Graph Embedding and Dimensionality Reduction - A Survey

Nishana S S
Dept. Computer Science & Engineering
SCT College of Engineering
Trivandrum,India
nishanalilu@gmail.com

Subu Surendran
Associate Professor, Dept. Computer Science & Engineering
SCT College of Engineering
Trivandrum,India
subusurendran@gmail.com

Abstract - Dimension reduction is defined as the process of mapping high-dimensional data to a lower-dimensional vector space. Most machine learning and data mining techniques may not be effective for high-dimensional data. In order to handle this data adequately, its dimensionality needs to be reduced. Dimensionality reduction is also needed for visualization, graph embedding, image retrieval and a variety of applications. This paper discuss the most popular linear dimensionality reduction method Principal Component Analysis and the various non linear dimensionality reduction methods such as Multidimensional scaling, Isomap, Locally Linear Embedding, Laplacian Eigen Map, Semidefinite embedding, Minimum Volume Embedding and Structure Preserving Embedding .

Keywords: Spectral embedding, Semidefinite programming,

1. INTRODUCTION

In machine learning, computer vision, computational Biology, pattern recognition, data compression and in real-world data, such as speech signals, digital photographs, and other applied areas, datasets involves high dimensional objects .In order to handle this data adequately, its dimensionality needs to be reduced. Dimensionality reduction is the transformation of high-dimensional data into a meaningful representation of lower dimensionality. Ideally, the reduced representation should have a dimensionality that corresponds to the intrinsic dimensionality of the data. The intrinsic dimensionality of data is the minimum number of parameters needed to account for the observed properties of the data. Many algorithms for dimensionality reduction have been developed to accomplish these tasks. Traditionally, spectral methods such as principal component analysis (PCA) have been applied to many graph embedding and dimensionality reduction tasks. In spectral methods the low dimensional representations are derived from the top or bottom eigenvectors of specially constructed matrices. These methods aim to find low-dimensional representations of data that preserve its inherent structure. They assume that

the underling manifold is a linear subspace. In order to solve the problem of dimensionality reduction in nonlinear cases, many recent techniques, including Multidimensional scaling(MDS), locally linear embedding (LLE), Laplacian eigenmaps (LEM), Isomap , semidefinite embedding(SDE), Minimum volume embedding(MVE) and structure preserving embedding(SPE) have been proposed. These all provide different techniques for capturing the non-linearity of the underlying manifold incorporating local distance information in different ways. LLE[5], only considers local pairwise information between points. Similarly, Laplacian Eigenmaps[1] operates in this local regime. Isomap[12], on the other hand, operates globally on the set of all distances between points. It uses local information to construct a k-nearest neighbor graph and estimates distances between far away points by considering the shortest path on the graph.

Graphs are essential for encoding information, and it is widely used to represent the data in many fields ranging from computational biology to computer vision. When the input data is a binary adjacency matrix rather than high-dimensional data, many of these graph-based dimensionality reduction algorithms can be applied directly. Graph embedding algorithms place nodes at points on some surface and connect points with an arc if the nodes have an edge between them. We aim to address the problem of how to organize graphs into a pattern-space in which similar structures are close to one-another and dissimilar structures

are far apart. There are a number of ways in which this can be achieved. One approach is to compute the distance between graphs and to use multidimensional scaling (MDS) to embed the individual graphs in a low-dimensional space. The second approach is to extract feature vectors from the graphs. A pattern-space can be constructed from such vectors by performing modal analysis on their covariance matrix. To overcome the

problem of how to map the structure of a graph onto a vector of fixed length, we turn to graph-spectral decomposition methods[11]. Spectral graph theory is a branch in mathematics which aims to characterize the properties of unweighted graphs using the eigenvalues and eigenvectors of the adjacency matrix or the closely related Laplacian matrix. While embedding the high dimensional data to low dimensional, the structure of the graph topology should not be distorted. To preserve the structure various connectivity algorithms such as k-Nearest Neighbor algorithm , bmatching or maximum weight spanning tree can be used.

The remainder of the paper is structured as follows. Section II deals with the different linear and non linear dimensionality reduction techniques. Section III concludes this survey.

2. DIMENSIONALITY REDUCTION

The problem of (nonlinear) dimensionality reduction can be defined as follows. Assume we have dataset represented in a n×D matrix X consisting of n datavectors x_i ($i \in \{1, 2, \ldots, n\}$) with dimensionality D. Assume further that this dataset has intrinsic dimensionality d (where d < D). Here, in mathematical terms, intrinsic dimensionality means that the points in dataset X are lying on or near a manifold with dimensionality d that is embedded in the D-dimensional space. Dimensionality reduction techniques transform dataset X with dimensionality D into a new dataset Y with dimensionality d, while retaining the geometry of the data as much as possible. In general, neither the geometry of the data manifold, nor the intrinsic dimensionality d of the dataset X are known. Therefore, dimensionality reduction is a problem that can only be solved by assuming certain properties of the data such as its intrinsic dimensionality.

2.1. Principal Component Analysis (PCA)

PCA is a linear dimension reduction method . It is based on the statistical representation of a random variable. Given a set of data on *n* dimensions, PCA aims to find a linear subspace of dimension *d* lower than *n* such that the data points lie mainly on this linear subspace . It is based on the covariance matrix of the variables. Such a reduced subspace attempts to maintain most of the variability of the data. Principal Components Analysis (PCA) constructs a low-dimensional representation of the data that describes as much of the variance in the data as possible. This is done by finding a linear basis of reduced dimensionality for the data, in which the amount of variance in the data is maximal. In mathematical terms, PCA attempts to find a linear mapping M that maximizes $M^T cov(X)M$, where cov(X) is the covariance matrix of the data X. This linear mapping is formed by the d principal eigenvectors (i.e., principal components) of the covariance matrix of the zero-mean data .Hence, PCA solves the eigenproblem $cov(X)M = \lambda M$. The eigenproblem is solved for the d principal eigenvalues λ. The low-dimensional data representations y_i of the datapoints x_i are computed by mapping them onto the linear basis M i.e., $Y = (X - X')M$. PCA has been successfully applied in a large number of domains such as face recognition , coin classification, and seismic series analysis . The main drawback of PCA is that the size of the covariance matrix is proportional to the dimensionality of the datapoints. As a result, the computation of the eigenvectors might be infeasible for very high-dimensional data. In datasets in which n < D, this drawback may be overcome by computing the eigenvectors of the squared Euclidean distance matrix $(X - X')(X - X')^T$ instead of the eigenvectors of the covariance matrix .

2.2. Multidimensional scaling (MDS)

Multidimensional scaling (MDS) is used to provide a visual representation of the pattern of proximities (i.e., similarities or distances) among a set of objects. Input to MDS is a matrix which defines a distance function $\delta_{i,j}$ which specifies the distance between i and j. MDS attempts to find an embedding from the input matrix into $\mathbf{R}^N$ such that distances are preserved. Suppose the input is a collection of I objects. The goal of MDS is to find I vectors $x_1, x_2, \ldots, x_I \in R^N$ such that $||x_i - x_j|| \approx \delta_{i,j}$ for all i,j ϵ I. If the dimension *N* is chosen to be 2 or 3, we may plot the vectors x_i to obtain a visualization of the similarities between the I objects. There are various approaches to determining the vectors x_i. Usually, MDS is formulated as an optimization problem where $x_1, x_2, \ldots, x_I$ is found as a minimizer of some cost function, for example,

$$\min_{x1,\ldots,xI} \sum_{i>j} (||xi - xj|| - \delta i,j)^2$$

A solution may then be found by numerical optimization techniques. For some particularly chosen cost functions, minimizers can be stated analytically in terms of matrix eigen decomposition.

2.3. Isomap

The drawback of Multidimensional scaling is that, it is based on the Euclidean distance and it doesn't take in to consideration any of the neighboring datapoints. If the high-dimensional data lies on or near a curved manifold, MDS consider two datapoints as near points, whereas their distance over the manifold is much larger than the typical interpoint distance. Isomap is a technique which incorporates the geodesic distances between datapoints to the Multidimensional scaling method . Geodesic distance is the distance between two points measured over the manifold. Isomap defines the geodesic distance to be the sum of edge weights along the

shortest path between two nodes. The top *n* eigenvectors of the geodesic distance matrix, represent the coordinates in the new *n*-dimensional Euclidean space.

Isomap involves 3 steps. First step is to calculate the geodic distance between datapoints by constructing a neighbourhood graph G by using any of the connectivity algorithms such as k Nearest neighbor. Next step is to find the shortest path between two points which gives a good estimate of the geodesic distance between two points which can be found out by using Dijkstra's algorithm.The last step is the low dimensional embedding which can be done by applying multidimensional scaling on the resulting distance metric. An important drawback of isomap algorithm is its topological instability. It makes connection error while constructing the neighborhood graph. Another drawback is that it can fail if the manifold is nonconvex .

2.4. Local Linear Embedding (LLE)

Local Linear Embedding (LLE) is a local technique for dimensionality reduction that is similar to Isomap in that it constructs a graph representation of the datapoints. It preserves local properties and hence it is less sensitive to the short circuit problem that arise in Isomap. Also, the preservation of local properties allows for successful embedding of nonconvex manifolds. LLE begins by finding a set of the nearest neighbors of each point. It then computes a set of weights for each point that best describe the point as a linear combination of its neighbors. Finally, it uses an eigenvector-based optimization technique to find the low-dimensional embedding of points.

Suppose we have the data set consisting of N vectors $X_1, X_2, \ldots, X_N$ each of dimensionality D. First step is to compute the neighbors of each data point X_i . Next Compute the weights W_{ij} that best reconstruct each data point from its neighbors. The reconstruction error is measured by the cost function

$$E(W) = \sum_i |Xi - \sum_j WijXj|^2$$

The weights W_{ij} summarize the contribution of the ith data point to the jth reconstruction. To compute the weights W_{ij} we minimize the cost function subject to two constraints. First, that each data point is reconstructed only from its neighbors, enforcing $W_{ij} = 0$ if X_j does not belongs to this set. Second, that the rows of the weight matrix sum to one $\sum_j Wij = 1$. This minimization function ensures that for any particular data point, they are invariant to rotations, rescalings, and translations of that data point and its neighbors. In the final step of the algorithm, each high dimensional data X_i is mapped to a low dimensional vector Y_i representing global internal coordinates on the manifold. This is done by choosing d dimensional coordinates Y_i to minimize the embedding cost function:

$$(Y) = \sum_i |Yi - \sum_j WijYj|^2$$

It can be minimized by solving a sparse NxN eigenvector problem, whose bottom non-zero eigenvectors provide an ordered set of orthogonal coordinates centered on the origin.

Sometimes LLE performs worse than isomap .It fails to provide good visualization of data objects. The difficulty arise while dealing with manifolds containing holes. In addition, LLE tends to collapse large portions of the data onto a single point in cases where the target dimensionality is too low.

2.5. Laplacian eigenmaps

Laplacian Eigenmaps uses spectral techniques to perform dimensionality reduction. This technique relies on the basic assumption that the data lies in a low dimensional manifold in a high dimensional space. Like LLE it preserves local properties so that it's not prone to shot circuiting. It is based on the intrinsic geometry structure of the manifold so that it exhibit stability in the embedding. The Laplacian Eigenmap algorithm first constructs a neighborhood graph G in which every datapoint xi is connected to its k nearest neighbors. For all points xi and xj in graph G that are connected by an edge, the weight of the edge is computed using the Gaussian kernel function $w_{ij} = e^{-\frac{||xi-xj||^2}{2t^2}}$ where t^2 indicates the variance of the Gaussian leading to a sparse adjacency matrix W. Finally each high dimensional data X_i is mapped to a low dimensional vector Y_i in a way so as to minimize the cost function

$$(Y) = \sum_{ij}(Yi - Yj)^2 \; Wij$$

Minimizing is an attempt to ensure that if **x***i* and **x***j* are "close," then *yi* and *yj* are close as well. Compute eigenvalues and eigenvectors for the generalized eigenvector problem,

$$L\mathbf{f} = \lambda D\mathbf{f}$$

where *D* is diagonal weight matrix, and its entries are column sums of *W*, $Dii = \sum_j Wji$,$L = D - W$ is the Laplacian matrix.The minimization function can be formulated as

$$(Y) = \sum_{ij}(Yi - Yj)^2 \; Wij = 2Y^T LY$$

This minimization function can be found out by solving the generalized eigen value problem and by using the the d eigenvectors f_i corresponding to the smallest nonzero eigenvalues form the low-dimensional data representation Y.

Although this type of embedding have some locality-preserving properties, they do not in general provide an isometric embedding. Another issue is that , this algorithm does not specify how it behaves if the manifold has a boundary. Also it implicitly assumed a uniform probability distribution on the manifold according to which the data points have been sampled.

2.6. Maximum variance unfolding (MVU)

Semidefinite embedding (SDE) or maximum variance unfolding (MVU) is an algorithm that uses semidefinite programming to perform non-linear dimensionality reduction of high-dimensional vectorial input data. MVU can be viewed as a non-linear generalization of Principal component analysis. The main objective behind MVU is to exploit the local linearity of manifolds and create a mapping that preserves local neighborhoods at every point of the underlying manifold. MVU algorithm starts by creating a neighborhood graph G by using K Nearest neighbor algorithm. Each input is connected with its k-nearest input vectors according to Euclidean distance metric and all k-nearest neighbors are connected with each other. The neighborhood graph is "unfolded" with the help of semidefinite programming. Semidefinite programming[6] aims to find an inner product matrix that maximizes the pairwise distances between any two inputs that are not connected in the neighborhood graph while preserving the nearest neighbors distances. Semidefinite programming is a generalization of linear programming.

Let X be the original input and Y be the embedding. If i ,j are two neighbors, then the local isometry constraint that needs to be satisfied is:

$$|X_i - X_j|^2 = |Y_i - Y_j|^2$$

MVU reformulates the optimization problem as a semidefinite programming problem (SDP) [6],[7] by defining a matrix K that is the inner product of the low-dimensional data representation Y . In SDP the optimization problem is formulated as:

$$\text{Maximize trace(K)}$$

$$\text{subject to } K \geq 0 \text{ , } \textstyle\sum_{ij} K_{ij} = 0 \text{ and } \forall\ i,j$$

$$\text{where, } Kii - 2Kij + Kjj = \|X_i - X_j\|^2$$

The low-dimensional embedding is finally obtained by application of multidimensional scaling on the learned inner product matrix.

Semidefinite embedding is much better in revealing the underlying dimension of the data compared to LLE and Laplacian eigenmaps. It also guarantees that the nearest neighbors in the embedding is the same as the original nearest neighbor for each point while the other two methods do not. On the other hand, semidefinite embedding is much slower and harder to scale to large data.

2.7. Minimum Volume Embedding (MVE)

Minimum Volume Embedding (MVE) is an algorithm for non-linear dimensionality reduction that uses semidefinite programming (SDP) and matrix factorization to find a low-dimensional embedding that preserves local distances between points while representing the dataset in many fewer dimensions. MVE follows an approach similar to SDE, in that it learns a kernel matrix using an SDP before applying Kernel Principal Component Analysis (KPCA). While SDE sometimes works well, the objective of pulling points apart (maximizing the trace of K) and unfolding by maximizing variance can create problems and use more dimensions than are necessary. So we would like to pull points apart in the dimensions that we wish but reduce the variance in dimensions that will be removed. Thus, we would like to grow the top few eigenvalues of K while shrinking the remaining ones . Ideally, if we knew the intrinsic dimensionality d of the manifold, MVE would therefore minimize the following cost function over the eigenvalues:

$$\min_{K \in K} f(K) = \min_{K \in K} - \textstyle\sum_{i=1}^{d} \lambda i + \sum_{i=d+1}^{N} \lambda i$$

$$\text{Subject to } Kv_i = \lambda i v_i \text{ , } {v_i}^T v_{i,} = \delta ij \text{ , } \lambda i \geq \lambda_{i+1} \text{ , } \forall\ i,j.$$

where λ are the eigenvalues of K in sorted order , v_i is the eigen vector and d is a user specified parameter which represents the dimensionality we need to embed.

$$f(K) = [tr(\textstyle\sum_{i=1}^{d} v_i\ v_i^T + \sum_{i=d+1}^{N} v_i\ v_i^T)] = B$$

Thus MVE problem can be defined as :

$$\min_{K \in K} f(K) = \min_{K \in K} tr(\textstyle\sum_{i=1}^{d} v_i\ v_i^T + \sum_{i=d+1}^{N} v_i\ v_i^T)$$

MVE algorithm begins by forming an affinity matrix A from the input data. A is then used to generate a connectivity matrix C where typically each point is connected to its k-nearest neighbors. Initialize K = A and find the eigen values and eigen vectors of K. Then by using SDP find K':

$$K' = \min_{K \in K} tr(KB)$$

Perform kernel PCA on K' to obtain d dimensional output vectors Y_1, Y_2 ,……, Y_N . In practice, the MVE algorithm does not seem to have any significant local minima since it converges reliably to the same solution.

2.8. Structure Preserving Embedding(SPE)

Structure Preserving Embedding (SPE) is an algorithm for embedding graphs in Euclidean space such that the embedding is lowdimensional and preserves the global topological properties of the input graph. Topology is preserved through a connectivity algorithms. It's not a dimensionality reduction method but rather a tool for embedding graphs in few dimensions. These two terms are closely related. Many nonlinear dimensionality reduction algorithms, such as Locally Linear Embedding (LLE), Maximum variance Unfolding (MVU), and Minimum Volume Embedding (MVE) begin by finding a sparse

connectivity matrix A that describes local pairwise distances. Preserving distances does not explicitly preserve the structure of this graph. LLE, MVU, and MVE, produce embeddings whose resulting connectivity no longer matches the inherent connectivity of the data. The problem arise from the fact that the distances between nodes that are connected are preserved; however, distances between unconnected nodes are free to vary, and thus can drastically change the graph's topology. So to preserve the graph topology we introduce a connectivity algorithm which enforces certain linear constraints on the learned kernel matrix K. For each node, the distances to all other nodes to which it is not connected must be larger than the distance to the furthest connected neighbor of that node. If we use k-nearest neighbor algorithm (knn) , this results in the linear constraints on K:

$$D_{ij} > (1 - A_{ij}) \max_m(A_{im}D_{im})$$

The k-nearest neighbor algorithm (knn) greedily connects each node to the k neighbors to which the node has shortest distance, where k is an input parameter. This linear constraint will vary as the connectivity algorithms varies. To choose a unique K from the admissible set in the convex hull generated by these linear constraints,we propose an objective function which favors lowdimensional embeddings.

The objective function $\max_{k \geq 0} tr(KA)$ subject to tr(K) $\leq$1 recovers a low-rank version of spectral embedding. Also applying the linear constraint enforced by knn, SPE preserves the global topology of the graph..ie solve SDP as

$$K' = \operatorname{argmax}_{k \geq 0} tr(KA)$$

$$\text{subject to } D_{ij} > (1 - A_{ij}) \max_m(A_{im}D_{im})$$

Apply SVD to *K'* and use the top eigenvectors as embedding coordinates.

3. CONCLUSION

This paper reviews different linear and non linear dimensionality reduction techniques and we discuss the various drawbacks of each technique and also the importance of dimensionality reduction. Dimensionality reduction is needed in various fields such as computational biology, visualization, image processing and so on. From our analysis we can conclude that structure preserving embedding is the most efficient among all other existing dimensional reduction technique which actually preserves the global topology of the input graph. That is the structure is preserved.

REFERENCES

[1] W.N. Anderson and T.D. Morley. Eigen values of the Laplacian of a graph. Linear and Multilinear Algebra, 18:141–145, 1985

[2] T. Cox and M. Cox. Multidimensional scaling. Chapman & Hall, London, UK, 1994.

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation, 15(6), 2002.

[4] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In Neural Information Processing Systems 14, 2001.

[5] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. Science, 290(5500), 2000.

[6] L. Vandenberghe and S. Boyd. Semidefinite programming. SIAM Review, 38(1):49–95, 1996.

[7] K. Q. Weinberger, F. Sha, and L. K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In Proceedings of the Twenty First International Conference on Machine Learning (ICML-04), pages 839–846, Banff, Canada, 2004.

[8] Blake Shaw &Tony Jebara,(2009).Structure Preserving Embedding Proceedings of the 26 th International Conference on Machine Learning.

[9] Shaw, B., & Jebara, T. (2007). Minimum volume embedding. Proc. of the 11th International Conference on Articial Intelligence and Statistics.

[10] Chung, F. R. K. (1997). Spectral graph theory. American Mathematical Society.

[11] Bin Luo, Richard C. Wilson, Edwin R. Hancock; Spectral embedding of graphs , Pattern Recognition Society. Published by Elsevier Science Ltd.2003.

[12] M. Balasubramanian and E.L. Schwartz. The Isomap algorithm and topological stability. Science, 295(5552):7, 2002.

[13] Arora, S., Rao, S., & Vazirani, U. (2004). Expanderows, geometric embeddings and graph partitioning. Symposium on Theory of Computing.
[14] Burer, S., & Monteiro, R. D. C. (2003). A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. Mathematical Programming (series B), 95(2), 329{357}.