An Unsupervised Clustering Tool for Unstructured Data*

Daniel Boley, Vivian Borst, Maria Gini
Department of Computer Science and Engineering, University of Minnesota
200 Union Street S.E., Minneapolis, MN 55455, USA

Abstract

We present an unsupervised clustering tool, Principal Direction Divisive Partitioning, which is a scalable and versatile top-down method applicable to any set of data that can be represented as numerical vectors. A description of the basic method, a summary of the main application areas where this has been used, and some recent results on the selection of significant words as well as the process of updating clusters as new data arrives is discussed.

1 Introduction

One key component of information filtering is to retrieve entries whose attributes satisfy certain properties. A structure imposed on an initially unstructured dataset can be used to localize the search for relevant data entries or to split the data so that the search can be carried out in parallel on multiple processors. The structure can also be the basis for organizing the presentation of the information to a user, allowing her to easily focus on the particular items desired.

This was the original goal of the agent WebACE [Boley *et al.*, 1998] for the management of large numbers of documents retrieved off the Web. WebACE is capable of automatically organizing documents that a user might select, including documents retrieved automatically by the agent during autonomously directed searches. This is an example of an application where gains could be enjoyed by imposing structure on a large unstructured dataset automatically. A prototype implementation of this agent, described in [Boley *et al.*, 1998], is based on the use of a client-side proxy server to collect URLs of web pages retrieved by a user in the usual way during an ordinary browsing session. Using the methods of this paper, the agent would then cluster the URLs, and the resulting topical groups would be presented to the user on a newly constructed web page.

The basis for the structuring process is the method of Principal Direction Divisive Partitioning (PDDP) [Boley, 1998]. This is a highly scalable and effective unsupervised method for partitioning data samples in a high dimensional attribute space. Initially developed in the context of document corpora to be classified based on text content, the method has been demonstrated successfully on datasets of widely different types and sources, including image texture, chemical toxicology, genome sequences, etc.

2 Algorithm Description

The original method starts with a root "cluster" containing the entire sample set of text documents. PDDP splits each cluster, halving it across the cluster's principal direction, and produces two "child" clusters. The algorithm successively splits these "child" clusters until some stopping test is met. The leaf nodes of the resulting binary tree are the sought after classification clusters and constitute a partition of the entire collection. Each stage of the algorithm consists of (i) selecting the cluster to split based on a cluster scatter value, and (ii) splitting that cluster into two child clusters. The key to the speed of the entire approach is the efficient computation of part (ii). The overall method is summarized in Fig. 1.

Each document is represented by a vector $\mathbf{d}$ of word frequencies, scaled to unit length, making the results independent of document length. This scaling is called the *norm scaling*. An alternate scaling is the TFIDF scaling defined using the recommended "nfc" alternative from [Salton and Buckley, 1988].

A cluster $\mathtt{C}$ containing k documents is represented by an $n \times k$ matrix $\mathbf{M}_\mathtt{C} = \{\mathbf{d}_1, \ldots, \mathbf{d}_k\}$, where n is the number of words. We define a linear discriminant function

$$g_\mathtt{C}(\mathbf{d}) = \mathbf{u}_\mathtt{C}^T(\mathbf{d} - \mathbf{w}_\mathtt{C}) = \sum_i u_i^\mathtt{C} \cdot (d_i^\mathtt{C} - w_i^\mathtt{C}), \qquad (1)$$

where $\mathbf{w}_\mathtt{C}$, $\mathbf{u}_\mathtt{C}$ are, respectively, the cluster centroid and the direction of maximal variance (*principal direction*) vectors associated with each node cluster $\mathtt{C}$. Specifically, $\mathbf{w}_\mathtt{C} \stackrel{\text{def}}{=} \mathbf{M}_\mathtt{C}\mathbf{e}^{[k]}/k$, where $\mathbf{e}^{[k]}$ is a vector of all ones of dimension k, and $\mathbf{u}_\mathtt{C}$ is the left singular vector corresponding to the largest singular value of $\mathbf{A} \stackrel{\text{def}}{=} \mathbf{M} - \mathbf{w}_\mathtt{C}(\mathbf{e}^{[k]})^T$. The linear discriminant function is used to define the splitting of the cluster: each document $\mathbf{d}$ is placed in the new left child if and only if $g_\mathtt{C}(\mathbf{d}) \leq 0$.

*This research was partially supported by NSF grant IIS-9811229. *Contact author*: Daniel Boley, `boley@cs.umn.edu`, tel: +1-612-625-3887, fax: +1-612-625-0572.

Algorithm `PDDP Fullbuild`.
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`.
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** $i \in \mathtt{C}$, if $v_i \leq 0$, assign document i to `L`, else assign it to `R`.
6. **Result:** A binary tree with $c_{\max}$ leaf nodes forming a partitioning of the entire dataset.

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

experi-ment	doc count	word count	description of word selection
F1	98	5623	all non-stop words
F5	98	399	frequent item sets
J1	185	10536	all non-stop words
J5	185	449	frequent item sets
J6	185	5106	all words w/ freq>1
K1	2340	21839	all non-stop words
K2	2340	7358	top 10 words (freq>1)
K3	2340	8104	top 20 words in each doc
K5	2340	1458	frequent item sets

Figure 2: Document datasets used for experiments.

method	*entropy*
Agglomeration	0.6843
PDDP	0.6895
ARHP	0.7874
K-means - LSI	0.8370
PDDP TFIDF	1.0576
AutoClass	2.0497
Agglom. TFIDF	2.3393

Figure 3: Entropies by PDDP (sec. 2) and several other methods (sec. 3) on the J1 dataset (with 16 clusters).

The computation of $\mathbf{u}_{\mathtt{C}}$ is the most expensive step in this whole process. PDDP uses a fast Lanczos-based solver for the singular values of the matrix of documents in the cluster [Golub and Van Loan, 1996]. A simple analysis shows that the cost of the algorithm is dominated by this singular vector computation, and the cost of the singular vector computation is proportional to the matrix fill [Boley *et al.*, 1998].

3 Related Work

Document clustering is an extremely active area of research, which we can describe only briefly. Existing approaches to clustering are generally based on either probabilistic methods, or distance and similarity measures (see [Frakes and Baeza-Yates, 1992]). Distance-based methods such as k-means analysis, hierarchical clustering [Jain and Dubes, 1988] and nearest-neighbor clustering [Lu and Fu, 1978] use a selected set of attributes appearing in different data samples as the dimensions. Each such attribute vector, representing a data sample, can be viewed as a point in this multi-dimensional space.

The methods compared against PDDP in Fig. 3 are some of the more popular existing methods for clustering. Agglomeration [Duda and Hart, 1973] is a classical bottom approach to cluster data embedded as vectors in a high-dimensional space, in contrast to PDDP which is a top-down approach. A "divide-and-conquer" agglomerative approach to efficiently partition news articles retrieved off the newswires was proposed in [Cutting *et al.*, 1992]. However the "divide" aspect of the agglomerative approach was based on a probablistic splitting, in contrast to the deterministic methods proposed in this paper. Efficient hypergraph partitioning algorithms based on association rules (ARHP) [Han *et al.*, 1998] have been used as an unsupervised clustering technique, but these methods require a certain amount of preprocessing to build the necessary graph structures. Latent Semantic Indexing (LSI) [Berry *et al.*, 1995] was proposed as a method to reduce both the dimensionality and the noise in a dataset in preparation for further processing. K-means analysis is a method for computing clusters using a set of sample cluster means as seeds for each cluster, and a distance measure in a high-dimensional space [Gose *et al.*, 1996]. AutoClass is a method based on multi-dimensional Bayesian statistical analysis [Cheeseman and Stutz, 1996]. The TFIDF scaling, as previously mentioned, is an alternative scaling method [Salton and Buckley, 1988], as opposed to simple norm scaling.

There has been some recent work on projections and hierarchical structures for data analysis. Bishop and Tipping [Bishop and Tipping, 1998] assemble a series of projections of the data into a hierarchical tree structure, but for the purpose of visualizing a set of numerical data. Schütze and Silverstein [Schütwe and Silverstein, 1997] explore the effects on the accuracy of clustering algorithms of various strategies for initial term selection, weighting, and projections. Singhal et al [Singhal *et al.*, 1996] explore the effect of different TFIDF scalings and other normalization strategies in a systematic study. Zamir et al [Zamir *et al.*, 1997] propose the method of *word intersection clustering* based on a *global quality function*. This is a measure of cluster cohesion which is a viable alternative to the scatter measures used in this paper. Zamir's work is reported in the context of new fast clustering methods for Web search results.

4 Applications

PDDP can be applied to any domain in which the attribute values can be represented numerically, though the scaling must be adapted to the particular domain at hand. In most cases, satisfactory results have been

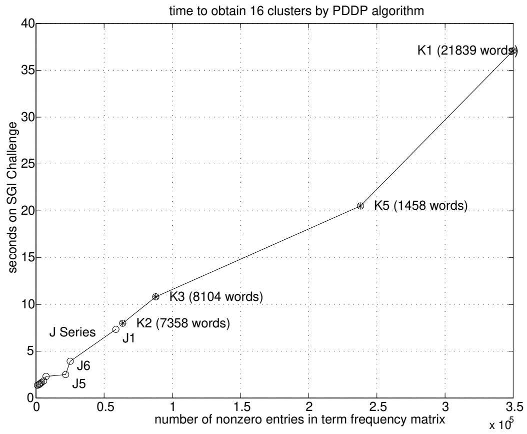

Figure 4: Time in seconds on a 196MHz SGI Challenge to obtain 16 clusters for various datasets using the PDDP Algorithm.

obtained using a simple norm scaling, in which each attribute vector is scaled to unit length. The simplicity of the algorithm allows it to be applied in a straightforward way to many different applications, as illustrated below.

Results on Text Documents. Using several collections of news articles retrieved from the Web, we have demonstrated the performance of the PDDP algorithm both from the point of view of the quality of the clusters generated and the scalability of the algorithm, These can be briefly illustrated by a few figures based on the datasets in Fig. 2. The quality of the clusters were estimated by computing an entropy measure using the same methods and labels as in [Boley, 1998]. For testing purposes only, each document was manually classified into one of ten topic categories. The total entropy is the weighted sum of the individual cluster entropies:

$$e_{\mathtt{C}} = -\sum_i \left(\frac{c(i,\mathtt{C})}{\sum_i c(i,\mathtt{C})}\right) \cdot \log\left(\frac{c(i,\mathtt{C})}{\sum_i c(i,\mathtt{C})}\right),$$
$$e_{\text{total}} = \frac{1}{m}\sum_{\mathtt{C}} e_{\mathtt{C}} \cdot (\text{number of documents in cluster } \mathtt{C}).$$

where $c(i,\mathtt{C})$ is the number of times topic i occurs in cluster $\mathtt{C}$. Lower entropy values correspond to better quality clusters. Fig. 3 shows that entropies obtained by PDDP were at least as good as those obtained by several classical methods.

Fig. 4 illustrates how the cost of PDDP is approximately linear in the volume of the data being clustered as measured by the number of nonzero entries in the term frequency matrix. New documents in the same domain do not add many new words to the dictionary, hence the times taken by PDDP tend to be linear in the number of documents. Even on these larger datasets, the quality of the clusters generated by PDDP remains competitive with ARHP, which is the only other algorithm shown in Fig. 3 which can scale to large datasets. A sample comparison of entropies on the large datasets is shown in Fig. 5.

dataset-method	*entropy*
K1-PDDP	0.4580
K1-ARHP	0.4573
K2-PDDP	0.8117
K2-ARHP	0.9365
K3-PDDP	0.8579
K3-ARHP	0.8672
K5-PDDP	0.4716
K5-ARHP	0.4936

Figure 5: A comparison of entropies from PDDP and ARHP methods when computing 16 clusters on some larger datasets. The ARHP actually computed between 16 and 18 clusters and also threw out up to 5 documents as unclassifiable.

Distinctive Word Selection. The PDDP method identifies automatically the words that are most distinctive about each cluster. The largest entries in the centroid vector are the words appearing the most often in a cluster, whereas the largest entries in the principal direction vector correspond to the words that are the most important in placing a document into a cluster as opposed to its sibling cluster. The difference between using the centroid and principal direction vectors can be illustrated by examining a typical cluster obtained by PDDP from the K1 dataset. The entire dataset consists of a mix of documents on health, entertainment, business, sports, and politics. One particular cluster has 12 documents with the headlines:

- 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

Many documents in this set include a navigation bar, and, as a consequence, the largest entries in the centroid vector are dominated by words appearing in that navigation bar (where the words have been stemmed during the preprocessing):

`help entertai health new polit sport tech bize index.`

In contrast, the largest entries in the principal direction vector are dominated by words that reflect much more closely the specific topics of the documents in the cluster:

`dengue fever dna gene gold probe diseas gubler guttman.`

One can also collect all the "most distinctive" words throughout the entire PDDP tree, and try to use just

dataset	*words selected*	*entropy*
K1 top 30 words	425	0.4071
K1 top 40 words	528	0.4650
K1 top 50 words	631	0.4844
K1 all words	21,839	0.3903

Figure 6: Quality of clusters obtained when the most distinctive words from each cluster are selected for partitioning the entire K1 dataset.

those words to cluster the entire dataset. Fig. 6 shows one sample result using this idea. For example, just using the top 30 words taken from all principal direction vectors to cluster the entire dataset yielded an entropy of .4071, little more than that obtained using all 21,839 words. We also collected the 40 top words obtained by clustering a random subset of the documents (one quarter of the whole dataset). The repeated tests resulted in an entropy average of .4427 with a worst case of .4799, depending on which documents were chosen. This means that one can choose the the words using only a small set of documents, and then cluster all the documents using only a few words. This would be especially relevant if most of the documents are not available at the start, as well as significantly reducing the cost by reducing the dimensionality of the space.

Chemical Toxicology Data. The physical properties of various organic compounds represents a set of numerical attribute values that can easily be clustered by PDDP. The versatility of PDDP can be demonstrated by noting that PDDP can be used as a predictor for some output attributes for the same organic compounds. In an initial experiment using 164 organic compounds and 175 physical characteristics, ranging from Ph under various conditions to various topological properties of the molecule, we attempted to predict the toxicity of the compounds as measured by the quantity (in mmol/l) needed to achieve a given death rate with rainbow trout. In preliminary experiments we applied the PDDP method to cluster the entire set, and then computed the "predicted" toxicity level of each compound as the average toxicity of the *other* compounds in the same cluster. In 91.46% of the cases, the predicted toxicity was close to the true measured value; i.e. within 10% of the maximum toxicity level.

Movie Ratings. Automated recommendation applications, like those used by Amazon.com, constitute large datasets where missing attribute values predominate. PDDP was tested in one such service, a movie recommendation application, Movielens [Riedl and others, 1998]. The basic data structure is a matrix of movie ratings by user and movie, in which most of the entries are missing. 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)
- Back to the Future (1985)
- Mission: Impossible (1996)
- Rock, The (1996)
- Raiders of the Lost Ark (1981)
- True Lies (1994)
- Toy Story (1995)
- Return of the Jedi (1983)
- Terminator 2: Judgment Day (1991)
- Indiana Jones and the Last Crusade (1989)

5 Classification and Updates with New Data

Insertion of New Data. The binary tree constructed by PDDP can be used to incorporate new data samples not available at the time the tree is originally constructed. The simplest approach is simply to incorporate each new data sample **d** into an existing PDDP tree. At each node in the tree, the placement of a document in the left or right child depends on the hyperplane passing through the centroid and normal to the principal direction. Thus any new vector can be placed by using the already computed hyperplane.

Classification Updates. After many data samples have been inserted into the tree using the method just described, the clusters may no longer represent a good partition of the corpus. This is especially true when new data samples are significantly different in content than those used to build the tree. Updating the tree, or adjusting a subtree that no longer has well defined clusters, is one way to avoid misclassifications. The decision when to update the tree is determined by a trigger value based on the scatter value of individual clusters. The tree update is quick and efficient when only the subtree containing the unbalanced leaf cluster is adjusted. When a leaf cluster is found to be unbalanced, its parent cluster is re-partitioned. If the parent cluster also becomes unbalanced, the same decision strategy and re-partition process is applied. Potentially, these adjustments could percolate all the way up to the root of the entire tree, in which case a full partitioning is carried out. However, the hierarchical nature of the tree permits one to easily localize adjustments to as small a part of the corpus as needed. The full builds can be limited to situations when all the interior nodes of the tree become unbalanced, as measured by the individual nodes' scatter values.

Experiments. In order to validate the strategies for inserting new data into an existing PDDP tree that we described earlier in this Section, we applied the methods to four datasets of small to modest size previously used in [Boley, 1998] (see Fig. 2). Thirty separate tests were run using fifty percent of each set to construct a tree and the remaining fifty percent being inserted. The results represent the mean of the thirty tests. For the update tests a trigger value based on the ratio between the maximum and average scatters among the clusters was used. The following four variants of PDDP were compared:

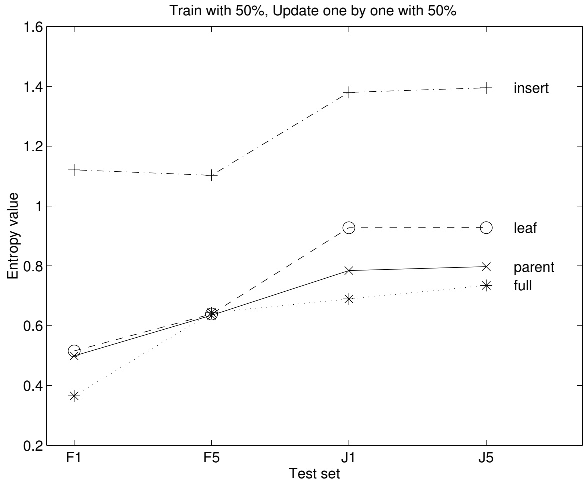

Figure 7: Average entropies obtained from four different PDDP update strategies on the four datasets from [Boley, 1998]. The entropies were obtained by averaging the results of 30 different runs per dataset.

`Fullbuild`: Re-partition the entire dataset from scratch for each new document using the PDDP method (Fig. 1).

`Insert`: Insert new documents with the method of Section 5.

`Update-Leaf`: Same as `Insert`, except an adjustment is potentially applied after each insert, based on the trigger value.

`Update-Parent`: Same as `Update-Leaf`, except that the parent of the leaf node receiving the new document is always re-partitioned after every insert.

From Fig. 7, it is apparent that the two update strategies were able to maintain entropies comparable with `Fullbuild` but distinctly better than the simple insert strategy. This performance was partly because the trigger mechanism would occasionally force a `Fullbuild`. The two update strategies were able to maintain these good entropies using only about one seventh of the cpu time required to repeat the `Fullbuild` for every document.

References

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

[Bishop and Tipping, 1998] C.M. Bishop and M.E. Tipping. A hierarchical latent variable model for data visualization. *IEEE Trans. Patt. Anal. Mach. Intell.*, 20(3):281–293, 1998.

[Boley *et al.*, 1998] D. Boley, M. Gini, K. Hastings, B. Mobasher, and J. Moore. A client-side web agent for document categorization. *J. Internet Research*, 8(5):387–399, 1998.

[Boley, 1998] D. L. Boley. Principal Direction Divisive Partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, Dec. 1998.

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

[Cutting *et al.*, 1992] D.R. Cutting, D.R. Karger, J.O. Pedersen, and J.W. Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In *ACM SIGIR Conf on Research and Development in Information Retrieval*, pages 318–329, 1992.

[Duda and Hart, 1973] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[Frakes and Baeza-Yates, 1992] W. B. Frakes and R. Baeza-Yates. *Information Retrieval Data Structures and Algorithms*. Prentice Hall, Englewood Cliffs, NJ, 1992.

[Golub and Van Loan, 1996] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins Univ. Press, 3rd edition, 1996.

[Gose *et al.*, 1996] Earl Gose, Richard Johnsonbaugh, and Steve Jost. *Pattern Recognition and Image Analysis*. Prentice Hall, 1996.

[Han *et al.*, 1998] E.H. 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.

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

[Lu and Fu, 1978] S.Y. Lu and K.S. Fu. A sentence-to-sentence clustering procedure for pattern analysis. *IEEE Transactions on Systems, Man and Cybernetics*, 8:381–389, 1978.

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

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

[Schütwe and Silverstein, 1997] H. Schütwe and C. Silverstein. Projections for efficient document clustering. In *ACM SIGIR 97*, pages 74–81, 1997.

[Singhal *et al.*, 1996] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *ACM SIGIR 96*, pages 21–29, 1996.

[Zamir *et al.*, 1997] O. Zamir, O. Etzioni, O. Madani, and R.M. Karp. Fast and intuitive clustering of web documents. In *KDD 97*, 1997.