Exploiting Regions of Influence to Visualize Class Boundaries

Pallav Tinna and Kamalakar Karlapalem

Abstract

Interactive visualization and analysis of the class boundaries is important because it tells us how and why the classes differ. However, the problem of modeling the boundary of classes of arbitrary size, shape and density is challenging. The boundary of a class should not be limited to the points/shape which encloses the points within the class but it should be, the points/shape which encloses the region of influence of a class. The "region of influence" refers to the space around the class where any point lying within the region is likely to be classified to the class based on a nearest neighbor classifier. We have developed interactive boundary visualization toolkit for classified datasets which provides insights about the classifier model used on the dataset. Our algorithm first generates a candidate boundary set for each class based on reverse k-nearest neighbors approach and extends this boundary iteratively through the region of influence of the class. Further, we present these boundary points enclosing the region of influence as a linear approximated shape using triangulation techniques. We show experimental results on 2D and 3D datasets.

1 Introduction

Boundary detection techniques play an important role in post-classification analysis. Class boundaries contain overlapping patterns of different classes and there is need to visually present these boundaries to reveal the type and extent of overlaps that occur between classes. Boundary points exhibit a subset of patterns that straddle the two classes and provide interesting characteristics about the inter-class relationships. Visualizing the class boundaries is relevant in learning how and why these classes differ and to have an insight of the knowledge used by the classifier in separating them. Visualization techniques [12] are an intuitive way to study class characteristics in the data by revealing class boundaries, exposing inter-class gaps and providing visual cues to analyze classification results. They also allow developers of the classification model to be more engaged in the reasoning about the classification model via interaction techniques.

Algorithms like BRIM [1], BORDER [24], Concave-Hull [17] provide set of points that are located at the margin of densely distributed data such as a class but do not consider the inter-class relationships in boundary detection. Techniques like SVM [22] outline the broad separation between the classes but it is difficult to visualize where and how the overlap of class properties that exist between two classes. Visual interpretation of SVM results [3] do not reveal factors or data points that are relevant in separation between two classes. Melnik [15] suggested the use of connectivity of decision regions in extracting qualitative information about the classification model where the connectivity is obtained by emphasizing on the regions where changes in the class label occurs. [25] proposed DBPS algorithm to retrieve data points on the decision boundary of the classes to get insight about the classifier and SOMDBV algorithm to visualize those boundaries in a 2D SOM map [13]. The boundary retrieved from DBPS algorithm contains only a set of points with no notion of shape represented by them. These algorithms do not assist in the retrieval of regions with different amounts of connectivity between different classes.

The region between the classes are likely to have influence and there is need to visually separate these regions containing different amount of influence from these classes. Therefore, our work is an extension of the idea of detecting the shape of class boundaries and followed by stretching these boundaries through the region of influence based on kNN information. This representation of class boundaries aids the ability to simultaneously visualize the separate and overlap of regions of influence of the different classes in the labeled dataset. Representation of overlapping classes boundaries using the triangulated objects helps to retrieve the regions in the space where inter-class relationships occur. The difference between the shape of the class boundaries proposed in above stated methods [1, 24, 17] and ours is that we incorporate the information provided by a kNN classifier in generation of class boundaries so as to visualize the direction and amount of spread of the class properties surrounding the class points. Unlike SVM, our approach stands on nearest-neighbor based approach and considers all the minimal decision criteria characterized by the neighborhood support set. Class boundaries based on region of influence presented in our work are not tight fitting class boundaries which makes them even more important when classes are close enough or overlapping.

In this paper, we extend the traditional definition of class boundary to "region of influence" which means any point within the region of influence of the class will be affected by the class properties as depicted by nearest neighbor (NN) classifier. The extent of the region of influence of a class is computed using the decision boundary modeled by NN classifier. The reasons of using NN classifier for the computation of regions of influence are : (i) it keeps the decision boundary closer to the class points. (ii) it is independent of actual distance values hence it gives a realistic expanded region of class influence. (iii) Thornton [21] demonstrated that many of the UCI repository datasets [16] contain data points that exhibit neighborhood properties.

In our approach, for each class an initial candidate boundary point set is generated from the classified dataset based on the reverse kNN property, then extended through the region of influence and finally presented using triangulation based techniques. We focus on the analysis of labeled data using visualization of boundaries based on regions of influence. The visual presentations of regions of influence has various applications in prediction-based systems, market analysis and location-based services.

The paper is organized as follows : Section 2 describes our proposed method of extraction and representation of regions of influence in detail. Section 3 presents the interactive visualization toolkit developed as part of our work along with experimental analysis of various real and synthetic datasets. Finally, conclu-

sions as well as future work is presented in section 4.

2 Regions of Influence Visualization

Given a D-dimensional dataset $X = \{x_1, x_2, ..., x_N\}$, N is the number of data points, $x_i = (x_{i1}, x_{i2}, ..., x_{iD})$ containing set of M classes, $C = \{c_1, c_2, ..., c_M\}$.

Definition 1 : **k-nearest neighbors** The k-nearest neighbors (kNN) of point p are the set of k points from dataset X which are closest to p based on a distance metric $dist()$, denoted as $kNN(X, p)$.

Definition 2 : **Reverse k-nearest neighbors** The reverse k-nearest neighbors ($kRNN$) [14] of point p are the points $x_i \in X$ that have p as one of their kNNs, denoted as

$$kKNN(p) = \{x_i | x_i \in X, p \in kNN(X, x_i)\} \quad (1)$$

where $kNN(X, x_i)$ is the set of k nearest neighbors of point x_i in dataset X based on some distance metric $dist()$. The $kRNN$ property provides a view of the data distribution around the point and estimates the density of points around a point relative to itself. Therefore, the number of $kRNNs$ vary based on this implicit density estimation captured by the points.

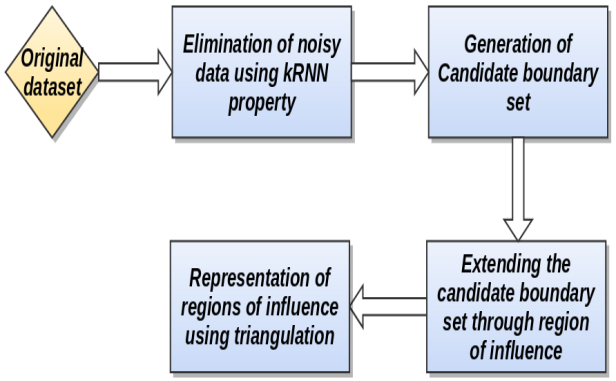


Figure 1: *Steps for generation of Regions of Influence visualization.*

Fig. 1 provides a simplified view of the steps involved in the development of class boundaries based on regions of influence.

2.1 Removal of noisy data

The first step of the process is pre-processing the dataset to eliminate the noisy data points. This step needs to be performed before the extraction of initial class boundaries based on kRNN because in a non-uniformly distributed dataset, the boundary detection is sensitive to noisy data. In this step, we find k-nearest neighbor points (kNN) of each point in the data and calculate their number of reverse k-nearest neighbors (denoted as *kRNN-value*). For each point x_i, its kRNN-value represents the number of points which look upon x_i as one of their kNNs. The points with kRNN-value less than *noise-threshold* are filtered as noises. Very low kRNN-value of a point suggests that the point lies isolated with respect to other class points and hence could be removed from the data. These points possess very less neighborhood properties of their class and hence there is need to pay further attention to explore the possibility of their incorrect labelling.

2.2 Generation of candidate boundary set (CBS)

Candidate boundary set is an initial set of class boundary points retrieved using reverse k-nearest neighbor definition specified above. For the generation of initial boundary set of a class c_i, denoted as CBS_i, we calculate intra-class kRNN-values for all points in the class by limiting the computations of k-nearest neighbors within the points belonging to the same class. In this step, intra-class kNN algorithm is performed to compute each point's k-nearest neighbors lying within the same class and the output information is stored in kNN-files. Now for each point in the class, its intra-class kRNN-values are calculated and points with kRNN-value less than *rBoundary threshold* are filtered and extracted as *candidate boundary-set* of the class. These candidate boundary sets (CBSs) consists of tight boundary points that exist at the edge of the dense group (class) of points. These boundary points provide an initial estimate of the influence region of the class.

However, the complexity of naive kNN algorithm is $O(N^2)$ and the complexity of naive kRNN algorithm is $O(N^3)$, which would be the bottleneck in all the computations specified above. We use I/O optimized GORDER (or the G-ordering kNN) technique [23] in all the kNN and kRNN calculations to make our system more efficient. Gorder is a block nested loop join that includes sorting, join scheduling, and distance computation filtering that results in reduction to both I/O and CPU costs in kNN and kRNN queries [23]. Our implementation for computation of kNN and kRNN queries of all points in the dataset is similar to Gorder kNN join described in BORDER technique [24].

2.3 From candidate boundary set to region of influence

The third step is the extension of generated CBS of each class through the region of influence using kNN knowledge.

Definition 3 : **(Region of influence around class)** The region around the class c_i where any point lying within the region is likely be classified to the c_i based on a neighborhood-based classifier. This notion of influence region represents subset of class features that exist even beyond the candidate class boundary. The region is parameterized based on distances from points contained in the class.

Definition 4 : **(Majority kNN class of a set of points)** Consider a point-set S containing a set of d-dimensional points, the class which is most common among kNNs of all the points contained in S is referred as *majority kNN class*, denoted as $majKNNClass(S)$. The computation of the most common class is done using majority voting as described in Algorithm 1.

Definition 5 : **(Hyperplane)** A d-dimensional hyperplane is represented as

$$\sum_{0 \le i < d} a_i x_i + a_d = 0 \quad (2)$$

where x_0 to x_{d-1} are Cartesian point coordinates and $(a_0, a_1, .., a_d)$ are coefficients subject to constraint that at least one of a_0 to a_d should be non-zero. In 2D, equation of hyperplane represents a line while in 3D it represents a plane. A hyperplane separates a d-dimensional space into two

Algorithm 1 Majority kNN Class

Input

S : point-set S containing a set of D-dimensional points from dataset X.

voteCount initialized to 0 for each class c_i

1: **for** each point p in S **do**
2: **for** each point p_i in $kNN(X,p)$ **do**
3: increment *voteCount* of cluster which p_i belongs to by 1;
4: **end for**
5: **end for**
6: **return** Cluster with maximum *voteCount*

halves positive and negative, which can be specified as per our convention. Given a set of d-dimensional points S (size(S) = d), it is possible that a unique hyperplane can be determined that passes through them. For our convention, we refer to the points involved in hyperplane construction as *vertices* of the hyperplane.

Definition 6 : **(kNNShift of a hyperplane)** Consider a hyperplane h constructed from a set of d-dimensional vertices. Let S be a point-set containing vertices of h and their centroid and *minDist* be the minimum distance between any pair of points in the noise-eliminated dataset. If all points in S are shifted along the orthogonal direction of the hyperplane by *minDist* until the majority kNN class of S changes. Fig. 2 provides a simple demonstration of kNNShift of a 3D-hyperplane h where its vertices are shown as *red* colored points. In the figure, class-A represents group of white-colored points and class-B represents group of black-colored points. Dotted arrows in the image connect vertices of hyperplane to their respective kNNs ($k = 2$). Initially, majority kNN class of hyperplane h is class-A (as $voteCount(A) = 6$ and $voteCount(B) = 0$). When h is shifted along the normal direction by *minDist*, for the new hyperplane h' (consisting of projected vertices from h) the majority kNN class changes from A to B (as $voteCount(A) = 1$ and $voteCount(B) = 5$). This change is referred as kNNShift of h. The notion of kNNShifts provides a limit to the spread of region of influence around a class.

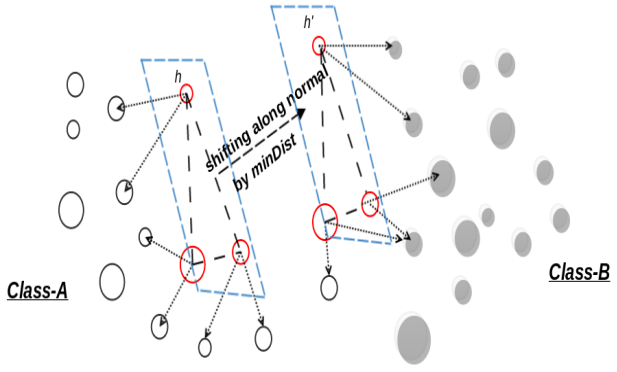


Figure 2: *Iterative kNNShift of a hyperplane h.*

Algorithm 2 describes the steps involved in spreading each of the class boundaries through the region of influence characterized by kNN-classifier. Let ROI_i stores the set of hyperplanes which enclose the region of influence around class c_i, it is initialized to $\emptyset$. Firstly, we select the point p with minimum kRNN value from the CBS of class c_i (denoted as CBS_i). Let point-set S contain point p and its kNNs within CBS_i (intra-CBS kNNs, denoted as $kNN(CBS_i, p)$), such that $size(S) = (k+1) \geq d$. We construct a distinct set of hyperplanes H from S. (Note : H contains all possible non-coplanar hyperplanes that can be constructed using the point-set S. H can contain maximum of ${}^{(k+1)}C_d$ hyperplanes). Points with least kRNN value lie at the fringe of the dense regions of class, thus hyperplanes constructed using these points help to extend the boundary in different directions.

Algorithm 2 Extending the CBS through region of influence

Input

CBS_i : CBS of class c_i.

minDist : minimum distance between any two pair of points.

Output

ROI_i : Region of influence around class c_i as a set of hyperplanes.

1: $ROI_i \leftarrow \emptyset$;
2: **while** $CBS_i \neq \emptyset$ **do**
3: $p \leftarrow$ point with minimum *kRNN-value* in CBS_i;
4: $S \leftarrow$ point-set $\{kNN_i(CBS_i, p) \cup p\}$ and $size(S) = (k+1) \geq d$;
5: compute a set of distinct hyperplanes H from S;
6: **for** each hyperplane h in H **do**
7: **if** h satisfies constraints I and II **then**
8: $P \leftarrow \{vertices(h) \cup centroid(vertices(h))\}$;
9: *normal* $\leftarrow$ orthogonal direction of h pointing from positive to negative half-space;
10: $Cur \leftarrow P$; $Prev \leftarrow \emptyset$;
11: **while** $majKNNClass(Cur)$ is c_i **do**
12: $Prev \leftarrow Cur$;
13: $Cur \leftarrow$ shift point-set Cur along *normal* of h with *minDist*;
14: **end while**
15: **if** $Prev \neq \emptyset$ **then**
16: $h' \leftarrow$ hyperplane containing points in $Prev$;
17: $ROI_i \leftarrow ROI_i \cup h'$;
18: **end if**
19: **end if**
20: **end for**
21: $CBS_i \leftarrow CBS_i \setminus S$;
22: **end while**
23: **return** ROI_i

We then perform filtering on H to reduce its size, based on the orientation of points in the CBSs with respect to hyperplanes. Hyperplanes which do not satisfy both the constraints I and II are removed from H.

I : Hyperplane should divide the points in CBS_i unequally into two half-spaces based on their orientation. The half-space containing more points from CBS_i is referred as *positive* half-space while other half-space containing less number of points from CBS_i is referred as *negative* half-space.

II : The negative half-space with respect to hyperplane (as defined in I) should contain at least k points from CBSs of other classes so that there exist a possibility of kNNShift.

The orthogonal direction of each hyperplane in H is, pointing from their *positive* half-space to *negative* half-space (as defined in I). Each hyperplane h in H is shifted along the orthogonal direction with distance *minDist* iteratively until $kNNShift$ is encountered, as demonstrated in *Definition 6*. The set of hyperplanes h' (projected just before the kNNShift) are retrieved and stored in ROI_i and the points in S from CBS_i. The algorithm is executed until all points are removed from CBS_i, that is for class c_i, the algorithm runs for $\lceil \frac{size(CBS_i)}{k+1} \rceil$ steps. These steps are performed independently for each of the classes. Intra-CBS kNN and kRNN values are pre-computed before the execution of algorithm 2 using GORDER technique to optimize its performance.

2.4 Representing regions of influence using triangulation

The region of influence around a class is formulated as a set of hyperplanes consisting of d-dimensional vertices. For visualization purposes, $2D$-hyperplane can be presented as a line-segment and $3D$-hyperplane as a triangle. However, presenting the region of influence as a set of disconnected line-segments or triangles is a crude display of "shape" of class boundaries. Hence, we produce a linear-approximation of the shape represented by these unorganized hyperplanic equations, using *alpha-shapes* [5]. Alpha-shapes is a Delaunay-triangulation based technique which provides a linear-approximation to unorganized spatial point-sets. Different family of shapes can be easily derived based on parameter settings of the constant α. We use efficient algorithms [4] to produce alpha-shapes to outline the shape represented by the regions of influence of the classes. In alpha-shapes, it is difficult to determine the best value of α that produces neither too crude nor too fine a shape for the classes boundaries. To avoid the need of setting different values of α for different classes, we find the smallest value (using binary search) such that all vertices of the hyperplanes either lie on the boundary or inside of the triangulated alpha-shape.

Therefore, the representation of class boundaries as alpha-shapes contains a set of triangulated facets in case of 3D (or line-segments in case of 2D) based on the value of α found above. Regions of influence around classes developed in our work can be overlapping. In order to understand what exactly causes the overlap, we use triangle-triangle intersection algorithms [19] to detect and visualize the overlaps that appear. Once the triangulated objects enclosing the region of influence around the classes are formed, the toolkit (which will be described next) allows to detect and visualize the intersections between different class boundaries representing regions of influence. To reduce the computational complexity of finding the intersections, we use fast OpenGL implementation of axis aligned bounding box technique [20]. These intersections between the triangulated class boundaries enables us to visualize connections between classes with ease.

3 Interactive Visualization Toolkit

We implemented a toolkit (Fig. 3) as part of our work which allows users to interact and manipulate the visualizations obtained from the process described above. This type of interaction is critical for the search of interesting characteristics about the classification model (or clustering technique [8, 9, 11]) used on the labeled dataset. The toolkit contains features which allow users to alter the visualizations based on feedback on different parameter

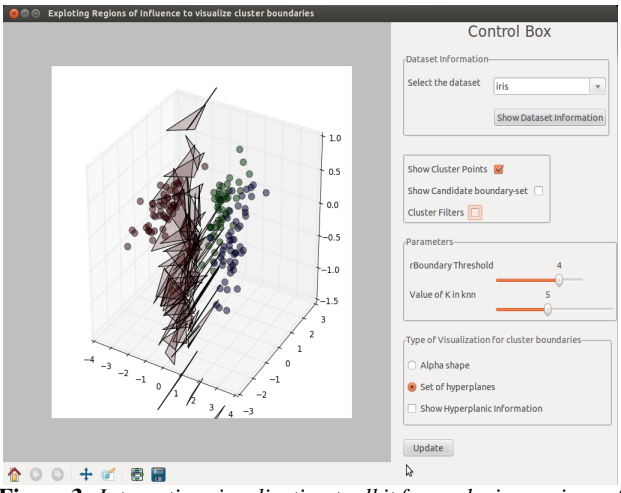


Figure 3: *Interactive visualization toolkit for exploring regions of influence based class boundaries.*

settings for (i) *noise threshold*, (ii) *rBoundary-threshold* and (iii) *value of 'k'* in all kNN computations. It presents two different visualization for regions of influence, (1) using crude presentation as set of hyperplanes and (2) using triangulated alpha-shapes. Another interaction technique available in the toolkit is to select a component of the boundary (a facet of the alpha-shape or a hyperplane) and highlight the set of candidate boundary points which contribute to its creation along with the direction of spread (Fig. 4 (c)). This helps developers of the classification model to reason why certain shape or extent of the class boundary appear in these visualizations. Users can also extract mathematical equations of the hyperplanes or facets of the triangulated alpha-shape to perform further analysis of the classification model.

3.1 Experimental Analysis

We performed experiments on various real and synthetic datasets using the toolkit to show the effectiveness of our algorithm in handling classes of arbitrary size, shape and density. And to visualize nature and characteristics of the classifier (or clustering technique) used on the labeled dataset in seperating data points into different sets. Fig. 4 shows results on 2D dataset (240 instances) containing 2 clusters - *red cluster (r)*, *green cluster (g)*, generated using *flame clustering* method [6]. While the procedure of generation of clusters is 'black-box' to us, we ran our algorithm to generate cluster boundaries. The regions of influence retrieved from the process shows an overlap (Fig. 4(e)). Points lying within the overlapping region (marked *blue* in fig. 4(e)) are likely to show subset of properties from both the classes and also suggests there is a possibility that these points were likely to be incorrectly predicted by the original classifier. While understanding the flame clustering procedure [6], we found that the points lying in the overlapping region of influence were classified using local-neighborhood approximation of fuzzy memberships. However, other points were cluster supporting objects (CSO) with full membership to represent one cluster. Hence, generation of cluster boundaries using regions of influence helps us visually understand the characteristics and nature of the classifier used on the dataset.

Fig. 5 shows results of boundary visualization extended through region of influence for the 2D dataset consisting of

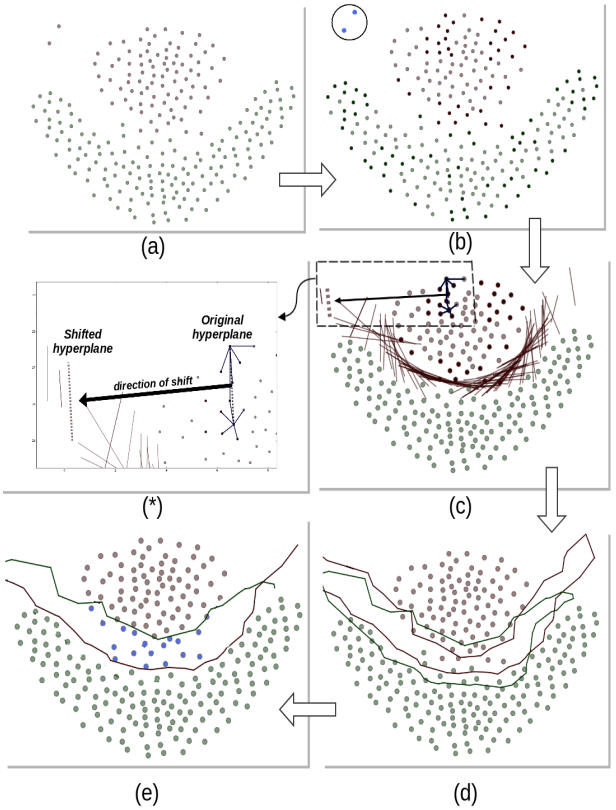


Figure 4: *(a) Original clustered FLAME dataset. (b) CBS (highlighted) of both classes. Circled blue points are filtered as noise (very low kRNN values) and removed from the dataset as they possess minimal neighborhood properties. (c) Representation of regions of influence around red class as set of hyperplanes. (*) Zoomed version of the squared box in image (c). Based on the selected hyperplane, toolkit shows the original hyperplane along with the direction of shift. (d) 2D alpha-shape representation of class boundaries for both classes. (e) Highlighting only the overlapping region of influence and points contained within the overlap (blue).*

7 uneven-sized clusters (connected via narrow bridges between them) retrieved from *aggregation clustering* [7]. However all 7 clusters are of spherical shape but with different sizes, density and orientations, cluster boundaries based on regions of influence have different amounts of extension for each cluster along different directions. This visualization reveals regions where the connectivity of their influence (inter-cluster relationships) occurs. There exist minimal overlap between regions of influence of the clusters suggesting that the algorithm used properly seperates points into different clusters. [7] demonstrates that aggregation clustering performs better than any of the clustering algorithms on this dataset.

We also conducted our experiments on Iris dataset from UCI repository [16] (Fig. 6, 7, 8, 9). The dataset is 4D consisting of 150 instances - 50 in each of the three classes, *Setosa* (*St*), *Versicolour* (*Vc*) and *Virginica* (*Vg*). We performed principal component analysis (PCA) [10] to reduce the number of dimensions to 3. As the visualization (Fig. 6) suggests there exist collisions between the 3D facets of triangulated alpha-shape

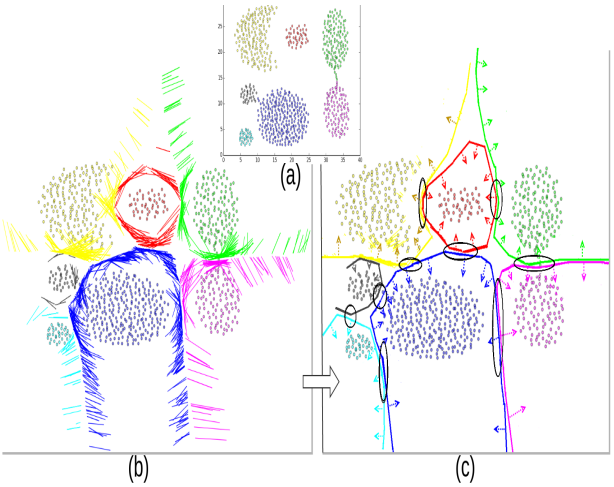
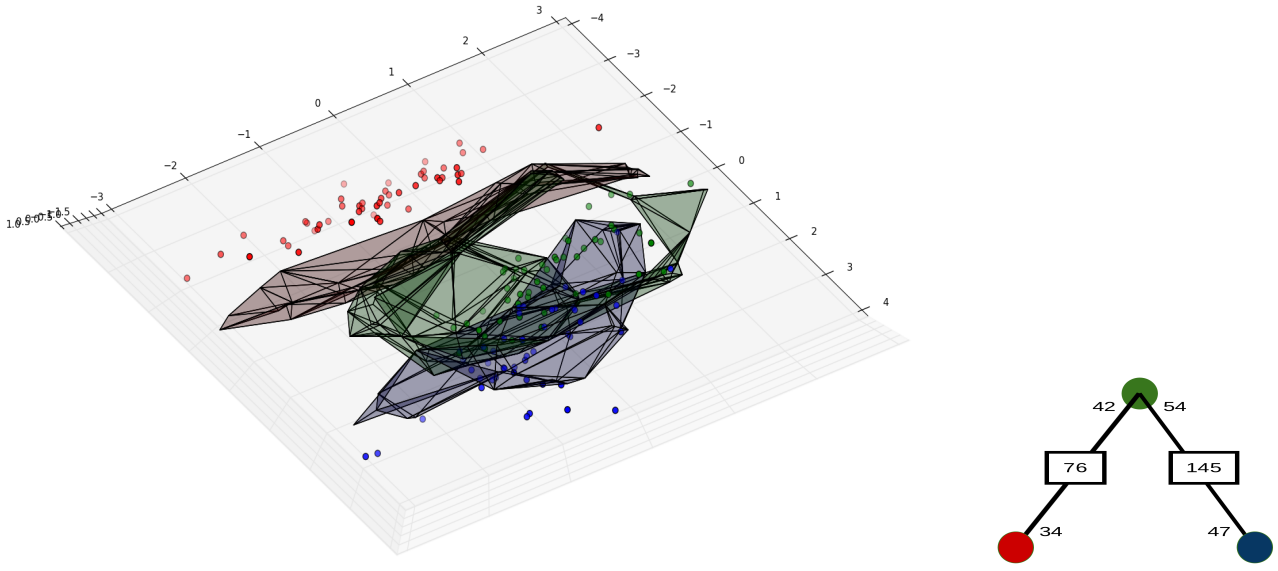


Figure 5: *(a) Original clustered aggregation dataset. (b) Cluster boundaries bounding regions of influence as a set of hyperplanes. (c) Cluster boundaries showing only one side of the linear approximated alpha-shape. Black rings highlight the regions of maximum connectivity of regions of influence of different clusters.*

boundary, hence we performed collision detection algorithm to capture their counts. The collisions graph (Fig. 6(b)) shows that the number of collisions between *St*-*Vc* were 76 arising from 34 and 42 distinct facets of *St* and *Vc* respectively while between *Vc*-*Vg* were 145 arising from 54 and 47 distinct facets of *Vc* and *Vg* respectively. Collisions between *Vc*-*Vg* boundary are nearly double that of *St*-*Vc* boundary while arising from similar number of distinct facets of the alpha-shape suggesting that *St* class is well separated with little contact with *Vc* class, while other two classes are more connected along their boundary. Using the toolkit we can interactively explore the points within the classes causing collisions between their regions of influence. Table 1 shows the processing time taken (in seconds) for the computation of regions of influence as set of hyperplanes for the experimental datasets discussed above, with value of *rBoundary-threshold=5*.

Table 1: *Processing time for the computation of Regions of influence as set of hyperplanes for different datasets (rBoundary threshold = 5)*

Dataset	value of k	Processing Time (Sec)
Flame	4	1.55
Size = 240	5	1.68
Dimension = 2	6	1.87
Num of classes = 2		
Aggregation	4	2.77
Size = 788	5	2.91
Dimension = 2	6	3.22
Num of classes = 7		
Iris PCA	4	2.05
Size = 150	5	2.37
Dimension = 3	6	2.83
Num of classes = 3		



(a) (b)

Figure 6: *3d PCA of IRIS dataset. (a) (Topview) All class boundaries (St (red), Vc (green) and Vg (blue)) as 3D alpha-shapes. (b) Graph showing number of collisions between classes along with the number of distinct facets of the alpha-shape which caused collisions.*

3.2 Evaluating classifiers

Our toolkit can also be used to evaluate the prediction capability of a classifier using class boundaries based on regions of influence. We performed repeated random sub-sampling validation (also known as Monte Carlo cross validation) [18] on Flame and Iris dataset for different values of *rBoundary-threshold*. In each run, dataset was divided randomly into training-set ($x\%$) and testing-set $(100-x)\%$ where x can be specified in the toolkit. Then,

1. A value for *rBoundary-threshold* was selected in the toolkit.
2. Class boundaries based on regions of influence were generated using the training-set for different values for k.
3. For each point in testing-set, (i) its class-label was predicted based on its orientation and distance with respect to the trained regions of influence of all classes and (ii) then validated with its actual class-label.
4. Based on selected value for *rBoundary-threshold* in step 1, the toolkit finds and fixes the value of k which provides maximum accuracy for all the test points.

These steps were performed for 20 runs with random splits and average error rates were calculated. Obtained *class-wise error** and *combined error*† rates in making correct predictions of the points in testing-set using trained class boundaries based on regions of influence are listed in Table 2. The table only shows top 3 values of *rBoundary-threshold* with maximum average accuracy for the dataset split. The data shows that even for the smaller size of the training-set ($x \leq 50$) majority of the test points lie within the region of influence of their class. Different values of thresholds provide different error rates for the classes due to difference in their topologies in terms of density, size and orientation etc. The toolkit allows to interactively explore the cross-validation setup to find optimal values for thresholds such that regions of influence provides the best estimate to the spread of class boundaries (with minimal combined error rate) and also highlights the test points in each class that are incorrectly predicted.

Thus the visualization toolkit presented in this paper enables interactive classification boundary visualization to explore the characteristics of classifier used on the dataset.

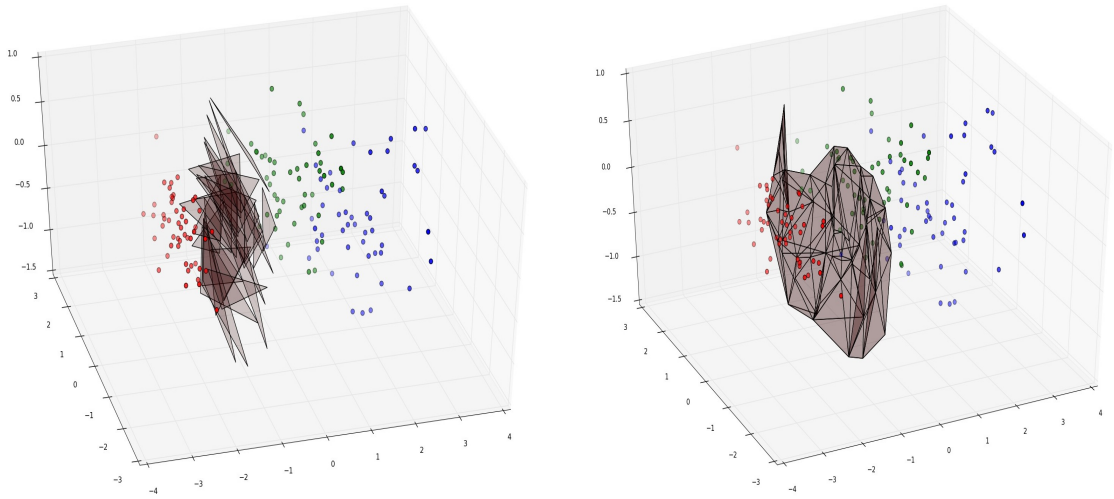
Table 2: *Performance of class boundaries based on regions of influence for Flame and Iris tested with random sub-sampling cross-validation with 20 runs for different values of rBoundary-threshold. Highlighted values in the table shows the minimum combined error for a setup.*

Dataset(*)	rBoundary-threshold	Class-wise error (class-label = error%)	Combined error%
Flame	*3*	*r = 15.56, g = 4.37*	*8.50*
(25 / 75)	*4*	*r = 6.08, g = 8.30*	***7.67***
	5	*r = 3.67, g = 10.76*	*8.39*
Flame	*3*	*r = 3.91, g = 7.79*	*5.67*
(50 / 50)	*4*	*r = 2.68, g = 5.24*	***4.83***
	5	*r = 2.53, g = 6.83*	*5.91*
Iris	*4*	*St = 0.27, Vc = 20.63, Vg = 11.92*	*15.16*
(25 / 75)	*5*	*St = 0.23, Vc = 17.94, Vg = 7.27*	*9.69*
	6	*St = 0.31, Vc = 15.62, Vg = 5.42*	***7.88***
Iris	*4*	*St = 0.00, Vc = 4.37, Vg = 9.27*	*7.50*
(50 / 50)	*5*	*St = 0.00, Vc = 1.27, Vg = 5.77*	***3.61***
	6	*St = 0.00, Vc = 2.19, Vg = 4.82*	*3.89*

(*) training / testing(%)

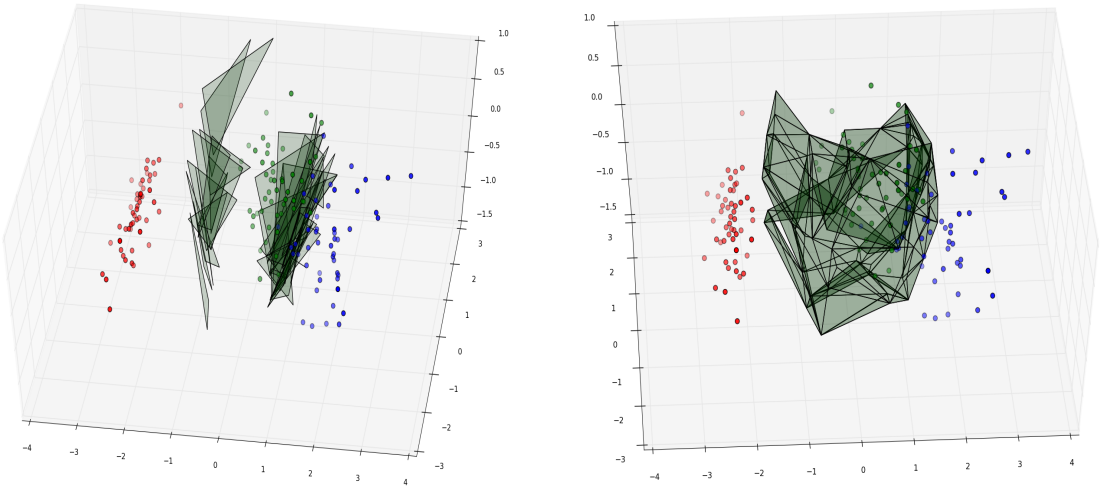
**class-wise error rate* are % of incorrectly predicted test points in each class.

†*combined error rat* is the overall % of all incorrectly predicted points in the testing-set.



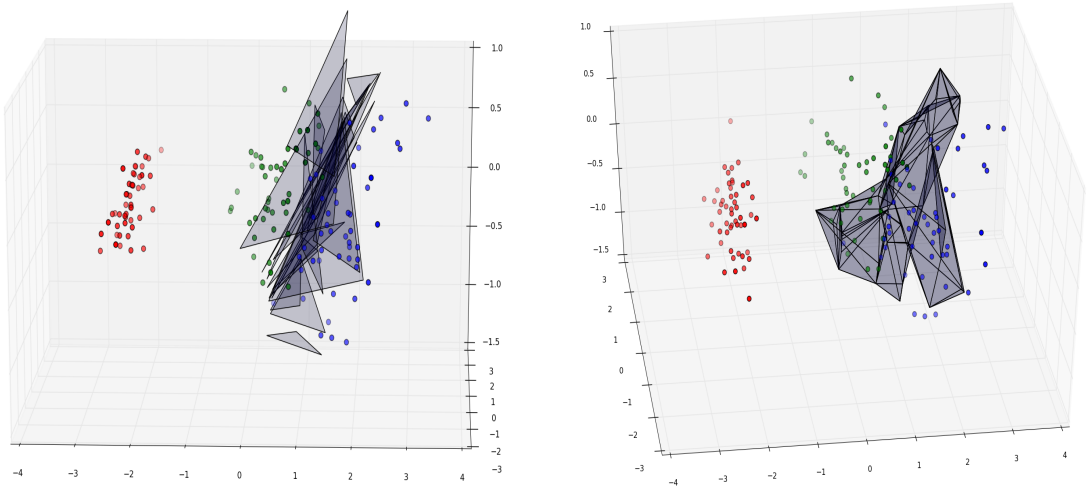
(a) **(b)**

Figure 7: *Representation of regions of influence around class Setosa (red) using (a) set of hyperplanes and (b) alpha-shapes. The boundary possess a linear alpha shape suggesting the class is linearly separable from other two.*



(a) **(b)**

Figure 8: *Representation of regions of influence around class Versicolour (green) using (a) set of hyperplanes and (b) alpha-shapes. Since the class lie between two classes, the alpha shape representation of the class encapsulates all the points in the class. The closed form of alpha-shape is only obtained for this class.*



(a) **(b)**

Figure 9: *Representation of regions of influence around class Virginica (blue) using (a) set of hyperplanes and (b) alpha-shapes.*

4 Summary and Future-work

Class boundary visualization based on regions of influence presented in this paper assists in comprehending the nature and characteristics of the classification model and visualizing the regions of connectivity between different classes. The visualization toolkit also enables analysis and evaluation of class boundaries via interactive techniques. The idea of visualizing the regions of influence around classes allows us to study the spread of the class features even beyond tight enclosure of class given by points in the class. Study of high-dimensional class boundaries is beyond the scope of this paper. Our intent is to present an interactive visualization toolkit which would take any type of 2D/3D dataset (lower dimensional mappings like principal component analysis [10], multi-dimensional scaling [2], self-organizing maps [13] or feature projections of high-dimensional data) and provide class boundaries visualization based on regions of influence. We plan to provide an interface within the tool to compare and study effects of these procedures on high-dimensional class boundaries. We also plan to allow the toolkit to visually compare the type and quality of different clustering algorithms on a dataset. Future work also includes effective representation of collisions detected between two triangulated class boundaries.

References

[1] Q. Bao-Zhi, Y. Feng, and S. Jun-Yi. Brim: An efficient boundary points detecting algorithm. In Proceedings of Advances in Knowledge Discovery and Data Mining, pages 761-768, 2007.

[2] C. Bentley and M. Ward. Animating multidimensional scaling to visualize n-dimensional data sets. In Proceedings of IEEE Symposium on Information Visualization, volume 126, pages 72-73, 1996.

[3] D. Caragea, D. Cook, H. Wickhamand, and V. Honavar. Visual methods for examining svm classifiers. Visual Data Mining: Theory, Techniques and Tools for Visual Analytics, pages 136-153, 2008.

[4] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. IEEE Transactions on Information Theory, IT-29(4), July 1983.

[5] H. Edelsbrunner and E. Mucke. Three-dimensional alpha shapes. In Proceedings of the 1992 Workshop on Volume visualization, Boston, Massachusets, USA, pages 75-82, October 1992b.

[6] L. Fu and E. Medico. Flame, a novel fuzzy clustering method for the analysis of dna microarray data. BMC bioinformatics, 8, 2007.

[7] A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. ACM Transactions on Knowledge Discovery from Data (TKDD), 1:1-30, 2007.

[8] J. Han and M. Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann, 2001.

[9] D. Hand, H. Mannila, and P. Smyth. Principles of Data Mining. The MIT Press, Cambridge, Massachusetts., 2001.

[10] J. E. Jackson. A users guide to principal components. 1991.

[11] A. K. Jain and R. C. Dubes. Algorithms for Clustering Data. Prentice-Hall, 1988.

[12] D. Keim. Information visualization and visual data mining. IEEE Transactions on Visualization and Computer Graphics, 8:1-8, 2002.

[13] T. Kohonen. The self-organizing map. In Proceeding of the IEEE, volume 78, pages 1464-1480, 1990.

[14] F. Korn and S. Muthukrishnan. Influence sets based on reverse nearest neighbor queries. In Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pages 201-212, 2000.

[15] O. Melnik. Decision region connectivity analysis: A method for analyzing high-dimensional classifiers. Machine Learning, Kluwer, Netherlands, pages 321-351, 2002.

[16] C. Merz and C. Blake. Uci repository of machine learning databases (http://archive.ics.uci.edu/ml). University of California at Irvine, Department of Computer Science.

[17] A. Moreira and M. Santos. Concave hull: a k-nearest neighbours approach for the computation of the region occupied by a set of points. International Conference on Computer Graphics Theory and Applications, pages 61-68, 2007.

[18] R. R. Picard and R. D. Cook. Cross-validation of regression models. Journal of the American Statistical Association, 79(387):575-583, 1984.

[19] C. Sabharwal, J. Leopold, and D. McGeehan. Triangle-triangle intersection determination and classification to support qualitative spatial reasoning. Research Journal of Computer Science and Computer Engineering with Applications, 48:13-22, 2013.

[20] H. A. B. Sulaiman, M. A. Othman, M. Z. A. A. Aziz1, and A. Bade. Implementation of axis-aligned bounding box for opengl 3d virtual environment. ARPN Journal of Engineering and Applied Sciences,

10:13-22, 2015.

[21] C. Thornton. Separability is a learners best friend. 4th Neural Computation and Psychology Workshop, London, pages 40-46, 1997.

[22] V. Vapnik. An overview of statistical learning theory. IEEE Transactions on Neural Networks, 10(5), 1999.

[23] C. Xia, W. Hsu, M. L. Lee, and B. C. Ooi. Gorder: An efficient method for knn join processing. In Proceedings of the 30th International Conference on Very Large Data Bases (VLDB), pages 756-767, 2004.

[24] C. Xia, W. Hsu, M. L. Lee, and B. C. Ooi. Border: Efficient computation of boundary points. IEEE Transactions on Knowledge and Data Engineering, 18(3):289-303, 2006.

[25] Z. Yan and C. Xu. Using decision boundary to analyze classifiers. 3rd International Conference on Intelligent System and Knowledge Engineering, pages 302-307, 2008.

Author Biography

Pallav Tinna received his B.Tech in Computer Science from International Institute of Information Technology, Hyderabad (IIIT-Hyderabad). He is currently Masters of Science student in Computer Science at IIIT-Hyderabad. His research interests include data visualization and machine learning.

Kamalakar Karlapalem is a Professor at IIT, Gandhinagar and on leave from IIIT-Hyderabad. His areas of research are visual data analytics, multi agent systems, database systems, and workflow systems. He was faculty member at HKUST, Hong Kong.