Large-Scale Manifold Learning

Ameet Talwalkar
Courant Institute
New York, NY
ameet@cs.nyu.edu

Sanjiv Kumar
Google Research
New York, NY
sanjivk@google.com

Henry Rowley
Google Research
Mountain View, CA
har@google.com

Abstract

This paper examines the problem of extracting low-dimensional manifold structure given millions of high-dimensional face images. Specifically, we address the computational challenges of nonlinear dimensionality reduction via Isomap and Laplacian Eigenmaps, using a graph containing about 18 *million nodes and* 65 *million edges. Since most manifold learning techniques rely on spectral decomposition, we first analyze two approximate spectral decomposition techniques for large dense matrices (Nyström and Column-sampling), providing the first direct theoretical and empirical comparison between these techniques. We next show extensive experiments on learning low-dimensional embeddings for two large face datasets: CMU-PIE (*35 *thousand faces) and a web dataset (*18 *million faces). Our comparisons show that the Nyström approximation is superior to the Column-sampling method. Furthermore, approximate Isomap tends to perform better than Laplacian Eigenmaps on both clustering and classification with the labeled CMU-PIE dataset.*

1. Introduction

The problem of dimensionality reduction arises in many vision applications, where it is natural to represent images as vectors in a high-dimensional space. Manifold learning techniques extract low-dimensional structure from high-dimensional data in an unsupervised manner. These techniques typically try to unfold the underlying manifold so that Euclidean distance in the new space is a meaningful measure of distance between any pair of points. This makes certain applications such as K-means clustering more effective in the transformed space.

In contrast to linear dimensionality reduction techniques such as Principal Component Analysis (PCA), manifold learning methods provide more powerful non-linear dimensionality reduction by preserving the local structure of the input data. Instead of assuming global linearity, these methods make a weaker local-linearity assumption, *i.e.*, for nearby points in high-dimensional input space, L_2 distance is assumed to be a good measure of geodesic distance, or distance along the manifold. Good sampling of the underlying manifold is essential for this assumption to hold. In fact, many manifold learning techniques provide guarantees that the accuracy of the recovered manifold increases as the number of data samples increases. In the limit of infinite samples, one can recover the true underlying manifold for certain classes of manifolds [22][4][8]. However, there is a trade-off between improved sampling of the manifold and the computational cost of manifold learning algorithms. This paper addresses the computational challenges involved in learning manifolds given millions of face images extracted from the Web.

Several powerful manifold learning techniques have recently been proposed, *e.g.*, Semidefinite Embedding (SDE) [23], Isomap [22], Laplacian Eigenmaps [3], and Local Linear Embedding (LLE) [19]. SDE aims to preserve distances and angles between all neighboring points. It is formulated as an instance of semidefinite programming, and is thus prohibitively expensive for large-scale problems. Isomap constructs a dense matrix of approximate geodesic distances between *all* pairs of inputs, and aims to find a low dimensional space that best preserves these distances. Other algorithms, *e.g.*, Laplacian Eigenmaps and LLE, focus only on preserving local neighborhood relationships in the input space. They generate low-dimensional representations via manipulation of the graph Laplacian or other sparse matrices related to the graph Laplacian [20]. In this work, we focus mainly on Isomap and Laplacian Eigenmaps, as both methods have good theoretical properties and the differences in their approaches allow us to make interesting comparisons between dense and sparse methods.

All of the manifold learning methods described above can be viewed as specific instances of Kernel PCA [14]. These kernel-based algorithms require spectral decomposition of matrices of size $n \times n$, where n is the number of samples. This generally takes O(n^3) time. When only a few eigenvalues and eigenvectors are required, there exist less computationally intensive techniques such as the

Jacobi, the Arnoldi, and the more recent Hebbian methods [12][13]. These iterative methods require computation of matrix-vector products at each step and involve several passes through the data. When the matrix is sparse, these techniques can be implemented relatively efficiently. However, when dealing with a large, dense matrix, as in the case of Isomap, these products become expensive to compute. Moreover, when working with 18M data points, it is not possible even to store the full matrix ($\sim$ 1600TB), rendering the iterative methods infeasible.

Random sampling techniques provide a powerful alternative for approximate spectral decomposition and only operate on a subset of the matrix. Recently, the Nyström approximation has been studied in the machine learning community [24] [10]. In parallel, an alternative Column-sampling technique has been analyzed in the theoretical Computer Science community [7]. However, the relationship between these approximations has not been well studied. In this work, we show the connections between these algorithms, and provide the first direct comparison between their performance.

Apart from spectral decomposition, the other main computational hurdle associated with Isomap and Laplacian Eigenmaps is large-scale graph construction and manipulation. These algorithms first need to construct a local neighborhood graph in the input space, which is an O(n^2) problem. Moreover, Isomap requires shortest paths between every pair of points resulting in O($n^2 \log n$) computation. Both steps are intractable when n is as large as 18M. In this work, we use approximate nearest neighbor methods, and show that random sampling based spectral decomposition requires the computation of shortest paths only for a subset of points. Furthermore, these approximations allow for an efficient distributed implementation of the algorithms.

We now summarize our main contributions. First, we present the largest scale study so far on manifold learning, using 18M data points. To date, the largest manifold learning study involves the analysis of music data using 267K points [18]. In vision, the largest study is limited to less than 10K images [15]. Second, we show connections between two random sampling based spectral decomposition algorithms and provide the first direct comparison of their performance. Finally, we provide a quantitative comparison of Isomap and Laplacian Eigenmaps for large scale face manifold construction on clustering and classification tasks.

2. Manifold learning

Given n input points, $X = \{x_i\}_{i=1}^n$ and $x_i \in \mathbb{R}^d$, the goal is to find corresponding outputs $Y = \{y_i\}_{i=1}^n$, where $y_i \in \mathbb{R}^k$, $k \ll d$, such that Y 'faithfully' represents X. We first briefly review the Isomap and Laplacian Eigenmaps techniques to discuss their computational complexity.

2.1. Isomap

Isomap aims to extract a low-dimensional data representation that best preserves all pairwise distances between input points, as measured by their geodesic distances along the manifold [22]. It approximates the geodesic distance as a series of hops between neighboring points. This approximation becomes exact in the limit of infinite data. Isomap can be viewed as an adaptation of Classical Multidimensional Scaling [5], in which geodesic distances replace Euclidean distances.

Computationally, Isomap requires three steps: (1) Find t nearest neighbors for each point in input space and construct an undirected neighborhood graph, $\mathcal{G}$, with points as nodes and links between neighbors as edges. This requires O(n^2) time. (2) Compute approximate geodesic distances, Δ_{ij}, between all nodes (i, j) by finding shortest paths in $\mathcal{G}$ using Dijkstra's algorithm at each node. Construct a dense, $n \times n$ similarity matrix, G, by centering Δ_{ij}^2, where centering converts distances into similarities. This step takes O($n^2 \log n$) time, dominated by calculation of geodesic distances. (3) Find the optimal k dimensional representation, $Y = \{y_i\}_{i=1}^n$, such that $Y = \mathrm{argmin}_{Y'} \sum_{i,j} \big(\|y_i' - y_j'\|_2^2 - \Delta_{ij}^2\big)$. The solution is given by,

$$Y = (\Sigma^k)^{1/2} (U^k)^T \tag{1}$$

where Σ^k is the diagonal $k \times k$ matrix storing the top k eigenvalues of G, and U^k are the associated eigenvectors. This step requires O(n^2) space for storing G, and O(n^3) time for its eigendecomposition. The time and space complexities for all three steps are intractable for $n = 18M$.

2.2. Laplacian eigenmaps

Laplacian Eigenmaps aims to find a low-dimensional representation that best preserves neighborhood relations as measured by a weight matrix W [3]. The algorithm works as follows: (1) Similar to Isomap, first find t nearest neighbors for each point. Then construct W, a sparse, symmetric $n \times n$ matrix, where $W_{ij} = \exp\big(-\|x_i - x_j\|_2^2/\sigma^2\big)$ if (x_i, x_j) are neighbors, 0 otherwise, and σ is a scaling parameter. (2) Construct the diagonal matrix D, such that $D_{ii} = \sum_j W_{ij}$, in O(tn) time. (3) Find the k dimensional representation by minimizing the normalized, weighted distance between neighbors as, $Y = \mathrm{argmin}_{Y'} \sum_{i,j} \left(\frac{W_{ij} \|y_i' - y_j'\|_2^2}{\sqrt{D_{ii} D_{jj}}} \right)$. Suppose, $\mathcal{L} = I - D^{-1/2} W D^{-1/2}$ is the symmetrized, normalized form of the graph Laplacian, given by $D - W$. Then, the solution is,

$$Y = (U^k)^T \tag{2}$$

where U^k are the bottom k eigenvectors of $\mathcal{L}$, excluding the last eigenvector corresponding to eigenvalue 0. Since $\mathcal{L}$ is sparse, it can be stored in O(tn) space, and iterative methods can be used to find these k eigenvectors relatively quickly.

To summarize, for large-scale manifold learning, the two main computational efforts required are neighborhood graph construction/manipulation and spectral decomposition of a symmetric positive semidefinite (SPSD) matrix. We discuss two approximate spectral decomposition techniques in the next section, and describe the graph operations in Section 4.

3. Approximate spectral decomposition

Spectral decomposition is intractable for large n, especially in the case of Isomap where G is dense and too large to be stored. Two different methods have recently been introduced for approximating spectral decomposition of a large matrix using a subset of the columns (or rows) of the matrix. These techniques have appeared in two different communities, and there exists no work analyzing their relationship and comparative performance. In this section, we address both of these issues.

3.1. Terminology

In manifold learning, one deals with an $n \times n$ SPSD matrix, G, which can be decomposed as $G = U\Sigma U^T$. Here Σ contains the eigenvalues of G and U are the associated eigenvectors. Suppose we randomly sample $l \ll n$ columns of G uniformly without replacement[1]. Let C be the $n \times l$ matrix of these sampled columns, and W be the $l \times l$ matrix consisting of the intersection of these l columns with the corresponding l rows of G. Since G is SPSD, W is also SPSD. Without loss of generality, we can rearrange the columns and rows of G based on this sampling such that:

$$G = \begin{bmatrix} W & G_{21}^T \\ G_{21} & G_{22} \end{bmatrix} \quad \text{and} \quad C = \begin{bmatrix} W \\ G_{21} \end{bmatrix}. \tag{3}$$

We next discuss two approximation techniques that use eigendecomposition of W or singular value decomposition (SVD) of C to generate approximations of U and Σ.

3.2. Nyström method

The Nyström method was presented in [24] to speed up the performance of kernel machines. It has since been used for Landmark MDS [17] and image segmentation [11]. The Nyström method uses W and C from (3) to approximate G. According to the Nyström method,

$$G \approx \tilde{G} = CW^+C^T, \tag{4}$$

where W^+ is the pseudoinverse of W. It has been shown that $\tilde{G}$ converges to G as l increases [10]. Substituting (3) into (4), the Nyström method reduces to approximating G_{22} in $\tilde{G}$ using W and C as,

$$\tilde{G} = \begin{bmatrix} W & G_{21}^T \\ G_{21} & G_{21}W^+G_{21}^T \end{bmatrix}. \tag{5}$$

The approximate eigenvalues ($\tilde{\Sigma}$) and eigenvectors ($\tilde{U}$) of G generated from the Nyström method are:

$$\tilde{\Sigma} = \left(\frac{n}{l}\right)\Sigma_W \tag{6}$$

$$\tilde{U} = \sqrt{\frac{l}{n}}CU_W\Sigma_W^+ \tag{7}$$

where $W = U_W\Sigma_W U_W^T$ [24].

To calculate approximations to the top k eigenvectors and eigenvalues of G, the runtime of this algorithm is O($l^3 + kln$), l^3 for eigendecomposition on W and kln for multiplication with C.

3.3. Column-sampling approximation

The Column-sampling method was initially introduced to approximate SVD for any rectangular matrix and has been shown to have bounded approximation error [7][9]. It approximates eigendecomposition of G by using the SVD of C directly[2]. Suppose $C = U_C\Sigma_C V_C^T$, then the approximate eigenvectors of G are given by the left singular vectors of C:

$$\tilde{U} = U_C = CV_C\Sigma_C^+, \tag{8}$$

and the corresponding approximate eigenvalues are scaled versions of the singular values of C:

$$\tilde{\Sigma} = \sqrt{\frac{n}{l}}\Sigma_C. \tag{9}$$

Combining (8), (9) and $\tilde{G} = \tilde{U}\tilde{\Sigma}\tilde{U}^T$, we get:

$$G \approx \tilde{G} = C\left(\sqrt{\frac{l}{n}}(C^TC)^{\frac{1}{2}}\right)^+ C^T. \tag{10}$$

Comparing (4) and (10), we see that the two approximations have very similar forms. Specifically, the Column-sampling method replaces W^+ in (4) with $\sqrt{\frac{l}{n}}(C^TC)^{\frac{1}{2}}$.

SVD on C costs O(nl^2) but since it cannot be easily parallelized, it is still quite expensive when n = 18M. However, since $C^TC = V_c\Sigma_c^2V_c^T$, one can get U_c and Σ_c by SVD on C^TC combined with (8). This is advantageous as C^TC can be computed easily even for large n since matrix multiplication can be parallelized. Thus, the time needed to calculate approximations to the top k eigenvectors and eigenvalues of G is O($nl^2 + l^3$), nl^2 to generate C^TC and l^3 for SVD on C^TC.

[1]Other sampling schemes have been suggested [10][7]. However, random sampling is least costly, and has been empirically shown to be competitive with other schemes.

[2]The Nyström method also uses sampled columns of G, but the Column-sampling method is named so because it uses direct decomposition of C, while the Nyström method decomposes its submatrix, W.

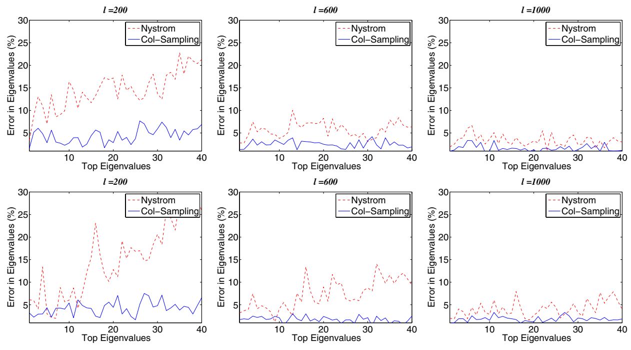
Figure 1. Mean error in top approximate eigenvalues, measured by percent deviation from the exact ones for varying numbers of samples (l). Top row: PIE-2.7K, bottom row: PIE-7K. Column-sampling gives more accurate eigenvalues than Nyström approximation for both datasets.

3.4. Approximation experiments

We conducted experiments to see how well the Nyström and Column-sampling methods approximate eigenvalues, eigenvectors, and low-dimensional embeddings. We worked with PIE-2.7K and PIE-7K, two subsets of the CMU-PIE face dataset (see Section 4.1) containing 2.7K left profile face images and 7K frontal face images, respectively. Starting with $A \in \mathbb{R}^{d \times n}$, a matrix containing mean-centered input images ($d = 2304$) as its columns, we constructed a SPSD Gram matrix $A^T A$ and compared its exact decomposition with the two approximate ones.

For eigenvalues, we measured the percentage error between the exact and the approximate ones for different values of l. For eigenvectors, accuracy was measured by the dot products, i.e., cosines of principal angles, between the exact and the approximate eigenvectors, which should be close to one for good approximations. For a fixed l, each experiment was repeated 10 times with different samples of columns. Figures 1 and 2 show that both approximations yield more accurate results as l increases. Also, the Column-sampling method generates more accurate eigenvectors and eigenvalues for both datasets. This is not surprising since the Column-sampling method performs SVD on a larger submatrix of G than does the Nyström method (C versus W). Another possible reason for the poor Nyström performance is that, unlike the Column-sampling method, the eigenvectors from the Nyström method are not exactly orthonormal. As shown in (7), the Nyström method extrapolates eigenvectors of G from eigenvectors of W, losing orthonormality in this process. In the future, it would be interesting to explore the construction of orthonormal approximations to the eigenvectors from the Nyström method.

We next compared the quality of low-dimensional embeddings, as defined in (1), constructed using the top $k = 100$ eigenvalues and eigenvectors for varying values of l. The average L_2 error in projection given by

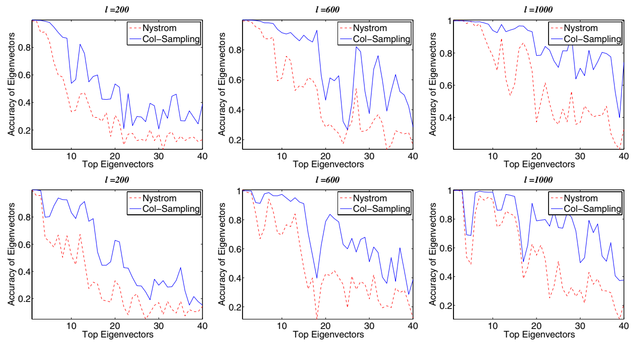
Figure 2. Mean accuracy of top approximate eigenvectors measured by dot product with exact ones for different numbers of samples (l). Top row: PIE-2.7K, bottom row: PIE-7K. Column-sampling gives more accurate eigenvectors than Nyström approximation for both datasets.

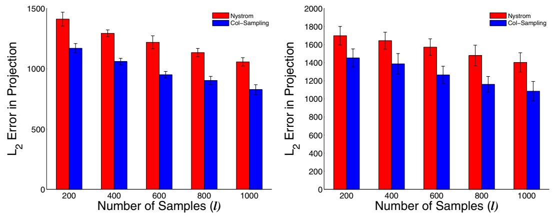
Figure 3. Average L_2 error for projections ($k = 100$) with varying l. Error bars show one standard deviation for 10 different runs for each l. Left: PIE-2.7K. Right: PIE-7K. Column-sampling gives lower error than Nyström for both datasets.

$\|\text{exact} - \text{approx}\|_2$ is shown in Figure 3. Consistent with the approximation accuracy of eigenvalues and eigenvectors independently, the Column-sampling method results in a lower projection error than the Nyström method. Moreover, as the number of samples increases, the projection error decreases for both methods.

4. Large-scale learning

The following sections outline the process of learning a manifold of faces. We first describe the datasets used in Section 4.1. Section 4.2 explains how to extract nearest neighbors, a common step between Laplacian Eigenmaps and Isomap. The remaining steps of Laplacian Eigenmaps are straightforward, so the subsequent sections focus on Isomap, and specifically on the computational efforts required to generate a manifold using Webfaces-18M.

4.1. Datasets

We used two datasets of faces consisting of 35K and 18M images. The CMU PIE face dataset [21] contains $41,368$ images of 68 subjects under 13 different poses and various illumination conditions. A standard face detector extracted $35,247$ faces (each 48×48 pixels), which comprised our 35K set (PIE-35K). We used this set because, being labeled, it allowed us to perform quantitative comparisons. The second dataset, named Webfaces-18M, contains 18.2 million images of faces extracted from the Web using the same face detector. For both datasets, face images were represented as 2304 dimensional pixel vectors which were globally normalized to have zero mean and unit variance. No other pre-processing, *e.g.* face alignment, was performed. Constructing Webfaces-18M, including face detection and duplicate removal, took 15 hours using a cluster of several hundred machines. We used this cluster for all experiments requiring distributed processing and data storage.

4.2. Nearest neighbors and neighborhood graph

The cost of naive nearest neighbor computation is $O(n^2)$, where n is the size of the dataset. It is possible to compute exact neighbors for PIE-35K, but for Webfaces-18M this computation is prohibitively expensive. So, for this set, we used a combination of random projections and spill trees [16] to get approximate neighbors. Computing 5 nearest neighbors in parallel with spill trees took $\sim$2 days on the cluster. Figure 4 shows the top 5 neighbors for a few randomly chosen images in Webfaces-18M. In addition to this visualization, comparison of exact neighbors and spill tree approximations for smaller subsets suggested good performance of spill trees.

We next constructed the neighborhood graph by representing each image as a node and connecting all neighboring nodes. Since Isomap and Laplacian Eigenmaps require this graph to be connected, we used depth-first search to find its largest connected component. These steps required $O(tn)$ space and time. Constructing the neighborhood graph for Webfaces-18M and finding the largest connected component took 10 minutes on a single machine using the OpenFST library [1].

For neighborhood graph construction, the 'right' choice of number of neighbors, t, is crucial. A small t may give too many disconnected components, while a large t may introduce unwanted edges. These edges stem from inadequately sampled regions of the manifold and false positives introduced by the face detector. Since Isomap needs to compute shortest paths in the neighborhood graph, the presence of bad edges can cause leakage or 'short-circuits'[2]. Here, we chose $t = 5$ and also enforced an upper limit on neighbor distance to alleviate the problem of leakage. We used a distance limit corresponding to the 95th percentile of neighbor distances in the PIE-35K dataset.

Table 1 shows the effect of choosing different values for t with and without enforcing the upper distance limit. As expected, the size of the largest connected component increases as t increases. Also, enforcing the distance limit reduces the size of the largest component. Figure 5 shows a few random samples from the largest component. Images not within the largest component are either part of

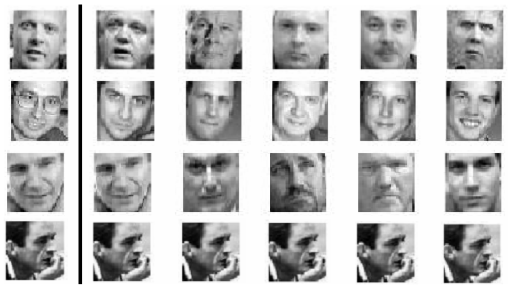
Figure 4. Visualization of neighbors for Webfaces-18M. The first image in each row is the target, and the next five are its neighbors.

	No Upper Limit		Upper Limit Enforced	
t	# Comp	% Largest	# Comp	% Largest
1	1.7M	0.05 %	4.3M	0.03 %
2	97K	97.2 %	285K	80.1 %
3	18K	99.3 %	277K	82.2 %
5	1.9K	99.9 %	275K	83.1 %

Table 1. Number of components in the Webfaces-18M neighbor graph and the percentage of images within the largest connected component ('% Largest') for varying numbers of neighbors (t) with and without an upper limit on neighbor distances.

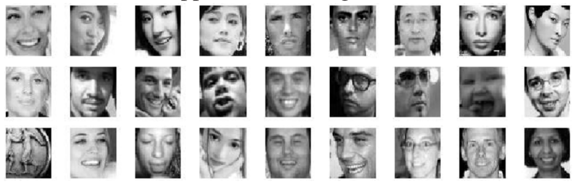
Figure 5. A few random samples from the largest connected component of the Webfaces-18M neighborhood graph.

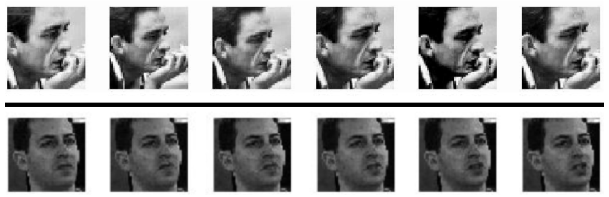
Figure 6. Visualization of disconnected components of the neighborhood graphs from Webfaces-18M (top row) and from PIE-35K (bottom row). The neighbors for each of these images are all within this set thus making the entire set disconnected from the rest of the graph. Note that these images are not exactly the same.

a strongly connected set of images (Figure 6) or do not have any neighbors within the upper distance limit (Figure 7). As shown in Figure 7, many of these 'single-image-components' are false positives. Clearly, the distance limit introduces a trade-off between filtering out non-faces and excluding actual faces from the largest component.[3]

4.3. Approximating geodesics

To construct the similarity matrix G in Isomap, one approximates geodesic distance by shortest-path lengths be-

[3]To construct embeddings with Laplacian Eigenmaps, we generated W and D from nearest neighbor data for images within the largest component of the neighborhood graph and solved (2) using a sparse eigensolver.

Figure 7. Visualization of disconnected components containing exactly one image. Although several of the images above are not faces, some are actual faces, suggesting that certain areas of the face manifold are not adequately sampled by Webfaces-18M.

tween every pair of nodes in the neighborhood graph. This requires $O(n^2 \log n)$ time and $O(n^2)$ space, both of which are prohibitive for 18M nodes. However, since we use sampling-based approximate decomposition as described in Section 3, we need only $l \ll n$ columns of G, which form the submatrix C. We thus computed geodesic distance between l randomly selected nodes (called landmark points) and the rest of the nodes, which required $O(ln \log n)$ time and $O(ln)$ space. Since this computation can easily be parallelized, we performed geodesic computation on the cluster. The overall procedure took 60 minutes for Webfaces-18M using $l = 10$K. The bottom four rows in Figure 9 show sample shortest paths for images within the largest component for Webfaces-18M, illustrating smooth transitions between images along each path.

4.4. Generating low-dimensional embeddings

Before generating low-dimensional embeddings in Isomap, one needs to convert distances into similarities using a process called centering [5]. For the Nyström approximation, we computed W by double centering D, the $l \times l$ matrix of squared geodesic distances between all landmark nodes, as $W = -\frac{1}{2}HDH$, where $H = I_l - \frac{1}{l}\mathbf{1}\mathbf{1}^T$ is the centering matrix, I_l is the $l \times l$ identity matrix and $\mathbf{1}$ is a column vector of all ones. Similarly, the matrix C was obtained from squared geodesic distances between the landmark nodes and all other nodes using single-centering as described in [6].

For the Column-sampling approximation, we decomposed C^TC, constructed by performing matrix multiplication in parallel on C. For both approximations, decomposition on an $l \times l$ matrix (C^TC or W) took about one hour. Finally, we computed low-dimensional embeddings by multiplying the scaled eigenvectors from approximate decomposition with C. For Webfaces-18M, generating low dimensional embeddings took 1.5 hours for the Nyström method and 6 hours for the Column-sampling method.

5. Manifold evaluation

Manifold learning techniques typically transform the data such that Euclidean distance in the transformed space between *any* pair of points is meaningful. Since K-means clustering computes Euclidean distances between all pairs of points, it is a natural choice for evaluating these techniques. We also compared the performance of various techniques using nearest neighbor classification. Since CMU-PIE is a labeled dataset, we first focused on quantitative evaluation of different embeddings using face pose as class labels. The PIE set contains faces in 13 poses, and such a fine sampling of the pose space makes clustering and classification tasks very challenging. In all the experiments we fixed the dimension of the reduced space, k, to be 100.

The first set of experiments was aimed at finding how well different Isomap approximations perform in comparison to exact Isomap. We used a subset of PIE with 10K images (PIE-10K) since, for this size, exact eigendecomposition could be done on a single machine within reasonable time and memory limits. We fixed the number of clusters in our experiments to equal the number of pose classes, and measured clustering performance using two measures, *Purity* and *Accuracy*. Purity measures the frequency of data belonging to the same cluster sharing the same class label, while Accuracy measures the frequency of data from the same class appearing in a single cluster. Thus, ideal clustering will have 100% Purity and 100% Accuracy.

Table 2 shows that clustering with Nyström Isomap with just $l = 1$K performs almost as well as exact Isomap on this dataset[4]. This matches with the observation made in [24], where the Nyström approximation was used to speed up kernel machines. Further, Column-sampling Isomap performs slightly worse than Nyström Isomap. The clustering results on the full PIE-35K set (Table 3) with $l = 10$K also affirm this observation. Figure 8 shows the optimal 2D projections from different methods for PIE-35K. The Nyström method separates the pose clusters better than Column-sampling, verifying the quantitative results.

The fact that Nyström outperforms Column-sampling is somewhat surprising given the experimental evaluations in Section 3.4. We believe that the poor performance of Column-sampling Isomap is due to the form of the similarity matrix G. When using a finite number of data points for Isomap, G is not guaranteed to be positive semidefinite (PSD). We verified that G was not PSD in our experiments, and a significant number of top eigenvalues, *i.e.*, those with largest magnitudes, were negative. The two approximation techniques differ in their treatment of negative eigenvalues and the corresponding eigenvectors. The Nyström method allows one to use eigenvalue decomposition (EVD) of W to yield signed eigenvalues, making it possible to discard the negative eigenvalues and the corresponding eigenvectors. On the contrary, it is not possible to discard these in the Column-based method, since the signs of eigenvalues are lost in the SVD of the rectangular matrix C (or EVD of

[4]The differences are statistically insignificant.

Methods	Purity (%)	Accuracy (%)
PCA	54.3 ($\pm$0.8)	46.1 ($\pm$1.4)
Exact Isomap	58.4 ($\pm$1.1)	53.3 ($\pm$4.3)
Nyström Isomap	59.1 ($\pm$0.9)	53.3 ($\pm$2.7)
Col-Sampling Isomap	56.5 ($\pm$0.7)	49.4 ($\pm$3.8)
Laplacian Eigenmaps	35.8 ($\pm$5.0)	69.2 ($\pm$10.8)

Table 2. Results of K-means clustering of face poses applied to PIE-10K for different algorithms. Results are averaged over 10 random K-means initializations.

Methods	Purity (%)	Accuracy (%)
PCA	54.6 ($\pm$1.3)	46.8 ($\pm$1.3)
Nyström Isomap	59.9 ($\pm$1.5)	53.7 ($\pm$4.4)
Col-Sampling Isomap	56.1 ($\pm$1.0)	50.7 ($\pm$3.3)
Laplacian Eigenmaps	39.3 ($\pm$4.9)	74.7 ($\pm$5.1)

Table 3. Results of K-means clustering of face poses applied to PIE-35K for different algorithms. Results are averaged over 10 random K-means initializations.

C^TC).

Tables 2 and 3 also show a significant difference in the Isomap and Laplacian Eigenmaps results. The 2D embeddings of PIE-35K (Figure 8) reveal that Laplacian Eigenmaps projects data points into a small compact region, as it tends to map neighboring inputs as nearby as possible in the low-dimensional space. When used for clustering, these compact embeddings lead to a few large clusters and several tiny clusters, thus explaining the high accuracy and low purity of the clusters. This indicates poor clustering performance of Laplacian Eigenmaps, since one can achieve even 100% Accuracy simply by grouping all points into a single cluster. However, the Purity of such clustering would be very low. Finally, the improved clustering results of Isomap over PCA for both datasets verify that the manifold of faces is not linear in the input space.

We also compared the performance of Laplacian Eigenmaps and Isomap embeddings on pose classification. The data was randomly split into a training and a test set, and K-Nearest Neighbor (KNN) was used for classification. $K = 1$ gives lower error than higher K as shown in Table 4. Also, the classification error is lower for both exact and approximate Isomap than for Laplacian Eigenmaps, suggesting that neighborhood information is better preserved by Isomap (Tables 4 and 5). Note that, similar to clustering, the Nyström approximation performs as well as Exact Isomap (Table 4). Better clustering and classification results, combined with 2D visualizations, imply that approximate Isomap outperforms exact Laplacian Eigenmaps. Moreover, the Nyström approximation is computationally cheaper and empirically more effective than the Column-sampling approximation. Thus, we used Nyström Isomap to generate embeddings for Webfaces-18M.

After learning a face manifold from Webfaces-18M, we analyzed the results with various visualizations. The top row of Figure 9 shows the 2D embeddings from Nyström Isomap. The top left figure shows the face samples from

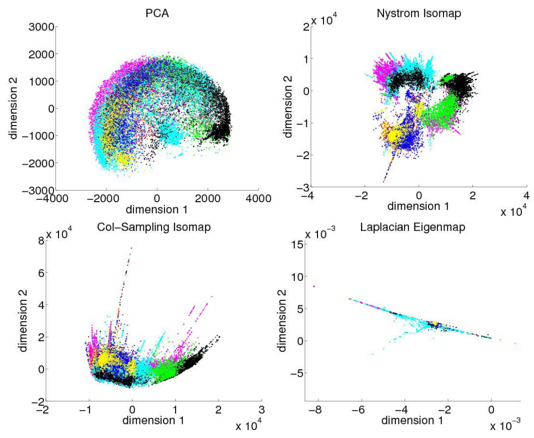

Figure 8. Optimal 2D projections of PIE-35K where each point is color coded according to its pose label. Top Left: PCA projections tend to spread the data to capture maximum variance, Top Right: Isomap projections with Nyström approximation tend to separate the clusters of different poses while keeping the cluster of each pose compact, Bottom Left: Isomap projections with Column-sampling approximation have more overlap than with Nyström approximation. Bottom Right: Laplacian Eigenmaps projects the data into a very compact range.

Methods	$K = 1$	$K = 3$
Exact Isomap	10.9 ($\pm$0.5)	14.1 ($\pm$0.7)
Nyström Isomap	11.0 ($\pm$0.5)	14.0 ($\pm$0.6)
Col-Sampling Isomap	12.0 ($\pm$0.4)	15.3 ($\pm$0.6)
Laplacian Eigenmaps	12.7 ($\pm$0.7)	16.6 ($\pm$0.5)

Table 4. K-nearest neighbor classification error $(\%)$ of face pose applied to PIE-10K subset for different algorithms. Results are averaged over 10 random splits of training and test sets. K=1 gives lower error.

Nyström Isomap	Col-Sampling Isomap	Laplacian Eigenmaps
9.8 ($\pm$0.2)	10.3 ($\pm$0.3)	11.1 ($\pm$0.3)

Table 5. 1-nearest neighbor classification error $(\%)$ of face pose applied to PIE-35K for different algorithms. Results are averaged over 10 random splits of training and test sets.

various locations in the manifold. It is interesting to see that embeddings tend to cluster the faces by pose. These results support the good clustering performance observed using Isomap on PIE data. Also, two groups (bottom left and top right) with similar poses but different illuminations are projected at different locations. Additionally, since 2D projections are very condensed for 18M points, one can expect more discrimination for higher k, *e.g.*, $k = 100$.

In Figure 9, the top right figure shows the shortest paths on the manifold between different public figures. The images along the corresponding paths have smooth transitions as shown in the bottom of the figure. In the limit of infinite samples, Isomap guarantees that the distance along the shortest path between any pair of points will be preserved as Euclidean distance in the embedded space. Even though the paths in the figure are reasonable approximations of straight lines in the embedded space, these results suggest that 18M

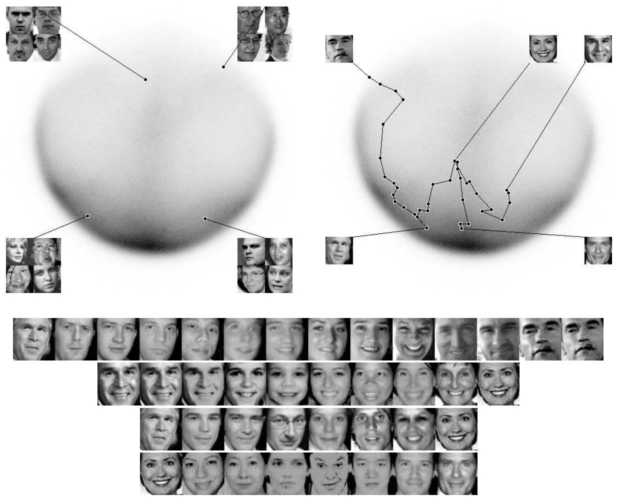

Figure 9. 2D embedding of Webfaces-18M using Nyström Isomap (Top row). Darker areas indicate denser manifold regions. Top Left: Face samples at different locations on the manifold. Top Right: Approximate geodesic paths between different celebrities. The corresponding shortest-paths are shown in bottom four rows.

faces are perhaps not enough samples to learn the face manifold exactly.

6. Conclusions and future work

We have presented large scale nonlinear dimensionality reduction using unsupervised manifold learning. The experimental results reveal that Isomap coupled with Nyström approximation can effectively extract low-dimensional structure from datasets containing millions of images. One of the drawbacks of Isomap is the assumption that centering of the dissimilarity matrix yields an SPSD matrix. This is not always true for dissimilarities that are not Euclidean distances. The presence of negative eigenvalues deteriorates the performance of the Column-sampling method more than the Nyström method. We plan to explore this issue further in the future. In addition, we plan to systematically investigate the effects of different data preprocessing methods such as face alignment, on manifold learning.

Acknowledgments

We thank Sergey Ioffe for use of his face detection code, Michael Riley for his assistance with OpenFST.

References

[1] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri. OpenFST: A general and efficient weighted finite-state transducer library. In *CIAA*, 2007. 5

[2] M. Balasubramanian and E. L. Schwartz. The Isomap algorithm and topological stability. *Science*, 295, 2002. 5

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, 2001. 1, 2

[4] M. Belkin and P. Niyogi. Convergence of laplacian eigenmaps. In *NIPS*, 2006. 1

[5] T. F. Cox, M. A. A. Cox, and T. F. Cox. *Multidimensional Scaling, Second Edition*. Chapman & Hall/CRC, 2000. 2, 6

[6] V. de Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *NIPS*, 2002. 6

[7] A. Deshpande, L. Rademacher, S. Vempala, and G. Wang. Matrix approximation and projective clustering via volume sampling. In *Symposium on Discrete Algorithms*, 2006. 2, 3

[8] D. Donoho and C. Grimes. Hessian eigenmaps: locally linear embedding techniques for high dimensional data. *Proc. of National Academy of Sciences*, 100(10):5591–5596, 2003. 1

[9] P. Drineas, R. Kannan, and M. W. Mahoney. Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. *SIAM J. Comput.*, 36(1), 2006. 3

[10] P. Drineas and M. W. Mahoney. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *JMLR*, 6, 2005. 2, 3

[11] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2), 2004. 3

[12] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 2nd edition, 1983. 2

[13] G. Gorrell. Generalized Hebbian algorithm for incremental singular value decomposition in natural language processing. In *EACL*, 2006. 2

[14] J. Ham, D. D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *ICML*, 2004. 1

[15] X. He, S. Yan, Y. Hu, and P. Niyogi. Face recognition using laplacianfaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(3):328–340, 2005. 2

[16] T. Liu, A. W. Moore, A. G. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. In *NIPS*, 2004. 5

[17] J. Platt. Fastmap, metricmap, and landmark MDS are all Nyström algorithms. pages 261–268. Society for Artificial Intelligence and Statistics, 2005. 3

[18] J. C. Platt. Fast embedding of sparse similarity graphs. In *NIPS*, 2003. 2

[19] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500), 2000. 1

[20] L. K. Saul, K. Q. Weinberger, J. H. Ham, F. Sha, and D. D. Lee. *Semisupervised Learning*. MIT Press, 2006. 1

[21] T. Sim, S. Baker, and M. Bsat. The CMU pose, illumination, and expression (PIE) database. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, May 2002. 4

[22] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500), 2000. 1, 2

[23] K. Q. Weinberger and L. K. Saul. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *AAAI*, 2006. 1

[24] C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *NIPS*, pages 682–688, 2000. 2, 3, 6