SCHISM: A New Approach to Interesting Subspace Mining

Karlton Sequeira* and Mohammed Zaki

Department of Computer Science,
Rensselaer Polytechnic Institute,
110 8th St, Troy, NY 12180, USA
E-mail: sequek@cs.rpi.edu E-mail: zaki@cs.rpi.edu
*Corresponding author

Abstract: High-dimensional data pose challenges to traditional clustering algorithms due to their inherent sparsity and data tend to cluster in different and possibly overlapping subspaces of the entire feature space. Finding such subspaces is called subspace mining. We present SCHISM, a new algorithm for mining interesting subspaces, using the notions of support and Chernoff-Hoeffding bounds. We use a vertical representation of the dataset, and use a depth-first search with backtracking to find maximal interesting subspaces. We test our algorithm on a number of high-dimensional synthetic and real datasets to test its effectiveness.

Keywords: subspace mining, clustering, interestingness measures, chernoff-hoeffding bounds, maximal subspaces, data mining

Reference to this paper should be made as follows: K. Sequeira and M. Zaki, (xxxx) 'SCHISM: A New Approach to Interesting Subspace Mining', *Int. J. Business Intelligence and Data Mining*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Karlton Sequeira received the B.E. degree in electronic engineering in 1999 from the University of Mumbai, India. He received the M.S. degree in computer science in 2002 from RPI, USA and is currently a Ph.D. candidate in the Computer Science Department at RPI. His research interests include algorithms for high-dimensional and heterogeneous datasets, computational biology and manifold learning.

Mohammed J. Zaki is an associate professor of Computer Science at RPI. He received his Ph.D. degree in computer science from the University of Rochester in 1998. His research interests focus on developing novel data mining techniques for bioinformatics, performance mining, web mining, and so on. He has published over 100 papers on data mining, co-edited 11 books (including "data mining in bioinformatics", Springer-Verlag, 2004), served as guest-editor for several journals, has served on the program committees of major international conferences, and has co-chaired many workshops (BIOKDD, HPDM, DMKD, etc.) in data mining. He is currently an associate editor for *IEEE Transactions on Knowledge and Data Engineering*, action editor for *Data Mining and Knowledge Discovery: An International Journal*, editor for *Int'l Journal of Data Warehousing and Mining*, and *ACM SIGMOD Digital Symposium Collection*. He received the National Science Foundation CAREER Award in 2001 and the Department of Energy Early Career Principal Investigator Award in 2002. He also received the ACM Recognition of Service Award in 2003. He is a member of IEEE, ACM,

and ISCB.

1 Introduction

Clustering is an unsupervised learning process, in which a multidimensional space is partitioned into disjoint regions, such that all points within any such region/cluster are similar to each other, but dissimilar with respect to points in other clusters. If the clustering is done using all available features, it is called a *full-dimensional* clustering. Many such algorithms like BIRCH, DBSCAN, CURE [14] have been proposed for this task. While they show acceptable performance on lower dimensional datasets, a large number of dimensions poses problems [15]. One of the main reasons, is that data is generally very sparse in high dimensional datasets. In addition, most of the full dimensional algorithms use distance metrics, which treat every dimension with equal importance. For high dimensional spaces, it has been argued that under certain reasonable assumptions on the data distribution, the ratio of the distances of the nearest and farthest neighbors to a given target is almost 1 for a variety of distance functions and data distributions [7]. In such a scenario, many full dimensional clustering algorithms have little meaning, as the pairwise distances between the points in distinct clusters need not provide an acceptable contrast.

One of the solutions to the problem of clustering high-dimensional datasets proposed is designing new distance metrics [1]. Another is reducing the dimensionality [12] and then running the *full-dimensional* algorithm on the lower dimensional dataset. Dimension reduction techniques are of two types: *feature selection*, in which one aims to find linear or non-linear combinations of the original set of dimensions, and *variable selection*, in which we select a subset of the original set of dimensions.

Prominent among the feature selection methods are the Karhunen-Loeve transformation (KLT) or singular value decomposition (SVD), which project the dataset from the original d to a k dimensional space, where $k \ll d$, and each new dimension is a linear combination of the original dimensions; after this clustering is done using only these k dimensions. However, it may not always be possible to reduce the dimensionality of the space. Consider the case, where there exist subsets of the embedded subspaces, which are unconstrained in each dimension. For such datasets, the ratio of the largest eigenvalue to the smallest one may not be large enough to discard the corresponding eigenvalues/dimensions and hence dimensionality reduction may not be feasible. Also, such a strategy may be inappropriate since clusters in the transformed feature space may be hard to interpret. Earlier literature [4, 21] cites examples in which KLT does not reduce the dimensionality without trading off considerable information, as the dataset contains subsets of points which lie in different and sometimes overlapping lower dimensional subspaces. In variable selection, some of the dimensions are selected heuristically without transformation [8]. This removes the problem of interpretability, but still only a fixed subspace is used for clustering.

There also exists the method of multidimensional scaling [19], which uses a similarity matrix to infer the underlying dimensionality. Here too, the transformed features space require expert interpretation. These challenges have caused the focus of much recent work in clustering to shift towards finding the interesting subspaces within a high-dimensional space[2, 3, 4, 9, 10, 17, 20, 21]. Other challenges encountered in subspace mining are that subspaces may share dimensions as well as objects, i.e., they may overlap. The subspace

mining problem has wide applications, especially with datasets having ordinal/nominal values, e.g., datasets found in bioinformatics, intrusion detection, etc.

In this paper, we tackle the problem of finding statistically 'interesting' subspaces in a high dimensional dataset using an algorithm called SCHISM (**S**upport and **C**hernoff-**H**oeffding bound-based **I**nteresting **S**ubspace **M**iner). We use the Chernoff-Hoeffding bound to prune the search for interesting subspaces, as a nonlinear function of the number of dimensions in which the subspace is constrained. We use a vertical representation of the dataset and capitalize on various advances made in itemset mining. We use a depth-first search with backtracking to find maximal interesting subspaces. We finally test our algorithm on a wide array of high-dimensional datasets.

2 Related Work

Let $A = \{A_1, A_2, \ldots, A_d\}$ be the set of dimensions. Each dimension A_i has a totally ordered and bounded domain. Then, $S = A_1 \times A_2 \times \ldots \times A_d$ is the high-dimensional space. The input DB, is a set of n points, $DB = \{p_i | i \in [1, n], p_i \in S\}\}$. We partition S into non-overlapping rectangular units, obtained by partitioning each dimension into ξ intervals of equal width.

Definition 1: *A subspace is an axis-aligned hyper-rectangle,* $[l_1, h_1] \times [l_2, h_2] \times \ldots \times [l_d, h_d]$*, where* $l_i = (aD_i)/\xi$*, and* $h_i = (bD_i)/\xi,\ a, b$ *are positive integers, and* $a < b \leq \xi$

If $h_i - l_i = D_i$, the subspace is unconstrained in dimension i whose range is given as D_i. A m-subspace is a subspace constrained in m dimensions, denoted as S_m.

Subspace clustering methods may be classified into two main categories: density-based and projected clustering.

Density-based Clustering: Agrawal et al. [4], proposed CLIQUE, which discretizes the domain of each of the d dimensions into a user-specified number, ξ, of equal-width intervals. This transformation from a possibly continuous to a discrete space allows for an analogy between frequent itemset mining [5] and subspace clustering. The corresponding intervals are then analogous to items and the points are analogous to a transaction. A k-subspace is analogous to a k-itemset. They use *support* (the fraction of points that lie in a subspace) to denote the density of a subspace; only those subspaces above a minimum density threshold s, are mined. Based on this definition of 'dense', it is trivial to see that, if any $(p+1)$-subspace is dense, then all p-subspaces enclosing it, are also dense. They first find all dense 1-subspaces and using a breadth-first search, iteratively find all dense subspaces. In the ith iteration, the dataset is scanned and the frequencies of those i-subspaces, for which there exist at least two dense $(i-1)$-subspaces (dense in the first $i-2$ dimensions), are updated. Using these frequencies, the dense i-subspaces may be determined. Using such a bottom-up Apriori-like [5] approach, they mine higher-dimensional 'dense', hyper-rectangular subspaces. To prune their search at a faster rate, they use the minimum-description length (MDL) principle as a heuristic, thereby making it an approximate search. They then merge 'dense' subspaces sharing faces, and use covering algorithms to mine the minimal descriptions of the subspaces.

Instead of support, Cheng [10], proposed using entropy as a measure of subspace interestingness. Subspaces satisfying an entropy threshold are mined. However, they do not suggest a principled or intuitive way to set the entropy threshold. Nagesh et al. in MAFIA[20], partition each dimension into variable width intervals, based on the distri-

bution of points. They first construct a very fine-grained histogram for each attribute. Non-overlapping windows of fixed bin size x, covering the histogram are then examined. Each window is assigned a value, equal to the number of elements in the most populous bin in it. Going from left to right, windows are merged if they are within β (a user specified parameter, typically 20%) of each other. The resulting interval is considered 'dense' if the number of points in it exceeds the threshold $(\alpha an)/D_i$, where n is the number of points in the dataset and α is a user-specified parameter, called the *cluster dominance factor*. Here, $(an)/D_i$ corresponds to the number of points expected to lie inside the interval having a windows in the i-th dimension, which has range D_i. Using adaptive width intervals minimizes rigidity of clusters obtained by CLIQUE. There is however, no strong intuition behind the selection of parameters $\alpha,\ \beta,\ x$.

Kailing et al. [18] suggest using a sample of the points in the dataset. They generate dense subspaces enclosing each point of the sample if it is a *core object*, i.e., if it has more than $MinPts$, a user-specified threshold, points within a threshold radius ϵ. The subspaces are then assigned a quality rating, which takes into account the number of dimensions in which the subspace is constrained and this rating is used to prune lower quality subspaces. By providing a rating, it is only possible for the user to determine the relative interestingness of a subspace w.r.t. another subspace. It is not easy for the user to know the absolute interestingness of the subspace.

Projected Clustering: Aggarwal [2, 3] uses projective clustering to partition the dataset into clusters occurring in possibly different subsets of dimensions in a high dimensional dataset. PROCLUS [2] seeks to find axis-aligned subspaces by a three stage approach. First, a set of k cluster medoids are iteratively selected, without replacement, from the dataset, using a greedy approach. Second, a CLARANS-related[14], hill-climbing technique is used to cluster the data. In each iteration, the dataset is covered by spheres L_i, centered at medoid m_i. For each m_i, a set of dimensions D_i, is determined for which, the points in L_i show little variation. The dataset is then partitioned using the nearest neighbor algorithm, with Manhattan distance computed from medoid m_i only over D_i. Finally, after the average intra-cluster Manhattan distance stabilizes, a cluster refinement phase is used. ORCLUS [3], finds arbitrarily oriented clusters by using a variant of the iterative agglomerative hierarchical clustering. Initially $k_e > k$ full dimensional points are randomly selected as subspace centers. In each iteration using the nearest neighbor algorithm, the distance of each point in the dataset to each d_e-dimensional subspace is determined and the dataset is partitioned. The eigenvectors corresponding to the smallest βd_e eigenvalues for the covariance matrix for the points in each resulting partition are retained and using merging, the number of clusters decreases by $\alpha < 1$. Both the algorithms require the number of clusters and the expected number of dimensions for each cluster to be input.

In DOC [21], Procopiuc et al. devise a Monte Carlo algorithm for finding projective clusters. They propose a mathematical formulation for the notion of optimal projective cluster based on the density of the points in the subspaces. In an intuition-wise ORCLUS-like algorithm [9], Chakrabarti et al. search for local correlations in the data and perform dimensionality reduction on the locally correlated clusters of data individually.

3 Interestingness Measure

Let X_p be the random variable (RV) denoting the number of points in a given subspace S_p. If the probability of finding no less than n_p points in S_p, is bounded by a reasonably low, user-specified threshold probability τ, S_p is considered to be interesting [a], i.e., $Pr(X_p \geq n_p) \leq \tau$ implies that S_p is an interesting subspace. Accordingly, we have

Definition 2: *A subspace is* **interesting** *if the number of points it contains is statistically significantly higher than that expected under the assumption that all dimensions are independent and uniformly distributed.*

It is obvious that a dataset that is scattered uniformly and independently, spanning the entire S, is of least interest from a clustering view-point, as the entropy is maximized. If a subspace deviates significantly from the uniform distribution, then it is potentially interesting [b]. If n_p points are found in S_p, CLIQUE considers S_p to be 'dense' if $n_p/n \geq s$, where s is the user-specified support threshold. MAFIA considers the subspace 'dense' if $n_p/n \geq \alpha E[X_p]$. SUBCLU denotes a p-hypersphere as dense, in that it contains a 'core' object, if its projected ϵ-neighborhood encloses the projections of more than $MinPts$ points.

In general, all density-based subspace finding algorithms, use an interestingness threshold function $thresh : Z^+ \rightarrow \Re$ where Z^+ is the set of positive integers, and denotes the number of constrained dimensions in a *candidate* subspace. The value of $thresh(p)$ corresponds to the interestingness threshold that must be exceeded for the candidate subspace to be called 'dense'. For example, support based pruning in CLIQUE, $thresh_{CLIQUE}(p) = s, \forall p \in [1, d]$, i.e., no matter what the number of constrained dimensions of a subspace, the pruning threshold is a constant. This is counterintuitive, in that as p increases, the volume of the p-subspace decreases exponentially, and hence the expected number of points in it should also decrease. This creates a bias towards p-subspaces, where $p \ll d$.

The $thresh$ function may be either constant (as in CLIQUE) or monotonically increasing or monotonically decreasing.

Lemma 1 (Effect of $thresh$ on monotonicity):
If any given subspace $S_{p+1} \subset S$ is interesting, then every p-subspace S_p, which encloses S_{p+1} and is unconstrained in one of the $(p+1)$ constrained dimensions of S_{p+1}, is always interesting if $thresh(p+1) \geq thresh(p), 1 \leq p \leq d-1$, for interestingness threshold function $thresh$.

Proof: If S_{p+1} is interesting, $\frac{n_{p+1}}{n} \geq thresh(p+1)$. But, $n_p \geq n_{p+1}$ because $S_{p+1} \subset S_p$. Thus, $\frac{n_p}{n} \geq \frac{n_{p+1}}{n} \geq thresh(p+1)$. If, $thresh(p+1) \geq thresh(p)$, then $\frac{n_p}{n} \geq thresh(p)$ and we are guaranteed to find all interesting subspaces.■

Lemma 2: *For p=1…d, let $thresh$ be monotonically increasing and let $thresh_2(p) = thresh(r)$, where r is the number of dimensions constrained in S_r, a maximally interesting subspace under $thresh$. If $S_r \subseteq S_p$, then S_p is also interesting under $thresh_2$.*

Proof: Since S_r is interesting, then $\frac{n_r}{n} \geq thresh(r)$. Also, as $S_r \subseteq S_p, 1 \leq p \leq r-1, n_p \geq n_r$. Hence, $\frac{n_p}{n} \geq \frac{n_r}{n} \geq thresh(r) = thresh_2(p)$. Thus, S_p is also interesting under $thresh_2$.■

[a] Typically τ is set to $O(\frac{1}{n}) \ll 0.05$, which is statistically significant
[b] See Eq. 3 and following comment

A consequence of the above lemmas is that it in order to find all the maximal interesting subspaces found by $thresh_2$, one must set $thresh(1)$ to a very low value so that it converges to $thresh_2$. This causes generation of a number of candidate subspaces which do not yield maximal interesting subspaces but add to computation time. Hence, it makes little sense to have a monotonically increasing threshold function. The $thresh$ function must be either constant or monotonically decreasing. However, it has been observed, that in order to find small interesting subspaces having few points inside it, the constant support threshold function has to be set very low,which makes subspace mining very slow. Hence, we propose a non-linear monotonically decreasing threshold function, which does not guarantee mining all interesting subspaces,[c] but does mine in a more reasonable time. The intuition behind why this might work, is that as p increases, the subspace becomes constrained in more and more dimensions, making its volume smaller and smaller. Hence the threshold too, must decrease as p increases, for the enclosed (S_{p+1}) and enclosing (S_p) subspaces to have comparable interestingness.

3.1 Chernoff-Hoeffding bound

We use the Chernoff-Hoeffding bound [11, 16] to bound the tail of the distribution of X_p and measure the level of interestingness. If $Y_i,\ i = 1 \ldots n$, are independently distributed RV, with $0 \leq Y_i \leq 1$ and $Var[Y_i] < \infty$, then for $Y = \sum_{i=1}^{n} Y_i, t > 0$,

$$Pr[Y \geq E[Y] + nt] \leq e^{-2nt^2} \tag{1}$$

where $E[Y] = \sum_{i=1}^{n} E[Y_i]$ by linearity of expectation.

Given a p-subspace S_p, let Y_i correspond to the RV that the ith point in DB, when projected onto the set of p constrained dimensions of S_p, lies within S_p. Then $Y = X_p$. Using Eq. (1) and for some real $t_p > 0$,

$$S_p \text{ is interesting if } \ Pr[X_p \geq n_p] \leq e^{-2nt_p^2} \leq \tau \tag{2}$$

where $E[X_p] + nt_p = n_p$, which implies that $t_p = \frac{n_p}{n} - \frac{E[X_p]}{n}$. Substituting t_p in the right hand term of (2), we have $e^{-2n\left(\frac{n_p}{n} - \frac{E[X_p]}{n}\right)^2} \leq \tau$ which on simplification gives,

$$\frac{n_p}{n} \geq \frac{E[X_p]}{n} + \sqrt{\frac{1}{2n} ln(\frac{1}{\tau})} \tag{3}$$

Thus, for a p-subspace to be interesting, (3) must hold. (3) makes no assumption, other than independence, about the comparative distribution. Hence, it can be used to find interesting subspaces in datasets, having non-uniform distributions.

Note that the interestingness measure, $thresh(p) = \frac{E[X_p]}{n} + \sqrt{\frac{1}{2n} ln(\frac{1}{\tau})}$ is a non-linear monotonically decreasing function in the number of dimensions p, in which S_p is constrained. Also, note that $thresh$ is analogous to the support and density threshold measures used to prune search in the CLIQUE [4] and MAFIA [20] algorithms respectively. In comparison with CLIQUE, the term $\sqrt{\frac{1}{2n} ln(\frac{1}{\tau})}$ corresponds to minimum density s set by the user. The interestingness threshold probability (τ) seems intuitively easier to set than

[c] For monotonically decreasing $thresh$, the Apriori principle may not be applicable. Consider, $\frac{n_p}{n} = \frac{n_{p+1}}{n} = thresh(p+1)$. If $thresh(p+1) < thresh(p)$, then $\frac{n_p}{n} < thresh(p)$ and S_p is not interesting although S_{p+1} is.

s. The chief difference is the term $\frac{E[X_p]}{n}$, which makes pruning conscious of the volume of the subspace and hence conscious of the number of constrained dimensions of the subspace on which it is being carried out. Also, the equivalence between s and the term $\sqrt{\frac{1}{2n}ln(\frac{1}{\tau})}$ provides insight into setting of parameter s, in that it should be inversely proportional to the square root of the dataset size. In comparison with MAFIA, the term $E[X_p]$ corresponds to $\alpha E[X_p]$. *Thus, SCHISM inadvertently unifies ideas from both CLIQUE and MAFIA.* Unlike earlier proposed interestingness measures [18], this one gives the user a sense of absolute interestingness.

Note that for some $v \in Z, 1 \leq v \leq d$, we have $E[X_p] \leq 1$, and thus $\frac{E[X_p]}{n} + \sqrt{\frac{1}{2n}ln(\frac{1}{\tau})} \approx \sqrt{\frac{1}{2n}ln(\frac{1}{\tau})}$. The threshold function thus converges to a constant when the number of constrained dimensions $p \geq v$; analogous to minimum threshold s in CLIQUE. To summarize,

$$thresh_{SCHISM}(d \geq p \geq v) = \sqrt{\frac{1}{2n}ln(\frac{1}{\tau})}$$

From Lemma 1, for $p \geq v$, S_{p+1} is interesting implies that S_p is interesting, as SCHISM is similar to CLIQUE and uses support-based pruning for a large part of the subspace mining process. Note that this constant $\sqrt{\frac{1}{2n}ln\left(\frac{1}{\tau}\right)}$ varies inversely as $\sqrt{n}$ and hence $thresh_{SCHISM}$ converges to a higher threshold for smaller datasets. This is a negative result, in that for large datasets, we require a lower pruning threshold to achieve similar interestingness. This is contrary to the way, most users would set a pruning threshold like s and provides more motivation for our idea.

While $thresh_{SCHISM}$ is constant for $p \geq v$, we can gain some improvements in empirical results by changing the rate of change in $thresh_{SCHISM}(p < v)$ to increase the likelihood of monotonic search. We do so by trading off some tightness of the bound by using a penalty term. If $f(p)$ is the penalty term, such that $\forall p \in [1, d], f(p) \leq 1$, then $e^{-2nt_p^2} \leq e^{-2nf(p)t_p^2}$. Using this in 2, $Pr(X_p \geq n_p) \leq e^{-2nf(p)t_p^2} \leq \tau$. After simplification,

$$thresh_{SCHISM}(0 < p < v) = \min\left\{u, \frac{E[X_p]}{n} + \sqrt{\frac{1}{2nf(p)}ln\left(\frac{1}{\tau}\right)}\right\}$$

The term u is used to upper bound $thresh_{SCHISM}(1)$, which is empirically too large for typical values of ξ. Typical values of $f(p)$ are $\frac{p}{a}(a \geq v \implies f(p) \leq 1), \frac{1}{(c-bp^2)}$. The last penalty term provides a parabolic as opposed to exponential drop in the threshold as p increases. Typically, $u = 0.05$. In summary, we have

$$thresh_{SCHISM}(p) = \begin{cases} \min\left\{u, \frac{E[X_p]}{n} + \sqrt{\frac{1}{2nf(p)}ln\left(\frac{1}{\tau}\right)}\right\} & if\ p < v \\ \sqrt{\frac{1}{2n}ln\left(\frac{1}{\tau}\right)} & if\ p \geq v \end{cases} \quad (4)$$

If each dimension of the d-dimensional dataset is discretized into ξ equi-width intervals and $p(i, j)$ corresponds to the probability, that a point in the dataset lies in the jth interval of the ith dimension. If $S_p = [l_1, h_1] \times \ldots \times [l_d, h_d]$ then,

$$E[X_p] = \sum_{i=1}^{d} \sum_{j \in [l_i, h_i]} p(i, j)$$

3.2 Under assumption of a multivariate uniform background distribution

If we assume that each dimension in the d-dimensional space is independent and uniformly distributed and discretized into ξ levels, then the probability that a point lies in a specific interval of any dimension is $\frac{1}{\xi}$. Hence, the probability that a point lies in a specific p-subspace (assuming it is constrained to a single interval in each of the p constrained dimensions) is $(\frac{1}{\xi})^p$. Thus, the probability of finding n_p points in any subspace S_p, is distributed as per the binomial distribution with mean $E[X_p] = n(\frac{1}{\xi})^p$. Here, $v = \lceil \frac{log(n)}{log(\xi)} \rceil$

To mine more subspaces than those found by the support threshold (u) in CLIQUE [4],

$$u \geq \frac{1}{\xi^v} + \sqrt{\frac{1}{2n} ln(\frac{1}{\tau})} \tag{5}$$

Substituting typical values e.g., $\tau = 10^{-5}, n = 10^5, \xi = 10, v = 5$ in Eq. (5), we get $u \geq .0076$. It is unusual to set u to lower values for typical high-dimensional datasets, as that would make *Apriori*-based search virtually impossible. Thus, using $thresh_{SCHISM}$ helps mine for smaller subspaces than CLIQUE.

4 SCHISM Algorithm

A number of the subspace mining algorithms [4, 10, 20] use a bottom-up, breadth-first search. In contrast, SCHISM, which is based on the GenMax algorithm that mines maximal itemsets [13], uses a depth-first search with backtracking to mine the maximal interesting subspaces. The main steps in SCHISM are shown in Fig. 1; we first discretize the dataset and convert it to a vertical format. Then we mine the maximal interesting subspaces. Finally, we assign each point to its cluster, or label it as an outlier.

SCHISM (DB, s, ξ, τ):
//s is the minimum support threshold
//ξ is the number of intervals per dimension
//τ is the user-specified interestingness threshold
1. DDB = **Discretize**(DB, ξ)
2. VDB=**HorizontalToVertical**(DDB)
3. MIS = **MineSubspaces**(VDB, s, ξ, τ)
4. **AssignPoints** (DB, MIS)

Figure 1 The SCHISM Algorithm

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8
I_1	-1	-1	-1	478	-1	673	774	-1
I_2	-1	163	475	260	-1	-1	-1	786
I_3	-1	949	-1	985	72	204	806	317

Table 1 Original subspaces

DB	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8
p_1	755	689	306	482	838	657	743	980
p_2	818	166	494	302	378	439	633	805
p_3	418	159	499	260	139	921	986	780
p_4	833	173	484	236	948	17	647	781
p_5	264	960	465	985	70	209	782	309
p_6	991	972	118	986	72	209	804	341
p_7	921	963	910	976	71	220	818	317
p_8	686	965	623	993	68	202	800	287
p_9	448	146	605	205	984	423	654	983

Table 2 Original dataset: DB

	d'_1	d'_2	d'_3	d'_4	d'_5	d'_6	d'_7	d'_8
I'_1	-1	-1	-1	34	-1	56	67	-1
I'_2	-1	11	24	32	-1	-1	-1	77
I'_3	-1	19	-1	39	40	52	68	73

Table 3 Discretized subspaces

DDB	d'_1	d'_2	d'_3	d'_4	d'_5	d'_6	d'_7	d'_8
p'_1	7	16	23	34	48	56	67	79
p'_2	8	11	24	33	43	54	66	78
p'_3	4	11	24	32	41	59	69	77
p'_4	8	11	24	32	49	50	66	77
p'_5	2	19	24	39	40	52	67	73
p'_6	9	19	21	39	40	52	68	73
p'_7	9	19	29	39	40	52	68	73
p'_8	6	19	26	39	40	52	68	72
p'_9	4	11	26	32	49	54	66	79

Table 4 Discretized dataset: DDB

Discretization: In SCHISM, we first discretize all points (figure 1, line 1). Given the original dataset DB, we divide each dimension into ξ bins, and give each interval a unique id (for example, the intervals in dimension d_0 are labeled from 0 to $\xi - 1$, those for d_1 are labeled from ξ to $2\xi - 1$, etc.). Consider the example dataset DB shown in Table 2, generated by our synthetic data generator (see section 5.1), with $n = 9, \xi = 10$ and $d = 8$. The seed subspaces used to generate DB are shown in Table 1.

Here -1 implies that the subspaces are unconstrained in that dimension. Thus, p_1 is generated from subspace I_1, points p_2, p_3, p_4 are generated from I_2, points p_5, p_6, p_7, p_8 are generated from I_3 and p_9 is an outlier. Table 4 shows the discretized dataset DDB obtained from DB; the corresponding discretized subspaces are shown in Table 3.

Data Transformation: The next step in SCHISM (figure 1, line 2) is to convert the dataset into a vertical tidset format [13], which lists for each subspace (initially a single interval) S_p, the set of points that belong to it, i.e., the tidset $t(S_p)$. Using a vertical format dataset gives us a number of advantages. Firstly, better memory utilization results from having only the relevant subspaces in memory at a time, as opposed to the horizontal format in which the entire dataset is scanned. Secondly, computing support of subspaces to be merged via tidset intersections is very fast. Fig. 3 (for $p = 1$) shows the tidsets for the

initial 'interesting' intervals. For example, for interval 11, its tidset is given as $t(11) = \{2, 3, 4, 9\}$.

MineSubspaces (VDB, ξ, τ):
1. Find IS_1 and IS_2 //sort IS_1 as optimization
2. **MIS-backtrack**($\phi, IS_1, MIS, 0, \xi, \tau$)
3. return MIS

MIS-backtrack($S_p, C_p, MIS, l, \xi, \tau$)
4. $\forall S_x \in C_p$
5. $S_{p+1} = S_p \cup S_x$
6. $P_{p+1} = \{S_y \in C_p \,|S_y > S_x\}$
7. If **MergeSubspaces**($MIS, (S_{p+1} \cup P_{p+1})$) return
8. C_{p+1} = **IS-candidate**($S_{p+1}, P_{p+1}, l, \xi, \tau$)
9. If C_{p+1} is empty
10. If S_{p+1} has enclosed no subspace in MIS,
11. $MIS = MIS \cup S_{p+1}$
12. else **MIS-backtrack**($S_{p+1}, C_{p+1}, MIS, p+1, \xi, \tau$)

MergeSubspaces(MIS, Z_{p+1})
13. If $\max_{S_f \in MIS} Sim(Z_{p+1}, S_f) > \rho \times \min(f, |Z_{p+1}|)$
14. $MIS = MIS - S_f$
15. $MIS = MIS \cup (S_f \cup (Z_{p+1})$
16. return true
17. return false

IS-candidate($S_{p+1}, P_{p+1}, l, \xi, \tau$)
18. $C_{p+1} = \phi$
19. $\forall S_y \in P_{p+1}$
20. $t(S'_y) = t(S_{p+1}) \cap t(S_y)$
21. If $\frac{|t(S'_y)|}{n} \geq thresh(|S'_y|)$
22. $C_{p+1} = C_{p+1} \cup S_y$
23. return C_{p+1}

Figure 2 Mining Interesting Subspaces

Mining Interesting Subspaces: In SCHISM, interesting subspaces are mined (figure 1, line 3), using a depth-first search with backtracking, allowing us to prune a considerable portion of the search space. The pseudo-code for `MineSubspaces` is shown in Fig. 2. The method first finds all interesting subspaces in one (IS_1) and two dimensions (IS_2). Next, we call the recursive `MIS-backtrack` procedure to mine the set of maximal interesting subspaces (MIS).

`MIS-backtrack` accepts as input, a single p-subspace S_p, and a set C_p of *candidate* p-subspaces that can be used to constrain (or extend) S_p in an interval of another dimension. Each $S_x \in C_p$ results in a potential new $(p+1)$-subspace, $S_{p+1} = S_p \cup S_x$ (line 5), for which we have to calculate the new candidate set C_{p+1} (line 8). We do this by using a *possible* set P_{p+1} (line 6) of potential candidate subspaces, which are all the unprocessed

subspaces in $S_y > S_x$ in C_p. If C_{p+1} is empty (line 9), then S_{p+1} is potentially maximal; it will be added to MIS if there is no maximal subspace that it encloses (lines 10-11). If C_{p+1} is non-empty, then we recursively call `MIS-backtrack`.

The call to `IS-Candidate` (line 8) constructs the new candidate set C_{p+1} for S_{p+1} for the next level. The basic idea is to intersect tidset of S_{p+1} with every possible subspace in P_{p+1} (line 20). We keep only those subspace extensions that pass the $thresh()$ function (lines 21-22).

Typically, a depth-first search with backtracking produces a number of subspaces, which may overlap considerably, leading to redundant subspaces. To avoid this behavior, we prune the search tree by using `MergeSubspaces` (line 7). If the subspace $Z_{p+1} = S_{p+1} \cup P_{p+1}$, resulting from constraining S_{p+1} with all its remaining possible intervals in P_{p+1}, is significantly similar (typically, we set the merging threshold $\rho = 0.8$) to some known $S_f \in MIS$ (line 13), we replace S_f with a typically more constrained subspace (lines 14-15), $S_f \cup Z_{p+1}$. As $\rho < 1$, we may merge subspaces, which are constrained to adjacent intervals in a few dimensions, thus compensating for uniform width intervals in each dimension. The Sim function (line 13) used to calculate the similarity of two subspaces A and B is given as $Sim(A, B) = \sum_{i=1}^{d} JaccardSimilarity(A_i, B_i)$, where A_i, B_i are the sets of interesting intervals spanned by A, B in the i-th dimension and $JaccardSimilarity(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$. For example for point $p'_1 \in DDB$ and seed subspace I'_2 (see Table 4 and 3 respectively), $Sim(p'_1, I'_2) = 2$, as they are identically constrained in the second (11) and third (24) dimensions.

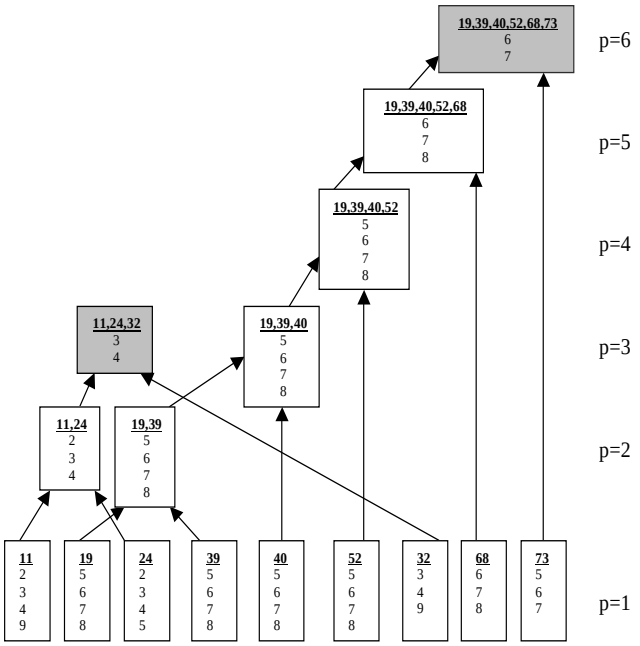

Figure 3 Lattice of running example

Example: Let's consider how SCHISM works on our example dataset DB. Let $u = 0.25, \tau = 4/n = 0.44$. Then, $IS_1 = \{11, 19, 24, 32, 39, 40, 52, 68, 73\}$. Likewise we compute IS_2. The initial call to `MIS-backtrack` is with $S_p = \emptyset$, and $C_p = IS_1$, which results in a recursive call of `MIS-backtrack` for each interval, with a new candidate set C_{p+1}. For example, the candidate-set for 11 is given by $C_1 = \{24, 32, 39, 40, 52, 68, 73\}$,

thus in the next level, we will try to extend 11 with $24, 32, \cdots, 73$, and so on, recursively. For our running example, when $S_{p+1} = 11$, $S_y = 24$, then $S'_y = \{11, 24\}$. Also $t(S_{p+1}) = \{2, 3, 4, 9\}$, and $t(S_y) = \{2, 3, 4, 5\}$, which gives $t(S'_y) = \{2, 3, 4\}$ (see Fig. 3). As $p = 1$, we use the interestingness measure for pruning the search (line 21), i.e., the second case in Equation 4. With $thresh_{SCHISM}(1) = 0.21$, $n_1/n = 3/9 = 0.33 > 0.22$; thus S'_y is interesting, and we add 24 to the candidate set. Proceeding in this manner, we get the lattice shown in Fig.3. The rectangles shaded in gray are the elements of MIS.

AssignPoints (DB, MIS):
1. for each point $p_i \in DB$
2. If $\max_j Sim(p_i, MIS_j) > ThresholdSim$
3. $p_i \rightarrow MIS_{argmax_j Sim(p_i, MIS_j)}$
4. else p_i is an outlier

Figure 4 Assign Points

Assigning Points to Clusters: Based on the intervals to which each subspace is constrained, we can estimate discrete probability distribution functions (p.d.f.) for each dimension for each mined subspace. If each dimension of the d-dimensional dataset is discretized into ξ equi-width intervals, then $B(i, j)$ corresponds to the probability, that subspace B is constrained in the jth interval of the ith dimension. Thus, $\forall i \in [1, d],\ \sum_{j=1}^{\xi} B(i, j) = 1.0$. If $b(i)$ is the number of intervals to which B is constrained in dimension i,

$$\forall i \in [1, d], B(i, j) = \begin{cases} \frac{1}{b(i)} & \text{if } B \text{ is constrained in dimension } i \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Let A correspond to the d-subspace surrounding a point p_i. Let Y and y be the RV denoting the similarity and the true similarity between A and B, respectively.

The *JaccardSimilarity* over any dimension between two such subspaces, can be expressed as the dot products of their p.d.f.s in that dimension. Hence,

$$Sim(A, B) = Y = \sum_{i=1}^{d} Y_i, \text{ where } Y_i = \sum_{j=1}^{\xi} A(i, j)B(i, j)$$

Then, $E(Y_i) = 1 \times \frac{1}{b(i)} \times \frac{b(i)}{\xi} = \frac{1}{\xi}$
Y is then, the sum of d Bernoulli RVs with mean $E[Y] = \frac{d}{\xi}$. Using Chernoff-Hoeffding bounds again, if $Pr[Y \geq y] \leq exp(-2dt^2) \leq \tau$ for reasonably small user-specified threshold τ, it implies that the similarity between A and B is unusually high and the point in A is with high probability generated from the subspace B. Now, $exp(-2dt^2) \leq \tau$ implies that $t \geq \sqrt{\frac{1}{2d} ln\left(\frac{1}{\tau}\right)}$. Substituting t in $y = E[Y] + dt = \frac{d}{\xi} + dt$, we get $ThresholdSim = \frac{d}{\xi} + \sqrt{\frac{d}{2} ln\left(\frac{1}{\tau}\right)}$.

Fig. 1, line 4 of SCHISM assigns each point to the most similar maximal interesting subspace (lines 2-3), or else labels it as an outlier (line 4). Figure 4 shows these steps. Additionally, we have to examine if the similarity is statistically significant by using $ThresholdSim$ computed above.

5 Experiments

We perform tests on a range of synthetic high-dimensional datasets using our data generator and a couple of real datasets. We evaluate SCHISM based on two metrics: i) speed: the time taken to find the interesting subspaces in the dataset, and ii) accuracy: measured in terms of entropy and coverage. For a clustering C, *entropy* is defined as $E(C) = -\sum_{C_j}(\frac{n_j}{n}\sum_i p_{ij}\log(p_{ij}))$, where $p_{ij} = \frac{n_{ij}}{n}$, C_j is the jth cluster in C, n_j is the number of points in C_j, and n_{ij} is the number of points assigned to C_j, which actually belong to subspace i. The lower the $E(C)$, the better the clustering. *Coverage* is the fraction of points in DB which are accurately labeled as not being outliers. Ideally this is 1.

5.1 Synthetic Data Sets

We generate synthetic datasets using our own data generator which employs techniques similar to those mentioned in [2, 3, 21]. We embed k multivariate Gaussian subspaces in a dataset of n points and d dimensions. We first serially generate the subspace centers and then generate points from the subspaces. Each dimension of a subspace is constrained, with probability c. As some dimensions are unconstrained with a non-zero probability, the resulting subspace is as per Definition 1. If a subspace is constrained in a dimension, the next subspace to be generated has the same dimension constrained with probability o and its mean is two standard deviations from the previous subspace's mean. This ensures that the subspaces have different volumes and they can overlap in some dimensions. If the points are normally distributed in the subspaces, the standard deviation for each dimension, for each subspace, is 20. Each point has integral coordinates in the range [0,1000]. For the constrained dimensions of subspace centers, the coordinates are chosen uniformly over this range. Let x be the fraction of the points generated as outliers and let the fraction of points generated for the i-th subspace embedded be α_i, such that $x + \sum_{i=1}^{k}\alpha_i = 1$. In order that the number of points in the subspaces differ, we use imbalance parameter $\kappa = \frac{\max_i \alpha_i}{\min_i \alpha_i}$, i.e., the ratio of the α_is of the subspace with the most points to the subspace with the least points. An outlier has each dimension chosen uniformly in [0,1000]. The points for each subspace are independently and normally distributed about its center and the coordinates for dimensions in which the center is unbounded are chosen uniformly in [0,1000]. Thus, the subspaces generated in the dataset are oriented parallel to the axes. For all experiments, unless otherwise mentioned, we set as parameters to our data generator, $k = 10,\ n = 1000,\ d = 50,\ c = 0.5,\ o = 0.5,\ \kappa = 4.0,\ x = 0.05$. Also, we set support threshold $u = 0.05,\ \tau = 1/n,\ \xi = 10$ as the parameters to SCHISM. Also, we use $f(p) = \frac{p}{d}$. The unconstrained dimensions are Gaussian distributed withe same standard deviation chosen uniformly from [1,30] and mean chosen uniformly in [0,1000].

Experiments were carried out on a Sun Sparc 650 MHz machine running on a Solaris O/S with 256 MB RAM. Since we have the seed subspaces we can easily evaluate the accuracy of SCHISM. We first mine the subspaces and partition the points in the space, so that they either belong to some interesting subspace or they are classified as outliers. Each of the following graphs, unless otherwise mentioned, shows the variation in performance, as a measure of two evaluation metrics: execution time and coverage (shown on y-axis), as a parameter of either SCHISM or the synthetic dataset is varied (shown on x-axis). *The entropy for all these experiments is below 0.004 and hence not shown*. This implies that SCHISM mines very pure clusters from our synthetic datasets. Ideally, the running time

curve should be flat or linear and the coverage curve should be flat at 1.0.

5.1.1 Effect of varying dataset parameters

Effect of dataset size and dimensionality: In Fig. 5, it is evident that as the dataset size increases, the coverage remains constant, while the running time grows linearly.

Note that in Fig. 6, as the dimensionality of the dataset increases, the coverage remains more or less constant, but the running time seems to grow exponentially initially, and then grows linearly from dimensions 200-300. In the worst case, this algorithm has exponential complexity in the number of dimensions, but in practice as shown here, the DFS algorithm coupled with the varying threshold function, significantly prunes the search space.

Effect of subspace size and dimensionality: In Fig. 7, we observe the variation in coverage and running time, as the ratio $\kappa = \frac{\max_i \alpha_i}{\min_i \alpha_i}$ increases from 2 to 12. We observe that as the ratio increases, the coverage dips marginally and the running time remains constant. The coverage decreases because the subspaces which contain a smaller number of points have, on average, as large a volume as those containing a larger number of points, leading to a lower density. A smaller fraction of their enclosing subspaces are likely to be identified as 'interesting' and hence only a small fraction of their points are detected as non-outliers, as compared to when the ratio is not so large.

In Fig. 8, we observe the variation in coverage and running time, as the probability of constraining a dimension in a subspace c, increases from 0.3 to 0.9. We observe that as c increases, the running time remains constant but larger fractions of the dataset are constrained to smaller volumes, making them more 'interesting' and hence coverage improves somewhat.

Performance on datasets with less dense subspaces: In this experiment we run SCHISM on Gaussian and hyper-rectangular datasets. We decrease the density of the Gaussian datasets by increasing the standard deviation of each constrained dimension in each subspace. For hyper-rectangular subspaces, each constrained dimension is constrained to an interval of width, chosen uniformly in [0.5w,1.5w]. Thus, the density is decreased by increasing w; the volume of the subspace and keeping the number of points assigned to it is the same.

From Fig. 9 and Fig. 10, it is clear that as density decreases, SCHISM's performance deteriorates. This is because a smaller percentage of the subspace's points tend to fall into the same interval as that of the subspace center, as the volume increases. In such a case, decreasing the number of intervals in the dimension (ξ) might help or we must search for less 'interesting' subspaces, i.e., decrease τ.

Effect of number of clusters k: Note from Fig. 5.1.1, that the running time remains constant as the number of embedded subspaces (k) increases, while coverage worsens after $k = 8$ as some clusters become very small and hence not 'interesting'.

5.1.2 Effects of varying algorithm parameters

Effect of τ: In Fig. 5.1.1, we decrease the user specified interestingness threshold τ from 10^{-12} to $10^{-.25}$, and observe its effect on the coverage and running time. Note that the coverage increases rapidly, implying that τ *is the main parameter* which determines

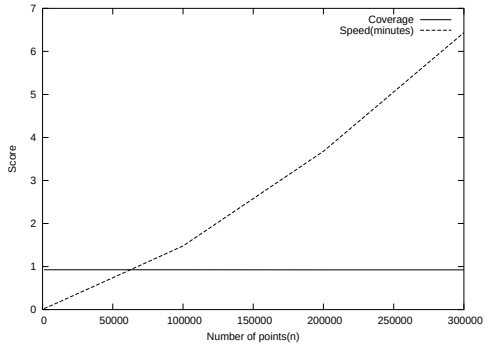

Figure 5 Dataset Size

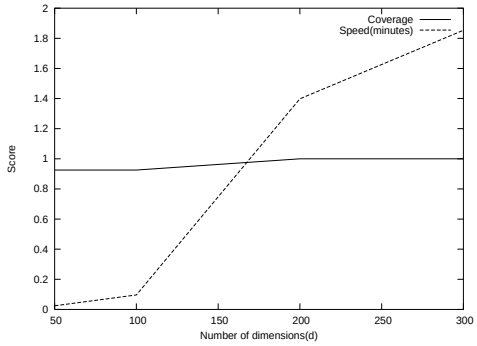

Figure 6 Data Dimensionality

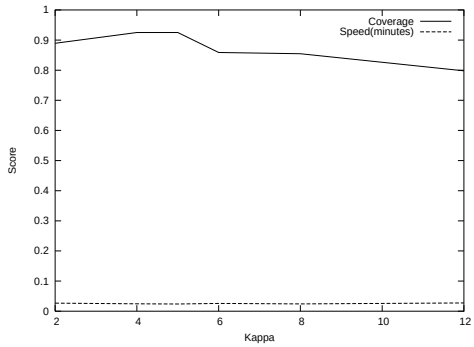

Figure 7 Imbalance factor

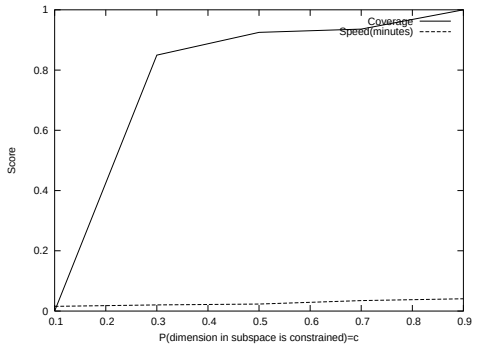

Figure 8 Subspace Dimensionality

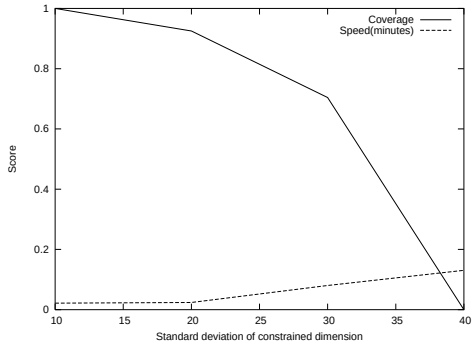

Figure 9 Gaussian Subspace Density

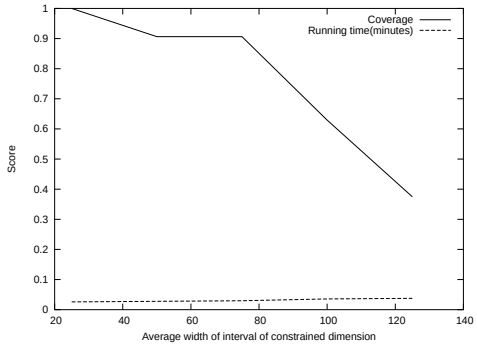

Figure 10 Hyper-rectangle Subspace Density

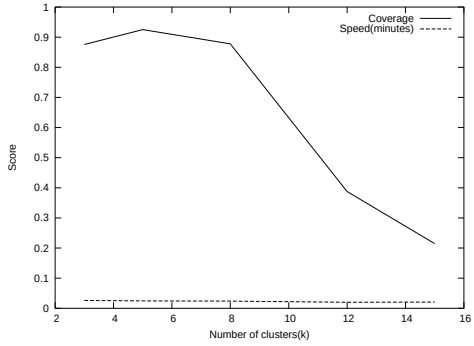

Figure 11 Num Clusters (k)

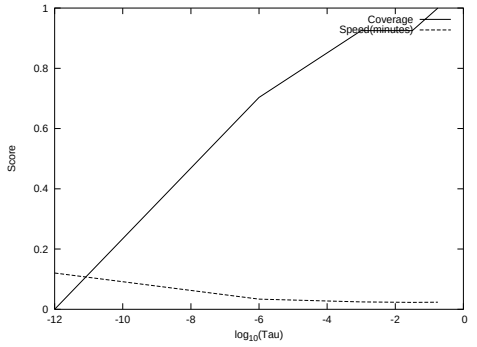

Figure 12 Interestingness Threshold

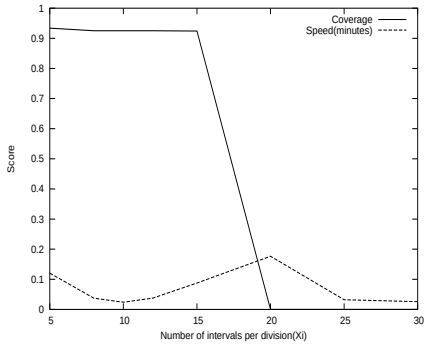

Figure 13 Effect of ξ

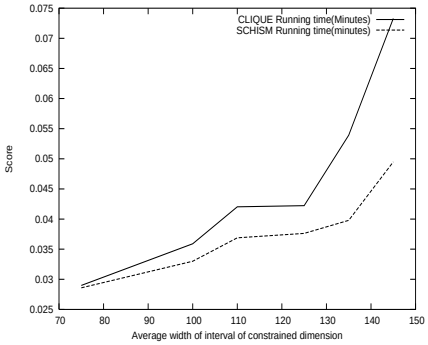

Figure 14 $thresh$: CLIQUE vs. SCHISM

how much of the search space is mined and hence running time drops rapidly too. Our experiments on the effect of u on SCHISM performance (which are not shown due to lack of space), indicate that τ has a more precise control on pruning than u.

Effect of ξ: From Fig. 5.1.1, we observe that, varying the number of intervals into which each dimension is discretized (ξ), has a small effect on SCHISM's performance for a considerable range of values for ξ. This is because the term ξ is incorporated into $thresh_{SCHISM}$. Outside this range however ($\xi > 15$), performance is severely degraded as the interval size becomes so small that very few contain enough points to be considered 'interesting'.

Effect of $thresh()$ function on performance: Here we compare the performance of the $thresh()$ function given in Eq. 4, with that of CLIQUE on the synthetic datasets. From Fig. 5.1.1, we observe that as the density of the hyper-rectangular clusters dips due to increase in the width of its constrained dimensions, the running time of CLIQUE [d] increases rapidly over that of SCHISM. Also, CLIQUE tends to split clusters into smaller ones. Its performance closely mirrors that of SCHISM for datasets having well-defined distinct clusters. However, when clusters overlap in a number of dimensions, the coverage and entropy suffers. Agrawal et al. [4], have shown that CLIQUE performs better than DBSCAN, SVD and CLARANS and hence we have used it as our benchmark.

5.2 Real Data Sets

We apply SCHISM to two well researched datasets from different domains. The PenDigits dataset [e] [6] contains 7,494 16-dimensional vectors. 30 different writers wrote approximately 250 digits, sampled randomly from the digit set [0,9] on a pressure-sensitive tablet. Each vector corresponds to the (x, y) coordinates of 8 points, spatially sampled from each of these handwritten digits. Note that the embedded subspaces, i.e., the digits 0-9, overlap considerably in the 16-dimensional space. SCHISM outputs 128 subspaces in 4

[d] Our implementation of CLIQUE involves simply replacing $thresh_{SCHISM}$ with $thresh_{CLIQUE}$ in our implementation to test the significance of our threshold function

[e] See ftp://ftp.ics.uci.edu/pub/machine-learning-databases/pendigits

Class	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
'0'	0	0	0	15	0	0	0	0	1
'1'	0	272	0	0	0	0	0	4	9
'2'	0	2	0	0	2	1	0	0	11
'3'	0	7	0	0	0	0	53	43	405
'4'	0	0	38	0	0	1	0	3	6
'5'	322	0	0	0	0	0	6	11	78
'6'	1	0	11	109	0	78	4	1	9
'7'	1	74	0	0	24	13	0	0	1
'8'	22	0	0	0	36	14	1	0	2
'9'	0	11	3	0	0	0	24	15	160

Class	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}
'0'	0	0	0	0	0	0	0	0
'1'	6	39	24	0	0	3	1	0
'2'	142	284	56	0	0	7	0	0
'3'	0	0	0	0	0	0	52	0
'4'	0	0	0	80	82	0	0	0
'5'	0	0	0	0	0	0	2	0
'6'	3	0	0	9	31	0	0	0
'7'	0	0	1	0	0	16	0	94
'8'	0	0	5	0	0	24	0	8
'9'	0	1	0	3	2	0	33	0

Table 5 Confusion Matrix for PenDigits Data Set

seconds, of which the 17 clusters with the highest entropies are shown in the confusion matrix in Table 5. It achieves a coverage of 69% and an entropy of 0.365 ($u = .01, \tau = 0.018$). CLIQUE achieves a coverage of 60.7% and an entropy of 0.49 in approximately 4 seconds too. As in DOC, ORCLUS, we provide a confusion matrix (Table 5), which is interpreted as follows: cell (i, j) of the matrix denotes the number of points having true class i, which were clustered by SCHISM into subspace j. Ideally, each row and each column have only a single non-zero entry implying E(C)=0. Note that samples of the digits $\{3, 9\}$ are both assigned by SCHISM to clusters C_7, C_8, C_9 due to their similarity in structure. The clusters not shown typically have all their samples from the same digit class.

The other dataset is the gene expression data for the yeast cell cycle [f], obtained for 2884 genes (rows) at 17 (columns) points in time. We obtained a number of clusters of which a few were highly populated and the others relatively empty. Ideally, clustering gene expression data should produce clusters of genes which are similar in function. However, almost all the genes have multiple functions and hence genes cannot be labeled by a single class. The highly populated clusters we mined using SCHISM, contained groups of genes which are known to have strong similarity in terms of function, e.g., out of 5 genes in our dataset known (see www.yeastgenome.org) to be involved in ribonuclease MRP activity, 4 (POP4,POP5,POP8,SNM1) are assigned to the same cluster, 4 genes (SEC7,AGE1,SFT1,COG6) out of 6 involved in intra-Golgi transport, are assigned to the same cluster, etc. ($\xi = 5, \tau = 0.018$). While SCHISM finds 59 such groups of genes

[f] See http://arep.med.harvard.edu/biclustering/yeast.matrix

which are clustered together in larger clusters, CLIQUE finds only 33, both doing so in approximately 8.5 seconds.

While, we attempted to compare our algorithm performance with that of SUBCLU [17], we found default parameter setting for SUBCLU to be unsatisfactory and manual setting to be extremely hard, as it took an unreasonably long time (on the order of a number of hours) to produce output for our synthetic and real datasets. The clusters produced generally split the embedded clusters into distinct clusters.

6 Conclusions

We define a new interestingness measure which provides absolute guarantees to the user about the interestingness of the subspaces reported, as per our definition of interesting. We use the interestingness measure itself to prune our search, as opposed to traditional methods[22], which determine interestingness of patterns after the search is completed, making the process faster. We use an algorithm which requires parameters which are relatively easy to set intuitively. These contributions can also be applied to the problem of finding interesting itemsets.

References and Notes

1 C. Aggarwal. Towards systematic design of distance functions for data mining applications. In *9th ACM SIGKDD Conference*, pages 9–18, 2003.

2 C. Aggarwal, C. Procopiuc, J. Wolf, P. Yu, and J. Park. A framework for finding projected clusters in high dimensional spaces. In *ACM SIGMOD Conference*, pages 61–72, 1999.

3 C. Aggarwal and P. Yu. Finding generalized projected clusters in high-dimensional spaces. In *ACM SIGMOD Conference*, pages 70–81, 2000.

4 R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *ACM SIGMOD Conference*, pages 94–105, 1998.

5 R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets ofitems in large databases. In *ACM SIGMOD Conference*, pages 207–216, 1993.

6 F. Alimoglu and E. Alpaydin. Methods of combining multiple classifiers based on different representations for pen-based handwriting recognition. In *5th Turkish Artificial Intelligence and Artificial Neural Networks Symposium*, 1996.

7 K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbors meaningful? In *ICDT Conference*, pages 217–235, 1999.

8 M. Brusco and J. Cradit. A variable selection heuristic for k-means clustering. *Psychometrika*, 66:249–270, 2001.

9 K. Chakrabarti and S. Mehrotra. Local dimensionality reduction: A new approach to indexing high dimensional spaces. In *26th International VLDB Conference*, pages 89–100, 2000.

10 C. Cheng, A. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *5th ACM SIGKDD Conference*, pages 84–93, 1999.

11 H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–509, 1952.

12 C. Ding, X. He, H. Zha, and H. Simon. Adaptive dimension reduction for clustering high-dimensional data. In *2nd IEEE ICDM Conference*, 2002.

13 K. Gouda and M. Zaki. Efficiently mining maximal frequent itemsets. In *1st IEEE ICDM Conference*, pages 163–170, 2001.

14 J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, CA, 2001.

15 A. Hinneburg and D. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *25th International VLDB Conference*, pages 506–517, 1999.

16 W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

17 K. Kailing, H. Kriegel, and P. Kroger. Density-connected subspace clustering for high-dimensional data. In *4th SIAM International Conference on Data Mining*, pages 246–257, 2004.

18 K. Kailing, H. Kriegel, P. Kroger, and S. Wanka. Ranking interesting subspaces for clustering high-dimensional data. *Lecture Notes In Artificial Intelligence*, 2838:241–252, 2003.

19 J. Kruskal and M. Wish. *Multidimensional Scaling*. Sage Publications, 1978.

20 H. Nagesh, S. Goil, and A. Choudhary. Adaptive grids for clustering massive data sets. In *1st SIAM International Conference on Data Mining*, 2001.

21 C. Procopiuc, M. Jones, P. Agarwal, and T. Murali. A monte-carlo algorithm for fast projective clustering. In *ACM SIGMOD Conference*, pages 418–427, 2002.

22 P. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. In *8th ACM SIGKDD Conference Proceedings*, 2002.