An L^∞ Norm Visual Classifier

Anushka Anand
University of Illinois at Chicago
Chicago, IL
aanand2@lac.uic.edu

Leland Wilkinson
SYSTAT Inc.
Chicago, IL
leland.wilkinson@systat.com

Dang Nhon Tuan
University of Illinois at Chicago
Chicago, IL
tdang@cs.uic.edu

***Abstract*—We introduce a mathematical framework, based on the L^∞ norm distance metric, to describe human interactions in a visual data mining environment. We use the framework to build a classifier that involves an algebra on hyper-rectangles. Our classifier, called VisClassifier, generates set-wise rules from simple gestures in an exploratory visual GUI. Logging these rules allows us to apply our analysis to a new sample or batch of data so that we can assess the predictive power of our visual-processing motivated classifier. The accuracy of this classifier on widely-used benchmark datasets rivals the accuracy of competitive classifiers.**

***Keywords*-Visual data mining; Supervised classification;**

I. Introduction

Visual data mining(VDM) is described as "the integration of the human mind's exploration abilities with the processing power of computers to form powerful knowledge discovering environments" [1]. The advantages of such visual data exploration over automatic data mining techniques that function as black boxes needing specialized tuning per use, are twofold [2]:

- it is intuitive, so tools would not impose knowledge requirements (about the data or machine learning tecnhiques) on the users while providing them insights into their data.
- such exploration would be general or adaptable to inhomogenous data.

Many have pointed out the unique strength of exploiting the highly evolved pattern-detection capabilities of the eye [3]. However, it is difficult to construct a visual tool that can be used to identify similar structure in different datasets. The research presented here explores a new approach that marries VDM and inference. Our focus was to study the pattern recognition and cognitive processing skills of humans for use in the design of algorithms embedded in a classifier. We propose a novel VDM environment framework that captures human interactions in visual processing of data as mathematical structures. And we present a novel classification algorithm that is based on the proposed framework.

A. The Supervised Classification Problem

We begin with a set of *training data*

$$T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \tag{1}$$

with real feature vectors $\mathbf{x}_i = [x_{i1}, \ldots, x_{ip}] \subset \mathbb{R}^p$ and positive integer class labels $y_i \in \{y_1, \ldots, y_g\} \subset \mathbb{N}$. The problem involves finding a classification function

$$X \mapsto Y \tag{2}$$

that maps $\mathbf{x} \in X$ to $\hat{y} \in Y$ with minimum *classification error* given by

$$E[\sum_i^n (y_i - \hat{y}_i)^2] \tag{3}$$

II. Related work

Approaches to the classification problem must deal with some serious obstacles. First, there is the *curse of dimensionality*. The second problem is computational complexity. Polynomial time algorithms are hopeless in high-dimensional spaces. The third problem is nonlinearity and the fourth is handling categorical variables.

There have been numerous approaches to overcoming these problems. One approach is to use hybrid classifiers [4][5]. Another is to reduce dimensionality through principal components, k-means clustering [6], or random projections [7][8].

Our algorithm is, to the best of our knowledge, the only integrated solution that addresses all of these problems satisfactorily. Our idea is similar to that of describing a mathematical learning model based on the visual system [9]. We use random projections in a novel way to create a VDM environment. The resulting algorithm goes beyond plugging a visualizer into an existing classification algorithm, as in [10] [11] [12].

Our framework is based on using rectangular description regions to capture human interactions with data. Perhaps the most widespread use of rectangular description regions is in recursive partitioning trees [13] [14]. These methods partition a space into nested rectangular regions that are relatively homogeneous over the values of a predicted variable. Our approach is not restricted to a partitioning, as our regions need not be disjoint or exhaustive. Although combinations (unions and products) of rectangles have been used recently to define clusters in data mining [15] [16] [17], this is the first paper that we are aware of that uses covering (as opposed to partitioning) rectangles efficiently in supervised classification.

III. The Visual Data Mining (VDM) Environment Framework

Our idea is neither statistical model nor probability distribution based. It is based on what we call Visual Model Based Transformations (VMBT).

A. Visual Model Based Transformations

To explain the motivation behind this idea, we use an analogy to the way the Fourier Transform is used in data analysis. In the Fourier Transform, we map a real-valued point to the complex plane in order to discover patterns that would be difficult to detect in the real domain. To interpret our results, we invert the transformation. With VMBTs, we map a point to a visual context, analyze structure in that world to obtain a visual model, and then back-transform our visual model to apply the result. This idea was introduced in [18] and an analytic application was demonstrated in [19]. Our system is based on Hypercube Description Regions that provide the basis for a formal description of structures suitable for VDM.

B. Hypercube Description Regions

In this paper, we employ the L^∞ or *sup* metric:

$$||x||_\infty = sup(|x_1|, |x_2|, \ldots |x_n|)$$

when we search for nearest neighbors in L^∞ space. Because we restrict our model to a finite-dimensional vector space, we will use $max()$ instead of $sup()$ from now on.

We define a *hypercube description region* (HDR) as the set of points less than a fixed distance from a single point (center) using the L^∞ norm. A *weighted hypercube description region* is an HDR that uses the weighted L^∞ norm:

$$||x||_\infty = max(w_1|x_1|, w_2|x_2|, \ldots w_n|x_n|).$$

We will assume the term HDR refers to this more general case (capturing rectangles) from now on.

C. Composite Hypercube Description Regions

We define *local structure* as the set of points defined by a single hypercube (an L^∞ ball); these points are members of an HDR.

We define *large-scale structure* as the set of points defined by two or more hypercubes under the operations of union, intersection and difference (set-theoretic complement). We call this large-scale structure a *composite hypercube description region* (CHDR).

CHDRs have the following three properties which are fundamental for this approach:

1) CHDRs are closed under the operations of union, intersection, and difference.
2) These operations can be provided with simple, intuitive user interfaces.
3) Complex structures in data are well approximated by CHDRs. This assertion is formalized and argued in [16].

The benefits of this alternative view of data structures are:

- It simplifies the specification of neighborhoods because they are product sets of intervals. Our specifications can be expressed in a basic algebra on intervals.
- It allows us to specify in simple expressions or *rules* relatively complex geometric objects through CHDRs.

Interval selection for set inclusion is faster than Euclidean-distance-based selection. Using intervals enables us to simplify the set-wise algebra on the original variables.

IV. The VisClassifier Algorithm

Our visual classifier algorithm, called VisClassifier, implements the VDM environment for the supervised classification task enabling human interactions to be captured into CHDRs (and consequently rules made up of intervals). The flow of the steps in VisClassifier is described in Algorithm 1 with each step described in detail in subsections below.

A. Step 1: Coding

There are two ways to handle categorical variables:

- dummy code them (create a separate binary variable for each category value). This is how most classifiers handle them.
- code them as unordered strings.

We do the second. This method has the advantage over tree classifiers as there is no need to make pairwise comparisons of categories. The viewer will make those comparisons visually without assistance because each categorical value ultimately appears in a display at a different discrete location (see Fig. 1). The viewer can select multiple categorical values in one rectangle or in several; the result is the same.

B. Step 2: Transforming

All numerical variables are checked for potential transformation. The purpose of a transformation is to reduce overplotting in our displays caused by long-tailed distributions. Our transformation is a reflected log:

$$f(x) = \begin{cases} 0 & x = 0 \\ sgn(x)log(1+|x|) & x \neq 0 \end{cases} \quad (4)$$

If the transform alleviates extreme skewness or kurtosis, we work from this point on with the transformed variable. After transforming, we rescale all columns of the data to the unit interval.

C. Step 3: Choosing

At each iteration, we choose the target class, C_k ($k = 1, \ldots, g$), as the one with the most remaining points, display its data points as yellow dots (see Fig. 1) and visually classify it. This is repeated until all points are classified or the user chooses to stop. Steps 4-7 work with the chosen C_k. Our classifier is a one-against-all classifier.

Algorithm 1: The sequence of steps of the VisClassifier algorithm

Data: Data set X with N records, P variables and G groups or classes

Result: Error rate on prediction results of VisClassifier on the testing data (a random sample from the input data)

//** IV-A Step 1: Coding **
foreach *Categorical variable CV* **do**
  Code values of *CV* as integers
//** IV-B Step 2: Transforming **
foreach *Numerical variable NV* **do**
  Check *NV* for potential transformation to reduce extreme skewness
$TrainingData \leftarrow RandomSample(X, N/2)$
$TestingData \leftarrow RandomSample(X, N/2)$
while $TotalNumberOfRemainingUnclassifiedPoints > 0$ *AND* $StoppingCondition = FALSE$ **do**
  //This iteration is done when the user hits the "Next Plot" button
  //** IV-C Step 3: Choosing **
  $C_k \leftarrow$ Group with $\max\{NumberOfRemainingUnclassifiedPoints\}$
  foreach v=1 *to* 100 **do**
    //** IV-D Step 4: Projecting **
    Compute 1000 random projections
    //** IV-E Step 5: Binning **
    Bin 2D plots of the two best projections (based on the Separation Index)
  //** IV-F Step 6: Ranking **
  Rank the 100 binned 2D plots on the Purity of C_k
  //** IV-G Step 7: Displaying **
  Draw the "purest" plot with data from C_k as Yellow dots and everything else as Gray dots
  Add the user-selected-rectangle (from interaction in the GUI) to the list of *CHDRs*
//Training ends when all data is classified or the user hits the Stop button making the $StoppingCondition = TRUE$
//** IV-H Step 8: Scoring **
foreach *Record* $t_i \in TestingData$ **do**
  if t_i *inside some* $c \in CHDRs$ **then**
    if $t_i.Group \neq c.Group$ **then** Increment $NumOfMisclassifications$
  else
    $closestC \leftarrow min_{c \in CHDRs}\{InfinityDistance(c, t_i)\}$
    if $t_i.Group \neq closestC.Group$ **then** Increment $NumOfMisclassifications$
  $ErrorRate \leftarrow NumOfMisclassifications/(N/2)$

D. Step 4: Projecting

We will be computing binned plots of pairs of variables in order to do visual classification. Binning all possible pairs of variables is impractical for more than, say, 100 variables. Furthermore, orthogonal pairwise projections are unlikely to reveal separation between classes, even for well-separated convex distributions. Consequently, we compute a list of random projections and derive our plots from that list.

1) Generating Projections: The projections we use are unit-weighted projection matrices. This approach follows a recent finding that using "database-friendly" unit weights instead of Gaussian weights does not result in a substantial loss of accuracy in approximating distances [20]. Our goal is to improve robustness of our scoring in new samples. Wainer [21] proved that, under very general circumstances, a prediction model based on replacing regression coefficients with unit weights will result in *smaller* expected loss in new samples and greater robustness against outliers. We assign our weights with the following probabilities:

$$w_j = \begin{cases} 1 & \text{with probability } 1/4 \\ 0 & \text{with probability } 1/2 \\ -1 & \text{with probability } 1/4 \end{cases} \quad (5)$$

Here w_j is the projection matrix weight of the j^{th} variable.

2) Selecting Projections: To generate a plot for a class C_k, we compute 1000 candidate random projections [20]. We evaluate each candidate random projection by computing a separation measure on the centroids of each class.

Let $\bar{\mathbf{x}}_k$ be the centroid vector of instances in C_k. For each projection $\mathbf{w}_m$, $(m = 1, \ldots, 1000)$, we compute the projected mean of C_k, namely, $\bar{x}_{m,k} = \bar{\mathbf{x}}_k \mathbf{w}_m$. We take the L^∞ distance between $\bar{x}_{m,k}$ and the closest of the other projected class means. The *Separation Index* for a candidate projection $\mathbf{w}_m$ for class C_k is given by:

$$S_{m,k} = min_{l=1,\ldots,g; l \neq k} |\bar{x}_{m,k}, \bar{x}_{m,l}| \quad (6)$$

From the list of candidate projections, we select the one with the greatest shortest distance because we want a projection that has the best chance of separating a class from all the other classes. So the selected projection $\mathbf{w}_k$ for class C_k is given by:

$$\mathbf{w}_k = argmax_{\mathbf{w}_m, m=1,\ldots,1000}\{S_{m,k}\} \quad (7)$$

Our method is close in spirit to [22]. It is different, however, because we want to find 2D projections that separate class C_k from the other classes; not separate *all* the classes in a plot.

E. Step 5: Binning

We 2D bin our data in order to plot the projected densities on large datasets without overlaps. We compute 100 binned plots, in each case using the two best projections on our Separation Index. For each bin, we store the count of points

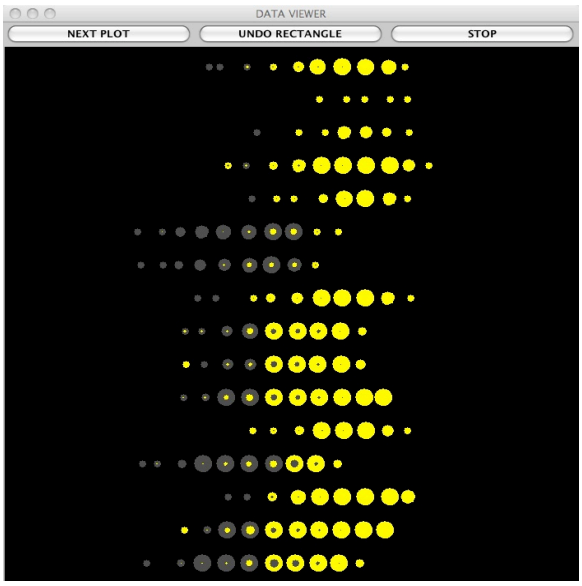

Figure 1. A rendering of the UCI Adult dataset. A categorical variable is plotted against a projection of several continuous variables. The yellow symbols highlight the current target class and the gray symbols highlight the non-target class. Yellow circles with gray centers and gray circles with yellow centers reveal the overlapping densities between the two classes.

in the bin as well as the centroid of the points in the bin. In our displays, we plot the centroids, not the bins themselves.

We base the number of bins on a formula in [23]. Given a dataset with n rows, we compute the marginal number of bins b using

$$b = \frac{3log_2(n)}{2} \tag{8}$$

There are b^2 bins per 2D plot. We use our binning to produce scatterplots with symbols located at the centroid of the points within each bin, so users are able to perceive contours of densities more sensitively.

F. Step 6: Ranking

Next, we rank our binned plots on a purity measure so that plots where C_k is well-separated from all other classes are seen first by the user. For a given target class value C_k, our purity measure is

$$P_k = \sum_{i=1}^{b}\sum_{j=1}^{b} n_{ij} I(c_{ijk}) \tag{9}$$

where

$$I(c_{ijk}) = \begin{cases} 1 & n_{ij} = n_{ijk} \\ 0 & otherwise \end{cases} \tag{10}$$

In other words, we sum the counts across all bins whose total counts of points falling in them (n_{ij}) are due only to class C_k counts (n_{ijk}).

G. Step 7: Displaying

Figure 1 shows how we display our results. After some experimenting with different types of displays, we chose to use 2D scatterplots, as most users are familiar with reading them and they make it easy to see contours of densities of the data. We use filled yellow and gray circles to represent the centroids of the bins. The symbols are sized according to the count within each bin, like a bubble plot. A gray circle inside a yellow indicates the presence of other classes as a minority. And a yellow circle inside a gray indicates a majority of other classes and a minority of the target class. The interior circle size varies based on the amount of overlap.

The display contains three buttons and a rendering area as seen in Fig. 1. The Next Plot button causes the program to move to the next target (yellow) class. This begins a new iteration, in which the user is shown fresh projections and can begin to construct a CHDR. The Undo Rectangle is a recursive rectangle remover for users to undo bad selections. The Stop button ends the training phase. We encourage users to stop when there appears to be only noise remaining. If the user pushes through to classify every single point, the program will stop. In that case, the score in the testing sample is likely to be lower. The scoring stage is initiated as soon as the training phase is done and the classifier's performance on a test sample is computed.

Fig. 2 shows how a user highlights a class with rectangles. When the user draws a blue rectangle and releases the mouse, all enclosed points turn blue. Users are encouraged to pick only solid yellows early in the game and to allow mixed circles later. This strategy clears out as many pure bins as possible early in the process in order to reveal points that might have been obscured earlier. We persist the rectangles so that users can see what they have already drawn. We ask users to think of the whole CHDR as a general shape. *Note:* A CHDR is a collection of rectangles the user creates in one 2D scatterplot capturing one class. While this shape can be disjoint, we try to discourage users from making dust – many tiny rectangles, each covering a point.

The user needs no domain knowledge about the data nor any statistical knowledge to use VisClassifier. She does not even need to know that she is classifying points. The display can be presented to users as a game. The objective is to highlight as many yellows as possible with only a few rectangles per plot. The ultimate score is the classification error from a new sample that is printed when the game concludes. Our game is simpler than Jawbreaker or Minesweeper. This is an important factor in our evaluations. We are more interested in exploring the perceptual and cognitive strategies involved in discrimination. Expertise would complicate that effort.

H. Step 8: Scoring

We transform and rescale a new point using the training parameters. Then we pass through the list of CHDRs. For

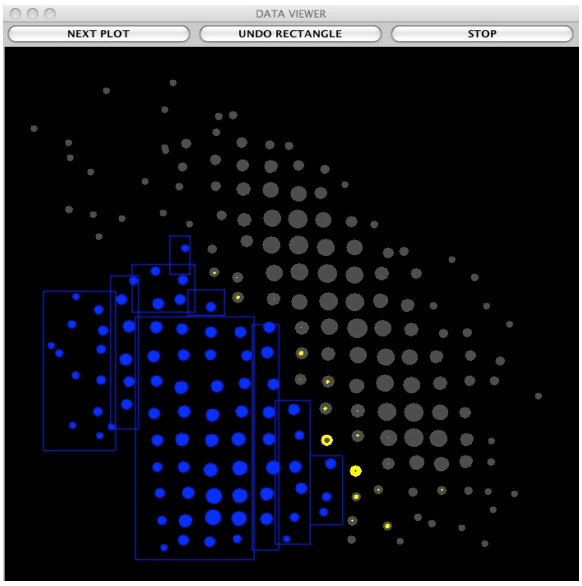

Figure 2. A rendering of the UCI Satellite dataset. The blue rectangles are unioned to form one CHDR that covers the blue points.

each CHDR, we project the point using the stored projections from the training data. Then we pass through the list of rectangles for that CHDR. The first rectangle to enclose our projected testing point determines the classification. If no enclosing rectangle is encountered by the end of the list, we assign the point to the nearest rectangle in the CHDR list by computing the L^∞ distance from the point to a rectangle.

V. Experimental Results

We conducted an informal experiment to evaluate the effectiveness of our design. Seven subjects, 5 male and 2 female, classified four datasets. The subjects had varying backgrounds but all were not familiar with how the algorithm or framework worked. Not all subjects evaluated every dataset but each dataset was assigned randomly to subjects in incomplete blocks. Each subject's classification was based on a new random generator seed, so it was highly unlikely that any two subjects saw the same plot. We found no significant performance differences between subjects.

The datasets were taken from the UCI Machine Learning Repository [24] and are listed in Table 1. Each represents a different challenge for classifiers:

- **Adult**[25] presents a mixture of categorical and continuous variables.
- **Optdigits**[26] is almost Bayes-optimal for the simple Linear Discriminant Analysis model and difficult-to-classify for simple axis-parallel classifiers such as Naive-Bayes.
- **Satellite**[27] has relatively non-normal class distributions that are not well separated.
- **Shuttle** [28] has groups that vary widely in frequency; approximately 80 percent of the data belong to class 1.

Table I
Characteristics of Datasets

	Adult	Optdigits	Satelite	Shuttle
# Training	48842	3823	4435	43500
# Testing	45222	1797	2000	14500
# Attributes	14	64	36	9
# Groups	2	10	6	7
Cat. vars.	Yes	No	No	No
Cont. vars.	Yes	Yes	Yes	Yes

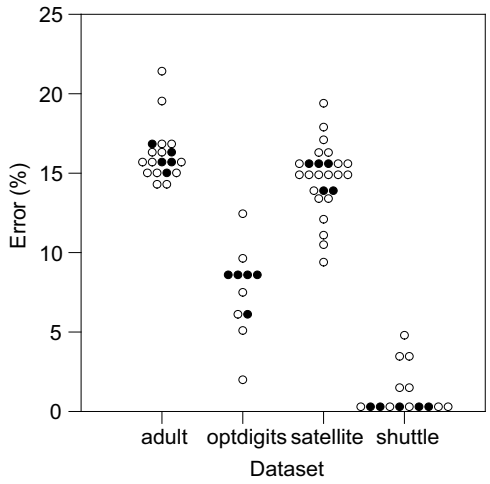

Figure 3. Experimental Results (solid circles = VisClassifier results, hollow circles = Other classifiers)

Fig. 3 shows the dot plot of the error rates of VisClassifier and other classifiers. The competitive classifier performance statistics were culled from papers referenced on the UCI site. They are based on 15 classifiers, including classification trees, kernel classifiers, discriminant analysis, KNNs, naive Bayes and AdaBoost. No classifier is best over all four datasets. VisClassifier is consistently competitive with the other classifiers and marginally better on shuttle.

VI. Discussion: Why Does VisClassifier Work?

It might be asked why we bother developing a classifier that depends on human observers. The purpose of this research project is not to produce a "best of breed" classifier. There is no such classifier, unconditional to a particular process generating the data [29]. Instead, we proposed and implemented a framework to understand how human perception might guide algorithm development.

Like classification trees, VisClassifier partitions local densities using rectangles. However, VisClassifier builds a decision list instead of a decision tree. Recursive partitioning trees are greedy, but VisClassifier is super-greedy. At each step, VisClassifier removes from the training set as many members of class C_k as possible. Once a split has been made, no further splits of that subset are possible. The set-complement of this subset is all that remains to be considered at each step. This strategy reduces the overlap

of densities when projected to a low-dimensional space.

Another aspect that distinguishes VisClassifier is its ability to reveal projected margins in 2D space (like an SVM). There is no convexity assumption within classes. Assuming a revealing 2D projection can be found, VisClassifier can handle a wide variety of class distributions. VisClassifier embeds continuous and categorical variables in the same space. While categorical variables are unordered, continuous-categorical plots look like stripes and categorical-categorical look like a rectangular lattice.

Finally, the method we devised for generating random projections resembles boosting [4]. That is, we select the best of the 1000 random marginal 1D generated projections using the Separation Index. Then, we construct 100 binned scatterplots using these best projections. We pick the best scatterplot to present using a different purity criterion. We chose rather large numbers (1000, 100) for this process, limited by the responsiveness of the system our subjects would tolerate.

VII. Conclusions

We have described a novel mathematical framework for translating human visual data processing interactions into rules defined by CHDRs. We applied this framework to the supervised classification problem and described VisClassifier, our novel classifier algorithm; it is not a hybrid or combination classifier.

Having developed this classifier with humans in the loop, we now intend to remove them. Specifically, we intend to replace the display component with a minimum-rectangular-cover algorithm. If this works and is able to retain the performance statistics, it will be one of the few instances, like projection pursuit [30] and principal curves [31], where visualization provides the foundation for a data mining algorithm.

References

[1] P. C. Wong, "Visual data mining," *IEEE CGA*, vol. 19, pp. 2–3, 1999.

[2] D. Kiem, "Information visualization and visual data mining," *IEEE TVCG*, vol. 7, pp. 100–108, 2002.

[3] S. J. Simoff, M. H. Böhlen, and A. Mazeika, Eds., *Visual Data Mining - Theory, Techniques and Tools for Visual Analytics*, ser. LNCS. Springer, 2008, vol. 4404.

[4] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *EuroCOLT*. Springer-Verlag, 1995, pp. 23–37.

[5] R. Tibshirani and T. Hastie, "Margin trees for high-dimensional classification," *JMLR*, vol. 8, pp. 637–652, 2007.

[6] C. Ding and X. He, "K-means clustering via principal component analysis," in *ICML*. ACM, 2004, p. 29.

[7] S. S. Vempala, *The Random Projection Method*. Providence, RI, USA: American Mathematical Society, 2004.

[8] D. Fradkin and D. Madigan, "Experiments with random projections for machine learning," in *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2003, pp. 517–522.

[9] T. Poggio and S. Smale, "The mathematics of learning: Dealing with data," *Notices of the AMS*, vol. 50, no. 5, pp. 537–544, 2003.

[10] M. Ankerst, C. Elsen, M. Ester, and H.-P. Kriegel, "Visual classification: an interactive approach to decision tree construction," in *KDD*. ACM, 1999, pp. 392–396.

[11] M. Ware, E. Frank, G. Holmes, M. Hall, and I. H. Witten, "Interactive machine learning: letting users build classifiers," *Int. J. Hum.-Comput. Stud.*, vol. 55, pp. 281–292, 2001.

[12] Y. Liu and G. Salvendy, "Interactive visual decision tree classification," in *Graph Drawing*. SpringerVerlag, 2007, vol. 4551, pp. 92–105.

[13] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Wadsworth, 1984.

[14] J. R. Quinlan, *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann, 1993.

[15] S. Bu, L. V. S. Lakshmanan, and R. T. Ng, "MDL summarization with holes," in *VLDB*, 2005, pp. 433–444.

[16] B. J. Gao and M. Ester, "Turning clusters into patterns: Rectangle-based discriminative data description," in *ICDM*. IEEE, 2006, pp. 200–211.

[17] K. Q. Pu and A. O. Mendelzon, "Concise descriptions of subsets of structured sets," *ACM Transactions on Database Systems*, vol. 30, no. 1, pp. 211–248, 2005.

[18] L. Wilkinson, A. Anand, and R. Grossman, "High-dimensional visual analytics: Interactive exploration guided by pairwise views of point distributions," *IEEE TVCG*, vol. 12, no. 6, pp. 1363–1372, 2006.

[19] G. Wills and L. Wilkinson, "Autovis: Automatic visualization," *Information Visualization*, vol. 8, 2009.

[20] D. Achlioptas, "Database-friendly random projections," in *PODS*. ACM, 2001, pp. 274–281.

[21] H. Wainer, "Estimating coefficients in linear models: It don't make no nevermind," *Psychological Bulletin*, vol. 83, no. 2, pp. 213–217, 1976.

[22] T. Hastie and R. Tibshirani, "Discriminant adaptive nearest neighbor classification," *PAMI*, vol. 18, no. 6, pp. 607–616, 1996.

[23] L. Wilkinson, *The Grammar of Graphics*. New York: Springer-Verlag, 1999.

[24] A. Asuncion and D. Newman, "UCI machine learning repository," http://www.ics.uci.edu/~mlearn/MLRepository.html, 2007.

[25] R. Kohavi and B. Becker, "Adult data set," http://archive.ics.uci.edu/ml/datasets/Adult, 1996.

[26] E. Alpaydin and C. Kaynak, "Optical recognition of handwritten digits," http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits, 1998.

[27] A. Srinivasan, "Statlog (landsat satellite) data set," http://archive.ics.uci.edu/ml/datasets/Statlog+(Landsat+Satellite), 1992.

[28] J. Catlett, "Statlog (shuttle) data set," http://archive.ics.uci.edu/ml/datasets/Statlog+(Shuttle), 2002.

[29] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley-Interscience Publication, 2000.

[30] J. H. Friedman and J. W. Tukey, "A projection pursuit algorithm for exploratory data analysis," *IEEE Transactions on Computers*, vol. C-23, pp. 881–890, 1974.

[31] T. Hastie and W. Stuetzle, "Principal curves," *JASA*, vol. 84, pp. 502–516, 1989.