

On high dimensional data spaces

S. Dey & S.A. Roberts

School of Computing, University of Leeds, UK

Abstract

Data mining applications usually encounter high dimensional data spaces. Most of these dimensions contain ‘uninteresting’ data, which would not only be of little value in terms of discovery of any rules or patterns, but have been shown to mislead some classification algorithms. Since, the computational effort increases very significantly (usually exponentially) in the presence of a large number of attributes, it is highly desirable that all irrelevant attributes be weeded out at an early stage.

Often, patterns of interest are embedded in lower dimensional *subspaces* of data. If the data space S has k attributes $\in \{a_1, a_2 \dots a_k\}$, then a n -dimensional subspace s_n of the data space S can be formed by selecting a combination of n attributes from the set $\{a_1, a_2 \dots a_k\}$, where $n \leq k$.

It is usual to tackle this problem by getting some attributes and subspaces identified by the user (or domain experts). For even moderately large number of attributes, the number of possible subspaces is so large, that it is quite unlikely that the ‘experts’ would be able to identify all the ‘interesting’ subspaces.

1 Introduction

The general problem, known as ‘the curse of high dimensionality’, has been studied extensively and several automatic methods for reduction of dimensionality have been reported in literature.

Data mining applications require that:

1. The results be comprehensible by the end-user
2. Data distributions potentially non-conformant to any of the canonical forms be handled
3. Potentially ‘interesting’ *subspaces* (as opposed to a subset of the original attributes) be identified

4. The algorithm be scalable
5. Noisy data be handled

In this paper we report a technique which addresses these five requirements.

2 Available Techniques

2.1 Principal Component Analysis

One of the most important and long established methods of dimensionality reduction is Principal Component Analysis (PCA). PCA has been studied and developed within the subject area of statistics for over two decades (see Jolliffe [1]). It attempts to define a linear combination of the original attributes so as to obtain 'explanatory' variables, the first few of which (the 'principal' components) 'explain' most of the variance in the original high dimensional data.

The concept of PCA has been progressed further by researchers in recent years (e.g. [2] to [7]). Kemsley [8] compares PCA to another (somewhat similar) method - Partial Least Squares (PLS) and reports that though PLS leads to greater reduction of dimensionality, it tends to introduce spurious groupings into the data.

2.2 Discretization

Another method for reduction of data and selection of (interesting) dimensions has been proposed by Liu and Setiono [9]. The method involves the discretization of continuous variables using the Chi-square distribution. The vertical and horizontal reduction (VHR) method checks for correlation between an attribute and the class values, based on which it tries to merge the ordered values of the attribute. The data is sorted on the values of an attribute and for that attribute adjacent values are placed in intervals. Next the χ^2 value for every pair of adjacent intervals is calculated. The pairs having the lowest χ^2 value are merged. Merging is continued until all pairs have χ^2 values above the parameter determined by the 'significance level'. The merging process is repeated with decreasing significance level until a given 'inconsistency level' is exceeded. This process is repeated for all continuous valued attributes. All attributes that merge to a single value are dropped, because they do not contribute towards the discriminating power of the dataset.

2.3 Feature Selection Techniques

Feature selection attempts to select the minimally sized subset of features according to the following criteria [10]:

1. the classification accuracy does not significantly decrease; and
2. the resulting class distribution, given only the values for the selected features, is as close as possible to the original class distribution, given all features.

An exhaustive evaluation of every possible subspace of $n < N$ features clearly requires the prohibitively expensive 2^N evaluations. Many heuristic computationally feasible methods have been developed. For example, [11] to [18].

Bauer et. al. [18] present a method based on signal-to-noise ratios along with comparisons with Principal Component Analysis and Discretization methods. A survey and comparison, of several other algorithms proposed since the 1970s, has been provided by Dash and Liu [10].

2.4 Discussion

The Principal Components Analysis and Partial Least Squares have strong theoretical basis and have been studied and tested for many years. However, they generate arbitrary combinations of the original attributes. These ‘artificial’ dimensions cannot be transformed back to real-world attributes easily understood by the end-user. Thus they are rendered generally unsuitable on account of requirement (1) in Section 1 above.

Discretization on the other hand, preserves the original attributes, but it assumes a chi-square distribution; a requirement that may not be satisfied by all real-world datasets encountered during data mining.

The host of feature selection algorithms available, preserve the original attributes and are not based on any data distribution, thus satisfying requirements (1) and (2). However, as the name suggests, these techniques attempt to identify relevant features (or attributes) rather than a set of *subspaces* (a combination of relevant attributes) of interest. The subset n of features (attributes), identified from the original N features, reduces the scale of the problem, but does not guarantee scalability for large N .

2.5 Agrawal's ‘Bottom up’ algorithm

Agrawal et. al. [19] present an algorithm (which is a part of their CLIQUE clustering method) that addresses the first four special requirements identified above. Their ‘bottom up’ algorithm attempts to identify *subspaces* that contain *clusters*, these being the subspaces of interest from the point of view of data mining.

They use a density based approach where a cluster is defined as a region that has a higher density of points than its surrounding region. The problem is to automatically identify projections of the input data onto a subset of the attributes with the property that these projections include regions of high density. To approximate the density of the data points, the data space is partitioned and the number of points that lie inside each cell (unit) of the partitioning is computed. The problem of identifying subspaces containing clusters lies in finding dense units in different subspaces.

The simplest way to identify dense units would be to create a histogram in all subspaces and count the points contained in each unit. This approach is infeasible for high dimensional data because the number of possible subspaces increases exponentially with the number of dimensions. The bottom-up algorithm exploits the monotonicity of the clustering criterion with respect to dimensionality.

It starts by determining 1-dimensional dense units in a single pass.. It then prunes the candidate 1-dimensional subspaces (the original set of attributes) based

on a strategy described below. Using the pruned set of $(k-1)$ -dimensional subspaces, it then generates the k -dimensional subspaces that are potentially dense. This is done based on the Monotonicity lemma [19] which states that: “If a collection of points S is a cluster in a k -dimensional space, then S is also part of a cluster in any $(k-1)$ -dimensional projections of this space”. One pass over the data determines the k -dimensional dense units. The pool of k -dimensional candidate subspaces is then pruned. The algorithm proceeds level by level until no more candidates are generated.

The candidate generation procedure produces the minimal number of candidates that can guarantee that all dense units will be found. The running time of the algorithm is therefore exponential in the highest dimensionality of any dense unit. If k is the highest dimensionality of any dense unit, k passes over the data will be required because one pass is required at every level of dimensionality. For m data points the time required would be $O(c^k + mk)$ for a constant c .

As the dimensionality of the subspaces considered increases, there can be an explosion in the number of dense units unless the pool of candidate subspaces is cut down by pruning. On the other hand, if the pruning strategy is too restrictive there is the possibility of missing some of the ‘interesting’ subspaces.

MDL-based Pruning: The basic idea underlying the Minimal Description Length (MDL) principle is to encode the input data under a given model and select the encoding that minimizes the code length.

For the purpose of pruning the pool of candidate subspaces, a measure called *coverage* has been defined. The coverage of a subspace x_{S_j} is the fraction of the database that is covered by the dense units in subspace S_j .

$x_{S_j} = \sum_{U_i \in S_j} \text{count}(U_i)$ where $\text{count}(U_i)$ is the number of points that fall within U_i

The objective is that subspaces with large coverage be selected and the rest pruned. To achieve this, the subspaces are sorted in the descending order of their coverage, and the sorted list divided into two sets: the selected set I and the pruned set P . For each set of candidate subspaces, the means of the cover fractions ($\mu_I(i)$ and $\mu_P(i)$) are computed, and for each subspace in that set the difference from the mean is computed. The code length is the sum of the bit lengths of the numbers that have to be stored. Applying the MDL principle: the optimal cut off point i is such that the code length is minimum. The code length $CL(i)$ is defined for the interval $1 < i < n$ where n is the number of subspaces in a pool of candidates.

Remarks: Two points regarding the use of MDL principle for pruning need to be highlighted:

1. The cut off point is based on the *relative* merit of the candidates within a set, determined by a measure (the coverage in this case), as opposed to an *absolute* merit based cut off.
2. Since, the Code Length is not defined for $i = 0$ and $i = n$, this strategy can neither select all the candidates, nor can it reject all of them.

The selection (or rejection) of candidates hinges on the measure of merit used. If the measure is a poor one, we could lose good candidates and / or generate a longer than necessary final list of subspaces. Therefore, the bottom-up algorithm presented by Agrawal et. al., though excellent in its handling of the five special requirements of data mining, is weakened by its pruning strategy.

3 An improved pruning strategy

In this paper we report an alternative pruning strategy that addresses the issues discussed above, and compare it with the MDL based pruning used by Agrawal et. al.

3.1 The measure of merit

The *coverage*, which is the fraction of the data points covered by the dense units, is an important indicator of ‘interestingness’. It measures how *widely* the subspace is *clustered*. It is controlled by the threshold value (τ) used to distinguish dense units from the surrounding background of non-dense units (or noise).

We introduce another component called *concentration*, which measures how *densely* the points within the clusters in the subspace are packed together. It is the ratio of the number of units to the number of dense units. For example, if for two candidate subspaces, the same fraction of the database is covered by the dense units (measured by the coverage component), our measure will assign a higher value to the one with fewer dense units (as measured by the concentration component) - indicating that it is the more ‘meritorious’ of the two because the points in its clusters are more densely packed.

Our *Measure of Merit (M)* is the product of *Coverage* and *Concentration*.

3.1.1 A benchmark for comparison: The Hopkins’ Statistic

In order to compare how our proposed *Measure of Merit* performs with respect to the *Coverage* measure used by Agrawal et al., we will need a ‘standard’ for comparison, in other words a direct measure of the ‘amount’ of clustering present in a dataset (or subspace) which is termed *Clustering Tendency* of the data.

Hopkins, [20] proposed a direct measure for clustering tendency of a set of data points. Known as the *Hopkins’ Statistic (H)*, it is based on the difference between the distance from a real point (in the dataset) to its nearest neighbour, U , and the distance from an ‘artificial’ randomly generated point within the data space to the nearest real data point, W .

The Hopkins’ Statistic is given by: $H = \sum U_i / (\sum U_i + \sum W_i)$.

It can be easily shown that if the dataset has no clusters, the value of H will approximate 0.5, and if the dataset is tightly clustered it would approximate 1. To obtain a stable value of H , eliminating any bias introduced by the selection of the random points and the artificially generated points, H has to be calculated manytimes with different seeds, and the average value of H calculated to

represent the true *Hopkins' Statistic* [21]. This makes the determination of H computationally complex.

Dubes and Zeng [22],[23] compared five commonly used methods for detecting presence of clusters in data on a wide variety of data sets. After an extensive series of Monte Carlo experiments, they concluded that in virtually every case the Hopkin's statistic had the highest probability of rejecting the null hypothesis (that the data space was a uniformly random space) when clusters were present.

Thus, while Hopkins' Statistic is too computationally intensive to use as a direct Measure of Merit for the purpose of the 'bottom-up' algorithm, it can be used as a benchmark to compare pruning based on our Measure of Merit with Agrawal's coverage and MDL based pruning.

3.1.3 Experiments with Synthetic data

Experiments were conducted on several sets of synthetic data to observe how the two methods behaved (with respect to *Hopkins' Statistic*) with changes in noise level, density of clusters and dimensionality of clusters.

The first dataset has 2000 data points and is entitled *SD01.txt*. Among its attributes (1-dimensional subspaces) are:

- V2 : Uniformly distributed
- V3 : 2 Normally distributed clusters - No noise
- V4 : 2 Normally distributed clusters - No noise
- V5 : 2 Normally distributed clusters - No noise
- V6 : 3 Uniformly distributed sparse clusters close to each other - No noise
- V7 : 1 Sparse and 2 dense clusters close to each other - No noise
- V8 : Uniformly distributed
- V9 : Uniformly distributed

From visual inspection (see Dey [24]) we can partially rank the 1-d subspaces according to "interestingness" as (V3, V4, V5), V7, V6, (V2, V8, V9), where brackets indicate equal ranking. Furthermore V6 appears nearly uniformly distributed.

The Bottom-up algorithm was run on the dataset with parameters: $\xi = 200$, $\tau = 30\%$. The results of the first level are summarised in the table below which indicates that our measure of merit, M , preserves the partial ranking expected, whereas the measure based on coverage alone reverses the ranking of V7 and V6 and places V5 close to V6 rather than closer to V3 and V4.:

Table 1: Measure of Merit of 1-d Subspaces of *SD01.txt* for $\xi = 200$, $\tau = 30\%$

Subspace	Coverage	Concentration	Measure of Merit (M)
V2	0.24	6.45	1.55
V3	0.91	4.76	4.33
V4	0.93	4.26	3.96
V5	0.69	6.06	4.18
V6	0.64	2.60	1.67
V7	0.60	4.65	2.79
V8	0.21	7.41	1.54
V9	0.24	6.45	1.54

Among the 2-dimensional subspaces, the following have been designed to be particularly *interesting*:

- V3-V4 : 2 distinct nearly circular clusters - No noise
- V4-V5 : 2 distinct elliptical clusters - No noise
- V3-V5 : 2 distinct elliptical clusters - No noise
- V3-V8 : 1 low density rectangular cluster, 1 sparse rectangular cluster
- V6-V8 : 3 sparse rectangular clusters close together
- V2-V8 : Completely uniform

Table 2 indicates that our measure, *M*, ranks the selected 2-d subspaces in accordance with the Hopkin's statistic. The measure based on coverage alone is nearly as good (although it fails to distinguish V3-V8 from V6-V8).

Table 2: Measure of Merit of selected 2-d Subspaces of SD01 . t x t for $\xi = 200$, $\tau = 30\%$

Subspace	Hopkin's Statistic (<i>H</i>)	Coverage	Concentration	Measure of Merit (<i>M</i>)
V3-V4	0.991	0.84	4.35	3.67
V4-V5	0.990	0.72	4.35	3.13
V3-V5	0.989	0.68	4.55	3.10
V3-V8	0.877	0.31	4.35	1.35
V6-V8	0.809	0.31	2.94	0.92
V2-V8	0.499	0.17	2.25	0.39

Among the 3-dimensional subspaces, the following were designed to be particularly *interesting*:

- V3-V4-V5 : 2 dense distinct clusters - No noise
- V3-V4-V7 : 2 distinct clusters - 1 very small and dense, 1 large and less dense - No noise
- V3-V4-V6 : 2 small dense clusters close together, 1 large less dense cluster - No noise
- V4-V5-V6 : 3 low density clusters close together
- V5-V6-V8 : 2 sparse irregular clusters close together
- V2-V8-V9 : Completely uniform

The relevant part of the results of the third level (3-d Subspaces) are summarised in the table below:

Table 3: Measure of Merit of selected 3-d Subspaces of SD01 . t x t for $\xi = 200$, $\tau = 30\%$

Subspace	Hopkin's Statistic (<i>H</i>)	Coverage	Concentration	Measure of Merit (<i>M</i>)
V3-V4-V5	0.982	0.64	4.44	2.84
V3-V4-V6	0.956	0.53	4.65	2.45
V3-V4-V7	0.950	0.66	2.06	1.35
V4-V5-V6	0.947	0.54	2.41	1.29
V5-V6-V8	0.812	0.18	4.35	0.80
V2-V8-V9	0.493	0.05	2.67	0.13

From Table 3 we see that subspaces with distinct clusters V3-V4-V5, V3-V4-V6 and V3-V4-V7 score higher than those with relatively less well formed

clusters: V4-V5-V6 and V5-V6-V8. The completely uniform subspace (V2-V8-V9) has a very low M value.

Once again we observe that our measure of merit produces the same ranking as Hopkin's statistic, whereas the measure based on coverage does not.

In our next experiment we use dataset SD02.txt which has 2400 data points. It has been created from dataset SD01.txt by adding 20% noise (i.e. 400 uniformly distributed data points).

Running the Bottom-up algorithm on the dataset with the same parameters: $\xi = 200$, $\tau = 30\%$, produces results that are similar to those from our previous experiment. In all cases (1-d, 2-d and 3-d) our ranking agrees with that from Hopkin's statistic. There are some changes in the ranking (see Dey [24] for full details), but these affect the ranking of subspaces which intuitively we consider to be close together (for example V4 changes places with V3).

3.1.4 Summary of results

From the observations made in our experiments, we can conclude that our bottom-up algorithm can successfully identify interesting subspaces in high dimensional data, and that it is fairly robust in the presence of noise. Also, the results show that our Measure of Merit (M) is a better measure of the 'true' merit and order of merit (taking Hopkins as the standard) than the *Coverage* measure used by Agrawal et. al.

3.2 Choice of the 'Number of Intervals' parameter (ξ)

The data points enclosed within the 'units' (during the process of selecting the dense units), can be viewed as a 'sample' in statistical terms. If the number of intervals used to split up the data into units (ξ) is large, the units will be small and each such unit will enclose a small number of data points. In statistical terms, the 'sample size' will be small. Sampling using small samples follows the variations in the data 'too' closely. For example, even insignificant random fluctuations in the uniformly distributed values of the attribute V2 lead to a *coverage* value of 0.24 and M value of 1.55 (Table 1 above). Statistically, a large sample (more than 50 data points) would 'smooth' out the small random variations and bring out the 'true' broader characteristics of the data.

The results (of the first level) of running the Bottom-up algorithm with parameters: $\xi = 35$, $\tau = 30\%$, on dataset: SD01.txt are given in the table below.

Table 4: Measure of Merit of 1-d Subspaces of SD01.txt for $\xi = 35$, $\tau = 30\%$

Subspace	Coverage	Concentration	M (for $\xi = 35$, $\tau = 30\%$)	M (for $\xi = 200$, $\tau = 30\%$)
V2	0.000	0.00	0.00	1.55
V3	0.910	4.38	4.01	4.33
V4	0.940	3.89	3.66	3.96
V5	0.681	5.83	3.97	4.18
V6	0.653	2.33	1.52	1.67
V7	0.595	4.37	2.60	2.79
V8	0.038	35.0	1.33	1.54
V9	0.000	0.00	0.00	1.54

It may be noted that whilst the value of M for the interesting attributes has not decreased significantly, M for the uniformly distributed attributes (uninteresting ones) has vanished. Our experiments show that this holds true for higher levels of dimensionality, as well as in the presence of noise in the data.

3.3 An ‘Absolute’ Cut-off value of M for pruning subspaces

In Section 2.5 while discussing the weaknesses of Agrawal's pruning strategy we have mentioned the need for the pruning to be based on an ‘absolute’ cut-off as opposed to the relative merit based pruning used by Agrawal. In this Section we discuss a way of arriving at an appropriate cut-off value.

When sampling is done using large samples, the *Sampling Distribution* (of the mean) approximates a *Normal Distribution* independent of the actual distribution of the population being sampled, and the *Sampling Error* is inversely proportional to the *Square Root of the Sample Size*.

Also, the probability of including values unrepresentative of the population (arising due to Sampling Error) is $P_k = C(1/\sqrt{n})^k$ for some constant C , where k is the number of dimensions of the subspace and n is the sample size.

This implies that as the dimensionality increases, the number of data points included in the dense units (i.e. *coverage*) will decrease. We would therefore expect the value of M for all the subspaces to decrease with increase in dimensionality. Our experimental results are consistent with this hypothesis.

Any cut-off value of M would therefore need to be ‘scaled’ depending on the dimensionality of the set of subspaces being pruned. We propose the *Scale parameter*: $S_k = (1 - 1/\sqrt{N/\xi})^{k-1}$ where k is the dimensionality and N is the number of data points. The cut-off value of M for 1-dimensional subspaces $C_M(1)$ can now be scaled to k -dimensional subspace: $C_M(k) = C_M(1) S_k$. The input parameter $C_M(1)$ determines how harshly the candidate subspaces are pruned. A higher value leads to harsh pruning, so that only the very promising subspaces are selected and the final list of interesting subspaces is short. A lower value allows pruning to be more lenient and a longer list of subspaces results.

Running the algorithm on SD02.txt with $\xi = 45$, $\tau = 35\%$, $C_M(1) = 1.70$ produced the following list of interesting subspaces:

Table 5: Pruned Subspaces for $\xi = 45$, $\tau = 35\%$, $C_M(1) = 1.70$ on SD02.txt

k	$C_M(k)$	Selected Subspaces	M	H	Pruned Subspaces	M	H
1	1.70	V3	3.96		V2	0.00	
		V4	5.22		V8	0.00	
		V5	3.66		V9	0.00	
		V6	1.81				
		V7	2.52				
2	1.47	V3-V4	4.59	0.823	V3-V6	1.46	0.755
		V3-V5	2.50	0.811	V5-V7	1.46	0.757
		V3-V7	2.25	0.787			
		V4-V5	3.05	0.818			
		V4-V6	1.53	0.770			

<i>k</i>	$C_M(k)$	Selected Subspaces	<i>M</i>	<i>H</i>	Pruned Subspaces	<i>M</i>	<i>H</i>
3	1.27	V4-V7	2.28	0.797			
		V5-V6	1.53	0.760			
		V6-V7	2.41	0.807			
		V3-V4-V5	2.70	0.803	V3-V5-V7	1.18	0.771
		V3-V4-V6	1.33	0.728	V4-V5-V7	1.22	0.776
		V3-V4-V7	1.37	0.784			
		V3-V5-V6	1.30	0.795			
		V4-V5-V6	1.40	0.758			
		V4-V6-V7	2.18	0.847			
		V5-V6-V7	1.37	0.793			
4	1.09	V3-V4-V5-V6	1.14	0.788			
		V3-V4-V5-V7	1.11	0.787			
		V3-V4-V6-V7	1.92	0.850			
		V3-V5-V6-V7	1.22	0.833			
		V4-V5-V6-V7	1.28	0.833			

This value of $C_M(1)$ led to a very lenient pruning (However, there was still a ten fold reduction in the number of subspaces that would need to be examined by the Data Mining algorithm (from $2^8 - 1 = 255$ to just 25)., selecting all subspaces that were even slightly more interesting than a uniformly distributed subspace. All the 4-d subspaces generated from the 3-d ones, based on the *Monotonicity lemma*, were selected. This would not have been possible if MDL based pruning had been used.

Increasing $C_M(1)$ to 1.80 (for example) leads to a much harsher pruning, reducing the length of the list to 17. Thus, the user has control over the pruning process via the parameter $C_M(1)$.

The results of running the algorithm on SD02.txt with $\xi = 45$, $\tau = 35\%$ and MDL Pruning are presented in the table below:

Table 6: Pruned Subspaces for $\xi = 45$, $\tau = 35\%$ using MDL Pruning on SD02.txt

<i>k</i>	Selected Subspaces	Coverage	Hopkins (<i>H</i>)	Pruned Subspaces	Coverage	Hopkins (<i>H</i>)
1	V3	0.79		V2	0.00	
	V4	0.81		V8	0.00	
	V5	0.65		V9	0.00	
	V6	0.85				
	V7	0.51				
2	V3-V6	0.75	0.755	V3-V4	0.71	0.823
	V4-V6	0.75	0.770	V3-V5	0.61	0.811
				V3-V7	0.45	0.787
				V4-V5	0.61	0.818
				V4-V7	0.44	0.797
				V5-V6	0.51	0.760
				V5-V7	0.26	0.757
				V6-V7	0.43	0.807
3	V3-V6-V7	0.41	0.789	V4-V6-V7	0.39	0.847

The pruning in this case was quite harsh - only 8 subspaces were selected. The actual subspaces selected were also different from those of our proposed method. Many good subspaces (as per *Hopkins' Statistic*) were rejected (e.g. V3-V4, $H=0.823$; V4-V5, $H=0.818$; V3-V5, $H=0.811$ of the 2-d Subspaces and V4-V6-V7, $H=0.847$ of the 3-d Subspaces) and poorer ones chosen instead (e.g. V3-V6, $H=0.755$; V4-V6, $H=0.770$ of the 2-d Subspaces and V3-V6-V7, $H=0.789$ of the 3-d Subspaces). This was because the subspaces were ranked differently by using *coverage* instead of our *Measure of Merit*.

4 Conclusions

This research set out to address the problems posed by high dimensional data spaces. Specifically, we concentrated on finding a technique satisfying the five requirements identified in Section 1.

From the study of various available approaches to the problem, we have identified the 'Bottom-up' algorithm presented by Agrawal et. al. [19] to be one that meets most of the requirements. We have proposed changes to that algorithm that enhance its efficacy. We have used specially designed synthetic data to study the effects of the proposed changes, and the results demonstrate the advantage of our approach.

References

- [1] Jolliffe, I.T., Principal component analysis, Springer, New York, 1986.
- [2] Xu, L., Yuille, A.L., Robust principal component analysis by self-organizing rules based on statistical physics approach, *IEEE Transactions on Neural Networks*, **6**(1), 1995.
- [3] Yang, T., Wang, S., Robust algorithms for principal component analysis, *Pattern Recognition Letters*, **20**(9), pp. 927-933, 1999.
- [4] Partridge, M., Jabri, M., Robust Principal Component Analysis, *Neural Networks for Signal Processing. Proc. of the IEEE Workshop*, pp. 289-298, 2000.
- [5] Casin, P., A generalization of principal component analysis to K sets of variables, *Computational Statistics & Data Analysis*, Vol. 35, pp. 417- 428, 2001.
- [6] Ravi, V., Reddy, P.J., Zimmermann, H.J., Pattern classification with principal component analysis and fuzzy rule bases, *European Journal of Operational Research*, Vol. 126, pp. 526-533, 2000.
- [7] King J.R.; Jackson D.A., Variable selection in large environmental data sets using principal components analysis, *Environmetrics*, **10**(1), pp. 67-77, 1999.
- [8] Kemsley, E.K., Discriminant analysis of high-dimensional data: a comparison of principal component analysis and partial least squares data reduction methods, *Chemometrics and Intelligent Laboratory Systems*, Vol. 33, pp. 47-61, 1996.
- [9] Liu, H., Setiono, R., Dimensionality reduction via discretization, *Knowledge-based Systems*, Vol. 9, pp. 67-72, 1996.

- [10] Dash, M., Liu, H., Feature Selection for Classification, *Intelligent Data Analysis*, Vol. 1, pp. 131-156, 1997.
- [11] Kononenko, I., Hong, S.J., Attribute selection for modelling, *Future Generation Computer Systems*, Vol. 13, pp. 181-195, 1997.
- [12] Liu, Y.H., Siddiqui, K.J., Using a Bicriteria Boolean Linear Programming Model for Parameter Selection in Large Multicategory Classification Problem, *Mathematical and Computer Modelling*, **24(2)**, pp. 75-81, 1996.
- [13] Somol, P., Pudil, P., Novovicova, J., Paclik, P., Adaptive floating search methods in feature selection, *Pattern Recognition Letters*, Vol. 20, pp. 1157-1163, 1999.
- [14] Shapira, Y., Gath, I., Feature selection for multiple binary classification problems, *Pattern Recognition Letters*, Vol. 20, pp. 823-832, 1999.
- [15] Holz, H.J., Loew, M.H., Multi-class classifier-independent feature analysis, *Pattern Recognition Letters*, Vol. 18, pp. 1219-1224, 1997.
- [16] Kuncheva, L.I., Jain, L.C., Nearest neighbor classifier: Simultaneous editing and feature selection, *Pattern Recognition Letters*, Vol. 20, pp. 1149-1156, 1999.
- [17] Liu, H., Setiono, R., Some issues on scalable feature selection, *Expert Systems with applications*, Vol. 15, pp. 333-339, 1998.
- [18] Bauer KW, Alsing SG, Greene KA, Feature screening using signal-to-noise ratios, *Neurocomputing*, Vol. 31, pp. 29-44, 2000.
- [19] Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P., Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications, *Proc. of the ACM SIGMOD International Conference on Management of Data*, Seattle, Washington, June 1998.
- [20] Hopkins, B., A New Method for Determining the Type of Distribution of Plant Individuals, *Annals of Botany*, Vol. 18, pp. 213-227, 1954.
- [21] Forina, M., Lanteri, S., Esteban Díez, I., New index for clustering tendency, *Analytica Chimica Acta*, Vol. 446, pp. 59-70, 2001.
- [22] Zeng, G., Dubes, R., A Comparison of Tests for Randomness, *Pattern Recognition*, Col. 18, pages 191-198, 1985.
- [23] Dubes, R., Zeng, G., A Test for Spatial Homogeneity in Cluster Analysis, *Journal of Classification*, Vol. 4, pp. 33-56, 1987.
- [24] Dey, S., Application of Knowledge Discovery Techniques to a Combination of Georeferenced, Tuple-oriented Data Spatial Data, PhD. Thesis, School of Computing, University of Leeds, 2002.