorithms is how accurately the algorithm classifies the points near the class boundaries (see also [6]). The points that are far from the class boundaries are the "slamdunk," easy cases, where the impact of misclassification is pretty minimal. However, the

1. From this point on, the use of the terms "critical nuggets" and "critical sets" refer to the same concept.

points near the class boundaries are more susceptible to misclassification. These points are critical in deciding the accuracy of any classification algorithm. The need for understanding this problem can be best explained by the real-world example of a data set describing some type of cancer related cases. Most classification algorithms can easily classify a full-blown cancer case or a clearly cancer-free case. On the other hand, the border-line cases which may exhibit subtle symptoms of cancer are critical, as early detection can save a life. Hence, uncertain regions in and around the class boundaries can be crucial for identifying critical nuggets.

2. *Certain boundary features can be critical.* Second, as a corollary to the first scenario, there are certain regions along the boundary (and, hence, the boundary points near those regions) where the problem of classification becomes more difficult, as compared to less problematic boundary points. As a simplistic example, consider a geographical data set that corresponds to a political boundary. Classifying records near sharply changing outlines (such as along a complex sea coast of a political boundary) is more difficult than straight edges of the boundary. For more complex data sets, there maybe certain inherent complex properties that render the points near the boundary difficult to classify. Such regions have a higher potential for harboring critical nuggets.

In summary, using the first scenario, the search for critical nuggets is narrowed to a region near the boundary separating the classes. On the basis of the second scenario, where certain boundary features are more complex than others, the criticality metric (the $CR_{score}$) has been defined in such a way that it yields higher scores for sets of data records that lie near complex boundary features. In other words, the greater the complexity of a boundary feature, the higher the probability of misclassification becomes, resulting in higher scores being assigned for points near that complex boundary.

## 3 Problem Description

### 3.1 Formal Notation

Consider a training data set $T_r$ comprised of $m$ data instances, $n$ attributes and two classes, denoted as "+" and "−" (these names are arbitrary). Consider also a sample neighborhood $N$ to be a subset of $T_r$, comprised of $d$ data instances (i.e., number of rows in $N$) of the data set $T_r$.

Besides the above notation, the following notation will also be used:

- $C$—any classification algorithm.
- $T_r^+$—the subset of $T_r$ comprised of only the "+" class.
- $T_r^-$—the subset of $T_r$ comprised of only the "−" class.
- $A$—the set of attributes in $T_r$ denoted as $\{A_1, A_2, A_3, \ldots, A_n\}$.
- $D$—the set of the data instances in $T_r$ denoted as $\{D_1, D_2, D_3, \ldots, D_m\}$.
- $d_j^+$—the number of instances in $N$ that switch classes when attribute $A_j$ is increased by $\delta_j$.
- $d_j^-$—the number of instances in $N$ that switch classes when attribute $A_j$ is decreased by $\delta_j$.
- $N[A_j]$—column vector of size $d \times 1$, formed by choosing only attribute $A_j$ from matrix (neighborhood) $N$.
- $N[A_1 : A_j]$—matrix of size $d \times j$, formed by choosing attributes $A_1$ through $A_j$.
- $N_1.N_2$—appending two matrices, column-wise (e.g., if $N_1$ and $N_2$ were each of size $2 \times 3$, the combined matrix would be of size $2 \times 6$).
- $M_0$—the model obtained by using classification algorithm $C$ on training set $T_r$.
- $P_0$—the vector of predicted class values by model $M_0$ when applying $M_0$ on a neighborhood of instances, $N$.
- $B^+$—the set of "+" points near the boundary separating the two classes, "+" and "−".
- $B^-$—the set of "−" points near the boundary separating the two classes, "+" and "−".
- $\overline{B^+}$—the set of "+" points not near the boundary separating the two classes, "+" and "−".
- $\overline{B^-}$—the set of "−" points not near the boundary separating the two classes, "+" and "−".

From the above definitions, it follows that: $|T_r^+| = |B^+| + |\overline{B^+}|$, $|T_r^-| = |B^-| + |\overline{B^-}|$ and $|T_r| = |T_r^+| + |T_r^-|$, where $|X|$ denotes the cardinality of set $X$.

### 3.2 Definition of Criticality

One can look at criticality as the intrinsic worth of a subset of records. This worth is realized when the records are collectively removed from the data set or their attribute values undergo perturbation. Initial steps in defining the critical metric ($CR_{score}$) relied on the effect of removing a neighborhood of data instances on a classification model. A classification model $M_0$ was initially derived by applying a classification algorithm $C$ on the training data $T_r$. Then, a neighborhood of data instances, $N$, was removed from $T_r$ and a new classification model $M_1$ was obtained by applying $C$ on $T_r - N$. The difference in predictions made by $M_0$ and $M_1$, divided by the number of data instances in $N$, was initially used as the criticality measure, $CR_{score}$. The greater the difference in predictions between $M_0$ and $M_1$, the higher the $CR_{score}$ was and vice versa. Some 2D data sets were used to validate this approach. However, this metric could not isolate all the critical areas even though some of the critical areas were obvious during a simple visual inspection of the 2D validation data set. Hence, a different approach was considered and upon validation using the 2D data sets, this new approach in deriving the $CR_{score}$ is outlined. More experiments (as described in Section 4) with some real-world data sets further support the choice of this metric.

Consider a training data set $T_r$ with $m$ data instances, each instance having $n$ attributes denoted as $A_j$ ($j \epsilon \{1, 2, \ldots, n\}$). The underlying assumption is that all attributes are numeric and not categorical. From $T_r$, form a neighborhood $N$, by choosing a data instance $D_i$ as a center and finding a group of points that belong to the same class as $D_i$ and lying within a distance $R$ from $D_i$. For simplicity, let us say that the neighborhood $N$ is comprised of $d$ data instances. The selection of parameters

$R$ and $D_i$ used in forming a neighborhood $N$ is further described in Section 3.8. First, a classification model $M_0$ is generated by applying a classification algorithm $C$ to the training data set $T_r$. Using the classification model $M_0$, one can predict the class labels for the different data instances in question. For the $d$ instances in neighborhood $N$, consider an attribute $A_j$. Also, for the $d$ instances, the attribute $A_j$ can be increased or decreased in magnitude. A parameter denoted by $\delta_j$ is used for this and $\delta_j$ varies for different attributes in neighborhood $N$. The calculation of this parameter is further explained in Section 3.5. After increasing $A_j$ by an extent $\delta_j$ for just the $d$ instances, the classification model $M_0$ for the new class labels for the $d$ instances is queried. The average number of data instances that have switched classes in neighborhood $N$ is computed and is denoted as $w_j^+$. If all the data instances in $N$ switch classes, then one can infer that $N$ is very sensitive to changes with respect to attribute $A_j$. The same test is applied on $N$ by decreasing $A_j$ by the same extent $\delta_j$ and find $w_j^-$ by querying the classification model $M_0$ for the new class labels. For the attribute $A_j$, the average of $w_j^+$ and $w_j^-$ is computed to get $w_j$. Repeating this process for all $n$ attributes, the average of the $w_j$ scores is computed as the $CR_{score}$ for the neighborhood $N$.

Formally, the critical score is defined as

$$CR_{score} = \frac{\sum_{j=1}^{n}(w_j)}{n}, \tag{1}$$

where: $w_j = \frac{w_j^+ + w_j^-}{2}$, $w_j^+ = \frac{d_j^+}{d}$, and $w_j^- = \frac{d_j^-}{d}$.

### 3.3 Properties of the Critical Score

Based on the score developed above, the following properties are outlined:

- Each $w_j$ value lies in the interval [0,1]. Each $w_j$ value is calculated by averaging $w_j^+$ and $w_j^-$ and $w_j^+ \epsilon [0,1]$ and $w_j^- \epsilon [0,1]$. Hence, $w_j \epsilon [0,1]$.
- $CR_{score} \epsilon [0,1]$, as there are $n$ instances of $w_j$ and $CR_{score}$ is averaged over $n$.
- In calculating the critical score, the main idea is to find as many attributes that are sensitive to small changes (such as the increase and decrease of $A_j$ by $\delta_j$) that propel an entire subset from one class to another. The greater the number of attributes that are sensitive to such changes, the higher is the resulting $CR_{score}$.
- A neighborhood of data instances $N_1$ is said to be more critical than a neighborhood $N_2$, if and only if $CR_{score}(N_1) > CR_{score}(N_2)$.

### 3.4 Computing the $CR_{score}$

Using the description in Section 3.2 on how the $CR_{score}$ is calculated, the algorithm $GetNuggetScore$ is developed and is outlined as Fig. 2.

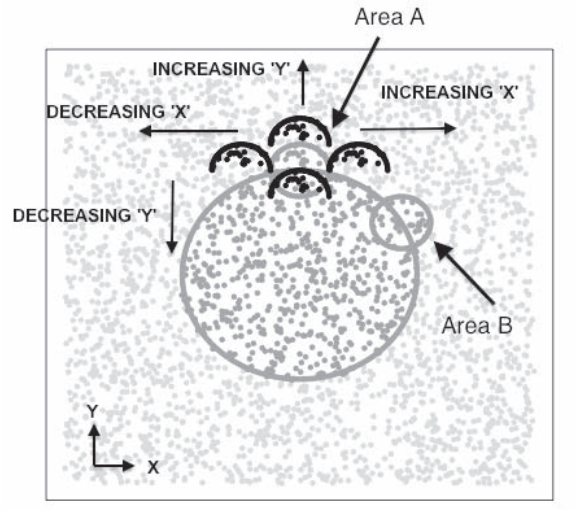
The computational complexity of the algorithm is derived as follows: Deriving the model $M_0$ is dependent on the complexity of the chosen classification algorithm ($C$). The complexity of the classification algorithm is denoted as $t(C)$. Each attribute $A_j$ is analyzed by checking if increasing or decreasing the values of the attributes by an extent $\delta_j$, switches the class label. Hence, for each attribute, the model $M_0$ is queried twice. There are $d$ data instances in $N$ and, thus, for each attribute there are $2 \times d$ queries. Since there are $n$ attributes, the complexity of the for-loop in Fig. 2 is $O(dn)$. When $d \ll n$, the complexity of the for-loop becomes $\approx O(n)$. The total complexity of the algorithm is $O(t(C) + dn)$ ($\approx O(t(C) + n)$ when $d \ll n$).

**Require:** $T_r$: the training set, $N$: a neighborhood of data instances and $R$: a distance parameter used in creating the neighborhood set, $N$
1: $M_0$ = Model resulting from applying $C$ on training set, $T_r$
2: $m$ = number of data instances in $T_r$
3: $n$ = number of attributes in $T_r$
4: $ScoresArray = \phi$
5: **for** each $j$ in {1,2, ..., $n$} **do**
6: $\delta_j$ = max($N[A_j]$) - min($N[A_j]$) {Finding the maximum and minimum values in vector $N[A_j]$}
7: **if** $\delta_j$=0 **then**
8: $\delta_j = R$
9: **end if**
10: $V = N[A_j] + \delta_j$ {Extract $A_j$, increment all values in $A_j$ by $\delta_j$}
11: $N_1 = N[1 : A_{j-1}].V.N[A_{j+1} : A_n]$ {Generate new matrix, keep previous columns and replace $A_j$ by $V$ instead}
12: $P_0$ = Query $M_0$ to obtain new class labels for $N_1$
13: $w_j^+$ = Average number of instances in $P_0$ that have switched classes
14: $V = N[A_j] - \delta_j$ {Decrement all values in column $A_j$ by $\delta_j$}
15: $N_2 = N[1 : A_{j-1}].V.N[A_{j+1} : A_n]$
16: $P_0$ = Query $M_0$ to obtain new class labels for $N_2$
17: $w_j^-$ = Average number of instances in $P_0$ that have switched classes
18: $w_j = (w_j^+ + w_j^-)/2$
19: Append $w_j$ to $ScoresArray$
20: **end for**
21: $CR_{score} = \frac{\sum(ScoresArray)}{n}$
22: **return** $CR_{score}$

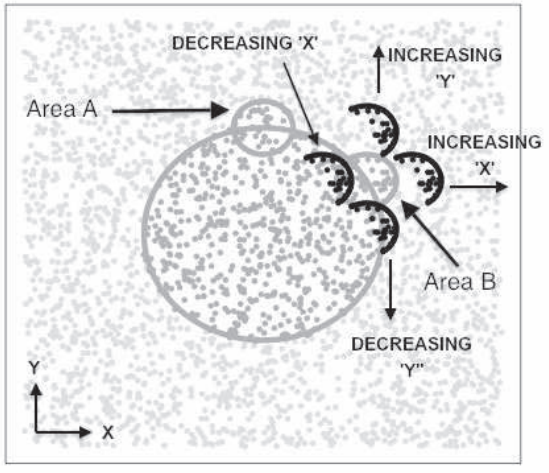
Fig. 2. The $GetNuggetScore$ algorithm.

### 3.5 Choosing $\delta_j$

In the algorithm $GetNuggetScore$, the parameter $\delta_j$ is used. This parameter is a measure of how much the attribute values should increase or decrease. For each attribute $A_j$, an appropriate value of $\delta_j$ is calculated by computing the range of values of $A_j$ in neighborhood $N$. This is to ensure that all the attribute values $A_j$ get a chance to help the data points in $N$ to switch class labels. Fig. 3 is used to analyze a closely related issue. The data set in Fig. 3 has two attributes—let us say attributes "$X$" and "$Y$" and two subsets of interest denoted as area "A" and area "B." Considering Fig. 3a, increasing and decreasing the attribute values of "$X$" by $\delta_j$ results in a majority of area A to be inside the opposite class. However, increasing and decreasing attribute values along "$Y$" results in the switching of class values only when increasing "$Y$" values. So, for this example, only along three out of four directions (increasing and decreasing of attributes), can there be a switch of classes. In this case, in three directions, all points in the neighborhood ended up shifting to the opposite class. Hence, using the defined $CR_{score}$, the score for area A would be $3/4 = 0.750$. Performing a similar analysis on area B, reveals (Fig. 3b), that in only two out of four directions (increasing attributes "$X$" and "$Y$") result in switching of class labels for all points in the neighborhood. Decreasing the attribute $Y$ by $\delta_j$




(a) An example to illustrate shifting of points in neighborhood 'Area A'.




(b) An example to illustrate shifting of points in neighborhood 'Area B'.

Fig. 3. Shifting of data instances.

**Require:** $T_r$: the training set, $N$: a neighborhood of data instances and $R$: a distance parameter used in creating the neighborhood set, $N$

1: $M_0$ = Model resulting from training $C$ using training set, $T_r$
2: $m$ = number of data instances in $T_r$
3: $n$ = number of attributes in $T_r$
4: $ScoresArray = \phi$
5: **for** each $j$ in {1,2, ..., $n$} **do**
6: $\delta_j$ = max($N[A_j]$) - min($N[A_j]$)
7: **if** $\delta_j$=0 **then**
8: $\delta_j = R$
9: **end if**
10: $V = N[A_j] + \delta_j$ {Extract $A_j$, increment all values in $A_j$ by $\delta_j$}
11: $N_1 = N[1 : A_{j-1}].V.N[A_{j+1} : A_n]$ {Generate new matrix, keep previous columns and replace $A_j$ by $V$ instead}
12: $P_0$ = Query $M_0$ to obtain new class labels for $N_1$
13: $w_j^+$ = Average number of instances in $P_0$ that have switched classes
14: $V = N[A_j] - \delta_j$ {Decrement all values in column $A_j$ by $\delta_j$}
15: $N_2 = N[1 : A_{j-1}].V.N[A_{j+1} : A_n]$
16: $P_0$ = Query $M_0$ to obtain new class labels for $N_2$
17: $w_j^-$ = Average number of instances in $P_0$ that have switched classes
18: threshold=1
19: **if** $w_j^+ \geq threshold$ **then**
20: $up\_counter = True$
21: **else**
22: $up\_counter = False$
23: **end if**
24: **if** $w_j^- \geq threshold$ **then**
25: $down\_counter = True$
26: **else**
27: $down\_counter = False$
28: **end if**
29: $sum\_score = w_j^+ + w_j^-$
30: **if** $(up\_counter \oplus down\_counter) = True$ **then**
31: $sum\_score$=RotationTest($M_0$,$N$,$A_j$,$R$)
32: **end if**
33: $w_j = (sum\_score)/2$
34: Append $w_j$ to $ScoresArray$
35: **end for**
36: $CR_{score} = \frac{\sum(ScoresArray)}{n}$
37: **return** $CR_{score}$

Fig. 4. The $GetNuggetScoreRevised$ algorithm using the $RotationTest$ method.

results in only a portion of the points moving to the opposite class, ultimately, reducing the $CR_{score}$. Assuming that only half the points in the neighborhood switch classes when decreasing the attribute $Y$, one would have a $CR_{score}$ of $(1 + 1 + 0.5)/4 = 0.625$.

The above analysis reveals that two subsets (i.e., areas A and B in Fig. 3), which in reality should be considered of equal importance when the entire data set is considered may end up with different $CR_{score}$ values. In other words, although area A plays the same role as area B, the two end up with considerably different $CR_{score}$ values (e.g., 0.750 and 0.625, respectively).

The cause of this phenomenon is the relative orientation of the axes system and the two areas (subsets) in general. This was indicated in the analytical steps described earlier in deriving those two values. This can be easily remedied as follows: We rotate one attribute (say axis "$X$" in Fig. 3) with respect to another attribute (axis "$Y$" in Fig. 3). We consider a sequence of rotations by some angle $\theta$ until a complete rotation of 360 degrees is achieved. The summation of weights, $w_j^+$ and $w_j^-$ (1) are recorded at each step. At the end of these iterations, the maximum value of the sum of weights, $w_j^+$ and $w_j^-$ are returned and recorded in memory for each attribute being considered. The final $CR_{score}$ value for a given subset is the average of all the previously recorded $w_j^+$ and $w_j^-$ values, after all the attributes have been considered. When this approach is used on the example depicted in Fig. 3, then both areas A and B are assigned similar $CR_{score}$ values (i.e., the value of 0.750).

The above remedied approach is incorporated as a modification to the $GetNuggetScore$ algorithm. The modified algorithm is called the $GetNuggetScoreRevised$ algorithm. The algorithm's steps are provided in Fig. 4. The modified algorithm includes a call to the $RotationTest$ heuristic, outlined as Fig. 5, which attempts to resolve the above discussed problem of two similar areas receiving different or nonrepresentative scores. In the $GetNuggetScoreRevised$ algorithm, one tests for nonrepresentative results for each

attribute (lines 18-32). Recall from Fig. 3b, that a nonrepresentative result occurred during the following scenario. When the points in the neighborhood were shifted along one direction, the class labels switched for all the points in the neighborhood. However, when the points were shifted in the opposite direction, it resulted in only a partial switching of class labels. A threshold parameter is used to test what percentage of instances in a neighborhood ended up switching class labels when a certain attribute value is increased or decreased. If all the instances end up switching class labels during either increasing or decreasing an attribute's values, then the algorithm $RotationTest$ need not be invoked since the nonrepresentative scores result only during partial switching of class labels. If none of the instances end up switching class labels along both directions, there is no necessity to invoke the $RotationTest$ either. However, if all the instances end up switching class labels along one direction and a partial switching of labels occurs in the opposite direction, then one can use the threshold parameter to decide whether to invoke the $RotationTest$ or not.

If one wishes to minimize the number of calls to this test, then the threshold parameter can be set as high as 1, to short circuit the test and reduce the computation time. However, if the need is to ensure that all critical nuggets are mined out without any conflict, then one can lower the threshold to a value between 0.5 and 1. Setting a threshold of less than 0.5 is not necessary, as that would mean the attribute in question is not switching labels when shifted in both directions. If the condition in line 30 is satisfied (an XOR Boolean operation is used), then the $RotationTest$ is invoked. The key idea in the $RotationTest$ is summarized as follows: for each attribute $A_j$, rotate the values corresponding to $A_j$ by an angle $\theta$ with respect to another attribute $A_k$ ($j \neq$ k). For each of the different angles considered and the different attributes $A_k$, the sum of weights, $w_j^+$ and $w_j^-$ is computed and recorded. After all the angles have been considered, the maximum value among the recorded sum of weights $w_j^+ + w_j^-$ is chosen. If the $RotationTest$ is invoked for each and every attribute, then the combined complexity of algorithms $GetNuggetScoreRevised$ and $RotationTest$ will be $O(n^2)$ (since the number of angles considered is a constant). However, during the experiments with some real-world data sets, the data in some of the data sets were such that the complexity of finding a $CR_{score}$ for each neighborhood was far lower than the worst case theoretical complexity of $O(n^2)$. The methodology for the RotationTest is outlined in Fig. 5.

**Require:** $M_0$: Model, $N$: a neighborhood of data instances, $A_j$: an attribute and $R$: a distance parameter used in creating the neighborhood set, $N$

1: $Array = \phi$
2: **for** each $\theta$ in {10,20, ..., 360} **do**
3: $TempArray = \phi$
4: $\delta_j$ = max($N[A_j]$) - min($N[A_j]$)
5: **if** $\delta_j$=0 **then**
6: $\delta_j = R$
7: **end if**
8: **for** each $k$ in {1,2, ..., $m$} and $k$ != $j$ **do**
9: $\delta_x = \delta_j * cos((\pi/180) \times \theta)$
10: $\delta_y = \delta_j * sin((\pi/180) \times \theta)$
11: $V_j = N[A_j] + \delta_x$
12: $V_k = N[A_k] + \delta_y$
13: **if** $j < k$ **then**
14: $N_1 = N[1 : A_{j-1}].V_j.N[A_{j+1} : A_{k-1}].V_k.N[A_{k+1} : A_n]$
15: **else**
16: $N_1 = N[1 : A_{k-1}].V_k.N[A_{k+1} : A_{j-1}].V_j.N[A_{j+1} : A_n]$
17: **end if**
18: $P_0$ = Query $M_0$ to obtain new class labels for $N_1$
19: $w_j^+$ = Average number of instances in $P_0$ that have switched classes
20: $V_j = N[A_j] - \delta_x$
21: $V_k = N[A_k] - \delta_y$
22: **if** $j < k$ **then**
23: $N_2 = N[1 : A_{j-1}].V_j.N[A_{j+1} : A_{k-1}].V_k.N[A_{k+1} : A_n]$
24: **else**
25: $N_2 = N[1 : A_{k-1}].V_k.N[A_{k+1} : A_{j-1}].V_j.N[A_{j+1} : A_n]$
26: **end if**
27: $P_0$ = Query $M_0$ to obtain new class labels for $N_2$
28: $w_j^-$ = Average number of instances in $P_0$ that have switched classes
29: Append $(w_j^+ + w_j^-)$ to $TempArray$
30: **end for**
31: Append $max(TempArray)$ to $Array$ {i.e., find the max score among the $k$ attributes}
32: **end for**
33: **return** $max(Array)$

Fig. 5. The $RotationTest$ method.

### 3.6 Searching Near the Class Boundary

Using the methodology for finding the $CR_{score}$, our goal is to find critical nuggets in $T_r$. The brute-force method would be to exhaustively examine all possible subsets, calculate their $CR_{score}$ values, and choose the critical nuggets based on the ordering of the $CR_{score}$ values. However, for a large data set, this would be computationally cumbersome due to the combinatorial explosion of the problem. The question then becomes: How can one computationally mine for such small-sized critical nuggets in large data sets?

Since the brute-force method of investigating all possible combinations, would be computationally hard, one can look at candidate sets that have a high likelihood of being critical nuggets. A possible area that can be investigated is near the class boundary that separates the classes of the training set. The basis for this is that points near the boundary are more susceptible to switching of classes. When certain attribute values of boundary points[2] are perturbed, the chances of a boundary point switching to the opposite class are higher than a point deep in the interior.

To validate the idea that boundary points have higher potential of having high $CR_{score}$ values, an experiment was conducted. The 2D randomly generated synthetic data set depicted in Fig. 1b was used for this experiment. Fig. 6 is the result of the experiment. For each point of class "+," the $GetNuggetScoreRevised$ algorithm was used to find a list of scores. The scores were then plotted as a heat map with higher scores being marked as darker gray dots and lower scores being marked as lighter gray dots. The heat map is represented in Fig. 6. It can be observed that there are darker shades along the boundary as compared to the interior. This indicates that the potential of finding critical nuggets is higher along the boundary. Hence, one can focus

2. From this point on, the use of the terms "boundary points" and "points near the boundary" refer to the data instances that lie on or are very close to the class boundary separating the two classes.

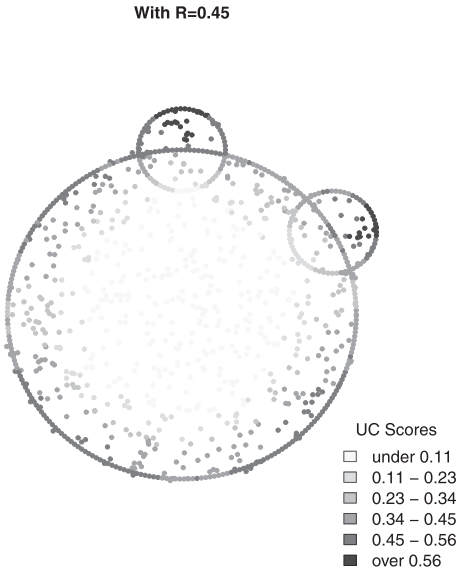



Fig. 6. Boundary points having higher UC scores.

the search along the boundary as compared to the interior. This would greatly reduce the search space as well. Similar experiments were conducted with other 2D sets and similar conclusions were reached.

To find an approximate boundary set from the training data, a boundary detection algorithm is proposed. The algorithm is tested using the 2D randomly generated synthetic data set depicted in Fig. 1b. In [16], the authors proposed a boundary detection algorithm to speed up classifications by Support Vector Machines (SVMs). Though our boundary detection algorithm is similar in spirit to the one in [16], our methodology is simpler as our goal in finding an approximate boundary is merely to reduce the search space in finding the critical nuggets. The proposed algorithm uses euclidean distances to rank the distances between points. The algorithm, $FindBoundary$, is outlined in Fig. 7. The algorithm works in two phases because this study focuses on two-class classification problems. In each phase, a boundary set is isolated for each class in the data set. So for isolating the boundary points for the "+" class,

**Require:** $T_r^+$ and $T_r^-$: The training sets for classes '+' and '-'
1: Initialize array $S = \phi$
2: **for** each point $i$ in $T_r^+$ **do**
3: Calculate distance to all points in $T_r^-$
4: Find the nearest 5 $T_r^-$ points to $i$
5: Using the above top 5 distances, compute an average to get a score $S_i$
6: Store each $S_i$ score in list $S$
7: **end for**
8: Sort all the scores in the list $S$
9: Plot a graph of sorted scores in S.
10: Find cut-off index $i$ where the slope of the graph starts increasing drastically. {Points near the boundary will have smaller scores and as one moves away from the boundary, the scores start increasing drastically.}
11: Points corresponding to scores $S_1$, $S_2$, ..., $S_i$ in list $S$ form the boundary set $B^+$.
12: **return** Boundary Set $B^+$.
13: Repeat algorithm with $T_r^-$ in the role of $T_r^+$ to get Boundary Set $B^-$.

Fig. 7. The $FindBoundary$ algorithm.

**Require:** $T_r$: the training set and $R$: the distance parameter to form the neighborhood set $N$
1: $ScoresArray$=$\phi$
2: Split $T_r$ into $T_r^+$ and $T_r^-$
3: $B^+$ = $FindBoundary(T_r^+)$
4: **for** each $p_0$ in $B^+$ **do**
5: $N = \{x \mid x \in B^+ \wedge |x - p_o| \leq R\}$ {Finding same class points, within a distance $R$ from $p_o$}
6: $CR_{score}$=$GetNuggetScoreRevised(T_r, N, R)$
7: Append $CR_{score}$ to $ScoresArray$
8: **end for**
9: Sort (descending) and rank scores in $ScoresArray$
10: Plot sorted scores in $ScoresArray$ as a histogram and use the histogram to find index $k$ that separates the highest $k$ scores from the rest of the scores.
11: Use $k$ to find top $k$ Critical Nuggets for class '+'.
12: Re-initialize the ScoresArray and repeat steps 2-11 with $B^-$ in the role of $B^+$, to find critical nuggets for class '-'.

Fig. 8. The $FindCriticalNuggets$ algorithm.

the algorithm works by calculating distances to all points in $T_r^-$ from each point in $T_r^+$. For every point in $T_r^+$, five closest $T_r^-$ points are chosen and the average of the five distances is calculated ($S_i$). The score $S_i$, computed for each of the $T_r^+$ points, is sorted and stored in a list $S$. To find the boundary set, one needs a cut off point that would provide the required subset. Using a visualization technique, a graph of the scores in $S$ is plotted. The points that lie closest to the class boundary will have the smallest average distances. A cut-off point is chosen where the slope of the graph increases sharply, indicating a threshold, beyond which, includes interior points with higher $S_i$ scores. This procedure is then carried out again for isolating boundary points for the "$-$" class.

#### 3.6.1 Complexity of the FindBoundary Algorithm

The complexity of the *for-loop* in the $FindBoundary$ algorithm is: $O(|T_r^+| \times |T_r^-|) = O(m^2)$. Sorting takes $O(mlogm)$. Thus the total complexity is $O(m^2) + O(mlogm) = O(m^2)$.

### 3.7 The *FindCriticalNuggets* Algorithm

The $FindCriticalNuggets$ algorithm works in two phases. In each phase it identifies critical nuggets for each one of the two classes. Using the reduced boundary set for each class, the data instances in the boundary set are considered one at a time. Each data instance in the boundary set is considered as a center for a neighborhood. A neighborhood is formed by finding all points that belong to the same class and lie within a distance $R$ from the center point. One class at a time is considered because the goal is to find critical nuggets that belong to one class but switch to the other class when their attribute values are perturbed (a total of two classes is assumed). If there are $|B^+|$ data instances in the boundary set that belong to the same class (say "+"), one can form $|B^+|$ neighborhoods by considering each instance in $B^+$ as a center. For each of the $|B^+|$ neighborhoods, the $CR_{score}$ is computed. The scores are then ranked and the higher scores are used to identify the critical nuggets in $T_r^+$. In the second phase, the other class (say "$-$") is considered. Hence, $|B^-|$ neighborhoods are then considered to compute the $CR_{score}$ values, which in turn are sorted and ranked to identify critical nuggets in $T_r^-$. The algorithm is outlined in Fig. 8.

In summary, the process of finding critical nuggets first involves the identification of an approximate boundary set, and next considering a neighborhood around each boundary point and finding its $CR_{score}$. Identifying an approximate boundary involves complexity of $O(m^2)$, where $m$ is the number of data instances in $T_r$. By using the boundary set and the $CR_{score}$ values different neighborhoods are investigated. There are $|B|$ neighborhoods ($|B| \ll m$) and for each neighborhood the worst complexity (including the rotation test) is $O(dn^2)$ where $d$ is the size of a typical neighborhood. This yields a total complexity for the entire process of identifying critical nuggets of $O(m^2 + t(C) + |B|dn^2)$, which can be further simplified to $O(m^2 + t(C) + n^2)$ (since $|B| \ll m$ and $d \ll n$).

### 3.8 Choosing $R$

In the $FindCriticalNuggets$ algorithm, the distance parameter $R$ is introduced to define a typical neighborhood. Choosing $R$ is an important decision in identifying critical nuggets. Choosing a too small $R$ value may yield single element critical nuggets (sets) while choosing a too large value of $R$ will yield large sized neighborhoods that may not be sensitive to small changes in their attribute values. Also choosing a large value of $R$ can increase the value of $d$, ultimately increasing the complexity of the algorithm. So for the experimental study, a range of $R$ values are considered. The range of $R$ values for our experiments depended on the data set. The general rule used during the study was to vary $R$ in the following range $[0, x]$—where $x$ was a value that caused the maximum size of the neighborhood to not exceed 20 percent of the size of the data set. The main intuition for setting this was to limit the size of the neighborhoods, as large neighborhoods include points that are located away from the class boundary and, hence, are less sensitive to small changes in attribute values. Larger neighborhoods also have lower $CR_{score}$ values and are not useful in the mining of critical nuggets.

To find the critical nuggets among neighborhoods formed by different values of $R$, the following analysis was conducted. Using different values of $R$, the

$$FindCriticalNuggets$$

algorithm was used to find the $CR_{score}$ values for each neighborhood (each neighborhood was formed using every element in the boundary set as a center). For each value of $R$, the top $k$ neighborhoods are identified based on their scores (The top $k$ subset of neighborhoods were identified by first sorting the scores in descending order. Visualization methods such as plotting the sorted scores as a histogram can be used to identify an appropriate cut-off value of $k$. $k$ is chosen by selecting an appropriate point in the histogram that delineates the small group of high scores from the majority of small scores). Therefore, if there are $r$ values of $R$, one would have $k \times r$ neighborhoods. Some of the $k \times r$ neighborhoods could have been formed around the same center. Hence, among the $k \times r$ neighborhoods, the scores are ranked based on unique centers. The top $k$ scores among the unique centers and their associated $R$ values are used to identify the top $k$ critical nuggets.

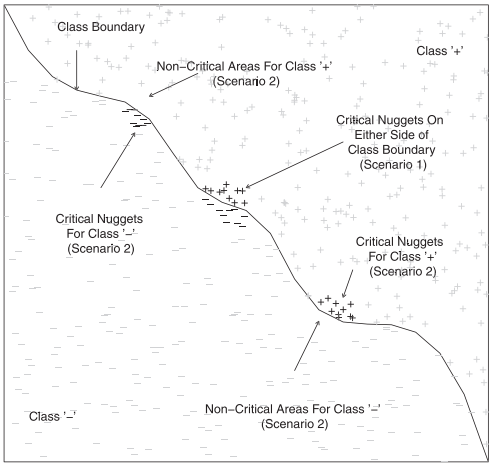



Fig. 9. Illustration for duality.

### 3.9 Duality

In the $FindCriticalNuggets$ algorithm, recall that the search for critical nuggets was on a class-by-class basis on two-class problems. Hence, for each class label, one obtains a set of critical nuggets. Since critical nuggets were identified for each of the class labels, a study was conducted to see if there were any relationships between the two sets of critical nuggets. A property that was investigated was duality—relationships between critical nuggets of the two classes. In Fig. 9, some critical nuggets for two classes are illustrated using darker shades of gray. The experiments mainly checked to see if the following scenarios occurred:

1. Scenario 1—Do critical nuggets belonging to different classes lie in "proximity" to one another? In other words, a check was done to see if the critical nuggets of different class labels lie in "proximity" but on opposite sides of the class boundary (see Fig. 9 for Scenario 1).
2. Scenario 2—Are there any sets (neighborhoods) that are noncritical for one class label, but lie in "proximity" to a critical nugget belonging to another class and vice versa (see also Fig. 9 for Scenario 2)?

Both scenarios have good potential as they help in broadly dividing the data set into three regions. These regions are summarized as follows (assuming two classes "+" and "−"):

1. Region 1—This is comprised of critical nuggets of one class that lie in close proximity to critical nuggets of the opposite class (i.e., subsets of the data set that have high $CR_{score}$ values and lie in close proximity, but on opposite sides of the class boundary) and vice versa. This is depicted as Scenario 1 in Fig. 9.
2. Region 2—Critical nuggets of one class that lie in close proximity to neighborhoods of the opposite class, neighborhoods that are not critical nuggets (and vice versa). This is depicted as Scenario 2 in Fig. 9.
3. Region 3—Neighborhoods (or subsets of the training data) that lie in the interior (not near the class boundary) of either class (characterized by very low $CR_{score}$ values).

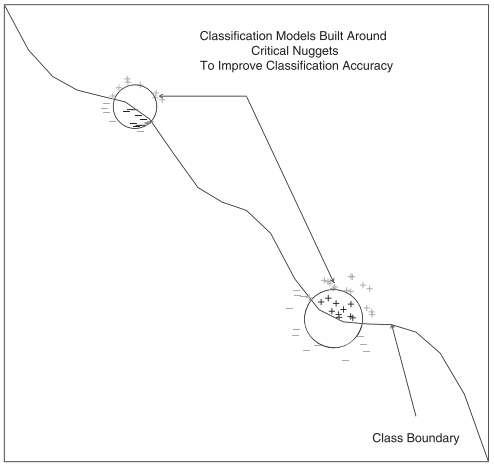



Fig. 10. Postprocessing using critical nuggets for improving accuracy.

The empirical results for this study indicate that if there are a group of nuggets from one class in close proximity with each other, then it is likely to have a corresponding group of nuggets for the other class on the other side of the boundary. In other words, a group of nuggets for one class indicates a region of the data set of potential high interest for both classes. Decomposing a data set into the three regions described as above has the potential to offer useful insights of the data. Such insights may assist the analyst to better understand the phenomenon or systems related to the data. Clearly, this is domain dependent.

### 3.10 Improving Classification Accuracy

Classification algorithms are usually judged based on the accuracy of their predictions. If the predictions include a minimum number of false positives and false negatives, the accuracy of an algorithm is rated as high. During the experimental stage with various data sets, tests were conducted to see if critical nuggets could help improve the classification accuracy. The identified nuggets were used in deriving additional small scale classification models. For each class, an additional classification model is built/trained by first deriving a new data set, which is a subset of the original training data set. The new data set was derived by relabeling a subset of the original data records into two new classes as follows:

- Data records that belong to the top $k$ (for finding $k$, see lines 9-11 in Fig. 8) critical nuggets (of say, the "+" class) become a part of one class.
- Data records that are near the top $k$ "+" class critical nuggets but NOT belonging to the set of "+" class critical nuggets are labeled as another class (this may include instances that belong to both "+" and "−" classes).

An additional model is built similarly using the critical nuggets from the other class (say, "−"). This is illustrated in Fig. 10. In this figure, the right region indicates critical nuggets belonging to the "+" class (labeled as dark shaded "+" symbols ) and surrounded by some neighboring points belonging to both "+" and "−" classes. The region on the top left illustrates critical nuggets belonging to the "−" class (labeled as dark shaded "−" symbols) and surrounded by both "+" and "−" neighbors. Using this newly derived data set, a classification model was setup. Since very few subsets qualify as critical nuggets, the newly relabelled data set has a skewed distribution of classes. There are very few data instances that belong to the first class of critical nuggets. At the same time, there is a disproportionately large number of noncritical neighbors. This skewed distribution can be remedied using a cost sensitive classifier [17]. One can model a cost sensitive learner that assigns higher costs for misclassifying the class that is less represented when compared to misclassifying objects of the more represented class. In our case, it is more important to identify a critical nugget correctly. So the cost classification model is derived by assigning a higher cost for not identifying a critical nugget correctly. Using the newly trained classification models built around critical nuggets, one can use it in tandem with the original classification model to predict the class labels of data records. According to the experiments described next, it turns out that this method of postprocessing of classified data records using the information gained from the critical nuggets helps in improving the classification accuracy in data sets. In summary, the steps in improving classification accuracy are outlined as follows:

1. Using a standard classification algorithm $C$, derive a classification model $M_0$.
2. For the first class (such as the "+" class in Fig. 10), build a data set comprising of two new classes:
   a. One class is comprised of only the top $k$ critical nuggets (e.g., dark shaded "+" points within the right circle in Fig. 10). Label this class as "I".
   b. The other class is comprised of noncritical records and it is derived by using the following principle: for each member in the set of critical nuggets (class "I" in the previous step), choose the closest neighbors that are not part of class "I" (e.g., points just outside the right circle in Fig. 10). Label these data instances as class "O".
3. Using a nearest neighbor classifier, build a cost-sensitive classification model, assigning higher costs for misclassifying a record belonging to critical nuggets. The derived classification model is denoted as $M^{+}_{nuggets}$.
4. Repeat steps 2-3 for the second class (such as the "−" class in Fig. 10) and the derived model is denoted as $M^{-}_{nuggets}$.

For a given test data instance or a new unlabeled data instance, the derived critical nuggets models ($M^{+}_{nuggets}$ and $M^{-}_{nuggets}$) are used along with the standard classification model $M_0$. We assign a class label guided by testing against the following set of rules:

- If $M^{+}_{nuggets}$ assigns a class label of "I" and $M^{-}_{nuggets}$ assigns a class label of "O", then the data instance is assigned a label of "+".
- If $M^{-}_{nuggets}$ assigns a class label of "I" and $M^{+}_{nuggets}$ assigns a class label of "O", then the data instance is assigned a label of "−".
- If $M^{+}_{nuggets}$ and $M^{+}_{nuggets}$ assign a class label of "O", then the data instance is assigned a class label based

on the class assigned by model $M_0$ obtained from the standard classification algorithm.

- If $M^{+}_{nuggets}$ and $M^{-}_{nuggets}$ assign a class label of "I," then the data instance is assigned a class label based on the class assigned by $M_0$.

## 4 An Experimental Study

Some experiments were conducted on 11 multidimensional real-world data sets from the UCI machine learning repository [18] and two 2D geographical synthetic data sets. The software was written in R [19] and Python [20] and utilized the data mining library Weka [21]. The algorithm $FindCriticalNuggets$ was applied to each of the data sets. The next two sections provide detailed summaries of the experimental analysis on one of the 2D synthetic data sets and one of the real-world data sets. These summaries include details such as some results of the $FindCriticalNuggets$ algorithm, validation of properties such as duality, and the improvements in classification accuracy using critical nuggets. A similar analysis has been conducted for 10 other real-world data sets, but for the sake of space, detailed explanations have been provided only for one of the real-world data sets. However, the classification accuracy improvements for all 11 real-world data sets and two 2D synthetic data sets have been provided in the last section.

All the data sets used in the experiments were normalized (values for each attribute in a data set lie within $[0, 1]$) using normalization routines available in the Weka library. Euclidean distance measures were used in computing distances. For all the data sets, data instances were normalized and 10 runs of 10-fold cross validations were performed. The following classification algorithms from Weka were used for this study: J48 (Weka's software implementation of the C4.5 [22] algorithm), IBk (Weka's implementation of K-nearest neighbor classifier [23], LMT (Weka's implementation of Logistic Model Trees [24]), and NaiveBayes [25]. Default options in Weka were used for the algorithms J48, LMT, and NaiveBayes and for the IBk algorithm, the nearest neighbor parameter of K was set to 5. Table 1 provides a description of the data sets used during the experimental study.

For the experiments, decisions had to be made with regard to the choice of a cost-sensitive classifier to handle the skewed data distribution (when building small classification models around critical nuggets) and the assignment of costs for the cost-sensitive classifier. For the improvement of accuracies using critical nuggets, two cost-sensitive algorithms (CostSensitiveClassifier [26] and MetaCost [17]) from Weka were considered to tackle the imbalanced data distribution. The algorithms, MetaCost and CostSensitiveClassifier, use a base classifier to instantiate the training process (the base classifier can be any standard classification algorithm). For the base classifier in MetaCost, the four classification methods used in the study (J48, LMT, IBk, and NaiveBayes) were tested against the various data sets to see which classification method yielded the best accuracy. IBk, as a base classifier, consistently performed better than the others because the small models created around the critical nuggets were suited for neighborhood-based classification techniques. Hence, IBk was chosen as the base classifier for MetaCost.

TABLE 1
Description of Data Sets Used

| Data Set | Number of Instances | Number of Attributes | Class (Distribution) |
|---|---|---|---|
| Synthetic Geographical (Georgia, USA) | 10,387 | 2 | + (2,649), - (7,738) |
| Synthetic Geographical (Idaho, USA) | 10,233 | 2 | + (2,963), - (7,270) |
| Wisconsin Breast Cancer (WDBC) | 569 | 30 | Benign (357), Malignant (212) |
| SPECT Heart | 267 | 42 | Normal (212), Abnormal (55) |
| Spambase | 4,601 | 57 | Spam(1,813), Not Spam(2,788) |
| German Credit Data | 1,000 | 24 | Good(700), Bad(300) |
| Pima Indian Diabetes | 768 | 8 | Positive(268), Negative(500) |
| Sonar | 208 | 60 | R(97), S(111) |
| Ionosphere | 351 | 34 | good(225), bad(126) |
| Cardiotocography2 [a] | 1,950 | 22 | Normal(1655), Suspect(295) |
| Liver Disorders | 345 | 6 | A(145), B(200) |
| Parkinsons | 195 | 22 | H(48),P(147) |
| Glass2 [b] | 163 | 9 | Y(87), N(76) |

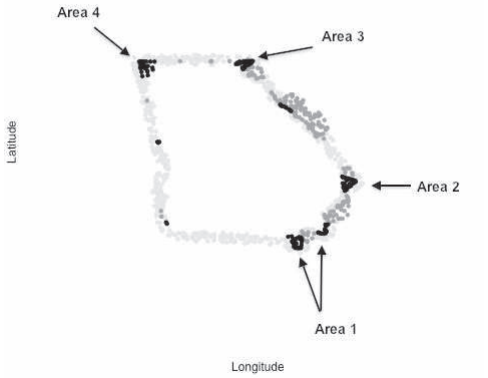
*a. A variant of the Cardiotocography data set of UCI repository, formed by considering data records belonging to only two out of three class attributes and ignoring the third class attribute called "Pathologic."*
*b. A variant of the Glass identification data set of UCI repository, formed by combining data records belonging to class attributes 1 and 3 and renaming the class attribute as class "Y," combining records having class attributes 2 and 4 and renaming as class "N" and excluding records belonging to class attributes 5, 6, and 7. This variant has been used in prior work such as [24].*

MetaCost in combination with IBk consistently yielded better accuracies for different data sets when compared to the CostSensitiveClassifier method. So MetaCost in combination with IBk was chosen as the cost-sensitive classifier for the experiments. Cost ratios for cost-sensitive classifiers are ideally provided by domain experts. However, in this case, three different cost ratios 2:1, 5:1, and 10:1 were initially considered. The ratio of 5:1 (a cost of 5 was allocated toward misclassification of a critical data instance as compared to a cost of 1 allocated toward misclassification of a noncritical data instance) provided higher improvements in accuracy as compared to the ratio of 2:1. Increasing it to 10:1 did not provide any significant improvement when compared to accuracy improvements with the 5:1 ratio. The main goal among all the data sets was to correctly classify records that belong to critical nuggets. By assigning a higher cost (5) as a penalty for misclassification of a critical data instance (as compared to equal cost penalties) was sufficient to meet the goal of improving classification accuracies for different data sets. Hence, 5:1 was used as a cost ratio for the study.

### 4.1 Analyzing the Synthetic Geographical Data Set

This training data set is a synthetic data set that conforms to the political boundary of the State of Georgia, USA. This is a


(a) Boundary Approximation: Black dots indicate '+' points and grey dots indicate '-' points.




(b) Critical Nuggets: Black dots indicate '+' points that are critical nuggets. Dark grey dots indicate '-' points that are critical nuggets. Light grey dots indicate '+' and '-' points that did NOT emerge as critical nuggets.

Fig. 11. Analyzing the synthetic geographical data set.

large data set that contains 10,387 observations comprising of two classes: "+" (2,649 instances) and "−" (7,738 instances). In other words, points inside the State of Georgia are defined as "+" points, while the ones outside the political boundary are defined as "−" points. A boundary set was approximated to 500 instances of class "+" and 500 instances of class "−". This 2D data set has an advantage as it provides visual validation to the results. Fig. 11a provides an approximation of the boundary set for the state of Georgia. Fig. 11b depicts the results of the $FindCriticalNuggets$ algorithm on the data set. Fig. 11b is similar to Fig. 11a, except that some of the critical nuggets for both classes have been superimposed. Notice that in Fig. 11b the black dots indicate "+" data instances that have been identified as critical nuggets and having high $CR_{score}$ values. Also notice that these black dots line up along areas that have visually interesting features such as sharp bends and curves. Similarly, the dark gray dots indicate "−" data instances that have been identified as critical nuggets having high $CR_{score}$ values.

*Duality* in this data set can be explained through the visual features. Areas 1, 2, 3, and 4 in Fig. 11b are visually interesting features. One can observe that critical nuggets for both classes have lined up near these visually interesting features. Also observe that critical nuggets for one class (black dots) tend to line up near the critical nuggets of the other class (dark gray dots). This finding is in line with the Scenario 1 illustration depicted in Fig. 9 and discussed in the earlier section on duality.

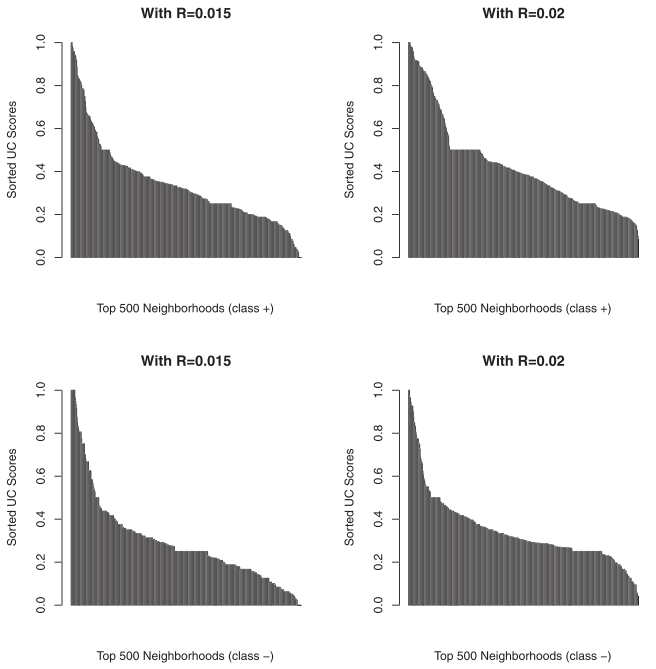



Fig. 12. The Geographical data set—Histograms of sorted scores for two different $R$ values.

*Histograms* of $CR_{score}$ values for two different $R$ values for each of the two classes are depicted as Fig. 12. Among the 500 different neighborhoods investigated, it can be observed that there are indeed very few sets with high $CR_{score}$ values, say greater than 0.75. This indicates the potential value of finding the critical nuggets in large data sets.

*Classification accuracy improvements* for this data set are outlined in rows 1-4 of Table 2 (to be described later in more detail). One can observe that the relative improvement in accuracy when using J48 as a classifier is 11.46 percent.

### 4.2 Analyzing the Wisconsin Breast Cancer (WDBC) Data Set

This data set has 569 data instances (357 Benign and 212 Malignant), 32 attributes (30 attributes when the record locator and class labels are skipped), and two types of class labels (Benign and Malignant). Using the $FindBoundary$ algorithm, an approximate boundary set comprising of 150 Benign and 150 Malignant data instances was selected. The main task was to apply the $FindCriticalNuggets$ algorithm to identify critical nuggets. The standard normalization function available in the Weka library was used to normalize this data set. For different values of $R$ and for a given class, the $FindCriticalNuggets$ algorithm was run. For this analysis, five different $R$ values were used: $\{0.20, 0.25, 0.30, 0.35, 0.40\}$. Increasing the range of $R$ values beyond 0.40 increased the maximum size of the neighborhood to exceed 20 percent of the size of the data set (see Section 3.8). The neighborhoods that had the top 20 scores were identified for each value of $R$, as the top 20 high scores

TABLE 2
Improvements in Accuracy Using Critical Nuggets

| Data set | Classifier | $a_0$ (Accuracy before Critical Nuggets) (%) | $a_1$ (Accuracy With Critical Nuggets) (%) | Accuracy Increase $(a_1 - a_0)$ (%) | Relative Improvement $(\frac{a_1 - a_0}{100 - a_0})$ (%) |
|---|---|---|---|---|---|
| Synthetic Geographical (Georgia, USA) | J48 | 98.43 | 98.61 | 0.18 | 11.46 |
| | LMT | 99.49 | 99.91 | 0.42 | 10.19 |
| | NaiveBayes | 82.78 | 83.73 | 0.95 | 5.52 |
| | IBk | 98.29 | 98.51 | 0.22 | 12.87 |
| Synthetic Geographical (Idaho, USA) | J48 | 98.83 | 98.93 | 0.10 | 8.55 |
| | LMT | 98.99 | 99.10 | 0.11 | 10.89 |
| | NaiveBayes | 80.02 | 81.33 | 1.31 | 6.56 |
| | IBk | 98.14 | 98.32 | 0.18 | 9.68 |
| Wisconsin Breast Cancer (WDBC) | J48 | 94.22 | 98.05 | 3.83 | 66.26 |
| | LMT | 97.47 | 99.12 | 1.65 | 65.22 |
| | NaiveBayes | 93.34 | 96.00 | 2.66 | 39.94 |
| | IBk | 95.36 | 97.68 | 2.32 | 50.00 |
| SPECT Heart | J48 | 73.22 | 93.93 | 20.71 | 77.33 |
| | LMT | 79.03 | 93.67 | 14.64 | 69.81 |
| | NaiveBayes | 68.13 | 89.36 | 21.23 | 66.61 |
| | IBk | 69.89 | 89.06 | 19.17 | 63.67 |
| Spambase | J48 | 92.67 | 96.28 | 3.61 | 49.25 |
| | LMT | 93.33 | 96.60 | 3.27 | 49.03 |
| | NaiveBayes | 79.62 | 89.38 | 9.76 | 47.89 |
| | IBk | 90.85 | 95.41 | 4.56 | 49.84 |
| German Credit | J48 | 73.13 | 79.76 | 6.63 | 24.67 |
| | LMT | 76.96 | 81.26 | 4.30 | 18.66 |
| | NaiveBayes | 75.42 | 82.50 | 7.08 | 28.80 |
| | IBk | 67.19 | 73.18 | 6.00 | 18.26 |
| Pima Indian Diabetes | J48 | 75.00 | 84.69 | 9.70 | 38.78 |
| | LMT | 76.80 | 87.29 | 10.49 | 45.22 |
| | NaiveBayes | 75.74 | 86.44 | 10.70 | 44.11 |
| | IBk | 70.22 | 84.09 | 13.87 | 46.57 |
| Sonar | J48 | 73.61 | 89.71 | 16.10 | 61.00 |
| | LMT | 77.07 | 90.05 | 12.98 | 56.61 |
| | NaiveBayes | 67.74 | 89.38 | 21.64 | 67.08 |
| | IBk | 86.44 | 95.43 | 8.99 | 66.30 |
| Ionosphere | J48 | 89.68 | 98.40 | 8.72 | 84.50 |
| | LMT | 91.96 | 98.03 | 6.07 | 75.50 |
| | NaiveBayes | 82.65 | 94.73 | 12.08 | 69.63 |
| | IBk | 86.63 | 91.62 | 4.99 | 37.32 |
| Cardiotocography2 | J48 | 93.73 | 94.50 | 0.77 | 12.28 |
| | LMT | 94.66 | 95.61 | 0.95 | 17.79 |
| | NaiveBayes | 86.39 | 89.43 | 3.04 | 22.34 |
| | IBk | 92.72 | 93.68 | 0.96 | 13.19 |
| Liver Disorders | J48 | 65.68 | 85.19 | 19.51 | 56.85 |
| | LMT | 69.71 | 85.22 | 15.51 | 51.20 |
| | NaiveBayes | 55.94 | 77.36 | 21.42 | 48.62 |
| | IBk | 63.01 | 82.14 | 19.13 | 51.72 |
| Parkinsons | J48 | 83.79 | 95.90 | 12.11 | 74.71 |
| | LMT | 84.56 | 97.08 | 12.52 | 81.09 |
| | NaiveBayes | 69.38 | 94.36 | 24.98 | 81.58 |
| | IBk | 95.79 | 98.67 | 2.88 | 68.41 |
| Glass2 | J48 | 79.14 | 96.32 | 17.18 | 82.36 |
| | LMT | 77.48 | 96.07 | 18.59 | 82.55 |
| | NaiveBayes | 61.96 | 94.36 | 32.40 | 85.17 |
| | IBk | 77.42 | 96.75 | 19.33 | 85.61 |

clearly separated the rest of the low scores (see Fig. 13). Among the 100 ($= 20 \times 5$) neighborhoods, the top 20 neighborhoods (each formed around unique centers) were selected based on their scores and these neighborhoods formed the critical nuggets. The above steps of finding the top 20 critical nuggets was done for the other class as well.

*Histograms* of the $CR_{score}$ values for two different values of $R$ are outlined as Fig. 13. The histograms reveal that only a very small number of neighborhoods qualify as critical nuggets among the 150 different neighborhoods surveyed

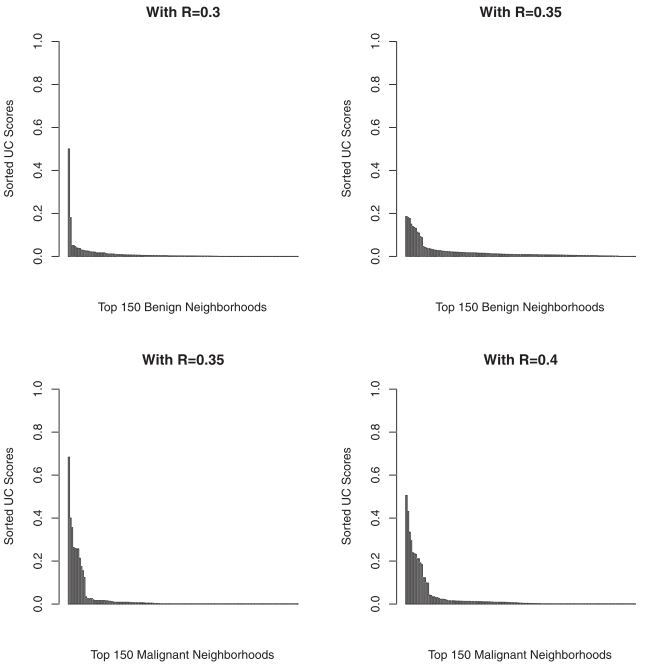



Fig. 13. The WDBC data set—Histograms of sorted scores for two different $R$ values.

for each value of $R$. This underscores the potential importance of finding such sets.

The *duality* properties of critical nuggets were verified for this data set. It was found that critical nuggets belonging to one class tend to be close to critical nuggets belonging to the opposite class (since one cannot visualize this data set, euclidean distances were used to estimate how close records were to each other). This finding was in line with Scenario 1 depicted in Fig. 9.

Results on *improving classification accuracy* for this data set are outlined in rows 5-8 of Table 2. Though the increase in accuracy may seem marginal (3.83 percent), the relative improvement in accuracy was high (66.26 percent).

Above all, the methodology for finding critical nuggets has isolated benign records, whose attribute values when slightly perturbed, end up switching to the malignant class. This is valuable information as it can help identify small sets of benign records that are susceptible to easily switching over to the malignant class.

### 4.3 Summary of Classification Accuracy Improvements

In Table 2, a summary of the classification accuracy improvements are provided. Eleven real-world data sets from the UCI Repository [18] and two synthetic data sets were used. The column labeled as $a_0$ provides the accuracy of the chosen classification algorithm on a given data set, without the knowledge of critical nuggets. The next column labeled as $a_1$ indicates the improvements in accuracy of classification results as a result of knowledge gained through identification of critical nuggets.

To understand the values in the last column of Table 2 (labeled "Relative Improvement"), consider any row of the table. Suppose we consider the results for the SPECT Heart data set using J48 as the classification method. Without the use of the nuggets that accuracy was equal to 73.22 percent.

TABLE 3
Significance of Improvements

| Comparison | Positive Ranks | Negative Ranks | p-value |
|---|---|---|---|
| J48 vs. J48+Critical Nuggets | 0 | 91 | $< 0.01$ |
| LMT vs. LMT+Critical Nuggets | 0 | 91 | $< 0.01$ |
| NaiveBayes vs. NaiveBayes+Critical Nuggets | 0 | 91 | $< 0.01$ |
| IBk vs. IBk+Critical Nuggets | 0 | 91 | $< 0.01$ |

With the use of critical nuggets the accuracy increased to 93.93 percent. The increase in accuracy is equal to $20.71\% = (93.93\% - 73.22\%)$. However, the maximum possible improvement would be equal to $100 - 73.22 = 26.78\%$. The previous increase of 20.71 percent represents $77.33\%(= \frac{20.71}{26.78})$ of the maximum possible improvement. The rest of the results in that column in Table 2 have been computed in a similar manner. The exceptionally high values in the last column of Table 2 indicate the high potential critical nuggets may offer in improving classification accuracy.

Using the results of Table 2, the Wilcoxon [27] test was used to test the statistical significance of accuracy improvements using critical nuggets as compared to using a standard classification algorithm (without the knowledge of critical nuggets). These results are summarized in Table 3. Notice that at 99 percent confidence level, the accuracy improvements using critical nuggets are statistically significant (p-values being less than 0.01) when compared to results obtained by using only the standard classification algorithms (J48, LMT, NaiveBayes, and IBk).

## 5 Conclusions and Future Research

This paper presents the notion of critical nuggets. A new metric, the $CR_{score}$, was introduced for measuring criticality of a subset or nugget. A simple rotation test was proposed to resolve conflicting scores when they occur. The proposed score was used to identify critical nuggets. The tests on a number of 2D synthetic data sets provided a visual validation that such nuggets are more likely to lie near class boundaries and in close proximity to the complex features along the class boundaries. Reducing the search to near the class boundaries saves computation time in identifying such nuggets. The $FindCriticalNuggets$ algorithm was outlined that used the boundary estimation method and the $CR_{score}$ to identify critical nuggets. Some important properties such as the dual nature of critical nuggets were discussed and the properties were validated through some sets of experiments. The proposed ideas were tested on some multidimensional real-world data sets. Results from the experiments on the real-world data sets revealed that only a very small number of subsets qualified as critical nuggets. Experimental results from the real-world data sets also indicated the importance of finding such subsets in large databases. The knowledge of critical nuggets also helped to reduce the number of false positives and false negatives and, thus, significantly improving the overall accuracy of classification tasks. Future work can be done on improving the $O(n^2)$ complexity of the boundary approximation algorithm. Work can also be done in extending these ideas to data sets with multiple classes (greater than 2) and data sets with mixed attributes. The postprocessing methodology of improving classification accuracy proposed in this work can also be compared with other techniques (including resampling techniques and other cost-sensitive classification methods) in the field of classification algorithms.

## References


[1] A. Koufakou and M. Georgiopoulos, "A Fast Outlier Detection Strategy for Distributed High-Dimensional Data Sets with Mixed Attributes," *Data Mining and Knowledge Discovery*, vol. 20, no. 2, special issue SI, pp. 259-289, Mar. 2010.

[2] R.A. Weekley, R.K. Goodrich, and L.B. Cornman, "An Algorithm for Classification and Outlier Detection of Time-Series Data," *J. Atmospheric and Oceanic Technology*, vol. 27, no. 1, pp. 94-107, Jan. 2010.

[3] M. Ye, X. Li, and M.E. Orlowska, "Projected Outlier Detection in High-Dimensional Mixed-Attributes Data Set," *Expert Systems with Applications*, vol. 36, no. 3, pp. 7104-7113, Apr. 2009.

[4] K. McGarry, "A Survey of Interestingness Measures for Knowledge Discovery," *Knowledge Eng. Rev.*, vol. 20, no. 1, pp. 39-61, 2005.

[5] L. Geng and H.J. Hamilton, "Interestingness Measures for Data Mining: A Survey," *ACM Computing Surveys*, vol. 38, article 9, http://doi.acm.org/10.1145/1132960.1132963, Sept. 2006.

[6] E. Triantaphyllou, *Data Mining and Knowledge Discovery via Logic-Based Methods*. Springer, 2010.

[7] E.M. Knorr, R.T. Ng, and V. Tucakov, "Distance-Based Outliers: Algorithms and Applications," *VLDB J.*, vol. 8, no. 3/4, pp. 237-253, 2000.

[8] D. Hawkins, *Identification of Outliers (Monographs on Statistics and Applied Probability)*. Springer, http://www.worldcat.org/isbn/041221900X, 1980.

[9] F. Angiulli and C. Pizzuti, "Outlier Mining in Large High-Dimensional Data Sets," *IEEE Trans. Knowledge Data Eng.*, vol. 17, no. 2, pp. 203-215, Feb. 2005.

[10] Y. Tao, X. Xiao, and S. Zhou, "Mining Distance-Based Outliers from Large Databases in Any Metric Space," *Proc. 12th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD)*, T. Eliassi-Rad, L.H. Ungar, M. Craven, and D. Gunopulos, eds., pp. 394-403, 2006.

[11] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Computing Survey*, vol. 41, no. 3, article 15, 2009.

[12] S.D. Bay and M. Schwabacher, "Mining Distance-Based Outliers in Near Linear Time with Randomization and a Simple Pruning Rule," *Proc. Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD)*, L. Getoor, T.E. Senator, P. Domingos, and C. Faloutsos, eds., pp. 29-38, 2003.

[13] A. Ghoting, S. Parthasarathy, and M.E. Otey, "Fast Mining of Distance-Based Outliers in High-Dimensional Datasets," *Data Mining and Knowledge Discovery*, vol. 16, no. 3, pp. 349-364, 2008.

[14] M.M. Breunig, H.-P. Kriegel, R.T. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers," *SIGMOD Record*, vol. 29, no. 2, pp. 93-104, 2000.

[15] L. Duan, L. Xu, Y. Liu, and J. Lee, "Cluster-Based Outlier Detection," *Annals of Operations Research*, vol. 168, no. 1, pp. 151-168, http://dx.doi.org/10.1007/s10479-008-0371-9, Apr. 2009.

[16] N. Panda, E.Y. Chang, and G. Wu, "Concept Boundary Detection for Speeding Up SVMs," *Proc. 23rd Int'l Conf. Machine Learning (ICML)*, W.W. Cohen and A. Moore eds., vol. 148, pp. 681-688, 2006.

[17] P. Domingos, "Metacost: A General Method for Making Classifiers Cost-Sensitive," *Proc. Fifth Int'l Conf. Knowledge Discovery and Data Mining*, pp. 155-164, 1999.

[18] A. Frank and A. Asuncion, "UCI Machine Learning Repository," http://archive.ics.uci.edu/ml, 2010.

[19] R Development Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, http://www.R-project.org, 2010.

[20] G. van Rossum et al., *Python: An Object Oriented Programming Language*, http://www.python.org, 1991.

[21] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explorations,* vol. 11, no. 1, pp. 10-18, http://doi.acm.org/10.1145/1656274.1656278,, 2009.
[22] J.R. Quinlan, *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.
[23] D. Aha and D. Kibler, "Instance-Based Learning Algorithms," *Machine Learning,* vol. 6, pp. 37-66, 1991.
[24] N. Landwehr, M. Hall, and E. Frank, "Logistic Model Trees," *Machine Learning,* vol. 95, no. 1/2, pp. 161-205, 2005.
[25] G.H. John and P. Langley, "Estimating Continuous Distributions in Bayesian Classifiers," *Proc. 11th Conf. Uncertainty in Artificial Intelligence,* pp. 338-345, 1995.
[26] B. Zadrozny, J. Langford, and N. Abe, "Cost-Sensitive Learning by Cost-Proportionate Example Weighting," *Proc. IEEE Third Int'l Conf. Data Mining (ICDM),* pp. 435-442, 2003.
[27] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics Bull.,* vol. 1, no. 6, pp. 80-83, http://dx.doi.org/10.2307/3001968, 1945.

**David Sathiaraj** received two MS degrees in industrial engineering and computer (systems) science from Louisiana State University (LSU). He will receive the PhD degree in computer science from LSU in May 2013. He has also been involved in software systems development in his role as the IT Manager for the NOAA Southern Regional Climate Center. He has been an active developer of the Applied Climate Information Systems (ACIS) project—a distributed computing system that collects, analyzes, and delivers climate data and products. His research interests include data mining, spatiotemporal analysis and visualization, distributed computing, and geographic information systems.

**Evangelos Triantaphyllou** received the dual MS degrees in environment and operations research, the MS degree in computer science, and the dual PhD degree in industrial engineering and operations research, all from Penn State University from 1984 to 1990. He is currently a professor in the Computer Science Department at Louisiana State University. His research focuses on decision-making theory and applications, data mining and knowledge discovery, and the interface of operations research and computer science.