Computational Statistics
DOI 10.1007/s00180-007-0090-8

ORIGINAL PAPER

Iterative Denoising

Kendall E. Giles · Michael W. Trosset · David J. Marchette · Carey E. Priebe

Received: 6 July 2007 / Accepted: 4 September 2007

Abstract One problem in many fields is knowledge discovery in heterogeneous, high-dimensional data. As an example, in text mining an analyst often wishes to identify meaningful, implicit, and previously unknown information in an unstructured corpus. Lack of metadata and the complexities of document space make this task difficult. We describe Iterative Denoising, a methodology for knowledge discovery in large heterogeneous datasets that allows a user to visualize and to discover potentially meaningful relationships and structures. In addition, we demonstrate the features of this methodology in the analysis of a heterogeneous Science News corpus.

Keywords Knowledge discovery · Text mining · Classification · Clustering

1 Introduction

A user who wants to understand a large, heterogeneous, and high-dimensional set of data and find interesting information and relationships in that data needs a sufficiently

K. E. Giles (✉)
Department of Statistical Sciences and Operations Research,
Virginia Commonwealth University, Richmond, VA 23284, USA
e-mail: kendallgiles@gmail.com

M. W. Trosset
Department of Statistics, Indiana University, Bloomington, IN 47405, USA

D. J. Marchette
Dahlgren Division, Naval Surface Warfare Center, Dahlgren, VA 22448, USA

C. E. Priebe
Department of Applied Mathematics and Statistics, Johns Hopkins University,
Baltimore, MD 21218, USA

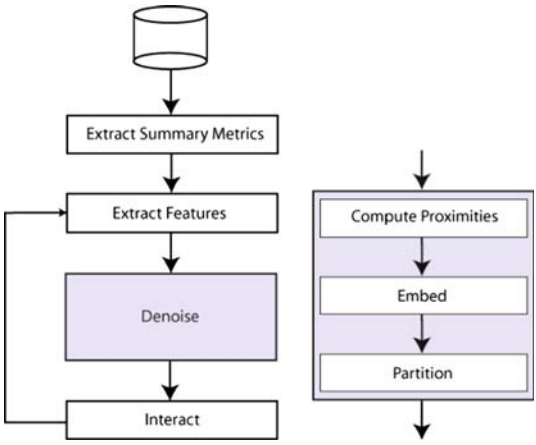
flexible and powerful computational framework in hand to facilitate data processing and knowledge discovery. For example, imagine that a user has been presented a large collection of text documents and wants to examine and understand those documents from an analytical perspective. The user might have an information retrieval task in mind, where it is desired to find a set of documents relevant to a specific query. Or the user might wish to understand relationships between multiple documents. The user might also wish to identify the topic of discussion in a collection of emails, or to cluster them according to relevant criteria. However, increasingly, the user must analyze large, complex, unstructured datasets, meaning that the dataset may not include class labels for the documents, the number of documents to be analyzed is large, and there may be local (as opposed to global) structures that characterize some of the data. So the user's task is to explore the data, extract meaningful, implicit, and previously unknown information from a large unstructured corpus.

From this scenario we can identify several relevant issues and needs. First, if we consider a word or phrase in one document as one dimension, then the dimensionality of the search space, from a performance perspective, would be prohibitively expensive and difficult for operations on a corpus even on the order of tens-of-thousands of documents and tens-of-thousands of words per document. The computational performance of processing such high dimensional data can be limiting. Moreover, visualizing and comprehending high dimensional spaces can be difficult for the user, who typically understands data best in two or three dimensions. Second, in large, complicated datasets, an important finding for the user might be relationships found in local structures, where features of the data may have differing relationships in different parts of the data. Third, the lack of existing class labels limits the ability of a user to analyze the corpus without first applying some structure to the data.

This paper presents a general methodological instantiation of a general machine learning decomposition framework, described in, e.g., Schalkoff (1991). In particular, our methodology, called Iterative Denoising, is designed for knowledge discovery in large heterogeneous datasets to tease out local structures and relationships of possible interest, display useful information to the user, and address scaleability and high-dimensionality concerns, as in Giles (2006).

2 Methodology

Based on the principle of integrated sensing and processing (Priebe et al. 2004a), the basic philosophy of our methodology is that, starting with a heterogeneous dataset $\mathcal{C}$, based on the selection of appropriate features we *denoise* $\mathcal{C}$ into $\{\mathcal{C}_1, \ldots, \mathcal{C}_J\}$, where each $\mathcal{C}_j$ is meant to be more homogeneous than $\mathcal{C}$ (Priebe et al. 2004b). We note that in order to highlight certain types of multivariate structure in the data, the features in $\mathcal{C}$ are transformed and/or represented in a lower-dimensional space before they are partitioned, and so we mean "denoising" to be a bit more than just the "clustering" of typical machine learning approaches. We also note that this lower-dimensional representation is important for user visualization and interaction. The denoising of each $\mathcal{C}_j$ continues recursively until the collection is homogeneous enough for inference to proceed. The resulting clusters are organized into a hierarchical, divisive tree.



(a) Complete Flowchart **(b)** Denoising Detail

Fig. 1 Iterative Denoising flowchart

We present a high-level version of our methodology in Fig. 1a. The first component in the figure is `Extract Summary Metrics`. We are initially given a dataset $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ consisting of n elements, where n is large. These elements, which may come from a database or stream, are the observations of interest—they can be text documents, images, web pages, computer network flows, etc. In this step we extract out useful (essential) summary metrics Δ for each document that we can refer to throughout future processing:

$$\Delta = \{\Delta_1, \ldots, \Delta_n\} = \texttt{essentials}(\mathcal{C}).$$

The exact form of the metrics is data-dependent, but one example of summary metrics is word counts for words in a document.

Next, once we have summarized the node's data, we want to `Extract Features` from the data—we use these features to measure the similarity or dissimilarity of objects in the node. Note that the features are dependent on the data in a particular node. For this reason, we call this function `cdfe`, for *corpus-dependent feature extraction*.

Let f_Δ be the set of features in the current collection of metrics Δ, then compute:

$$X_\Delta = \texttt{cdfe}(\Delta),$$

where X_Δ is a $|\Delta| \times |f_\Delta|$ matrix. Note that both the features and the number of features depend on the collection of documents in the current node represented by Δ. For example, for text documents the features might be a *mutual information* measure based on associations of occurrences of Ngrams among documents in the corpus, as in Lin and Pantel (2002), or they may be simple word-frequency counts.

Next, we want to `Denoise` the current node. Using the desired features we partition the X_Δ after it has been represented in a lower-dimensional space, to highlight certain types of multivariate structure. As mentioned above, we end up with a number of child

partitions or cells (γ of them) that are more homogeneous than the parent node. This denoising process is expanded in detail in the next section.

An important attribute of our methodology is that users may want to `Interact` with the resulting visualized representations. We are not only providing to the users lower-dimensional-space representations to highlight (possibly) desired structures in the data, but we are also allowing the user to interact with the data. For example, the user may wish to change the displayed geometry relationships between objects, say to reflect some metadata intelligence the user has received that is not reflected in the original data (see Priebe et al. 2004b). But through interaction, the user dynamically affects the growth of the tree.

For each resulting child node, which represents data from the previously discussed denoising and user interaction steps, we recursively start over from the `Extract Features` component. It is necessary to extract new features for each node because the objects in the child node are a subset of the objects that were in the parent node, and thus the feature metrics need to be recomputed to reflect the changed node membership. The flow continues, recursively denoising and growing the tree, until some stopping criteria has been met for a particular node (such as reaching the node's minimum number of objects) or for the tree itself (such as reaching the maximum desired level of the denoising tree). Because our methods are intended for interactive use with large data sets, and because our methods proceed by recursively denoising previously identified subsets of data, we call our methodology *Iterative Denoising*.

2.1 Denoising

Given a node and its corresponding feature matrix, X_Δ, we seek to partition the objects in the node into two or more subsets of objects of greater homogeneity than the entire node. This is essentially the problem of clustering, i.e., of identifying subsets of objects that exhibit "internal cohesion" and "external isolation" (Cormack 1971). The fundamental challenge of `Denoising` is to identify and/or develop clustering methodologies that scale well to large data sets and that facilitate user interaction, a limitation of most clustering algorithms.

Note that one cannot claim that subsets of a node are more homogeneous than the node itself unless one has some way of measuring which pairs of objects are nearby and which pairs of objects are far apart. Every clustering methodology necessarily relies on some measure of pairwise proximity; hence, the first step of `Denoising` is to `Compute Proximities`.

Assuming that our proximities are symmetric, then the introduction of proximities allows us to represent the objects in the current node as an edge-weighted undirected graph G. Each vertex in the graph corresponds to an object; the edge weights are the pairwise proximities. This representation of the data transforms the clustering problem into a graph partitioning problem.

Given a threshold on the proximities we construct G' to be the unweighted graph with edges corresponding to those of G with weight less than the threshold. Then, a natural way to partition G' into a specified number of subgraphs is to choose the partition so as to balance the number of vertices in each subgraph and to minimize

the number of edges between subgraphs. Unfortunately, this problem is NP-complete (Garey et al. 1974).

A number of heuristic and approximate approaches to graph partitioning have been suggested. For example, Kernighan and Lin (1970) proposed an exchange algorithm that swaps pairs of nodes between clusters. Similar approaches are summarized in Alpert and Kahng (1995), but none of these algorithms scale well to large datasets. One way of addressing scalability is through recursive partitioning, in precisely the same spirit as Iterative Denoising. In multilevel approaches to graph partitioning, the original graph is approximated by a sequence of increasingly smaller graphs. The smallest graph is then partitioned and that partition is propagated back to the original graph (Hendrickson and Leland 1995), as implemented by, for example, METIS (Karypis and Kumar 1998).

A fundamental difficulty with traditional graph partitioning methods, however, is that they do not represent the data in ways that facilitate visualization and user interaction. To accommodate the abilities of most users, we attempt to represent the objects as points in a low-dimensional Euclidean space, in such a way that proximate pairs have small Euclidean distances. The construction of such representations is called *embedding*, or (in psychometrics and statistics) *multidimensional scaling*. Thus, after we `Compute Proximities`, we `Embed`.

Our desire to embed the data in a *low-dimensional* space means that we are not simply embedding the data, but also reducing the dimensionality of the data. In fact, the conceptually distinct steps of `Compute Proximities` and `Embed` can be discerned in various methods for nonlinear dimension reduction, or *manifold learning*, such as in the popular Isomap (Tenenbaum et al. 2000) approach to manifold learning. Though we formalize and generalize these common approaches by noting distinct steps in the process, as discussed in further detail in the next section, the steps of `Denoising` are summarized in Fig. 1b.

2.2 Compute proximities

The proximity of two objects is generally measured by computing *similarities* or *dissimilarities*. Because most embedding algorithms approximate dissimilarities with Euclidean distances, similarities are often transformed to dissimilarities prior to embedding. The conventional transformation exploits a well-known connection between squared Euclidean distances and Euclidean inner products (see, e.g., Critchley 1988).

Note that an appropriate measure of (dis)similarity is application-specific. Recall that summary features, e.g., mutual information measures of association between documents for specific Ngrams, have already been extracted. We compute proximities as follows:

1. We conceive the $|f_\Delta|$ features of object i as a vector, $y_i \in \mathbb{R}^{|f_\Delta|}$.
2. For each pair of objects i and j, we compute

$$r_{ij} = \frac{\langle y_i, y_j \rangle}{\|y_i\| \, \|y_j\|}.$$

Notice that, were we to center the feature vectors before performing this operation, r_{ij} would be Pearson's product-moment correlation coefficient.

3. We construct a weighted undirected graph G with edge weights r_{ij}. For each vertex, we desire the k nearest vertices, as measured by shortest path length, where k is specified by the user. However, to reduce the computational complexity of finding nearest vertices, we do not insist on finding the exact set of k nearest vertices; instead, we settle for an approximation thereof.
4. We construct an unweighted undirected graph G' in which vertices i and j are connected by an edge if either vertex i belongs to the set of (approximate) k nearest neighbors of vertex j, or vertex j belongs to the set of (approximate) k nearest neighbors of vertex i.
5. We construct the adjacency matrix, $A = [a_{ij}]$, of the unweighted graph G', i.e., $a_{ij} = 1$ if an edge connects vertices i and j, otherwise $a_{ij} = 0$. The adjacencies are crude—but efficiently computed—pairwise similarities.

2.3 Embed

Roughly speaking, there are two general approaches to embedding: approaches that fit distances and approaches that fit inner products. The distance approach encompasses both extremely fast heuristic methods like FastMap (Faloutsos and Lin 1995) and more principled methods that require numerical optimization of an error criterion, e.g., the majorization algorithm of de Leeuw (1988) for minimizing the raw stress criterion. The latter tend to be prohibitively expensive for large data sets, but see Trosset and Groenen (2005) for an algorithm that decreases the raw stress criterion with a computational complexity of $O(n)$.

The inner product approach encompasses classical multidimensional scaling (CMDS) (Torgerson 1952; Gower 1966), as used in Isomap, as well as various other techniques for constructing "eigenmaps," e.g., Belkin and Niyogi (2003), Roweis and Saul (2000), and Donoho and Grimes (2003). What these methods have in common is the extraction of d eigenvectors from a symmetric, centered, matrix B of Euclidean inner products. The eigenvalues and eigenvectors of this matrix are then used to construct a configuration of points in $\mathbb{R}^d$.

To apply CMDS to our adjacency matrix, A, we must first convert similarities (adjacencies) to dissimilarities. As explained in Saerens et al. (2004), this is accomplished implicitly by constructing a *Laplacian eigenmap*. The (implicit) dissimilarities are average commute times between pairs of vertices, based on a Markov-chain model of a random walk through the graph.

Define the diagonal matrix $D = [d_{ij}]$, $d_{ij} = 0$ for $i \neq j$, by

$$d_{ii} = \sum_k a_{ik},$$

the number of vertices to which vertex i is adjacent. The symmetric, positive semidefinite matrix $L = D - A$ is the Laplacian matrix of the unweighted graph G'. Some years ago, Fiedler (1973) argued that the eigenvector corresponding to the smallest

positive eigenvalue of L facilitates clustering the vertices of G'. This insight is the inspiration for *spectral clustering*.

To construct the d-dimensional representation that we will call *Fiedler space*, let $0 < \lambda_1 \leq \cdots \leq \lambda_d$ denote the smallest nonzero eigenvalues and let $v_1, \ldots, v_d$ denote the corresponding eigenvectors. The Cartesian coordinates of the points in Fiedler space are then obtained as rows of the matrix

$$F = \left[\frac{v_1}{\sqrt{\lambda_1}} \middle| \cdots \middle| \frac{v_d}{\sqrt{\lambda_d}} \right].$$

Other scalings of the eigenvectors are also used, but dividing each eigenvector by the square root of its eigenvalue corresponds to embedding by CMDS after the transformation to dissimilarity implicit in Saerens et al. (2004). The feature matrix, F, provides Cartesian coordinates used to visualize and partition the objects in the current node of the Iterative Denoising tree.

2.4 Partition

Finally, we measure dissimilarity in Fiedler space by Euclidean distance, then use standard clustering methods to further partition the current node of the Iterative Denoising tree. Our approach allows us to choose any of the myriad algorithms available for clustering points in Euclidean space; see, for example, Everitt (1993), Gordon (1999), and Mirkin (2005) for surveys of various approaches.

To date, our development of Iterative Denoising has relied on k-means clustering. In this approach to clustering, the user specifies $\gamma = k$, the number of subsets in a partition of $x_1, \ldots, x_N \in \mathbb{R}^d$. Algorithms for k-means clustering then attempt to find a partition, $\{C_1, \ldots, C_k\}$, that minimizes the squared error criterion

$$W(C_1, \ldots, C_k) = \sum_{i=1}^{k} \sum_{x_j \in C_i} \left\| x_j - \bar{x}_i \right\|^2,$$

where $\bar{x}_i = \bar{x}(C_i)$ is the mean of the $x_j \in C_i$. There exist a number of algorithms that monotonically decrease W and converge to a locally optimal partition, but algorithms that guarantee global solutions are usually overwhelmed by several hundred x_i. For this reason, we are content to find good (not necessarily optimal) partitions.

3 Implementation description

While the previous sections describe the Iterative Denoising methodology, we note that there are numerous possible algorithmic implementations. For example, for the `Compute Proximities` step, we could choose to implement dissimilarities or approximate nearest neighbor adjacencies; for the `Embed` step we could perform multidimensional scaling or Laplacian eigenmapping—all such implementations would be within our methodology, and analysis of these implementation differences will fuel

Table 1 Text corpus metrics

Symbol	Description
$m_{..}$	The total number of words in $\mathcal{C}$
$m_{o\cdot}$	The total number of all words in that document
m_{ow}	The number of times word w appears in o
$m_{\cdot w}$	The total number of times the word appears in $\mathcal{C}$

future research. We describe our initial implementation of Iterative Denoising in this section; the following section describes our use of this implementation to analyze a text corpus.

3.1 Extract summary metrics

Because the first component, `Extract Summary Metrics`, is only performed once per dataset and is not part of the main Iterative Denoising recursive structure, we designed the `Extract Summary Metrics` component as a stand-alone function that can be run as a separate program or called from the main program. We also made this a stand-alone function because this is the only step that is dataset-dependent—it is here that the raw data is abstracted into a collection of summary metrics. As mentioned previously, in our initial implementation we focused on text documents, and so this step's description of the implementation is specific to abstracting text.

In order to extract summary metrics from the raw dataset, we decomposed the problem into three substeps. First, we clean each document using an implementation of the Porter stemming algorithm (Porter 1980). This algorithm removes common suffixes from English words and non-word text, and returns a set of word stems or tokens. For example, the stem for the words *connect*, *connected*, *connecting*, *connection*, and *connections* is *connect*. Second, we convert the word stems into Ngrams (where Ngrams can be defined as sequences of word stems) using the `count` script of the Ngram Statistics Package (Banerjee and Pedersen 2003). This script inputs raw text files, creates a list of the Ngrams in those files, and outputs the Ngrams with their frequencies, in descending order by frequency.

Next, using a large hash structure, documents $\times$ unique Ngrams (words), we output for each document o: the word w, the number of times word w appears in o (m_{ow}), the total number of times the word appears in $\mathcal{C}$ ($m_{\cdot w}$), and the Ngram number. This gives us a set of metrics Δ. From these summary metrics we can also determine $m_{..}$, the total number of words in corpus $\mathcal{C}$, and for each document $o \in \mathcal{C}$ the total number of all words in that document $m_{o\cdot}$. The extracted metrics for processing text documents are summarized in Table 1.

3.2 Extract features

In this step we use the summary metrics Δ computed in `Extract Summary Metrics` to extract the features of the documents. For our implementation, we extracted mutual information values for each document and unique word in the corpus.

Using our previously noted essentials, this mutual information value is computed as:

$$\texttt{MI}_{ow} = \texttt{log}\frac{m_{ow}}{m_{o\cdot}} \Big/ \frac{m_{\cdot w}}{m_{\cdot\cdot}}.$$

The number of features is the number of distinct words in the corpus; for each document we compute that number of mutual information values.

3.3 Compute proximities

From our summary features we wish to create a structure that reflects object similarities. As described in Sect. 2.2, we do so by constructing an unweighted undirected graph for which vertices correspond to objects and edges connect pairs of objects whose proximity attains a specified threshold. One natural way to implement proximity-thresholding is to use a nearest-neighbor search algorithm. In $\mathbb{R}^d$, traditional (exact) nearest neighbor algorithms use either $n^{O(d)}$ space or $O(dn)$ time. However, due to the curse of dimensionality, exact searches perform little better than sequential searches as n gets large. Recent work on approximate nearest neighbor algorithms (Houle 2003, Houle and Sakuma 2005, Arya et al. 1998, Kushilevitz et al. 1998, Indyk and Motwani 1998, Gionis et al. 1999, Clarkson 1999) has attempted to circumvent the curse of dimensionality by essentially relaxing the exact nearest neighbor search restriction in return for faster search performance. It is precisely this relaxation that we exploit to realize our proximity-thresholded graph G'.

In our implementation we use the SASH data structure (Houle 2003; Houle and Sakuma 2005) to perform nearest neighbor searches to find the K nearest neighbors to a particular object. Using SASH, we realize $G = \texttt{sash}(X_\Delta)$. However, G may not be symmetric due to the geometry of the object similarities, and so we add missing edges to G as $G' = \texttt{symmetrize}(G)$. From G' we create the adjacency matrix A as noted in Sect. 2.2. Since A is large and sparse, we actually do not store A but use a sparse representation both for storage and matrix computations, which we will further detail in the next section.

For n objects, the discussion in Sect. 2.2 conveys the impression that it is necessary to pre-compute all $n(n-1)/2$ proximities. In fact, it is not necessary to compute all $O(n^2)$ proximities in order to find approximate nearest neighbors. Because the SASH data structure is organized as a multi-level hierarchy of random samples, where objects in a given level are connected only to approximate nearest neighbors drawn from the level immediately above, not all pairs of proximities are necessarily computed.

3.4 Embed

In order to embed our high-dimensional objects in a low-dimensional space, as discussed in Sect. 2.3, we want to convert our computed similarities to dissimilarities in part by computing the d smallest positive eigenvalues of our Laplacian eigenmap L. One difficulty in computing these eigenvalues is that L may be quite large. But L may also be sparse, and so we utilize the ARPACK library (Lehoucq and Yang 1988) to

exploit this sparsity in order to address scalability issues. ARPACK is a collection of Fortran routines that solve large eigenvalue problems by implementing a variant of the Arnoldi process (Arnoldi 1951) [which in the symmetric case reduces to a variant of the Lanczos process (Lanczos 1950)]. Because ARPACK does not have a mode in which it calculates the d smallest positive eigenvalues, we ask for the d' smallest eigenvalues, then use the $d < d'$ smallest positive eigenvalues and corresponding eigenvectors to construct an embedding. The discrepancy, $d' - d$, equals the number of connected components of the graph G'. ARPACK is an iterative method that successively computes vectors and asks for matrix–vector products. ARPACK does not actually store or factor the matrix, L, but rather queries a user-provided function that computes the product of L with an ARPACK-provided vector.

3.5 Partition

As noted in Sect. 2.4, to measure dissimilarity in Fiedler space by Euclidean distance, we use k-means clustering. In our implementation we use the k-means implementation of Lloyd's algorithm in Kanungo et al. (2004). k-means clustering supposes that, given a set of $n \in \mathbb{R}^d$ data points and a number of desired centers γ, minimize the mean-squared distance from each data point to its nearest center. Lloyd's algorithm, as implemented in Kanungo et al. (2004), observes that the optimal placement of a center is at the centroid of its associated cluster. For a set of k centers $z \in Z$, Lloyd's algorithm iteratively moves every center z to the centroid of the corresponding neighborhood of data points $V(z)$ until convergence.

3.6 Interact

For visualization and user interaction, we utilize SpaceTree (Grosjean et al. 2002). Among other changes, we modified the original code to allow for images to be displayed when a user clicks on a particular denoising tree code. Our `Interact` component outputs an XML file containing information on node contents and tree hierarchy information in the format desired by SpaceTree.

4 Application: Science News corpus

As an example, we used a heterogeneous corpus $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ of n text documents from the Science News (SN) website. Table 2 shows the number of documents in $\mathcal{D}$ per class, where $|\mathcal{D}| = 1{,}047$. In addition, we note the symbol scheme used to identify class membership in the following figures. We again stress that our framework is an unsupervised approach, and so the class labels are just for validation; we applied the previously described implementation of Iterative Denoising to this database to examine the structures and relationships contained therein.

Each document in the corpus is represented in its own text file. The result of our `Extract Summary Metrics` step was a collection of 1,047 files, each containing the Ngrams for that document and appropriate summary counts. For this example,

Table 2 Science news corpus

Class	Number of documents	Symbol
Anthropology	54	○ Open circle
Astronomy	121	♦ Closed diamond
Behavioral Sciences	72	□ Open square
Earth Sciences	137	△ Open triangle
Life Sciences	205	◇ Open diamond
Math & CS	60	■ Closed square
Medicine	280	• Closed circle
Physics	118	▼ Closed triangle

we only used monograms. Once the summary metrics were extracted, we began the Iterative Denoising recursion by choosing some initial parameter values and invoking our implementation on the dataset. We used initial parameters of $K = 20$ nearest neighbors, $\gamma = 3$ partition cells, and $d = 4$ dimensions for our Fiedler space embedding. Depending on the size of the nodes deeper into the Iterative Denoising tree, we adjusted K to keep the K/n ratio small.

Figure 2 shows the resulting Iterative Denoising tree. Each node in the figure shows the node index, the K used for that node, counts for each class (in the order given in Table 2), and the size of the node. Below each node label is a view of that node's Fiedler Space embedding of the documents in that node. There are a total of four levels to the tree, though for space limitations the entire tree is not shown. In general, Iterative Denoising trees are not symmetric—iteration occurs if the node is sufficiently large and non-homogeneous. For example, Node 12 is small and relatively pure—this node contains a large collection of Behavioral Sciences documents that have been extracted from among the Medicine documents, so it may be sufficient for iteration to stop there. However, Node 13 is large and mixed, so iteration can proceed to another level. Where appropriate, some of the Fiedler Space embeddings will be shown below in a magnified form for exposition. However, this tree view shows an example of the overall Iterative Denoising framework, where a large collection of non-homogeneous documents, in Node 1, is iteratively denoised to produce relatively homogeneous collections of documents in the leaves of the tree, as seen in Node 12. Finally, it should be remembered that iteration proceeds as a function of corpus-dependent feature extraction—new features and dimension reduction processing are recomputed for subsets of a node rather than relying on simple hierarchical clustering, as described in Sect. 2 and as detailed in the following.

In Fig. 2, and throughout, each document is shaded according to its partition, each document's location is noted by a symbol appropriate to the class of the document, and each location is plotted according to the smallest two Fiedler vectors. Note that partition boundaries are not linear—partitioning occurs in the entirety of the embedding space (in this example, $d = 4$, though only the first two dimensions are plotted). The resulting geometric relationships immediately reveal a large cluster of Medicine documents (closed circles) in the southeast portion of the figure, and a cluster of Astronomy documents (closed diamonds) in the southwest. It is not unreasonable to suggest

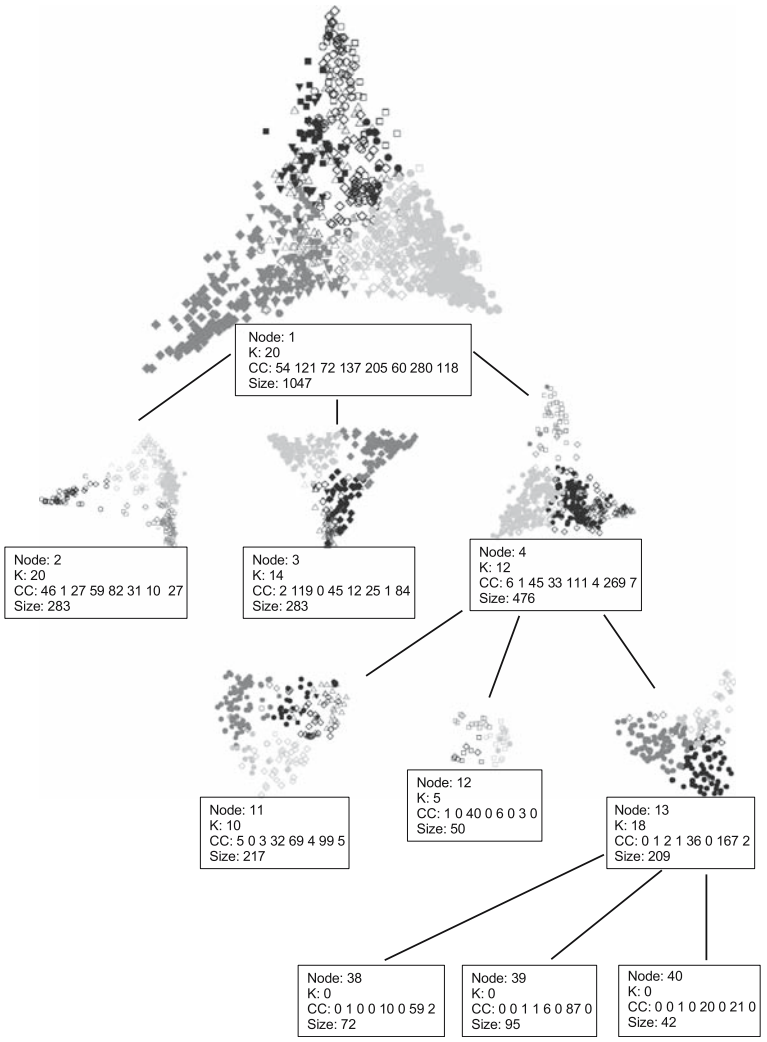
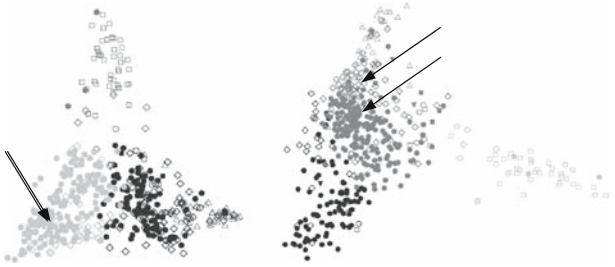


Fig. 2 An iterative denoising tree on science news corpus

that Astronomy and Medicine are two very distinct fields, and so it is interesting, and a partial validation of our approach, that Astronomy and Medicine documents are recognized and realized as separate clusters. Similarly, a collection of Physics documents (closed triangles) is next to Astronomy, and a collection of Behavioral Sciences (open squares) documents is next to Medicine. Though the cluster is less well defined, on the left after Physics is a collection of Earth Sciences (open triangles), along with a collection of Math/CS (closed squares). Similarly, on the right after Medicine and Behavioral Sciences is a collection of Life Sciences (open diamonds) and Anthropology (open circles). So, though some clusters overlap, from the root node it can be seen that Iterative Denoising has reasonably clustered the documents by type, and the clusters have been arranged according to an intuitive affinity of document contents—the



(a) **With** CDFE **(b)** **Without** CDFE

Fig. 3 Node 4

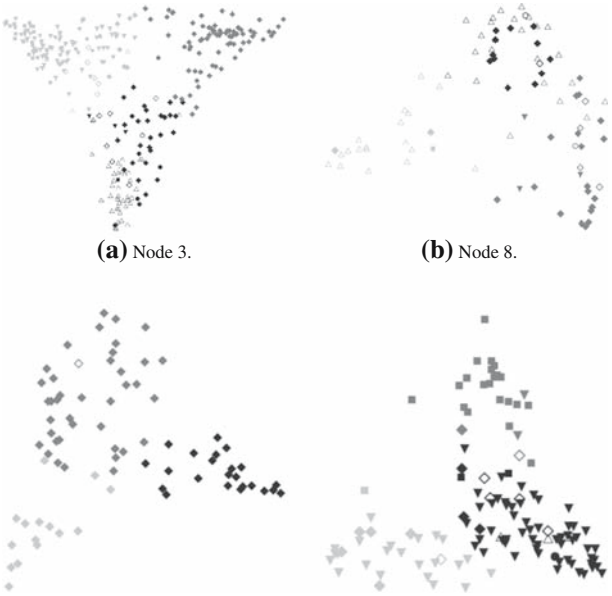
physical sciences on the left (e.g., physics, astronomy, earth sciences, math/CS) and the human/life sciences on the right (e.g., medicine, life science, behavioral science, anthropology).

The structural relationships in the documents of Node 1 are reflected in the resulting partitionings. Partition 1 (darkest shade) contains documents at the apex of Node 1, and so contains a large mixture of all document types, whereas Partition 2 (medium shade) is weighted more with physical sciences documents and Partition 3 (lightest shade) is weighted with more life sciences documents. Though some clustering by document type was evident in Node 1, as a result of corpus-dependent feature extraction from another iteration of Iterative Denoising on each partition, Nodes 2, 3, and 4 exhibit clearer cluster boundaries. Node 4, for example, contains in particular evident clusters of Behavioral Sciences, Earth Sciences, and Medicine documents, as shown in Fig. 3. This node also contains a large collection of Life Sciences documents, though this collection is difficult to see because they are largely intermixed with the Medicine documents.

As a specific example, note that there are two arrows drawn very close together in Fig. 3a (note: Fig. 3b will be discussed in Sect. 4.2). These arrows point to two specific documents that are in very close proximity. Iterative Denoising has placed one Life Sciences document, "Skin cells reveal they have hairy origins" by J. Travis, close to a Medicine document titled "New inner ear hair cells grow in rat tissue" by Nathan Seppa. The first document details the work of a research group that believes that hair follicles are the origins of growing skin cells, while the second document details another research group trying to grow new inner ear hair cells. These two documents, by two different authors in two different fields, both discuss a common theme of research on cells and hair.

Through another iteration of Iterative Denoising, Node 4 is split into Nodes 11, 12, and 13, as shown in Fig. 2. Node 12, as mentioned previously, contains most of the Behavioral Sciences documents, Node 11 contains relatively distinct clusters of Medicine, Earth Sciences, and Life Sciences, and Node 13 contains largely Medicine and Earth Sciences. Though not shown, the tree-view class labels show two relatively pure leaves of Medicine documents in Nodes 38 and 39, and a two-class node of Medicine and Earth Sciences in Node 40.

Node 3, shown in Fig. 4a, shows distinct clusters of Astronomy, Earth Sciences, Physics, and Math/CS documents, though in this node the Physics and Math/CS



(a) Node 3. **(b)** Node 8. **(c)** Node 9. **(d)** Node 10.

Fig. 4 Nodes 3, 8–10 of Fiedler space embedding

clusters are not that distinct. Similar to the recursion for Node 4, Fig. 4b, c, d show the Level 3 embeddings of Node 3. Node 8 shows mainly the two classes of Earth Sciences and Astronomy, Node 9 shows a homogeneous class of Astronomy documents, except for one Life Sciences document, and Node 10 shows mainly two distinct clusters of Physics and Math/CS. Again, whereas Physics and Math/CS clusters were overlapped in Fig. 4, their clusters are largely distinct in Node 10.

4.1 A detailed analysis of clustered documents

While the above illustrates how, on the whole, Iterative Denoising denoises and clusters documents according to rough document types, it can be seen, however, that this clustering is not perfect, as shown in Fig. 4b. The two dominant classes, Astronomy and Earth Sciences, contain 33 and 42 documents, respectively. In addition, there are also two Anthropology, three Physics, one Math/CS, and five Life Sciences documents in this cluster. If the goal of Iterative Denoising was only to create homogeneous clusters of documents in an unsupervised fashion based on assigned document types, then the addition in particular of at least the life sciences documents would indicate a possible shortcoming of the approach, considering that this node is dominated by the physical sciences. However, consideration of the placement of these life sciences documents in this physical sciences node suggests that Iterative Denoising can cluster by document subtype in addition to document type. Table 3 shows a summary of some of the Life Sciences and Anthropology documents placed in this node. Each document is followed by its nearest physical sciences neighbor, for comparison. Along with each

Table 3 Similarities in Node 8 document neighbors

Class	Title	Representative sentence
Anthropology	Primordial Water A meteorite's salty tale	A water-rich, icy projectile, such as a comet, could have plowed into the newborn asteroid and spilled some of its water.
Astronomy	Searching for Life in a Martian Meteorite A seesaw of results	Jeffrey L. Bada of the Scripps Institution of Oceanography in La Jolla, Calif., says he's all but convinced that cell walls and other biological artifacts, if found, come from meltwater that passed through the meteorite during its 13,000-year sojourn in the Antarctic.
Life Sciences	Myriad Monsters Confirmed in Water Droplets	Within these droplets danced a variety of little animals, some "so exceedingly small that millions of millions might be contained in one drop of water," he reports.
Physics	Big guns, bench work: How life could've come from above	Could life's building blocks have stowed away on such space debris and then survived an impact with Earth?
Life Sciences	Bacteria under ice: Some don't like it hot	Bacteria with odd lifestyles have come under increasing scrutiny of late, with most research focused on the so-called thermophilic species, which prefer scalding homes.
Earth Sciences	Core Concerns The hidden reaches of Earth are starting to reveal some of their secrets	Peering deep into the bowels of the planet, he saw vast currents of molten iron alloy swirling at temperatures above 5,000 kelvins, nearly as hot as the surface of the sun.

document class label, we show the title of the document and a representative sentence from that document. The first document, "Primordial Water A meteorite's salty tale", details the analysis of water-containing meteorites found on Earth that might explain how water originated on this planet. If these meteorites contained water that originated from somewhere off Earth, then this would give weight to the theory that Earth got its water from bombardment by water-containing meteors. This document's neighboring Astronomy document, "Searching for Life in a Martian Meteorite A seesaw of results", also is concerned about the contents of meteorites—especially whether or not a particular meteor contains fossils of bacteria originating from Mars. The debate is whether the structures found are bacteria structures, and whether or not the bacteria could have seeped into the meteor through meltwater once the meteor landed on Earth. So, both documents deal with the analysis of meteors and the tales they might tell about life on Earth. Certainly it seems plausible that these two documents be placed together due to the similarity of their contents. In fact, this Anthropology document could be considered well-placed here, since this document could be a welcome find for a researcher or analyst searching documents relating to the study of meteorites on Earth, who might

not have otherwise discovered the questionably labeled Anthropology document had he been looking exclusively in Astronomy or physical sciences documents.

One pair where the relationship is not as evident is the Life Sciences "Bacteria under ice: Some don't like it hot", and the Earth Sciences "Core Concerns The hidden reaches of Earth are starting to reveal some of their secrets". The former article describes how a large portion of bacteriological research focuses on bacteria that thrive in scalding environments, though research focusing on bacteria that lives in extreme cold environments is of great interest. The latter article details researchers studying computer models of the Earth's (hot) inner core. The relationship seems to be that both articles detail the study of extreme environments—one of ice and one of molten iron. A common author may also explain the geometric affinity of these two documents. We detail one additional document pair. The Life Sciences document "Myriad Monsters Confirmed in Water Droplets", written as if it were a Science News article from the year 1677, describes experiments with viewing tiny creatures inside drops of water using a new scientific apparatus called a microscope. Its closest physical sciences neighbor is a Physics document, "Big guns, bench work: How life could've come from above", that describes experiments with testing whether meteors could have brought the building-blocks for life to Earth. So, both documents evoke the study of life on the small-scale in small containers—in drops of water and chunks of rock. Each of the remaining life sciences documents has some similarly reasonable connection to its closest physical sciences neighbor. From this detailed analysis, Iterative Denoising can emphasize subtype relationships, such as documents relating to the study of meteorites, over global document labels, such as "Astronomy" and "Anthropology". It is also interesting that a common theme of all the documents in this table seems to reflect a common theme of the study of and the search for life in extreme environments. This feature of Iterative Denoising suggests that this document classification approach may have practical benefits for analysts and data miners.

4.2 Illustration by comparison of two key features

Finally, we illustrate two features of the methodology that distinguish it from conventional machine learning approaches. First, we present a specific example of denoising (where the projection at the branch node is superior for some purpose to that at the root) compared to hierarchical clustering. Second, we demonstrate the utility of corpus-dependent feature extraction specific to text mining. In Iterative Denoising, the features (e.g., word-weights) are recomputed on the subset, as opposed to repartitioning the subset based on features computed at the root, and we give an example where this processing choice affects discovered relationships between documents.

We illustrate the first point by considering, for simplicity, a four-class subset of the original document corpus. Here, the corpus is composed of Astronomy, Physics, Medicine, and Math/CS documents, for $n = 579$, with class counts and symbols as in Table 2. Using initial parameters of $K = 10$ nearest neighbors, $\gamma = 3$ partition cells, and $d = 3$ dimensions for our Fiedler space embedding, Fig. 5 shows the Fiedler embedding for the root node. Partition membership is again noted by shade. Note that the four classes cluster cleanly, with one partition relatively homogeneous for

Fig. 5 Four-class science news, root node

Astronomy, one partition relatively homogeneous for Medicine, and the final partition containing a mix of Physics and Math/CS.

With corpus-dependent feature extraction, Iterative Denoising embeds the mixed partition as in Fig. 6a after one iteration. Whereas, in the root node, one partition was mixed Physics and Math/CS, after one iteration of Iterative Denoising that partition has been denoised into one relatively homogeneous partition containing most of the Physics documents, one relatively homogeneous partition containing a large portion of the Math/CS documents, and one mixed partition. Table 4a shows the resulting confusion matrix.

Contrast this performance with that of hierarchical clustering. Figure 6b shows the results of taking the mixed partition in the root node and then performing k-means clustering. Here the quality of the clusters is not as good compared to the quality of the leaves using corpus-dependent feature extraction. As can be seen from the resulting confusion matrix in Table 4b, a comparatively large number of physics documents are in all three partitions, and the partition containing the most Physics documents also contains a large number of Math/CS documents. In addition, the partition containing the largest number of Math/CS documents also contains a large number of Physics documents. Here, Partition 1 is estimated to be Math and CS, Partition 2 is estimated to be Physics, and Partition 3 is estimated to be Physics. So, with corpus-dependent feature extraction, the clusters *appear* to be more homogeneous by class than without corpus-dependent feature extraction.

(a) With CDFE. **(b) Without** CDFE.

Fig. 6 Iterative Denoising versus hierarchical clustering

Table 4 Physics and Math/CS confusion matrices

Class	Partition		
	1	2	3
(a) **With** corpus-dependent feature extraction			
Astronomy	6	2	0
Physics	81	28	1
Medicine	1	2	0
Math and CS	4	17	34
(b) **Without** corpus-dependent feature extraction			
Astronomy	0	5	3
Physics	11	36	63
Medicine	0	1	2
Math and CS	37	0	18

There are a number of ways that we can quantify some assessment of node quality, but we choose an intuitively-appealing and commonly used entropy function to assess node quality. Essentially, the impurity of a node τ is the probability $p(y = 1|\tau)$ for some binary response variable. Here, $y = 1$ if the predicted class is the true class, 0 otherwise. The impurity of the node is given by (see Berk 2006):

$$i(\tau) = [-p\texttt{log}(p)] - [(1-p)\texttt{log}(1-p)].$$

Here, $i(\tau) = 0$ when all the node observations are all the correct class or none are of the correct class, and $i(\tau) = 0.6931472$ when half the observations are labeled correctly and half incorrectly (i.e., the worst case). The "goodness" of a partitioning by a particular classifier g is then given by the difference between the impurity of the parent node and the probability-weighted impurity scores of the m partitions:

$$\Delta I(g, \tau) = i(\tau) - \sum_{j=1}^{m} p(\tau_j) i(\tau_j).$$

With $\Delta I(g, \tau)$, larger values mean better (more homogeneous) partitioning.

If n_{j1} are the number of observations in partition j that have been classified incorrectly, n_{j2} are the number of observations in partition j that have been classified correctly, $n_{j.}$ are the number of observations in partition j, and $n_{..}$ are the number of observations in the parent node, then we can estimate $\Delta I(g, \tau)$ as:

$$\hat{i}(j) = -(n_{j1}/n_{j.})\texttt{log}(n_{j1}/n_{j.}) - (n_{j2}/n_{j.})\texttt{log}(n_{j2}/n_{j.}), \quad \text{and}$$

$$\widehat{\Delta I}(g, \tau) = \hat{i}(\tau) - \sum_{j=1}^{m} (n_{j.}/n_{..})\hat{i}(j).$$

Using these measures of node quality, for the partitioning **with** corpus-dependent feature extraction, summarized in Table 4a, $\widehat{\Delta I}(\text{cdfe}, \tau) = 0.254$, while the partitioning

goodness **without** corpus-dependent feature extraction, summarized in Table 4b, $\widehat{\Delta I}(\text{no cdfe}, \tau) = 0.133$. So, between the two approaches, subjectively and quantitatively, in this example corpus-dependent feature extraction has produced leaves of better quality than hierarchical clustering.

We illustrate the second point, the utility of corpus-dependent feature extraction to text mining, by returning to Node 4 of the original document corpus, shown in Fig. 3a. In this node, which was produced by an iteration of Iterative Denoising on Partition 3 of the root node, a relationship was noted between two similar documents from two different classes and written by two different authors that were placed in close proximity in the Fiedler space embedding. By recomputing the features for a subset of documents, corpus-dependent feature selection has an effect of 'tuning' the features for that subset by only considering the documents in that subset. A common alternative is to reuse the features that were computed at the root node based on all the documents in the root. One benefit of having word-weights for a node be computed based on only the documents in that node is that relationships may be found that are obscured when using global-computed word-weights.

As an example, Fig. 3b shows the resulting geometric relationships when global word-weights are used instead of word-weights computed based on only the documents in a subset. While much of the global document type clusters are similar to those seen in Fig. 3a, note the geometric distance between the two similar documents that were found in close proximity when using corpus-dependent feature extraction. Here, it might be difficult for an analyst to find these similar documents, whereas before their close proximity suggested they might have some relationship. In Fig. 3b, the nearest neighbor to the Life Sciences document "Skin cells reveal they have hairy origins" by J. Travis is the Medicine document "The Y copies another chromosome's gene", also by J. Travis. The only relationship between these two documents appears to be just the author. So, in this case using global word-weights obscured the topic area relationship between two documents that might have been of interest to an analyst, a relationship that was discovered by Iterative Denoising using corpus-dependent feature extraction.

5 Conclusions

Though further analysis is warranted, we have detailed our Iterative Denoising framework and have demonstrated its performance on a heterogeneous document corpus. In our analysis of a real-world dataset, we illustrated the usefulness of our methodology for allowing an analyst to discover potentially meaningful relationships in high-dimensional data. These results suggest that Iterative Denoising can assist the user in the discovery of relationships of interest between documents. For example, the user may have a known document that he/she wishes to place in context with other unknown documents, and so visually comparing the proximity of the known document with the unknown documents may provide useful information. Also, the user may wish to uncover common themes within a large corpus, and so the user would explore the different identified clusters to find broad categories or ideas shared among the documents. Thus, the adjudication of these tasks can be effectively and practically aided by Iterative Denoising.

References

Alpert C, Kahng A (1995) Recent directions in netlist partitioning: a summary. Integr VLSI J 19(1):1–81

Arnoldi W (1951) The principle of minimized iterations in the solution of the matrix eigenvalue problem. Q J Appl Math 9:17–29

Arya S, Mount D, Netanyahu N, Silverman R, Wu A (1998) An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. J ACM 45(6):891–923

Banerjee S, Pedersen T (2003) The design, implementation, and use of the ngram statistics package. In: Proceedings of the fourth international conference on intelligent text processing and computational linguistics. Mexico City, Mexico

Belkin M, Niyogi P (2003) Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput 15(6):1373–1396

Berk R (2006) An introduction to ensemble methods for data analysis. Sociol Methods Res 34(3):263–295

Clarkson K (1999) Nearest neighbor queries in metric spaces. Discrete Comput Geom 22(1):63–69

Cormack R (1971) A review of classification (with discussion). J R Stat Soc Ser A (General) 134(3):321–367

Critchley F (1988) On certain linear mappings between inner-product and squared-distance matrices. Linear Algebra Appl 105:91–107

de Leeuw J (1988) Convergence of the majorization method for multidimensional scaling. J Classif 5:163–180

Donoho D, Grimes C (2003) Hessian eigenmaps: locally linear embedding techniques for high-dimensional data. Proc Natl Acad Sci 100(10):5591–5596

Everitt B (1993) Cluster analysis, 3rd edn. Halsted Press, New York

Faloutsos C, Lin K (1995) FastMap: a fast algorithm for indexing, data-mining, and visualization of traditional and multimedia datasets. In: Proceedings of the 1995 ACM SIGMOD international conference on management of data, pp 163–174

Fiedler M (1973) Algebraic connectivity of graphs. Czech Math J 23(98):298–305

Garey M, Johnson D, Stockmeyer L (1974) Some simplified NP-complete problems. In: Proceedings of the sixth annual ACM symposium on theory of computing, pp 47–63

Giles K (2006) Knowledge discovery in computer network data: a security perspective. Ph.D. dissertation. Johns Hopkins University, Baltimore

Gionis A, Indyk P, Motwani R (1999) Similarity search in high dimensions via hashing. In: Proceedings of 25th VLDB conference, pp 518–529

Gordon A (1999) Classification, 2nd edn. Chapman & Hall/CRC, Boca Raton

Gower J (1966) Some distance properties of latent root and vector methods in multivariate analysis. Biometrika 53:325–338

Grosjean J, Plaisant C, Bederson B (2002) Spacetree: supporting exploration in large node link tree, design evolution and empirical evaluation. In: Proceedings of IEEE symposium on information visualization, pp 57–64

Hendrickson B, Leland R (1995) A multilevel algorithm for partitioning graphs. In: Supercomputing '95: Proceedings of the 1995 ACM/IEEE conference on supercomputing (CDROM), ACM Press

Houle M (2003) Sash: a spatial approximation sample hierarchy for similarity search, Technical Report RT-0517, IBM Tokyo Research Laboratory

Houle M, Sakuma J (2005) Fast approximate similarity search in extremely high-dimensional data sets. In: 21st International Conference on Data Engineering, pp 619–630

Indyk P, Motwani R (1998) Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of 30th ACM symposium on theory of computing, pp 604–613

Kanungo T, Mount D, Netanyahu N, Piatko C, Silverman R, Wu A (2004) A local search approximation algorithm for k-means clustering. Comput Geom Theory Appl 28:89–112

Karypis G, Kumar V (1998) A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J Sci Comput 20(1):359–392

Kernighan B, Lin S (1970) An efficient heuristic procedure for partitioning graphs. Bell Syst Tech J 49(2):291–307

Kushilevitz E, Ostrovsky R, Rabani Y (1998) An algorithm for approximate closest-point queries. In: Proceedings of the 30th ACM symposium on theory of computing, pp 614–623

Lanczos C (1950) An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. J Res Natl Bur Stand 45(4):255–282

Lehoucq R, Yang C (1998) ARPACK users guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods. SIAM, Philadelphia

Lin D, Pantel P (2002) Concept discovery from text. In: Proceedings of conference on computational linguistics, pp 577–583

Mirkin B (2005) Clustering for data mining: a data recovery approach. Chapman & Hall/CRC, Boca Raton

Porter M (1980) An algorithm for suffix stripping. Program 14(3):130–137

Priebe C, Marchette D, Healy D (2004a) Integrated sensing and processing decision trees. IEEE Trans Pattern Anal Mach Intell 26(6):699–708

Priebe C, Marchette D, Park Y, Wegman E, Solka J, Socolinsky A, Karakos D, Church K, Guglielmi R, Coifman R, Lin D, Healy D, Jacobs M, Tsao A (2004b) Iterative denoising for cross-corpus discovery. In: Antoch J (ed), COMPSTAT: Proceedings in computational statistics, 16th symposium. Physica-Verlag, Springer, pp 381–392

Roweis S, Saul L (2000) Nonlinear dimensionality reduction by locally linear embedding. Science 290(5500):2323–2326

Saerens M, Fouss F, Yen L, Dupont P (2004) The principal components analysis of a graph and its relationships to spectral clustering. In: Proceedings of the 15th European conference on machine learning. Lecture Notes in Artificial Intelligence, pp 371–383

Schalkoff R (1991) Pattern recognition: statistical structural and neural approaches. Wiley, New York

Tenenbaum J, DeSilva V, Langford J (2000) A global geometric framework for nonlinear dimensionality reduction. Science 290(5500):2319–2322

Torgerson W (1952) Multidimensional scaling: I theory and method. Psychometrika 17:401–419

Trosset M, Groenen P (2005) Multidimensional scaling algorithms for large data sets. Comput Sci Stat