Blind Data Classification using Hyper-Dimensional Convex Polytopes

Brent T. McBride and Gilbert L. Peterson

Department of Electrical and Computer Engineering
Air Force Institute of Technology
2950 Hobson Way
Wright-Patterson AFB, OH 45433-7765

Abstract

A blind classification algorithm is presented that uses hyper-dimensional geometric algorithms to locate a hypothesis, in the form of a convex polytope or hyper-sphere. The convex polytope geometric model provides a well-fitted class representation that does not require training with instances of opposing classes. Further, the classification algorithm creates models for as many training classes of data as are available resulting in a hybrid anomaly/signature-based classifier. A method for handling non-numeric data types is explained. Classification accuracy is enhanced through the introduction of a tolerance metric, ß, which compensates for intra-class disjunctions, statistical outliers, and input noise in the training data. The convex polytope classification algorithm is successfully tested with the voting database from the UCI Machine Learning Repository (Blake and Merz 1998), and compared to a simpler geometric model using hyper-spheres derived from the k-means clustering algorithm. Testing results show the convex polytope model's tighter fit in the attribute space provides superior blind data classification.

Introduction

Many popular data classification methods, such as Artificial Neural Networks, Support Vector Machines, and Decision Trees, are not blind. This indicates that on a decision with two or more classifications, they must be trained upon instances of the various data classifications against which they will be tested. If they are tested against an unfamiliar class instance, the learned hypothesis would not be able to reliably distinguish the foreign instance from the classes of the training set. A blind classification method recognizes that a foreign instance is not a member of any of its training classes and identifies it as an anomaly. Anomaly detection is useful when there is incomplete domain knowledge available for training, as in one side of a two sided classification problem.

This paper presents a blind classification method using hyper-dimensional geometric constructs to create a class model without referencing other classes. Section 2 introduces applicable geometric concepts generalized to arbitrary dimensions. Section 3 describes how class instance data is formatted for use in the geometric classifier. Sections 4 and 5 present two different hyper-geometric construct-based classifiers, one using convex polytopes and the other using hyper-spheres. Section 6 covers the testing methodology used to evaluate both classifiers and section 7 presents the testing results.

Applicable Geometric Concepts

Central to the primary geometric classifier algorithm is the concept of a *polytope*. A *d*-polytope is a closed geometric construct bounded by the intersection of a finite set of hyperplanes, or halfspaces, in *d* dimensions. It is the generalized form of a point, line, polygon, and polyhedron in zero, one, two, and three dimensions, respectively (Coxeter, 1973), but it is also defined for arbitrarily higher dimensions. As the number of dimensions rises, a polytope's structure becomes increasingly complex and unintuitive for the human brain accustomed to modeling objects in the three-dimensional physical world (see Table 1).

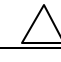
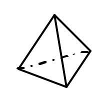
Dimensions	Polytope Name	Example
0	point	•
1	line	—
2	polygon	
3	polyhedron	
4	polychoron	N/A
d	*d*-polytope	N/A

Table 1- Dimensional Progression of Polytopes

A polytope is convex if a line segment between any two points on its boundary lies either within the polytope or on its boundary. A convex hull of a set, *S*, of points in

d dimensions is the smallest convex d-polytope that encloses S (O'Rourke 1998). Each vertex of this enclosing polytope is a point in S. As a two dimensional analogy, imagine that a set of nails (points) is driven into a table or other flat surface (plane). Next, a single rubber band is placed tightly around all of the nails. The polygon outlined by the rubber band is the convex hull of the points marked by the nails (see Figure 1a). The nails touching the rubber band where it bends are the vertices of the polygon.

Each of the polytopes drawn in the third column of Table 1 is a special kind of polytope called a *simplex*. A d-simplex is the simplest (i.e. has the smallest number of vertices, edges, and facets) possible polytope that can exist in d dimensions. A d-simplex is always convex and contains exactly $d+1$ non-coplanar vertices. Thus, a d-dimensional convex hull cannot be computed for fewer than $d+1$ points.

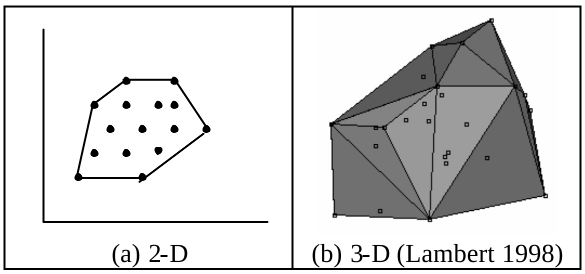


Figure 1- Convex Hulls

The secondary classifier makes use of the generalized circle, or hyper-sphere. A d-sphere is a hyper-sphere in d dimensions that is defined simply by a center point and a radius. This construct is significantly easier to deal with than the convex polytope. This fact has both its advantages and disadvantages, as will be explored in a later section.

Both geometric classifiers evaluate separations between points in d-dimensional space using generalized *Euclidean distance*. The Euclidean distance between any two points in a continuous d-space is derived from the Pythagorean Theorem (Weisstein, 1999) and is given by Equation 1. It is common to compare distances squared to avoid performing a computationally expensive square root calculation.

$$\forall p, q \in \Re^d : dist(p,q) = \sqrt{\sum_{i=1}^{d} (p_i - q_i)^2} \qquad (1)$$

Data Representation

The data on which the hyper-geometric classifier trains and tests is comprised of a set of instances. Each instance belongs to a class and possesses an ordered list of attributes. Each class instance with d attributes is mapped to a real-valued d-vector which represents a point, p, in a continuous d-space. Point p is defined symbolically as:

$$p = \{p_1, p_2, \cdots, p_d\} \in \Re^d$$

The attribute values are mapped to the $p_i \in \Re^1$. Many classification problems involve non-numeric attributes. For example, an attribute describing the weather could contain values such as “sunny”, “cloudy”, or “rainy”. Such non-numeric attribute values may be arbitrarily mapped to real numbers, but the chosen mapping must be consistently applied. This arbitrary mapping leads to equally arbitrary point coordinate assignments and cluster disjunctions within the attribute space. These disjunctions, if not addressed, can negatively affect the classification accuracy of the derived geometric class model. A method to handle these disjunctions is explained in the next section.

Convex Polytope Classifier

Once the training instances for class C have been mapped to a set, T, of d-vectors, a geometric class model is created. If the desired geometric shape is a convex d-polytope, then the convex hull of T is computed:

$$ConvexHull(T) \equiv H \subseteq T$$

For classification purposes, the convex hull of T is represented as a set of vertices, H, which are the vertices of the smallest convex d-polytope that encloses T. Note that this is a blindly-created model because it is derived solely from instances of class C. It knows nothing about the attribute values of other classes.

A distinct test point, p, is declared to be a match (member of class C) iff it is bounded by the polytope defined by H. This is determined by computing:

$$ConvexHull(H \cup \{p\}) \equiv D$$
$$p \notin D \Leftrightarrow match(p,C)$$
$$p \in D \Leftrightarrow \neg match(p,C)$$

If p is not bounded by H's polytope, then D will represent a larger polytope of which p is a vertex. If p is bounded by the polytope, then $D \equiv H$. An alternate method, which handles the special case of $p \in H$ (i.e. p need not be distinct), is to compare the hyper-volumes of the polytopes represented by D and H. If $volume(D) > volume(H)$, then p does not match the model and is declared anomalous with respect to class C.

Spatial disjunctions of data clusters and statistically extreme points caused by input noise or other factors can result in a convex hull model that is too voluminous, enclosing much of the attribute hyper-space that does not rightly define class *C*. This extra space translates to a poorly fitted model that is highly susceptible to declaring false positives.

Convex Hull Tolerance Parameter: *ß*

To compensate for disjunctions and lessen the impact of statistical outliers, a tolerance feature controlled by parameter $0 \leq b \leq 1$ is added. It breaks up training points into groups of smaller convex polytopes and provides a tighter fit around the training data, as follows:

1. Select values *MIN* and *MAX*, such that *MIN* < *MAX*.
2. Scale each dimension of the vertices in *T* between *MIN* and *MAX*. All points in *T* then lie inside of a *d*-hypercube with opposing extreme points at $\{MIN^d\}$ and $\{MAX^d\}$. The distance squared between these two extreme points:
$$dist(\{MIN^d\},\{MAX^d\})^2 = d(MAX - MIN)^2$$
provides the upper bound on the distance squared between any two points in *T*:
$$\forall p,q \in T : dist(p,q)^2 \leq d(MAX - MIN)^2$$
3. Let *G* be an undirected, unweighted graph. For each vertex in *T*, add a corresponding node to *G*. Create an edge between each distinct pair of nodes, *p* and *q*, in *G* where:
$$dist(p,q)^2 \leq b^2 d(MAX - MIN)^2$$
4. Partition *G* into unconnected sets, such that in each set every node is connected by at least one edge and no edges cross between sets. Throw out any set with fewer than d+1 non-coplanar points (the minimum needed to create a simplex in d-space--this is where statistical outliers are discarded).

The multiple convex hulls constructed around the partitioned sets of *G* comprise the model of class *C*. Test point *p* is then scaled by the same factors used on *T* in step 2 and is declared a match iff it resides within any of *C*'s convex polytope models. With *ß*=1, the algorithm behaves as the unmodified version and creates a single convex polytope model. As *ß* decreases, the potential number of smaller polytopes will increase and their combined hyper-volume in the attribute space will decrease. In testing, this leads to a lower probability of false positives at the expense of a higher probability for false negatives, or a loss of generality. At *ß*=0, no convex hull models are created and all test points are rejected. Finding the right *ß* value for each class model to fit the training data properly and achieve a good balance between false positives and negatives requires experimentation.

If instances from multiple classes are available for training, then hyper-geometric models, or signatures, are created for each class. Test instances are compared against each of the models and classified as an anomaly if no matches are found. This classifying algorithm is thus a hybrid signature/anomaly detector. Classification accuracy and specificity increases with the level of training data available.

When the classifier contains signatures for more than one class, model overlapping is possible. This is especially true when poorly discriminating attributes are used or when a model's *ß* value is too high. Multiple class matches may be acceptable in the case of non-mutually exclusive classes (i.e.: a person with dual citizenship). For mutually exclusive classes, the accuracy of overlapping matches may be improved with a tie-breaking protocol, perhaps relying upon the test point's distance from each geometric model's center or nearest boundary.

There are many freely available programs that compute convex hulls in high dimensions. (Avis, Bremner, and Seidel 1997) evaluated the performance of many of the most popular programs, including cdd+, PORTA, qhull, and lrs. The qhull program (Barber and Huhdanpaa 2002), version 2002.1, is used with this convex polytope classifier, in part because it can compute hyper-volumes. Qhull has a time complexity of $O(n^{\lfloor d/2 \rfloor})$, for *n* input points in *d*-space (Barber, Dobkin, and Huhdanpaa 1996).

Hyper-sphere Classifier: k-means

Hyper-spheres may also be used as the geometric class model, in lieu of convex polytopes. Using this paradigm, a class is described by a set, *S*, of *k* hyper-spheres. The k-means clustering algorithm partitions the training set, *T*, into *k* different clusters. Each cluster has a *centroid*, the average of all points in the cluster. The k-means algorithm attempts to minimize the sum of squared within group errors, or the sum of the distance squared between each point and the centroid of its assigned cluster.

k behaves as a tolerance parameter for the hyper-sphere classification algorithm by controlling the partitioning of *T*. The cluster centroids produced by the k-means algorithm become the center points of the *k* hyper-spheres in *S*. The radius of each hyper-sphere is given by the distance between the corresponding centroid and the most distant point in its cluster. Point *p* is bounded by a hyper-sphere with center point *c* and radius *r* iff $dist(p,c) \leq r$. A point is declared a member of class *C* iff it is enclosed by any of the *k* hyper-spheres in *S*.

The hyper-sphere classifier uses the SimpleKMeans program of the WEKA java suite, version 3.2.3 (Witten and Frank 2002). The time complexity of the k-means algorithm is $O(knr)$, for *k* clusters, *n* points, and *r* iterations (Wong, Chen, and Yeh 2000). Testing a point for inclusion in *S*'s *k* hyper-spheres takes $O(kd)$ time.

The obvious advantage the hyper-sphere model has over a convex polytope is that its time complexity is linear, not exponential, in *d*. Thus, a hyper-sphere can create a model with much higher dimensionality than is feasible with a convex polytope. The main advantage the convex polytope paradigm holds is that it can create much tighter-fitting models than are possible using a hyper-sphere—an important requirement for good blind classification, as testing will show.

Testing Methodology

Both the convex polytope and the hyper-sphere geometric classifiers are tested with the voting database from the UCI Machine Learning Repository (Blake and Merz, 1998). It contains the voting records of members of the 1984 US. House of Representatives on 16 key votes. Each instance in the database represents an individual member of congress who belongs to one of two classes: Republican or Democrat. The instance attributes are the individual's choices on the 16 votes. Each attribute has one of three values: "yea" (voted for, paired for, or announced for), "nay" (voted against, paired against, or announced against), and "unknown" (voted present, voted present to avoid a conflict of interest, or did not vote).

The non-numeric attribute values "nay", "unknown", and "yea" are arbitrarily mapped to real numbers -1, 0, and 1, respectively. Due to the dimensional complexity of the convex hull algorithm, the classifiers train and test with only the first seven of the 16 available attributes.

The testing regimen creates a total of 6,400 blind models of the two classes, using the two different hyper-geometric structures and sixteen different values for their applicable tolerance parameters (*ß* or k). These models are tested for matches a total of 147,200 times with test points from both classes.

The results are displayed in the box-whisker graphs of Figure 2 through Figure 5. Each box-whisker structure displays the maximum, 3rd quartile, median, 1st quartile, and minimum match percentages over 100 trials with a given tolerance parameter, training class, and testing class. In each trial, 90% of the training class instances are randomly selected for training. The testing set is comprised of the remaining 10% of the training class (top graphs) plus a random 10% of the non-training class (bottom graphs).

Next, the best *ß* values for both models are used in a test to gauge how overall detection accuracy is affected by the training set size, as a percentage of the whole database. Overall accuracy is defined as average probability of the classifier correctly identifying a test instance. The results are given in Figure 6.

Results Analysis

Strong test results are represented by a high mean match percentage for the top graphs (correct matches) and a low mean percentage on the bottom graphs (false positives). A narrow inner-quartile range (distance between the Q3 and Q1 bars) reflects consistent/stable performance.

As expected, the convex polytope model's tighter fit of the data space produces better results than that achieved with hyper-spheres.

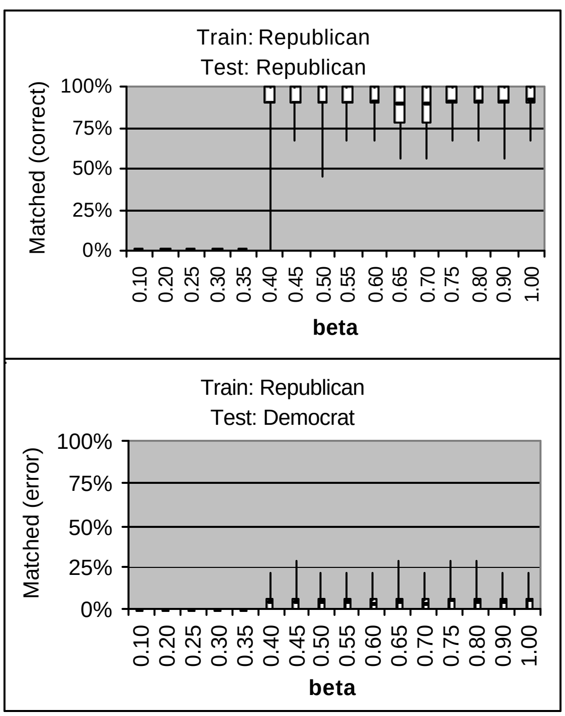


Figure 2 - Republican Model (Convex Polytopes)

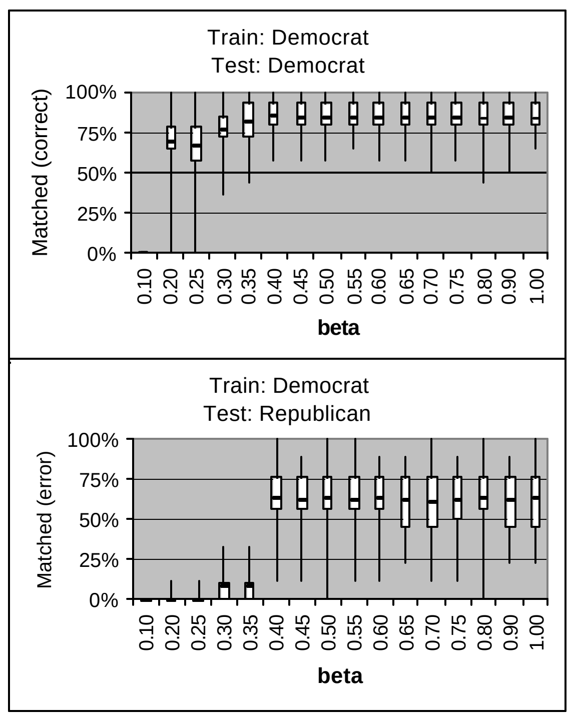


Figure 3 - Democrat Model (Convex Polytopes)

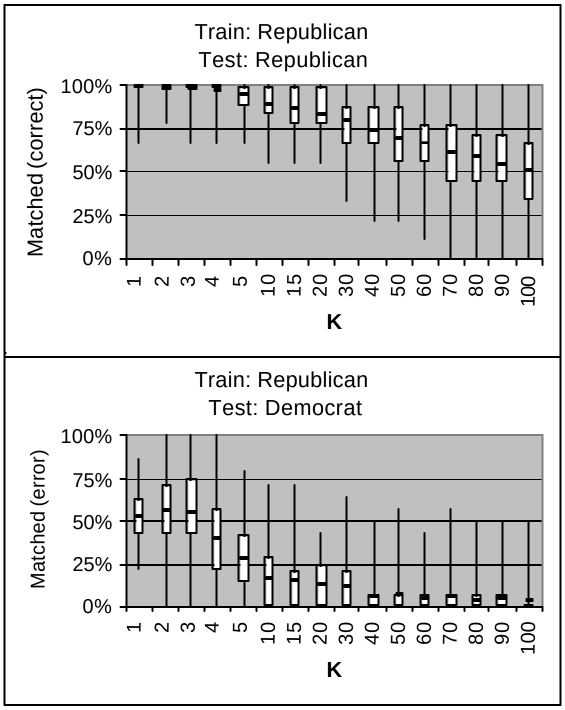


Figure 4- Republican Model (Hyper-Spheres)

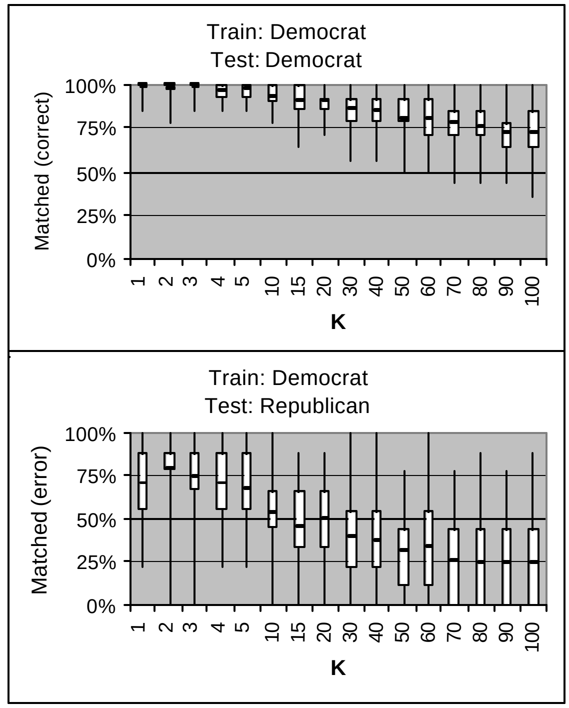


Figure 5- Democrat Model (Hyper-Spheres)

The best ß values for the Republican blind model (Figure 2) range roughly between 0.45 and 1.0. The best ß for the Democrat model (Figure 3) falls at about 0.35. At these ß values both models exhibit good, stable classification accuracy with low incidence of false positive and false negative matching errors. This is especially impressive considering that the classifiers use less than half of the available attributes!

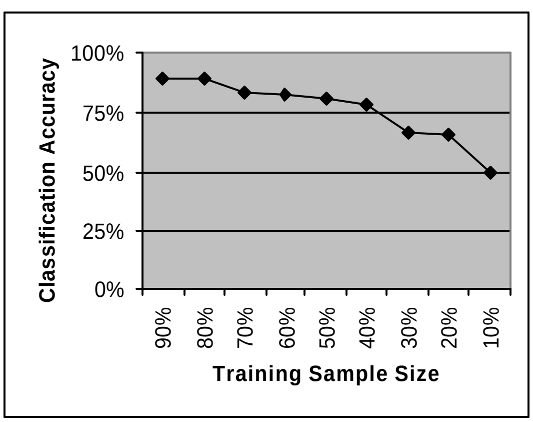


Figure 6 - Training Sample versus Detection Accuracy

The hyper-sphere models (Figure 4 and Figure 5) do not fare so well. Their performance is less stable, as evidenced by their wider bar-whisker graphs. At every value of *k* the models exhibit inferior balancing of false positive and false negative errors.

Figure 6 shows that the overall classification accuracy for the convex polytope model declines only gradually as a smaller percentage of the database is used for training. Note that an overall accuracy of 50% is equivalent to classification via the flip of a coin. The classifier drops to this level when the training sample size is not large enough to form a working convex hull model.

Preliminary testing with a few other UCI Machine Learning Repository databases suggests that the tight fit provided by a convex polytope does not perform as well for training sets requiring more generality. For example, for each of the monks1, monks2, and monks3 databases, which were designed to test induction algorithms, the convex polytope classifier achieved a blind classification accuracy of about 70%. Accuracy was increased to about 80% by applying some domain knowledge in the form of two class models using the previously described tie-breaking protocol. On the other hand, the classifier achieved 99% accuracy on the Iris database.

Since a blind classifier does not, by definition, compare and contrast attribute values between opposing classes, it is at a clear disadvantage for sparse training sets (requiring greater generality), compared to more traditional non-blind classifiers. This is the tradeoff for being able to recognize anomalous classes. The blind hyper-geometric classifier performs best on diverse training sets whose points are well-representative of the class' true attribute space topology. In other words, a good blind class model can only be achieved if the class is well known.

Future Possibilities

The negative impact of the convex hull complexity limitation on the number of dimensions may be lessened by creating multiple classifiers, each using different partitions of the attribute space, and then boosting or bagging to combine the collective results. Alternate implementations of a tolerance feature can be explored to perhaps increase the generality of the hyper-geometric classifier. It may be possible to reduce the hyper-sphere model over-inclusiveness problem by scaling down the radii of the hyper-spheres.

Structures other than convex polytopes and hyper-spheres may be used in a hyper-geometric classifier. A polytope created by the alpha-shapes algorithm may be especially promising. Such a polytope need not be convex, potentially allowing for an even better spatial fit around training data. Further, the algorithm already has a built-in tolerance value in *alpha*. However, the alpha-shapes algorithm suffers from similar dimensional complexity issues as the convex hull algorithm. While an alpha-shape is possible in hyper-dimensional space (Edelsbrunner and Mucke 1994.), the authors have yet to find an alpha-shape tool that can exceed three dimensions.

Conclusion

To be an effective blind classifier, a geometric construct must be versatile enough to create a good fit around a wide variety of training data topologies in the attribute space. Testing results on the voting database show that a hyper-dimensional convex polytope can offer good blind classification, even when dealing with non-numeric attribute values or training on only a portion of the attribute space. Further testing on a wider variety of standardized databases from the UCI Machine Learning Repository and other sources will determine the true range of this method's usefulness.

Acknowledgements

The work on this paper was supported (or partially supported) by the Digital Data Embedding Technologies group of the Air Force Research Laboratory, Rome Research Site, Information Directorate, Rome NY. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation there on. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Air Force Research Laboratory, or the U.S. Government

References

Avis, D; Bremner, D., and Seidel R. 1997. How Good are Convex Hull Algorithms? ACM Symposium on Computational Geometry, 20-28.

Barber, C.B., Dobkin, D.P., and Huhdanpaa, H.T. 1997. The Quickhull Algorithm For Convex Hulls. ACM Trans. on Mathematical Software. 22, 469-483. http://www.acm.org;

Barber, C.B and Huhdanpaa. 2002. Qhull, version 2002.1. http://www.thesa.com/software/qhull/

Blake, C.L. & Merz, C.J. 1998. UCI Repository of Machine Learning Databases. University of California, Department of Information and Computer Science, Irvine, CA.
http://www.ics.uci.edu/~mlearn/MLRepository.html

Coxeter, H. S. M. 1973. Regular Polytopes, 3rd ed. New York: Dover.

Edelsbrunner, H. and Mucke, E. 1994. Three-dimensional Alpha Shapes. ACM Transactions on Graphics, 13(1):43-72.

Lambert, Tim. 1998. Convex Hull Algorithms applet. UNSW School of Computer Science and Engineering
http://www.cse.unsw.edu.au/~lambert/java/3d/hull.html

O'Rourke, J. 1998. Computational Geometry in C, 2nd ed. Cambridge, England: Cambridge University Press.

Weisstein, E. 1999. Distance. Wolfram Research, CRC Press LLC.
http://mathworld.wolfram.com/Distance.html

Witten I.H. and Frank E. 2002. WEKA, version 3.2.3, Java Programs for Machine Learning University of Waikato, Hamilton, New Zealand.
www.cs.waikato.ac.nz

Wong C., Chen C., and Yeh S. 2000. K-Means-Based Fuzzy Classifier Design, The Ninth IEEE International Conference on Fuzzy Systems, vol. 1, pp. 48-52.