

Model-based Subspace Correlation Clustering

Rave Harpaz*, Robert Haralick

Pattern Recognition Laboratory, CS Dept.
The Graduate Center, City University of New York
365 Fifth Avenue New York, N.Y. 10016[†]
rbharpaz@sci.brooklyn.cuny.edu, haralick@netscape.com

Abstract

The detection of correlations is an important data mining task because correlations may reveal a dependency or some cause and effect relationship between the features under consideration. In data modeling correlations may also be used to reduce the dimensionality of the data. In recent applications of clustering such as microarray analysis, collaborative filtering, and text mining, object similarity is no longer measured by physical distance, but rather by the behavior patterns objects manifest. Coherent behavior patterns induce large correlations among their defining features. Hence, the identification of correlations is a means by which so called *pattern clusters* can also be discovered. Current state of the art algorithms typically postulate specific cluster models that are able to capture only specific behavior patterns, and omit the possibility that other information carrying patterns may co-exist in the data. This in turn may lead to a large bias in the results. We cast the problem of searching for clusters that induce large correlations among some subset of features into the problem of searching for groups of points that fit lines. The advantage of this approach is that it allows the clustering of different behavior patterns or correlations simultaneously. It also allows the clustering of patterns that are overlooked by existing methods. A formal stochastic line cluster model is presented and its connection to correlation is established. Based on this model an algorithm designed to search for line clusters embedded in subspaces of the data is discussed. The algorithm uses a stochastic model fitting technique, feature selection, and a random walk. A broad evaluation on synthetic data demonstrates that our method attains high accuracy in cluster and subspace detection, and superiority over a related method. Applied on real data we show that our method is able to identify statistically significant clusters, some which are overlooked by related methods.

Keywords: Clustering, Correlation, Subspace, Random Walk, Feature Selection.

*Corresponding author

[†]Tel. 1-212-817-8192 Fax 1-212-817-1510

1 Introduction

Interest in clustering as a data mining technique has increased substantially in recent years due to new areas of application such as DNA microarray analysis in bioinformatics, text clustering, and recommendation or collaborative filtering systems in E-commerce. Many of these applications are now characterized by high dimensional data. An important advance in this area was the introduction of *subspace clustering* [1, 2, 3, 4] in an attempt to face the new challenges posed by high dimensional data. A subspace cluster consists of a subset of points and a corresponding subset of features (dimensions), such that these points form a dense region in a subspace defined by the set of corresponding features.

Traditional clustering methods including those used in subspace clustering focus on grouping objects with similar values. They define object similarity by the “physical” distance between the objects over all or a subset of dimensions, which in turn may not be adequate to capture correlations in the data. A set of points may be located far away from each other yet induce large correlations among some subset of dimensions. The detection of correlations is an important data mining task because correlations may reveal a dependency or some cause and effect relationship between the features under consideration. Another important application of correlations is in data modeling where correlations may be used to carry out (local) dimensionality reduction by eliminating correlated (redundant) features. In recent studies these correlations were often discussed and presented in terms of the behavior patterns objects manifest, hence the name *pattern clustering* often associated with methods aimed at this type of problem. In gene expression microarray clustering the goal is to identify groups of genes that exhibit similar expression patterns under some subset of conditions (dimensions), from which gene function or regulatory mechanisms may be inferred. In recommendation or collaborative filtering systems, sets of customers with similar interest patterns need to be identified so that customers’ future interests can be predicted and proper recommendations be made. From a correlation point of view it can be shown that objects exhibiting coherent behavior patterns induce large correlations among their defining features. Hence, the identification of large correlations is a means by which *pattern clusters* can also be discovered. Because pattern clusters are based on specific linear models they capture only specific linear dependencies between features, which in turn give rise to specific types of (linear) correlations.

The most widely studied pattern cluster models are the *shift* and *scaling* models, which induce

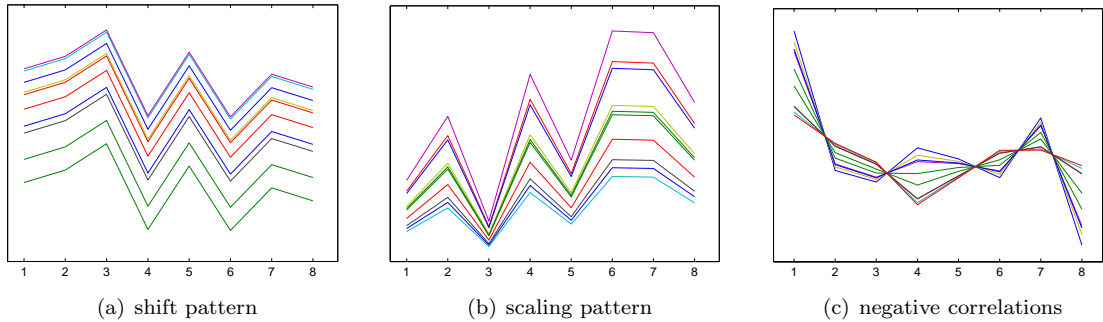


Figure 1: Parallel coordinate plots of three different pattern clusters.

only positive correlations and are typically referred to as *biclusters* [5, 6, 7] in the microarray clustering literature, and *slope one* clusters in the collaborative filtering literature [8]. In the case of a shift pattern the behavior pattern of one object under a set of features is offset from another by some constant, whereas in the case of scaling the behavior pattern of one object is a scalar multiple of another. The corresponding linear dependencies between each pair of features x_i, x_j captured by these two types of patterns are of the form $x_j = x_i + c_{ij}$ and $x_j = b_{ij}x_i$ respectively, where b_{ij} and c_{ij} are constant coefficients. From the above it is now clear why the shift and scaling patterns induce large correlations. Fig. 1 shows *parallel coordinate* plots of three different types of patterns clusters each containing ten points embedded in an 8-dimensional space: a shift pattern inducing only positive correlations, a scaling pattern also inducing only positive correlations, and a pattern inducing both positive and negative correlations. These type of plots are used to emphasize symmetry or cohesion in behavior patterns. Note that the pattern inducing negative correlations does not manifest the same symmetry as the other two.

In recent studies [9, 10, 11, 12] it has been suggested that other types of information carrying patterns such as patterns inducing negative correlations or patterns capturing more complex linear dependencies, such as $x_j = b_{ij}x_i + c_{ij}$ of which the shift and scaling are special cases, are completely overlooked by most clustering methods, and that current state of the art algorithms are not flexible enough to mine different patterns simultaneously. It has also been suggested [7] that traditional similarity measures, such as the *cosine* or the *Pearson correlation* measures, are not adequate to capture correlations or pattern clusters when those localize to subspaces of the data, as they are strongly biased by the data residing in the “irrelevant” dimensions. While there is no consensus on what types of patterns should be considered meaningful, in practice pattern based clustering algorithms

postulate a unique underlying “globally expressed” pattern or cluster model, while overlooking or rejecting the possibility that other types of information carrying patterns may co-exist in the data. This in turn is less truthful to the data and may lead to a large bias in the results.

In recent work [13] it was shown that different types of pattern clusters, can all be generalized to linear manifolds¹ of which a line (a 1D linear manifold) is a special case. The main insight provided by the study is that within the set of relevant features to a pattern cluster, the points form a line and in the full space a linear manifold of higher dimension. Hence by searching for lines or linear manifolds rather than specific patterns it is possible to cluster simultaneously all the different types of pattern clusters. In the same study it was also found that the geometrical difference between the different pattern clusters is the orientation (e.g. a line cluster of slope one represents a shift pattern) and translation of the cluster in the space. Fig. 2 shows the geometry of three different types of pattern clusters each including 10 points embedded in a 3-dimensional space. Notice how they all form lines. In [14] an algorithm designed to identify lower dimensional linear manifold clusters is presented. While effective in identifying linear manifolds clusters, it suffers from several shortcomings that render it from truly exploiting the full potential that linear manifolds may offer to the problem of pattern and correlation clustering. One of its main problems is its inability to cluster lower dimensional pattern or correlation clusters as these correspond to higher dimensional linear manifolds.

In this paper we present a new method designed to concurrently identify different types of correlations and pattern clusters, including those overlooked by existing methods, which are embedded in subspaces of the data. Our method is based on the observation that correlations manifest themselves as lines in the data space, and thus cast the problem into the problem of searching for groups of points that fit lines (line clusters) in subspaces of the data. We start by presenting a formal stochastic “line cluster” model in section 2, and present a general result that establishes its connection to correlations. In section 3 we present an algorithm called SLCLUS (Subspace Line CLUStering), which is based on a stochastic *model fitting* technique. It uses a *line detector* procedure to search for line clusters in subspaces of the data, a *forward-feature-selection* technique to extend and refine a cluster, and a *random walk* on the feature’s lattice to select an initial set of features to initiate the clustering process. In section 4 we discuss how to set the inputs to the algorithm, while in section 5 we discuss its algorithmic complexity. In section 6 an extension to fuzzy clustering is proposed, in

¹A linear manifold is a translated subspace. A subspace is a subset of points closed under linear combination.

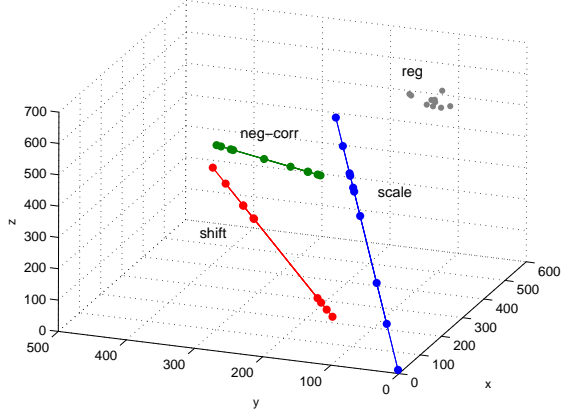


Figure 2: The geometry of the three pattern clusters in the data space. All are manifested by lines but are oriented and translated differently depending on the type of pattern they manifest. A regular cluster (reg) is also plotted to emphasize the difference between pattern and regular clusters.

section 7 an empirical evaluation of the algorithm on both real and synthetic data is presented, and in section 8 we conclude the paper.

2 The Line Cluster Model

Suppose a line cluster X exists in a k -dimensional axis-parallel subspace². Let \mathbf{x} be a $k \times 1$ vector representing some point in X , β be a unit norm $k \times 1$ vector that spans a $1D$ subspace, $\bar{\beta}$ be a $k \times k-1$ matrix whose $k-1$ column vectors form an orthonormal basis that spans the space orthogonal to the space spanned by β .

Definition 1 (The Line Cluster Model). *Let μ be some point in \mathbb{R}^k , ϕ be a zero mean random scalar distributed according to $U(-R/2, +R/2)$ where R is the range of the data, and ϵ be a $k-1 \times 1$ random vector distributed according to $N(\mathbf{0}, \sigma^2 I)$, where $\sigma \ll R$. Then each $\mathbf{x} \in X$ is modeled by,*

$$\mathbf{x} = \mu + \beta\phi + \bar{\beta}\epsilon. \quad (1)$$

Explanation: The idea is that each point in a cluster lies close to a line (1D linear manifold) of finite extent, which is defined by μ , a translation vector, the space spanned by the vector β , and the

²The term subspace in the context of clustering is often misused to indicate a subset of the original measurement features (axis-parallel subspace) which is a special case of a subspace. We will follow the trend and leave it to the reader to distinguish between the two throughout the paper.

range parameter R . Since $E[\phi] = 0$, $E[\epsilon] = \mathbf{0}$, and

$$E[\mathbf{x}] = E[\boldsymbol{\mu} + \boldsymbol{\beta}\phi + \overline{\boldsymbol{\beta}}\epsilon] = \boldsymbol{\mu} + \boldsymbol{\beta}E[\phi] + \overline{\boldsymbol{\beta}}E[\epsilon] = \boldsymbol{\mu} + \mathbf{0} + \mathbf{0} = \boldsymbol{\mu},$$

the cluster mean is $\boldsymbol{\mu}$. On the line the points are assumed to be uniformly distributed in direction $\boldsymbol{\beta}$ according to $U(-R/2, +R/2)$, where ϕ can be viewed as the displacement or distance of a point from the line's center. The assumption of uniformity is not binding, and can be replaced by any other distribution with mean zero. What characterizes this type of cluster is the third component that models a small random error associated with each point on the line. The idea is that each point may be perturbed in directions that are orthogonal to the subspace spanned by $\boldsymbol{\beta}$, that is the subspace spanned by the $k - 1$ columns of $\overline{\boldsymbol{\beta}}$. We model this behavior by requiring that ϵ be a $k - 1 \times 1$ random vector, normally distributed according to $N(\mathbf{0}, \sigma^2 I)$, where σ is much smaller than R . The requirement that ϵ has a diagonal covariance matrix with equal variances σ^2 along the diagonal is common to other quantitative models such as the shift, scaling, and regression models that assume homoscedasticity (constant variance). This also simplifies the model making statistical inference (done later) easier. The error, ϵ , can be thought of as the displacement or distance of a point to its projection onto the line, which essentially transforms the line into a thin elongated cylinder.

2.1 Lines and Correlations

Fundamental to our method is the connection between line clusters and correlation, established by the following proposition.

Proposition 1. *A set of points induce perfect correlations among a set of k features if and only if the set of points perfectly fit a line cluster in this set of features.*

Proof: Following the model given in eq. (1) a set of points will perfectly fit a line cluster if they do not include an error term which translates them away from the line. Formally, a perfect fit implies that $\epsilon = 0$ and that each point within the set of k features can be modeled by $\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\beta}\phi$. By perfect correlations we mean that the correlation coefficient ρ_{ij} between any pair of features i and j or equivalently the random components x_i and x_j of the random vector \mathbf{x} , are equal to 1 or -1 . The proposition can now be stated formally as

$$\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\beta}\phi \Leftrightarrow \forall i, j \in \{1, \dots, k\} \rho_{ij} = \pm 1.$$

(\Rightarrow): Given $\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\beta}\phi$, each component or feature of \mathbf{x} denoted by x_i is equal to $\mu_i + \phi\beta_i$ where μ_i and β_i are the i -th components of vectors $\boldsymbol{\mu}$ and $\boldsymbol{\beta}$. Hence,

$$\text{Var}[x_i] = \text{Var}[\mu_i + \phi\beta_i] = \beta_i^2 \text{Var}[\phi],$$

$$\begin{aligned} \text{Cov}(x_i, x_j) &= \text{E}[x_i x_j] - \text{E}[x_i]\text{E}[x_j] = \text{E}[(\mu_i + \phi\beta_i)(\mu_j + \phi\beta_j)] - \mu_i \mu_j \\ &= \text{E}[\mu_i \mu_j + \mu_i \phi \beta_j + \mu_j \phi \beta_i + \phi^2 \beta_i \beta_j] - \mu_i \mu_j \\ &= \mu_i \mu_j + 0 + 0 + \beta_i \beta_j \text{E}[\phi^2] - \mu_i \mu_j \\ &= \beta_i \beta_j \text{E}[\phi^2] = \beta_i \beta_j \text{Var}[\phi], \end{aligned}$$

and therefore

$$\begin{aligned} \rho_{ij} &= \frac{\text{Cov}(x_i, x_j)}{\sqrt{\text{Var}[x_i]}\sqrt{\text{Var}[x_j]}} = \frac{\beta_i \beta_j \text{Var}[\phi]}{\sqrt{\beta_i^2 \text{Var}[\phi]}\sqrt{\beta_j^2 \text{Var}[\phi]}} \\ &= \frac{\beta_i \beta_j \text{Var}[\phi]}{\sqrt{\beta_i^2 \beta_j^2 \text{Var}[\phi]}} = \frac{\beta_i \beta_j}{|\beta_i \beta_j|} = \pm 1. \end{aligned}$$

(\Leftarrow): Given $\forall i, j \in \{1, \dots, k\}$ $\rho_{ij} = \pm 1$. Assume $\text{Var}[x_i] = \alpha_i^2$ and $\text{Var}[x_j] = \alpha_j^2$, and let $\frac{x_i}{\alpha_i} - \frac{x_j}{\alpha_j}$ be a new random variable (r.v.). Then,

$$\begin{aligned} \text{Var}\left[\frac{x_i}{\alpha_i} - \frac{x_j}{\alpha_j}\right] &= \text{Var}\left[\frac{x_i}{\alpha_i}\right] + \text{Var}\left[\frac{x_j}{\alpha_j}\right] - 2\text{Cov}\left(\frac{x_i}{\alpha_i}, \frac{x_j}{\alpha_j}\right) \\ &= \frac{\text{Var}[x_i]}{\alpha_i^2} + \frac{\text{Var}[x_j]}{\alpha_j^2} - 2\frac{\text{Cov}(x_i, x_j)}{\alpha_i \alpha_j} \\ &= 1 + 1 - 2\rho_{ij} = 2(1 - \rho_{ij}). \end{aligned}$$

Now $\rho_{ij} = 1$ implies that $\text{Var}\left[\frac{x_i}{\alpha_i} - \frac{x_j}{\alpha_j}\right] = 0$ which in turn implies that the r.v. $\frac{x_i}{\alpha_i} - \frac{x_j}{\alpha_j} = c$, i.e., equals a constant, from which we can then establish a linear relationship between x_i and x_j of the form $x_j = b_{ij}x_i + c_{ij}$. Similarly, $\rho_{ij} = -1$ implies that $\text{Var}\left[\frac{x_i}{\alpha_i} + \frac{x_j}{\alpha_j}\right] = 0$, which again establishes a linear relationship between x_i and x_j of the form $x_j = b_{ij}x_i + c_{ij}$. By introducing $\beta_i, \beta_j, \mu_i, \mu_j$, where $b_{ij} = \beta_j/\beta_i$ and $c_{ij} = -\beta_j\mu_i/\beta_i + \mu_j$, and substituting them into $x_j = b_{ij}x_i + c_{ij}$ we get

$$\frac{x_i - \mu_i}{\beta_i} = \frac{x_j - \mu_j}{\beta_j}.$$

Since both sides of the equation define a r.v. we may choose to call this r.v. ϕ , yielding that

$$x_i = \mu_i + \beta_i \phi \quad \text{and} \quad x_j = \mu_j + \beta_j \phi.$$

Now collecting all the scalar equations of the form above, and putting then in vector format gives the final result that $\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\beta}\phi$. \square

Using elements of the this proof it can also be shown that the more a set of points deviates from a predefined line (the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\beta}$ are fixed) modeled by eq. (1), the less correlated the features underlying the points will be, where the amount of correlation depends on the size of σ^2 .

3 The Algorithm

Supported by proposition 1 the problem of searching for groups of points which induce large correlations in subspaces (subsets of features/dimensions) of the data is cast into the problem of searching for line clusters embedded in subspaces of the data. We reiterate, that by searching for correlations, pattern clusters as a special case of correlations can also be identified. The line clustering algorithm called SLCLUS (Subspace Line CLUStering) is based on a stochastic *model fitting* technique. It uses a *line detector* procedure to search for line clusters in subspaces of the data, a *forward-feature-selection* technique to extend and refine a cluster, and a *random walk* on the feature's lattice to select an initial set of features to initiate the clustering process. The algorithm in its most generic form can be stated as follows: find a line cluster in an initial set of dimensions using a random walk on the feature's lattice. Using forward-feature-selection add dimensions (features) to extend and refine the line cluster until no more dimensions can be added, in which case a line cluster is assumed to be found. Remove the identified line cluster from data set and reapply the first two steps on remaining set of points. The algorithm is designed so that it detects the the largest possible clusters embedded in the largest possible subspaces. The rational is that correlations induced by larger clusters in a larger set of features provide stronger evidence pertaining to the relationship between the objects under consideration, and from a statistical point of view is less likely to occur by chance.

3.1 Line Detectors

Four line detectors were evaluated amongst which the most accurate and efficient was chosen to be used by SLCLUS; a RANSAC [15] based method, two versions of LMCLUS [14], and an adaptation of K-means [16] to line clustering, all of which are stochastic (based on sampling). RANSAC (Random Sample Consensus) is a model fitting algorithm that repeatedly samples points from the data to instantiate the model parameters (e.g., two points that may define a line), finds data points that fit the model, and returns the “best” model along with its fitted points. The two versions of LMCLUS we used as part of our evaluation were forced to search only for 1D manifolds, one returning the first identified manifold, and the other the “best” among all the 1D manifolds identified. The K-means adaptation we used proceeded as normal, but instead of using points as cluster centers it uses lines as cluster centers, and returns the best cluster. Each of these algorithms was evaluated using different criteria to qualify the “best” clusters or models.

The *Hough transform* [17] which may also be used as a line detector requires the quantization of the “parameter space” into bins, and selects the bins with the largest count corresponding to a set of parameters which define a model that passes through most points. In higher dimensional spaces more parameters are required to define a model (even as simple as a line model), i.e., the dimensionality of the parameter space gets larger. As a result the average number of counts per single bin is very low, making it hard to distinguish meaningful bins from others. Moreover, in higher dimensional spaces the quantization of the parameter space and accumulation of counts renders the method from being efficient. Due to these deficiencies the Hough transform was omitted from our evaluation of line detectors.

Each of the candidate line detector methods was tested on several different synthetic data sets, and its accuracy (number of points correctly assigned) and efficiency (time in seconds) were measured. Because of their stochastic nature, each method was run several times on each data set, and the averages were recorded. We found LMCLUS and RANSAC to be the most accurate, and among the two RANSAC to be more efficient. A sample of these results is presented in table 1 for illustrative purposes. Based on this finding, in addition to RANSAC’s simplicity and relatively intuitive input parameters, we chose a RANSAC based method as our line detector. The final version of the line detector algorithm is outlined in Fig. 3. The last step (return statement) ensures that the largest line clusters are detected by the overall line clustering algorithm.

	RANSAC		LMCLUS _b		LMCLUS ₁		line K-means	
	accuracy	time	accuracy	time	accuracy	time	accuracy	time
1	0.95	0.05	1.00	2.68	1.00	1.79	0.56	0.05
2	0.95	0.16	0.69	12.2	0.41	3.26	0.58	0.40
3	0.84	0.16	0.92	9.46	0.90	3.44	0.57	0.53
4	0.93	0.35	0.91	17.2	0.93	4.13	0.62	0.78
5	0.92	0.48	0.83	21.9	0.87	4.41	0.72	1.06
6	0.91	0.89	0.61	18.1	0.64	5.03	0.73	1.62
7	0.78	0.10	0.95	6.19	0.95	3.08	0.67	0.22
8	0.91	0.16	0.86	11.9	0.84	3.77	0.72	0.32
9	0.81	0.16	0.81	10.1	0.83	3.37	0.70	0.33
10	0.74	0.48	0.47	29.5	0.41	4.18	0.46	0.90

Table 1: A summary of the line detector selection experiment. The table shows the accuracy and running time (in seconds) of each of the line detector methods applied on a sample of 10 synthetic data sets. LMCLUS_b is the version that returns the best line cluster among all identified, and LMCLUS₁ the version returning the first cluster identified.

Algorithm LineDetector (X, s, d, t)
repeat s times
 sample two points $\mathbf{x}_1, \mathbf{x}_2$ from X
 determine line parameters $\boldsymbol{\mu} = \mathbf{x}_1, \boldsymbol{\beta} = (\mathbf{x}_2 - \mathbf{x}_1) / \|\mathbf{x}_2 - \mathbf{x}_1\|$
 determine the number of points in the data that are within
 distance d ("inliers") from the line
return the line with most inliers if it includes at least t inliers

Figure 3: A RANSAC based line detector algorithm.

3.2 Feature Selection

Feature selection techniques attempt to discover the most relevant attributes of the data as part of a machine learning or model building process. In our case feature selection is used to select the best and largest possible set of features in which a set of points fits a line. Finding the optimal set of features by an exhaustive search through all possible subsets of features is typically infeasible. For this reason most feature selection techniques employ a greedy hill-climbing approach. The basic procedure involves identifying an initial model, and iteratively improving the model by adding or removing features in accordance with some criteria until there is no improvement or when a predetermined number of steps has been reached. *Forward selection* techniques are *bottom-up* methods which start with an empty or small set of features and at each step add the most “relevant” feature to the model. *Backward elimination* techniques are *top-down* approaches which start with the full set of features and remove the most “irrelevant” feature at each step. *Stepwise* methods employ a combination of two, and depending on the direction of search are called *forward stepwise* and *backward stepwise* [18]. Supported by the following proposition we chose a bottom-up approach to extend and refine line clusters.

Proposition 2 (Downward Closure Property of Lines). *If there exists a line in a set of k dimensions then there exists a line in all $k - 1$ subsets of these k dimensions.*

Proof: Based on the model given in eq. (1) and similar to the proof of proposition 1 each of the k components x_i of each point constituting a line can be modeled by $x_i = \mu_i + \beta_i\phi$. We can therefore select any subset of $k - 1$ components where each of the components is modeled as above, and join them together into a vector producing $\mathbf{x}' = \boldsymbol{\mu}' + \boldsymbol{\beta}'\phi$, the model of a $k - 1$ -dimensional line. \square

Proposition 2 tells us that if a set of points form a line cluster in some set of dimensions it is possible to commence the search for the cluster in a smaller set of dimensions, and iteratively extend it in a bottom-up manner using forward selection. Moreover, the search for clusters in lower dimensions is typically easier (faster) than a search for clusters in higher dimensions. Proposition 2 also provides pruning power. That is, if a line cluster is not visible in a smaller set of dimensions it is not necessary to search for it in higher dimensions. Using this property we can also devise a termination condition for the algorithm, i.e., if the line detector is not able to detect any more clusters in a small initial set of dimensions then the algorithm should terminate. The combination of the bottom-up approach and proposition 2 also ensures that the line clusters are found in the largest

possible subspaces.

Because a top-down approach will not be able to exploit the advantages of the downward closure property of lines we choose to adopt a bottom-up forward (or stepwise) selection approach. It now remains to determine how an initial set of features is selected to start the clustering process. One possibility is to start the process with line clusters of the smallest possible dimensionality, i.e., two-dimensional line clusters. This can be achieved by searching through all possible 2D subspaces for a line cluster using the line detector algorithm. Needless to say that this possibility is computationally feasible as it is only quadratic in the dimensionality of the data. However, confirmed by experiments presented in section 7, clusters tend to overlap when projected from higher dimensional spaces into lower dimensional spaces. Hence, the projection of several line clusters embedded in higher dimensional spaces into a lower dimensional space may either mask each other or appear as a single cluster. This in turn may “confuse” feature selection used to extend a cluster in the determination of which features are relevant to the cluster. Ideally the clustering process should start with a larger set of initial features, close in number to the dimensionality of the subspace in which some line cluster exists. This will not only improve cluster detection accuracy but also improve efficiency as the extension process will be shorter. To achieve this goal we propose a method that is based on a *random walk* on the *features lattice*.

3.3 Selecting an Initial Set of Features by Random Walk

The potential of random walk approaches in context of *level-wise* algorithms such as those frequently used to mine association rules is discussed in [19]. However, completely different from our approach, rather than walking up the itemset or features lattice the random walk used by SLCLUS walks down the features lattice so that subspaces and line clusters of higher dimensionality are examined and intercepted sooner. The basic idea can be stated as follows: starting from the full set of features, randomly remove one feature at a time, after each removal invoke the line detector algorithm to detect line clusters in the subspace defined by the remaining features, and repeat the process until a line cluster is detected, or until no more features are left to be removed. As mentioned, ideally we would like the random walk to stop sooner, i.e., after the removal of fewer features, so that initial line clusters will be intercepted at higher dimensions closer to the dimensionality of the subspace in which they are embedded.

More formally, let F be the full set of features, and \mathcal{L}_F be the lattice (poset) defined by $(\mathcal{P}(F), \subseteq)$.

If F_1 and F_2 are two elements in \mathcal{L}_F (subsets of F), we say that F_2 is more *general* than F_1 or F_1 more *specific* than F_2 denoted by $F_2 \prec F_1$ if $F_2 \subset F_1$. Let $F' \subseteq F$ be a set of features (subspace) in which some line cluster exists, and F'' be the set of features remaining after each feature removal during the walk. If at a certain point during the random walk $F'' \subseteq F'$, i.e., a subset of features that constitute a higher dimensional line cluster is detected, we can stop the random walk and use F'' as our starting point (initial set of features). Then using forward selection we can extend the line cluster currently residing in the subspace defined by the features of F'' to the higher dimensional subspace in which the cluster exists defined by the features of F' . Due to the downward closure property of lines, once this condition is met it is not necessary to continue the walk (remove features) as any subset of F'' will also contain a line cluster. Thus, the method can be restated as: perform a *random walk* on \mathcal{L}_F starting from F and moving in the *generalization* direction until either $F'' \subseteq F'$ or $F'' = \emptyset$.

In order to examine the statistical properties of the random walk we assume the line detector procedure will always be able to identify a line cluster in a subspace if such a cluster exists. Let X denote a random variable counting the number of features remaining upon termination of the random walk, i.e., the dimensionality of the subspace in which a line cluster (if one exists) is first detected. The larger X is the better. We say the random walk *fails* if it fails to detect a line cluster when one exists, i.e. when $X < 2$. Likewise we say the random walk *succeeds* when $X \geq 2$, that is, a set of at least two features that are a subset of the set of features in which a line cluster exists were encountered by the walk. Let d denote the dimensionality of the data, c the number of line clusters in the data, and k the dimensionality of the subspaces in which each of line clusters is embedded. The requirement that all the clusters reside in exactly a k -dimensional subspace and not in subspaces of different dimensionalities is considered a limiting case. We note however that these k -dimensional subspaces need not necessarily be the same subspaces. Furthermore, let $P(X = x|d, c, k)$ denote the probability that the random walk intercepted a set F'' of x features such that F'' is a subset of at least one of the feature sets F'_1, F'_2, \dots, F'_c in which some of the c k -dimensional clusters are embedded, and where the dimensionality of the full space is d , i.e.,

$$P(X = x|d, c, k) = P\left(\bigcup_{i=1}^c (F'' \subseteq F'_i \cap |F''| = x)\right).$$

Because we have no prior knowledge of the the subspaces in which the clusters exist we will assume

F'_1, F'_2, \dots, F'_c are all drawn randomly and independently with replacement (allowing for clusters to exist in the same subspaces) from \mathcal{L}_F .

The statistical properties of the walk we are interested in finding and examining are: $P(\text{success}) = P(X \geq 2|d, c, k)$, this will tell us how likely we are to intercept a subset of features that are part of a cluster. $E[X|d, c, k]$, this will tell us at which dimension or how soon we can expect to intercept such a subset of features. The larger this value the better, as it will indicate that we can expect to intercept a cluster sooner. Given d , what should k be so that $E[X|d, c, k] \geq 2$? or alternatively what should the ratio between d and k be so that $E[X|d, c, k] \geq 2$? This will tell us how large should the subspace in which clusters exists be so that we can expect the random walk to succeed, or what is the range of subspace dimensionalities in which the walk is effective. This will guide us to decide on which types of data sets (containing clusters in lower or higher dimensional subspaces) the random walk can be applied successfully. It is reasonable to assume that the larger the dimensionality of the subspaces with respect to the dimensionality of the data the more likely it is that the walk will succeed or intercept a cluster sooner. Another question we would like to answer is whether the random walk will perform better when more clusters exist in the data? It seems reasonable to assume that the answer is yes.

We first consider the simpler case in which only one cluster exists in the data and then extend the results to multiple clusters.

Lemma 1.

$$P(X \geq x|d, 1, k) = \frac{\binom{k}{x}}{\binom{d}{x}}.$$

Proof: We are given $|F| = d$, $|F'| = k$ and $X \leq k$. $X = x$ implies that the random walk yielded a subset of features F'' such that $F'' \subseteq F'$ and $|F''| = x$. This in turn happens when $d-x$ features have been removed (sampled) from the set F by the random walk, and of those features removed exactly $k-x$ are removed (sampled) from the set F' (to yield the set $F'' \subseteq F'$), and $d-x - (k-x) = d-k$ features removed (sampled) from the set $F - F'$ where $|F - F'| = d - k$. The probability of this event occurring, similar to other hypergeometric problems, is given by

$$\frac{\binom{k}{k-x} \binom{d-x}{d-x}}{\binom{d}{d-x}} = \frac{\binom{k}{x}}{\binom{d}{x}}.$$

However, this probability does not take into account the order in which the features are removed. It includes other successful random walks that stopped earlier, i.e., $X > x$ ($F'' \subseteq F'$ and $F'' > x$), but would have lead to $X = x$ if the walks were allowed to continue removing features. After all, if $F'' \subseteq F'$ then any subset of F'' will also be a subset of F' . Therefore,

$$\frac{\binom{k}{x}}{\binom{d}{x}} = P(F'' \subseteq F' \cap |F''| \geq x) = P(X \geq x | d, 1, k). \quad \square$$

Lemma 2.

$$P(X = x | d, c, k) = \sum_{i=1}^c (-1)^{i-1} \binom{c}{i} \left[\left(\frac{\binom{k}{x}}{\binom{d}{x}} \right)^i - \left(\frac{\binom{k}{x+1}}{\binom{d}{x+1}} \right)^i \right].$$

Proof: because F'_1, F'_2, \dots, F'_c are independently sampled

$$\begin{aligned} P(F'' \subseteq F'_1 \cap |F''| \geq x) &= \dots = P(F'' \subseteq F'_c \cap |F''| \geq x) \\ &= P(F'' \subseteq F' \cap |F''| \geq x) = P(X \geq x | d, 1, k) = \frac{\binom{k}{x}}{\binom{d}{x}}. \end{aligned}$$

Using the *inclusion-exclusion* principle

$$\begin{aligned} P(X \geq x | d, c, k) &= P\left(\bigcup_{i=1}^c (F'' \subseteq F'_i \cap |F''| \geq x)\right) \\ &= \sum_{i=1}^c (-1)^{i-1} \binom{c}{i} P\left(\bigcap_{j=1}^i (F'' \subseteq F'_j \cap |F''| \geq x)\right) \\ &= \sum_{i=1}^c (-1)^{i-1} \binom{c}{i} P(F'' \subseteq F' \cap |F''| \geq x)^i \\ &= \sum_{i=1}^c (-1)^{i-1} \binom{c}{i} P(X \geq x | d, 1, k)^i \\ &= \sum_{i=1}^c (-1)^{i-1} \binom{c}{i} \left(\frac{\binom{k}{x}}{\binom{d}{x}} \right)^i, \end{aligned}$$

and using the identity $P(X = x) = P(X \geq x) - P(X \geq x + 1)$ we get

$$\begin{aligned}
P(X = x|d, c, k) &= P(X \geq x|d, c, k) - P(X \geq (x + 1)|d, c, k) \\
&= \left[\sum_{i=1}^c (-1)^{i-1} \binom{c}{i} \left(\frac{\binom{k}{x}}{\binom{d}{x}} \right)^i \right] - \left[\sum_{i=1}^c (-1)^{i-1} \binom{c}{i} \left(\frac{\binom{k}{x+1}}{\binom{d}{x+1}} \right)^i \right] \\
&= \sum_{i=1}^c (-1)^{i-1} \binom{c}{i} \left[\left(\frac{\binom{k}{x}}{\binom{d}{x}} \right)^i - \left(\frac{\binom{k}{x+1}}{\binom{d}{x+1}} \right)^i \right]. \quad \square
\end{aligned} \tag{2}$$

Knowing $P(X = x|d, c, k)$ by lemma 2 allows us to compute the statistics of interest, e.g., $P(\text{success}) = 1 - (P(X = 0|d, c, k) + P(X = 1|d, c, k))$ and $E[X|d, c, k] = \sum_{i=0}^k i \cdot P(X = i|d, c, k)$. Using Figs. 4-6 we summarize our conclusions. Figs. 4 and 5 show the change in the probability of the random walk succeeding and the expected dimensionality of the clusters intercepted by the random walk respectively, as the number of clusters c in the data set and the dimensionality of the subspaces k in which the clusters exist are varied. Each curve represents a different number of clusters in the data set, where the lowest curve represents a data set with one cluster and the highest with ten clusters ($c = 1, \dots, 10$). For illustrative purposes the dimensionality of the data is fixed at 50, but similar patterns can be observed for other data dimensionalities. It is clear from Fig. 4 that as the number of clusters increases and/or the dimensionality of the subspaces in which clusters are embedded is increased the probability of success increases. It is also evident that even when a small number of clusters exist in the data but the clusters are embedded in higher dimensional subspaces there is a high probability of success. Hence, one conclusion that can be drawn is that the random walk is likely to succeed when a large (not necessarily extremely large) number of clusters exist in the data and the clusters are embedded in a relatively higher dimensional subspace. Fig. 5 similarly shows that the random walk is more effective, that is, clusters are intercepted sooner and at higher dimensions when the clusters are embedded in higher dimensional subspaces. The more effective region of the random walk seems to be approximately the upper third range of subspace dimensionalities. The figure also shows that the expected value converges to an exponential and that the addition of more clusters beyond a certain point will not enhance the capability of the random walk detecting clusters sooner. Fig. 6 shows the dimensionality of the subspace required, given varying data dimensionalities and varying number of clusters so that we can expect the random walk to succeed. By computing the slope of each of the curves we may conclude that beyond a certain

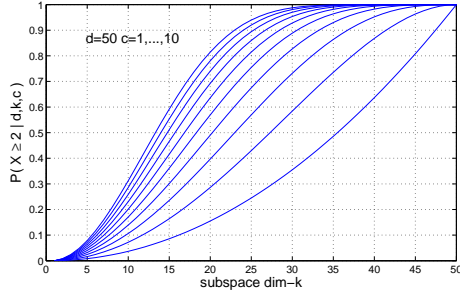


Figure 4: Probability of a random walk succeeding.

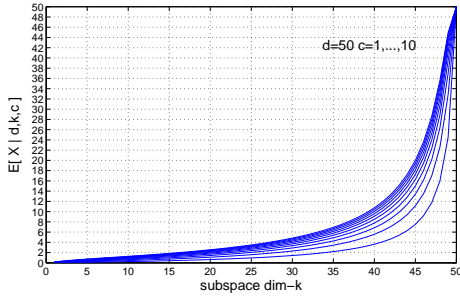


Figure 5: The expected dimensionality of a cluster intercepted by the random walk.

number of clusters the required dimensionality of the subspaces in which clusters are embedded, so that on expectation the random walk will succeed, should be approximately larger than a third of the data dimensionality. One more important conclusion that can be drawn from the figures is that if the random walk fails, it is likely that the clusters are embedded in lower dimensional subspaces, in which case we can revert to our first solution, which is to initialize the clustering process by searching for 2D line clusters in all possible 2D subspaces.

3.4 The Distance of a Point to a Line

The line detector algorithm requires the computation of a point’s distance to a line. The squared distance (henceforth distance) of a point to a line is the norm squared of its projection to the space orthogonal to the line. Formally, the distance δ of a point \mathbf{x} modeled by eq. (1) to a k -dimensional

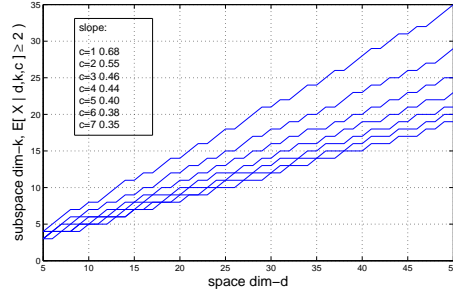


Figure 6: The required dimensionality of the subspace in which clusters exist so that we can expect the random walk to succeed.

line is given by:

$$\begin{aligned}
 \delta &= \|(I - \beta\beta')(\mathbf{x} - \boldsymbol{\mu})\|^2 = \|(I - \beta\beta')(\beta\phi + \bar{\beta}\boldsymbol{\epsilon})\|^2 \\
 &= \|\beta\phi - \beta\phi + \bar{\beta}\boldsymbol{\epsilon} - 0\|^2 = \|\bar{\beta}\boldsymbol{\epsilon}\|^2 = (\bar{\beta}\boldsymbol{\epsilon})'\bar{\beta}\boldsymbol{\epsilon} \\
 &= \boldsymbol{\epsilon}'\bar{\beta}'\bar{\beta}\boldsymbol{\epsilon} = \boldsymbol{\epsilon}'\boldsymbol{\epsilon} = \sum_{i=1}^{k-1} \epsilon_i^2.
 \end{aligned}$$

According to the line cluster model $\epsilon_i \sim N(0, \sigma^2)$. Therefore the distance δ normalized by σ^2 will have

$$\frac{\delta}{\sigma^2} = \sum_{i=1}^{k-1} \frac{\epsilon_i^2}{\sigma^2} \sim \chi_{k-1}^2,$$

a chi-squared distribution with $k - 1$ degrees of freedom. Hence,

$$\mathbb{E}[\delta] = \mathbb{E}[\sigma^2 \chi_{k-1}^2] = (k - 1)\sigma^2 \quad \text{and} \quad \text{Var}[\delta] = \text{Var}[\sigma^2 \chi_{k-1}^2] = 2(k - 1)\sigma^4.$$

Because the distance grows with the dimensionality of the subspace in which it measured, and since the search for line clusters will be computed across different dimensionalities, we normalize the distance by its degrees of freedom ($k - 1$) or equivalently the dimensionality of the space orthogonal to the line. This creates a uniform or normalized distance measure which is independent of the dimensionality of the subspace in which it is measured. Therefore the normalized distance $\delta/(k - 1)$ has

$$\mathbb{E}\left[\frac{\delta}{k - 1}\right] = \sigma^2 \quad \text{and} \quad \text{Var}\left[\frac{\delta}{k - 1}\right] = \frac{2\sigma^4}{k - 1}. \quad (3)$$

The expected value and variance of the normalized distance will be used as heuristics to set the input parameters to the algorithm, discussed in section 4.

Lemma 3. $\|(I - \beta\beta')(\mathbf{x} - \boldsymbol{\mu})\|^2 = \|\mathbf{x} - \boldsymbol{\mu}\|^2 - \|\beta'(\mathbf{x} - \boldsymbol{\mu})\|^2$.

Proof: Let $\mathbf{y} = \mathbf{x} - \boldsymbol{\mu}$,

$$\begin{aligned}
\|(I - \beta\beta')\mathbf{y}\|^2 &= \|\mathbf{y} - \beta\beta'\mathbf{y}\|^2 \\
&= (\mathbf{y} - \beta\beta'\mathbf{y})'(\mathbf{y} - \beta\beta'\mathbf{y}) \\
&= \mathbf{y}'\mathbf{y} - 2\mathbf{y}'\beta\beta'\mathbf{y} - \mathbf{y}'(\beta\beta')^2\mathbf{y} \\
&= \mathbf{y}'\mathbf{y} - \mathbf{y}'\beta\beta'\mathbf{y} \quad (\beta\beta' \text{ is Idempotent}) \\
&= \|\mathbf{y}\|^2 - \|\beta'\mathbf{y}\|^2.
\end{aligned}$$

Lemma 3 provides us a much more efficient way of computing the distance. If k is the dimensionality of the subspace in which the distance is computed, then computing it using lemma 3 gives us a speedup of $O(k)$, which for high dimensional spaces becomes a significant factor.

3.5 The Score (Fit) function

At each forward selection step the quality of the line cluster returned by the line detector must be assessed according to some criteria in order to determine whether or not to proceed to the next step. The criteria used in this paper to assess the quality of a cluster is the “fit” of the set of points constituting the cluster to the line in which they are embedded. The fit is defined to be the average-normalized-squared-distance (average error) of the points to the line.

If k is the dimensionality of the subspace in which a cluster is detected, n the number of points constituting the cluster, X the cluster points, and \mathbf{x}_i the i -th point. Then the fit or score function $J(X)$ is defined as

$$J(X) = \frac{1}{n(k-1)} \sum_{i=1}^n (\|\mathbf{x}_i - \boldsymbol{\mu}\|^2 - \|\beta'(\mathbf{x}_i - \boldsymbol{\mu})\|^2). \quad (4)$$

Prior to the fit computation, $\boldsymbol{\mu}$ and β must be estimated. $\boldsymbol{\mu}$ is estimated by computing the sample mean of the cluster. Using least-squares it can be shown that an estimate for β is the largest eigenvector of the cluster’s covariance matrix, which can be computed using the *power method*.

To guide the algorithm to give certain preferences to the size and dimensionality of the cluster,

$J(X)$ can be modified as follows:

$$J'(X) = J(X)n^a(k-1)^b. \quad (5)$$

For example guiding the algorithm to prefer even higher dimensional subspaces we can set b to some value less than zero. Based on eq. (3)

$$E[J(X)] = \sigma^2 \quad \text{and} \quad \text{Var}[J(X)] = \frac{2\sigma^4}{n(k-1)}, \quad (6)$$

which will also be used as heuristics to set the input parameters, and discussed in section 4.

3.6 Putting it All Together

SLCLUS (Fig. 7) commences by a random walk to select an initial set of features to initiate the clustering process. If the walk failed then the initial set of features is determined by searching through all possible 2D subspaces for the best 2D line cluster. If no such cluster exists then SLCLUS terminates. If either the random walk succeeds or a 2D line cluster is detected, then SLCLUS proceeds by repeatedly calling the *forward selection* procedure (Fig. 8) to extend the cluster into higher dimensions and refine it, until no “improvement” can be made. A cluster is improved if its fit $J(X)$ decreased. At this point a line cluster is assumed to be found, it is outputted, removed from the data, and the algorithm is reapplied on the remaining set of points until no more points are left to be clustered. The forward selection procedure extends a cluster one dimension at a time, by choosing the dimension whose data when added to an existing cluster, attains the maximum reduction in the fit $J(X)$. We note that by calling the line detector procedure from within the forward selection procedure the algorithm ensures that line clusters are refined by also peeling off unrelated points from them. This refinement is necessary when the projection of several line clusters appear as one line cluster in lower dimensional subspaces. In addition, if the random walk fails it is likely that a cluster is embedded in a lower dimensional subspace, in which case searching through all possible 2D subspaces is likely to reveal it since it is less masked or overlapped by other cluster projections.

```

Algorithm SLCLUS ( $D, s, d, t, J$ )
 $J(X) = \infty$ 
 $X = \text{perform random walk}(D)$ 
if ( $J(X) > J$ )
     $X = \text{find the best 2D-line cluster}(D)$ 
    if ( $J(X) > J$ )
        terminate
while ( $|dims(X)| < |dims(D)|$ )
     $X' = \text{ForwardSelection}(X, s, d, t)$ 
    if ( $J(X') < J(X)$ )
         $X = X'$ 
    else
        break
output  $X$ 
 $D = D - pts(X)$ 
if ( $D \neq \emptyset$ )
    goto first step
return

```

Figure 7: The subspace line clustering algorithm. D is the data set. X is a line cluster data structure containing the set of points in the cluster ($pts(X)$) and the set of cluster dimensions ($dims(X)$). X is updated whenever points/dimensions are added/removed from the cluster.

```

Algorithm ForwardSelection ( $X, s, d, t$ )
 $Y = X$ 
while (unexamined dimensions of  $X$  remain)
    select an unexamined dimension of  $X$ 
    add dimension data to  $X$ 
     $X' = \text{LineDetector}(X, s, d, t)$ 
    if ( $J(X') < J(Y)$ )
         $Y = X'$ 
    restore  $X$ 
return  $Y$ 

```

Figure 8: The forward selection subroutine used to gradually extend the subspace in which line clusters are detected and to peel off points which do not belong to the cluster.

4 Setting the Input Parameters

The algorithm requires the input of four parameters; s , a sample size parameter used to specify the maximum number of attempts that should be made by the line detector to identify a line cluster; d , the maximum distance of a point to a line allowed for it to be considered an “inlier”, i.e., d can be considered an error tolerance; t , a threshold used to specify that a large enough cluster has been detected; J , a “fit” threshold used to determine whether the algorithm should terminate, i.e., that no more clusters with a sufficient enough fit are left in the data.

t is relatively intuitive but may require some domain knowledge about the number of points we expect to see in a cluster. For example, in gene expression clustering typical functional categories contain a small amount of genes, and therefore t should be set to a low value.

d and J are also relatively intuitive and easy to set. They pertain to the error tolerance or deviation from the line we are willing to allow, and indirectly effect the magnitude of correlations the generated clusters will induce. E.g., setting them to higher values will likely generate clusters inducing lower correlations. Assuming we are willing to accept clusters whose average deviation (distance) from a line is σ (i.e, we are assuming that $\epsilon \sim N(\mathbf{0}, \sigma^2 I)$), then using the statistics derived in eq. (3) and eq. (6) as heuristics, we can set d and J to their expected value plus a number of their standard deviations,

$$d = \sigma^2 + c\sigma^2 \sqrt{\frac{2}{(k-1)}}, \quad (7)$$

and

$$J = \sigma^2 + c\sigma^2 \sqrt{\frac{2}{n(k-1)}}, \quad (8)$$

where c is the number of standard deviations. Since these values depend on k -the dimensionality of the subspace in which a cluster is either searched for or reported, the two parameters must be adjusted dynamically by the algorithm depending on k . Hence, the two parameters should simply be set to $d = J = \sigma$.

s should be selected large enough to ensure with high probability that at least one of the samples of two points are within error tolerance to a line, i.e., come from the same line cluster. This in turn will ensure that the sampled points can be used to approximate the line in which a possible cluster is embedded and to collect its remaining points. Let p be the probability that two sampled points come from the same cluster, X be a geometric random variable denoting the number of trails (samples)

K	p	s	
		$\epsilon = 0.05$	$\epsilon = 0.01$
2	0.2500	10	16
4	0.0625	46	71
6	0.0278	106	163
8	0.0156	190	292
10	0.0100	298	458
12	0.0069	430	661
14	0.0051	586	900
16	0.0039	765	1177
18	0.0031	969	1490
20	0.0025	1197	1840

Table 2: Values for input parameter s .

needed to get one success (two points coming from the same cluster), and $F(X)$ its cumulative distribution function. If we want to ensure with probability $1 - \epsilon$ (where $0 < \epsilon < 1$) a success then the number of trials s needed should satisfy

$$F(X) = P(X \leq s) = 1 - (1 - p)^s \geq 1 - \epsilon,$$

yielding that s should be set to

$$s \geq \frac{\log \epsilon}{\log(1 - p)}. \tag{9}$$

It now remains to determine p , the probability that two sampled points come from the same cluster. One way is to assume that there are no more than K clusters of approximately the same size in the data set, and set $p = 1/K^2$. Another way is to use t and set $p = (|D|/t)^2$, where $|D|$ is the number of points being clustered. Table 2 uses $p = 1/K^2$ to show some values of s , for corresponding values of K and ϵ .

5 Algorithmic Complexity and Optimizations

SLCLUS's overall worst case time complexity in terms of the size of the data- n , the dimensionality of the data- d , the largest dimensionality of the subspaces in which line clusters are embedded- k , is $O(n, d^3, k^3)$. Due to space limitations a detailed proof is omitted. Nonetheless it can easily be derived by observing that the complexity of line detector procedure is $O(nk)$, the complexity of estimating the model parameters is $O(nk^2)$, the complexity of the procedures used to initiate the clustering process (finding the best 2D line cluster, and the random walk) is $O(d^2n)$ and $O(nd^3)$,

and the complexity of the forward selection procedure is $O(dni^2)$, where i is the dimensionality of the subspace currently being examined.

The flexibility of SLCLUS comes at a price. Its main bottleneck is associated with the estimation of the line parameter β which is used to compute the fit $J(X)$. Rather than estimating it by computing a covariance matrix and its associated largest eigenvector using the power method, it is possible to obtain slightly lesser accurate estimates of it by other methods. One possible method is to use the two points sampled by the line detector as a less accurate estimate. That is, along with the line cluster points, the line detector procedure also returns the corresponding parameters $\mu = \mathbf{x}_1$ and $\beta = (\mathbf{x}_1 - \mathbf{x}_2)/\|\mathbf{x}_1 - \mathbf{x}_2\|$ that were used to identify the line. This will result in an algorithm whose complexity is $O(n, d^2, k^2)$. Another more accurate estimate of β can be obtained using a monte carlo approach similar to the one used by the line detector procedure. After a line is detected, using only the line points, we repeat s times: sample two points $\mathbf{x}_1, \mathbf{x}_2$, let $\mu = \mathbf{x}_1$, and let $\beta = (\mathbf{x}_1 - \mathbf{x}_2)/\|\mathbf{x}_1 - \mathbf{x}_2\|$, compute $J(X)$ the fit, and return the β that yielded the best fit, i.e., smallest value of $J(X)$. The rationale is that because we are using only the line points it is likely (can be shown mathematically) that with a large enough sample size we will be able to sample two points that can define a line which is very close to the true line that passes through the data. Also because the true β is obtained through a least squares method that essentially seeks to minimize $J(X)$, the smaller it is, the closer the corresponding sampled β is to its true value. The resulting complexity of the algorithm using this method is also $O(n, d^2, k^2)$. We note however that even without the optimizations the complexity of SLCLUS with respect to the dimensionality of the subspaces in which clusters are embedded is only cubic, as opposed to related subspace clustering algorithms such as CLIQUE [1] and MAFIA [3], which are exponential in the dimensionality of the subspaces.

6 Extensions to Fuzzy and Soft Clustering

In many applications like gene expression clustering some genes may belong to multiple functional categories, and thus “hard” (discrete) clustering may not always be adequate. Unlike hard clustering, “soft” clustering allows for objects to belong to multiple clusters, whereas “fuzzy” clustering associates each object with each cluster by its degree of membership typically ranging in $[0, 1]$.

Upon termination of the algorithm the data is first modeled as a mixture of line cluster models

based on the generated clusters. For each of the generated clusters we estimate the line parameters $\boldsymbol{\mu}$ and $\boldsymbol{\beta}$ and compute two histograms h and \bar{h} corresponding to estimates of the pdf's of the cluster points on the line and their distance away from the line respectively. h and \bar{h} may be thought of as estimates of the pdf's of the random variables ϕ and $\|\boldsymbol{\epsilon}\|^2$ appearing in eq. (1). Using h and \bar{h} we can then estimate the density of a point \mathbf{x} given a cluster C (or its model parameters) as

$$p(\mathbf{x}|C) = h(\boldsymbol{\beta}'(\mathbf{x} - \boldsymbol{\mu}))\bar{h}(\|(I - \boldsymbol{\beta}\boldsymbol{\beta}')(\mathbf{x} - \boldsymbol{\mu})\|^2). \quad (10)$$

That is, the density of a point given a line cluster is the joint probability of its component on the line (projection onto the line) and its component away from the line (distance). Because ϕ and $\boldsymbol{\epsilon}$ are assumed independent according to the line cluster model (the noise distribution is independent of the signal's distribution), this joint probability is simply the product of the marginals estimated by h and \bar{h} . Using $p(\mathbf{x}|C)$ we can then devise various probabilistic measures of association/membership which can be used to generate soft, fuzzy, and hard clusterings. Assuming the clustering generated K clusters C_1, C_2, \dots, C_K . Let $P(C_i)$ be the fraction of points in cluster C_i , then using a Bayesian approach, the measure of association of a point \mathbf{x} to cluster C_i is given by

$$P(C_i|\mathbf{x}) = \frac{p(\mathbf{x}|C_i)P(C_i)}{\sum_{j=1}^K p(\mathbf{x}|C_j)P(C_j)}, \quad (11)$$

establishing the basis of a fuzzy clustering. To perform soft clustering, we simply select the clusters for which $P(C_i|\mathbf{x}) \geq \tau$, where τ is some predetermined threshold. Similarly, to assign points to clusters in a hard way, we select the cluster for which $P(C_i|\mathbf{x})$ is the largest. This scheme of classification also allows us to assign newly unobserved points to clusters.

7 Empirical validation

We experimented with SLCLUS by applying it on both real and synthetic data sets. The algorithm was implemented in a Linux based Matlab environment.

7.1 Experiments with Synthetic Data

To better understand the algorithm, several dozen data sets were generated according to the line cluster model specified in eq. (1). The aim of the experiment with the synthetic data was to evaluate

the algorithm’s accuracy in the detection of both the clusters that were generated and the subspaces in which they were embedded, along with an empirical evaluation of the random walk’s effectiveness and its contribution to the algorithm’s performance.

To determine the algorithm’s accuracy in cluster detection we used an *external cluster validity* criteria called *Cluster Purity* [20], which measures the degree of correspondence between the point class labels of each output cluster with its mapped input cluster (input cluster with the largest number of points in common), weighted by the size of the output cluster. The determination of the algorithm’s accuracy in subspace detection was measured by the degree of correspondence between the subspace feature labels of the output and input clusters. Both accuracy measures range in $[0, 1]$, where larger values indicate better accuracy. Due to the stochastic nature of the algorithm each test was repeated between 10-100 times depending on the size and dimensionality of the data, and average accuracies were recorded.

For our first set of experiments with synthetic data we generated low-dimensional data sets in order to avoid as much as possible the overlap of cluster projections. This in turn enabled us to examine the general performance of the algorithm without the random walk (the clustering process starts with clusters in 2D subspaces). These data sets consisted of 5-50 dimensions, where for each selected dimensionality three types of data sets were created. One with a small number of clusters (4-6 clusters), one with a small number of clusters to which noise points were added representing approximately 30% of the data. And a third type of data set containing a larger number of clusters (8-11). We also ensured that the dimensionality of the subspaces in which clusters were embedded ranged across the dimensionality of the data. From a pool of dozens of synthetic data sets on which the algorithm was applied, a representative sample of seven of each of the three different types (21 in total) of data sets was selected for illustrative purposes. The performance (accuracy) of the algorithm applied on these data sets is summarized in table 3. The table shows that the algorithm is able to maintain high levels of accuracy in cluster point detection (denoted by ‘pts’ in the table) and good overall performance in the detection of the subspaces which the clusters were embedded (denoted by ‘dim’ in the table). However, the table shows that as the dimension of the data sets increases, the accuracy in subspace detection deteriorates. We attribute this behavior to the overlap of cluster projections phenomena, i.e., the possibility that the projection of several clusters embedded in high dimensional spaces into lower dimensional spaces appear as single clusters when the algorithm commences the search, which “confuses” it in the determination of which features are relevant to

data dim	small		small+noise		large	
	pts	dim	pts	dim	pts	dim
5	0.96	0.99	0.89	0.97	0.92	0.94
8	0.99	1.0	0.94	0.99	0.98	0.99
10	0.99	0.98	0.97	0.98	0.93	0.94
15	0.97	0.93	0.96	0.93	0.97	0.91
20	0.99	0.79	0.90	0.90	0.96	0.95
30	0.94	0.96	0.82	0.95	0.96	0.91
50	0.97	0.65	0.93	0.92	0.90	0.67

Table 3: SLCLUS’s performance, without random walk when applied on low-dimensional data.

data dim	-RW		+RW	
	pts	dim	pts	dim
60	0.66	0.50	1.00	0.94
75	0.66	0.56	1.00	0.86
80	1.00	0.66	1.00	0.99
90	0.79	0.41	1.00	0.93
100	0.85	0.59	1.00	0.85

Table 4: SLCLUS’s performance, with and without the random walk, applied on high-dimensional data.

each cluster. The next two sets of experiments demonstrate that the addition of the random walk as a mechanism of selecting a “better” initial set of dimensions which are used to start the clustering process, is able to rectify this problem.

For the second set of experiments with synthetic data we generated higher dimensional data sets, and conforming with the conclusions made in section 3.3, embedded the line clusters in subspaces whose dimensionality ranged in the upper third range of subspace dimensionalities. For example, for a data set whose dimensionality is 60 we embedded line clusters in subspaces whose range of dimensionalities was $[40, 60]$. On each of the generated data sets we applied SLCLUS with and without the random walk, in order to evaluate the random walk’s utility. A sample of five data sets containing between 3-6 clusters each and whose data dimensionality ranged in 60-100 were selected for illustrative purposes. The performance of SLCLUS applied on these data sets is summarized in table 4. The table shows that when the clusters are embedded in higher dimensional subspaces employing the random walk (+RW in table) provides a significant gain in performance (specially in subspace detection) over initializing the clustering process with 2D line clusters (-RW in table). The addition of the random walk in a sense provides a mechanism to minimize and in some cases completely avoid the effects of the overlap of cluster projections.

In our third and last set of experiments with synthetic data we generated lower and higher

data dim	+RW	
	pts	dim
40	1.00	0.85
55	0.99	0.80
70	0.95	0.72
85	0.99	0.75
100	0.99	0.79
120	1.00	0.85

Table 5: SLCLUS’s performance applied on data sets with uniformly distributed subspace dimensionalities.

dimensional data sets, but this time embedded the line clusters in subspaces whose dimensionality ranged uniformly across the data dimensionality, resulting in data sets whose clusters were embedded in lower, moderately higher, and high dimensional subspaces of the data. The algorithm was applied on these data sets in its complete form, i.e., with the random walk. The aim was to evaluate overall performance when no prior knowledge about subspace dimensionalities is available. The performance of the algorithm applied on a representative sample of these type of data sets is summarized in table 5. The table shows that the algorithm is able to achieve high accuracy in cluster point detection, and moderately high accuracy (at an acceptable level) in subspace detection. Combined with the results of the second set of experiments we validate our conclusion from section 3.3 that the random walk is more effective when the clusters are embedded in higher-dimensional subspaces.

In summary, we have demonstrated that in a controlled environment the algorithm is able to perform very well in many cases. It was able to achieve very high accuracy in cluster point detection for all cases, and relatively high accuracy in subspace detection for most cases. In the worst case the algorithm miss-detected on average only 20% of subspace dimensions. We also demonstrated that the addition of the random walk as a method of selecting an initial set of features to start the clustering process significantly enhances algorithm performance for high dimensional data sets. It also evident that in this setting it is not possible to completely avoid the effects of the overlap of cluster projections, which seem to effect only the detection of cluster subspace dimensions.

Although not designed to target the same clusters as our line clustering algorithm, we have also implemented the FLOC algorithm [6] (a more efficient implementation of the biclustering algorithm [5]) as a representative of the pattern clustering/biclustering family of algorithms. However, since it is not designed to identify line clusters we generated synthetic data that follows the bichuster or shift pattern cluster model, which as stated earlier is a special case of the line cluster model (lines with slope

one). These data sets were of varying dimensionality (low and high dimensional data) and included clusters that were embedded in subspaces whose dimensionality ranged across the dimensionality of the data. We then applied both algorithms on these data sets and compared their performance. Even though the FLOC algorithm takes as input the number of clusters assumed to exist in the data, an advantage when comparing algorithms in a supervised or controlled environment, the line clustering algorithm was able to outperform the FLOC algorithm by an average margin of 0.2 in the accuracy of cluster point detection and an average margin of 0.5 in accuracy of subspace detection. We note however that the FLOC algorithm was able to complete the experiments faster than the line clustering algorithm, a result consistent with the computational complexity of the algorithms.

We also note that a version of SLCLUS that employed a forward-stepwise feature selection approach was implemented and evaluated. We believed that the forward-stepwise approach would result in better accuracy, but discovered that on average not many backward elimination (feature removal) steps were executed, and better accuracy was not achieved. Due to the overhead associated with the backward steps we preferred use the simpler forward selection approach.

7.2 Experiments with Real Data

We conducted extensive tests on three data sets. Two data sets that have become standard benchmarking data sets for clustering gene expression data—the *yeast Saccharomyces Cerevisiae* cell cycle expression data [21] obtained from <http://arep.med.harvard.edu/biclustering/>, and the *Colon Cancer* data [22] obtained from <http://microarray.princeton.edu/oncology/>. The third—*Jester* obtained from <http://ieor.berkeley.edu/~goldberg/jester-data/>, is a data set used in an online joke collaborative filtering/recommender system [23].

7.2.1 Yeast Data

The yeast data contains 2884 genes (objects) and 17 time points/conditions (dimensions), and is a very attractive data set for evaluating clustering algorithms because many of the genes contained in it are biologically characterized and have already been assigned to different phases of the cell cycle. Applied on this data set SLCLUS discovered 62 line clusters. We compared these clusters with the 100 biclusters reported by the biclustering algorithm [5], which were obtained from <http://arep.med.harvard.edu/biclustering/>. The clusters' size detected by SLCLUS ranged in [15, 255] and on average much smaller than the clusters reported by the biclustering algorithm. As mentioned

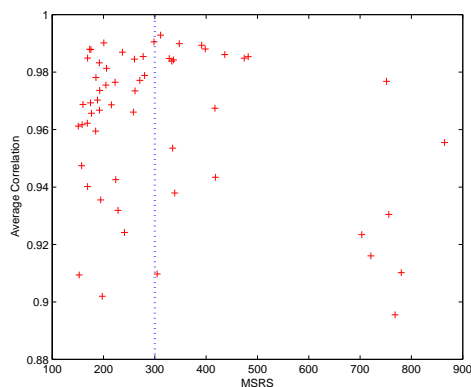


Figure 9: mean squared residue score (MSRS) versus average correlation of yeast clusters detected by SLCLUS.

already, from a biological standpoint this makes sense, as the whole yeast genome contains roughly only 6000 genes, and typical functional categories of the yeast genome contain dozens rather than hundreds of genes that were included in some of the biclusters. The dimensionality of the clusters detected by our algorithm ranged in $[3, 16]$ and was on average smaller than the dimensionality of the biclusters. We also compared the *mean squared residue score* (MSRS) which is part of the *shift pattern* cluster model and used by the biclustering algorithm as a criteria to identify shift pattern clusters. The authors of the biclustering algorithm used $MSRS=300$ as a threshold to qualify “worthy” biclusters. SLCLUS detected on average clusters with a slightly larger MSRS. However, this is reasonable since our algorithm was not restricted to searching only for shift pattern clusters for which this score was designed. Nonetheless, our algorithm was successful in finding clusters that induce large correlations. We used *average correlation*, defined to be the average of the absolute value of the correlation coefficient between each pair of features belonging to a cluster, to quantify the degree of correlation induced by a cluster. After the removal of 3 outlier clusters (clusters with average correlation less than 0.8) the mean average correlation was found to be 0.96. Fig. 9 is a plot of MSRS versus average correlation of the clusters detected by our algorithm. The figure shows that there are gene clusters in the data that induce large correlations and may be functionally related, yet do not follow the shift pattern cluster model (have $MSRS > 300$), supporting one of the main motivations for this work. We note that some of the clusters found by our algorithm did follow the shift pattern cluster model.

We also evaluated the biological significance of the clusters SLCLUS produced by means of

Genes in Cluster	MIPS Functional Category	Genes in Category	Clustered Genes	P-value
211	ribosome biogenesis	215	27	1.698e-09
	protein synthesis	359	35	5.649e-09
	cytoplasm	554	37	2.696e-05
17	cytoplasm	554	15	1.565e-14
	protein synthesis	359	13	1.178e-13
	ribosome biogenesis	215	11	6.229e-13
	subcellular localisation	2256	16	8.658e-07
193	amino acid biosynthesis	118	13	5.462e-05
49	amino acid metabolism	204	6	6.740e-06
16	cell cycle and DNA processing	628	9	5.772e-06

Table 6: MIPS gene function enrichment.

function enrichment [21]—the degree to which the clusters grouped genes of common function. This was done by computing for each cluster P-values (using the hypergeometric distribution) of observing a certain number of genes within a cluster from a particular MIPS (Munich Information Center for Protein Sequences, <http://mips.gsf.de/>) functional category. Table 6 shows five of them that had smaller P-values, indicating that SLCLUS is able to detect statistically significant clusters.

7.2.2 Cancer Data

The cancer data contains 2000 genes (objects) and 62 tissue samples (dimensions), of which 40 are colon tumor and 22 normal colon samples. The goal in this experiment was to identify gene clusters that can differentiate the cancerous tissues from the normal ones. These clusters may later afford researchers the ability design classifiers for diagnostic purposes. Applied on this data set SLCLUS detected 81 line clusters. The size of the clusters was generally small, the largest cluster contained 21 genes. The dimensionality of the subspaces in which the clusters were embedded ranged in [4, 16]. The average MSRS of the clusters was 15243, indicating that most of the clusters did not follow the shift pattern cluster model. However, their mean average correlation (after removal of two outlier clusters) on the other hand was around 0.83. Again indicating that related groups of genes may exist in the data, yet are overlooked by most clustering methods.

We also found seven gene clusters that were present in either only the normal tissues or only the cancerous tissues. One cluster contained only normal tissues and the remaining six only cancerous tissues. They contained a small number of genes 16-18, and were embedded in subspaces of dimensionality ranging in [4, 9], i.e., contained between 4-9 tissues. The average correlation of these

clusters was around 0.88, higher than the rest of the clusters, providing evidence that the genes within these clusters may be functionally related and can be used for discriminatory purposes. Most of the remaining clusters contained a mixture of tissues none with an overwhelmingly majority of normal or cancerous tissues.

7.2.3 Jester Data

The Jester data set contains a million continuous ratings in the range $[-10.00, 10.00]$ of 100 jokes (dimensions) from 73421 users (objects). Not all users rated all jokes, thus the data contains missing values. We extracted a subset of 6916 users that rated all jokes. Applied on this data set SLCLUS identified 252 line clusters with the following statistics. The average size (number of users) of the clusters was 28 and the average dimensionality (number of jokes) of the clusters was 10. Again we examined the MSRS of the clusters to get an idea of what types of patterns were identified. Surprisingly we discovered that most of the clusters followed the shift pattern (slope one clusters in the collaborative filtering literature) with a very low average MSRS of approximately 5. This finding demonstrates that while being able to identify a wider and more general spectrum of patterns SLCLUS does not overlook the more common patterns, which as stated are a special case of the line cluster model, when those are present or more evident in the data. After the removal of 29 outlier clusters the average correlation was found to be approximately 0.7, and not as high as in the preceding two experiments.

7.2.4 A Note on Cluster Validation

Until a consensus is reached pertaining to the question of which patterns carry biologically relevant information, and appropriate cluster validation techniques are devised (when ground truth is not available), the clustering of gene expression data may be considered a subjective process which depends on the cluster models assumed in advance, and which yields different views or interpretations of the data depending on the clustering technique fitted to the assumed cluster models. In this respect the line clustering paradigm makes less assumptions as it generalizes several more specific cluster models. The derivation and evaluation of such validity measures is beyond the scope of this paper. Hence, the experimental results presented in this section are by no means an attempt to report discoveries of biologically relevant and previously unknown results. The aim is to merely point out and demonstrate a method which is able to identify pattern clusters that could be of biological

relevance, some of which are overlooked by other methods.

8 Conclusions

The problem of searching for clusters that induce large correlations among some subset of features or the so called “pattern clusters” as a special case, was cast into the problem of searching for groups of points the fit lines in subspaces of the data. The advantage of this paradigm of clustering as opposed to existing methods is that it allows the clustering of different types of patterns or correlations simultaneously. It also allows the clustering of patterns and correlations that are overlooked by existing methods. A formal stochastic line cluster model was presented, and based on it an algorithm that searches for line clusters embedded in subspaces of the data was proposed. The potential of the algorithm was demonstrated by applying it on real and synthetic data sets, however its flexibility comes at a price. Its main bottleneck is associated with the computation of the leading eigenvector of a cluster’s covariance matrix, which is used to estimate model parameters. Some optimizations were proposed. In future work we plan to investigate these and other optimizations. In addition we also plan to investigate cluster validation methods appropriate for correlation and pattern clustering.

References

- [1] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the international conference on Management of data*, pages 94–105, 1998.
- [2] Charu C. Aggarwal, Joel L. Wolf, Philip S. Yu, Cecilia Procopiuc, and Jong Soo Park. Fast algorithms for projected clustering. In *Proceedings of the international conference on Management of data*, pages 61–72, 1999.
- [3] Harsha S. Nagesh and Alok Choudhary Sanjay Goil. Mafia: Efficient and scalable subspace clustering for very large data sets. Technical Report 9906-010, Northwestern University, 1999.
- [4] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: A review. *Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining*, 6(1):90–105, 2004.

- [5] Y. Cheng and G. Church. Biclustering of expression data. In *International Conference on Intelligent Systems for Molecular Biology*, pages 93–103, 2000.
- [6] Jiong Yang, Wei Wang, and Haixun Wang Philip Yu. δ -clusters: Capturing subspace correlation in a large data set. In *Proceedings of the 18th International Conference on Data Engineering*, pages 517–528, 2002.
- [7] Haixun Wang, Wei Wang, Jiong Yang, and Philip S. Yu. Clustering by pattern similarity in large data sets. In *Proceedings of the International Conference on Management of Data*, pages 394–405, 2002.
- [8] Daniel Lemire and Anna Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *Proceedings of SIAM Data Mining*, 2005.
- [9] S. Erdal, O. Ozturk, D. Armbruster, H. Ferhatosmanoglu, and W.C. Ray. A time series analysis of microarray data. In *Proceedings of the 4th IEEE Symposium on Bioinformatics and Bioengineering*, pages 366–375, 2004.
- [10] Christian Böhm, Karin Kailing, Peer Kröger, and Arthur Zimek. Computing clusters of correlation connected objects. In *Proceedings of the International Conference on Management of Data*, pages 455–466, 2004.
- [11] Lizhuang Zhao and Mohammed J. Zaki. Microcluster: Efficient deterministic biclustering of microarray data. *IEEE Intelligent Systems*, 20(6):40–49, 2005.
- [12] Jesus S. Aguilar-Ruiz. Shifting and scaling patterns from gene expression data. *Bioinformatics*, 21(20):3840–3845, 2005.
- [13] Rave Harpaz and Robert M. Haralick. Exploiting the geometry of gene expression patterns for unsupervised learning. In *Proceedings of the 18th International Conference on Pattern Recognition*, volume 2, pages 670–674, 2006.
- [14] Robert Haralick and Rave Harpaz. Linear manifold clustering in high dimensional spaces by stochastic search. *Pattern Recognition*. doi:10.1016/j.patcog.2007.01.020.
- [15] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

- [16] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. Berkeley, University of California Press, 1967.
- [17] J. Illingworth and J. Kittler. A survey of the hough transform. *Computer Vision, Graphics, and Image Processing*, 44(1):87–116, 1988.
- [18] Isabelle Guyon and Andre Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.
- [19] Dimitrios Gunopulos, Heikki Mannila, and Sanjeev Saluja. Discovering all most specific sentences by randomized algorithms. In *Proceedings of the 6th International Conference on Database Theory*, pages 215–229, 1997.
- [20] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.
- [21] Saeed Tavazoie, Jason D. Hughes, Michael J. Campbell, Raymond J. Cho, and George M. Church. Systematic determination of genetic network architecture. *Nature Genetics*, 22(3):281–285, 1999.
- [22] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. In *Proceedings of the National Academy of Sciences*, volume 96, pages 6745–6750, 1999.
- [23] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.