Distributed Knowledge Discovery with Non Linear Dimensionality Reduction

Panagis Magdalinos, Michalis Vazirgiannis and Dialecti Valsamou

Athens University of Economics and Business Athens, Greece
pmagdal@aueb.gr,mvazirg@aueb.gr,dvalsamou@gmail.com

Abstract. Data mining tasks results are usually improved by reducing the dimensionality of data. This improvement however is achieved harder in the case that data lay on a non linear manifold and are distributed across network nodes. Although numerous algorithms for distributed dimensionality reduction have been proposed, all assume that data reside in a linear space. In order to address the non-linear case, we introduce D-Isomap, a novel distributed non linear dimensionality reduction algorithm, particularly applicable in large scale, structured peer-to-peer networks. Apart from unfolding a non linear manifold, our algorithm is capable of approximate reconstruction of the global dataset at peer level a very attractive feature for distributed data mining problems. We extensively evaluate its performance through experiments on both artificial and real world datasets. The obtained results show the suitability and viability of our approach for knowledge discovery in distributed environments.

Keywords: distributed non linear dimensionality reduction, distributed data mining

1 Introduction

During the last decade, the evolution of the internet as well as the emergence of novel applications, such as peer-to-peer (P2P) systems, has led to an unprecedented information explosion. Information is distributed among network nodes, making the cost of centralizing and processing data prohibitive. Consequently, distributed data mining (DDM) has emerged as a highly challenging task.

Dimensionality reduction (DR) is an important step of data mining as high dimensional data lead to the degradation of query processing performance, a phenomenon known as the curse of dimensionality [8]. Thus typical tasks, such as clustering or classification, become ineffective. DR is then required in order to decrease the number of dimensions and reveal potentially interesting structures in data. With the advent of DDM, distributed dimensionality reduction (DDR) has emerged as a necessity in many applications.

A prominent such application is knowledge discovery from text collections distributed in a P2P network. Latest theoretical and experimental evidence point out that documents lay on a non linear high dimensional manifold ([5], [3]).

Consequently, non linear dimensionality reduction (NLDR) is necessary in order to recover the low dimensional structure. Although numerous DDR algorithms have been proposed, all assume that data lay on a linear space. Thus there is the need for definition of distributed NLDR techniques.

To this end, we introduce *Distributed Isomap* (D-Isomap). D-Isomap corresponds to the decentralized version of the well known NLDR algorithm Isomap [18]. D-Isomap has been specifically designed and tuned in order to be applicable in large scale, structured P2P networks. We evaluate its performance and assess its viability and suitability for distributed environments through extensive experiments on artificial and real world datasets.

The contribution of this work is manifold. In section 2, we provide a review of the Isomap and DDR families of algorithms. In section 3 we introduce D-Isomap, a distributed NLDR algorithm which to the best of our knowledge is the first of its genre. Furthermore, we provide a cost model that assesses the computational and network resources required for the embedding of a dataset in a low dimensional space with D-Isomap. Finally, in section 4, we demonstrate the non linear nature of our approach through extensive experiments on well known non linear manifolds and justify its applicability in mining document collections distributed in P2P networks.

2 Related Work

DR algorithms are usually classified with respect to the way they manage data ([16]). *Linear algorithms* assume that high dimensional data lay on a linear or approximately linear manifold of significantly lower dimensionality. On the other hand, *non linear methods* assume that such linearity does not exist and operate on small fractions of the high dimensional manifold that can be perceived as locally linear. Due to space limitations, in the remaining of this section, we focus on the Isomap algorithm and its variations while in the end we provide a short overview of prominent DDR methods and motivate the need for a distributed NLDR approach.

Isomap [18] is a non linear technique that operates on points' pairwise geodesic distances. Isomap first constructs a nearest neighbor (NN) graph, where each point is represented as a node having edges to its k NN points. Edges are weighted according to the Euclidean distance of the points connecting. Global pairwise distances are calculated based on the shortest paths between all points (geodesic distances). The low dimensional mapping is derived by applying classic metric multidimensional scaling [19](MDS) on the geodesic distance matrix.

Isomap deficiencies to deal with curved manifolds or project large datasets gave rise to extensions such as C-Isomap and L-Isomap [16]. C-Isomap employs a different edge weighting scheme taking into account the mean distance of each point from its NNs. L-Isomap on the other hand attempts to address the excessive memory requirements of MDS by introducing Landmark MDS (LMDS). LMDS applies MDS on a set of sampled points and uses triangulation for the projection of the remaining dataset. Another problem of Isomap is the defini-

tion of non connected NN graphs. In such cases the algorithm operates on the largest connected component and discards the rest. A solution is provided by Incremental Isomap [20] (I-Isomap) which guarantees the construction of a fully connected graph and is able to update the embedding when data is inserted or deleted.

DDR algorithms assume data distributed across a set of nodes and the existence of some kind of network organization scheme. The simplest case, where organization exists by construction, are structured P2P networks. In such networks, a protocol (usually based on distributed hast tables - DHT) ensures that any peer can efficiently route a search to a peer that has a specific file. Examples include Chord [17] and CAN [15]. In unstructured networks, the organization may be induced by means of physical topology (i.e. a router) or by means of self-organization [11]. In both cases however, a node undertakes all computations that have to be done centrally. The most prominent approaches in the area are adaptations of PCA ([9], [13], [14]). Two distributed alternatives of Fastmap [1] have also been proposed, but their application relies heavily on the synchronization of the network elements thus can only be applied in controllable laboratory environments. Recently, K-Landmarks [11] has appeared as a promising solution for DDR in unstructured P2P networks.

Unfortunately, all these methods are linear, in the sense that they assume that data lay on a linear or approximately linear low dimensional manifold. However, latest results point out that data usually lay on a non linear manifold ([5], [3]) thus linear methods fail to provide adequate results. Consequently, there is an apparent need for decentralized NLDR techniques. To the best of our knowledge, D-Isomap is the first attempt towards this direction.

3 Distributed Non Linear Dimensionality Reduction

D-Isomap capitalizes on the basic steps of Isomap and applies them in a network context, managing to successfully retrieve the underlying manifold while exhibiting tolerable network cost and computational complexity. In the rest of this section we present in details each step of the algorithm and review the cost induced by its application in a structured P2P network. Throughout the analysis we assume that N points, residing in R^d, are distributed in a P2P network of M peers. Each peer stores N_i points ($\sum_{i=1}^{M} N_i = N$). The objective is to recover the manifold residing in R^n using a graph defined by the k NNs of each point.

3.1 Data Indexing and Nearest Neighbours Retrieval

The first step of Isomap necessitates the definition of a kNN graph for each point. The latter, although applied in a distributed environment, should yield results of accuracy approximating that of a centralized approach. This, in conjunction with our initial goal for low network cost, highlights the need for a structured, DHT based, P2P network like Chord. Chord is a P2P lookup protocol where

peer identifiers are arranged in a circle. Each node has a successor and a predecessor. The successor of a peer is the next node in the identifier circle when moving clockwise. On the other hand, the predecessor, is the next peer in the identifier circle when moving counter-clockwise. A message in Chord may require to traverse $O(logM)$ hops before reaching its destination.

In order to enable rapid lookup of points similar to each peer's local data we consider locality sensitive hashing [2] (LSH) that hashes similar points to the same bucket with high probability. LSH defines L hash tables, each related with a mapping function $g_i, i = 1...L$. Each g_i is defined by f hash functions $h_j(), j = 1...f$, randomly chosen from the same family of LSH functions $\mathcal{H}$. Every $h_{i,j}()$ maximizes the collision probability for data points that are close to each other in the original space. Since we measure similarity based on the euclidean distance, we use $h_{r,b}(x) = \lfloor \frac{rx+b}{w} \rfloor$, where x is the point under process, r is a random vector which coordinates are defined by the Gaussian distribution and w, b random numbers with the property $b \in [0, w)$.

The mapping of hash identifiers to peer identifiers is accomplished by employing a similarity preserving transformation that depicts a vector from R^f in R^1 [6]. For a given vector x, LSH produces an f-dimensional vector; the l_1 norm of this vector defines a similarity preserving mapping to R^1. Additionally, it can be proved that the obtained l_1 values are generated from the normal distribution $\mathcal{N}(\frac{f}{2}, \frac{f}{w}\mu_{l(x_i)})$, where $\mu_{l(x_i)}$ is the mean value of all points' Euclidean norm. Consequently, each hash value v is indexed by peer $p_i = (\frac{l_1(v)-\mu_{l_1}+2\sigma_{l_1}}{4*\sigma_{l_1}} * M) mod M$.

The simplest way to retrieve the kNNs of a point p is to aggregate from all hash tables the points hashed in the same bucket as p. Afterwards, retrieve the actual points, calculate their distances from p and retain the kNNs. In order to reduce the required messages we halt the procedure as soon as ck points have been retrieved (in our experiments we set $c = 5$). Additionally, for each point, we define a range $bound_p$ that enables a queried peer to return only a subset of the points that indexes using Theorem 1. We use as bound the mean distance that a point exhibits from the points of its local dataset.

Theorem 1. *Given f hash functions $h_i = \lfloor \frac{r_i x^T + b_i}{w} \rfloor$ where r_i is an $1xn$ random vector, $w \in N$, $b_i \in [0, w), i = 1...f$, the difference δ of the l_1 norms of the projections x^f,y^f of two points $x, y \in R^n$ is upper bounded by $\frac{\| \sum_{i=1}^{f} |r_i| \| \|x-y\|}{w}$ where $\|x - y\|$ is the points' euclidean distace.*

Proof: Since $|a| - |b| \leq |a - b| \leq |a + b| \leq |a| + |b|$ we derive $l_1(x^f) \leq \frac{1}{w} \sum_{i=1}^{f} (|r_i x^T| + |b_i|) \leq \frac{1}{w} (\sum_{i=1}^{f} |r_i|) |x|^T + \frac{1}{w} \sum_{i=1}^{f} |b_i|$. We assume $A = (\sum_{i=1}^{f} |r_i|)$ and employ the inequality in order to derive $\delta = |l_1(x^f) - l_1(y^f)| \leq |\frac{1}{w} A(|x|^T - |y|^T)| \leq |\frac{1}{w} A||(|x|^T - |y|^T)| \leq |\frac{1}{w} A||(x^T - y^T)| = |\frac{1}{w} A||x - y|^T = \frac{1}{w} A|x - y|^T$ since w and $|r_i|$ are positive. In parallel, for any two vectors a, b we know that $\|ab^T\| \leq \|a\| \|b\|$. Consequently, $\delta \leq \frac{1}{w} \|A|x - y|^T\| \leq \frac{1}{w} \|A\| \|x - y\|$. Based on the latter we obtain $\delta \leq \frac{\|A\| \|x-y\|}{w}$ $\square$

The first part of the procedure is presented in Algorithm 1. At first, Each peer, hashes its local points and transmits the derived l_1 values to the corresponding peers. This procedure yields a network cost of $O(N_i L)$ messages per

Algorithm 1 Data indexing and kNN retrieval

Input: Local dataset in $R^d(D)$, ♯ peers (M), ♯ hash tables L, hash functions g, ♯ NNs (k), peer identifier (id), parameter c (c)
Output: The local neighbourhood graph of peer id (X)
for $i = 1$ to N_{id}, $j = 1$ to L **do**
  $hash_j(p_i) = g_j(p_i)$ - where p_i is the i-th point of D
  $peer_{ind} = (\frac{l_1(hash_j(p_i)) - \mu_{l_1} + 2\sigma_{l_1}}{4 * \sigma_{l_1}} * M) mod M$
  Send message $(l_1(hash_j(p_i)), id)$ to $peer_{ind}$ and store $(peer_{ind}, p_i, j)$
end for
if peer is creating its local NN graph **then**
  for $i = 1$ to N_{id}, $j = 1$ to L **do**
    Send message $(id, hash_j(p_i), bound_{p_i})$ to $(peer_{ind}, p_i, j)$
    Wait for response message $(host, p_{ind}, l_1(p_{ind}))$
    If total number of received points is over ck, request points from host nodes, sort them according to their true distance from p_i and retain the k NNs of p_i
  end for
else
  Retrieve message $(id, hash_j(p_i), bound_{p_i})$ from $peer_{id}$
  Scan local index and retrieve relevant points according to Theorem 1
  Forward retrieved points' pointers to querying node
end if

peer or a total of $O(NL)$ messages. The process of recovering the kNNs of a point requires ck messages thus is upper bounded by $O(ckN)$. Time requirements on peer level are $O(N_iLf + N_iklogk)$ induced by the hashing and ranking procedure. Finally memory requirements are $O(N_ik)$, due to retaining the NNs of each point.

3.2 Distributed Geodesic Distances Definition

Each point p has successfully recovered the location of its kNNs and created the corresponding graph G_p. Now, each peer should identify the shortest paths (SP) from its local points to all points in the dataset using only local information $(\cup_{j=1}^{N_i} G_j)$. For this, we will use distributed SP computation techniques, extensively used for network routing purposes. A well known algorithm is the Distance Vector Routing (DVR) or Distributed Bellman-Ford (DBF) which is core part of many internet routing protocols such as RIP, BGP, ISO and IDRP [10].

For every point p, its host peer maintains a distance vector $DIST[p]$ that stores the distances from p to all points of the dataset. Initially, only the cells corresponding to p's NNs are populated while the rest are set to ∞. The procedure is initiated by sending the $DIST[p]$ to all peers maintaining p's NNs. Each receiver evaluates its current SPs to points appearing in $DIST[p]$ and if a new SP is identified updates distance vector $DIST[q]$ -where q is a point in p's set of NNs- and goes back to the sending step. The process is repeated until no update takes place, thus SPs have been computed.

Algorithm 2 Definition of geodesic distances

Input: peer id (id), local dataset (D), distances from NNs ($DIST$), time (t)
Output: SP distances of local points to the rest of the dataset ($DIST$)
for $i = 1$ to N_i **do**
  Send $(DIST[i], i, id)$ to peers hosting NNs of p_i
end for
while Time to receive a message $< t$ **do**
  Receive message $(DIST[n], p, peer_j)$ - distances of p_p's NN n residing in $peer_j$
  if $d(p,j) > d(p,n) + d(n,j)$ for any $j \in DIST[n]$ **then**
    $DIST[p][j] = d(p,n) + d(n,j)$
  end if
  if update took place **then**
    Send $(DIST[p], p, id)$ to peers hosting NNs of p_p
  end if
end while
Substitute ∞ with $5 * max(DIST)$

The algorithm is asynchronous and does not have an explicit termination criterion. However it is self-terminating ([10]) since message transmissions will halt as soon as no updates take place. Consequently, each peer, after waiting time t to receive a message considers the process finalized. In order to guarantee the creation of a connected SP graph we substitute in the end all remaining ∞ values with five times the largest local geodesic distance. Based on the description, the algorithm is presented in Algorithm 2.

DBF resembles a progressive range search where each point p learns in loop i distance information from points that are i edges away in the graph. Therefore, DBF execution requires $O(kDN^2)$ messages, where D is the diameter (the longest path) of the network. In our case, D depicts the number of peers that maintain parts of a single shortest path (from any point p to any point q), thus $D = M$ and the network cost is upper bounded by $O(kMN^2)$. Although the latter is relatively large, efficient implementation of the procedure can significantly reduce the total overhead. This can be accomplished by transmitting only updated SP information in the form $(p_{source}, p_{destination}, dist)$. Memory requirements are low, $O(N_i N)$ due to retaining the local points' distances in main memory throughout computations. Finally, time requirements are $O(M)$.

3.3 Approximating the Multidimensional Scaling

At this point, each peer has retrieved the SP distances of its own points to the rest of the dataset. The final step is to apply eigendecomposition on the global distance matrix, which is essentially the MDS step of Isomap. Although several methods for parallel computation of this procedure exist (i.e. [7]), they exhibit excessive network requirements, making their application infeasible. An approach that yields zero messages yet rather satisfactory results is the approximation of the global dataset at peer level with landmark based DR techniques. Instead of trying to map all data simultaneously to the new space, landmark-based DR

Algorithm 3 Distributed Isomap

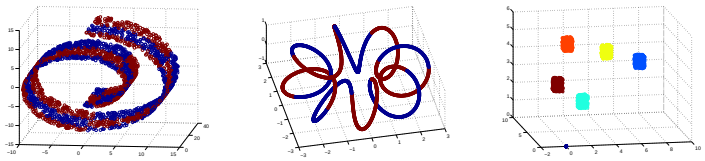
Input: Local dataset in $R^d(D)$, ♯ peers (M), ♯ hash tables (L), hash functions (g), ♯ NNs (k), peer identifier (p), lower dimensionality (n), parameter c (c), aggregator peer (p_a), ♯ landmarks (a),time (t)
Output: The embedding of the global dataset in R^n (GL)
Set X = Define local neighbourhoods(D, M, L, g, k, p, c) - Algorithm 1
Set Y = Distributed Bellman-Ford(p, D, X, t) - Algorithm 2
LAN = local points (set of landmark points)
if $N_i < a$ **then**
  if $p <> p_a$ **then**
    Randomly select a subset of local points and transmit them to p_a
    Retrieve global landmarks LAN from p_a
  else
    Receive LAN_i from peer i
    Define LAN by selecting a landmarks and transmit it to all peers
  end if
end if
GL = LMDS(Y,LAN,n) or FEDRA (Y,LAN,n)

algorithms use a small fraction of points and project them in the new space. Based on the assumption that these points remain fixed (landmarks in the new space), the rest of the dataset is projected using distance preservation techniques.

Two approaches directly applicable in our case are LMDS [16] and FEDRA [12]. LMDS operates by selecting a set of a landmark points, with the constraint $a > n$ and projects them in the new space with the use of MDS. Afterwards, a distance-based triangulation procedure, which uses as input distances to already embedded landmark points, determines the projection of the remaining points. FEDRA behaves similarly to LMDS however selects exactly n landmarks. LMDS requires $O(naN_i + a^3)$ time and $O(aN_i)$ space while FEDRA requires $O(nN_i)$ and $O(n^2)$ respectively. The salient characteristic of this step is that by using as landmarks the local points of a peer we manage to kill two birds with one stone. On one hand we embed the local dataset in R^n while simultaneously each peer derives an approximation of the global dataset. Consequently, each node is able to access global knowledge locally.

A potential failure may appear if the landmarks in a peer are not sufficient for the embedding to take place (i.e. for LMDS $a < n$). In such case, a network wide landmark selection process can be applied. The simplest way, is to assign a peer with the role of aggregator and then all peers transmit at least $\lceil \frac{n}{M} \rceil$ local points. Landmarks are randomly selected from the accumulated points and transmitted back to all nodes thus inducing $O(nNM)$ network load. Based on the previous analysis we derive D-Isomap and present it in Algorithm 3. The application of D-Isomap requires $O(N_iLf + N_iklogk)$ time and $O(n^2 + N_i(N + k))$ space per peer and a total of $O(NL + kMN^2)$ messages from all peers.

A final issue is related to the addition or deletion of data. Upon the arrival of a point, we apply Algorithm 1 and derive its kNNs. Afterwards, the SPs



(a) Swiss Roll (b) Helix (c) 3D Clusters

Fig. 1. The Swiss Roll, Helix and 3D Clusters datasets

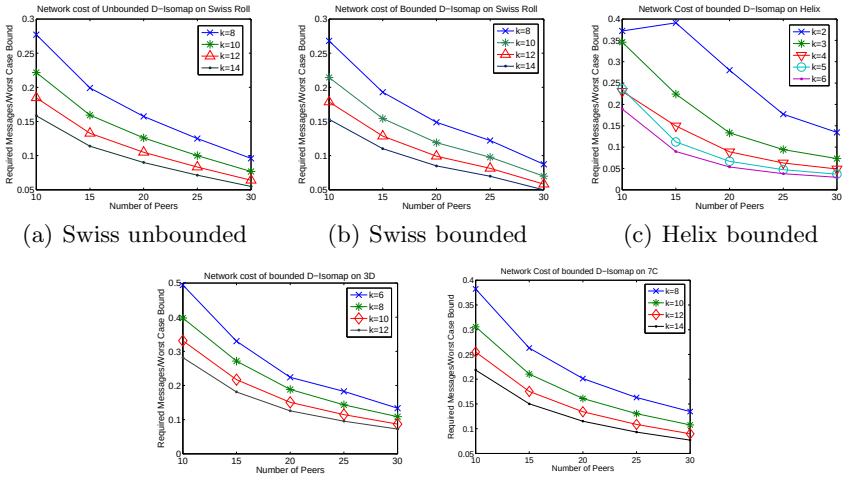
can be easily obtained using the fact that given a set of nodes in a graph, i.e $(s, n_1, n_2, ..., n_k, e)$, the distances from s to each n_i and from each n_i to e, the shortest path from s to e is the one minimizing the overall distance. Therefore, we relay on the retrieved k-NNs and calculate the SPs of the new point from all local landmarks. Finally, we obtain its embedding through LMDS or FEDRA. The procedure requires $O(ck)$ messages. The case of deletion is much simpler, since the host node will only transmit message $(point_{id}, del)$ and force the deletion of the point from the bucket of the indexing peer. On the other hand, the arrival or departure of peers is handled by the Chord protocol itself.

4 Experiments

In this section we present the experimental evaluation of D-Isomap, which indeed verifies the expected performance and promotes it as an attractive solution for hard DDR and DDM problems. We carried out two types of experiments. First, we compared the manifold unfolding capability of D-Isomap in a distributed context against Isomap and L-Isomap in various network settings. In the second set of experiments, we evaluated D-Isomap's performance against Isomap, L-Isomap and LSI [4] in numerous supervised and usupervised DDM experiments using a medium sized text collection. The obtained results prove the suitability and viability of our algorithm for DDM problems, where each node holds a subset of the available information.

In the first set of experiments we employed three $3D$ non linear manifolds, namely the Swiss Roll, Helix and 3D Clusters each consisting of 3000 points (Figures 1(a), 1(b), 1(c)). In the case of the Swiss Roll an NLDR algorithm should unfold the roll into a parallelogram while in the case of Helix it should extract a circle. Concerning the 3D Clusters, we target in retaining the cluster structure in the new space. Each dataset was randomly distributed in a network of M peers ($M = 10, 15, 20, 25$ and 30). Depending on the dataset, we varied the value of k; for the Swiss Roll we set $k = 8$ and progressively augmented it by 2 until 14. For the 3D Clusters we started from $k = 6$ and reached 12 using the same step. For Helix we ranged k from 2 to 6 with a step of 1. In all experiments we set $c = 5$, $L = 10$, $f = 10$ and $w = 16$.

We assess the quality of D-Isomap by comparing the produced manifold (on peer level) against those produced centrally by Isomap and L-Isomap. For L-Isomap we set $a = 300$ in all experiments. In the subsequent graphs, D_F-Isomap



(a) Swiss unbounded (b) Swiss bounded (c) Helix bounded

(d) 3D bounded (e) Text collection

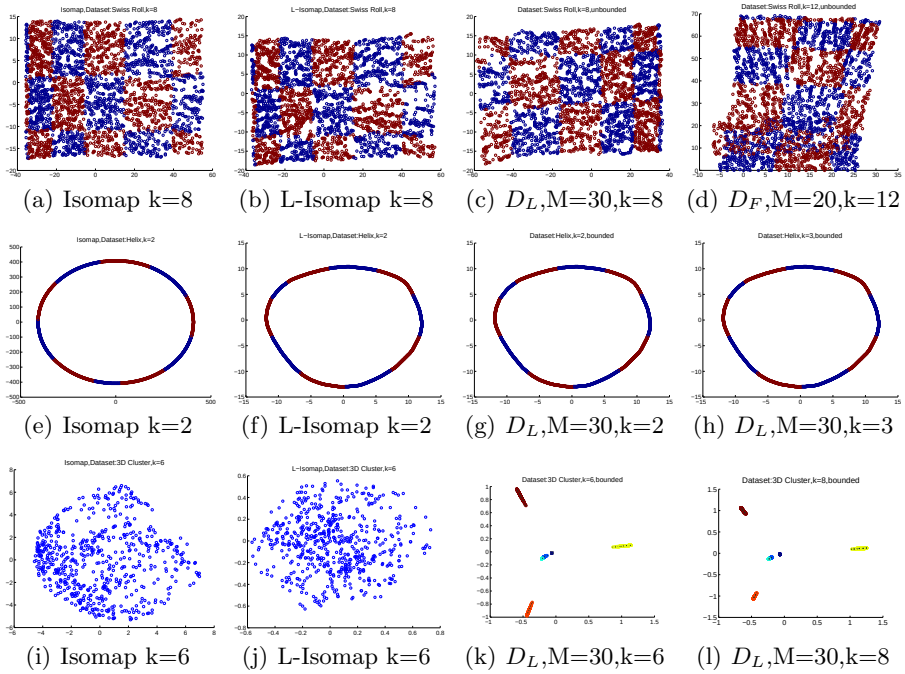
Fig. 2. Network cost reported as a fraction of the worst case bound $\frac{RequiredMessages}{WorstCaseBound}$

or D_F indentifies D-Isomap configured with FEDRA and D_L-Isomap or D_L, D-Isomap deployed with LMDS. We used MATLAB R2008a for the implementation of the algorithms and E2LSH [2] for LSH. Due to space limitations, we report only a subset of the experiments[1].

In Figure 2 we present the required number of messages for the projection of each dataset with D-Isomap as a fraction of the worst case network cost ($\frac{RequiredMessages}{WorstCaseBound}$) as derived by Section 3.3. First we validated the bound of Theorem 1 with the Swiss Roll. The results (Figures 2(a), 2(b)) indicate a reduction in the number of messages; consequently we employed the bounded version of the algorithm for all experiments. Figures 2(b), 2(c) and 2(d) provide the network load for the projection of each dataset with D-Isomap. The results highlight that D-Isomap behaves better in terms of network cost as the network size grows. The reason is simple; as the network grows, the buckets retained by each peer are smaller, therefore the messages are reduced. Moreover, messages are not affected seriously by changes in k so we observe a reduction in the percentage as k grows larger.

Figures 3(a), 3(b) depict the results obtained from Isomap and L-Isomap when applied on Swiss Roll for $k = 8$. Both algorithms have successfully revealed the underlying manifold. D_L-Isomap also recovered the correct 2D structure (Figure 3(c)) without being affected by the limited size of local data (only 3.3% of the global dataset). We report only one case of failure, for $M = 30$ and $k = 14$ where the embedding was skewed due to inappropriate selection of NNs.

[1] All experiments accompanied by the source code, the datasets and the graphs obtained from each peer are available at http://www.db-net.aueb.gr/panagis/PAKDD2010



(a) Isomap k=8 (b) L-Isomap k=8 (c) D_L,M=30,k=8 (d) D_F,M=20,k=12

(e) Isomap k=2 (f) L-Isomap k=2 (g) D_L,M=30,k=2 (h) D_L,M=30,k=3

(i) Isomap k=6 (j) L-Isomap k=6 (k) D_L,M=30,k=6 (l) D_L,M=30,k=8

Fig. 3. Isomap, L-Isomap and D-Isomap on Swiss Roll (top), Helix (middle) and 3D Clusters (bottom)

D_F-Isomap produces acceptable results however of lower quality compared to D_L-Isomap (Figure 3(d)). This is due to the fact that FEDRA operates using only 2 points while LMDS employs the whole local dataset at each node.

Similar quality results were obtained from D_L-Isomap during the evaluation of Helix. Our algorithm managed to recover the circle structure of Helix (Figures 3(g), 3(h)) providing results comparable to L-Isomap (Figure 3(f)) and Isomap (Figure 3(e)). The inability of D_F-Isomap to work with a limited number of landmark points was more evident this time, producing an arc instead of a circle. The effectiveness of D-Isomap was proved when it was applied on the 3D Clusters dataset. Unlike Isomap and L-Isomap that failed to produce a connected graph (Figures 3(i), 3(j)), D_L-Isomap successfully managed to replicate the cluster structure in the new space (Figures 3(k)-3(l)) since by construction produces connected graphs. Again, D_F-Isomap failed to recover the whole cluster structure and preserved only three out of five clusters.

The results obtained from 3D Clusters inspired the application of D-Isomap on a DDM problem. As evaluation dataset, we used the titles of all papers published in ECDL, ECML/PKDD, FOCS, KDD, SIGMOD, SODA and VLDB conferences between 2006 and 2008 [2]. The dataset consists of 2167 papers, rep-

[2] The authors would like to thank Dr. G. Tsatsaronis who kindly provided the dataset

resented as 4726-dimensional vectors using a TF-IDF populated vector space model [4]. We randomly distributed the dataset among M peers (M = 10, 15, 20, 25 and 30) and embedded it in 10, 15, 20, 25 and 30 dimensions. We used the same values for L,f,a,c and w as before and ranged k from 8 to 14 with a step of 2. The embedded datasets from each peer were used as input for classification and clustering. We employed F-measure ([4]) in order to assess the quality of the results. In all experiments we report the relative quality amelioration $R = \frac{F_{m,new}}{F_{m,orig}}$. R represents the ratio of the F-measure ($F_{m,new}$) obtained in the low dimensional dataset over the F-measure ($F_{m,orig}$) obtained in the original case.

D_F-Isomap and D_L-Isomap were compared against Isomap, L-Isomap and LSI. For the central algorithms, reported values correspond to the mean of 10 executions. All results have been validated with a 10-fold cross validation. For D-Isomap we applied the same methodology on each peer level and report the mean value obtained across nodes. The statistical significance of D-Isomap's results has been verified by a t-test with confidence level 0.99. We employed k-Means and k-NN [4] for clustering and classification respectively; for k-NN we set $kNN = 8$ in all experiments. Although this may not be optimal, it does not affect our results, since we report the relative performance of the classifier.

Table 1(a) provides the clustering results obtained using $k = 8$ for the definition of the NNs for D-Isomap, Isomap and L-Isomap. The results highlight the applicability of D-Isomap in DDM problems as well as the non linear nature of text corpuses. Both flavours of our algorithm produce results marginally equal and sometimes superior to central LSI. The low performance of Isomap and L-Isomap should be attributed to the definition of non-connected NN graphs. Table 1(b) provides the classification results obtained for the same value of k. D-Isomap is outperformed only by central LSI while in cases ameliorates the quality of k-NN. The latter comprises an experimental validation of the curse of dimensionality. The network load induced by D-Isomap in this experiment is provided in 2(e).

5 Conclusion

In this paper we have presented D-Isomap, a novel distributed NLDR algorithm which to the best of our knowledge is the first attempt towards this direction. We presented in details each step of the procedure and assessed the requirements posed to each network node by its application. Through extensive experiments we validated the capability of D-Isomap to recover linear manifolds from highly non linear structures. Additionally, we highlighted its applicability in DKD problems through experiments on a real world text dataset. The high quality results inspire us to pursue the extension D-Isomap towards P2P document retrieval and web searching.

Table 1. Experiments on text. Is, LIs are used for Isomap,L-Isomap respectively

(a) Clustering experiments using k-Means

		Number of peers (M)												
		10		*15*		*20*		*25*		*30*				
		D_F	D_L	D_F	D_L	D_F	D_L	D_F	D_L	D_F	D_L	LSI	Is	LIs
Low Dim (n)	*10*	0.96	0.95	0.96	0.95	0.96	0.95	0.96	0.95	0.96	0.95	0.95	0.73	0.88
	15	0.96	0.95	0.96	0.95	0.96	0.95	0.96	0.95	0.96	0.95	0.98	0.70	0.85
	20	0.96	0.95	0.96	0.95	0.96	0.95	0.95	0.95	0.96	0.95	0.97	0.73	0.76
	25	0.95	0.95	0.95	0.95	0.96	0.95	0.95	0.95	0.95	0.95	0.90	0.71	0.81
	30	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.66	0.78

(b) Classification experiments using k-NN

		Number of peers (M)												
		10		*15*		*20*		*25*		*30*				
		D_F	D_L	D_F	D_L	D_F	D_L	D_F	D_L	D_F	D_L	LSI	Is	LIs
Low Dim (n)	*10*	1.11	1.04	1.10	1.07	1.11	1.08	1.10	1.08	1.10	1.08	1.14	0.84	0.76
	15	1.11	1.05	1.10	1.07	1.10	1.08	1.10	1.08	1.10	1.09	1.20	0.90	0.76
	20	1.10	1.05	1.10	1.07	1.10	1.08	1.10	1.09	1.10	1.09	1.24	0.90	0.80
	25	1.10	1.06	1.10	1.07	1.10	1.08	1.10	1.09	1.10	1.09	1.25	0.89	0.81
	30	1.10	1.06	1.10	1.07	1.10	1.08	1.10	1.09	1.10	1.09	1.24	0.90	0.79

References

1. Abu-Khzam, F.N., Samatova, N.F., Ostrouchov, G., Langston, M.A., Geist, A.: Ddr algorithms for widely dispersed data. IASTED PDCS pp. 167–174 (2002)
2. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM 51(1) (2008)
3. Cai, D., He, X., Han, J.: Document clustering using locality preserving indexing. IEEE TKDE 17(12), 1624–1637 (2005)
4. Chakrabarti, S.: Mining the Web: Discovering Knowledge from Hypertext Data. Morgan Kaufmann (2002)
5. Gu, Q., Zhou, J.: Local relevance weighted maximum margin criterion for text classification. SIAM SDM pp. 1135–1146 (2009)
6. Haghani, P., Michel, S., Aberer, K.: Distributed similarity search in high dimensions using locality sensitive hashing. ACM EDBT pp. 744–755 (2009)
7. Henry, G., Geijn, R.: Parallelizing the qr algorithm for the unsymmetric algebraic eigenvalue problem. SIAM JSC pp. 870–883 (1994)
8. Hinneburg, A., Aggarwal, C.C., Keim, D.A.: What is the nearest neighbor in high dimensional spaces? VLDB pp. 506–515 (2000)
9. Kargupta, H., Huang, W., Sivakumar, K., Park, B.H., Wang, S.: Collective pca from distributed heterogeneous data. PKDD (2000)
10. Kurose, J.F., Ross, K.W.: Computer Networking: A Top-Down Approach Featuring the Internet. Addison-Wesley (2000)
11. Magdalinos, P., Doulkeridis, C., Vazirgiannis, M.: K-landmarks: Distributed dimensionality reduction for clustering quality maintenance. PKDD pp. 322–334 (2006)
12. Magdalinos, P., Doulkeridis, C., Vazirgiannis, M.: Fedra: A fast and efficient dimensionality reduction algorithm. SIAM SDM pp. 509–520 (2009)

13. Qi, H., Wang, T., Birdwell, D.: Global pca for dr in distributed data mining. Ch. 19 in SDMKD, CRC pp. 327–342 (2004)
14. Qu, Y., Ostrouchov, G., Samatova, N., Geist, A.: Pca for dr in massive distributed data sets. 5th International Workshop on High Performance Data Mining (2002)
15. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. ACM SIGCOMM pp. 161–172 (2001)
16. de Silva, V., Tenenbaum, J.B.: Global versus local methods in nonlinear dimensionality reduction. NIPS pp. 705–712 (2002)
17. Stoica, I., Morris, R., Karger, D., Kaashoek, F.M., Hari: Chord: A scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM (2001)
18. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290(5500), 2319–2323 (December 2000)
19. Togerson, W.: Theory and methods of scaling. Wiley (1958)
20. Zhao, D., Yang, L.: Incremental isometric embedding of high-dimensional data using connected neighborhood graphs. IEEE TPAMI 31(1), 86–98 (2009)