Retrieval of optimal subspace clusters set for an effective similarity search in a high-dimensional spaces

© Ivan Sudos
Saint-Petersburg State University
Saint-Petersburg
iv.teh.adr@gmail.com

Abstract

High dimensional data is often analysed resorting to its distribution properties in subspaces. Subspace clustering is a powerfull method for elicication of high dimensional data features. The result of subspace clustering can be an essential base for building indexing structures and further data search. However, a high number of subspaces and data instances can conceal a high number of subspace clusters some of which are difficult to analyse within search algorithm. This paper presents a model of generic indexing approach based on detected subspace clusters and the way to find an optimal set of clusters to have an acceptable tradeoff between search speed and relevance.

1 Introduction

Search and clustering are two most extensive problems of data analysis in high dimensional spaces. Since the roots of complexity that occurs in this domain were established, plenty of particular aspects have been elicited and regarded so far. Many of approaches for solving high-dimensional problems regard only these particular complexity aspects, as it is quite sophisticated and practice detached to solve the problem in general. In recent twenty years a number of solutions have been proposed to solve certain problems of searching and clustering in high dimensional spaces. Clustering and indexing are strongly associated with each other in high dimensional spaces: resolving of indexing problems can cause the need to resolve clustering problems.

Regarding only search and clustering problems we will refer to a high-dimensional vector space as a search space. The general challenge for search and clustering in high dimensional spaces is called "curse of dimensionality" first stated by R. Bellman [1]. It has two key aspects. The first one lies within the following fact: if the number of dimensions grows the information under analysis in the search space become cumbersome. And the second one is the metric related problem: in high dimensions we often can't state that two vectors are similar or different.

In most of cases we usually can't build reliable algorithm and data structures (indexes) to handle exact match search with acceptable latency. Here in the first place we consider approximate similarity search.

Most of the approaches to search and indexing problems can be divided into following categories:

1. Low-dimensional algorithms adaptations, like ones that use R-trees. Here we try to fix some particular problems of search algorithm for low-dimensional search spaces to make it somehow feasible in high-dimensional space. However, it tends to work acceptable for relatevely low number of dimensions (doesn't exceed dozens)

2. Data distribution based algorithms. These alorithms take into account distribution's properties of the data. Number of them proceed dimensional reduction techniques like principal component analysis or subspace clustering to fight the curse of dimensionality.

3. Random projection based algorithms. These algorithms tries to decrease the volume of scanned information in a search space by grouping it's elements with a degree of randomnicity. Some realizations of locally sensitive hashing [16] relates to this category.

All of the approaches in one way or another try to solve the curse of dimensionality. This implies they try to reduce time comlexity of search or increase relevance, or do the both. In this paper we consider the second category of solutions. Here we have a point of contact with clustering problem. Analysis of data distribution is closely related to clusteing. Clustering in high dimensions have several approaches and each approach apart can use its

Proceedings of the 14th All-Russian Conference "Digital Libraries: Advanced Methods and Technologies, Digital Collections" — RCDL-2012, Pereslavl Zalesskii, Russia, October 15-18, 2012.

own cluster model. This paper stays only on subspace and projection clustering approches [2]. In accordance with [2], subspace clustering aims to find all clusters in all possible subspaces and projection clustering assignes each vector to exactly one subspace cluster. For example, a set of photos can be placed to one cluster if projection clustering algorithm finds no difference in their color hystogram characteristics. At the mean time, a subspace clustering algorithm will form(if any exists) clusters for all possible characteristics: hystograms, shapes, gradients etc. Despite it looks not so flexible as principal component analysis methods that can detect arbitrary manifolds or just clusters in not axis-parallel dimensions, subspace clustering have one important advantage: locality property [3]. It means that subspace clustering algorithms can determine a set of relevant dimensions (relevant subspace) locally for each part of the search space or subset of data vectors.

Our goal is to understand how clustering results can be coupled with similarity search in high-dimensional spaces. This paper introduce a generic approach to utilize detected subspace clustering within search. We state a key optimization problem that allows to find the best tradeoff between search relevance and speed. The implies selection of the best subset of subspace clusters that can provide the best relevance with a guaranteed retrieval minimal complexity.

2 Related works

Many of works were presented on indexing and clustering apart. Applications of clustering for building a search index are implicitly described in works about nearest neighbours in high dimensional spaces. Some works show how subspace clustering results can be used as base for tree-like index structures. Although a link between parameters of detected clusters and efficiency of nearest neighbours search is not presented. Several papers regard a subspace clustering process quality from the point of view of data redundancy.

Here we first state the optimization problem that arises when index structure is based on detected subspace clusters. The problem links search effectivity (speed and relevance) and properties of detected clusters. Thereby our work aims to link subspace clustering and nearest neighbour search. Index structures can be based on detected subspace clusters in various ways, however an existing indexing approaches don't consider affects of underlying clustering.

Some papers that considers indexing and clustering problems in high dimensional spaces are reported below.

Indexing: [5] introduces Bregman ball trees index structure that is reconsideration of ball trees using Bregman divergence instead of classic metric. Though it is not supposed to be used in high dimensions it introduces a feasible concept of applying Bregman divergences to known indexing structures instead of metrices.

Having distance functions like Bregman divergences in use, the search can be more complicated. The computational difficulty of such functions is higher then simle functions like Euclid metric. This aspect is taken into account in this paper.

X-tree [6] is spacial tree based on hyper rectangle partitioning of search space shows how well known low-dimensional index structure R-tree can be adapted for relatevly high number of dimensions by rejecting rectangles overlaping. However X-tree have its capability limits. This is an example of case when taking into account all dimensions simultaneously leads to index structure size of the same order as the data.

Clustering: The clustering techincs used in this paper refer to projection and subspace clustering. The key survey on clustering algorithms in high dimensions was conducted in [2]. Algorithms are categorized and compared. Main groups distinguished are: *Axis-parallel subspace and projection clustering*, *Arbitrary oriented subspaces clustering* and *Pattern based clustering*. We are also interested in description of particular algorithms of subspace and projection clustering and clustering models they use [3]. A surevey compares different subspace clustering performance. Clustering quality evalutation is regarded along with results of proposed optimization problem solution [7].

Clustering within indexing A recent work [4] shows us how we can build tree-like indexing structure in high dimensional space atop known clusters set. This work doesn't imply special method of clusterization and thus doesn't regard how clustering influences on efficiency of search. Though this search approach helps to evaluate clustering results experimentally.

3 Subspace clustering and similarity search

The common goal of indexing is to prune search space by compression and/or elicit relations between parts of the data (trees and space partitioning). Such approaches as Locally sensitive hashing and VA-files are compression [8] techinques that allows to approximat a group of near vectors with a single object. These techniques suffer a curse of dimensionality as well in high dimensional spaces. Locally sensitive hashing leads to low relevancse

due to distance invisibility. Space partitioning is not feasible since a number of hyperrectangles can exceed or become close to the number of data vectors and the search will be not less complex as and exhaustive bypass through all the data. Clustering suffers from distanse invisibility in high dimensions as it was described above. In our model we consider subspace clustering as a solution to proceed data compression to be used as a base of indexing structure. This considiration is maximally abstracted from the complete indexing structure and the algorithm of similarity search. Thus we assume that a search algorithm and an index structure operates a set of objects that represent grouped (clustered) data and perform retrieval on the smaller space thus with smaller precision. This requires a function that calculates relevance of the cluster for a query q and this function should avoid full scan of cluster members as it leads to an exhaustive search. So far search algorithm is assumed to do the following:

- Consider subspace clusters as a primary result of data compression.
- Use some approximate fast enough distance function to calculate relevance of the given subspace cluster with respect to a given query vector q.
- Find the most relevant cluster(s) and take them into the furhter consideration.

The following considerations show how subspace clustering features affect the tradeoff between search speed and relevance.

Let the subspace cluster $c = \{V, S\}$ be any subset V of data vectors so that maximum deviation in the subspace S of any $v \in V$ from the rest of V is less then given threshold h *and* V contains not less elements then p. So that the cluster is considered to be any clot of vectors in the any subspace with boundered parameters of density and a number of elements.

The first consideration is the measure of data compression provided with subspace clustering. The general assumption about query is that it is an uniformely distributed vector in the search space. We assume that the complexity of search algorithm is monotonic non-decreasing function $f(N, q)$ of N where N is a number of objects in the search space and q is a query vector. Since subspace clustering compression is performed N depicts a number of detected subspace clusters to be used by search algorithm. Without respect to the retrieval algorithm we assume the need to calculate relevance of a random detected cluster for a query q. So $Q(q, c)$ is a function that calculates relevance of c with respect to q. Let $Q \in O(g(|V|, \dim(S)))$.

The second consideration is the measure of relevance of detected clusters. Again, suppose we have an uniformly distributed query q. Let $C = \{c_1 \dots c_N\}$ will be the set of detected subspace clusters. Each subspace cluster c_i represents a pair $\{O_i, S_i\}$ of set of vectors it contains: $V = \{v_1 \dots v_k\}$ and a set of dimensions $S = \{d_1 \dots d_l\}$ that determines a subspace. Let introduce a relevance of a given cluster c of size k for a given query q:

$$R(c, q) = R_{\dim}(c, q) \frac{\sum\limits_{v \in V} \frac{1}{dist(q, v)}}{k}$$

The right side represents a product of: *subspace relevance function* $R_{\dim}(c, q)$ that represents a relevance of subspace where the data forms cluster c by the average inverted distance from a query to a member v of the cluster. We accept the Manhattan distance function and Euclidean distance function here as they were prooved [9] to be the only suitable for a distance measurment in high dimensional spaces.

Relevance calculation of cluster c with respect to query q should avoid iteration through all cluster's members. The relevance calculation function is supposed to be approximate. Having this consideration we introduce an approximate cluster relevance as

$$R_{\text{approx}}(c, q) = R_{\dim}(c, q) Q(c, q)$$

where $Q(c, q)$ is an approximate distance function. Let denote complexity of it as $g(|V|, |S|) = g_q(|V|, |S|, V, S)$. In further we will show an examples of this functions.

The general idea of this paper is to understand how the set of detected clusters should be selected to obtain the required search speed and relevance ratio. Denote the set of all the possible subspace clusters for a given search space as C. The subset C^* of C is an argument of optimization and the optimization problem is to obtain such C^* that will produce the best (in terms of retrieval speed and relevance) result for a random query q. The first optimization problem:

$$\begin{cases} E(R(c^*, q)) \to \max \\ c^* = \arg\max\limits_{c \in C} R_{\text{approx}}(c, q) \\ N \le N_{\max} \\ E(g_q(|V|, |s|)) \le \gamma \quad \forall q \end{cases} \tag{1}$$

The first optimization problem's goal is to maximize mean relevance of *the most* suitable cluster *determined by a given relevance calculation function* R_{approx} for a random query q. The constraints are to keep the number of objects under analysis (subspace clusters) below the given bound $N_{\max}$ and keep calculation complexity of the R_{approx} in a given bounds.

Let's introduce another optimization problem:

$$\begin{cases} \min\limits_{q \in H} R(c^*, q) \geq R_{\min} \\ c^* = \arg\max\limits_{c \in C} R_{\text{approx}}(c, q) \\ N \to \min \\ E(g_q(|V|, |s|)) \to \min \quad \forall q \end{cases} \tag{2}$$

The second one's goal is to find the simplest clusters set that keeps relevance rate in the given bounds. The latest problem is off less interest as it is difficult to user to assign relevance constraints apriori. The stated problems implies calculation of mean relevance calculation, though it is near impossible to perform for a whole search space. The introduction of mean relevance denotes the following:

The goal of optimization is to select a subset of subspace clusters such that the mean value of real relevance of given cluster for given query is the highest when the approximate relevance value is the highest.

This Optimization problems can be formulated for a low dimensional spaces as well. Though the abcence of the need to take into account subspace and simple mechanisms of query point classification (like MBR []) lead to the simple solution as regarding only the set of clusters that satisfy given density and size. For a high dimensions the most principal difference is a number of possible subspace clusters.

The number of detected clusters in all subspaces can be significantly bigger then in a low dimensional space.

That is because of 2^d number of all possible subspaces for a d-dimensional space. If all possible subspace clusters are considered, the relevance of the search is supposed to be higher since it is possible to pick up the nearest cluster for a given query q. But in this case N can be significantly high and bypass complexity of all the clusters can approach the complexity of an exhaustive search. The other tradeoff can be observed when dealing with subspace clusters in a relativly high dimensional subspaces. The relevance of this clusters can be higher but it is hard to analyse accessory of q to these clusters since there are a lot of attributes to be taken into account.

The rest of paper is dedicated to analysing the internal structure of functions presented in a optimization problems and possible algorithmical solutions for a stated problems.

3.1 Dimensionality effects on relevance

$R_{\dim}(C)$ was introduced as a function that reflects relevance of subspace where cluster C is detected. $R_{\dim}(C)$ have a direct impact on the search relevance of the given cluster. Most of papers [2], [6] [9] that tries to evaluate subspace or projection clustering omit details of subspace impact on clustering quality. The are three known groups of methods for automatical evaluation of cluster dimensionality relevance:

- Rate subspaces and thus subspace clusters higher if their dimensionality is higher [10], [2]
- Introduce generic cost function $K(O, S)$ that rates relevancy of given subspace cluster (O, S) [10], [11]
- Measure cluster separability within a given subspace. [9] That means subspace cluster (O, S) is rated higher if each vector in O is far enough from any other vector in subspace S that is not in the cluster.

The first approach is fair enough in case of the nearest neighbour retrieval: the user is interested in matching all the coordinates of given query vector q until certain subset of dimensions is not specified. In that case dimension clustering relevance can be determined as $R(c) = R(S, O) = |S|$.

The second approach can be used to assign weight to particular dimensions or vectors. This approach is senseble in case of specific origins of the data. Let's regard a vectors in a finite dimensional vector space that are obtained by orthogonal projection of time-series data in an infinite dimensionsal space. Then the user can be interested in a uniform sample in time. At the other side sequental dense subset of times can be more relevant. In these cases K can be denoted as

$$K(O, S) = \sum_{0 < i \leq N} (s_i^2) - n \left(\frac{\sum\limits_{0 < i \leq N} (s_i)}{n} \right)^2$$

(variance multiplied n) or

$$K(O, S) = \frac{n}{s_n - s_1}$$

correspondently.

The third way to introduce K is generally refers to automatic detection of relevant dimensions for a given subset of vectors. The one way is proposed in [9]. To detect the most relevant set of dimensions for a given vector v_q one have to compose distribution function of distances $\Phi(d) = P(\text{dist}(v_q, v) < d), d \in \Re$ where v is a random vector from a vector space. This approach requires computation of distribution function that in turn requires itaratibreon over the whole search space. The are two drawbacks here. It is rather heavy operation to compute distribution function for a single vector and have a $O(N^2)$ cost. And the second drawback is that most of subspace/projection clustering algorithms imply detection of relevant vectors themselves and can expose this information both with subspace cluster.

3.2 Cluster geometry effect on relevance

Influence of cluster geometry on distance to a query point is studied in this section. Distance function is used to evaluate relevance of the clusters with respect to a query. The relevance function was introduced above as $R_{\text{approx}}(c,q) = R_{\dim}(c,q)Q(c,q)$. And the $Q(c,q)$ is the approximate classificatoin function used to mitigate relevance calculation time. The Idea is to find out what clusters is suitable to be evaluated with Q. Some shapes of clusters causes the situation when Q determines them as relevant when $\frac{\sum_{v \in V} \frac{1}{\text{dist}(q,v)}}{k}$ turns out to be low. A Bregman divergence can be used as a measure of accessory to the cluster. A Bregman divergence [12] distance functions such as Mahalanobis distance [13] can be calculated using only mean vector of the cluster and covariance matrix that can be significantly faster than bypassing all the cluster's members. For this purpose mean and covariance matrix should be stored along with the object that represents subspace cluster.

Another variant of Q is an indicator function of Minimal Boundaring Rectangle (MBR) accessory. This function is generally simpler to be calculated though for a relatively high-dimensonal hyper rectangles can be very inaccurate as the volume of hyperrectangle increases exponentially as number of dimensions grows.

3.3 Examples

Having determined possible approximated relevance functions we will give some examples that reflects 1.

Example 1 Let's have

$$R_{\text{approx}}(c,q) = (|S|I(q, MBR(c)))^{-1},$$

where $I(q, MBR(C)) = 1$ if q belongs to the minimal boundaring rectangle that contains c and $I(q, MBR(C)) = 0$ otherwise. S is the set of subspace dimensions as was intoduced above. Assume there is query q and it falls into two minimal boundaring rectangles in 2 different subspaces as depicted at figure 3.3. These rectangles bound 2 subspace clusters.

The subspace cluster in the left has more dimensions - 5, when the right one has only 3. So that if to accept $R_{\text{approx}}(c,q) = 5$ and thereby take into consideration left cluster we will select the subspace cluster (and so that the nearst neighbours) less relevant then right cluster in terms of real relevance. The average distance to q from the right cluster is significantly less. The solution of 1 let us escape from such false guess and leave only the most relevant subspace clusters in the indexing set.

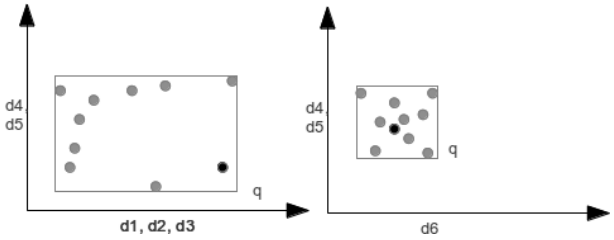

Figure 1: approximate relevance error:
R_{approx} value is bigger for left picture,
however, a real relevance is bigger at right one

Example 2 Using of Mahalanobis distance as a base for R_{approx} can lead to erroneous guesses as well. Lets have

$$Q(c,q) = \sqrt{(q-\mu(c))^T \operatorname{Cov}^{-1}(c)(q-\mu(c))}$$

where $\mu(c)$ is a centroid of c and $Cov(c)$ is sample covariance. This function is named Mahalanobis distance. Let $R(c) = R(S,O) = |S|$. Thereby approximate relevance function is

$$R_{\text{approx}}(c,q) = \left(|S|\sqrt{(q-\mu(c))^T Cov^{-1}(c)(q-\mu(c))}\right)^{-1}.$$

The situation shown at figure 2 occurs when linear classifier (like Mahalanobis distance) can't do reliable guess against some distributions other then normal one. The subspace cluster at the left is less distant in terms of Mahalanobis distance, but average distance to a set at the right picture is less.

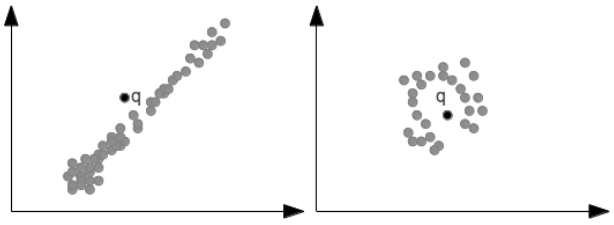

Figure 2: Mahalanobis distance relevance error

In the given examples it is possible to discard one of two clusters regarded. To leave only the clusters that will have the maximum relevance when selected by maximum approximate relevance value is the goal of 1.

4 Solution of optimization problem

The solution of the problem 1 in general case is still open problem until there is no additional assumptions about the query vector distibution. The main difficulty is to calculate corresponding mean of

the relevance. In relativly low dimensional subspaces (2-3 dimensional) the experiments shows that the Branches and bounds method is applicable. The most complex part is to calculate the mean calculation that leads to multople integral sums calculations.

An approach we propose to solve 1 is based on reduction of 1 to a well-known knapsack problem [15]. Suppose a set of B items. Each item i has its own value v_i and weight w_i. Knapsack problem implies selection of M items. The total weight of selected items must be below threshold ω and the total value must be maximized. In case of 1 the items are subspace clusters; The weight depicts a a value difficulty of approximated relevance computation. To define a value of the subspace cluster we propose to perform a test: for each vector x within given suspace cluster c_i a pair

$$(u(x); w(x)) := (R(x, c_i); R_{\text{approx}}(x, c_i))$$

is calculated. The value v_i is then taken as $v_i = \text{corr}(u, w)$ (a correlation). Having this as a value, corresponding knapsack problem can be solved in common way, for example, recurrently. This approach is to be proven, though for an experimental data it yields results.

Another problem that uprises is how to put the most complete set of existing subspace clusters to a consideration. For an axis-parallel clustering it can be performed using bottom-up approaches like CLIQUE [3] and MAFIA [14] as they are aimed to detect all possible clusters in all the possible subspaces. Once the all available subspace clusters are obtained the reduction should be proceeded: the best clusters should be selected in accordance with 1.

Now, we can formulate an algorithm that elicit near optimal subset of subspace clusters to be used in indexing. For a geneneral case solution here we propose to use our approach with knapsack problem preceded with MAFIA algorithm.

Algorithm 1 Clustering optimization:

```
Execute MAFIA algorithm to obtain full set of subspace clusters;
for all subspace cluster c_i do
    calculate weight w_i based on R_approx
    for all vector x in cluster c_i do
        calculate u(x) := R_approx(x, c_i)
        calculate w(x) := R(x, c_i)
        write pair (u(x), w(x)) to array A
    end for
    v_i := correlation(u, w)
end for
solution := SolveKnapsack(v_1 ... v_N, w_1 ... w_N)
output solution
```

5 Experiments

The proposed approached were tested using time-series data records of daily earth temperature sets with $17 \cdot 10^6$ data instances and 500 thousands of instanses, dimensionality of 200 and 20 correspondently. While the first dataset was simulated, the second one is real observation of earth temperature since 1880 available at [17]. The MAFIA algorithm managed to obtain 1630 subspace clusters with average dimensionality 4.6. Since our algorithm was applied it was able to prune similarity search time (using sequental traversal over the clusters) more then 5 times (the maximum number of clusters was bounded to 311) having the same search result as without such adaptation. The other dataset initially contained significantly less subspace clusters found by MAFIA. While it initially contained 41 subspace clusters in 23 subspaces (with initial dimensionality of 20) the optimization process was aimed to bound cluster number by 30. The performance of search was increased near proportionally from 14.018 seconds to 11.95. The both experiments shows no loss in relevance for 100 randomly generated queries.

6 Conclusions

Using subspace clustering feasible approach to build indexing structure though it highly depends on data distribution. Data set can contain a large number of subspace clusters some of them are redundant in scope of indexing. To know how to remove redundancy the corresponding relevance model should be proposed. Our model assumes generic influences of distribution factors on relevance and relevance calculation and the optimization problem states compression ration problem explicitly for a search in a high dimensional spaces. If the data have no features avaliable to proceed some redundancy removal thereby resolving of optimization problem and using subspace clustering as a base for compression and indexing structure can be not feasible at all. The solution of stated optimization problem in a general case is a big challenge, though knowledge of particular relevance calculation functions and affects of clusters geometry can lead to a feasible particular solution.

References

[1] R.E. Bellman. Dynamic programming. Princeton University Press. 1957.

[2] H.P. Kriegel, P. Kroger, A. Zimek. Clustering High-Dimensional Data: A Survey on Subspace Clustering, Pattern-Based Clustering, and Correlation Clustering. ACM Trans. Knowl. Discov. 2009.

[3] L. Parsons, E. Haque, H. Liu Subspace Clustering for High Dimensional Data: A Review. Sigkdd Explorations. 2004.

[4] S. Gunnemann, H. Kremer, D. Lenhard, T. Seidl. Subspace Clustering for Indexing High Dimensional Data: A Main Memory Index based on Local Reductions and Individual Multi-Representations. EDBT. 2011.

[5] F. Nielsen, P. Piro, M. Barlaud. Tailored Bregman Ball Trees for Effective Nearest Neighbors. EDBT. 2008.

[6] S. Berchtold, D.A. Keim, H.P. Kriegel. The X-tree: An Index Structure for High-Dimensional Data VLDB. 1996.

[7] E. Muller, S. Gunnemann, I. Assent, T. Seidl. Evaluating Clustering in Subspace Projections of High Dimensional Data. VLDB. 2009.

[8] N. Cristianini, J. Shawe-Taylor. An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press, Cambridge, UK. 2000.

[9] A. Hinneburg, C. Aggarwal, D.A. Keim. What is the nearest neighbor in high dimensional spaces? VLBD. 2000.

[10] E. Muller, I. Assent, S. Gunnemann. Relevant Subspace Clustering: Mining the Most Interesting Non-Redundant Concepts in High Dimensional Data ICDM 2010.

[11] K. Sequeira, M. Zaki. SCHISM: A new approach for interesting subspace mining. ICDM. 2004.

[12] L. M. Bregman. The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming. USSR Computational Mathematics and Mathematical Physics 7(3). 1967.

[13] J. Ekstrom. Mahalonobis distance beyond normal distribution. UCLA Press. 2005.

[14] S. Goil, H. Nagesh, A. Choudhary. MAFIA: Efficient and Scalable Subspace Clustering for Very Large Data Set. SIGMOD. 1999.

[15] M. Silvano, T. Paolo. Knapsack problems: Algorithms and computer interpretations. Wiley-Interscience. 1990.

[16] G.A. Indyk, P. Motwani. Similarity Search in High Dimensions via Hashing. VLDB. 1999.

[17] Earth surface temperature measurments. http://berkeleyearth.org/data/. 2012.