SANDIA REPORT
SAND97-0589 • UC-705
Unlimited Release
Printed March 1997

Extending Applicability of Cluster Based Pattern Recognition with Efficient Approximation Techniques

R. F. Martinez, G. C. Osbourn

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Approved for public release; distribution is unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.
NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A04
Microfiche copy: A01

SAND97-0589
Unlimited Release
Printed March 1997

Distribution
Category UC-705

Extending Applicability of Cluster Based Pattern Recognition with Efficient Approximation Techniques

R.F. Martinez and G.C. Osbourn
Vision Science, Pattern Recognition and Multisensor Algorithms Dept.

Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185

Abstract

The fundamental goal of this research has been to improve computational efficiency of the Visually Empirical Region of Influence (VERI)[1] based clustering and pattern recognition (PR) algorithms we developed in previous work. The original clustering algorithm, when applied to data sets with N points, ran in time proportional to N^3 (denoted with the notation O (N^3)), which limited the size of data sets it could find solutions for. Results generated from our original clustering algorithm were superior to commercial clustering packages. These results warranted our efforts to improve the runtimes of our algorithms. This report describes the new algorithms, advances and obstacles met in their development. The report gives qualitative and quantitative analysis of the improved algorithms' performances. With the information in this report, an interested user can determine which algorithm is best for a given problem in clustering (2-D) or PR (K-D), and can estimate how long it will run using the runtime plots of the algorithms before using any software.

Acknowledgment

This work was supported by the United States Department of Energy under Contract DE-AC04-94AL85000.

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy.

Contents

Introduction 7
Reduced Inhibitor/Clustering Set (RICS) Approximation Algorithm 8
Fast $O(N_{test}^2)$ Clustering Algorithm 10
Fast $O(N_{test}*N_{train})$ Exact PR Algorithm 11
Reduced/Sparse Training Set Generation 12
Divide & Conquer Algorithms 16
D&C Clustering Algorithm #1 19
D&C Clustering Algorithm #2 19
D&C Clustering Algorithm #3 25
D&C Clustering Algorithm #4 29
Discrimination Approximation Algorithms 34
Discrimination (PR) Approximation Algorithm #1 37
Discrimination (PR) Approximation Algorithm #2 41
Discrimination (PR) Approximation Algorithm #3 45
Discrimination (PR) Approximation Algorithm #4 49
Discrimination (PR) Approximation Algorithm #5 54
Conclusions 63
References 64

Figures

1 6-D Data NN Benchmarking Results 9
2 Cluster Results Using NN inhibitors/groupers -- Fixed Average 9
3 Original (Unreduced) Training Set 13
4 Reduced Training Set -- One Iteration 14
5 Tradeoff of Runtimes and PR Results -- Sparse Training Sets 15
6 Software Generated Cluster Patterns 18
7 Two Partitioning Steps of D&C Clustering and Final Result 20
8 Cluster Algo #2 Runtime Plot 21
9 Cluster Algo #2 ACS Sum Characteristics 22
10 Cluster Algo #2 Runtimes -- Vary Partition Size 23
11 Cluster Algo #2 Qualitative Results -- 3 Partition Sizes 24
12 Cluster Algo #3 Runtime Plot 26
13 Cluster Algo #3 Runtimes -- Vary Partition Size 27
14 Cluster Algo #3 Qualitative Results -- 3 Partition Sizes 28
15 Cluster Algo #4 Runtime Plot 30
16 Cluster Algo #4 Runtimes -- Vary Partition Size 31
17 Cluster Algo #4 Qualitative Results -- 3 Partition Sizes 32
18(a) PR schematic 35
18(b) Test Point Projection -- Test For More Groupers 35
18(c) Grouped Point Projection -- Test For More Inhibitors 35

Figures (cont)

19 Discrimination Algo #1 Runtime Plot 38
20 Discrimination Algo #1 Qualitative Results, 0.1% Training Data 39
21 Discrimination Algo #1 Qualitative Results, 16% Training Data 40
22 Discrimination Algo #2 Runtime Plot 41
23 Discrimination Algo #2 Qualitative Results, 0.1% Training Data 43
24 Discrimination Algo #2 Qualitative Results, 16% Training Data 44
25 Discrimination Algo #3 Runtime Plot 46
26 Discrimination Algo #3 Qualitative Results, 0.1% Training Data 47
27 Discrimination Algo #3 Qualitative Results, 16% Training Data 48
28 Discrimination Algo #4 Runtime Plot 50
29 Discrimination Algo #4 Qualitative Results, 0.1% Training Data, 6,3 NN 51
30 Discrimination Algo #4 Qualitative Results, 16% Training Data, 6,3 NN 52
31 Analysis of Discrimination Algorithm Runtime 53
32 Discrimination Algo #5 Runtime Plot, 6,3 NN 54
33 Discrimination Algo #5 Qualitative Results, 0.1% Training Data, 6,3 NN 56
34 Discrimination Algo #5 Qualitative Results, 16% Training Data, 6,3 NN 57
35 Discrimination Algo #5 Runtime Plot 6,6 NN 58
36 Discrimination Algo #4 Qualitative Results, 0.1% Training Data, 6,6 NN 60
37 Discrimination Algo #4 Qualitative Results, 16% Training Data, 6,6 NN 61

Extending Applicability of Cluster Based Pattern Recognition with Efficient Approximation Techniques

1. Introduction

Clustering and PR algorithms are not new algorithms. One purpose of a clustering algorithm is to determine (or impose) an order on an unknown set of points in a data set. After a data set has been clustered, and depending on the quality of the clustering result, the data will be ordered into clusters. The clustering algorithm used determines how or why the points in a cluster are related to each other. In a PR algorithm we utilize a training set of known information, and a test set with unknown classes of points. The training set contains experimental examples of each class to be identified, and these class data are used to develop criteria for automatically identifying future examples (test points) of these classes. For each of the test points we want to examine where it spatially occurs in the classified training points and assign a class to the test point based on the class criteria established.

As development of improved sensors continues, the sensor community has a greater need for effective classification algorithms to analyze and distinguish sensor input signals. The VERI based algorithms (1) we initially developed offer useful performance advantages compared to other algorithms available in the classification community. However, the original VERI algorithms required computational times that were too long for problems with relatively large data sets. The fundamental goal of this research has been to improve computational efficiency of the VERI based clustering and pattern recognition (PR) algorithms, so that the performance advantages can be made available to a much larger set of problems. This report describes the progress and achievements made in improving the runtimes of our algorithms.

A chronological ordering of the work done to develop new algorithms is presented. Section two describes our initial work to develop fast approximation algorithms using only a subset of the data to compute results. From the initial approximation work we developed faster exact clustering and PR algorithms. Section three describes a technique to reduce the size of a training data set and still have it perform similarly to the larger set for PR applications. An algorithm to perform this function is developed and results are illustrated. Section four presents four different clustering approximation algorithms with their respective qualitative and quantitative results from operating on a set of cluster patterns. Section five presents PR approximation algorithms we developed and also shows results from classifying a series of data sets. Section 6 gives our conclusions and recommendations about the algorithms. We assume that the reader is familiar with the VERI algorithm described in Ref. 1, and definitions of the terminology associated with the algorithm (groupers, inhibitors, VERI template, etc.) can be found there.

2. Reduced Inhibitor/Clustering Set Approximation Algorithm

2.1 Motivation to Improve Algorithm Runtimes

The initial clustering algorithm we developed in previous work used a straight forward approach of examining all ($O(N^2)$ pairs) point pair combinations to determine if they cluster (group) together or are inhibited by any third point which falls within the VERI template (1). The algorithm is an $O(N^3)$ algorithm, which would severely limit the size of data sets which could be operated on in a practical amount of time.

The number of potential pairings examined for each point, and the number of points used as inhibitor candidates will determine how much time is required to compute the groupings for each point in the data set. If the number of grouping candidates is fixed to be a number less than the total number of points in the test set, and the total number of points examined as inhibitor candidates is also a fixed number, then the total number of points examined for inhibitors/clustering is greatly reduced from the number the exact algorithm would have examined. We call an algorithm which uses reduced inhibitor/clustering sets (RICS), a RICS algorithm. The key issue for such an approach is how much the RICS results will differ from the exact VERI calculation as fewer grouper/inhibitor candidate points are examined.

2.2 Reduced Inhibitor/Clustering Set Clustering Algorithm Description

We developed a RICS version of our clustering algorithm. Based on the nature of VERI, nearest neighbor (NN) points (using the Euclidean metric) of test and training points serve well as clustering and inhibiting candidates. The RICS clustering algorithm uses a fixed number of NN's of a test point as grouping candidates for each point in the data set. A fixed number of NN's of the point pair examined for grouping serve as inhibitor candidates.

We examined the RICS clustering approximation algorithm's performance as a function of the number of NN's used as inhibitor/clustering points. Several data sets over a range of dimensions were used to test the RICS algorithm. Data sets of two, three, six and ten dimensions, with homogeneous, random and dissimilar density clusters were tested, each with a range of separations between the clusters in the patterns and with varying number of NN's used as the inhibitor/clustering set. Examples of data used are clusters patterns bounded in a unit (hyper) cube with (a) 85 points in each cluster, (b) 225 points in each cluster, (c)500 points in each cluster and (d) 900 points and 100 points in each cluster. The qualitative results using the RICS algorithm with these test patterns were benchmarked using our exact clustering algorithm.

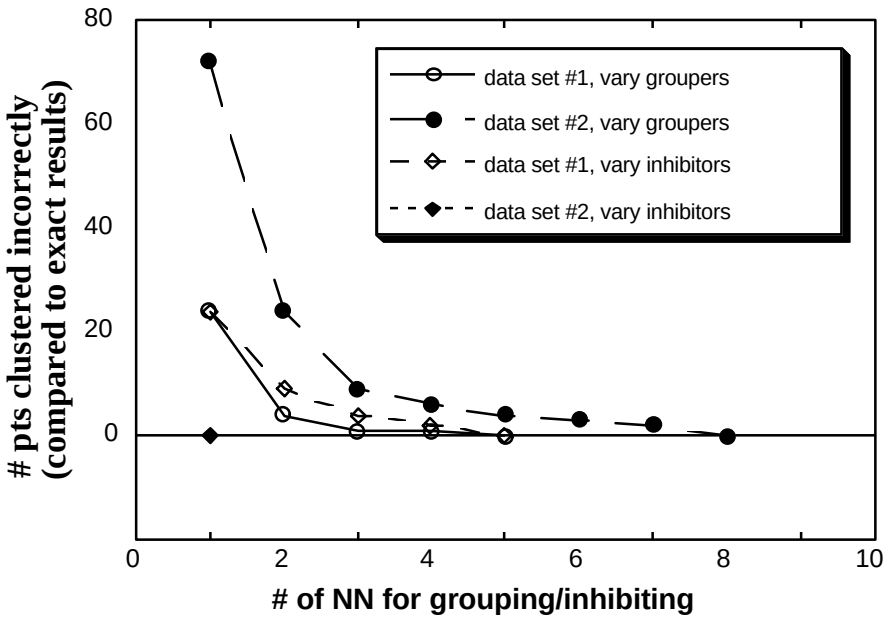

Fig. 1 6-D Data NN Benchmarking Results

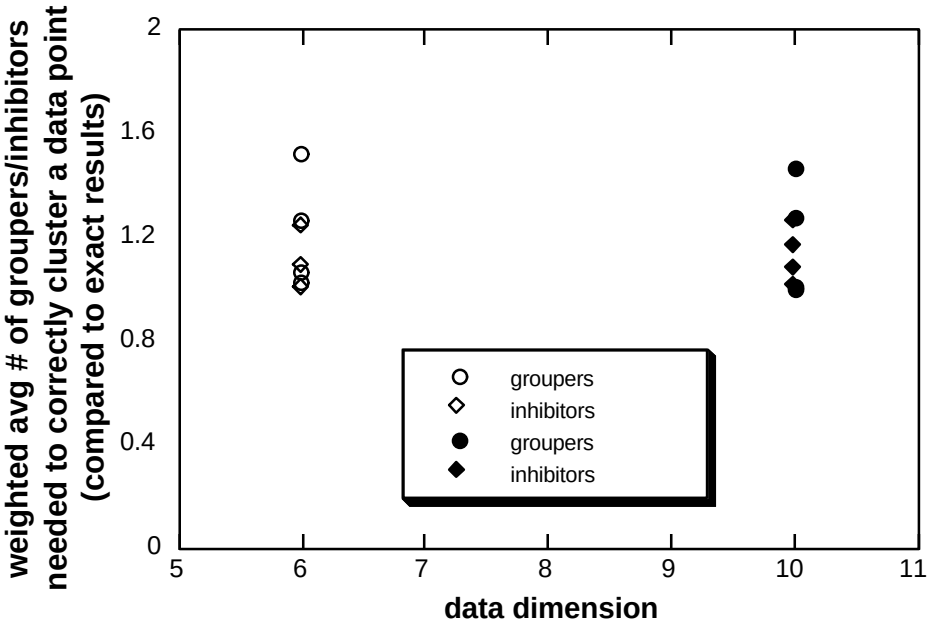

Fig. 2 Cluster Results Using NN inhibitors/groupers -- Fixed Average

In figure 1, RICS results for two 6-D data sets are illustrated. These data sets both had two clusters of points in a unit hyper-cube, with 225 random points in each cluster. The separation between the clusters was different for the two data sets. The clustering results from varying the number of inhibitor and grouping candidates are compared with the exact clustering results produced for these data sets. The figure shows the number of points which clustered incorrectly as a function of the number of NN points used as groupers/inhibitors in a RICS clustering approximation algorithm. When two or more NN's are used as the grouper/inhibitor candidates, the number of clustering errors tends to remain constant and few. Fig. 2 shows an example of the weighted average number of groupers and inhibitors needed to achieve the PR results of the exact method. The weighting was computed by multiplying the number of grouper/inhibitor candidates used to find groupings times the percentage of new correct groupings found with that number of grouper/inhibitor candidates. The four 6-D data sets each had two clusters of points in a unit hyper-cube, with 225 random points in each cluster. The separation between the clusters was varied in the data sets. Four 10-D data sets were used to demonstrate RICS results in multiple hyper dimension cases. The 10-D data sets each had two clusters of random points in a unit hyper-cube. Two data sets had 500 points in each cluster, and two had 900 and 100 points in each cluster respectively. The separation between the clusters with the same number of points in each cluster was varied in the data sets. The fixed weighted average result illustrated led to a surprising (favorable) discovery; while varying the dimensions of the data, the average number of NN's points needed for a point to achieve the results of the exact PR method remained constant. These results suggested the possibility of an exact clustering procedure which might use less than O(N) grouping candidates and/or less than O(N) inhibitor tests for each potential grouping. This led to our development of a faster exact clustering algorithm $O(N_{test}^2)$ (sec. 2.3) and an $O(N_{test}*N_{train})$ (sec. 2.4) exact PR algorithm.

2.3 Fast $O(N_{test}^2)$ Clustering Algorithm

This algorithm was developed based on the constant NN findings described in section 2.2. Each of the N data points on which the clustering process will be performed finds a fixed number of its NN's in the data set. All pairs of points are examined for grouping. The NN's of a grouping pair are used as inhibitor candidates. Many groupings are inhibited by the NN's. The groupings which occur are only candidate groupings at this stage as some essential inhibitor points may not have been examined. The candidate groupings are screened using all remaining unique test points as inhibitor candidates, yielding an exact clustering solution in $O(N^2)$ time.

The method uses a fixed number of NN's as inhibitor candidates for each initial grouping attempt (O(1)) to test inhibitors). There are $O(N^2)$ initial groupings attempts done requiring $O(N^2) * O(1) = O(N^2)$ time to perform . Each of the N points will group with at most a fixed number (K) points K<< N. Proof of the fixed number of groupings is explained in G.C. Osbourn (Ref. 2). Therefore, K*N groupings are screened for exactness

using N -2 inhibitor candidates (K*N* N-2 operations to screen), which still leaves us with an $O(N^2)$ algorithm. If we use a simple ($O(N^2)$ technique to find the NN's, and we do it on-line, we still do not change the order of the algorithms runtime.

2.4 Fast $O(N_{test}*N_{train})$ Exact PR Algorithm

In the PR algorithm we utilize a training set of known information (a data set with N_{train} classified points), and a test set with unknown classes of points (N_{test} points with all points unclassified). For each of the N_{test} points we want to examine which training points group with each test point, and assign a class (or classes) to the test point based on the classes of the training points that grouped with it. Based on the NN findings described in section 2.2, we developed a faster version of the exact PR algorithm. Similar to the clustering algorithm in section 2.3, the PR algorithm incorporates the technique of using a fixed number of NN's of a point pair (in this case a test and training point) as inhibitor candidates to create a set of candidate groupings which are then screened for exactness.

For each of the N_{test} test points, the algorithm tries to group (cluster) with the N_{train} training points one at a time. The algorithm uses a fixed number of NN's of the training point and a fixed number of training points, which are NN's of the test point, as inhibitor candidates to approximate the exact groupings. This results in N_{train}*c work per test point, and $O(N_{test}* N_{train})$ work to approximate the exact groupings for all test points. Using Proof of Fixed Number of Groupings Per Point (Ref. 2), each test point will group with at most K training points. $K << N_{train.}$ The approximate groupings are screened with all N_{train} - 1 unique training points to get exact results (which requires $K*N_{test}*N_{train}-1$ work) in $O(N_{test}*N_{train})$ time.

2.5 General Observations in Exact Algorithm Development

Our RICS algorithm research led to an important discovery. Using the RICS results, we developed an exact clustering algorithm which runs in $O(N^2)$ time, compared to the original $O(N^3)$ runtimes. A faster exact PR algorithm was also developed. Its runtime was improved from $O(N_{test}*N_{train}^2)$ to $O(N_{test}*N_{train})$. The development of these algorithms demonstrated to us that indeed faster runtimes were possible without using approximated solutions.

3. Reduced/Sparse Training Sets

3.1 Reduced/Sparse Training Set Motivation

As mentioned above, using the NN information from the RICS algorithm we developed an exact PR which has its runtime reduced by a factor of N_{train} compared to the previous exact PR algorithm. The training set (the data comprising N_{train}) represents the known (classified) information in a PR system. We wanted to know if it was possible to reduce the number of data points in the training set and still retain a PR system capable of producing results with minimum degradation in the test point classification process. A smaller training set which still retains the PR capabilities of the original training set will allow further reductions in the runtimes of the exact $O(N_{test}*N_{train})$ algorithm. What we wanted was an algorithm capable of producing a new training set, based on an existing one, which would be a less dense set of points with expanded NN separations. This sparse set must retain the original training set's geometrical and internal structure.

3.2 Reduced/Sparse Training Set Algorithm

We developed a VERI based sparse training set algorithm. The sparse training set is obtained by using VERI to select an appropriate subset of the original training set. A sparse training data set is built up one point at a time on a class by class basis. The algorithm decides on whether to include one of the original training set points into the new sparse set by attempting to group it with the existing points in the sparse set. A sparse set of points for a class is initialized by selecting two seed points from the class in the original training set. Controlling the separation between the seed points helps determine where points will be added to the sparse set for a class of training points. Points from the original training set which are near the seed points will not be included in the sparse set. The metric we found useful for determining the seed points is to select the first seed point at random from the training set and then select the second point as the one which is closest to a user specified Euclidean distance from the first seed point. A factor of the average NN distance of the points in a class serves as an appropriate distance to specify to select the second seed point. This value is based on a fundamental characteristic of a set of data points and gives the user some control of where the seed points will occur. For example, if a large factor of the NN distance is specified as the distance at which the second point must occur, the second seed point will be the point which is furthest from the first seed point.

Points from the original class are used as inhibitor candidates if they are farther than a user specified distance from the point pair being examined for grouping. By controlling the distance which an inhibitor training must be separated from the grouping points, we can gain some control on how sparse the points will be in the final sparse set created for a class. The further the specified separation an inhibitor must have from the grouping pair, the easier it is for the grouping pair to successfully group, thereby preventing that point from being added to the sparse set. Points are added to the sparse set being created for a class if they do not group (cluster) with any of the existing points in the sparse set. These points which are added are called outliers. For each of the outlier points added, an area (volume) of the original unreduced training set is represented without redundancy. The sparse set for a class is completed when all points in the original class have been examined as candidates for the sparse set.

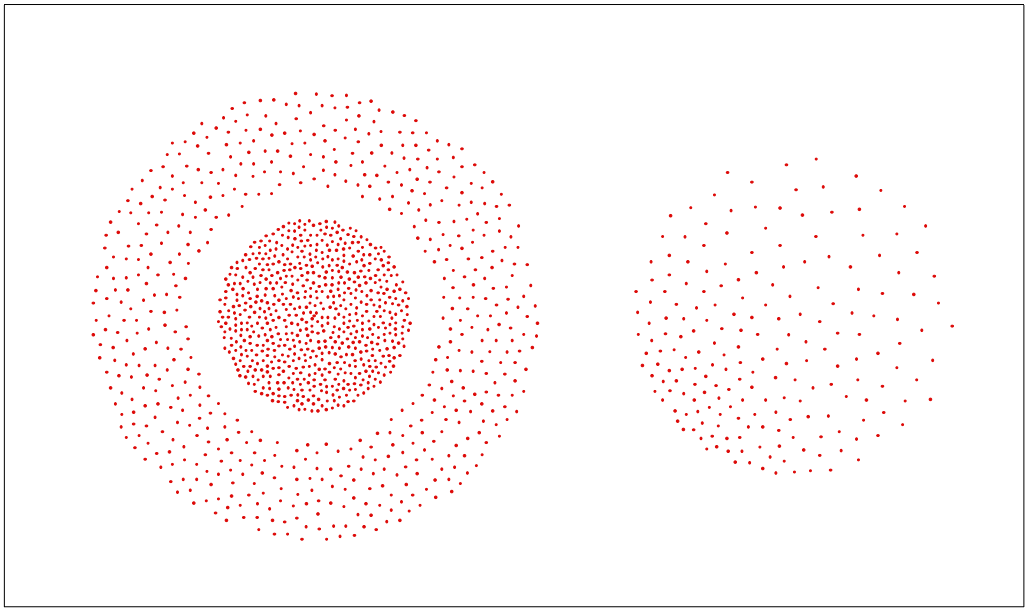

Fig. 3 Original (Unreduced) Training Set

3.3 Reduced Training Set Algorithm Results

We successfully implemented and evaluated an algorithm for producing sparse training sets from K-D data. Our algorithm allows direct improvement of runtimes by reducing N_{train} while controlling the associated degradation of the PR results. Figures 3 and 4 illustrate an example of reducing a training set's size using the Sparse Training Set Algorithm. Fig. 3 is the original training set with N_{train} =1438 points. It consists of three clusters. The clusters on the left are a cluster within a cluster (hierarchical clusters), each of uniform density, with the inner cluster having a dissimilar density with respect to the outer cluster. The clusters on the left side of the figure are well separated from the cluster

on the right side, which is a circular cluster with non-uniform density. Fig. 4 shows the sparse training set generated from the original training set after one iteration of the reducing algorithm, here N_{train} =409 with the geometrical and internal structure of the clusters similar to the original training set. In general, a sparse result can be used as input to the algorithm to produce a sparser training set.

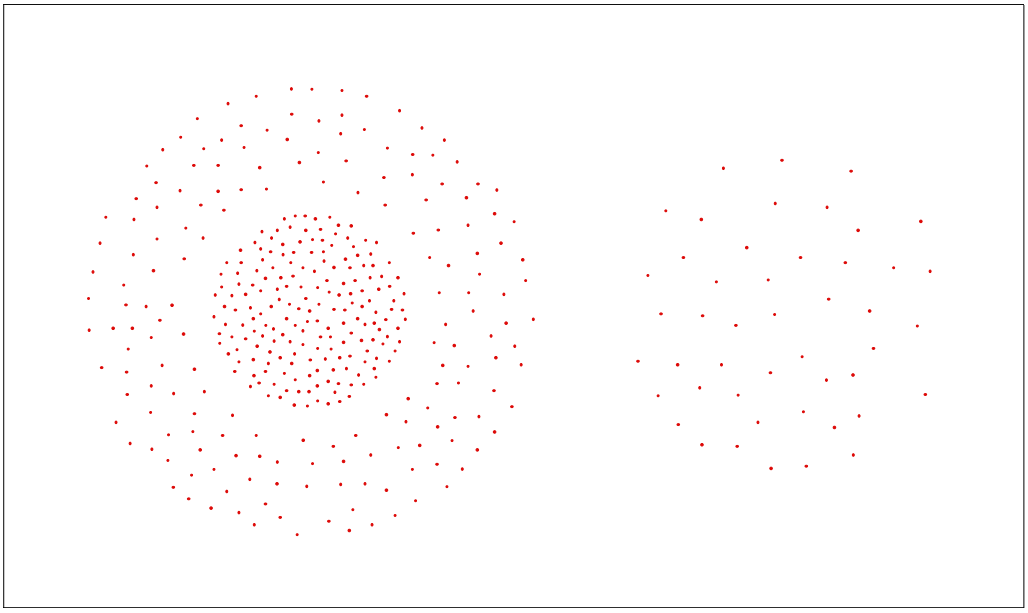

Fig. 4 Reduced Training Set -- One Iteration

3.4 Algorithm Runtime Analysis

The K-D algorithm produces a subset of N_{train} points in its solution. The original class has $O(N_{train})$ points in it, therefore $O(N_{train})$ inhibitor tests are performed each time a point pair grouping with a point from the original class and a point in the sparse set is evaluated. Initially, grouping pairs are examined with the candidate point for the sparse set and the 2 seed points. If the point is added to the sparse set, 3 grouping pairs are examined by the next candidate, and so on until there are $O(N_{train})$ points in the final sparse class. From this we have $2+3+4+...+ O(N_{train}) = N_{train} * (N_{train}+1)/2 -1$, which is $O(N_{train}^2)$ grouping pairs examined. There are $O(N_{train})$ inhibitor checks for each point added to a sparse class. Therefore, the sparse training set algorithm runs in $O(N_{train}^3)$ time. While this is not a favorable dependence on N_{train}, we note that this algorithm is used as a preprocessing step for the PR algorithm, and can be run off line on large computers.

3.5 Reduced/Sparse Training Set Application and Observations

PR runtimes for a large, 6-dimensional training set of chemical sensor data were successfully minimized by reducing the size of the training sets. The data consists of responses of 6 chemical sensors to 29 chemical classes -- individual and binary mixtures of varying concentrations. The 0.1% training set represents responses from new sensors, while the 16% training set represents aging sensors with 16% range of reduction in sensitivity.

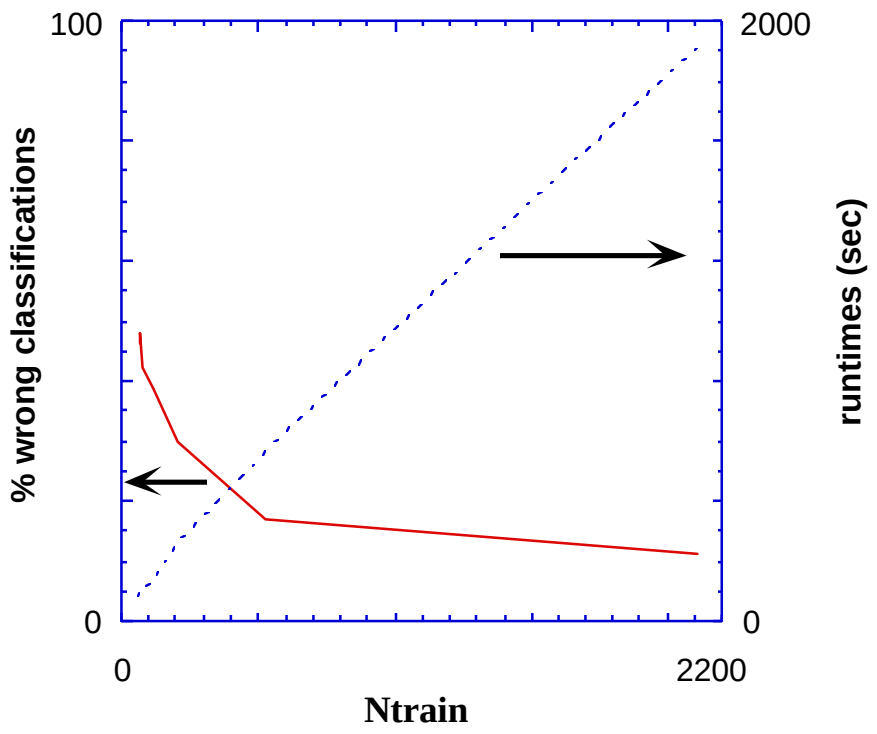

Fig. 5 Tradeoff of Runtimes and PR Results -- Sparse Training Sets

Fig. 5 illustrates the tradeoff of runtimes and computed degradation in chemical identifications for the chemical sensor data as a function of the number of training points. Since this initial work, the Reduced/Sparse Training Set Algorithm has become an integral part of our PR work when a new training set is developed. An area for future work would be the improvement of the $O(N^3)$ runtimes of this algorithm.

4. Divide & Conquer Clustering Algorithms

4.1 Divide & Conquer Clustering

An approximate clustering solution can be useful if it can be obtained fast and the qualitative results do not differ much from the exact solution. We have investigated 2-D clustering approximation algorithms using the divide and conquer (D&C) method (Ref. 3) of breaking a data set into subsets with fixed numbers (m) of points, independent of the total number of points in the data set, and applying the clustering algorithm to data points in the rectangles. Some of the algorithms use a recursive (Ref. 3) method of breaking a data set into subsets. Each points' clustering solution is found by examining the fixed number of points in one of the subsets as potential groupers or inhibitors. The complete data set is no longer analyzed to find a test points clustering solution. However, not all of the points a data point will group with, or points that may inhibit groupings occur in the same subset. It is possible for inter-subset inhibitors or groupers to exist. An iterative technique is used to find the inter-subset groupers and inhibitors that may exist. When all grouping/inhibiting candidates have been found, a final clustering pass is performed to screen the initial groupings.

More specifically, the D&C technique partitions data into rectangles of approximately m (constant) points which are spatially collocated. The time to partition the data can be O(N*log N) if efficient sorting techniques are used. $O(m^2)$ VERI based clustering is performed on each point pair inside each of the rectangles to find a set of initial grouping candidates for each point. Point numbers of successful groupings are added to a grouper/inhibitor list for the data points which successfully group. A fixed number of NN's of each point are found and are also added to the test point grouper/inhibitor lists.

Not all of the points a data point will group with, or points that may inhibit groupings occur in the same rectangle. It is possible for inhibitors or groupers to exist in adjacent rectangles. In order to not examine all adjacent rectangles when performing groupings for a point, the VERI technique is used to determine if groupings may exist between points across rectangle boundaries. A point is projected to its rectangle boundaries and the projected point is used as a grouping candidate. A fixed number of NN's from the original point are used as inhibitor candidates. If the grouping between the original point and the projected point is inhibited, it is not likely other grouping candidates exist in the adjacent rectangle; otherwise, a projected-to point grouping requires the original point be flagged as one with the possibility of finding more grouping/inhibitor candidates in adjacent rectangles. Another iteration of D&C and clustering is performed with the flagged points. We call the points which may have more grouper/inhibitor candidates across rectangle boundaries Active Candidate Seekers (ACS). The D&C and clustering processes continues while the chance of finding more groupers/inhibitors for ACS exists -- i.e. while ACS exist. Each iteration of the algorithm was expected to produce a different set of rectangle boundaries to project to, eliminating some of the ACS. The D&C and clustering

work would be complete when there are no longer any ACS or a small fixed number remain. However, we discovered that it is possible in some data sets for the identical rectangles and ACS to be produced in successive iterations, so that the algorithm fails to terminate. The versions of clustering algorithms described in this paper differ mainly in the approaches to attempt to ensure that the algorithm terminates and in the way the ACS is selected.

After the grouper/inhibitor candidates are found for all points in the data set, a final, more precise clustering pass is done for each point using all the grouper/inhibitor candidate points found for the point. Each test point (I) tries to group with each point (J) in its grouper/inhibitor list that was created, the remaining unique points (K) in I's and J's lists are inhibitor candidates for the grouping attempt.

There are three types of clustering results which can occur for a test point, two of which are errors. A point can correctly group (based on VERI) with its neighbors in the data set. This is the result we get using the exact clustering algorithm. Using approximation algorithms allows the two error conditions to occur. A point in a clustering approximation can group with too many points (excess grouping) or with too few points (missed grouping), relative to the exact clustering results.

Each iteration of the D&C portion of the algorithm requires at most O(N*log N) time to partition the O(N) ACS set. Assuming that a fixed maximum number of iterations occur, the clustering and projection tests require constant time to perform in each of the rectangles. There are at most N/m (i.e. O(N)) rectangles in each iteration, so all of the work after each partitioning is O(N). Therefore, this algorithm should run in O(N*log N) time for sufficiently large N. This is an interesting result; given a large enough data set, it shows the time required to cluster the data is actually bounded by the time required to D&C the data set.

The D&C and clustering techniques applied to clustering algorithms are described in the following sections. Referring to figures in a following section, we see that figures 7(a) and 7(b) illustrate what a 2-D test set looks like after the D&C algorithm has been used to partition a data set. The rectangles bounding the data represent how the algorithm partitioned the ACS.

4.2 Data Set Generator Algorithm

We needed to develop a variety of data sets to fully test the performances of the D&C algorithms. A procedure was developed to generate data sets which contained various clusters, which we do not describe here. We could define the shape of a cluster (from a fixed set of shapes), the density of points in a cluster, and the cluster separations. We found the data generator could require a large amount (in the order of minutes) of time to find a new point in a dense cluster. A limit had to be imposed on the amount of time spent searching for new points, and this produced data sets with incomplete clusters. Since we were primarily concerned with the numbers of points in the data and not the details of the

cluster shapes, this caused no difficulty. The data set generator algorithm created the cluster patterns used to evaluate our clustering approximation algorithms described here in section 4.

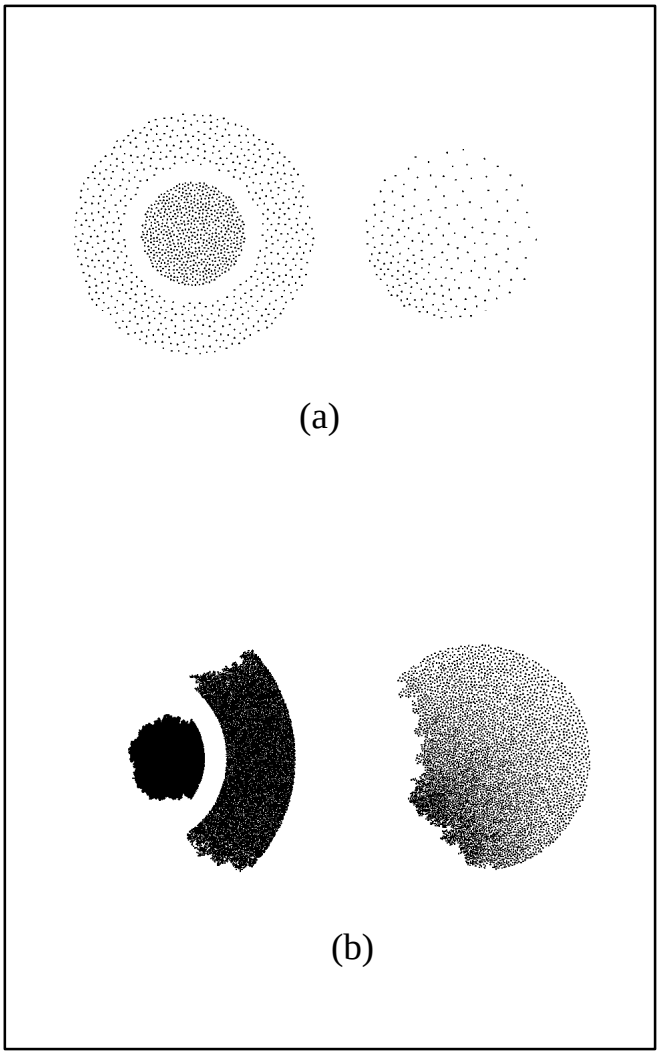

Fig. 6 Software Generated Cluster Patterns

Figure 6 illustrates two of the data sets generated to use in evaluating our clustering algorithms. Figure 6(b) (N = 32625) illustrates how the resulting patterns generated were sometimes not the complete cluster shapes we had specified. However, as discussed above, they were still useful for our testing purposes. The test sets were evaluated using the exact clustering algorithm. The exact clustering results were used to qualitatively analyze the approximation results. The total number of exact groupings for the test set with N = 1438 is 3115, N = 17192 is 34968, N = 32625 is 49923, and N = 105656 is 95723. An exact clustering solution was never generated for the test set with N = 152694.

Even with the $O(N^2)$ algorithm it would have required a prohibitive amount of computing time (~10^6 seconds, see fig. 8). However, it was useful for computing runtimes for the approximation algorithms.

4.3 D&C Clustering Algorithm #1

Our initial D&C clustering algorithm uses a straight forward method to partition the data. For each iteration of partitioning the ACS set into rectangles, the dimension of the data with the largest statistical variance was selected as the dimension to partition the data along. This was a nonrecursive approach, once the dimension was selected, the partitioning consisted of selecting subsets of m points from the ACS set as the points in the rectangles. A fixed number of points were in each partition rectangle along the dimension in which the data was cut.

4.3.1 Clustering Approximation Results

We soon learned the straight forward approach of partitioning data only along one axis for each iteration of the D&C code was not without drawbacks. Analyzing the results, we found that subsequent iterations might produce the same partitions and thus the same ACS set. This can prevent the ACS set from decreasing in size each iteration. Many cases were found where the ACS does not converge to a fixed number of points to stop the D&C iterations. This algorithm was a starting point for the development of several improved versions which are documented below.

4.4 D&C Clustering Algorithm #2

Our next D&C algorithm is an improvement over clustering algorithm #1. A method which would more effectively remove ACS from each iteration was developed. The algorithm uses an improved method to determine which axis the data will be partitioned on. For each iteration a recursive algorithm is used to partition the data. The data is recursively partitioned by selecting the dimension with the largest statistical variance as the one on which to partition. Initially in an iteration, the partitioning dimension is determined from examining all the ACS, then half of the ACS, then one fourth... until a fixed number of points still remain to be partitioned. The resulting recursive partitioning of the ACS set at each iteration is a data set which has been cut along all (2) dimensions. The remainder of the algorithm performs the same as described for D&C Clustering in section 4.

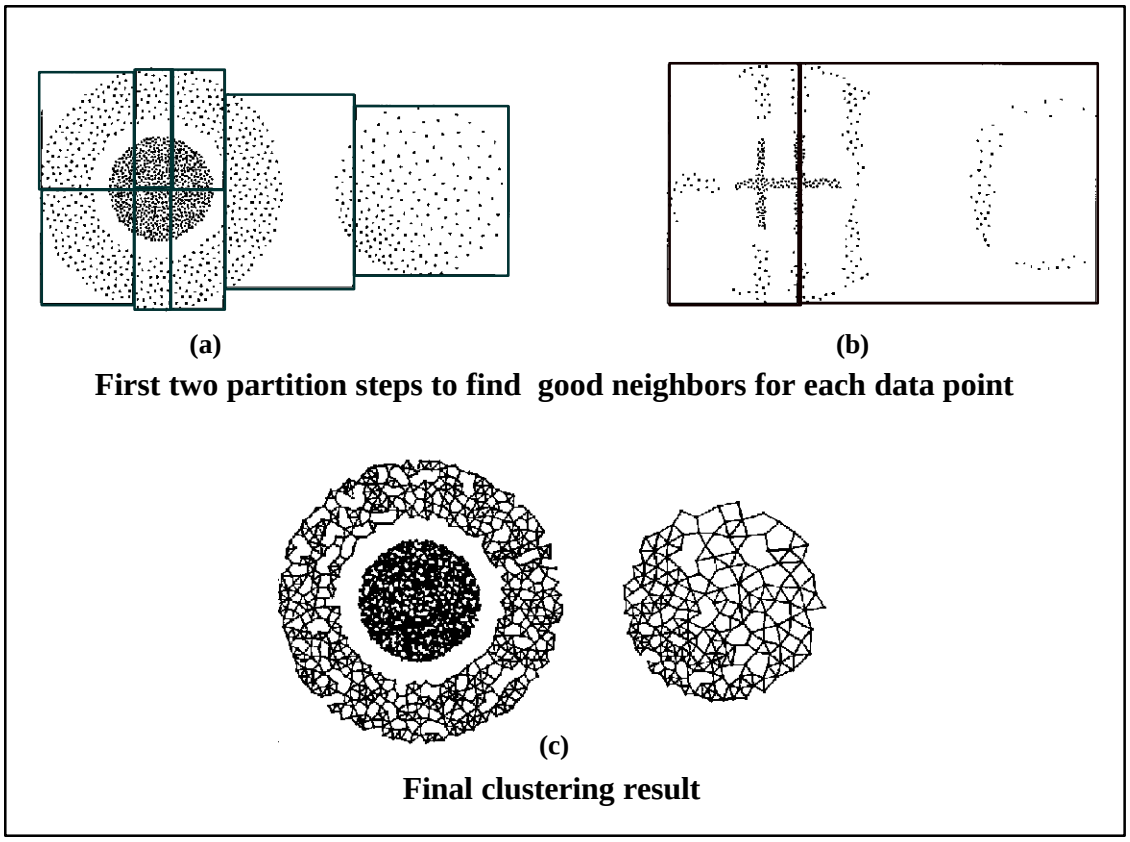

Fig. 7 Two Partitioning Steps of D&C Clustering and Final Result

Fig. 7 shows an example of two recursive partitionings of a data set (ACS) where preliminary groupings and searches for NN's of the test points are performed. The partitioning was performed using *D&C Clustering Algorithm #2.* The first partitioning iteration on the test set (7a) illustrates how the data is partitioned along both dimensions with approximately the same number of points in each rectangle. The second iteration (7b) illustrates how many points remain in this data set as ACS and which points these are. The ACS are points spatially located near the rectangle boundaries created from the previous partitioning iteration. Some points far from the inner rectangle boundaries in (7b) exist because they grouped with points which were projected to external rectangle boundaries -- boundaries which are not adjacent to other rectangles. The succeeding versions of the clustering algorithm did not perform a projected point grouping test to external rectangle boundaries. The final clustering result (7c) is also shown where groupings between points are represented by a line segment connecting the points.

4.4.1 Algorithm Runtime Analysis

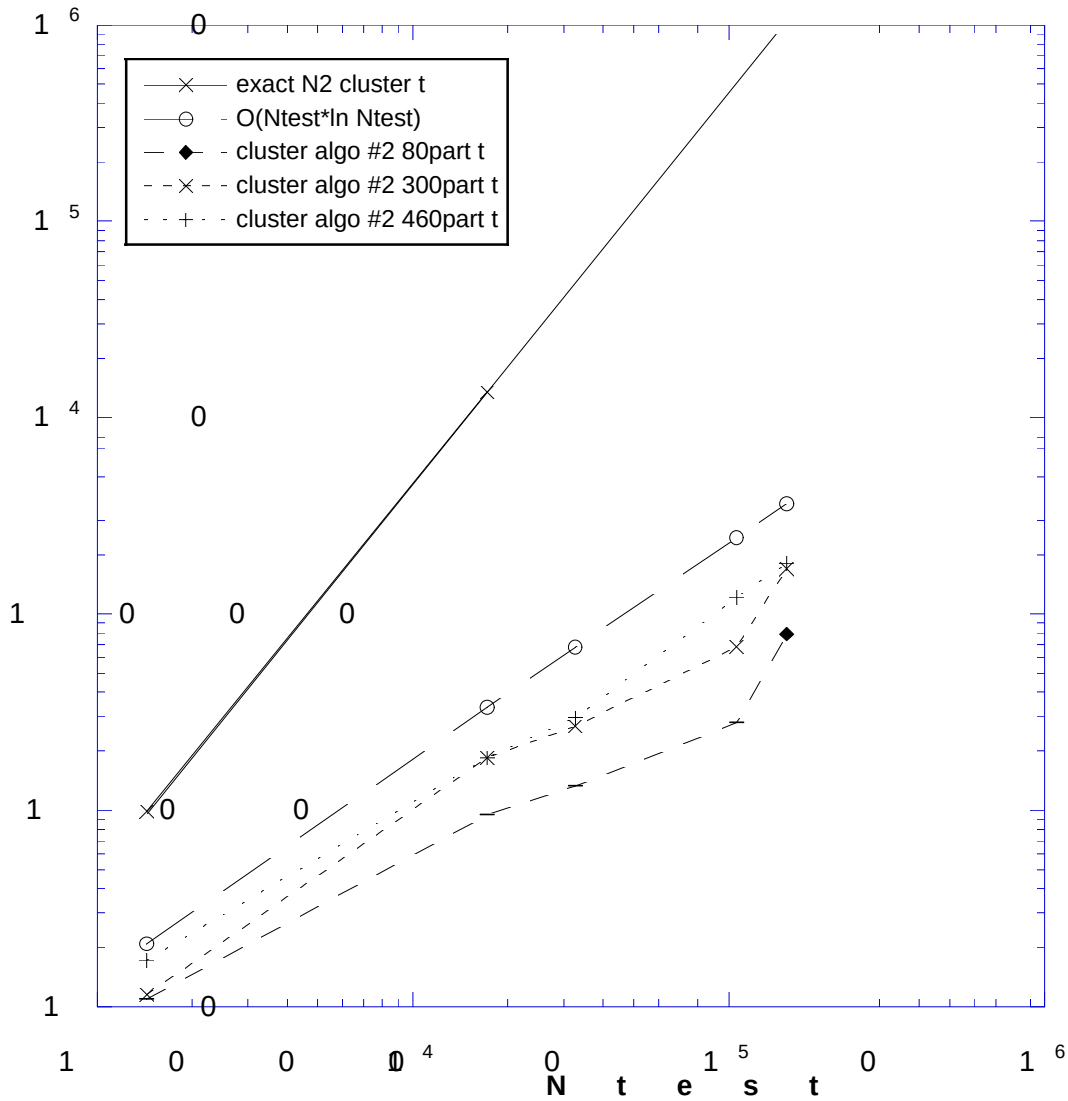

Fig. 8 Cluster Algo #2 Runtime Plot

Fig. 8 illustrates cluster algorithm #2 example runtimes using maximum partition rectangle sizes of m =80, 300 and 460 points, respectively. The time required to execute the exact $O(N^2)$ clustering algorithm is shown. Since its runtime is established, two true runtime points are plotted and an $O(N^2)$ line is used to extrapolate the results. The runtimes of clustering algorithm #2 are bounded by O(N*ln N) as illustrated by the O(N*ln N) reference curve.

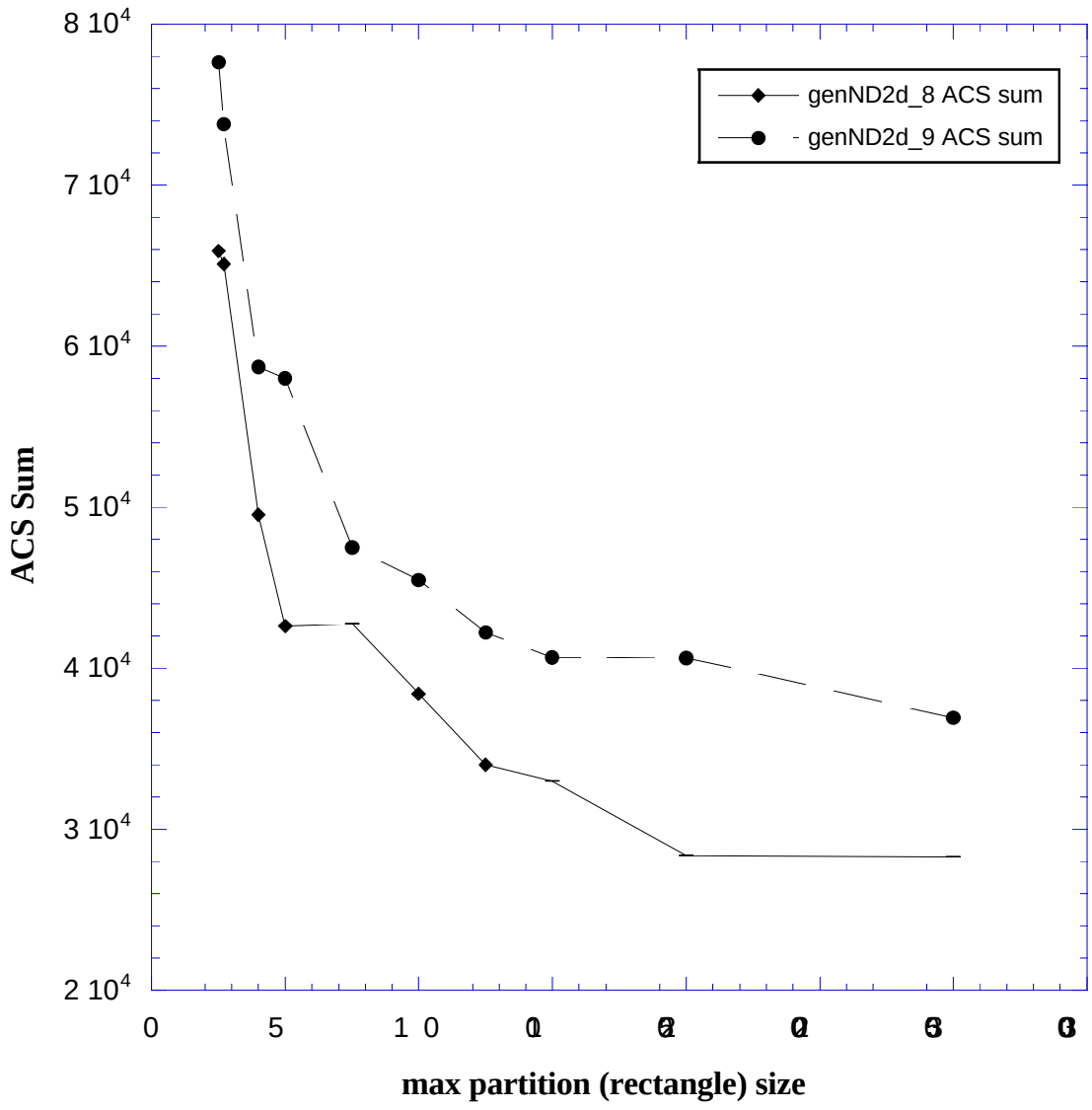

Fig. 9 Cluster Alg

Fig. 9 illustrates how the total number of ACS changes as a function of the partition size. The data plotted are the sum of the ACS for all the iterations required to complete clustering algorithm #2 using two data sets. Larger partition rectangles contain more points which do not group with projected points at rectangle boundaries and are therefore removed from the ACS set, while small partition rectangles allow more points to group with the rectangle boundaries. Referring to Fig. 10, we see the runtimes are not directly related to the sum of ACS above rectangle sizes of approximately 150. In this range, the time to process the larger number of points in the rectangles dominates the total time. Fewer ACS exist, but more time is required to process the increased number of points in larger rectangles.

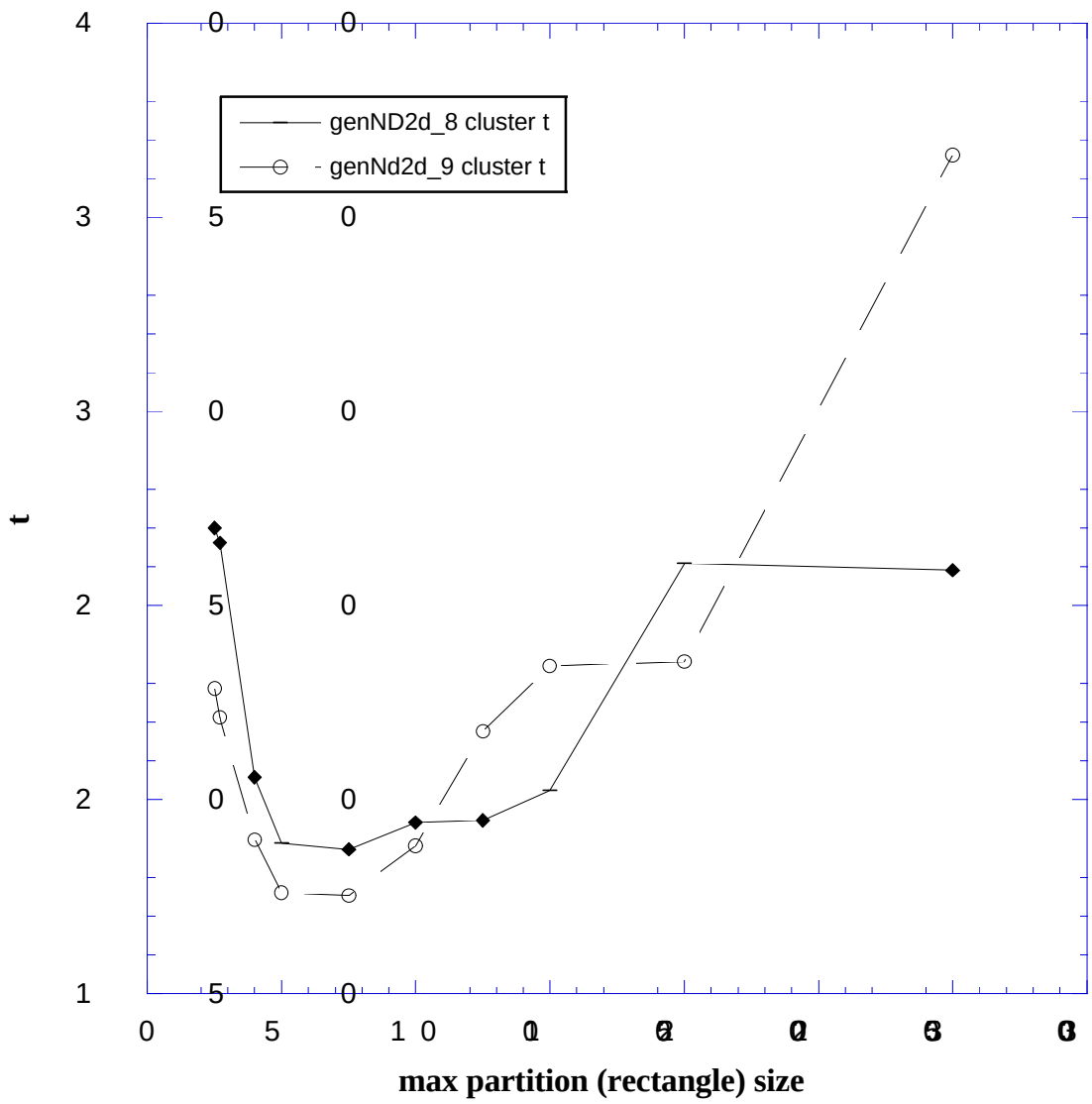

Fig. 10 Cluster Algo #2 Runtimes -- Vary Partition Size

Fig. 10 shows how the runtime of cluster algorithm #2 changes as a function of partition size using two data sets. One data set has N_{test}=20747 (genND2d_8), the other has N_{test}=30542 (genND2d_9). Optimal runtimes occur when the partition rectangle size is in the range of approximately 50-150 points. Below that range, the total number of ACS increases (see Fig. 9), causing the algorithm to execute more iterations to finish.

Using clustering algorithm #2 it was still possible, but less likely than clustering algorithm #1 to produce partition sizes on data sets which would cause the algorithm to "stall" on a set of ACS. By this we mean that the same set of partition boundaries and the same "stagnant" set of ACS points is produced in each subsequent iteration. In such a case, the algorithms are designed to display an error message and terminate.

4.4.2 Clustering Approximation Qualitative Results

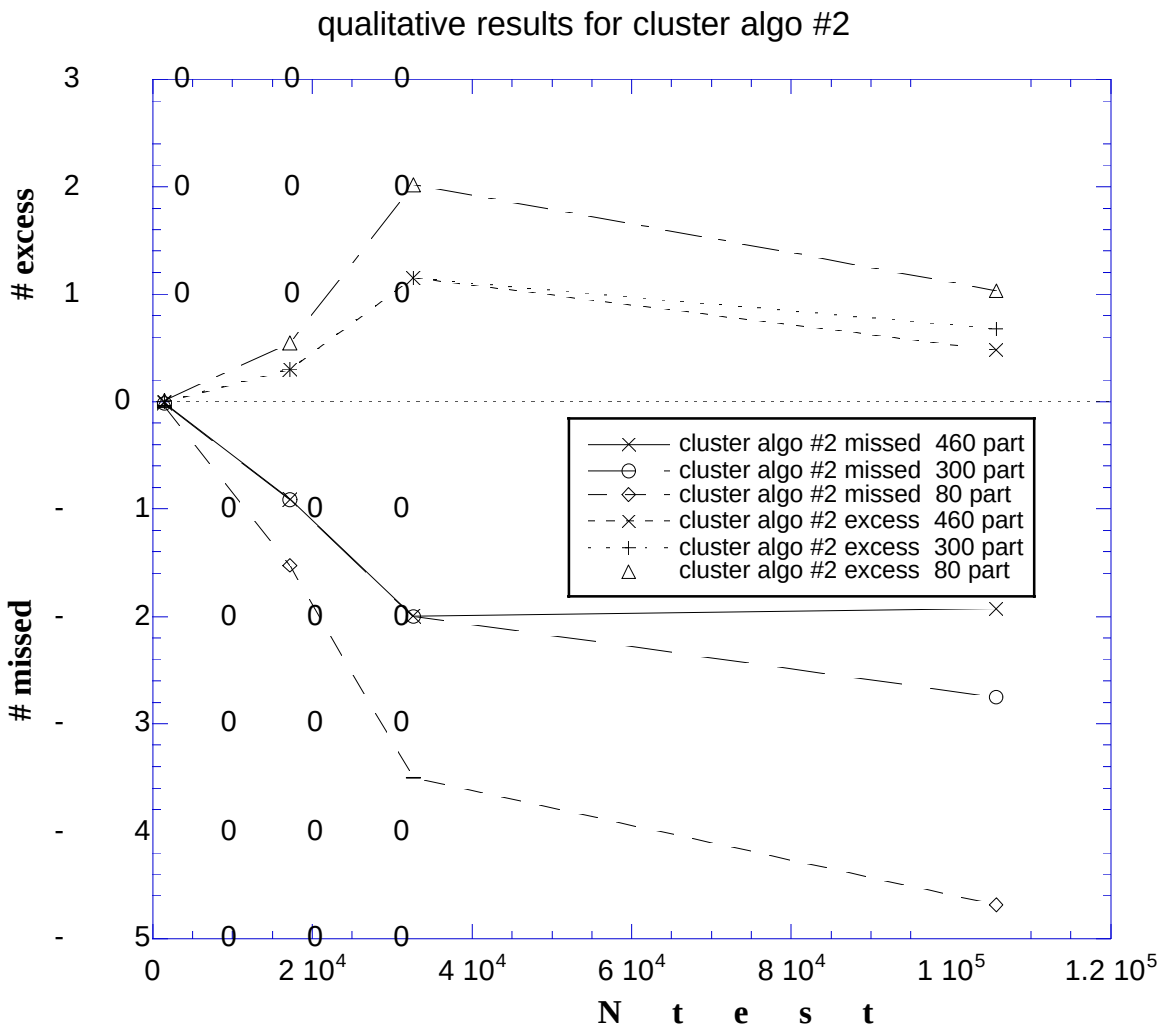

Fig. 11 Cluster Algo #2 Qualitative Results -- 3 Partition Sizes

Fig. 11 shows the number of missed and excess groupings which occurred over the range of test set sizes using partition rectangle sizes of 80, 300 and 460 maximum points per rectangle. The number of errors decreases as the size of the rectangles increases. Assuming that each point in the data set on average has more than one grouping, these errors represent less than 10% of the total correct groupings. This algorithm produced more missed grouping errors than excess errors over the range of test points examined.

4.5 D&C Clustering Algorithm #3

The next D&C algorithm again attempted to improve the process of removing ACS from each iteration of the algorithm. An attempt is made to cut the ACS in a dimension where stagnant points along a rectangle boundary cause a single dimension to be fixated on in the partitioning process, causing an infinite iteration problem. We arbitrarily change the choice of partitioning dimension in some of the iterations to avoid getting stuck with the same one each iteration. Within these selected iterations, we also selected which recursive partition steps did not use the dimension of largest variance. For example, the user can specify that on every second D&C iteration the fourth recursive call be modified. When those conditions are met, the D&C recursive call is made using the opposite dimension the program would have normally selected to partition on. The remainder of the algorithm performs the same operations as described for D&C Clustering in section 4.

4.5.1 Algorithm Runtime Analysis

This algorithm has the same runtime analysis as clustering algorithm #2 (O(N*log N)), with larger constants in front of the runtime equation. The partitioning iterations are interrupted enough to further alleviate the “stagnant” ACS set problem. This version can run using smaller partition rectangles than the previous versions and not get into the infinite iteration problem. However, since small rectangles have fewer points available as inhibitor candidates which could prevent the projected point groupings, the ACS set can remain large. As a result, there are still some small partition rectangle sizes which can cause the algorithm to hang up.

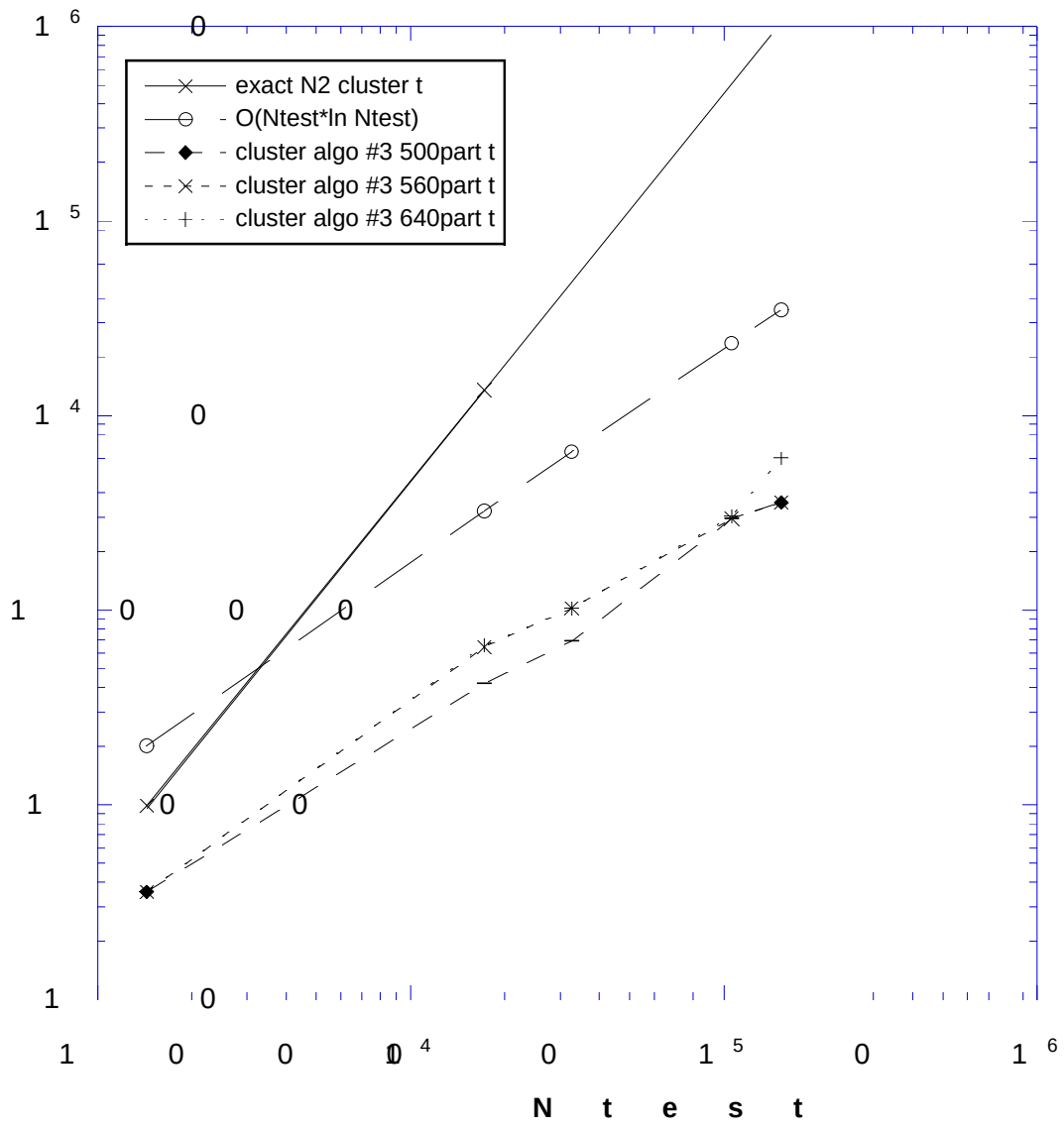

Fig. 12 Cluster Algo #3 Runtime Plot

Fig. 12 illustrates cluster algorithm #3 example runtimes using maximum partition rectangle sizes of 500,560 and 640 points. The time required to execute the exact $O(N^2)$ clustering algorithm is shown. Its runtime is established, two runtime points are plotted and an $O(N^2)$ line is used to extrapolate the results. The runtimes of clustering algorithm #3 are bounded by O(N *ln N) as illustrated by the O(N*ln N) reference curve.

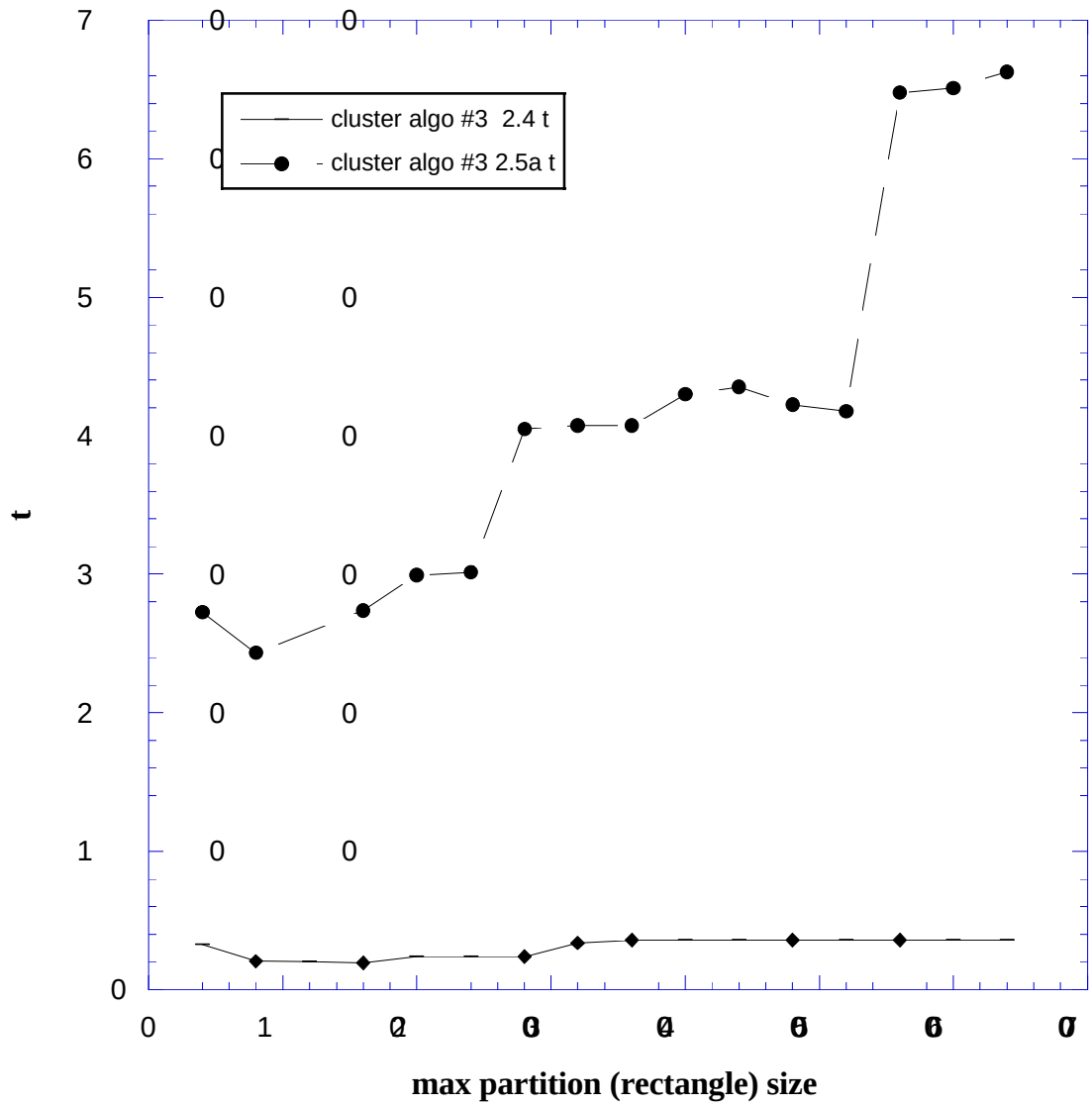

Fig. 13 Cluster Algo #3 Runtimes -- Vary Partition Size

Fig. 13 shows how the runtime of cluster algorithm #3 changes as a function of partition size. Almost constant runtimes are demonstrated by the two data sets in the partition rectangle range of approximately 40-500 points, and test set #2.4 runs in constant time for the whole range of partition sizes examined (40-460). Test set #2.4 has N_{test} =1438, 2.5a has N_{test} =17192.

4.5.2 Clustering Approximation Qualitative Results

The clustering results using algorithm #3 are improved compared to those of the previous versions. The qualitative results from this algorithm start to approach the exact results. We found a useful set of input parameters to alter the partitioning process. On every third D&C iteration the fourth recursive call would be altered. These parameters allowed the algorithm to remove points from the ACS set faster than other parameters examined.

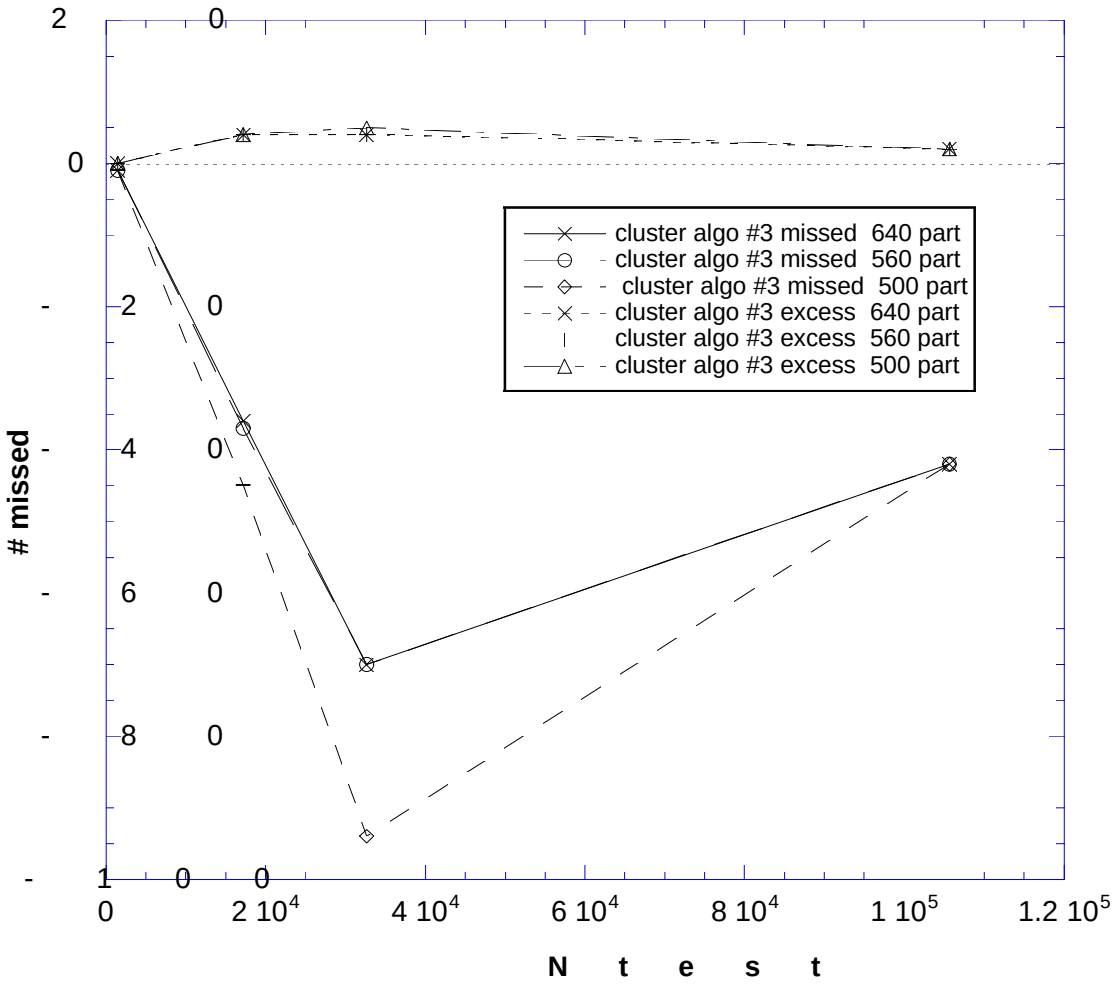

Fig. 14 Cluster Algo #3 Qualitative Results -- 3 Partition Sizes

Fig. 14 shows the number of missed and excess groupings which occurred using cluster algorithm #3 over a range of test set sizes and using partition rectangle sizes of 500,560 and 640 maximum points per rectangle. The number of errors decreases as the size of the rectangles increases. This algorithm produced more missed grouping errors than excess errors (almost none) over the range of test points examined. The number of errors is two orders of magnitude lower than the number of errors produced by cluster algorithm #2 using the same data.

4.6 D&C Clustering Algorithm #4

In the next version of the approximation clustering algorithm we want to improve on the estimates of whether or not a point can have groupers or inhibitors across rectangle boundaries -- this is another attempt to determine the ACS set. While the point projection method returns results which may be acceptable in some cases, it still has failures. Fig. 14 shows that clustering algorithm #3 had many times more missed groupings than excess groupings. The missed groupings can be attributed to the inadequacy of our point-projection method for estimating the presence of potential groupers in an adjacent rectangle. The point projection technique performs a VERI grouping test with the template perpendicular to the boundary the point was projected to. It is possible for a point to have groupers in an adjacent rectangle such that the orientation of the template on the data points is different from the orientation of the template on the data point and its projected point. Because of the change in template orientation, points which inhibit the projected point grouping may not necessarily inhibit all groupings to points in the adjacent rectangles. Groupings which could have occurred are missed.

Still using the D&C modifications from clustering algorithm #3 to partition the data, we introduce another concept to produce a better estimate of whether or not points may have groupers across a rectangle boundary and be included in the ACS set. As described in Osbourn and Martinez[1], the VERI algorithm uses a region of influence (ROI) encompassing a pair of grouping candidates to determine if they group together. If no other points from the data set lie in the ROI (template), the grouping of the two points is a success. Similarly, we can define an Inverse Template (IT), for a grouping point and an inhibiting point, which determines the volume around the grouping point in which grouping with other points is not inhibited by the inhibiting point. This IT can be computed from the VERI shape. For computational simplicity, we use an approximate IT shape (a rectangle) which contains the true IT. What this version of the algorithm does for each test point in the ACS is use the points in its grouper/inhibitor list as inhibitor candidates to generate an approximate IT (a rectangle containing the IT) for the grouper/inhibitor pair. If the inverse template can project across one rectangle boundary, then the point has a good chance of finding groupers in an adjacent rectangle, and is flagged for the next ACS. The fact that the approximate IT is larger than the true IT means that some points which occur in the rectangle may appear to be likely grouping candidates, but are not. We expect this method to produce larger ACS sets at each iteration than the projection methods in clustering algorithms #1-#3. The remainder of the algorithm performs the same operations described for D&C Clustering in section 4.

4.6.1 Algorithm Runtime Analysis

This algorithm has the same runtime analysis as clustering algorithm #3 (O(N*log N)), with larger constants in front of the runtime equation.

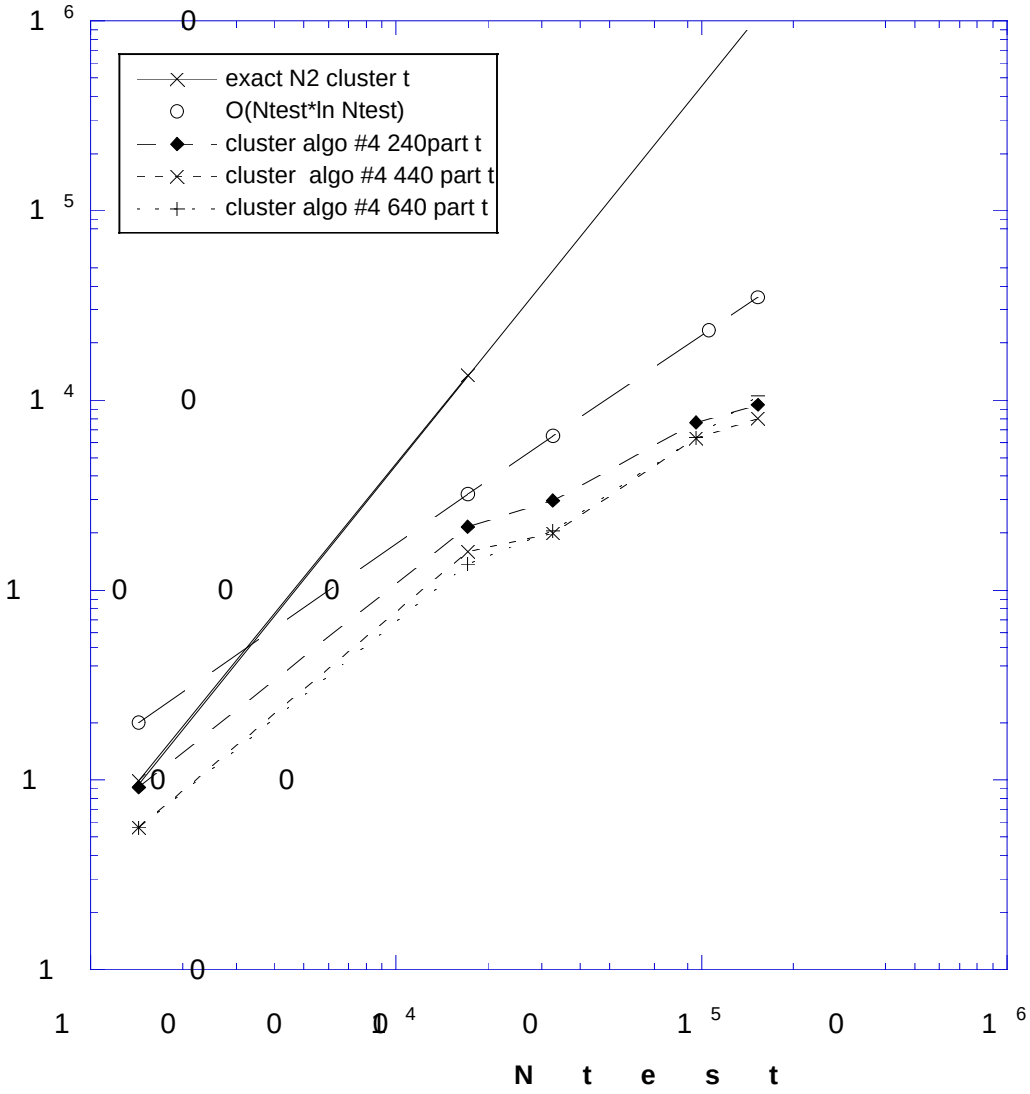

Fig. 15 Cluster Algo #4 Runtime Plot

Fig. 15 illustrates cluster algorithm #4 example runtimes using maximum partition rectangle sizes of 240,440 and 640 points. The time required to execute the exact $O(N^2)$ clustering algorithm is shown.. Cluster algorithm #4 has O(N*log N) runtime behavior, but because the constant in front of that runtime equation is large, the time to cluster small (<2000 points) test sets using cluster algorithm #4 begins to approach (or exceed) the time required to execute the exact algorithm on small data sets. The runtimes of clustering algorithm #4 are bounded by O(N * ln N) as illustrated by the O(N * ln N) reference curve.

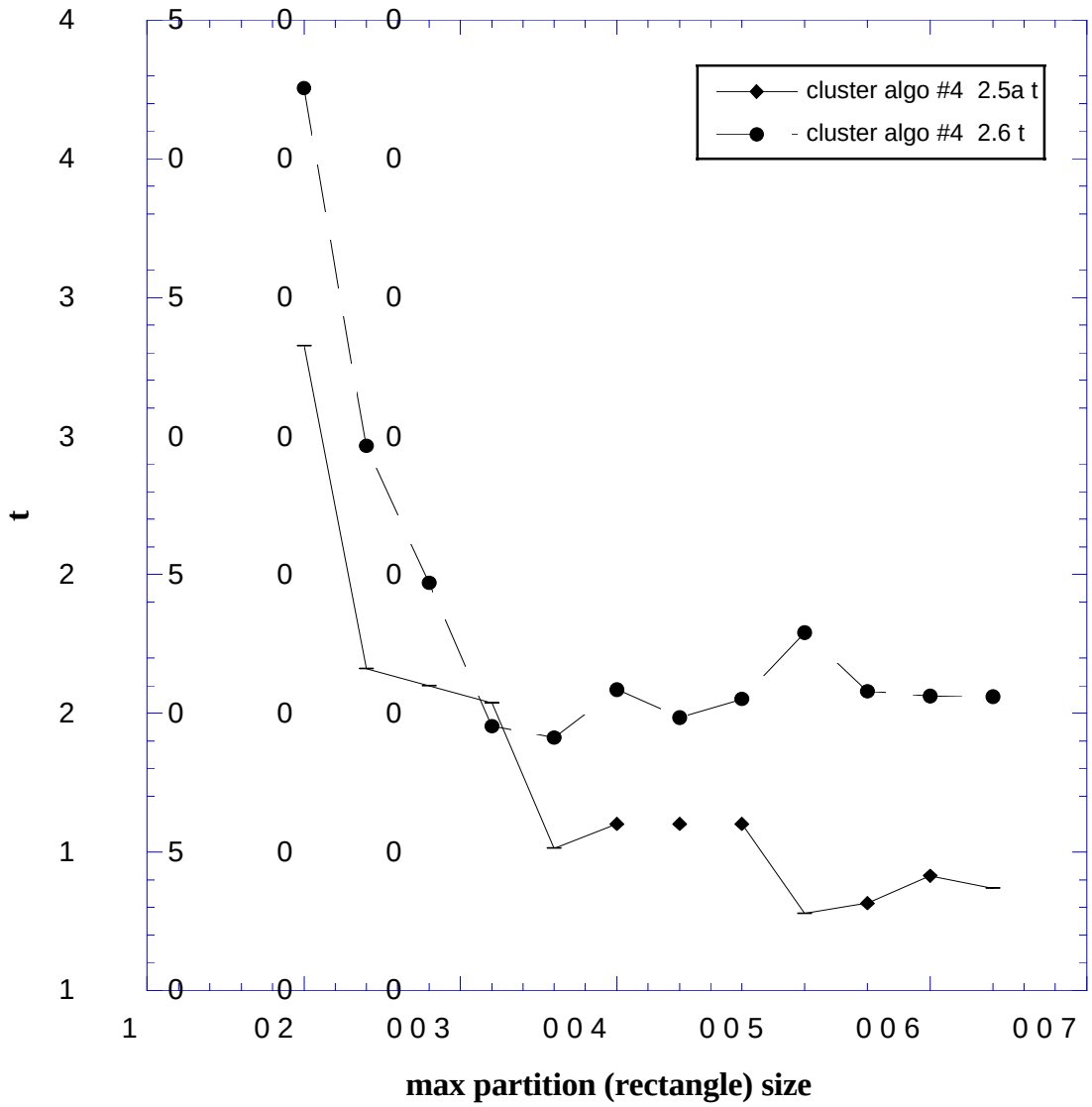

Fig. 16 Cluster Algo #4 Runtimes -- Vary Partition Size

Fig. 16 shows how the runtime of cluster algorithm #4 changes as a function of partition size for two data sets. Test set #2.5a has N = 17192, 2.6 has N = 32625. Almost constant runtimes are demonstrated by the two data sets in the partition rectangle range of approximately 300-650 points.

4.6.2 Clustering Approximation Qualitative Results

The clustering results using clustering algorithm #4 are again improved compared to those of the previous versions. The partitioning iterations are altered, this in conjunction with the inverse template is effective in determining which points to retain as ACS. If a true inhibitor point is not found some excess groupings can occur, but under most circumstances missed groupings will not occur. A case when groupers/inhibitors can be missed exists when a point being examined for groupings occurs close to a boundary of two adjacent rectangles. Only one of the adjacent rectangles is examined for groupers/inhibitors when in fact candidates could exist in the other rectangle too. The clustering results from this algorithm closely resemble the exact results, in most cases they are exact.

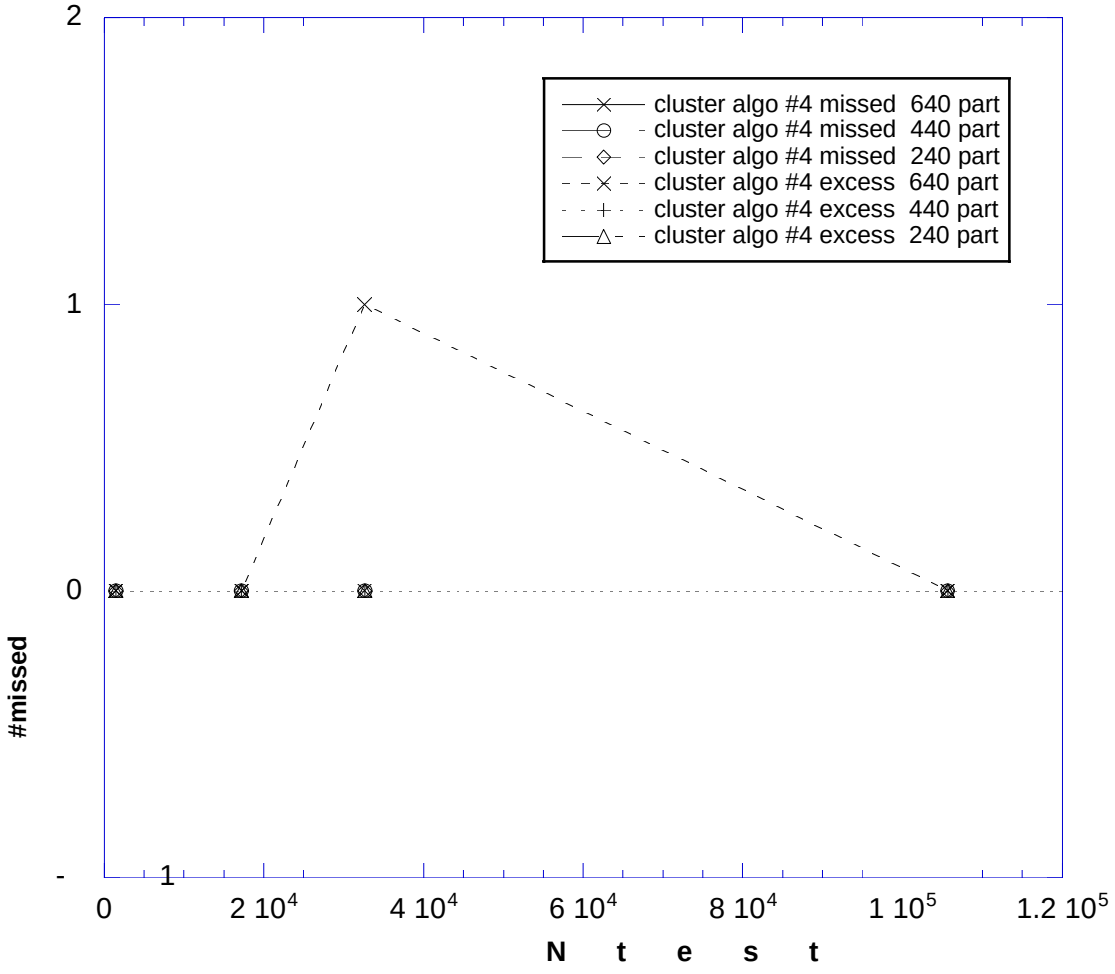

Fig. 17 Cluster Algo #4 Qualitative Results -- 3 Partition Sizes

Fig. 17 shows the absolute number of missed and excess groupings which occurred using cluster algorithm #4 over a range of test set sizes and using partition rectangle sizes of 240,440 and 640 maximum points per rectangle. With partition size of 640, one grouping was missed. All other clustering results were correct.

4.7 General Cluster Approximation Results/Observations

Algorithms #3 and #4 were evaluated and used more extensively from this set of cluster approximation algorithms. Clustering algorithm #4 using the IT produces essentially exact clustering results, but on average runs >200% slower than algorithm #3, which produces results approaching exact. The partition rectangle size which generates the fastest runtime is dependent on the data set being evaluated. However, a range of rectangle sizes which produced faster timing results was observed for each algorithm using 2-D data. For algorithm #2, the range is approximately 50-150 points, for algorithm #3 the range is about 40-500 points, and for cluster algorithm #4 the range is about 300-650 points per rectangle. In general, optimal runtime does not imply best clustering results. The problem of infinite iterations can be circumvented if small partition sizes on the lower end (or below) of the optimal ranges are avoided.

5. Discrimination Approximation Algorithms

This next section describes the approximation discrimination (PR) algorithms we developed to improve the runtime efficiency of our exact $O(N_{test}*N_{train})$ PR algorithm. PR algorithms use a set of known classified points (training set) to determine the class of unknown data points (test points). The class(es) assigned to a test point usually are a function of the class(es) of the training points in which the test point spatially occurs. A training set is a well defined and classified set of points, it does not change unless new or old class information is added to or deleted from the set. Therefore, we can take advantage of its static nature and apply one time preprocessing functions to the training set off-line. The preprocessing allows us to set up and store data structures which enable us to efficiently search the training set. The on-line algorithm can be designed to use the data structures and efficiently classify test points.

The training data is D&C partitioned into (hyper)rectangles with a fixed maximum number of points per rectangle and is stored in a search tree in a preprocessing step. The partitioning operation performed on the training set is similar to the partitioning of the data set done in clustering, but with one major difference; the training set is partitioned only one time and the data structure generated to represent the partitioned training set is stored to file.

A key feature of the data structure created to represent the training set is that it allows $O(\log N_{train})$ search times to find where in the training set's rectangles a test point occurs. A fixed number of NN's are also found and stored for each training point off-line. The discrimination algorithms operate on K-D data, K>=2. Test points are classified one at a time.

The first major operation performed in our PR algorithms is to find where in a training set a test point occurs -- which rectangle in the preprocessed training set it is bounded by. Once this rectangle has been found, all of the training points in that rectangle are examined as potential groupers. A fixed number of NN's of the training point grouping candidates are used as inhibitor candidates. A fixed number of test point NN's in the current rectangle are found and added to the grouper/inhibitor list. Following this step, it is necessary to seek additional groupers/inhibitors that may be in adjacent rectangles. Two types of grouping candidates are distinguished here -- internal candidates and external candidates. Internal candidates are the training points a test point attempts to group with which are in the rectangle a test point occurs in, external candidates are the training points in adjacent rectangles (with respect to the test point's rectangle) where a test point may find grouping candidates.

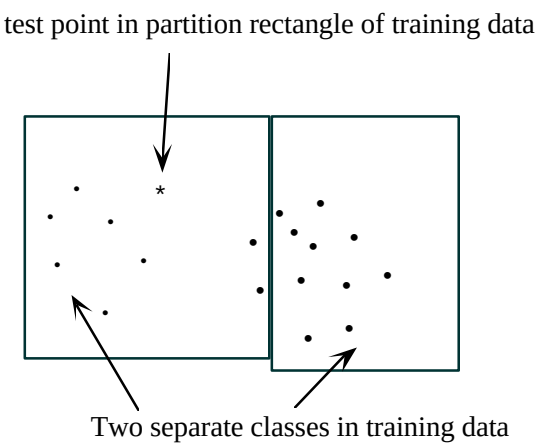

Fig. 18(a) PR schematic

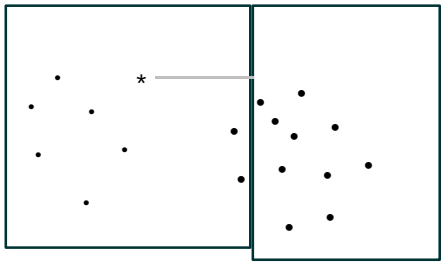

Fig. 18(b) test point projection -- test for more groupers

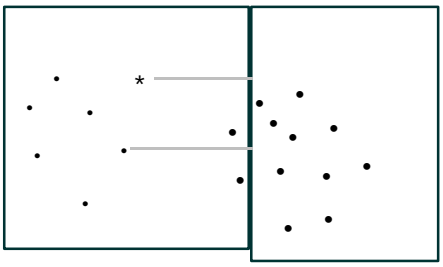

Fig. 18(c) grouped point projection -- test for more inhibitors

Fig. 18(a) illustrates a schematic showing a partitioned training set with a test point in the rectangle where it spatially occurs in the training data. The training points in the same rectangle as the test point are the internal grouping candidates. The rectangle sharing the same boundaries as the rectangle where the test point occurs is where the external grouping candidates exist. In our PR approximation algorithms, projections of the test point are used to find extra grouping and inhibiting candidates. This is analogous to the projections used in some of our clustering algorithms. The projected point, on the rectangle boundary, is tested for grouping with the test point. A fixed number of the test points NN's are used as the inhibitor candidates for the grouping attempt with the projected point. If the test point successfully groups with the projected point, then a search for external grouping candidates and NN's is done in the adjacent rectangle on which the projection of the ad hoc point occurred. Fig. 18(b) illustrates a test point projection. The same method of finding groupers in the rectangle where the test point occurs (internal candidates) is used to find groupers in the adjacent rectangles. After all the initial pair groupings of a test point and training points have been examined, a secondary screening of the groupings is performed. All the training points in the projected-to rectangles, in the test points NN list and in the test points rectangle (if necessary) will be used as inhibitor candidates of the grouping.

We also test if it is possible for more inhibitors to exist in rectangles different from those already examined. For each of the surviving groupings, a test is performed to determine the likelihood of finding more inhibitors in adjacent rectangles. The test is done by projecting a grouped-to training point to the test point's rectangle boundaries and using the projected point as another inhibitor candidate of the grouping. If the grouping attempt is inhibited, the training points in the projected-to rectangle are used as more inhibitor candidates of the grouping. If the grouping has not been inhibited, the same projection technique is applied to the test point in a final attempt to find more inhibitor candidates. Fig. 18(c) illustrates an example of these projections.

After searching for more inhibitor candidates and using them in an attempt to inhibit the test points groupings, a final PR result is computed. The class value of the training points which a test point remained grouped to are used to compute the classification result of a test point. In these approximation results, the class(es) of training points a test point groups with are stored as the classification result.

As mentioned before, the search tree data structure applied to the training data allows $O(\log N_{train})$ search times to find where in the training set a test point spatially occurs (what rectangle it's in). Once the rectangle has been found, it is constant time to process the test point. There are N_{test} test points
to process, therefore the on-line PR algorithm should run in $O(N_{test}*\log N_{train})$ time for sufficiently large N_{train}.

The training data used to evaluate the discrimination algorithms is derived from the chemical sensor data described in sec. 3.5. Seven sparse training sets from the 0.1%

training data and three sparse training sets from the 16% training data were generated. The original training set with 11202 points was used as a test set and classified with each of the sparse training sets. If a test set with known classes (a training set) is classified , four possible classifications can result. The test point can group with the correct class in the training set (ok), it can not group with any training points (out), it can group with an incorrect class (nok) or it can group with multiple classes of test points (mult). These four classifications (ok,out,nok and mult) are used to produce qualitative results for our PR algorithms.

The versions of our PR approximation algorithms differ mainly in the methods used to determine the internal and external grouping candidates of test points in the partitioned training data. training points. The differences are described in a section devoted to each algorithm.

5.1 Discrimination (PR) Approximation Algorithm #1

Once the rectangle in the partitioned training data where a test point occurs has been found, a test point is paired for potential grouping with all the training points in the rectangle where it occurs. In this version, to find internal grouping candidates, a test point and training point pair are examined for potential grouping using only a fixed number of NN's from the training point grouping candidates as inhibitor candidates. The training points successfully grouped with and a fixed number of test point NN's in the current rectangle are added to the grouper/inhibitor list of the test point.

5.1.1 Algorithm Runtime Analysis

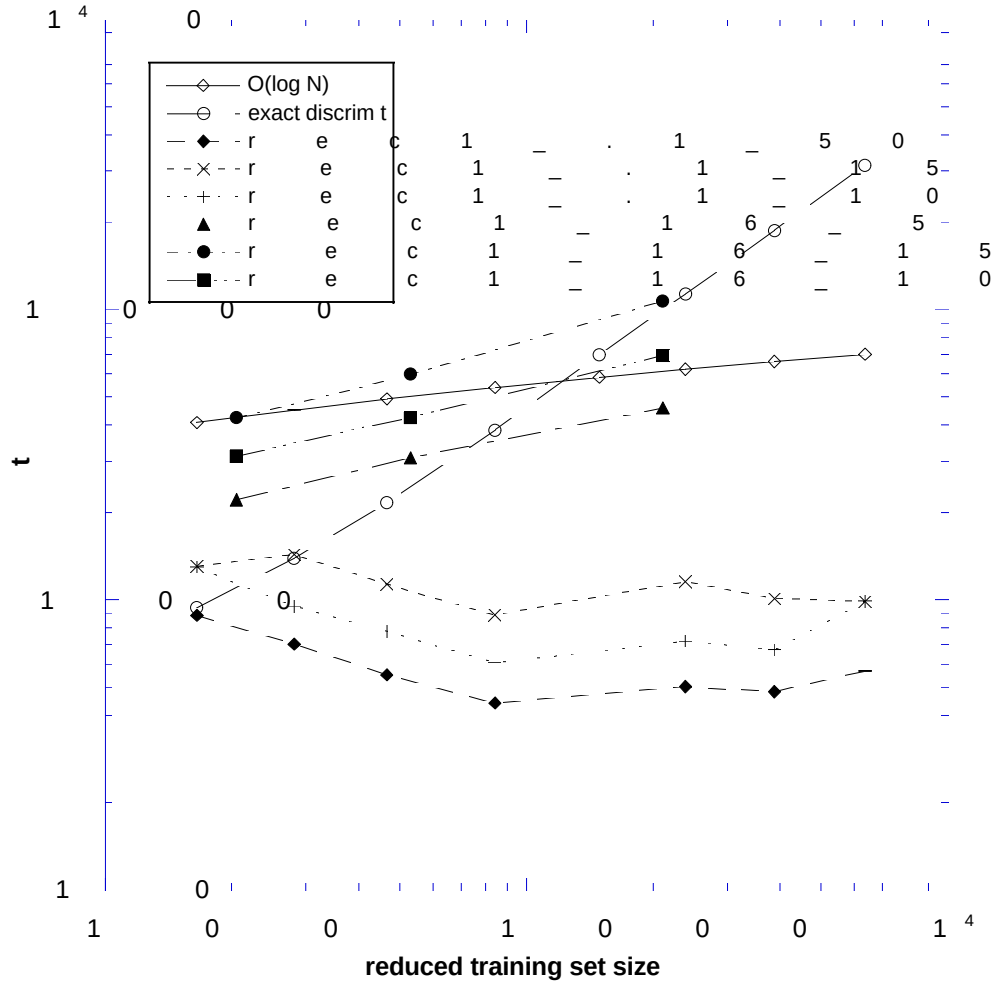

Fig. 19 Discrim Algo #1 Runtime Plot

Fig. 19 illustrates discrimination algorithm #1 example runtimes using maximum partition rectangle sizes of 50,100 and 150 points. Since the same test set was classified with each training set, the runtime plot of this test and training set pairs should be O(log N_{train}) for sufficiently large N_{train}. The constant in front of the runtime equation is large enough that the time to do PR with small (<2000 points) training sets (reduced from 16% training data) exceeds the time required to execute the exact algorithm. The runtimes of discrimination algorithm #2 on the 0.1% training data are constant, with larger rectangle sizes requiring more time to complete execution than small rectangles. However, down at the data set sizes of 100-300 (extrapolate results), runtimes on the 0.1% data are longer than the time required for the exact algorithm to run on those same sized data sets.

We are not seeing the asymptotic O(log N_{train}) runtimes with training data sets of this size. The runtimes are being dominated more by the time to process the test point and not by the time to find the test points rectangle in the training set. The training data with the 16% error has a runtime slope greater than that of the O(log N_{train}) curve. This can be attributed to the extra work of finding extra grouping and inhibitor candidates (see discrimination algorithm #5 timing results).

5.1.2 Qualitative Approximation Results

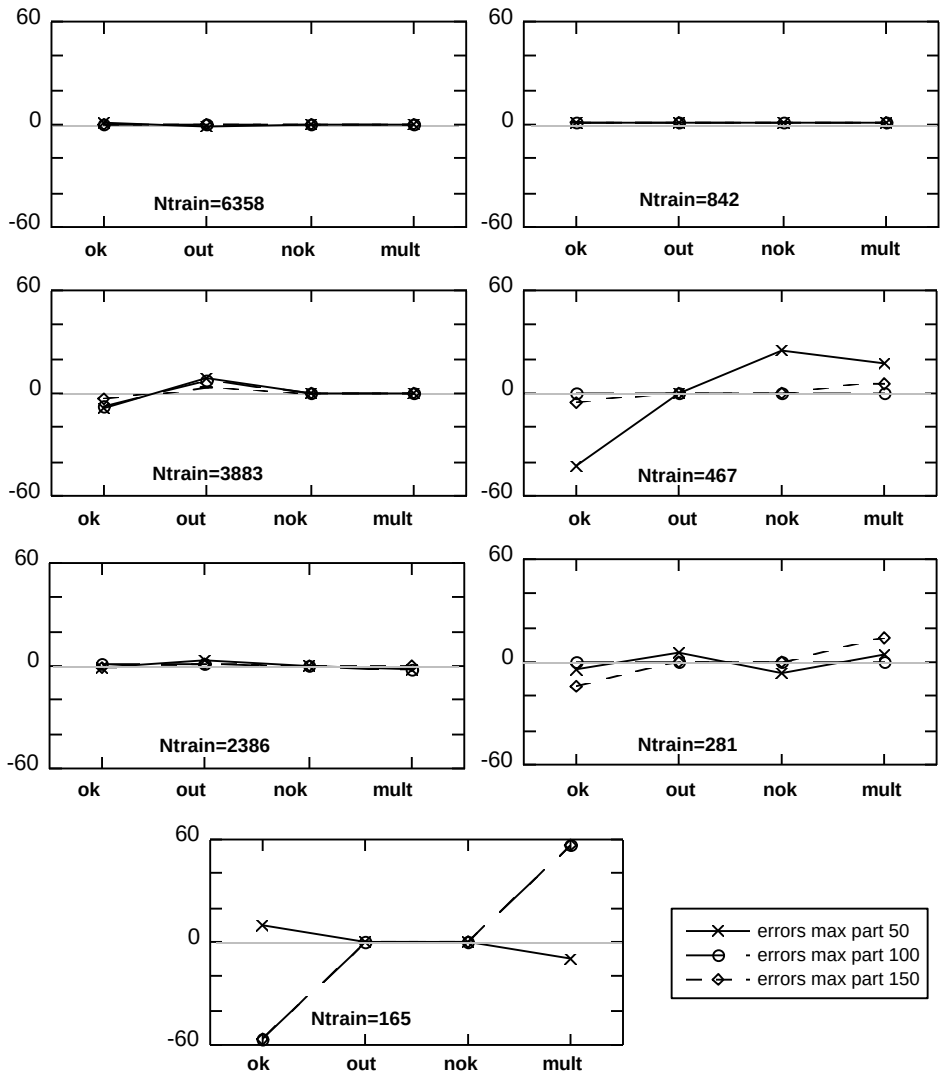

Fig. 20 Discrimination Algo #1 Qualitative Results, 0.1% Training Data

Fig. 20 illustrates PR algorithm #1 qualitative results using the seven sparse 0.1% training sets. The figure shows how the results degrade (relative to the exact results) as smaller training sets are used with the smallest partition size (50) used. The dashed line through 0 represents the exact algorithms results. The deviations from exact results were small with

the partition sizes of 100,150 points. Some points went from being (ok) to (out) (N_{train} =3883). Most errors were from less points that grouped (ok) and extra points which grouped (nok) and (mult) (N_{train} =467) or extra points that grouped (ok) and less points which grouped (mult) (N_{train} =165). The smaller partition rectangles generate more errors than the larger rectangles. In the smallest training set (N_{train} =165) the larger partition sizes (100, 150) had more errors than the partition size of 50. The classification results using the larger training sets produced less errors than the smaller training sets.

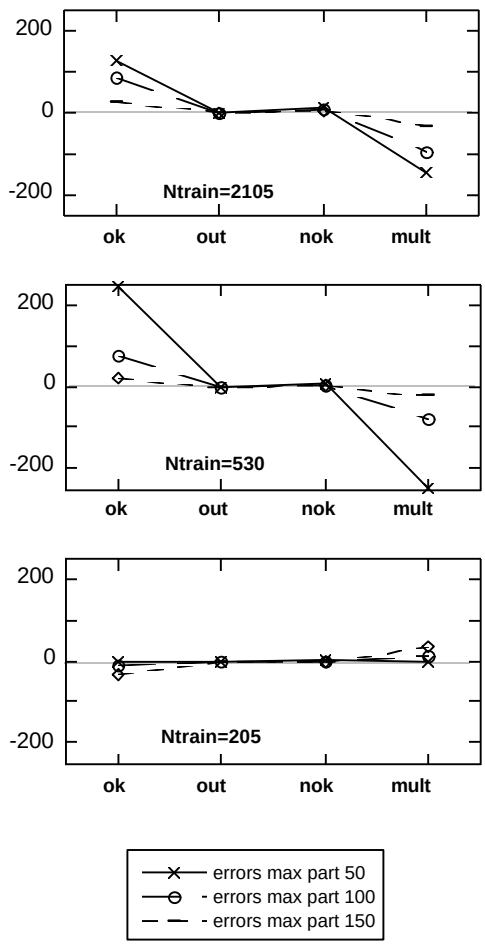

Fig. 21 Discrimination Algo #1 Qualitative Results, 16% Training Data

Fig. 21 illustrates PR algorithm #1 qualitative results using the three sparse 16% training sets. The dashed line through 0 represents the exact algorithms results. Using the reduced 16% training data generated more errors than the 0.1% training data. The smaller partition rectangles generate more errors than the larger rectangles, but unlike the 0.1% training data, the classification results using the smaller training sets produced less errors than the larger training sets. Most errors were from extra points that classified as (ok) and less as (mult).

5.2 Discrimination (PR) Approximation Algorithm #2

In this version, to find internal grouping candidates, a test point and training point pair are examined for potential grouping using a fixed number of NN's from the training point grouping candidate as inhibitor candidates. PR algorithm #1 and #2 are not very different. In this version, the NN's of the test point are also examined as inhibitor candidates if the training points NN's do not inhibit the grouping. In PR Approximation Algorithm #1, the NN's of the test point are added to the grouper/inhibitor list of the test point, but not examined as inhibitor candidates in the internal grouping candidate tests. The training points successfully grouped with and a fixed number of test point NN's in the current rectangle are added to the grouper/inhibitor list of the test point. The other details of the algorithm, i.e., methods for finding external grouping candidates, secondary screening of groupings, finding extra inhibitor candidates, and the final grouping tests are the same as PR approximation #1 and are described in section 5.

5.2.1 Algorithm Runtime Analysis

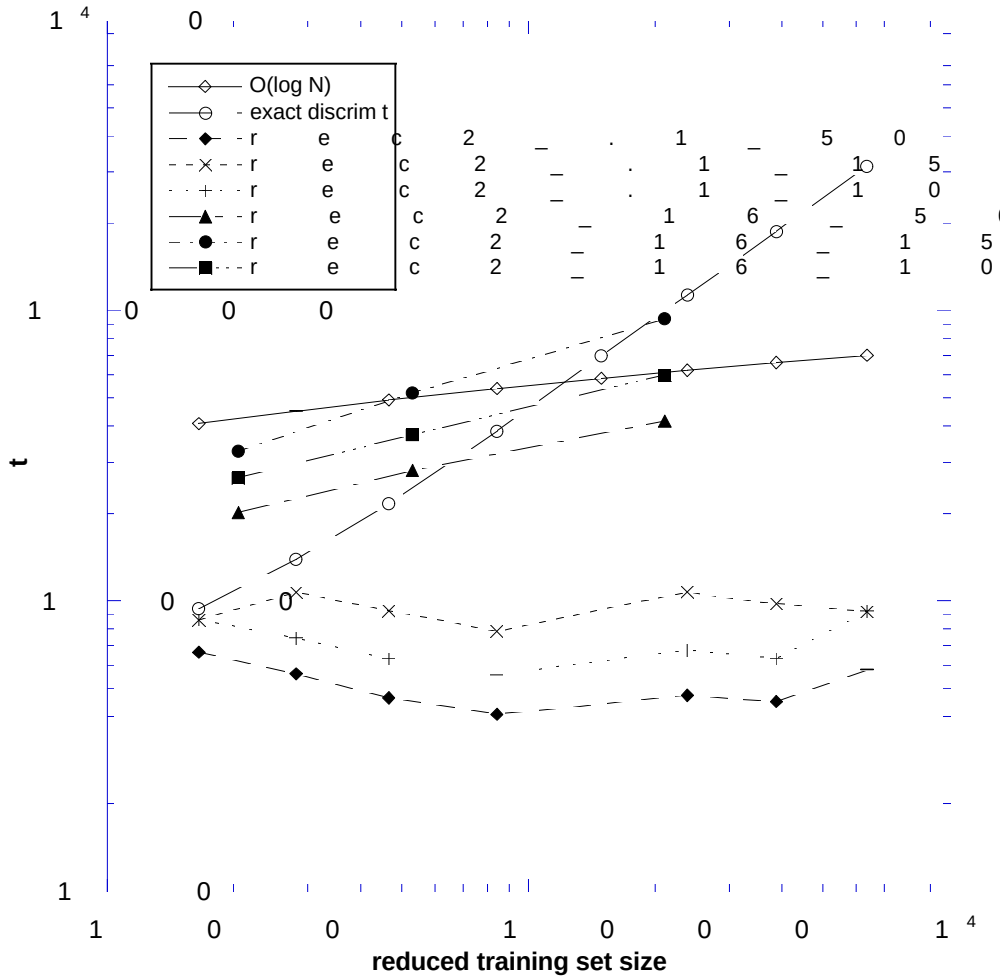

Fig. 22 Discrim Algo #2 Runtime Plot

Fig. 22 illustrates discrimination algorithm #2 example runtimes using maximum partition rectangle sizes of 50, 100 and 150 points. Since the same test set was classified with each

training set, the runtime plot of this test and training set pairs should be O(log N_{train}) for sufficiently large N_{train}. The constant in front of the runtime equation is large enough that the time to do PR with small (<2000 points) training sets (sparse sets from 16% training data) exceeds the time required to execute the exact algorithm. The runtimes of discrimination algorithm #2 on the 0.1% training data are constant, with larger rectangle sizes requiring more time to complete execution than small rectangles.

As in discrimination algorithm #1, we are not seeing the asymptotic O(log N_{train}) runtimes with training data sets of this size. The runtimes are being dominated more by the time to process the test point and not by the time to find the test points rectangle in the training set. The training data with the 16% error has a runtime slope greater than that of the O(log N_{train}) curve. This can be attributed to the extra work of finding extra grouping and inhibitor candidates (see discrimination algorithm #5 timing results).

This version runs about 10-20% faster than PR algorithm #1 on the same test patterns. More initial grouping were inhibited with the test points NN's as inhibitor candidates than in the previous version which didn't use the test points NN's as initial inhibitor candidates.

5.2.2 Qualitative Approximation Results

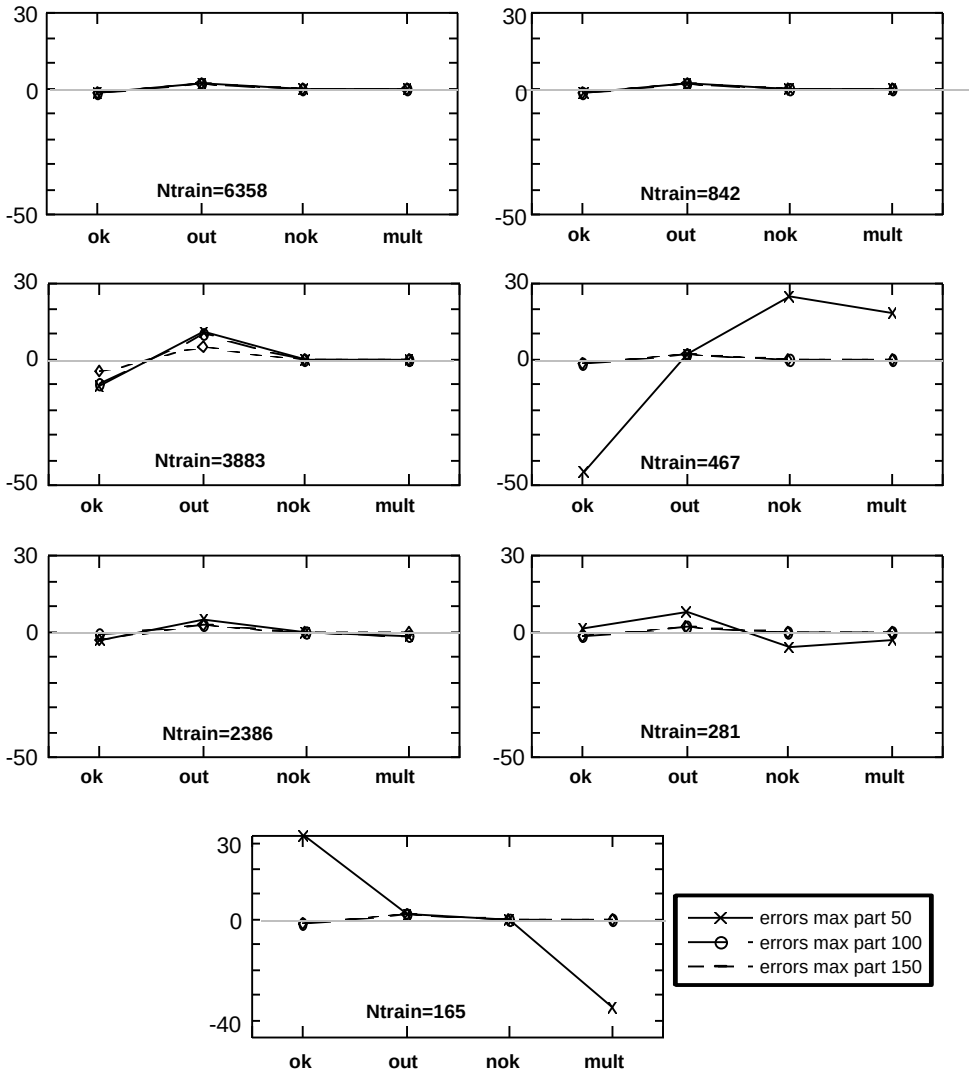

Fig. 23 Discrimination Algo #2 Qualitative Results, 0.1% Training Data

Fig. 23 illustrates PR algorithm #2 qualitative results using the seven sparse 0.1% training sets. The figure shows how the results degrade (relative to the exact results) as smaller training sets are used with the smallest partition size (50) used. The dashed line through 0 represents the exact algorithms results. The deviation from exact results was small with the partition sizes of 100,150. Some points went from being (ok) to (out) (N_{train} =3883). Most errors were from less points that grouped (ok) and extra points which grouped (nok) and (mult) (N_{train} =467) or extra points that grouped (ok) and less points which grouped (mult) (N_{train} =165). The smaller partition rectangles generate more errors than the larger rectangles. The classification results using the larger training sets produced less

errors than the smaller training sets. By using the test point's NN's as inhibitor candidates in the internal grouping tests, this version produced less errors than PR algorithm #1.

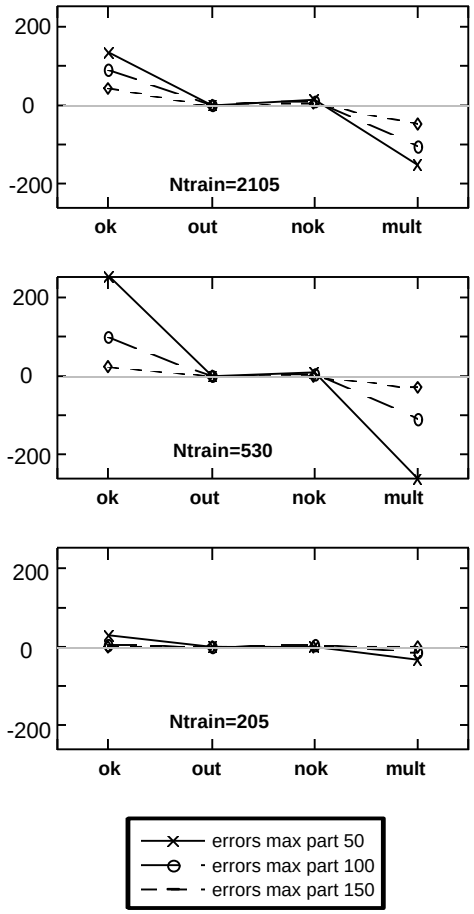

Fig. 24 Discrimination Algo #2 Qualitative Results, 16% Training Data

Fig. 24 illustrates PR algorithm #2 qualitative results using the three sparse 16% training sets. The dashed line through 0 represents the exact algorithms results. Using the reduced 16% training data generated more errors than the 0.1% training data. The smaller partition rectangles generate more errors than the larger rectangles, but unlike the 0.1% training data, the classification results using the smaller training sets produced less errors than the larger training sets. Most errors were from extra points that classified as (ok) and less as (mult).

5.3 Discrimination (PR) Approximation Algorithm #3

The order in which the training points are used as inhibitor candidates is modified from previous versions. The ordering is modified with the expectation that the training points in the NN list of a test point may be better inhibitor candidates than the NN's of a training point. It is possible that a test point's NN's are closer to the test point than a training points NN's are to it. A fixed number of test point NN's in the test point's rectangle are found and added to the test points grouper/inhibitor list and are used as inhibitor candidates. If the test point's NN's do not inhibit the grouping, a fixed number of NN's of the training point grouping candidates are used as inhibitor candidates.

After all the groupings of a test point with the training points using the test and training point NN's as inhibitor candidates have been found, a secondary screening of the groupings is performed. As in finding the internal groupings of a test point, the order in which the training points are used as inhibitor candidates in the secondary screening of groupings are modified from previous versions. All the training points in the test points rectangle, the test points NN list and the training points rectangle (if different from the test points rectangle) will be used as inhibitor candidates of the grouping. The other details of the algorithm, i.e., methods for finding external grouping candidates, finding extra inhibitor candidates, and the final grouping tests are described in section 5.

5.3.1 Algorithm Runtime Analysis

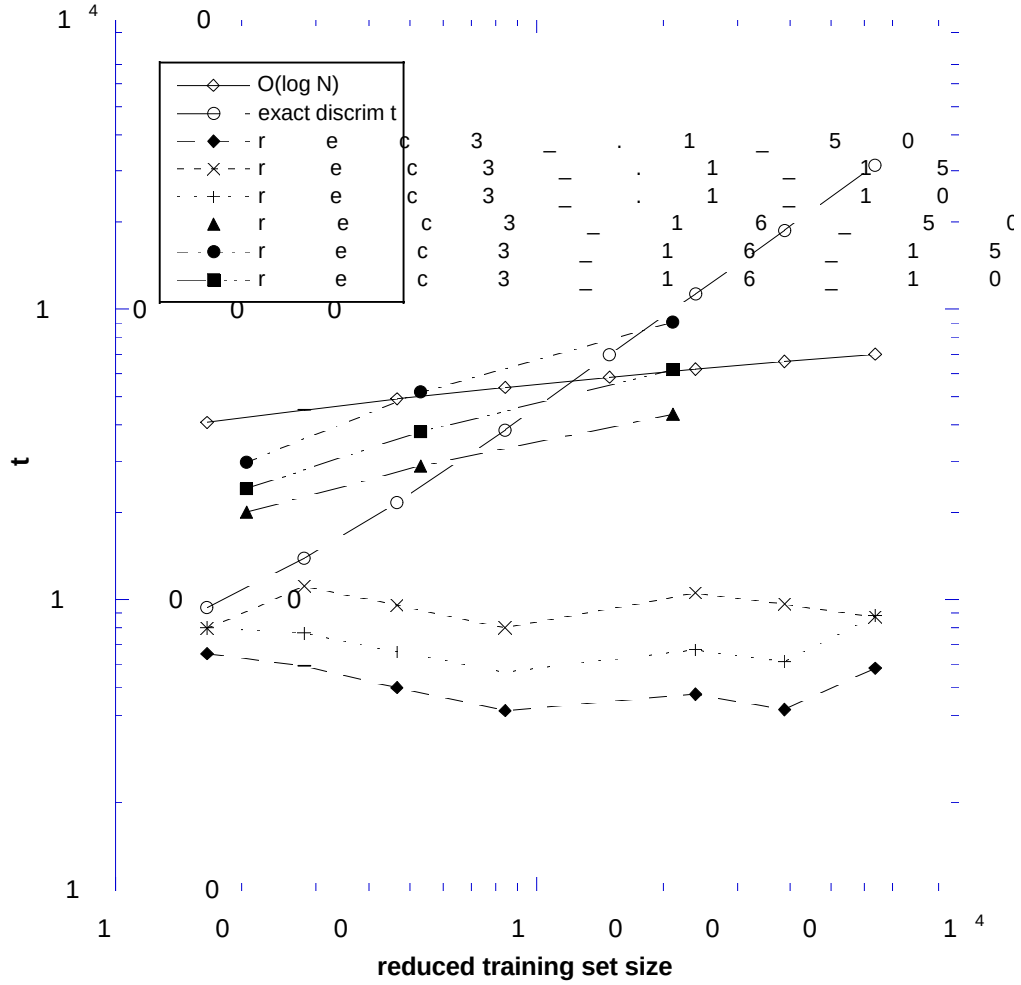

Fig. 25 Discrim Algo #3 Runtime Plot

Fig. 25 illustrates PR approximation algorithm #3 example runtimes using maximum partition rectangle sizes of 50,100 and 150 points. For each training set the same fixed sized test set of 11202 points was classified. Therefore the runtime plot of these test and training set pairs should be O(log N_{train}) for sufficiently large N_{train}. The time required to execute the exact discrimination algorithm is shown.

The constant in front of the runtime equation is large, the time to do PR using small (<2000 points) training sets (reduced from 16% training data) exceeds the time required to execute the exact algorithm. The runtimes of discrimination algorithm #3 on the 0.1% training data are constant, with larger rectangle sizes requiring more time to complete execution than small rectangles. As discussed in sec 5.2.1, we are not seeing the asymptotic O(log N_{train}) runtimes with the 16% training data. These timing results are very similar to the timing results of PR algorithm #2.

5.3.2 Qualitative Approximation Results

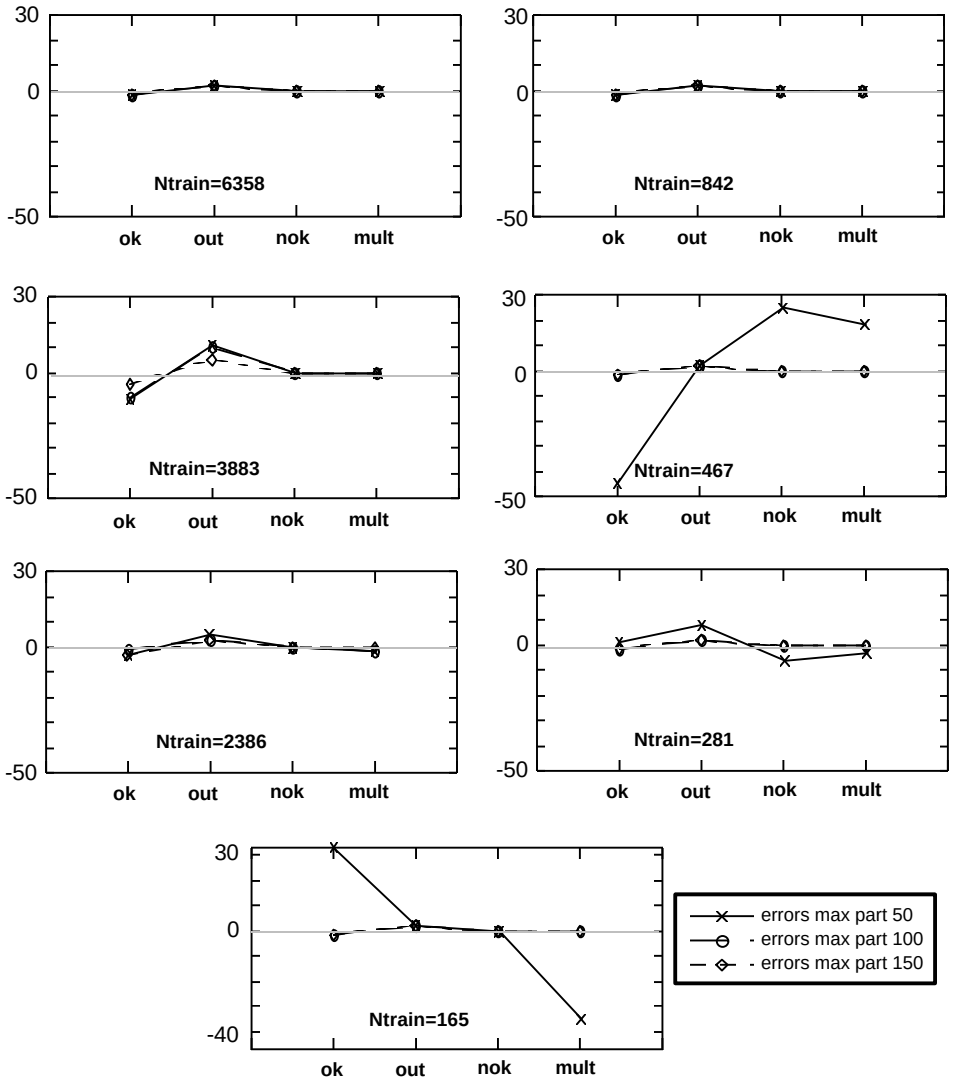

Fig. 26 Discrimination Algo #3 Qualitative Results, 0.1% Training Data

Fig. 26 illustrates PR approximation algorithm #3 qualitative results using the seven sparse 0.1% training sets. The figure shows how the results degrade (relative to the exact results) as smaller training sets and the smallest partition size (50) are used. The deviation from exact results was small with the partition sizes of 100,150. Some points went from being (ok) to (out) (N_{train} =3883). Most errors were from less points that grouped (ok) and extra points which grouped (nok) and (mult) (N_{train} =467) or extra points that grouped (ok) and less points which grouped (mult) (N_{train} =165). These qualitative results are similar to those of PR algorithm #2.

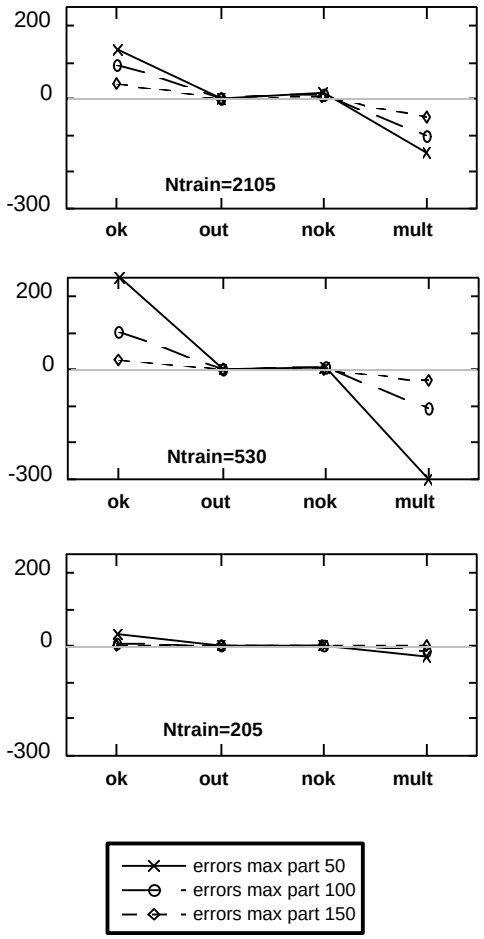

Fig. 27 Discrimination Algo #3 Qualitative Results, 16% Training Data

Fig. 27 illustrates PR approximation algorithm #3 qualitative results using the three sparse 16% training sets. Using the these training data sets generated more errors than the reduced 0.1% training data sets. The smaller partition rectangles generate more errors than the larger rectangles, but the larger training sets produced more errors than the smallest training set. Most errors were from an excess of points that grouped (ok) and less points grouped (mult). These qualitative results have more errors in the case where N_{train} = 530 with the maximum rectangle size of 50 than those of PR algorithm #2.

5.4 Discrimination (PR) Approximation Algorithm #4

In order to speed up the internal grouping time, the only points which are used as inhibitor candidates for the internal groupings are a fixed number of NN's of the test point and the training point. As mentioned in section 5, in order to find external grouping candidates the test point is projected to the boundaries of the rectangle where it occurs and creates ad hoc points to use as grouping candidates. A fixed number of the test points NN's are used as the inhibitor candidates for the grouping attempt with the projected point. If the test point successfully groups with the projected point, then a search for more grouping candidates and test point NN's is performed in the adjacent rectangle. The extra groupings and NN's found are added to the test points grouper/inhibitor list.

In this version, we increase the number of inhibitor candidates examined in a grouping attempt by a test point and training point pair with expectations of improving the (ok) classifications which occur for the test points. All the accumulated points in the NN list of the test point are used as inhibitor candidates when searching in adjacent rectangles for external grouping candidates. As adjacent rectangles are searched for external grouping candidates, NN's of the test point are found and added to the grouper/inhibitor list of the test point and used as inhibitor candidates. The more adjacent rectangles searched for external grouping candidates, the more NN's are added to the test point's list and used in the test point's subsequent grouping tests as inhibitor candidates. A fixed number of the training points NN's are used as inhibitor candidates if the grouping is not inhibited by the test points NN's. This number is increased to six points from the usual three used by the other algorithms. The other details of the algorithm, i.e., secondary screening of groupings, finding extra inhibitor candidates, and the final grouping tests are described in section 5.

5.4.1 Algorithm Runtime Analysis

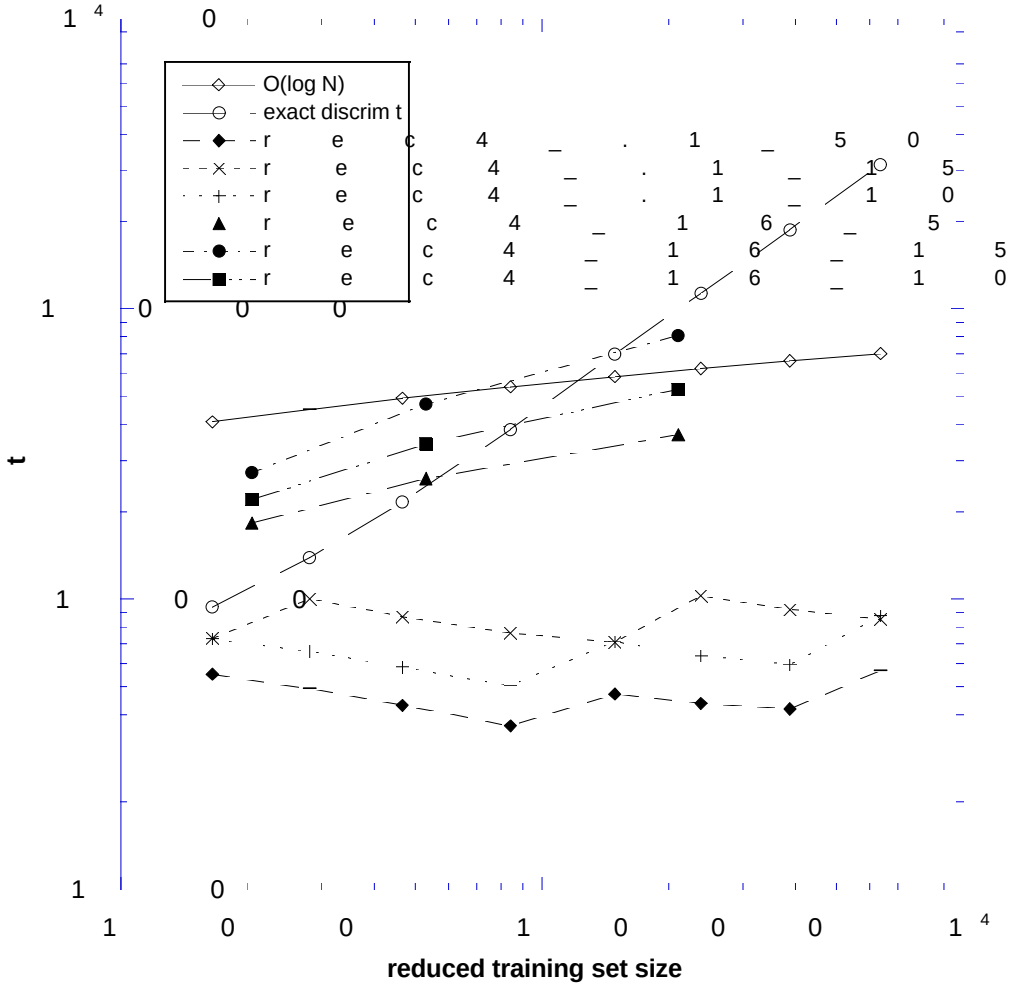

Fig. 28 Discrim Algo #4 Runtime Plot

Fig. 28 illustrates discrimination algorithm #4 example runtimes using maximum partition rectangle sizes of 50,100 and 150 points. For each training set the same fixed sized test set was classified. Therefore the runtime plot of this test and training set pairs should be $O(\log N_{train})$ for sufficiently large N_{train}. The time required to execute the exact discrimination algorithm is also shown.

The constant in front of the runtime equation is large. The time to do PR using small (<2000 points) training sets (reduced from 16% training data) exceeds the time required to execute the exact algorithm. The runtimes of discrimination algorithm #4 on the 0.1% training data are constant, with larger rectangle sizes requiring more time to complete execution than small rectangles. At times the partition rectangles have the same number of points in them even if different maximum sizes are specified. For this reason some of the timing points in the partition sizes 100 and 150 are similar. As discussed in section

5.2.1, we are not seeing asymptotic O(log N_{train}) runtimes with training data sets of this size.

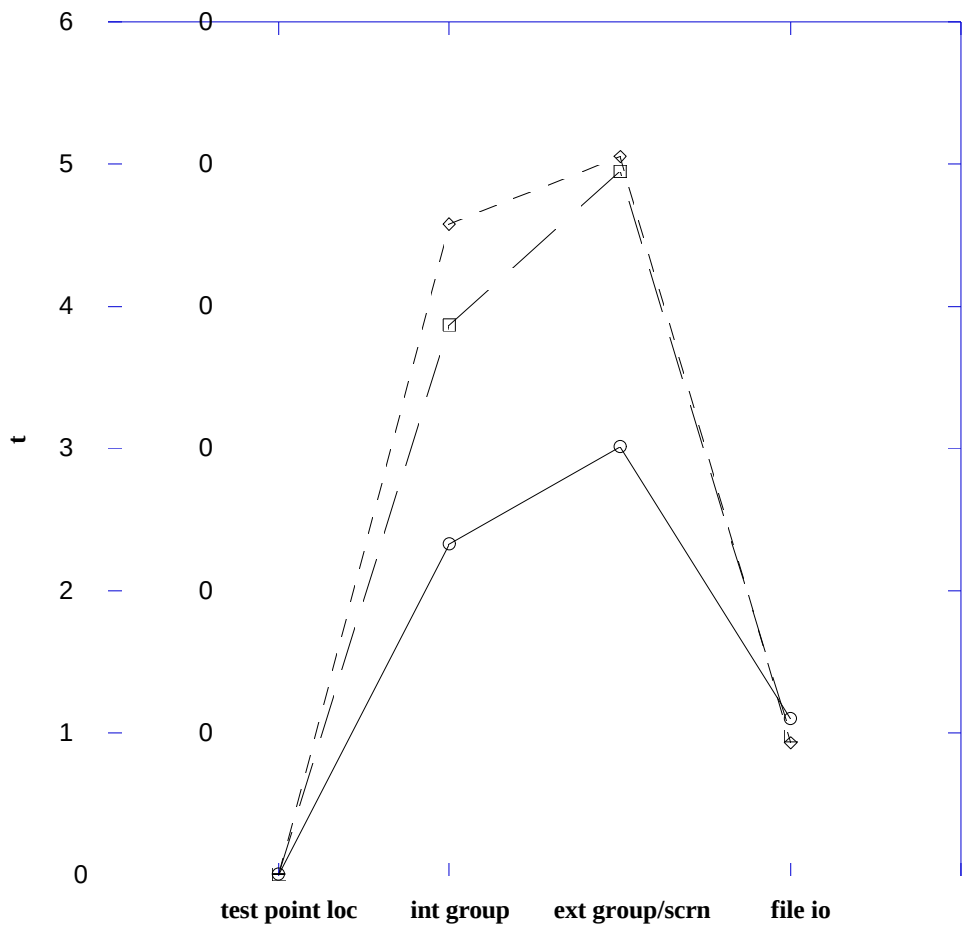

Fig. 29 Analysis of Discrimination Algorithm Runtime

In Figure 29 the runtimes of three classified data sets are analyzed, showing how PR approximation algorithm #4 utilizes it's time. The first label (test point loc) plots the total time spent finding the test points rectangle in the partitioned training data. This time is negligible -- in the noise of the total time. The label (int group) is used to represent the total time spent finding internal groupers. About 40% of the total time is spent finding internal groupers. The label (ext group/screen) represents the time spent searching for external groupers and screening the groupings the test points make. The largest fraction of time, about 50%, is spent searching for external groupings and screening the groupings. The label (file io) represents the time spent (about 10%) reading in test points and outputting classification results to file.

5.4.2 Qualitative Approximation Results

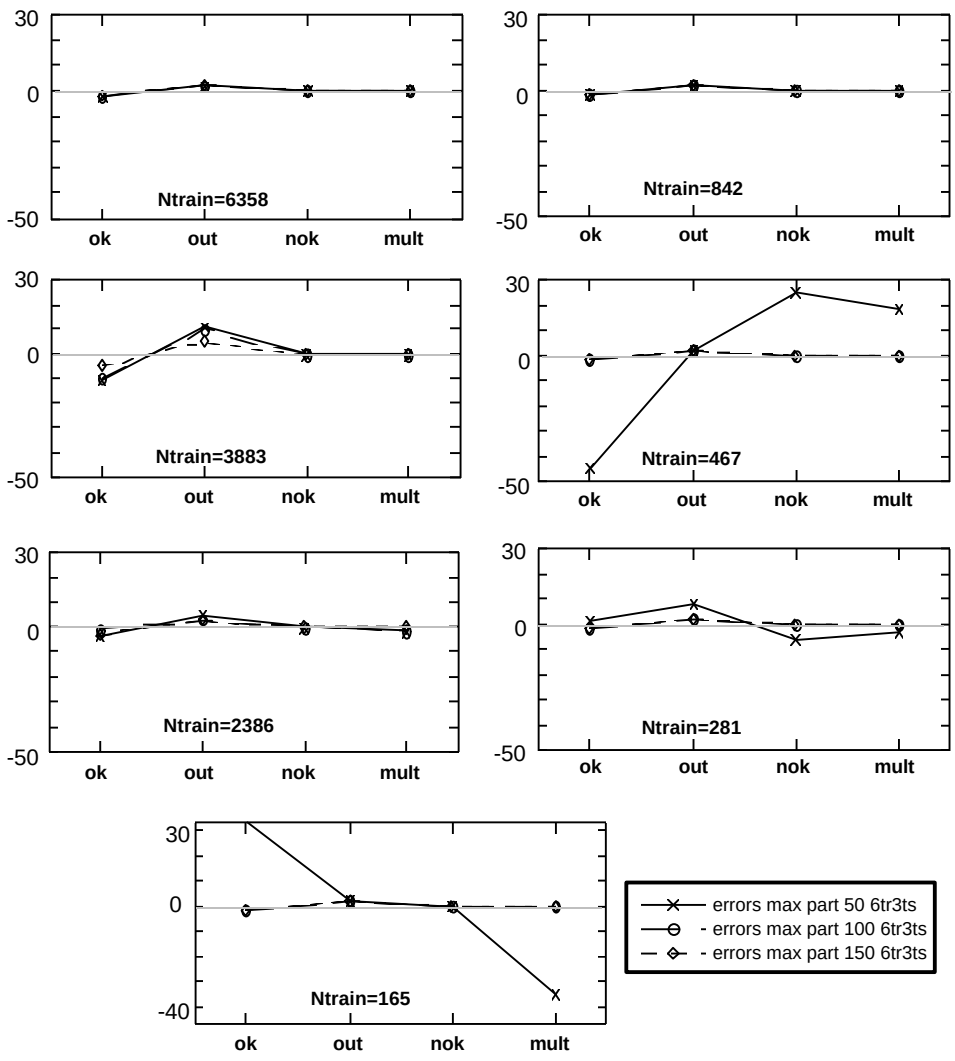

Fig. 30 Discrimination Algo #4 Qualitative Results, 0.1% Training Data, 6,3 NN

Fig. 30 illustrates PR approximation algorithm #4 qualitative results using the seven sparse 0.1% training sets. The figure shows how the results degrade (relative to the exact results) as smaller training sets and the smallest partition size (50) are used. The deviation from exact results was small with the partition sizes of 100,150. Some points went from being (ok) to (out) (N_{train} =3883). Most errors were from less points that grouped (ok) and extra points which grouped (nok) and (mult) (N_{train} =467) or extra points that grouped (ok) and less points which grouped (mult) (N_{train} =165). These qualitative results are similar to those of PR algorithm #2 and #3.

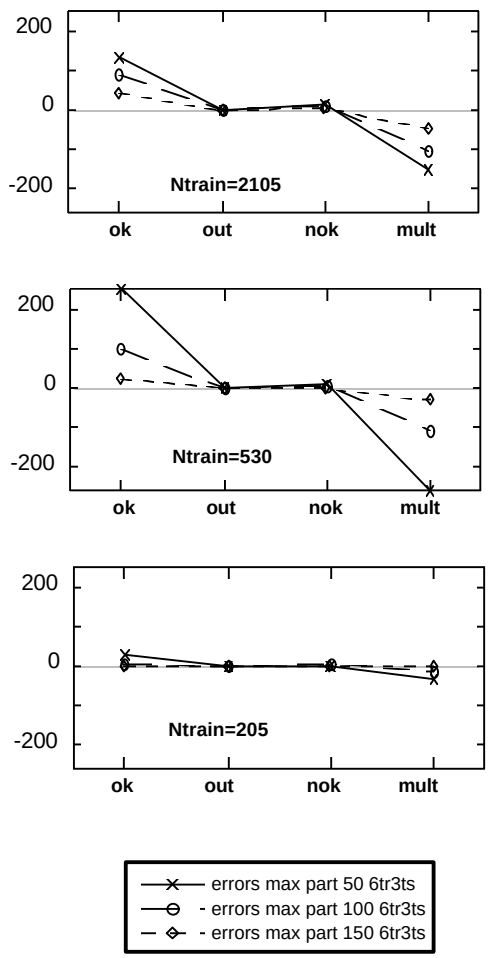

Fig. 31 Discrimination Algo #4 Qualitative Results, 16% Training Data, 6,3 NN

Fig. 31 illustrates PR approximation algorithm #4 qualitative results using the three sparse 16% training sets. Using the training sets with 16% error introduced generated more relative errors than the .1% training data. The smaller partition rectangles generate more errors than the larger rectangles, but the smaller training sets produce less overall errors than the larger training sets. Most errors were from an excess of points that grouped (ok) and less points grouped (mult). These qualitative results are similar to those of PR algorithm #2 and #3.

5.5 Discrimination (PR) Approximation Algorithm #5

In analyzing the runtimes of PR approximation algorithm #4 we found a large fraction of time was devoted to the secondary screening process and to finding and evaluating extra inhibitor candidates (see Fig. 31). In this version the processes of secondary screening of groupings and of finding and evaluating extra inhibitor candidates are not performed.

The method used to find internal and external grouping candidates is the same as in PR algorithm #4, and is described in section 5.4.

5.5.1 Algorithm Runtime Analysis 6,3 NN

Two different sets of results were generated for PR algorithm #5. In one case 6 NN's of a training point and 3 NN's of a test point were used as inhibitor candidates. In the other case, 6 NN's of a training point and 6 NN's of a test point were used as inhibitor candidates.

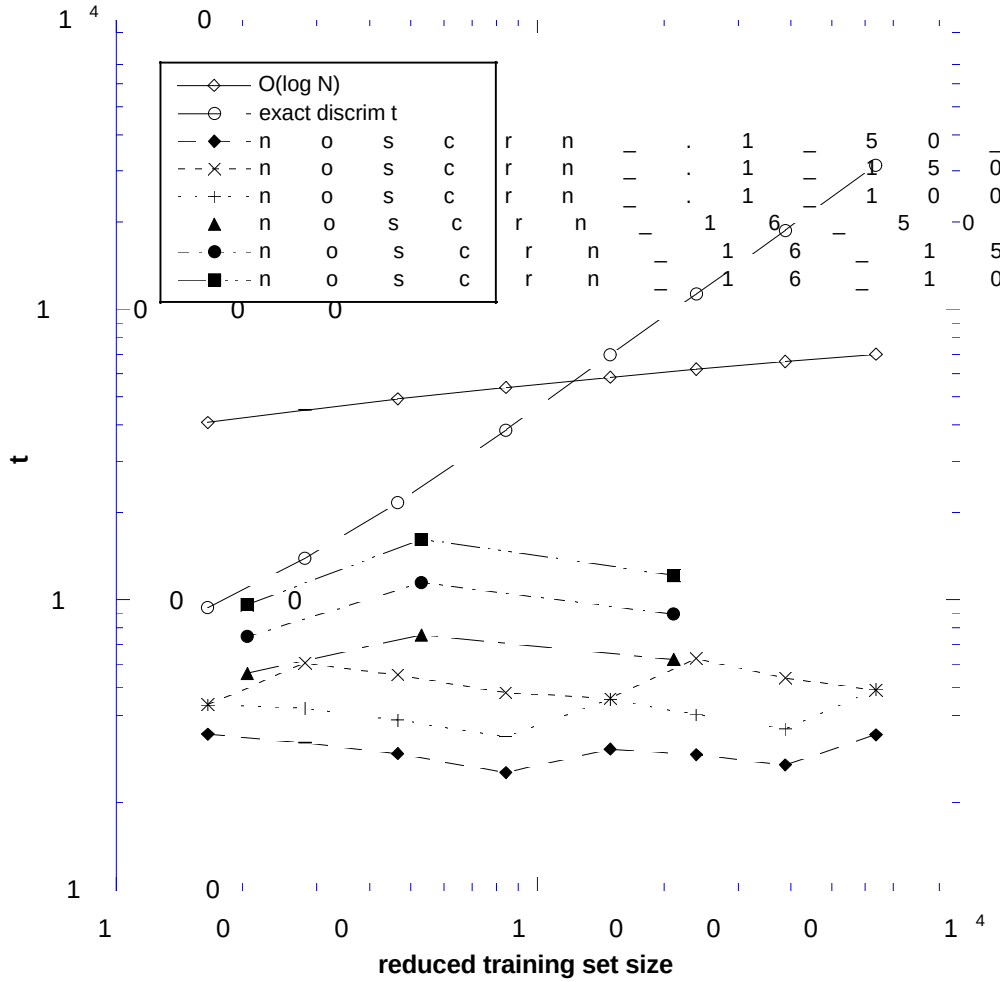

Fig. 32 Discrim Algo #5 Runtime Plot, 6,3 NN

Fig. 32 illustrates PR approximation algorithm #5 example runtimes using maximum partition rectangle sizes of 50,100 and 150 points. In this case 6 NN's of a training point and a dynamic list of training points consisting of 3 NN's of a test point per adjacent rectangle visited were used as the only inhibitor candidates for grouping attempts a test point made. For each training set the same fixed sized test set was classified. Therefore the runtime plot of this test and training set pairs should asymptotically be O(log N_{train}) for sufficiently large N_{train}. The time required to execute the exact discrimination algorithm is shown.

The runtimes of PR approximation algorithm #5 on the 0.1% training data are constant, with larger rectangle sizes requiring more time to complete execution than small rectangles. At times the partition rectangles have the same number of points in them even if different maximum sizes are specified. For this reason some of the timing points in the partition sizes 100 and 150 are similar.

The most important feature to note is that the 16% training data no longer has a runtime slope greater than that of the O(log N_{train}) curve. It ran in constant time. This result is different from the previous versions of the PR algorithm where extra screening of the groupings and searching for more groupers was occurring. This difference can be attributed to not doing the extra work of screening groupings and finding and evaluating extra inhibitors. The approximation PR algorithm was faster than the exact PR algorithm and faster than the other versions with one exception. In the next case of algorithm #5 where 6 NN's of a training point and 3 NN's of a test point were used as the only inhibitors, the time to run partition rectangle size 100 is faster than here.

5.5.2 Qualitative Approximation Results 6,3 NN

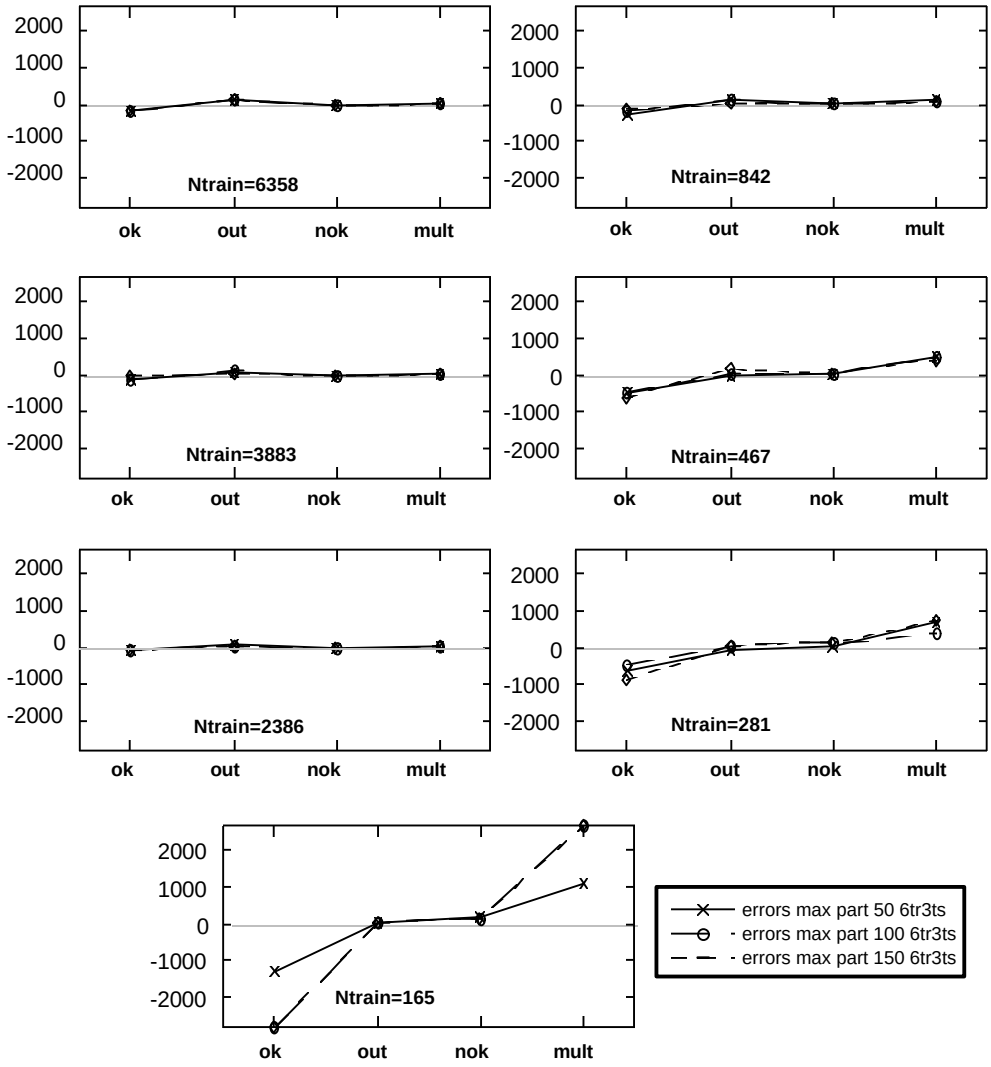

Fig. 33 Discrimination Algo #5 Qualitative Results, 0.1% Training Data, 6,3 NN

Fig. 33 illustrates PR approximation algorithm #5 qualitative results using the seven sparse 0.1% data sets. The figure shows how the results degrade (relative to the exact results) as smaller training sets are used with the larger partition sizes (100,150). More variations in the grouping results occur with PR approximation algorithm #5 than with the other versions.

Less screening allows more groupings to exist in the final result. Most errors were from less points that grouped (ok) and extra points which grouped (mult).

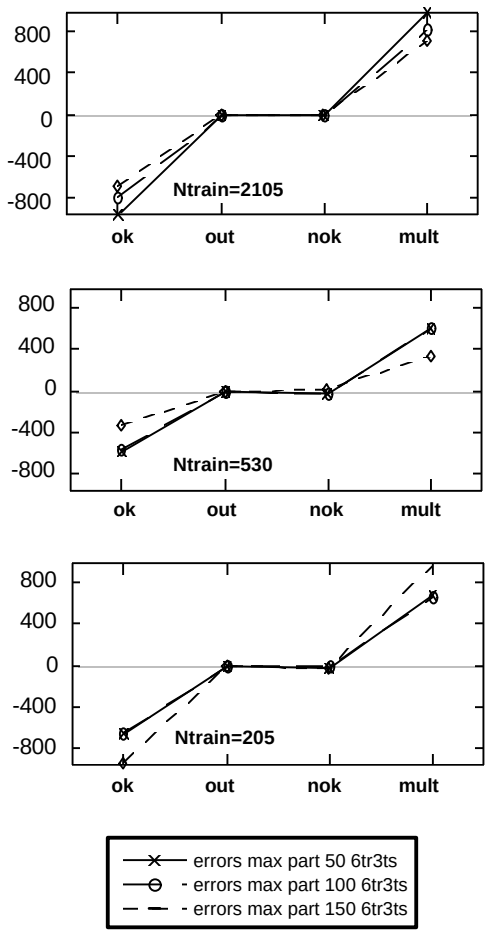

Fig. 34 Discrimination Algo #5 Qualitative Results, 16% Training Data, 6,3 NN

Fig. 34 illustrates PR approximation algorithm #5 qualitative results using the three sparse 16% training sets. The figure shows how the results degrade (relative to the exact results) as smaller training sets are used with the larger partition sizes (100,150). Again, more variations in the grouping results occur with this PR approximation algorithm than with the other versions. Less screening allows more groupings to exist in the final result. Most errors were from less points that grouped (ok) and extra points which grouped (mult).

5.5.3 Algorithm Runtime Analysis 6,6 NN

Two different sets of results were generated for PR approximation algorithm #5. In this other case, 6 NN's of a training point and 6 NN's of a test point were used as inhibitor candidates.

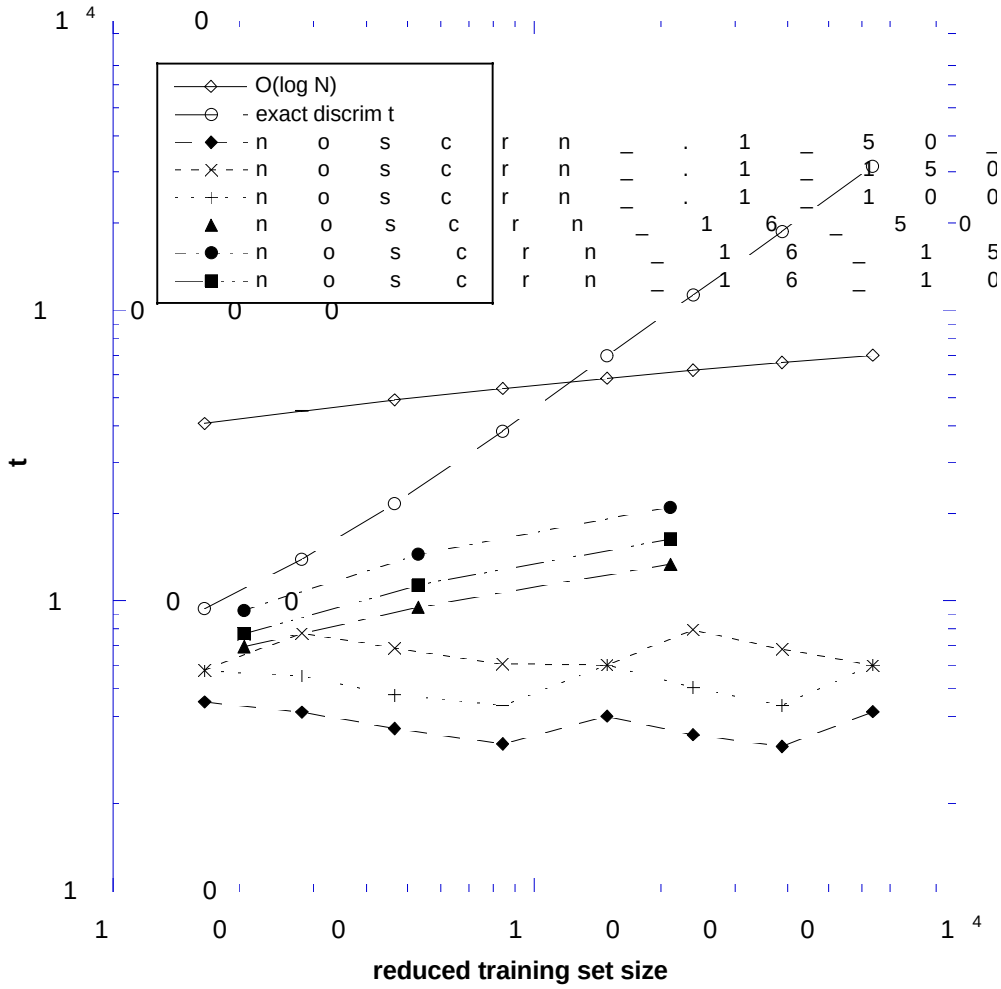

Fig. 35 Discrim Algo #5 Runtime Plot, 6,6 NN

Fig. 35 illustrates PR approximation algorithm #5 example runtimes using maximum partition rectangle sizes of 50,100 and 150. In this case 6 NN's of a training point and a dynamic list of training points consisting of 6 NN's of a test point per adjacent rectangle visited were used as the only inhibitor candidates for grouping attempts a test point made. For each training set the same fixed sized test set was classified. Therefore the runtime plot of this test and training set pairs should be O(log N_{train}) for sufficiently large N_{train} . The time required to execute the exact discrimination algorithm is shown.

The runtimes of PR approximation algorithm #5 on the 0.1% training data are constant, with larger rectangle sizes requiring more time to complete execution than small rectangles. At times the partition rectangles have the same number of points in them even if different maximum sizes are specified. For this reason some of the timing points in the partition sizes 100 and 150 are similar.

The 16% training data has a runtime slope which more closely approximates that of the O(log N_{train}) curve. This result is different from the previous versions of the PR approximation algorithm where extra screening of the groupings and searching for more groupers was occurring. This difference can be attributed to not doing the extra work of finding extra grouping and inhibitor candidates done for each test point classification. PR approximation algorithm #5 was faster than the exact PR algorithm and faster than the other versions of the PR approximation algorithms. This case where 6 NN's of both the test and training sets were used as the inhibitor groupings ran slower than the case with 3 NN's of the test point and 6 NN's of the training point used as inhibitors

5.5.4 Qualitative Approximation Results 6,6 NN

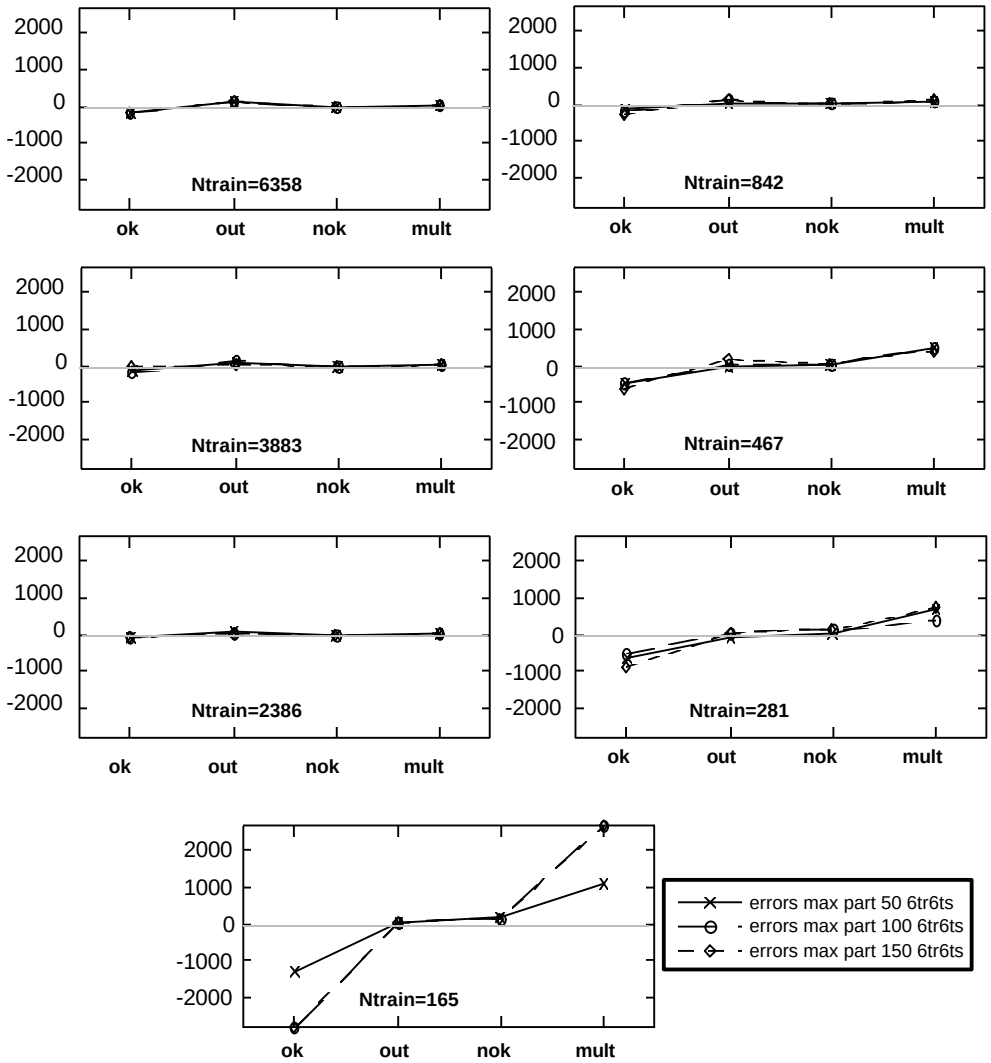

Fig. 36 Discrimination Algo #5 Qualitative Results, 0.1% Training Data, 6,6 NN

Fig. 36 illustrates PR approximation algorithm #5 qualitative results using the seven sparse 0.1% data sets. Fig. 36 shows how the results degrade (relative to the exact results) as smaller training sets are used with the larger partition sizes (100,150). Most errors were from less points that grouped (ok) and extra points which grouped (mult). These qualitative results are similar to the results using 6 and 3 NN's. This suggests that even though more NN's are available as inhibitor candidates, they are not inhibiting many groupings.

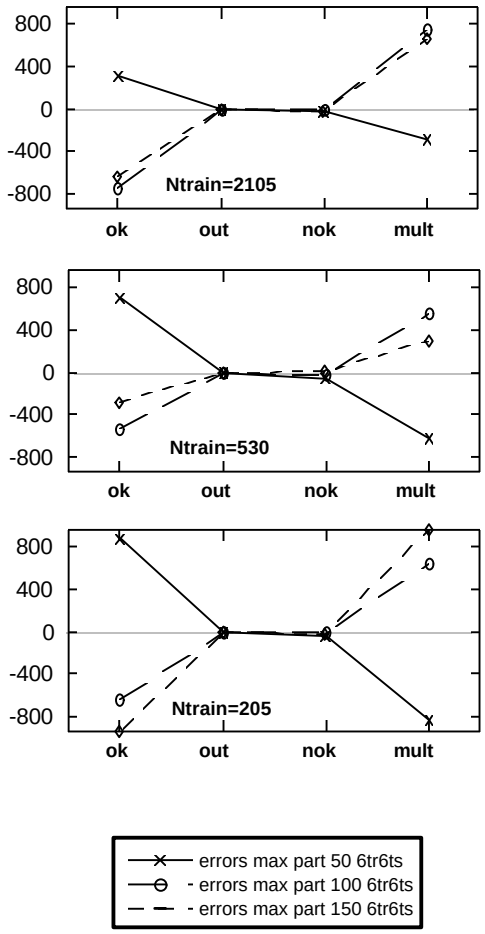

Fig. 37 Discrimination Algo #5 Rualitative Results, 16% Training Data, 6,6 NN

Fig. 37 illustrates PR approximation algorithm #5 qualitative results using the three sparse 16% training sets. The figure shows how the results degrade (relative to the exact results) as smaller training sets are used with the larger partition sizes (100,150).

In this case the smallest partition rectangle (50) found less (mult) classed points and more (ok) points which increased as the training set got smaller. This is different (opposite) to the case using 6 and 3 NN's. This suggests that in the smaller partition rectangles where less points (of different classes) exist, the extra 3 NN's are enough to prevent some of the extra groupings a test point makes. Since there are not many other classes of points in the smaller rectangles a test point cannot be grouped as frequently with multiple classes and therefore the test point that was once classified as (mult) is now is classified (ok). Less screening allows more groupings to exist in the final result. The other errors which occurred were from less points that grouped (ok) and extra points which grouped (mult).

5.6 General Discrimination Algorithm Observations

There were qualitative and quantitative improvements made between PR approximation algorithm #1 and algorithms #2-#4. PR approximation algorithms #2 -#4 perform similarly quantitatively and qualitatively. The changes made to create each version of the algorithm were not significant enough to alter the algorithm performances. PR approximation algorithm #5 runs faster than the other versions, but does not produce the same quality of results. In its worse cases we saw about two orders of magnitude more errors using the 0.1% training sets and about four times more errors using the 16% training data. Any of PR approximation algorithms #2 -#4 will suffice as a good alternative to using the exact PR algorithm, but will provide no speedup in runtimes unless the training set are sufficiently large, where N_{train} >= 3000 points.

6. Conclusions

Several algorithms were developed to improve the runtime performance of clustering and PR problems. An algorithm to reduce the number of points in a training set was developed. With this reducing algorithm smaller training sets can be generated with spatial and qualitative properties similar to the unreduced training set. Faster exact versions of PR and clustering algorithms were developed. Quantitative and qualitative results of the exact and approximated clustering and PR algorithms operating on a variety of data sets were illustrated. With this information about the algorithms, users can now reference this document and determine runtimes for their PR and clustering problems. With the quantitative data of the approximation algorithms, users can determine if an exact or approximated solution is needed and decide which algorithm to use to find solutions for problems at hand.

References

1 G.C. Osbourn and R.F. Martinez**,** EMPIRICALLY DEFINED REGIONS OF INFLUENCE FOR CLUSTERING ANALYSIS**,** Pattern Recognition, Vol. 28, No. 11, 1793-1806 (1995).

2 G.C. Osbourn, Unpublished.

3 Sedgewick, Robert, Algorithms, Addison Wesley (1988).