K-Landmarks: Distributed Dimensionality Reduction for Clustering Quality Maintenance*

Panagis Magdalinos[1], Christos Doulkeridis[1], and Michalis Vazirgiannis[1,2**]

[1]Athens University of Economics and Business
Patision 76, 10434, Athens, Greece
[2]INRIA FUTURS
Parc Club Orsay Universite 91893, France
{pmagdal,cdoulk,mvazirg}@aueb.gr

Abstract. Due to the vast amount and pace of high-dimensional data production and their distribution among network nodes, the fields of Distributed Knowledge Discovery (DKD) and Distributed Dimensionality Reduction (DDR) have emerged as a necessity in many application areas. While a wealth of centralized dimensionality reduction (DR) algorithms is available, only few have been proposed for distributed environments, most of them adaptations of centralized ones. In this paper, we introduce K-Landmarks, a new DDR algorithm, and we evaluate its comparative performance against a set of well known distributed and centralized DR algorithms. We primarily focus on each algorithm's performance in maintaining clustering quality throughout the projection, while retaining low stress values. Our algorithm outperforms most other algorithms, showing its suitability for highly distributed environments.

Keywords: Distributed dimension reduction, distributed knowledge discovery

1 Introduction

Distributed Knowledge Discovery (DKD) has emerged as one of the most challenging tasks in large scale distributed data management. This is partly due to the inapplicability of centralized approaches in current research problems, which is more evident with the advent of new application areas that are inherently distributed, such as sensor networks and peer-to-peer (P2P) systems. The main characteristic of P2P systems is the lack of global knowledge, in the sense that no peer can gather all available data. In large scale P2P networks, data is distributed to peers (in horizontal partitioning manner) making the cost of centralized assembly and subsequent computation of any centralized algorithm prohibitive. On the other hand, globally described data can be of very high dimensionality, while peer local dimensions can be different from each other (vertical partitioning).

* This work is co-funded by the European Social Fund (75%) and National Resources (25%) - (EPEAEK II) - PYTHAGORAS.

** The work of Dr. Vazirgiannis is supported by the EU Marie Curie Intra-European Fellowship.

Dimensionality reduction techniques tackle these problems through the definition of various methods for embedding the data from the initial space R^n to the target space R^k, where $k < n$. These algorithms are extremely useful in various disciplines related to knowledge discovery. The latter becomes a difficult task as the number of dimensions increases, because of two distinct problems: the "empty space phenomenon", and the "curse of dimensionality" [4]. The former denotes the fact that data in high dimensional spaces is sparsely situated, having almost equal distance from one another. The latter refers to the fact that the sample needed to estimate a function of several variables to a given degree of accuracy grows exponentially with the number of variables. A thorough investigation of both problems can be found in [5].

The motivation for our work emerges from the need to apply dimensionality reduction on data distributed in a P2P network. This task is directly applicable in P2P information retrieval applications, where documents are represented as high dimensional points using the vector space model. Distributed dimensionality reduction (DDR) algorithms are then necessary to decrease the representation costs and to reveal potentially interesting or hidden structure in the data.

Towards this objective, we focus on the DKD problem, assuming that the data set is partitioned horizontally (i.e. non overlapping sets of identically structured tuples) and distributed on peers. We identify the following requirements for DDR algorithms: 1) each point's projection should be computed independently from other points, 2) distances between points should be preserved, 3) the algorithm should be fast and linear to the number of projected points, and 4) the algorithm should incur low communication cost (in a distributed context).

In this paper, we present a DDR algorithm, called *K-Landmarks*, aiming to retain clustering quality at the projected space. K-Landmarks first selects an aggregator node that picks k points (henceforth called *landmark points*) from the whole dataset of cardinality d, and projects them from R^n to R^k with FastMap [3]. The projections of the remaining $d - k$ points are computed by requesting the preservation of distances, meaning that each point projected must be at equal distance from all landmark points, both in the original and in the projection space. Our algorithm is not an adaptation of a centralized algorithm; on the contrary it is inherently distributed. Preliminary work describing the initial idea and algorithm was presented in [8]. In this paper, we present additionally the formal description of the algorithm, its geometric interpretation, the proof of convergence and new extensive experiments on various UCI datasets[1], comparing our algorithm's performance with the most promising DDR algorithms in the literature. The rest of the paper is organized as follows: in Section 2 we present a brief overview of the related work. Section 3 describes K-Landmarks, while in Section 4 the conducted experiments are presented. In Section 5 we conclude the paper and sketch future research directions.

[1] `http://www.ics.uci.edu/~mlearn/MLSummary.html`

2 Related Work

In the following presentation we mainly concentrate on distributed approaches and provide only a brief outline of the most prominent centralized algorithms. For the rest of this paper we assume that the goal is to project d data vectors defined in R^n and represented as a matrix X_{dxn} in the R^k subspace.

One of the first dimensionality reduction methods was Multidimensional Scaling (MDS) [4] that is now referenced as classic-MDS. FastMap [3] was proposed as a solution to the high computational complexity of MDS while Landmark MDS (LMDS) [2] addresses the high memory requirements of classic-MDS. The dimensionality reduction techniques widely used in practice, due to their conceptual simplicity, are Principal Components Analysis (PCA) and Singular Value Decomposition (SVD) [4]. Adaptations of the convergence criterion of MDS and PCA have resulted in the definition of Independent Component Analysis and Projection Pursuit algorithm respectively [4].

In the field of DDR we report two promising approaches, the Distributed PCA and the Distributed FastMap. The intuition of distributed PCA (DPCA) [11] is based on the aggregation of a fragmented covariance matrix, which is computed by the equation: $dC = X^T(I - d^{-1}\mathbf{1}\mathbf{1}^T)X$. For each node i possessing d_i data the following statistics are denoted: x_i as the vector of column means, k_i as the number of required principal components from node i, X_i as the local dataset and Λ_i, U_i as the matrices of the k_i largest eigenvalues and corresponding eigenvectors (in descending order) of location i. Furthermore the expression $I - d^{-1}\mathbf{1}\mathbf{1}^T$, is transformed into $(I - V) + (V - d^{-1}\mathbf{1}\mathbf{1}^T)$, where $x = d^{-1}\mathbf{1}^T X$ is the n column means vector, $\mathbf{1}$ is a vector containing ones (1s) and V is a diagonal matrix ($v_{ii} = d_i^{-1}\mathbf{1}\mathbf{1}^T$). DPCA is based on the following decomposition scheme of the covariance matrix (see [11] for details):

$$dC = \sum_{i=1}^{s} U_i \Lambda_i^2 U_i^T + \sum_{i=1}^{s} d_i (x_i - x)(x_i - x)^T \quad (1)$$

Initially, an aggregator node is selected that performs the merging and local k_i values are set. Then the statistics (d_i,k_i,x_i,Λ_i,U_i) are calculated on each network node and communicated to the aggregator. The latter, based on equation (1), calculates the global covariance matrix dC, and transmits its first k eigenvectors together with the global mean value x back to the s network nodes. Finally, each node computes its dataset embedding as follows: $D_i = (X_i - \mathbf{1}x^T)U_k$. Another approach, called Collective PCA, is considered in [6]. However it solves the problem of vertical data partitioning, while we focus on the horizontal case.

In [9] two distributed adaptations of FastMap are proposed, the One-Time Distributed FastMap and the Iterative Distributed FastMap. The former iterates on the data of each node independently, and communicates the generated pivot points to a randomly selected aggregator. Received pivots are used as input to FastMap which generates a global pivot set that is broadcasted and used for the subsequent projection of local datasets. On the other hand, the Iterative Distributed FastMap employs an iteration-by-iteration pivots computation scheme where global pivots are computed on iteration basis according to the find-distant-objects heuristic. Although the two adaptations do not guaran-

	Algorithmic Complexity	Memory Requirements	Addition of a new point	Network load
PCA	$O(n^2d + n^3)$	$O(n^2 + nd)$	$O(kn)$	—
DPCA	$O(n^2d_i + n^3)$	$O(n^2 + nd_i)$	$O(kn)$	$O(nsk)$
FastMap	$O(dk)$	$O((k+n)d + d^2)$	$O(k)$	—
One-Time D.FastMap	$O(d_ik)$ or $O(d_ik + sk^2)$	$O((k+n)d_i + d_i^2)$	$O(k)$	$O(skn + k^2)$
Iterative D.FastMap	$O(d_ik)$ or $O(d_ik + sk^2)$	$O((k+n)d_i + d_i^2)$	$O(k)$	$O(skn + k^2)$
LMDS	$O(kfd + f^2 + f^3)or$ $O(kfd + f^2 + f^3 + k^2d + k^3)$	$O(f(n+k) + f^2)or$ $O(n^2 + f^2)$	$O(kf)$	—
Distributed LMDS	$O(kfd_i + f^2)$ or $O(kfd_i + f^2 + f^3)$	$O(f(n+k))$ or $O(f(n+k) + f^2)$	$O(kf)$	$O(fn + fk)$
PAA	$O(d)$	$O(n)$	$O(1)$	0

Table 1. Assessment of the various algorithms. Number of points (d), initial dimensionality (n), projection space (k), nodes (s), and number of sampled points (f).

tee that the set of pivots selected will be identical with the ones of centralized FastMap, in large data collections they approximate well the original set and provide quality results marginally equal to the original approach.

Piecewise Aggregate Approximation (PAA) [7] is a simple and effective algorithm that can be considered as DDR, which substitutes a set of $\lfloor n/k \rfloor$ variables with their mean value. The only drawback that PAA exhibits is its dependence on the size of the rolling window ($\lfloor n/k \rfloor$). If the latter is big ($k << n$) then sharp changes in data will be lost.

The final algorithm outlined is our proposition concerning the distributed adaptation of LMDS, which we refer to as D-LMDS. LMDS is an algorithm that by construction has been developed to work with only a fraction of the total data. In our variation, the dataset is assumed to be distributed among s network nodes. We initially select an aggregator node that will be assigned the classic MDS computation. Afterwards, each node selects f_i points ($\sum_{i=1}^{s} f_i = f$) from its local dataset and forwards them to the aggregator. The latter performs classic MDS and produces their embedding in the R^k subspace. Then, the aggregator forwards all landmark points and their embeddings to network nodes. Finally, each node applies for each of the local points distance-based triangulation.

Table 1 provides a short comparative assessment of the algorithms presented above. Under certain assumptions all presented algorithms provide potential solutions to the DDR problem. DPCA or D-LMDS will deteriorate quickly and need to recompute the decomposition in the case that many new points are added. In addition, the sampling procedure in the case of D-LMDS will not depict the current state of the network and will have to be recomputed.

Similar disadvantages occur in all other algorithms, except PAA. This is because all are adaptations of already existing centralized approaches that have not been designed for distributed environments. For example, the application of the Iterative Distributed FastMap can only take place in a context where communication between nodes as well as their availability is guaranteed (to ensure the

large scale message exchange that is necessary) otherwise the synchronization of network nodes is practically impossible. Moreover, the addition of new points in an existing projection imposes the re-execution of FastMap, as its credibility lays in the pivots selection. On the other hand, PAA seems to be a viable solution.

It is therefore obvious that a new approach is required, that will combine the salient features of the aforementioned algorithms in terms of network load, algorithmic complexity and quality of results, while being immune to subsequent (after execution) changes in the processed data (i.e. massive addition or deletion of points). Moreover, the algorithm sought has to be inherently distributed and adaptable to potential network failures and topology changes. Finally it has to apply to the full extend of distributed applications, starting from controllable laboratory environment and reaching large scale P2P networks.

3 The K-Landmarks Algorithm

The K-Landmarks algorithm is a novel algorithm for DDR, designed specifically for distributed environments. It conforms to the four requirements stated in the introduction and in addition it is immune to data changes. K-Landmarks capitalizes on the general principles of D-LMDS, while differentiating in the way each step is performed and exhibiting lower complexity and network traffic. Given d resources represented as points in R^n, distributed arbitrarily in a network of s nodes, with each node storing d_i resources, we want to find a projection of the data in R^k, while retaining distances among points and the ability to achieve clustering quality comparable to the one in the original space.

Theorem 1. *A set of k points defined in R^n can be embedded in the R^k subspace without loss of distance information (zero stress projection[2]).*

Proof. The set of processed points define matrix X_{kxn}. The latter can be projected in R^k through the transformation $X'_{kxk} = X_{kxn}Q^T_{kxn}$ where the Q rows are the singular vectors of X. The relationship between the inner products matrix of the projected data and the inner products matrix of the original data is given by the following computations:

$$C' = {X'}_{kxk}{X'}^T_{kxk} = X_{kxn}Q^T_{kxn}(X_{kxn}Q^T_{kxn})^T = X_{kxn}X^T_{kxn} = C$$

Moreover, each cell (i,j) of C is populated by the value $x_i x_j^T$ and based on equality $C = C'$ we conclude that $x_i x_j^T = x'_i {x'_j}^T$. Then the new distance between points x'_i,x'_j is:

$$d(x'_i, x'_j) = \sqrt{\sum_{p=1}^{k}(x'_{ip} - x'_{jp})^2} = \sqrt{\sum_{p=1}^{k}({x'}^2_{ip} + {x'}^2_{jp} - 2x'_{ip}x'_{jp})}$$

$$= \sqrt{{x'}_i{x'}_i^T + {x'}_j{x'}_j^T - 2{x'}_i{x'}_j^T} = \sqrt{x_i x_i^T + x_j x_j^T - 2x_i x_j^T} = d(x_i, x_j)$$

Hence, the projection of k points from R^n to R^k can be defined without loss of distance information and consequently zero stress.

[2] Stress measure, defined as $\sum(d_{ij} - d'_{ij})^2/d^2_{ij}$, where d_{ij} is the distance of points i,j in the original space and d'_{ij} their distance in the projection space, signifies the quality of the projection in terms of distances preservation.

Algorithm 1 K-Landmarks algorithm.

1: **Input:** Projection dimensionality (k), number of landmark points from node i (k_i), local dataset defined in R^n
2: **Output:** local dataset defined in R^k
3:
4: **if** node is aggregator **then**
5: Select k_i points from local dataset
6: Create landmark set, LS=$\varnothing$
7: Create projected landmark set, PLS=$\varnothing$
8: **for** $i = 1$ **to** s **do**
9: Receive k_i landmarks from node i
10: LS=$LS \cup k_i$
11: **end for**
12: PLS = FastMap(k, LS)
13: Communicate PLS, LS to all nodes
14: **else**
15: Select k_i points from local dataset
16: Send points to aggregator
17: Receive LS, PLS
18: **end if**
19: **for** $i = 1$ **to** all local points x **do**
20: Solve $\|x_i^{(k)} - PLS_i^{(k)}\| = \|x_i^{(n)} - LS_i^{(n)}\|$ for i=1 to k
21: **end for**
22: **return** $x_i^{(k)}$ //the projection of $x_i^{(n)}$ in R^k

Algorithm 1 is the formal description of K-Landmarks. In the first step, an aggregator node is selected. The latter uses k landmark points (LS) sampled from the network (line 9) and projects them to R^k with FastMap (line 12). The original set of landmark points and the generated mapping (PLS) are forwarded to all nodes (line 13), which in turn project local points independently (line 20). The successful projection of a point in R^k maintains its distances from the landmark points both in the original and in the projection space.

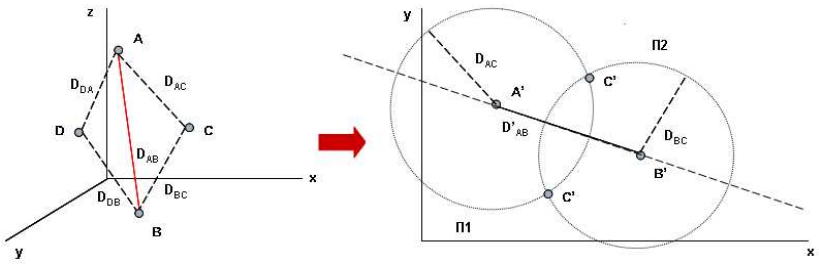

Fig. 1. Geometrical interpretation of the K-Landmarks algorithm. Projection of point C from R^3 (left) to R^2 (right). A, B are the landmark points.

The equation: $\|x_i^{(k)} - PLS_i^{(k)}\| = \|x_i^{(n)} - LS_i^{(n)}\|$ represents a hypersphere centered at $PLS_i^{(k)}$ with radius $||x_i^{(n)} - LS_i^{(n)}||$. The algorithm searches the common trace of all k hyperspheres, which is the projection of point x_i in the embedding space. The result is obtained by solving the above system of non-linear equations with the Newton method. An example of the algorithm is depicted in Fig. 1. Note that the hyperspheres defined by the algorithm expose two common traces, which are symmetric to the line defined by points $A'B'$. This is due to the fact that the coordinates of each point result from a square root computation. We choose to retain only positive results and therefore the implementation of our algorithm discards the second solution. In any case, one can select either the projection laying on plane $\Pi 1$ or $\Pi 2$ without affecting the final result, as long as the same plane selection algorithm is employed for point D.

Theorem 2. *For any non linear system of equations defined by K-Landmarks, the Newton method produces a solution if the triangular inequality is sustained in the original space.*

Proof. For any point C of the initial space and any pair of landmark points A and B, a triangle ABC is defined. Without loss of generality, we assume that $\|\overrightarrow{CB}\| \leq \|\overrightarrow{CA}\|$ and based on the triangular inequality we derive:

$$\|\overrightarrow{CA}\| - \|\overrightarrow{CB}\| \leq \|\overrightarrow{AB}\| \leq \|\overrightarrow{CA}\| + \|\overrightarrow{CB}\| \quad (1)$$

The system defined for the projection is the following:

$$\|\overrightarrow{CA}\| = \|\overrightarrow{C'A'}\| \text{ and } \|\overrightarrow{CB}\| = \|\overrightarrow{C'B'}\|$$

This system has no solution if there exists no common trace between the aforementioned hyperspheres. This is translated to: $\|\overrightarrow{A'B'}\| < \|\overrightarrow{C'A'}\| - \|\overrightarrow{C'B'}\|$ or $\|\overrightarrow{A'B'}\| > \|\overrightarrow{C'A'}\| + \|\overrightarrow{C'B'}\|$ and equally:

$$\|\overrightarrow{A'B'}\| > \|\overrightarrow{CA}\| + \|\overrightarrow{CB}\| \quad (2) \text{ or } \|\overrightarrow{A'B'}\| < \|\overrightarrow{CA}\| - \|\overrightarrow{CB}\| \quad (3)$$

However, based on [10] and Theorem 1 we obtain a zero stress projection from FastMap, thus deriving:

$$\|\overrightarrow{A'B'}\| = \|\overrightarrow{AB}\| (4)$$

Consequently, based on (1), (4) we conclude that equations (2), (3) are never true, meaning that the system in question always has a solution (there always exists a projection) provided that the triangular inequality is sustained in the original space.

One thing that has not yet been discussed is the selection of initial points. K-Landmarks employs three different initialization techniques namely: random, MaxMin and MaxDist. In random selection each node selects k_i points from its dataset randomly. The MaxMin heuristic [2] enforces the selection of points that maximize the minimum distance from any of the already selected landmark points while the MaxDist selects the furthest from the existing landmark points. Both heuristics select their first point randomly and are applied on local datasets for the retrieval of the k_i points requested by the algorithm.

To sum up, the proposed algorithm differs significantly from other widely employed DDR approaches since it achieves the projection of the vast majority of points independently from the rest, implying that only the (few) landmark

	Algorithmic Complexity	Memory Requirements	Addition of a new point	Network load
K-Landmarks	$((d_i - k_i)k^3/3)$	$O(kn + k^2)$	$O(k^3/3)$	$O(nk + k^2)$

Table 2. K-Landmarks evaluation matrix.

points' projection will be done in a centralized manner. Moreover, the projection remains unaffected by subsequent additions of data points. This is due to the fact that any new point will be mapped analogously close or far from the landmark points depending on its distance from the latter in the original space. Consequently, no re-computation of the projection is needed in order to guarantee projection quality preservation. Furthermore, the minimization criterion employed by the algorithm ($\sum_{|LS|} |distance_{orig} - distance_{new}|$) is applied to each point independently, contrary to the widely employed stress that is applied to the whole dataset. Finally, the network load imposed (see Table 2) is lower than the load of other algorithms.

Apart from the above, the proposed algorithm is inherently distributed, in contrast to the other distributed algorithms described in Section 2. One can imagine its usage in a P2P environment. These networks exhibit certain intrinsic peculiarities, such as instability, bandwidth restrictions, etc. If K-Landmarks is used, the sampling procedure will be carried out once in the lifetime of the network and the result will be forwarded to all nodes entering the network at any time. The added value of the approach is apparent, as its immunity to additions saves both local and network resources.

4 Experiments

In this section, we study the comparative performance of K-Landmarks on various datasets (Table 3) from the UCI Machine Learning Repository. We highlight the use of datasets both of higher dimensionality (up to 617 dimensions) and cardinality (up to 2000 objects) compared to other relevant research papers [3, 9]. Our goal is to achieve results of quality close to well-known centralized and distributed algorithms.

4.1 Experimental Setup

The experimental scenario involves clustering various data sets before and after the application of a DDR algorithm. The quality metrics we use are the stress value and the clustering quality maintenance. Stress evaluates the quality of the projection in terms of distances' preservation, while the second measure enables us to observe how DDR affects the clustering quality. We capitalize on the well known clustering quality index F-Measure [1] and define clustering quality preservation as the ratio of the F-measure values before and after the DDR process, i.e. $\frac{F-Measure(R^k)}{F-Measure(R^n)}$. The clustering algorithm employed is K-Means [1].

The experiments also allow measuring the effect of changing the projection dimensionality on the stress and F-measure values. Each dataset was evaluated

Dataset	Objects	Dimensions	Classes	Description
Ionosphere	351	34	2	Radar observations
Isolet5	1559	617	26	Letters of the alphabet
Musk	476	166	2	Molecules descriptions
P.I.Diabetes	768	8	2	Medical observations
Segmentation	2000	19	7	Outdoor images segments
Synthetic control	600	60	6	Randomly generated data

Table 3. Datasets used in the experiments.

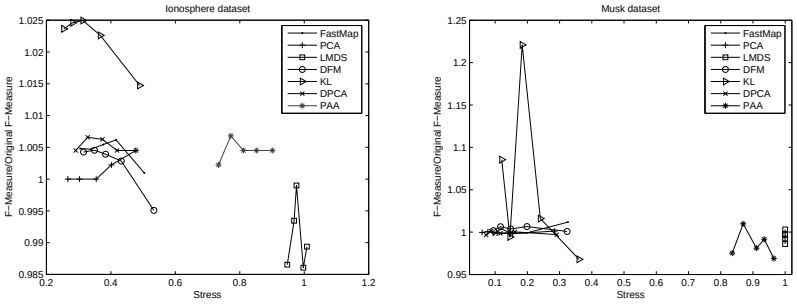

Fig. 2. Clustering Quality vs. stress for the "Ionosphere" and "Musk" datasets.

with 5 different projection dimensions (5 points for each algorithm on the charts), which were defined as a fraction of the initial dimensionality of the dataset. Each time the projection dimensionality is increased by 2% of the initial dimensions.

Assume a network of s nodes, with d data vectors distributed evenly among them. The vectors are defined in R^n and projected in R^k. The algorithms we use in comparison to K-Landmarks are PCA, FastMap, Distributed FastMap, Distributed PCA, Distributed LMDS and PAA. The notation employed in the diagrams is the following: KL refers to K-Landmarks, LMDS to the D-LMDS and DFM to distributed FastMap. DPCA uses the k principal eigenvectors generated by each peer separately, while K-Landmarks and LMDS randomly select from each node $\lceil k/s \rceil$ and $\lceil k+1/s \rceil$ points respectively. The reason for the selection of $\lceil k+1/s \rceil$ points for LMDS lays in the original publication [2], where it is advised to choose at least $k+1$ landmarks. Also, the Newton method employed by K-Landmarks is initialized with the k first coordinates of each processed vector in the original space. All experiments have been carried out on a commodity 2.4GHz Pentium IV machine with 768MB of RAM. The results are mean values of 50 executions and each setup simulates a network of $s = 20$ nodes.

4.2 Results

In the first set of experiments, we measure for each algorithm and target dimensionality value the F-measure quality preservation (y-axis) versus the respective stress value of the projection (x-axis). Our aim is to identify the DDR algorithms that exhibit low stress while maintaining clustering quality.

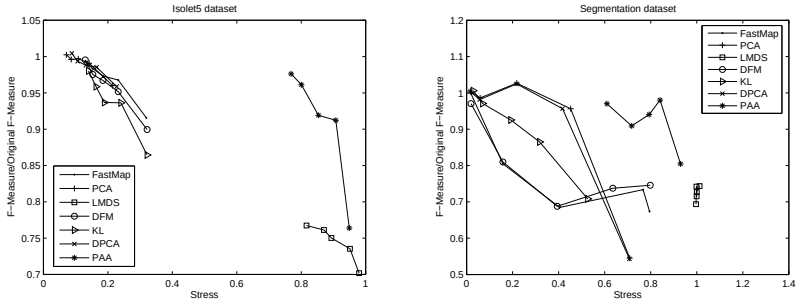

Fig. 3. Clustering Quality vs. stress for "Isolet5" and "image segmentation" datasets.

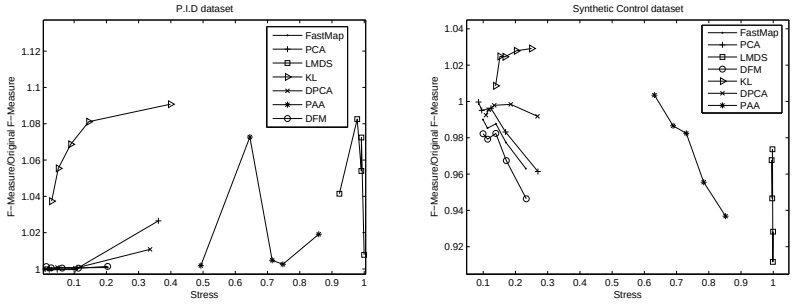

Fig. 4. Results from experiments on "P.I.diabetes" and "synthetic control" dataset.

In Fig. 2 (left) experiments on the "Ionosphere" dataset show that K-Landmarks outperforms all its competitors, even the centralized approaches such as PCA and FastMap. Most of the K-Landmarks measurements reflect extremely low stress and high clustering quality maintenance, close to or higher than 100% (i.e. centralized clustering quality). This is due to the empty space phenomenon and the curse of dimensionality. LMDS exhibits the worst behavior proving rather unstable. Similar behavior is observed using the "musk" dataset (Fig. 2 - right).

Experiments on the "Isolet5" and "image segmentation" datasets (Fig. 3) provide better insight. All algorithms except from PAA and LMDS performed similarly, with PCA performing better. PAA also exhibits satisfactory results in the clustering quality maintenance aspect, but higher stress than the rest. Finally, LMDS proves to be rather unsatisfactory in both quality axes.

The experiments on "pima indians diabetes" and "synthetic control" (Fig. 4) gave similar results to the "ionosphere" dataset. K-Landmarks achieves the best overall performance, showing both low stress and high clustering quality maintenance. PAA and LMDS achieve marginally equal clustering quality results, compared to the rest of the approaches, but with higher stress values. However PAA proves to be better than LMDS, when both measures are considered.

In Fig. 5, we measure F-measure maintenance and stress for different projection dimensionality values (k) for the "image segmentation" dataset. The aim is to study the effect of k on these two measures. Stress values decrease

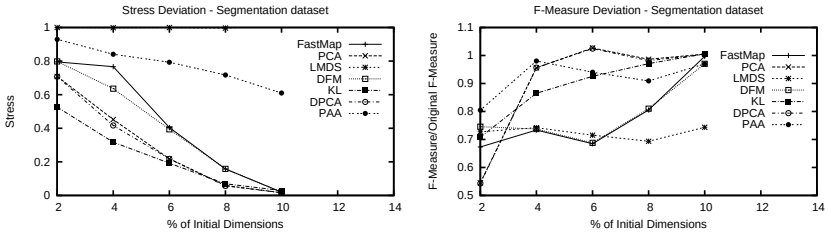

Fig. 5. Stress and F-Measure deviation in the "image segmentation" dataset.

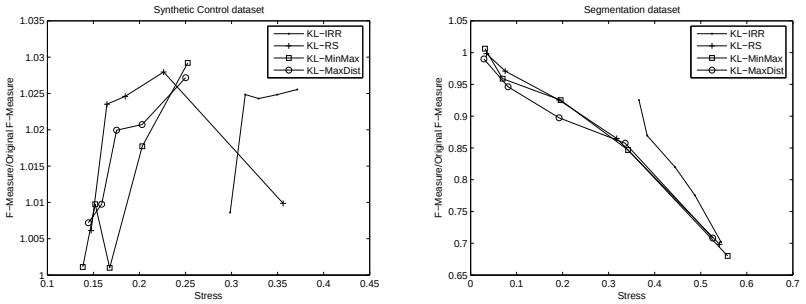

Fig. 6. K-Landmarks immunity to the addition of data.

monotonously as k increases (Fig. 5 - left), because the ability to express distances between data increases too. The same general tendency appears in the case of clustering quality maintenance (Fig. 5 - right). For the majority of the algorithms, clustering quality maintenance ameliorates with k. The conclusion is that DDR algorithms perform better with increasing projection dimensionality. We clarify that the average variance of all measured values in the 50 executions for all datasets is in the order of 10^{-4}, showing the stability of our algorithm.

In the last set of experiments, we demonstrate the robustness of K-Landmarks to retain clustering quality, with regards to new data points added to the dataset. In Fig. 6, we study the algorithm's different initialization setups. The new setup depicted is KL-IRR, in which the algorithm was initiated with k points that were randomly generated and did not belong to the dataset (we remind here the 3 other initialization setups: random, MaxMin and MaxDist).

The datasets are subsequently added to the projection. In the "synthetic control" dataset, we added 600 points, while for the "image segmentation" dataset we added 2000. In both cases points were inserted after the algorithm's execution with the initial k landmarks. The obtained results exhibit an aggravation tendency in the stress measure, yet the clustering quality remains almost the same. These experiments show K-Landmarks is barely affected by the massive insertion of data points and can guarantee clustering quality maintenance with a slight loss in the preservation of the original distances. Another conclusion, clearly depicted on the right diagram, is that the results obtained by our algorithm are not affected by the way initial points are selected. Therefore, based on our experiments, we suggest the random selection initialization scheme (KL-RS).

5 Conclusions and Future Work

In this paper we proposed a new effective, inherently distributed, algorithm for DDR, K-Landmarks. We compared experimentally our approach to well known DDR approaches with regards to stress and clustering quality preservation. The results show that K-Landmarks is a robust algorithm outperforming existing DDR approaches. Moreover, it is comparable and sometimes even superior to centralized methods. Clustering quality is retained with dimensionality reduction, in spite of the small loss in distance preservation. Our future work will mainly focus on the evaluation of K-Landmarks with text data and its combination with distributed clustering approaches.

References

1. S. Chakrabarti. *Mining the Web - Discovering Knowledge from Hypertext Data*. Morgan Kaufmann Publishers, 2003.
2. V. de Silva and J. B. Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Stanford Mathematics, 2004.
3. C. Faloutsos and D. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of SIGMOD'95*, 1995.
4. I. K. Fodor. A survey of dimension reduction techniques. Technical report, LLNL, UCRL-ID-148494, 2002.
5. A. Hinneburg, C. Aggarwal, and D. Keim. What is nearest neighbor in high dimensional spaces? In *Proceedings of VLDB'00*, 2000.
6. H. Kargupta, W. Huang, K. Sivakumar, and E. L. Johnson. Distributed clustering using collective principal component analysis. *Knowledge and Information Systems*, 3(4):422–448, 2001.
7. E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, 2001.
8. P. Magdalinos, C. Doulkeridis, and M. Vazirgiannis. A novel effective distributed dimensionality reduction algorithm. In *Proceedings of FSDM'06*, 2006.
9. F. N.Abu-Khzam, N. Samatova, G. Ostrouchov, M. A.Langston, and A. Geist. Distributed dimension reduction algorithms for widely dispersed data. In *Int. Conference on Parallel and Distributed Computing Systems (PDCS'02)*, 2002.
10. J. C. Platt. FastMap, MetricMap and Landmark MDS are all Nyström algorithms. In *10th International Workshop on Artificial Intelligence and Statistics*, 2005.
11. Y. Qu, G. Ostrouchov, N. Samatova, and A. Geist. Principal component analysis for dimension reduction in massive distributed data sets. In *Proceedings of the 5th International Workshop on High Performance Data Mining*, 2002.