Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 99-029

Unsupervised Clustering: A Fast Scalable Method for Large Datasets

Daniel Boley and Vivian Borst

July 23, 1999

Unsupervised Clustering: A Fast Scalable Method for Large Datasets*

Daniel Boley and Vivian Borst
Department of Computer Science and Engineering
University of Minnesota

Abstract

Fast and effective unsupervised clustering is a fundamental tool in unsupervised learning. Here is a new method to explore large datasets that enjoys many favorable properties. It is fast and effective, and produces a hierarchical structure on the underlying dataset, without using a training set. It also yields auxiliary information on the significance of the different attributes.

1 Introduction

Explosive growth in the volume of data available electronically has created a need to be able to automatically explore large data collections. Unsupervised clustering algorithms are classical tools which have increasingly been reexamined for their applicability to data mining efforts.

Ideally, these algorithms would be fast and scalable, require little or no a-priori understanding of the data contents or attributes and need no costly graph building or association rule preprocessing. In many applications it would be useful if the algorithm could also impose a natural hierarchy on the data set, compute properties for the set as a whole and handle cases where attribute information is missing. Principal Direction Divisive Partitioning (PDDP) is such an algorithm [5]. Originally applied in the context of text documents retrieved from the WWW as part of the WebACE project [3], PDDP has proven to be computationally efficient while providing high quality clusters. In addition to producing a partitioning of the data, the PDDP method

- yields weights showing which words were most significant in distinguishing one document cluster from another,
- implements an automated stopping test based on the distribution of the data,
- allows a straightforward way to process datasets with missing attribute values, and

*This work was partially supported by NSF grant IIS-9811229.

- generates a hierarchical tree of clusters which can easily be updated locally.

The method has been successfully applied in a variety of application domains in addition to text documents, such as vision-based texture analysis and movie recommendation services.

2 Algorithm Description

The PDDP algorithm employs the vector space model, where each data sample is represented by a vector of numerical attribute values. The data samples are embedded in a very high dimension Euclidean space, and the algorithm partitions this space with a collection of hyperplanes calculated to achieve good separation among the data samples. The data space is separated by a hyperplane into two half-spaces. The process continues recursively by separating each half-space with new hyperplanes computed independently. The method builds a binary tree of many polytope regions from the top down, until a stopping test is satisfied.

Since the PDDP algorithm operates directly with the collection of numerical attribute vectors, only a limited amount of preprocessing is necessary to generate the input data necessary for PDDP. This method was originally developed as part of the WebACE Project [3] in the context of text documents where each document is represented by a scaled vector of word counts. The preprocessing consisted of removing the stop words and common word endings, and counting the number of occurrences of each word in each document. The result was a vector $\mathbf{d}$ of word counts associated with each document. All these vectors were combined into a single matrix $\mathbf{M}$ in which each column corresponded to a document and each row corresponded to a particular word. In this domain, the matrix was generally very sparse, often less than 1% of the entries were nonzero. This sparsity results in a very fast and memory efficient method for carrying out the splitting process.

The clustering via PDDP is a recursive process that operates directly on the matrix $\mathbf{M}$. PDDP starts with a single "cluster" encompassing the entire dataset, divides this cluster into subclusters recursively using a two step process. At each stage, PDDP (a) selects a cluster to split, and (b) splits that cluster into two subclusters which become children of the original cluster. The result is a binary tree hierarchy imposed on the data collection. At every stage, the leaf nodes in the tree form a partition of the entire data collection. In the process of going to the next stage, one of those leaf nodes is selected and split in two. The behavior of the algorithm is controlled by the methods used to accomplish steps (a) and (b), and these methods are independent of one another. For step (a), PDDP usually selects the cluster with the largest *scatter value*, which is the sum of all the squared distances from each document to the cluster centroid $\mathbf{w}$, though any suitable criterion can be used.

Once a node is selected in step (a), it is split in step (b), and this splitting process is the single most expensive step in the whole computation. The key to the computational efficiency of the entire approach is the efficient computation of the vectors needed in this step. Suppose PDDP were to split cluster `C` consisting of k documents of word counts. It places each document $\mathbf{d}$ in the left or right child of cluster `C` according to the sign of the

Algorithm PDDP.
0. **Start** with $n \times m$ matrix $\mathbf{M}$ of (scaled) document vectors,
   and a desired number of clusters $c_{\max}$.
1. **Initialize** Binary Tree with a single Root Node.
2. **For** $c = 2, 3, \ldots, c_{\max}$ **do**
3. **Select** leaf node C with largest `scatter value`,
   and L & R := left & right children of C [step (a) in the text].
4. **Compute** $\mathbf{v}_{\mathtt{C}} = g(\mathbf{M}_{\mathtt{C}}) \equiv \mathbf{u}_{\mathtt{C}}^T(\mathbf{M}_{\mathtt{C}} - \mathbf{w}_{\mathtt{C}}\mathbf{e}^T)$
5. **For** $i \in$ C, if $v_i \leq 0$, assign document i to L,
   else assign it to R [step (b) in the text].
6. **Result:** A binary tree with $c_{\max}$ leaf nodes
   forming a partitioning of the entire data set.

Figure 1: Summary of the method to do a full build of the PDDP tree from scratch.

linear discriminant function

$$g_{\mathtt{C}}(\mathbf{d}) = \mathbf{u}_{\mathtt{C}}^T(\mathbf{d} - \mathbf{w}_{\mathtt{C}}) = \sum_{i=1}^{n} u_i(d_i - w_i), \tag{1}$$

where $\mathbf{u}_{\mathtt{C}}$, $\mathbf{w}_{\mathtt{C}}$ are vectors associated with C to be determined. If $g_{\mathtt{C}}(\mathbf{d}) \leq 0$, the document $\mathbf{d}$ is placed in the new left child, otherwise $\mathbf{d}$ is placed in the new right child. Thus the behavior of the algorithm at each node in the binary tree is determined entirely by the two vectors $\mathbf{u}_{\mathtt{C}}$, $\mathbf{w}_{\mathtt{C}}$ associated with the cluster C.

The vector $\mathbf{w}_{\mathtt{C}} \stackrel{\text{def}}{=} (1/k)\sum_j \mathbf{d}_j$ is the *mean* or *centroid* vector. The vector $\mathbf{u}_{\mathtt{C}}$ is the direction of maximal variance. This direction corresponds to the largest eigenvalue of the sample covariance matrix for the cluster. The computation of $\mathbf{u}_{\mathtt{C}}$ is the most costly part of this step. It can be performed quickly using a Lanczos-based solver for the singular values of the matrix of documents in the cluster. This algorithm is able to take full advantage of the fact the matrices are extremely sparse, often with less than 1% of its entries nonzero.

The overall method can be summarized in Figure 1. As the method is "divisive" in nature, splitting each cluster into exactly two pieces at each step, the result is a binary tree whose leaf nodes are the sought-after clusters.

To demonstrate the PDDP algorithm and introduce its performance properties, consider the "iris" data collection (referred to in [7, p218]), which consists of 150 flowers: numbers 1-50 are of type *setosa*, numbers 51-100 *versicolor*, and number 101-150 *virginica*. To illustrate the binary tree on a simple case, data from 6 flowers in the set were chosen : `1, 2, 51, 52, 101, 102` with attributes shown in Fig. 2.

Fig. 3 shows the binary tree that results when the PDDP algorithm is used to to split this collection of six flowers into 3 clusters. The top box in Fig. 3 represents the root: it contains the indices of all six flowers. The column headed *centroid* is the centroid vector for all six flowers, and the column headed *direction* is the principal direction vector for all six flowers. The root has two children, one of which is a leaf node. In the leaf nodes, the

flower	1	2	51	52	101	102
type	setosa	setosa	versicolor	versicolor	virginica	virginica
sepal length	5.10	4.90	7.00	6.40	6.30	5.80
sepal width	3.50	3.00	3.20	3.20	3.30	2.70
petal length	1.40	1.40	4.70	4.50	6.00	5.10
petal width	0.20	0.20	1.40	1.50	2.50	1.90

Figure 2: Raw attributes for six flowers selected to illustrate the PDDP algorithm in Fig. 3.

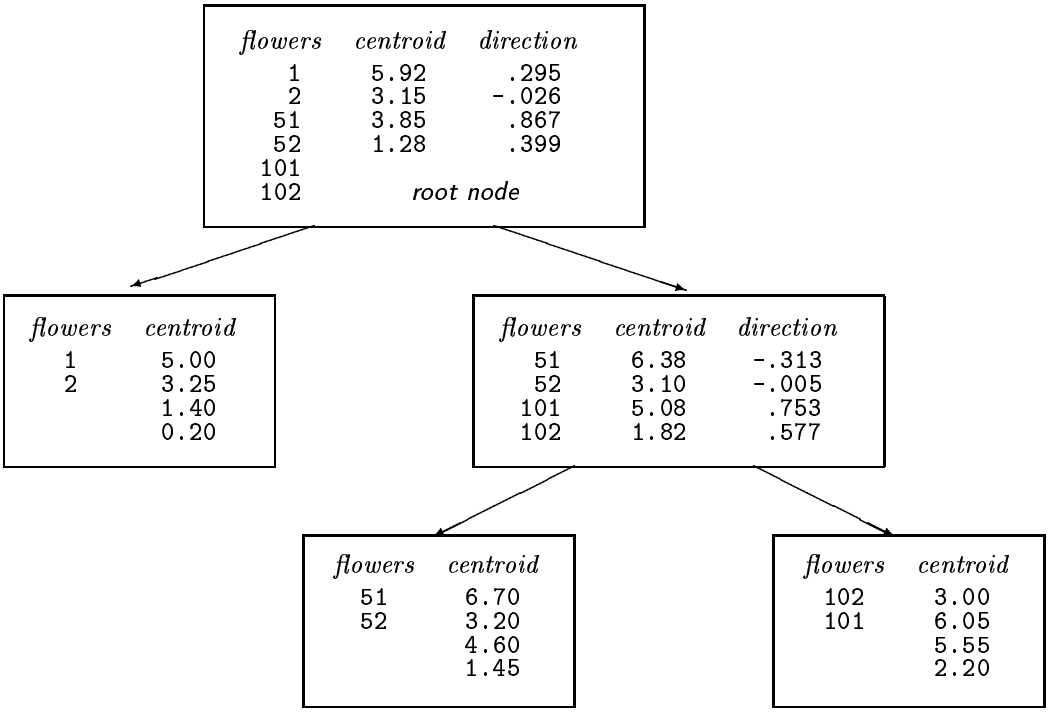

Figure 3: PDDP binary tree generated from the six irises shown in Fig. 2. Each box represents a node, listing the indices of the flowers represented by that node together with the average values of the attributes over all flowers in that node ("*centroid vector*"). In the two non-leaf nodes, the "*principal direction vector*" is shown, which is not computed at all for the leaf nodes.

cluster:	1	2	3	4
setosa	50	0	0	0
versicolor	0	46	2	2
virginica	0	0	21	29

Figure 4: PDDP Clustering result on the entire dataset of 150 irises, using the scatter-based automated stopping test and norm scaling on the input data. Each entry in the table is a count showing the number of flowers of that type in the computed PDDP cluster. If the algorithm had been artificially stopped at three clusters, the three clusters would have been 1, 2, and (3+4).

method	*entropy*
Agglomeration - norm scaling [7]	0.6843
PDDP - norm scaling	0.6895
Hypergraph [8]	0.7874
K-means - LSI [1]	0.8370
PDDP - TFIDF scaling [10]	1.0576
AutoClass [6]	2.0497
Agglomeration - TFIDF scaling	2.3393

Figure 5: Entropies by various methods on a set of 185 documents with a 10538 word dictionary.

principal direction vectors are not computed because they are not needed. For the non-leaf nodes, shown are the centroid and principal direction vector for the four flowers `51, 52, 101, 102`. In this simple case, the algorithm was able to partition the flowers consistently with their types. At each stage, the scatter value was used to select the next node to split, and an automatic stopping test employed using this scatter value. Though the scatter values are not shown in Fig. 3, the interior non-leaf node had a higher scatter than its "sibling" leaf node. Using the same stopping test on the entire dataset, the algorithm yielded 4 clusters shown in Fig. 4.

3 Quality of Clusters

Performance testing has demonstrated PDDP provides high quality clusters at a relatively low computational cost. Most of the experience has been on text documents, but the algorithm has also been successful on datasets of movie ratings, texture images, toxicity databases, etc. On a document set of 185 documents using a dictionary of 10538 words, PDDP was compared against several other algorithms, and the results are shown in Fig. 5. The quality of the clusters were estimated by computing an entropy measure using the same methods and labels as in [5, 4]. Each document was manually given a label according to the topic of the document. These labels were not used by any of the classification methods discussed in this paper or for any of the performances tests. The labels were used only to measure the entropy of the resulting clusters as a measure of quality. The entropy of a given cluster $\mathtt{C}$ is defined by

$$e_{\mathtt{C}} = -\sum_i \left(\frac{c(i,\mathtt{C})}{\sum_i c(i,\mathtt{C})} \right) \cdot \log \left(\frac{c(i,\mathtt{C})}{\sum_i c(i,\mathtt{C})} \right),$$

where $c(i,\mathtt{C})$ is the number of times label i occurs in cluster $\mathtt{C}$. The entropy for a cluster is zero if the the labels of all the documents are the same, otherwise it is positive. The total entropy is the weighted average of the individual cluster entropies:

$$e_{\text{total}} = \frac{1}{m} \sum_{\mathtt{C}} e_{\mathtt{C}} \cdot (\text{number of documents in cluster } \mathtt{C}).$$

As a consequence, the lower the entropy the better the quality.

cluster:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
business	90	0	0	0	7	0	5	12	0	6	0	1	18	3	0	0
health	0	150	166	171	3	0	1	1	0	0	0	0	0	2	0	0
politics	2	0	0	0	100	1	2	0	0	1	0	2	1	5	0	0
sports	0	0	0	0	1	62	35	0	0	1	0	0	0	42	0	0
technology	8	0	0	0	0	1	14	24	0	8	0	1	4	0	0	0
entertain.	24	0	0	4	11	4	22	61	135	131	148	159	143	137	204	206

Figure 6: Table showing how documents were distributed to 16 different clusters by topic labels. This data set has 2340 documents, and was based on a dictionary of 21839 words. Further subdivisions are necessary to classify the large collection of entertainment documents in this set.

It is difficult to compare PDDP with the more classical methods because, unlike PDDP, most classical methods do not scale linearly with the size of the problem. For example, in Fig. 5, the PDDP, Hypergraph and LSI methods took under 2 minutes on a Sun workstation, but Agglomeration and Autoclass each took at least 30 minutes. On larger datasets, unmodified Agglomeration and Autoclass became prohibitively expensive. There is no absolute scale to measure the quality of clusters, but one can see in Fig. 6 how documents in a larger dataset consisting of 2,340 text documents in six broad categories were distributed among 16 clusters.

4 Performance

The cost of the PDDP method depends almost entirely on the cost of its most expensive step, which is the computation of the principal direction vector. The computation of this

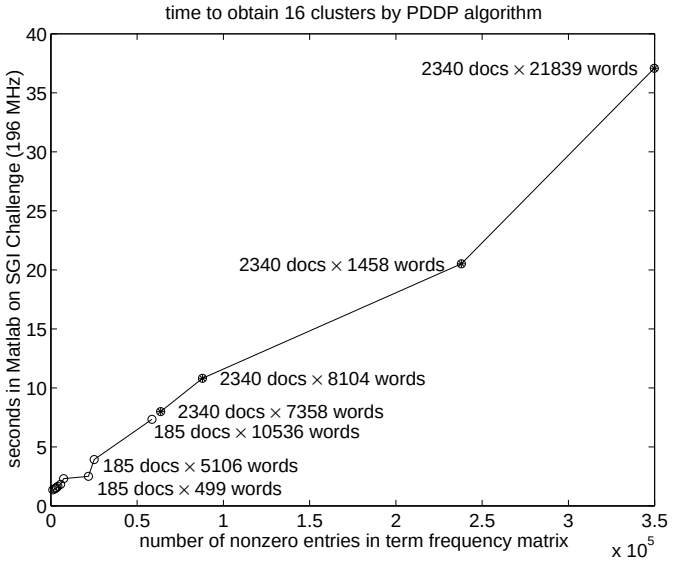

Figure 7: Time to compute 16 clusters for various datasets using interpreted MATLAB code on an SGI challenge workstation (196 MHz), against the number of nonzeroes in the data matrix.

vector is carried out with a Lanczos-based eigensolver, as already mentioned, whose cost is proportional to the number of nonzeroes in the data matrix. Thus the PDDP method scales linearly with the size of the data matrix. This behavior is shown in Fig. 7, where it is seen that the cost depends more on the number of nonzeroes than the actual number of documents or words. For example, Fig. 7 shows that the time to obtain the clusters shown in Fig. 6 was approximately 37 seconds on an SGI Challenge workstation.

5 Inherent Properties

The basic PDDP algorithm is extremely flexible and extendible, and exhibits several inherent favorable properties. The primary properties of scalability and the quality of clusters generated have already been discussed. But as an immediate consequence of the top-down hierarchical nature of the algorithm, it enjoys several other favorable properties.

Identifying Leading Significant Data Attributes. Each node in the PDDP's binary tree represents a cluster of data samples, and associated with each such node is a centroid vector $\mathbf{w}$ and principal direction vector $\mathbf{u}$. These two vectors yield a lot of information about the significance of the individual attributes when used as a basis to structure the underlying dataset. Each entry in $\mathbf{w}$ is an average attribute value among samples in that cluster. In the context of word documents, the largest entries in $\mathbf{w}$ correspond to the most common words through out the whole cluster. On the other hand, when placing a given sample in a left as opposed to a right cluster, the terms in the discriminant function $g_{\mathtt{C}}(\mathbf{d})$ of equation (1) that are most significant are those terms with the largest coefficients u_i. Each such term corresponds to a particular attribute value (i.e. word in the context of text documents) Hence the largest entries in the $\mathbf{u}$ vector correspond to those attributes that are most significant in placing data samples in one child cluster as opposed to the other. For example, for text documents these significant words can be much more informative of what is in a cluster of documents samples than the most common words, as illustrated in Fig. 8.

Building a Hierarchical Taxonomy. The significant words found in the principal direction vectors can be used to construct a hierarchical taxonomy of a dataset. At each level in the tree, PDDP can label each cluster with the words that were most significant in generating that cluster, as done in Fig. 8. Those labels can be displayed to the user. Fig. 9 shows a sample for a large dataset. This can be incorporated into a web browser or web agent to organize a collection of documents viewed by the user, or to generate queries to a web search engine to find new documents related to the set already viewed.

Automating the Stopping Test. The quantities computed during the course of the PDDP algorithm lead naturally to an automated stopping test that captures, to a limited extent, the groupings in the data. A popular rule of thumb is to compute a number of clusters equal to the square root of the total number m of data samples. An alternative is to stop when the largest cluster contains no more than a given number of data samples, for example $\sqrt{m}$. However, the scatter values already computed during the PDDP algorithm already reflect the "clumpiness" of the data samples in Euclidean space. So

vector	*leading words generated by PDDP for this cluster*
centroid	help entertai health new polit sport tech bize index
principal dir.	dengue fever dna gene gold probe diseas gubler guttman

The 12 raw headlines embedded within each document in the cluster.

+Enzyme Triggers Emphysema in Smokers
+Drug May Offer New Way to Beat Colds
+Lens Coat Cuts Cataract Cost
+Gold Probe Detects DNA
+"Dry Brushing" Beats Dental Plaque
+Dengue Fever Strikes "Dr. Quinn"
+Pregnancy Factor Protects Fetus
+New Doping Agent for Athletes Reported
+Dengue Fever Strikes "Dr. Quinn"
+Gold Probe Detects DNA
+FDA Moves to Ban Laxative Ingredient
+Muscles Adapt to Exercise Lifestyle

Figure 8: Sample cluster generated by PDDP from a set of news articles, showing the original document headlines and the leading words extracted by PDDP from the centroid and principal direction vectors. Notice the principal direction words are better indicators of the document topics, compared to the centroid words. Some articles were repeated on later dates with slightly different contents. Many of the leading words in the centroid arise from WWW navigation bars within each document, and the principal direction vector is able to eliminate those automatically.

```
(829) sport.bize.new.health.index.scoreboard.polit.tech.previou.world
(487) film.festiv.annual.intern.hollywood.angel.lo.award.director.upcom
(491) studi.research.risk.cancer.patient.cell.heart.diseas.women.drug
(533) compani.internet.stock.market.million.house.comput.busi.percent.servic
```

Figure 9: Fragment of a list of hierarchical cluster labels generated by the PDDP algorithm for 4 "interior" clusters on the way to the situation of Fig. 6. The number in parentheses is the size of each cluster.

a stopping test based on the scatter value reflects this "clumpiness." This leads to the stopping test used in these experiments in the cases where the number of clusters were not fixed in advance. As the PDDP algorithm proceeds, the clusters get smaller and hence the maximum scatter value in any individual cluster get smaller. The stopping test ends the processing when the maximum scatter value decreases to below a given threshold. In order to make the threshold automatically reflect the distribution of data samples within the particular data set, the threshold is not fixed. Rather, a dynamic threshold based on a so-called *centroid scatter value* is used. The centroid scatter value is the scatter value computed on the collection of centroids treated as individual attribute vectors. This value is a dynamic measure of the overall distribution of the data samples within the attribute space. As the PDDP algorithm proceeds, the centroid scatter value increases while the maximum cluster scatter value decreases. The method terminates when the centroid scatter value exceeds the maximum cluster scatter value at any particular point. Experience in all the domains mentioned in this paper show that this stopping test appears to be effective in determining a reasonable number of clusters, though there is generally no objective measure on the "optimal" number of clusters, unless the data samples have been previously categorized or labeled.

Accommodating Missing Values. Often attribute vectors contain entries that are missing, and it is inappropriate to bias missing values toward one end of the scale or the other. Treating missing values as zero, for example, would tend to treat them as closer to certain valid attribute values as opposed to some others. The PDDP algorithm employs a simple device to avoid this bias. At each stage in the splitting process, PDDP computes the average value for each attribute, but each average is computed only over the valid attribute values. The result is a centroid vector in which each entry represents only existant attribute values. PDDP then considers each missing value to be equal to the average for that attribute. For example, let $\mathbf{d}$ be the attribute vector for a particular sample, and suppose that $\mathbf{w}$ in equation (1) is computed only over the valid attribute values. If the particular entry d_i is missing, then d_i is temporarily filled in with w_i. The result is that the i-th term in the sum in equation (1) is zero and hence makes no contribution to the decision about placing the sample in the left versus the right child. To keep the missing attribute value from biasing later splittings within the PDDP process, this temporary filling in is carried out again at every stage in the PDDP process. The result is that the missing values never push any sample to one side or the other. Of course the text document domain is not the best domain on which to apply this technique, since the lack of a word in a document is just as significant as the presence of a word; however, this technique has been used successfully in other domains (see for example §6).

Incorporating New Data. The binary tree constructed by the PDDP algorithm can be used to incorporate new data samples not available at the time the tree is initially constructed. This three structure allows for update operations to be performed on a subset of the data, localizing adjustments to specific subtrees.

Each node in the tree represents a cluster of data samples, and the PDDP algorithm stores the centroid vector and the principal direction vector for each cluster in each

node. Given a new data sample $\mathbf{d}_{\text{new}}$, it is a simple matter to percolate the data sample down the tree starting at the root. At each node in the tree, the function $g_{\mathtt{C}}(\mathbf{d})$ of equation (1) is applied using the centroid $\mathbf{w}$ and principal direction $\mathbf{u}$ associated with the current node. The resulting sign of $g_{\mathtt{C}}(\mathbf{d})$ is used to place the new data sample in the left or right child node. This computation is repeated at each level until it reachs a leaf node.

It is even possible to dynamically update the tree, as proposed in [2]. Comparing the original and modified scatter values in the nodes which have received new data samples, a subtree is re-partitioned dynamically. Results in [2] indicate that the quality of clusters obtained this way are close to those obtained by a just repeating the "full-build," but at considerable savings.

6 New Application Domains

There is nothing intrinsic about the PDDP method that restricts its use to text documents. It can be applied to any domain in which the attribute values can be represented numerically. In different domains the attribute values vary over different ranges, and different scalings can have a marked effect on the performance of the PDDP algorithm. However, in most cases, quite satisfactory results have been obtained using a relatively simple norm scaling, in which each attribute vector is scaled to unit length. The behavior of many clustering algorithms, especially Hierarchical Agglomeration, depends critically on both the scaling used and a "distance function" between two attribute vectors or clusters. The PDDP method is also based on an embedding of the attribute vectors in a high dimensional Euclidean space, hence the scaling is also important for the PDDP, but the "distance function" is not used during the splitting process (step (b)). Below are two non-text domains where the PDDP method has been successfully applied.

Vision-based Texture Analysis. The PDDP method has been used to classify images according to their texture using a very simple gray level co-occurrence representation. For example, the 20 images in Fig. 10 were encoded by a simple horizontal co-occurrence matrix, in which the $i-j$-th entry is the number of times a pixel of intensity i is immediately to the left or right of a pixel of intensity j, where the pixel intensities vary from 0 to 255. The resulting 256×256 matrix for each image was reshaped as a attribute vector of 256^2 elements. The collection of all these attribute vectors were then clustered by the PDDP algorithm in the same way as a collection of text documents. When asked to find five clusters on the example of Fig. 10, the PDDP method with this straight forward encoding was able to separate the five types of images perfectly. When using the automatic scatter-based stopping test, the PDDP clustering method continued for just one more step, splitting the cluster containing the quilt images in two for a total of six clusters. Experiments show that PDDP has some success is distinquishing textures that vary only in magnitude. Further tests are currently being carried out a larger data set of linen images with a variety of different weaves.

Movie Ratings. Automated recommendation applications, like those used by Amazon.com, constitute large data sets where missing attribute values predominate. These recommendation services offer individualized predictions of user preferences based on the history of ratings of that individual and those of other users with similar patterns of ratings. PDDP was tested in one such service, a movie recommendation application, Movielens [9]. Given a matrix of movie ratings by many users, in which most of the entries are missing, it is possible to cluster users based on their movie preferences. In the same way as with text documents, this process yields a list of the movies that are most significant in distinguishing users one from another. When this was carried out recently on a set of 943 users and 1682 movies with 100,000 ratings (the remaining 94% of the entries were missing), the 10 movies most useful in distinguishing users were found to be:

- Independence Day (ID4) (1996)
- Mission: Impossible (1996)
- Raiders of the Lost Ark (1981)
- Toy Story (1995)
- Terminator 2: Judgment Day (1991)
- Back to the Future (1985)
- Rock, The (1996)
- True Lies (1994)
- Return of the Jedi (1983)
- Indiana Jones and the Last Crusade (1989)

It is also possible to cluster the movies by transposing the data matrix. The result is a collection of meaningful subsets of movies. This was used to reduce the computation space of Movielens' predictive algorithm. The prediction accuracy of using the PDDP movie clusters was compared to that of clusters generated randomly, by movie genre, and with hypergraph partitioning. In an initial set of tests, where the number of clusters varied from two to twelve, PDDP performed as well as the best of the other methods. Performance was estimated by comparing the prediction, computed using a subset of data, against the eventual real user rating. This evaluation is still continuing.

7 Conclusion

Many classical unsupervised clustering algorithms have suffered from a trade-off between speed and quality of the resulting clusters. The result is that such algorithms have not seen a wide application even in the face of the recent explosion of available electronic data. Classically, large datasets have been explored by extracting a small subset of data for use as a training set, or by extracting a small subset of attributes to reduce the dimensionality, or both. A fast, effective method could revolutionize the way such large bodies of data are explored by allowing one to analyse a dataset as a single large collection. The properties exhibited here for the PDDP algorithm show that the PDDP method can often be effective when used on a whole dataset, even when it is reasonably large, yielding more meaningful results with little preprocessing. To date, the PDDP method has been limited by its implementation: it is still implemented in its original development form as interpreted code in Matlab. Future performance as a compiled production code can only be estimated from the current results, but should be very promising.

References

[1] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37:573–595, 1995.

[2] D. Boley and V. Borst. Unsupervised updating of a classification tree in a dynamic environment. In *Autonomous Agents'99 Conference*, 1999. to appear.

[3] D. Boley, M. Gini, R. Gross, E.-H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Document categorization and query generation on the World Wide Web using WebACE. *AI Review*, 1999. to appear.

[4] D. Boley, M. Gini, R. Gross, S. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Partitioning-based clustering for web document categorization. *Decision Support Systems*, 1999. to appear.

[5] D. L. Boley. Principal Direction Divisive Partitioning. *Data Mining and Knowledge Discovery*, 1999. to appear.

[6] P. Cheeseman and J. Stutz. Bayesian classification (Autoclass): Theory and results. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI/MIT Press, 1996.

[7] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[8] E. Han, G. Karypis, V. Kumar, and B. Mobasher. Hypergraph based clustering in high-dimensional data sets: A summary of results. *Bulletin of the Technical Committee on Data Engineering*, 21(1), 1998.

[9] J. Riedl et al. Movielens. `http://movielens.umn.edu/`, 1998.

[10] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

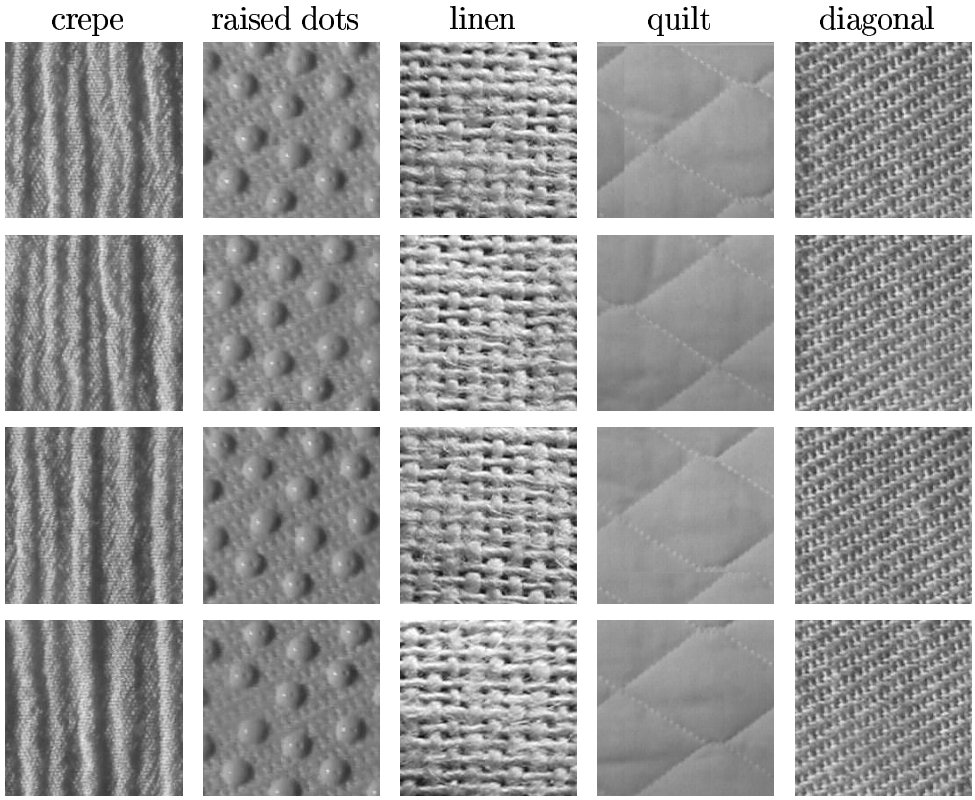

Figure 10: The twenty textile images PDDP was able to successfully cluster according to texture. The images were partitioned using their gray level frequencies that were encoded in a gray-scale co-occurrence representation.