Journal of Machine Learning Research 14 (2013) 3129-3152 Submitted 5/12; Revised 5/13; Published 10/13

# Large-scale SVD and Manifold Learning

**Ameet Talwalkar** AMEET@CS.BERKELEY.EDU
*University of California, Berkeley*
*Division of Computer Science*
*465 Soda Hall*
*Berkeley, CA 94720*

**Sanjiv Kumar** SANJIVK@GOOGLE.COM
*Google Research*
*76 Ninth Avenue*
*New York, NY 10011*

**Mehryar Mohri** MOHRI@CS.NYU.EDU
*Courant Institute and Google Research*
*251 Mercer Street*
*New York, NY 10012*

**Henry Rowley** HAR@GOOGLE.COM
*Google Research*
*Amphitheatre Parkway*
*Mountain View, CA 94043*

**Editor:** Inderjit Dhillon

## Abstract

This paper examines the efficacy of sampling-based low-rank approximation techniques when applied to large dense kernel matrices. We analyze two common approximate singular value decomposition techniques, namely the Nyström and Column sampling methods. We present a theoretical comparison between these two methods, provide novel insights regarding their suitability for various tasks and present experimental results that support our theory. Our results illustrate the relative strengths of each method. We next examine the performance of these two techniques on the large-scale task of extracting low-dimensional manifold structure given millions of high-dimensional face images. We address the computational challenges of non-linear dimensionality reduction via Isomap and Laplacian Eigenmaps, using a graph containing about 18 million nodes and 65 million edges. We present extensive experiments on learning low-dimensional embeddings for two large face data sets: CMU-PIE (35 thousand faces) and a web data set (18 million faces). Our comparisons show that the Nyström approximation is superior to the Column sampling method for this task. Furthermore, approximate Isomap tends to perform better than Laplacian Eigenmaps on both clustering and classification with the labeled CMU-PIE data set.

**Keywords:** low-rank approximation, manifold learning, large-scale matrix factorization

## 1. Introduction

Kernel-based algorithms (Schölkopf and Smola, 2002) are a broad class of learning algorithms with rich theoretical underpinnings and state-of-the-art empirical performance for a variety of problems, for example, Support Vector Machines (SVMs) and Kernel Logistic Regression (KLR) for classifi-

cation, Support Vector Regression (SVR) and Kernel Ridge Regression (KRR) for regression, Kernel Principle Component Analysis (KPCA) for non-linear dimensionality reduction, SVM-Rank for ranking, etc. Despite the favorable properties of kernel methods in terms of theory, empirical performance and flexibility, scalability remains a major drawback. Given a set of $n$ datapoints, these algorithms require $O(n^2)$ space to store the kernel matrix. Furthermore, they often require $O(n^3)$ time, requiring matrix inversion, Singular Value Decomposition (SVD) or quadratic programming in the case of SVMs. For large-scale data sets, both the space and time requirements quickly become intractable. Various optimization methods have been introduced to speed up kernel methods, for example, SMO (Platt, 1999), shrinking (Joachims, 1999), chunking (Boser et al., 1992), parallelized SVMs (Chang et al., 2008) and parallelized KLR (Mann et al., 2009). However for large-scale problems, the storage and processing costs can nonetheless be intractable.

In this work,[1] we focus on an attractive solution to this problem that involves efficiently generating low-rank approximations to the kernel matrix. Low-rank approximation appears in a wide variety of applications including lossy data compression, image processing, text analysis and cryptography, and is at the core of widely used algorithms such as Principle Component Analysis, Multidimensional Scaling and Latent Semantic Indexing. Moreover, kernel matrices can often be well approximated by low-rank matrices, the latter of which are much easier to store and operate on. Although SVD can be used to find 'optimal' low-rank approximations, SVD requires storage of the full kernel matrix and the runtime is superlinear in $n$, and hence does not scale well for large-scale applications.

For matrices of special form such as tridiagonal matrices, fast parallelized decomposition algorithms exist (Dhillon and Parlett, 2004), but such methods are not generally applicable. Kernel functions are sometimes chosen to yield sparse kernel matrices. When dealing with these sparse matrices, iterative methods such as the Jacobi or Arnoldi techniques (Golub and Loan, 1983) can be used to compute a compact SVD. More recent methods based on random projections (Halko et al., 2009) and statistical leverage scores (Mahoney, 2011) can yield high-quality approximations more efficiently than these standard iterative methods for sparse matrices. However, all of these methods require operating on the full matrix, and in applications where the associated kernel matrices are dense, working with full kernel matrix can be intractable. For instance, given a data set of 18M data points, as in application presented in this work, storing a dense kernel matrix would require 1300TB, and even if we could somehow store it, performing $O(n^2)$ operations would be infeasible.

When working with large dense matrices, sampling-based techniques provide a powerful alternative, as they construct low-rank matrices that are nearly 'optimal' while also having linear space and time constraints with respect to $n$. In this work, we focus on two commonly used sampling-based techniques, the Nyström method (Williams and Seeger, 2000) and the Column sampling method (Frieze et al., 1998). The Nyström approximation has been studied in the machine learning community (Williams and Seeger, 2000; Drineas and Mahoney, 2005), while Column sampling techniques have been analyzed in the theoretical Computer Science community (Frieze et al., 1998; Drineas et al., 2006; Deshpande et al., 2006). However, the relationship between these approximations had not been well studied. Here we provide an extensive theoretical analysis of these algorithms, show connections between these approximations and provide a direct comparison between their performances.

1. Portions of this work have previously appeared in preliminary forms in the Conference on Vision and Pattern Recognition (Talwalkar et al., 2008) and the International Conference on Machine Learning (Kumar et al., 2009).

We then examine the performance of these two low-rank approximation techniques on the task of extracting low-dimensional manifold structure given millions of high-dimensional face images. The problem of dimensionality reduction arises in many computer 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. This makes certain applications such as $K$-means clustering more effective in the transformed space. Instead of assuming global linearity as in the case of linear dimensionality reduction techniques, manifold learning methods typically make a weaker local-linearity assumption, that is, 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 (Tenenbaum et al., 2000; Donoho and Grimes, 2003). However, there is a trade-off between improved sampling of the manifold and the computational cost of manifold learning algorithms, and we explore these computational challenges in this work.

We focus on Isomap (Tenenbaum et al., 2000) and Laplacian Eigenmaps (Belkin and Niyogi, 2001), as both methods have good theoretical properties and the differences in their approaches allow us to make interesting comparisons between dense and sparse methods. Isomap in particular involves storing and operating on a dense similarity matrix, and although the similarity matrix is not guaranteed to be positive definite,[2] sampling-based low-rank approximation is a natural approach for scalability, as previously noted by de Silva and Tenenbaum (2003) in the context of the Nyström method. Hence, in this work we evaluate the efficacy of the Nyström method and the Column sampling method on the task of large-scale manifold learning. We also discuss our efficient implementation of a scalable manifold learning pipeline that leverages modern distributed computing architecture in order to construct neighborhood graphs, calculate shortest paths within these graphs and finally compute large-scale low-rank matrix approximations.

We now summarize our main contributions. First, we show connections between two random sampling based singular value decomposition algorithms and provide the first direct comparison of their performances on a variety of approximation tasks. In particular, we show that the Column sampling method is superior for approximating singular values, singular vectors and matrix projection approximations (defined in Section 3.2), while the Nyström method is better for spectral reconstruction approximations (also defined in Section 3.2), which are most relevant in the context of low-rank approximation of large dense matrices. Second, we apply these two algorithms to the task of large-scale manifold learning and 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 (Platt, 2004).

## 2. Preliminaries

In this section, we introduce notation and present basic definitions of two of the most common sampling-based techniques for matrix approximation.

2. In the limit of infinite samples, Isomap can be viewed as an instance of Kernel PCA (Ham et al., 2004).

### 2.1 Notation and Problem Setting

For a matrix $\mathbf{T} \in \mathbb{R}^{a\times b}$, we define $\mathbf{T}^{(j)}$, $j = 1\ldots b$, as the $j$th column vector of $\mathbf{T}$ and $\mathbf{T}_{(i)}$, $i = 1\ldots a$, as the $i$th row vector of $\mathbf{T}$. We denote by $\mathbf{T}_k$ the 'best' rank-$k$ approximation to $\mathbf{T}$, that is, $\mathbf{T}_k = \text{argmin}_{\mathbf{V}\in\mathbb{R}^{a\times b},\text{rank}(\mathbf{V})=k}\|\mathbf{T}-\mathbf{V}\|_\xi$, where $\xi \in \{2, F\}$, $\|\cdot\|_2$ denotes the spectral norm and $\|\cdot\|_F$ the Frobenius norm of a matrix. Assuming that $\text{rank}(\mathbf{T}) = r$, we can write the compact Singular Value Decomposition (SVD) of this matrix as $\mathbf{T} = \mathbf{U}_T\mathbf{\Sigma}_T\mathbf{V}_T^\top$ where $\mathbf{\Sigma}_T$ is diagonal and contains the singular values of $\mathbf{T}$ sorted in decreasing order and $\mathbf{U}_T \in \mathbb{R}^{a\times r}$ and $\mathbf{V}_T \in \mathbb{R}^{b\times r}$ are corresponding the left and right singular vectors of $\mathbf{T}$. We can then describe $\mathbf{T}_k$ in terms of its SVD as $\mathbf{T}_k = \mathbf{U}_{T,k}\mathbf{\Sigma}_{T,k}\mathbf{V}_{T,k}^\top$. Let $\mathbf{K} \in \mathbb{R}^{n\times n}$ be a symmetric positive semidefinite (SPSD) kernel or Gram matrix with $\text{rank}(\mathbf{K}) = r \leq n$. We will write the SVD of $\mathbf{K}$ as $\mathbf{K} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^\top$, and the pseudo-inverse of $\mathbf{K}$ as $\mathbf{K}^+ = \sum_{t=1}^r \sigma_t^{-1}\mathbf{U}^{(t)}\mathbf{U}^{(t)\top}$. For $k < r$, $\mathbf{K}_k = \sum_{t=1}^k \sigma_t\mathbf{U}^{(t)}\mathbf{U}^{(t)\top} = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{U}_k^\top$ is the 'best' rank-$k$ approximation to $\mathbf{K}$.

We focus on generating an approximation $\widetilde{\mathbf{K}}$ of $\mathbf{K}$ based on a sample of $l \ll n$ of its columns. We assume that we sample columns uniformly without replacement as suggested by Kumar et al. (2012) and motivated by the connection between uniform sampling and matrix incoherence (Talwalkar and Rostamizadeh, 2010; Mackey et al., 2011), though various methods have been proposed to select columns (see Chapter 4 of Talwalkar (2010) for more details on various sampling schemes). Let $\mathbf{C}$ denote the $n \times l$ matrix formed by these columns and $\mathbf{W}$ the $l \times l$ matrix consisting of the intersection of these $l$ columns with the corresponding $l$ rows of $\mathbf{K}$. Note that $\mathbf{W}$ is SPSD since $\mathbf{K}$ is SPSD. Without loss of generality, the columns and rows of $\mathbf{K}$ can be rearranged based on this sampling so that $\mathbf{K}$ and $\mathbf{C}$ be written as follows:

$$\mathbf{K} = \begin{bmatrix} \mathbf{W} & \mathbf{K}_{21}^\top \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \quad \text{and} \quad \mathbf{C} = \begin{bmatrix} \mathbf{W} \\ \mathbf{K}_{21} \end{bmatrix}. \tag{1}$$

The approximation techniques discussed next use the SVD of $\mathbf{W}$ and $\mathbf{C}$ to generate approximations for $\mathbf{K}$.

### 2.2 Nyström Method

The Nyström method was first introduced as a quadrature method for numerical integration, used to approximate eigenfunction solutions (Nyström, 1928). More recently, it was presented in Williams and Seeger (2000) to speed up kernel algorithms and has been used in applications ranging from manifold learning to image segmentation (Platt, 2004; Fowlkes et al., 2004; Talwalkar et al., 2008). The Nyström method uses $\mathbf{W}$ and $\mathbf{C}$ from (1) to approximate $\mathbf{K}$. Assuming a uniform sampling of the columns, the Nyström method generates a rank-$k$ approximation $\widetilde{\mathbf{K}}$ of $\mathbf{K}$ for $k < n$ defined by:

$$\widetilde{\mathbf{K}}_k^{nys} = \mathbf{C}\mathbf{W}_k^+\mathbf{C}^\top \approx \mathbf{K}, \tag{2}$$

where $\mathbf{W}_k$ is the best $k$-rank approximation of $\mathbf{W}$ with respect to the spectral or Frobenius norm and $\mathbf{W}_k^+$ denotes the pseudo-inverse of $\mathbf{W}_k$. If we write the SVD of $\mathbf{W}$ as $\mathbf{W} = \mathbf{U}_W\mathbf{\Sigma}_W\mathbf{U}_W^\top$, then from (2) we can write

$$\widetilde{\mathbf{K}}_k^{nys} = \mathbf{C}\mathbf{U}_{W,k}\mathbf{\Sigma}_{W,k}^+\mathbf{U}_{W,k}^\top\mathbf{C}^\top = \left(\sqrt{\frac{l}{n}}\mathbf{C}\mathbf{U}_{W,k}\mathbf{\Sigma}_{W,k}^+\right)\left(\frac{n}{l}\mathbf{\Sigma}_{W,k}\right)\left(\sqrt{\frac{l}{n}}\mathbf{C}\mathbf{U}_{W,k}\mathbf{\Sigma}_{W,k}^+\right)^\top,$$

and hence the Nyström method approximates the top $k$ singular values and vectors of $\mathbf{K}$ as:

$$\widetilde{\mathbf{\Sigma}}_{nys} = \left(\frac{n}{l}\right)\mathbf{\Sigma}_{W,k} \quad \text{and} \quad \widetilde{\mathbf{U}}_{nys} = \sqrt{\frac{l}{n}}\mathbf{C}\mathbf{U}_{W,k}\mathbf{\Sigma}^{+}_{W,k}. \tag{3}$$

The time complexity of compact SVD on $\mathbf{W}$ is in $O(l^2k)$ and matrix multiplication with $\mathbf{C}$ takes $O(nlk)$, hence the total complexity of the Nyström approximation is in $O(nlk)$.

### 2.3 Column Sampling Method

The Column sampling method was introduced to approximate the SVD of any rectangular matrix (Frieze et al., 1998). It generates approximations of $\mathbf{K}$ by using the SVD of $\mathbf{C}$.[3] If we write the SVD of $\mathbf{C}$ as $\mathbf{C} = \mathbf{U}_C\mathbf{\Sigma}_C\mathbf{V}_C^\top$ then the Column sampling method approximates the top $k$ singular values ($\mathbf{\Sigma}_k$) and singular vectors ($\mathbf{U}_k$) of $\mathbf{K}$ as:

$$\widetilde{\mathbf{\Sigma}}_{col} = \sqrt{\frac{n}{l}}\mathbf{\Sigma}_{C,k} \quad \text{and} \quad \widetilde{\mathbf{U}}_{col} = \mathbf{U}_C = \mathbf{C}\mathbf{V}_{C,k}\mathbf{\Sigma}^{+}_{C,k}. \tag{4}$$

The runtime of the Column sampling method is dominated by the SVD of $\mathbf{C}$. The algorithm takes O($nlk$) time to perform compact SVD on $\mathbf{C}$, but is still more expensive than the Nyström method as the constants for SVD are greater than those for the O($nlk$) matrix multiplication step in the Nyström method.

## 3. Nyström Versus Column Sampling

Given that two sampling-based techniques exist to approximate the SVD of SPSD matrices, we pose a natural question: which method should one use to approximate singular values, singular vectors and low-rank approximations? We next analyze the form of these approximations and empirically evaluate their performance in Section 3.3.

### 3.1 Singular Values and Singular Vectors

As shown in (3) and (4), the singular values of $\mathbf{K}$ are approximated as the scaled singular values of $\mathbf{W}$ and $\mathbf{C}$, respectively. The scaling terms are quite rudimentary and are primarily meant to *compensate* for the 'small sample size' effect for both approximations. Formally, these scaling terms make the approximations in (3) and (4) unbiased estimators of the true singular values. The form of singular vectors is more interesting. The Column sampling singular vectors ($\widetilde{\mathbf{U}}_{col}$) are orthonormal since they are the singular vectors of $\mathbf{C}$. In contrast, the Nyström singular vectors ($\widetilde{\mathbf{U}}_{nys}$) are approximated by *extrapolating* the singular vectors of $\mathbf{W}$ as shown in (3), and are *not* orthonormal. As we show in Section 3.3, this adversely affects the accuracy of singular vector approximation from the Nyström method. It is possible to orthonormalize the Nyström singular vectors by using QR decomposition. Since $\widetilde{\mathbf{U}}_{nys} \propto \mathbf{C}\mathbf{U}_W\mathbf{\Sigma}^{+}_W$, where $\mathbf{U}_W$ is orthogonal and $\mathbf{\Sigma}_W$ is diagonal, this simply implies that QR decomposition creates an orthonormal span of $\mathbf{C}$ rotated by $\mathbf{U}_W$. However, the complexity of QR decomposition of $\widetilde{\mathbf{U}}_{nys}$ is the same as that of the SVD of $\mathbf{C}$. Thus, the computational cost of orthogonalizing $\widetilde{\mathbf{U}}_{nys}$ would nullify the computational benefit of the Nyström method over Column sampling.

3. The Nyström method also uses sampled columns of $\mathbf{K}$, but the Column sampling method is named so because it uses direct decomposition of $\mathbf{C}$, while the Nyström method decomposes its submatrix, $\mathbf{W}$.

### 3.2 Low-rank Approximation

Several studies have empirically shown that the accuracy of low-rank approximations of kernel matrices is tied to the performance of kernel-based learning algorithms (Williams and Seeger, 2000; Talwalkar and Rostamizadeh, 2010). Furthermore, the connection between kernel matrix approximation and the *hypothesis* generated by several widely used kernel-based learning algorithms has been theoretically analyzed (Cortes et al., 2010). Hence, accurate low-rank approximations are of great practical interest in machine learning. As discussed in Section 2.1, the optimal $\mathbf{K}_k$ is given by,

$$\mathbf{K}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{U}_k^\top = \mathbf{U}_k \mathbf{U}_k^\top \mathbf{K} = \mathbf{K} \mathbf{U}_k \mathbf{U}_k^\top$$

where the columns of $\mathbf{U}_k$ are the $k$ singular vectors of $\mathbf{K}$ corresponding to the top $k$ singular values of $\mathbf{K}$. We refer to $\mathbf{U}_k \mathbf{\Sigma}_k \mathbf{U}_k^\top$ as *Spectral Reconstruction*, since it uses both the singular values and vectors of $\mathbf{K}$, and $\mathbf{U}_k \mathbf{U}_k^\top \mathbf{K}$ as *Matrix Projection*, since it uses only singular vectors to compute the projection of $\mathbf{K}$ onto the space spanned by vectors $\mathbf{U}_k$. These two low-rank approximations are equal only if $\mathbf{\Sigma}_k$ and $\mathbf{U}_k$ contain the true singular values and singular vectors of $\mathbf{K}$. Since this is not the case for approximate methods such as Nyström and Column sampling these two measures generally give different errors. Thus, we analyze each measure separately in the following sections.

#### 3.2.1 MATRIX PROJECTION

For Column sampling using (4), the low-rank approximation via matrix projection is

$$\widetilde{\mathbf{K}}_k^{col} = \widetilde{\mathbf{U}}_{col,k} \widetilde{\mathbf{U}}_{col,k}^\top \mathbf{K} = \mathbf{U}_{C,k} \mathbf{U}_{C,k}^\top \mathbf{K} = \mathbf{C}((\mathbf{C}^\top \mathbf{C})_k)^+ \mathbf{C}^\top \mathbf{K}, \tag{5}$$

where $(\mathbf{C}^\top \mathbf{C})_k^{-1} = \mathbf{V}_{C,k} (\mathbf{\Sigma}_{C,k}^2)^+ \mathbf{V}_{C,k}^\top$. Clearly, if $k = l$, $(\mathbf{C}^\top \mathbf{C})_k = \mathbf{C}^\top \mathbf{C}$. Similarly, using (3), the Nyström matrix projection is

$$\widetilde{\mathbf{K}}_k^{nys} = \widetilde{\mathbf{U}}_{nys,k} \widetilde{\mathbf{U}}_{nys,k}^\top \mathbf{K} = \frac{l}{n} \mathbf{C} (\mathbf{W}_k^2)^+ \mathbf{C}^\top \mathbf{K}. \tag{6}$$

As shown in (5) and (6), the two methods have similar expressions for matrix projection, except that $\mathbf{C}^\top \mathbf{C}$ is replaced by a scaled $\mathbf{W}^2$. The scaling term appears only in the expression for the Nyström method. We now present Theorem 1 and Observations 1 and 2, which provide further insights about these two methods in the context of matrix projection.

**Theorem 1** *The Column sampling and Nyström matrix projections are of the form* $\mathbf{U}_C \mathbf{R} \mathbf{U}_C^\top \mathbf{K}$*, where* $\mathbf{R} \in \mathbb{R}^{l \times l}$ *is SPSD. Further, Column sampling gives the lowest reconstruction error* (*measured in* $\|\cdot\|_F$) *among all such approximations if* $k = l$.

**Observation 1** *For* $k = l$*, matrix projection for Column sampling reconstructs* $\mathbf{C}$ *exactly. This can be seen by block-decomposing* $\mathbf{K}$ *as:* $\mathbf{K} = [\mathbf{C} \quad \bar{\mathbf{C}}]$*, where* $\bar{\mathbf{C}} = [\mathbf{K}_{21} \quad \mathbf{K}_{22}]^\top$*, and using* (5)*:*

$$\widetilde{\mathbf{K}}_l^{col} = \mathbf{C}(\mathbf{C}^\top \mathbf{C})^+ \mathbf{C}^\top \mathbf{K} = [\mathbf{C} \quad \mathbf{C}(\mathbf{C}^\top \mathbf{C})^+ \mathbf{C}^\top \bar{\mathbf{C}}] = [\mathbf{C} \quad \bar{\mathbf{C}}].$$

**Observation 2** *For* $k = l$*, the span of the orthogonalized Nyström singular vectors equals the span of* $\widetilde{\mathbf{U}}_{col}$*, as discussed in Section 3.1. Hence, matrix projection is identical for Column sampling and Orthonormal Nyström for* $k = l$.

Matrix projection approximations are not necessarily symmetric and require storage of and multiplication with $\mathbf{K}$. Hence, although matrix projection is often analyzed theoretically, for large-scale problems, the storage and computational requirements may be inefficient or even infeasible.

| Data Set | Data | $n$ | $d$ | Kernel |
|---|---|---|---|---|
| PIE-2.7K | faces | 2731 | 2304 | linear |
| PIE-7K | faces | 7412 | 2304 | linear |
| MNIST | digits | 4000 | 784 | linear |
| ESS | proteins | 4728 | 16 | RBF |
| ABN | abalones | 4177 | 8 | RBF |

Table 1: Description of the data sets used in our experiments comparing sampling-based matrix approximations (Sim et al., 2002; LeCun and Cortes, 1998; Talwalkar et al., 2008). '$n$' denotes the number of points and '$d$' denotes the data dimensionality, that is, the number of features in input space.

#### 3.2.2 SPECTRAL RECONSTRUCTION

Using (3), the Nyström spectral reconstruction is:

$$\widetilde{\mathbf{K}}_k^{nys} = \widetilde{\mathbf{U}}_{nys,k}\widetilde{\mathbf{\Sigma}}_{nys,k}\widetilde{\mathbf{U}}_{nys,k}^\top = \mathbf{C}\mathbf{W}_k^+\mathbf{C}^\top. \tag{7}$$

When $k=l$, this approximation perfectly reconstructs three blocks of $\mathbf{K}$, and $\mathbf{K}_{22}$ is approximated by the Schur Complement of $\mathbf{W}$ in $\mathbf{K}$. The Column sampling spectral reconstruction has a similar form as (7):

$$\widetilde{\mathbf{K}}_k^{col} = \widetilde{\mathbf{U}}_{col,k}\widetilde{\mathbf{\Sigma}}_{col,k}\widetilde{\mathbf{U}}_{col,k}^\top = \sqrt{n/l}\,\mathbf{C}\big((\mathbf{C}^\top\mathbf{C})_k^{\frac{1}{2}}\big)^+\mathbf{C}^\top. \tag{8}$$

In contrast with matrix projection, the scaling term now appears in the Column sampling reconstruction. To analyze the two approximations, we consider an alternative characterization using the fact that $\mathbf{K} = \mathbf{X}^\top\mathbf{X}$ for some $\mathbf{X} \in \mathbb{R}^{N\times n}$. We define a zero-one sampling matrix, $\mathbf{S} \in \mathbb{R}^{n\times l}$, that selects $l$ columns from $\mathbf{K}$, that is, $\mathbf{C} = \mathbf{K}\mathbf{S}$. Further, $\mathbf{W} = \mathbf{S}^\top\mathbf{K}\mathbf{S} = \mathbf{X}'^\top\mathbf{X}'$, where $\mathbf{X}' \in \mathbb{R}^{N\times l}$ contains $l$ sampled columns of $\mathbf{X}$ and $\mathbf{X}' = \mathbf{U}_{X'}\mathbf{\Sigma}_{X'}\mathbf{V}_{X'}^\top$ is the SVD of $\mathbf{X}'$. We now present two results. Theorem 2 shows that the optimal spectral reconstruction is data dependent and may differ from the Nyström and Column sampling approximations. Moreover, Theorem 3 reveals that in certain instances the Nyström method is optimal, while the Column sampling method enjoys no such guarantee.

**Theorem 2** *Column sampling and Nyström spectral reconstructions of rank $k$ are of the form $\mathbf{X}^\top\mathbf{U}_{X',k}\mathbf{Z}\mathbf{U}_{X',k}^\top\mathbf{X}$, where $\mathbf{Z} \in \mathbb{R}^{k\times k}$ is SPSD. Further, among all approximations of this form, neither the Column sampling nor the Nyström approximation is optimal (in $\|\cdot\|_F$).*

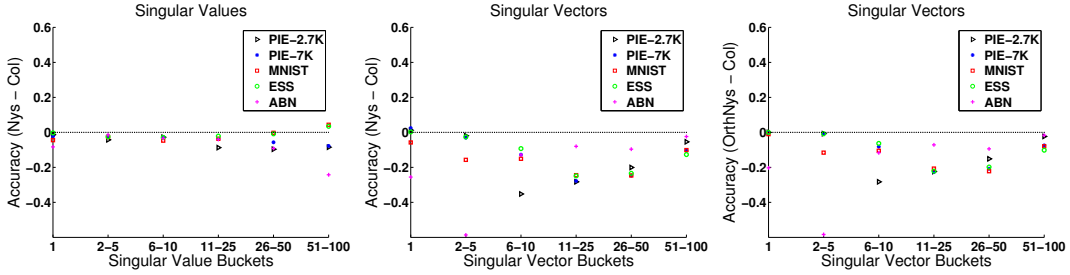
**Theorem 3** *Let $r = \mathrm{rank}(\mathbf{K}) \le k \le l$ and $\mathrm{rank}(\mathbf{W}) = r$. Then, the Nyström approximation is exact for spectral reconstruction. In contrast, Column sampling is exact iff $\mathbf{W} = \big((l/n)\mathbf{C}^\top\mathbf{C}\big)^{1/2}$.*

### 3.3 Empirical Comparison

To test the accuracy of singular values/vectors and low-rank approximations for different methods, we used several kernel matrices arising in different applications, as described in Table 3.3. We worked with data sets containing less than ten thousand points to be able to compare with exact SVD. We fixed $k$ to be 100 in all the experiments, which captures more than 90% of the spectral energy for each data set.

For singular values, we measured percentage accuracy of the approximate singular values with respect to the exact ones. For a fixed $l$, we performed 10 trials by selecting columns uniformly at random from $\mathbf{K}$. We show in Figure 1(a) the difference in mean percentage accuracy for the two methods for $l = n/10$, with results bucketed by groups of singular values, that is, we sorted the singular values in descending order, grouped them as indicated in the figure, and report the average percentage accuracy for each group. The empirical results show that the Column sampling method generates more accurate singular values than the Nyström method. A similar trend was observed for other values of $l$.

For singular vectors, the accuracy was measured by the dot product, that is, cosine of principal angles between the exact and the approximate singular vectors. Figure 1(b) shows the difference in mean accuracy between Nyström and Column sampling methods, once again bucketed by groups of singular vectors sorted in descending order based on their corresponding singular values. The top 100 singular vectors were all better approximated by Column sampling for all data sets. This trend was observed for other values of $l$ as well. Furthermore, even when the Nyström singular vectors are orthogonalized, the Column sampling approximations are superior, as shown in Figure 1(c).




(a) (b) (c)

Figure 1: Comparison of singular values and vectors (values above zero indicate better performance of Nyström). (a) Top 100 singular values with $l = n/10$. (b) Top 100 singular vectors with $l = n/10$. (c) Comparison using orthogonalized Nyström singular vectors.

Next we compared the low-rank approximations generated by the two methods using matrix projection and spectral reconstruction. We measured the accuracy of reconstruction relative to the optimal rank-$k$ approximation, $\mathbf{K}_k$, via relative accuracy $= \frac{\|\mathbf{K}-\mathbf{K}_k\|_F}{\|\mathbf{K}-\widetilde{\mathbf{K}}_k^{nys/col}\|_F}$, which will approach one for good approximations. As motivated by Theorem 1, Column sampling generates better reconstructions via matrix projection (Figure 2(a)). In contrast, the Nyström method produces superior results for spectral reconstruction (Figure 2(b)). These results may appear somewhat surprising given the relatively poor quality of the singular values/vectors for the Nyström method, but they are in agreement with the consequences of Theorem 3 and the fact that the kernel matrices we consider (aside from 'DEXT') are nearly low-rank. Moreover, the performance of these two approximations are indeed tied to the spectrum of $\mathbf{K}$ as stated in Theorem 2. Indeed, we found that the two approximations were roughly equivalent for a sparse kernel matrix with slowly decaying spectrum ('DEXT' in Figure 2(b)), while the Nyström method was superior for dense kernel matrices with exponentially decaying spectra arising from the other data sets used in the experiments.

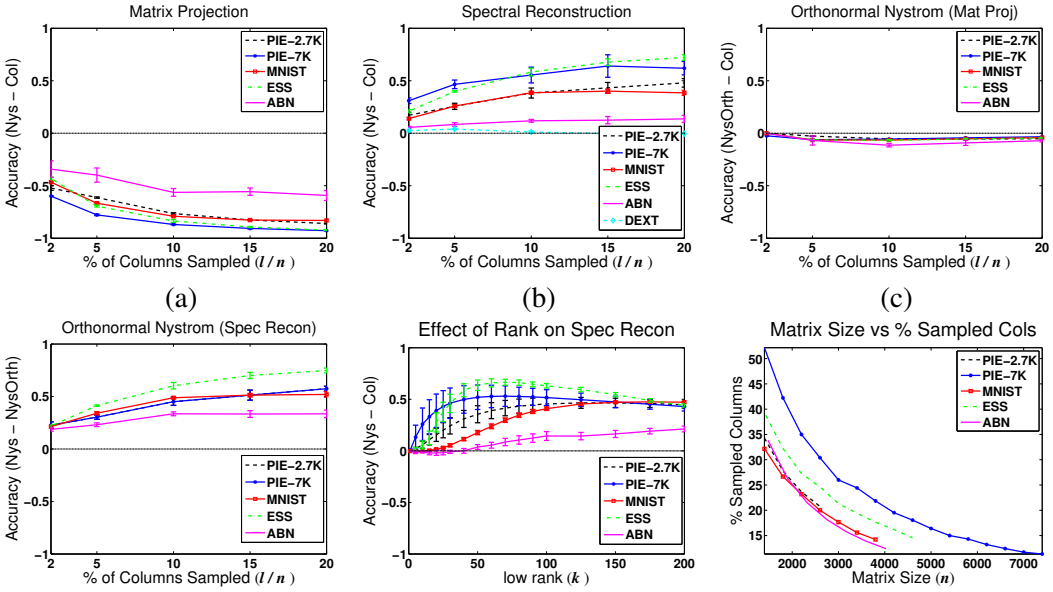



Figure 2: Plots (a)-(e) compare performance of various matrix approximations, and in all cases values above zero indicate better performance of the first listed method. (a) Matrix projection, Nyström versus Column sampling, $k = 100$. (b) Spectral reconstruction, Nyström versus Column sampling, $k = 100$. (c) Matrix projection, Orthonormal Nyström versus Column sampling, $k = 100$. (d) Spectral reconstruction, Nyström versus Orthonormal Nyström, $k = 100$. (e) Spectral reconstruction, Nyström versus Column sampling varying ranks. (f) Percentage of columns $(l/n)$ needed to achieve 75% relative accuracy for Nyström spectral reconstruction as a function of $n$.

The non-orthonormality of the Nyström method's singular vectors (Section 3.1) is one factor that impacts its accuracy for some tasks. When orthonormalized, the Nyström matrix projection error is reduced considerably as shown in Figure 2(c), and supported by Observation 2. Also, the accuracy of Orthonormal Nyström spectral reconstruction is worse relative to the standard Nyström approximation, as shown in Figure 2(d). This result can be attributed to the fact that orthonormalization of the singular vectors leads to the loss of some of the unique properties described in Section 3.2.2. For instance, Theorem 3 no longer holds and the scaling terms do not cancel out, that is, $\widetilde{\mathbf{K}}_k^{nys} \neq \mathbf{C}\mathbf{W}_k^{+}\mathbf{C}^{\top}$.

We next tested the accuracy of spectral reconstruction for the two methods for varying values of $k$ and a fixed $l$. We found that the Nyström method outperforms Column sampling across all tested values of $k$, as shown in Figure 2(e). Next, we addressed another basic issue: how many columns do we need to obtain reasonable reconstruction accuracy? We performed an experiment in which we fixed $k$ and varied the size of our data set ($n$). For each $n$, we performed grid search over $l$ to find the minimal $l$ for which the relative accuracy of Nyström spectral reconstruction was at least 75%.

Figure 2(f) shows that the required percentage of columns ($l/n$) decreases quickly as $n$ increases, lending support to the use of sampling-based algorithms for large-scale data.

Finally, we note another important distinction between the Nyström method and Column sampling, namely, out-of-sample extension. Out-of-sample extension is often discussed in the context of manifold learning, in which case it involves efficiently deriving a low-dimensional embedding for an arbitrary test point given embeddings from a set of training points (rather than rerunning the entire manifold learning algorithm). The Nyström method naturally lends itself to such out-of-sample extension, as a new point can be processed based on extrapolating from the sampled points (de Silva and Tenenbaum, 2003). Moreover, it is possible to use Column sampling to learn embeddings on the initial sample, and then use the Nyström method for subsequent out-of-sample-extension. Hence, given a large set of samples, both the Nyström method and Column sampling are viable options to enhance the scalability of manifold learning methods, as we will explore in Section 4.

## 4. Large-scale Manifold Learning

In the previous section, we discussed two sampling-based techniques that generate approximations for kernel matrices. Although we analyzed the effectiveness of these techniques for approximating singular values, singular vectors and low-rank matrix reconstruction, we have yet to discuss the effectiveness of these techniques in the context of actual machine learning tasks. In fact, the Nyström method has been shown to be successful on a variety of learning tasks including Support Vector Machines (Fine and Scheinberg, 2002), Gaussian Processes (Williams and Seeger, 2000), Spectral Clustering (Fowlkes et al., 2004), Kernel Ridge Regression (Cortes et al., 2010) and more generally to approximate regularized matrix inverses via the Woodbury approximation (Williams and Seeger, 2000). In this section, we will discuss how approximate embeddings can be used in the context of manifold learning, relying on the sampling based algorithms from the previous section to generate an approximate SVD. We present the largest study to date for manifold learning, and provide a quantitative comparison of Isomap and Laplacian Eigenmaps for large scale face manifold construction on clustering and classification tasks.

### 4.1 Manifold Learning

Manifold learning aims to extract low-dimensional structure from high-dimensional data. Given $n$ input points, $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ and $\mathbf{x}_i \in \mathbb{R}^d$, the goal is to find corresponding outputs $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^n$, where $\mathbf{y}_i \in \mathbb{R}^k$, $k \ll d$, such that $\mathbf{Y}$ 'faithfully' represents $\mathbf{X}$. Several manifold learning techniques have been proposed, for example, Isomap (Tenenbaum et al., 2000), Laplacian Eigenmaps (Belkin and Niyogi, 2001), Local Linear Embedding (LLE) (Roweis and Saul, 2000), Hessian Eigenmaps (Donoho and Grimes, 2003), Structural Preserving Embedding (SPE) (Shaw and Jebara, 2009) and Semidefinite Embedding (SDE) (Weinberger and Saul, 2006). Isomap aims to preserve all pair-wise geodesic distances, whereas LLE, Laplacian Eigenmaps and Hessian Eigenmaps focus on preserving local neighborhood relationships. SDE and SPE are both formulated as instances of semidefinite programming, and are prohibitively expensive for large-scale problems. We will focus on the Isomap and Laplacian Eigenmaps algorithms as they are well-studied and highlight the differences between global versus local manifold learning techniques. We next briefly review the main computational efforts required for both algorithms, which involve neighborhood graph construction and manipulation and SVD of a symmetric similarity matrix.

#### 4.1.1 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 (Tenenbaum et al., 2000). It approximates the geodesic distance assuming that input space distance provides good approximations for nearby points, and for faraway points it estimates distance as a series of hops between neighboring points. Isomap involves three steps: i) identifying $t$ nearest neighbors for each point and constructing the associated undirected neighborhood graph in $O(n^2)$ time, ii) computing approximate geodesic distances via the neighborhood graph and converting these distances into a similarity matrix $\mathbf{K}$ via double centering, which overall requires $O(n^2 \log n)$ time, and iii) calculating the final embedding of the form $\mathbf{Y} = (\mathbf{\Sigma}_k)^{1/2}\mathbf{U}_k^\top$, where $\mathbf{\Sigma}_k$ is the diagonal matrix of the top $k$ singular values of $\mathbf{K}$ and $\mathbf{U}_k$ are the associated singular vectors, which requires $O(n^2)$ space for storing $\mathbf{K}$, and $O(n^3)$ time for its SVD. The time and space complexities for all three steps are intractable for $n = 18$M.

#### 4.1.2 LAPLACIAN EIGENMAPS

Laplacian Eigenmaps aims to find a low-dimensional representation that best preserves local neighborhood relations (Belkin and Niyogi, 2001). The algorithm first computes $t$ nearest neighbors for each point, from which it constructs a sparse weight matrix $\mathbf{W}$.[4] It then minimizes an objective function that penalizes nearby inputs for being mapped to faraway outputs, with 'nearness' measured by the weight matrix $\mathbf{W}$, and the solution to the objective is the bottom singular vectors of the symmetrized, normalized form of the graph Laplacian, $\mathcal{L}$. The runtime of this algorithm is dominated by computing nearest neighbors, since the subsequent steps involve sparse matrices. In particular, $\mathcal{L}$ can be stored in $O(tn)$ space, and iterative methods, such as Lanczos, can be used to compute its compact SVD relatively quickly.

### 4.2 Approximation Experiments

Since we use sampling-based SVD approximation to scale Isomap, we first examined how well the Nyström and Column sampling methods approximated our desired low-dimensional embeddings, that is, $\mathbf{Y} = (\mathbf{\Sigma}_k)^{1/2}\mathbf{U}_k^\top$. Using (3), the Nyström low-dimensional embeddings are:

$$\widetilde{\mathbf{Y}}^{nys} = \widetilde{\mathbf{\Sigma}}_{nys,k}^{1/2}\widetilde{\mathbf{U}}_{nys,k}^\top = \left((\mathbf{\Sigma}_W)_k^{1/2}\right)^+ \mathbf{U}_{W,k}^\top \mathbf{C}^\top.$$

Similarly, from (4) we can express the Column sampling low-dimensional embeddings as:

$$\widetilde{\mathbf{Y}}^{col} = \widetilde{\mathbf{\Sigma}}_{col,k}^{1/2}\widetilde{\mathbf{U}}_{col,k}^\top = \sqrt[4]{\frac{n}{l}}\left((\mathbf{\Sigma}_C)_k^{1/2}\right)^+ \mathbf{V}_{C,k}^\top \mathbf{C}^\top.$$

Both approximations are of a similar form. Further, notice that the optimal low-dimensional embeddings are in fact the square root of the optimal rank $k$ approximation to the associated SPSD matrix, that is, $\mathbf{Y}^\top\mathbf{Y} = \mathbf{K}_k$, for Isomap. As such, there is a connection between the task of approximating low-dimensional embeddings and the task of generating low-rank approximate spectral reconstructions, as discussed in Section 3.2.2. Recall that the theoretical analysis in Section 3.2.2

4. The weight matrix should not be confused with the subsampled SPSD matrix, $\mathbf{W}$, associated with the Nyström method. Since sampling-based approximation techniques will not be used with Laplacian Eigenmaps, the notation should be clear from the context.

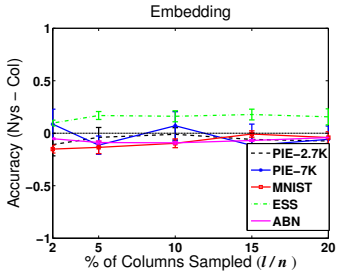



Figure 3: Comparison of embeddings (values above zero indicate better performance of Nyström).

as well as the empirical results in Section 3.3 both suggested that the Nyström method was superior in its spectral reconstruction accuracy. Hence, we performed an empirical study using the data sets from Table 3.3 to measure the quality of the low-dimensional embeddings generated by the two techniques and see if the same trend exists.

We measured the quality of the low-dimensional embeddings by calculating the extent to which they preserve distances, which is the appropriate criterion in the context of manifold learning. For each data set, we started with a kernel matrix, $\mathbf{K}$, from which we computed the associated $n \times n$ squared distance matrix, $\mathbf{D}$, using the fact that $\|\mathbf{x}_i - \mathbf{x}_j\|^2 = \mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij}$. We then computed the approximate low-dimensional embeddings using the Nyström and Column sampling methods, and then used these embeddings to compute the associated approximate squared distance matrix, $\widetilde{\mathbf{D}}$. We measured accuracy using the notion of relative accuracy defined in Section 3.3.

In our experiments, we set $k = 100$ and used various numbers of sampled columns, ranging from $l = n/50$ to $l = n/5$. Figure 3 presents the results of our experiments. Surprisingly, we do not see the same trend in our empirical results for embeddings as we previously observed for spectral reconstruction, as the two techniques exhibit roughly similar behavior across data sets. As a result, we decided to use both the Nyström and Column sampling methods for our subsequent manifold learning study.

### 4.3 Large-scale Learning

In this section, we outline the process of learning a manifold of faces. We first describe the data sets used in our experiments. We then explain 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.[5]

#### 4.3.1 DATA SETS

We used two faces data sets consisting of 35K and 18M images. The CMU PIE face data set (Sim et al., 2002) contains 41,368 images of 68 subjects under 13 different poses and various illumination conditions. A standard face detector extracted 35,247 faces (each $48 \times 48$ pixels), which comprised our 35K set (PIE-35K). Being labeled, this set allowed us to perform quantitative comparisons. The

5. To run Laplacian Eigenmaps, we generated $\mathbf{W}$ from nearest neighbor data for the largest component of the neighborhood graph and used a sparse eigensolver to compute the bottom eigenvalues of $\mathcal{L}$.

second data set, named Webfaces-18M, contains 18.2 million images extracted from the Web using the same face detector. For both data sets, face images were represented as 2304 dimensional pixel vectors that were globally normalized to have zero mean and unit variance. No other pre-processing, for example, face alignment, was performed. In contrast, He et al. (2005) used well-aligned faces (as well as much smaller data sets) to learn face manifolds. Constructing Webfaces-18M, including face detection and duplicate removal, took 15 hours using a cluster of 500 machines. We used this cluster for all experiments requiring distributed processing and data storage.

#### 4.3.2 NEAREST NEIGHBORS AND NEIGHBORHOOD GRAPH

The cost of naive nearest neighbor computation is $O(n^2)$, where $n$ is the size of the data set. 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 (Liu et al., 2004) to get approximate neighbors. Computing 5 nearest neighbors in parallel with spill trees took $\sim$2 days on the cluster. Figure 4(a) 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 (Allauzen et al., 2007).

For neighborhood graph construction, an 'appropriate' 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 adversely impact these computations. This is known as the problem of leakage or 'short-circuits' (Balasubramanian and Schwartz, 2002). 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 $95^{\text{th}}$ percentile of neighbor distances in the PIE-35K data set. Figure 4(b) 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. See Appendix D for visualizations of various components of the neighborhood graph.

#### 4.3.3 APPROXIMATING GEODESICS

To construct the similarity matrix $\mathbf{K}$ in Isomap, one approximates geodesic distance by shortest-path lengths between 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, we need only $l \ll n$ columns of $\mathbf{K}$, which form the submatrix $\mathbf{C}$. 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 and stored the output in a distributed fashion. The overall procedure took 60 minutes for Webfaces-18M using $l = 10$K. The

(a)

| | 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 % |

(b)

Figure 4: (a) Visualization of neighbors for Webfaces-18M. The first image in each row is the input, and the next five are its neighbors. (b) 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.

bottom four rows in Figure 5 show sample shortest paths for images within the largest component for Webfaces-18M, illustrating smooth transitions between images along each path.[6]

#### 4.3.4 GENERATING LOW-DIMENSIONAL EMBEDDINGS

Before generating low-dimensional embeddings using Isomap, one needs to convert distances into similarities using a process called centering (Tenenbaum et al., 2000). For the Nyström approximation, we computed $\mathbf{W}$ by double centering $\mathbf{D}$, the $l \times l$ matrix of squared geodesic distances between all landmark nodes, as $\mathbf{W} = -\frac{1}{2}\mathbf{HDH}$, where $\mathbf{H} = \mathbf{I}_l - \frac{1}{l}\mathbf{1}\mathbf{1}^\top$ is the centering matrix, $\mathbf{I}_l$ is the $l \times l$ identity matrix and $\mathbf{1}$ is a column vector of all ones. Similarly, the matrix $\mathbf{C}$ was obtained from squared geodesic distances between the landmark nodes and all other nodes using single-centering as described in de Silva and Tenenbaum (2003).

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

### 4.4 Manifold Evaluation

Manifold learning techniques typically transform the data such that Euclidean distance in the transformed space between *any* pair of points is meaningful, under the assumption that in the original space Euclidean distance is meaningful only in local neighborhoods. 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 data set, we first focused on quantitative evaluation of different

6. Our techniques for approximating geodesic distances via shortest path are used by Google for its "People Hopper" application which runs on the social networking site Orkut (Kumar and Rowley, 2010).

| Methods | Purity (%) | Accuracy (%) |
|---|---|---|
| PCA | 54.3 (±0.8) | 46.1 (±1.4) |
| Isomap | 58.4 (±1.1) | 53.3 (±4.3) |
| Nys-Iso | 59.1 (±0.9) | 53.3 (±2.7) |
| Col-Iso | 56.5 (±0.7) | 49.4 (±3.8) |
| Lap. Eig. | 35.8 (±5.0) | 69.2 (±10.8) |

(a)

| Methods | Purity (%) | Accuracy (%) |
|---|---|---|
| PCA | 54.6 (±1.3) | 46.8 (±1.3) |
| Nys-Iso | 59.9 (±1.5) | 53.7 (±4.4) |
| Col-Iso | 56.1 (±1.0) | 50.7 (±3.3) |
| Lap. Eig. | 39.3 (±4.9) | 74.7 (±5.1) |

(b)

Table 2: $K$-means clustering of face poses. Results are averaged over 10 random $K$-means initializations. (a) PIE-10K. (b) PIE-35K.

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.

We first compared different Isomap approximations to exact Isomap, using a subset of PIE with 10K images (PIE-10K) so that exact SVD required by Isomap was feasible. 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. Table 2(a) shows that clustering with Nyström Isomap with just $l = 1$K performs almost as well as exact Isomap on this data set. Column sampling Isomap performs slightly worse than Nyström Isomap. The clustering results on the full PIE-35K set (Table 2(b)) with $l = 10$K affirm this observation. As illustrated by Figure 7 (Appendix E), the Nyström method separates the pose clusters better than Column sampling verifying the quantitative results in Table 2.

One possible reason for the poor performance of Column sampling Isomap is due to the form of the similarity matrix $\mathbf{K}$. When using a finite number of data points for Isomap, $\mathbf{K}$ is not guaranteed to be SPSD (Ham et al., 2004). We verified that $\mathbf{K}$ was not SPSD in our experiments, and a significant number of top eigenvalues, that is, 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 $\mathbf{W}$ to yield signed eigenvalues, making it possible to discard the negative eigenvalues and the corresponding eigenvectors. 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 $\mathbf{C}$ (or EVD of $\mathbf{C}^\top\mathbf{C}$). Thus, the presence of negative eigenvalues deteriorates the performance of Column sampling method more than the Nyström method.

Table 2(a) and 2(b) also show a significant difference in the Isomap and Laplacian Eigenmaps results. The 2D embeddings of PIE-35K (Figure 7 in Appendix E) reveal that Laplacian Eigenmaps projects data points into a small compact region. 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 data sets verify that the manifold of faces is not linear in the input space.

| Methods | $K=1$ | $K=3$ | $K=5$ |
|---|---|---|---|
| Isomap | 10.9 ($\pm$0.5) | 14.1 ($\pm$0.7) | 15.8 ($\pm$0.3) |
| Nys-Iso | 11.0 ($\pm$0.5) | 14.0 ($\pm$0.6) | 15.8 ($\pm$0.6) |
| Col-Iso | 12.0 ($\pm$0.4) | 15.3 ($\pm$0.6) | 16.6 ($\pm$0.5) |
| Lap. Eig. | 12.7 ($\pm$0.7) | 16.6 ($\pm$0.5) | 18.9 ($\pm$0.9) |

(a)

| Nys-Iso | Col-Iso | Lap. Eig. |
|---|---|---|
| 9.8 ($\pm$0.2) | 10.3 ($\pm$0.3) | 11.1 ($\pm$0.3) |

(b)

Table 3: Nearest neighbor face pose classification error (%). Results are averaged over 10 random splits of training and test sets. (a) PIE-10K with $K = \{1,3,5\}$ neighbors. (b) PIE-35K with $K = 1$ neighbors.

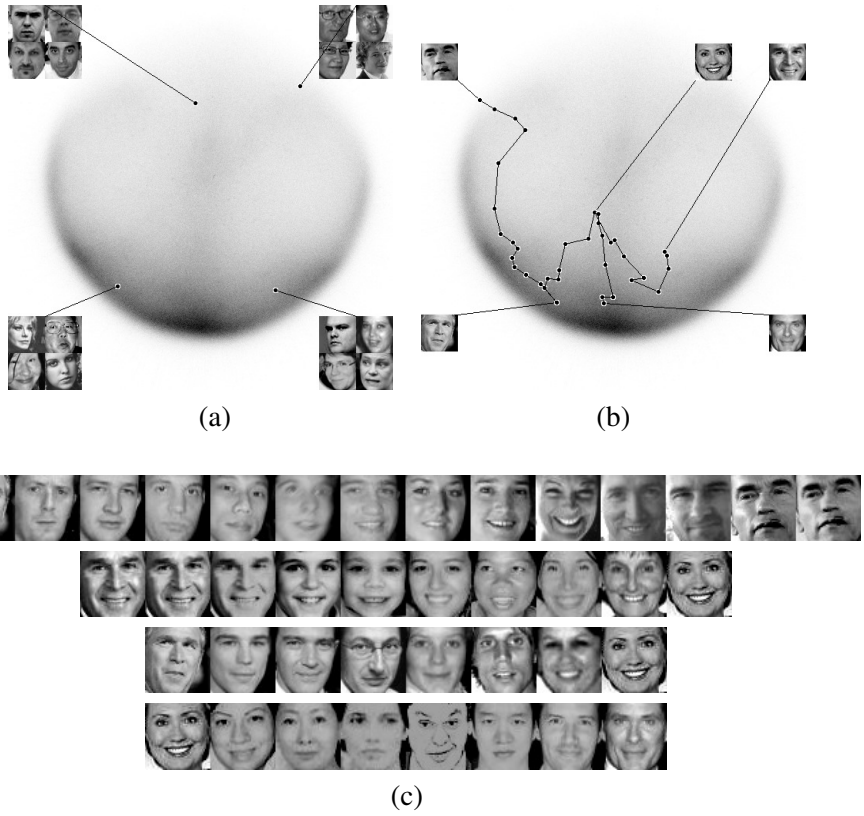


Figure 5: 2D embedding of Webfaces-18M using Nyström Isomap (Top row). Darker areas indicate denser manifold regions. (a) Face samples at different locations on the manifold. (b) Approximate geodesic paths between celebrities. (c) Visualization of paths shown in (b).

Moreover, we compared the performance of Laplacian Eigenmaps and Isomap embeddings on pose classification.[7] The data was randomly split into a training and a test set, and $K$-Nearest Neigh-

7. KNN only uses nearest neighbor information for classification. Since neighborhoods are considered to be locally linear in the input space, we expect KNN to perform well in the input space. Hence, using KNN to compare low-level embeddings indirectly measures how well nearest neighbor information is preserved.

bor (KNN) was used for classification. $K = 1$ gives lower error than higher $K$ as shown in Table 3(a). 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 (Table 3). Note that, similar to clustering, the Nyström approximation performs as well as Exact Isomap (Table 3(a)). Better clustering and classification results (along with superior 2D visualizations as shown in Appendix E), 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 5 shows the 2D embeddings from Nyström Isomap. The top left figure shows the face samples from 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 $k$, for example, $k = 100$.

In Figure 5, 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 either (i) 18M faces are perhaps not enough samples to learn the face manifold exactly, or (ii) a low-dimensional manifold of faces may not actually exist (perhaps the data clusters into multiple low dimensional manifolds). It remains an open question as to how we can measure and evaluate these hypotheses, since even very large scale testing has not provided conclusive evidence.

## 5. Conclusion

We have studied sampling based low-rank approximation algorithms, presenting an analysis of two techniques for approximating SVD on large dense SPSD matrices and providing a theoretical and empirical comparison. Although the Column sampling method generates more accurate singular values, singular vectors and low-rank matrix projections, the Nyström method constructs better low-rank approximations, which are of great practical interest as they do not use the full matrix. Furthermore, our large-scale manifold learning studies illustrate the applicability of these algorithms when working with large dense kernel matrices, reveal that Isomap coupled with the Nyström approximation can effectively extract low-dimensional structure from data sets containing millions of images.

## Appendix A. Proof of Theorem 1

**Proof** From (5), it is easy to see that

$$\widetilde{\mathbf{K}}_k^{col} = \mathbf{U}_{C,k}\mathbf{U}_{C,k}^\top\mathbf{K} = \mathbf{U}_C\mathbf{R}_{col}\mathbf{U}_C^\top\mathbf{K},$$

where $\mathbf{R}_{col} = \left[\begin{smallmatrix} \mathbf{I}_k & 0 \\ 0 & 0 \end{smallmatrix}\right]$. Similarly, from (6) we can derive

$$\widetilde{\mathbf{K}}_k^{nys} = \mathbf{U}_C \mathbf{R}_{nys} \mathbf{U}_C^\top \mathbf{K} \text{ where } \mathbf{R}_{nys} = \mathbf{Y}(\mathbf{\Sigma}_{W,k}^2)^+ \mathbf{Y}^\top,$$

and $\mathbf{Y} = \sqrt{l/n}\mathbf{\Sigma}_C \mathbf{V}_C^\top \mathbf{U}_{W,k}$. Note that both $\mathbf{R}_{col}$ and $\mathbf{R}_{nys}$ are SPSD matrices. Furthermore, if $k = l$, $\mathbf{R}_{col} = \mathbf{I}_l$. Let $\mathbf{E}$ be the (squared) reconstruction error for an approximation of the form $\mathbf{U}_C \mathbf{R} \mathbf{U}_C^\top \mathbf{K}$, where $\mathbf{R}$ is an arbitrary SPSD matrix. Hence, when $k = l$, the difference in reconstruction error between the generic and the Column sampling approximations is

$$\begin{aligned}
\mathbf{E} - \mathbf{E}_{col} =& \|\mathbf{K} - \mathbf{U}_C \mathbf{R} \mathbf{U}_C^\top \mathbf{K}\|_F^2 - \|\mathbf{K} - \mathbf{U}_C \mathbf{U}_C^\top \mathbf{K}\|_F^2 \\
=& \operatorname{Tr}\left[\mathbf{K}^\top (\mathbf{I}_n - \mathbf{U}_C \mathbf{R} \mathbf{U}_C^\top)^\top (\mathbf{I}_n - \mathbf{U}_C \mathbf{R} \mathbf{U}_C^\top) \mathbf{K}\right] \\
& - \operatorname{Tr}\left[\mathbf{K}^\top (\mathbf{I}_n - \mathbf{U}_C \mathbf{U}_C^\top)^\top (\mathbf{I}_n - \mathbf{U}_C \mathbf{U}_C^\top) \mathbf{K}\right] \\
=& \operatorname{Tr}\left[\mathbf{K}^\top (\mathbf{U}_C \mathbf{R}^2 \mathbf{U}_C^\top - 2\mathbf{U}_C \mathbf{R} \mathbf{U}_C^\top + \mathbf{U}_C \mathbf{U}_C^\top) \mathbf{K}\right] \\
=& \operatorname{Tr}\left[((\mathbf{R} - \mathbf{I}_n)\mathbf{U}_C^\top \mathbf{K})^\top ((\mathbf{R} - \mathbf{I}_n)\mathbf{U}_C^\top \mathbf{K})\right] \\
\geq& 0.
\end{aligned}$$

We used the facts that $\mathbf{U}_C^\top \mathbf{U}_C = \mathbf{I}_n$ and $\mathbf{A}^\top \mathbf{A}$ is SPSD for any matrix $\mathbf{A}$. ■

## Appendix B. Proof of Theorem 2

**Proof** If $\alpha = \sqrt{n/l}$, then starting from (8) and expressing $\mathbf{C}$ and $\mathbf{W}$ in terms of $\mathbf{X}$ and $\mathbf{S}$, we have

$$\begin{aligned}
\widetilde{\mathbf{K}}_k^{col} =& \alpha \mathbf{K}\mathbf{S}((\mathbf{S}^\top \mathbf{K}^2 \mathbf{S})_k^{1/2})^+ \mathbf{S}^\top \mathbf{K}^\top \\
=& \alpha \mathbf{X}^\top \mathbf{X}' \left((\mathbf{V}_{C,k} \mathbf{\Sigma}_{C,k}^2 \mathbf{V}_{C,k}^\top)^{1/2}\right)^+ \mathbf{X}'^\top \mathbf{X} \\
=& \mathbf{X}^\top \mathbf{U}_{X',k} \mathbf{Z}_{col} \mathbf{U}_{X',k}^\top \mathbf{X},
\end{aligned}$$

where $\mathbf{Z}_{col} = \alpha \mathbf{\Sigma}_{X'} \mathbf{V}_{X'}^\top \mathbf{V}_{C,k} \mathbf{\Sigma}_{C,k}^+ \mathbf{V}_{C,k}^\top \mathbf{V}_{X'} \mathbf{\Sigma}_{X'}$. Similarly, from (7) we have:

$$\begin{aligned}
\widetilde{\mathbf{K}}_k^{nys} =& \mathbf{K}\mathbf{S}(\mathbf{S}^\top \mathbf{K}\mathbf{S})_k^+ \mathbf{S}^\top \mathbf{K}^\top \\
=& \mathbf{X}^\top \mathbf{X}' \left(\mathbf{X}'^\top \mathbf{X}'\right)_k^+ \mathbf{X}'^\top \mathbf{X} \\
=& \mathbf{X}^\top \mathbf{U}_{X',k} \mathbf{U}_{X',k}^\top \mathbf{X}.
\end{aligned} \tag{9}$$

Clearly, $\mathbf{Z}_{nys} = \mathbf{I}_k$. Next, we analyze the error, $\mathbf{E}$, for an arbitrary $\mathbf{Z}$, which yields the approximation $\widetilde{\mathbf{K}}_k^Z$:

$$\mathbf{E} = \|\mathbf{K} - \widetilde{\mathbf{K}}_k^Z\|_F^2 = \|\mathbf{X}^\top (\mathbf{I}_N - \mathbf{U}_{X',k} \mathbf{Z} \mathbf{U}_{X',k}^\top) \mathbf{X}\|_F^2.$$

Let $\mathbf{X} = \mathbf{U}_X \mathbf{\Sigma}_X \mathbf{V}_X^\top$ and $\mathbf{Y} = \mathbf{U}_X^\top \mathbf{U}_{X',k}$. Then,

$$\begin{aligned}
\mathbf{E} =& \operatorname{Tr}\left[\left(\mathbf{U}_X \mathbf{\Sigma}_X \mathbf{U}_X^\top (\mathbf{I}_N - \mathbf{U}_{X',k} \mathbf{Z} \mathbf{U}_{X',k}^\top) \mathbf{U}_X \mathbf{\Sigma}_X \mathbf{U}_X^\top\right)^2\right] \\
=& \operatorname{Tr}\left[\left(\mathbf{U}_X \mathbf{\Sigma}_X (\mathbf{I}_N - \mathbf{Y}\mathbf{Z}\mathbf{Y}^\top) \mathbf{\Sigma}_X \mathbf{U}_X^\top\right)^2\right] \\
=& \operatorname{Tr}\left[\mathbf{\Sigma}_X (\mathbf{I}_N - \mathbf{Y}\mathbf{Z}\mathbf{Y}^\top) \mathbf{\Sigma}_X^2 (\mathbf{I}_N - \mathbf{Y}\mathbf{Z}\mathbf{Y}^\top) \mathbf{\Sigma}_X\right] \\
=& \operatorname{Tr}\left[\mathbf{\Sigma}_X^4 - 2\mathbf{\Sigma}_X^2 \mathbf{Y}\mathbf{Z}\mathbf{Y}^\top \mathbf{\Sigma}_X^2 + \mathbf{\Sigma}_X \mathbf{Y}\mathbf{Z}\mathbf{Y}^\top \mathbf{\Sigma}_X^2 \mathbf{Y}\mathbf{Z}\mathbf{Y}^\top \mathbf{\Sigma}_X\right].
\end{aligned} \tag{10}$$


Figure 6: (a) A few random samples from the largest connected component of the Webfaces-18M neighborhood graph. (b) 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. (c) 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.

To find $\mathbf{Z}^*$, the $\mathbf{Z}$ that minimizes (10), we use the convexity of (10) and set:

$$\partial \mathbf{E}/\partial \mathbf{Z} = -2\mathbf{Y}^\top \mathbf{\Sigma}_X^4 \mathbf{Y} + 2(\mathbf{Y}^\top \mathbf{\Sigma}_X^2 \mathbf{Y})\mathbf{Z}^*(\mathbf{Y}^\top \mathbf{\Sigma}_X^2 \mathbf{Y}) = 0$$

and solve for $\mathbf{Z}^*$, which gives us:

$$\mathbf{Z}^* = (\mathbf{Y}^\top \mathbf{\Sigma}_X^2 \mathbf{Y})^+ (\mathbf{Y}^\top \mathbf{\Sigma}_X^4 \mathbf{Y})(\mathbf{Y}^\top \mathbf{\Sigma}_X^2 \mathbf{Y})^+.$$

$\mathbf{Z}^* = \mathbf{Z}_{nys} = \mathbf{I}_k$ if $\mathbf{Y} = \mathbf{I}_k$, though $\mathbf{Z}^*$ does not in general equal either $\mathbf{Z}_{col}$ or $\mathbf{Z}_{nys}$, which is clear by comparing the expressions of these three matrices.[8] Furthermore, since $\mathbf{\Sigma}_X^2 = \mathbf{\Sigma}_K$, $\mathbf{Z}^*$ depends on

8. This fact is illustrated in our experimental results for the 'DEXT' data set in Figure 2(b).

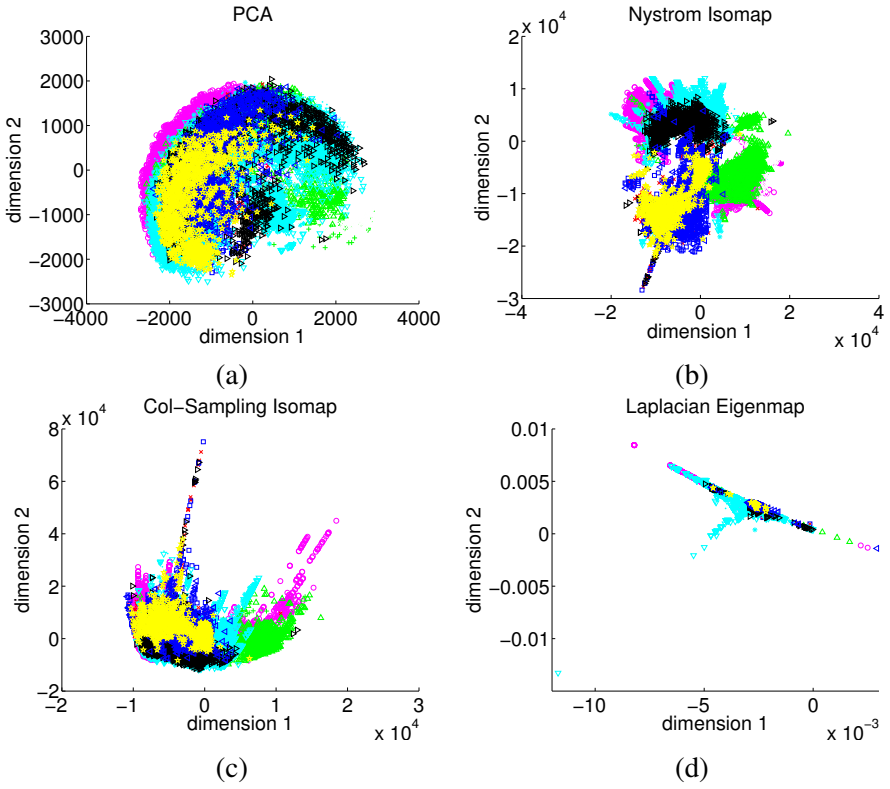



Figure 7: Optimal 2D projections of PIE-35K where each point is color coded according to its pose label. (a) PCA projections tend to spread the data to capture maximum variance. (b) Isomap projections with Nyström approximation tend to separate the clusters of different poses while keeping the cluster of each pose compact. (c) Isomap projections with Column sampling approximation have more overlap than with Nyström approximation. (d) Laplacian Eigenmaps projects the data into a very compact range.

the spectrum of $\mathbf{K}$. ■

## Appendix C. Proof of Theorem 3

**Proof** Since $\mathbf{K} = \mathbf{X}^\top \mathbf{X}$, $\text{rank}(\mathbf{K}) = \text{rank}(\mathbf{X}) = r$. Similarly, $\mathbf{W} = \mathbf{X}'^\top \mathbf{X}'$ implies $\text{rank}(\mathbf{X}') = r$. Thus the columns of $\mathbf{X}'$ span the columns of $\mathbf{X}$ and $\mathbf{U}_{X',r}$ is an orthonormal basis for $\mathbf{X}$, that is, $\mathbf{I}_N - \mathbf{U}_{X',r}\mathbf{U}_{X',r}^\top \in \text{Null}(\mathbf{X})$. Since $k \geq r$, from (9) we have

$$\|\mathbf{K} - \widetilde{\mathbf{K}}_k^{nys}\|_F = \|\mathbf{X}^\top(\mathbf{I}_N - \mathbf{U}_{X',r}\mathbf{U}_{X',r}^\top)\mathbf{X}\|_F = 0,$$

which proves the first statement of the theorem. To prove the second statement, we note that $\text{rank}(C) = r$. Thus, $\mathbf{C} = \mathbf{U}_{C,r}\mathbf{\Sigma}_{C,r}\mathbf{V}_{C,r}^\top$ and $(\mathbf{C}^\top\mathbf{C})_k^{1/2} = (\mathbf{C}^\top\mathbf{C})^{1/2} = \mathbf{V}_{C,r}\mathbf{\Sigma}_{C,r}\mathbf{V}_{C,r}^\top$ since $k \geq r$. If $\mathbf{W} = (1/\alpha)(\mathbf{C}^\top\mathbf{C})^{1/2}$, then the Column sampling and Nyström approximations are identical and

hence exact. Conversely, to exactly reconstruct $\mathbf{K}$, Column sampling necessarily reconstructs $\mathbf{C}$ exactly. Using $\mathbf{C}^\top = [\mathbf{W} \quad \mathbf{K}_{21}^\top]$ in (8) we have:

$$\begin{aligned}
\widetilde{\mathbf{K}}_k^{col} = \mathbf{K} \quad &\Longrightarrow \quad \alpha \mathbf{C} \big( (\mathbf{C}^\top \mathbf{C})_k^{\frac{1}{2}} \big)^+ \mathbf{W} = \mathbf{C} \\
&\Longrightarrow \quad \alpha \mathbf{U}_{C,r} \mathbf{V}_{C,r}^\top \mathbf{W} = \mathbf{U}_{C,r} \mathbf{\Sigma}_{C,r} \mathbf{V}_{C,r}^\top \\
&\Longrightarrow \quad \alpha \mathbf{V}_{C,r} \mathbf{V}_{C,r}^\top \mathbf{W} = \mathbf{V}_{C,r} \mathbf{\Sigma}_{C,r} \mathbf{V}_{C,r}^\top \qquad (11) \\
&\Longrightarrow \quad \mathbf{W} = \frac{1}{\alpha} (\mathbf{C}^\top \mathbf{C})^{1/2}. \qquad (12)
\end{aligned}$$

In (11) we use $\mathbf{U}_{C,r}^\top \mathbf{U}_{C,r} = \mathbf{I}_r$, while (12) follows since $\mathbf{V}_{C,r} \mathbf{V}_{C,r}^\top$ is an orthogonal projection onto the span of the rows of $\mathbf{C}$ and the columns of $\mathbf{W}$ lie within this span implying $\mathbf{V}_{C,r} \mathbf{V}_{C,r}^\top \mathbf{W} = \mathbf{W}$. ■

## Appendix D. Visualization of Connected Components in Neighborhood Graph

Figure 6(a) shows a few random samples from the largest component of the neighborhood graph we generate for Webfaces-18M. Images not within the largest component are either part of a strongly connected set of images (Figure 6(b)) or do not have any neighbors within the upper distance limit (Figure 6(c)). There are significantly more false positives in Figure 6(c) than in Figure 6(a), although some of the images in Figure 6(c) are actually faces. Clearly, the distance limit introduces a trade-off between filtering out non-faces and excluding actual faces from the largest component.

## Appendix E. Visualization of 2D Embeddings of PIE-35K

Figure 7 shows the optimal 2D projections from different methods for PIE-35K.

## References

C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri. OpenFST: A general and efficient weighted finite-state transducer library. In *Conference on Implementation and Application of Automata*, 2007.

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

M. Belkin and P. Niyogi. Laplacian Eigenmaps and spectral techniques for embedding and clustering. In *Neural Information Processing Systems*, 2001.

B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Conference on Learning Theory*, 1992.

E. Chang, K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, and H. Cui. Parallelizing support vector machines on distributed computers. In *Neural Information Processing Systems*, 2008.

C. Cortes, M. Mohri, and A. Talwalkar. On the impact of kernel approximation on learning accuracy. In *Conference on Artificial Intelligence and Statistics*, 2010.

V. de Silva and J. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Neural Information Processing Systems*, 2003.

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

I. Dhillon and B. Parlett. Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices. *Linear Algebra and its Applications*, 387:1–28, 2004.

D. Donoho and C. Grimes. Hessian Eigenmaps: locally linear embedding techniques for high dimensional data. *Proceedings of the National Academy of Sciences*, 100(10):5591–5596, 2003.

P. Drineas and M. W. Mahoney. On the Nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

P. Drineas, R. Kannan, and M. W. Mahoney. Fast Monte Carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM Journal on Computing*, 36(1), 2006.

S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2002.

C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, 2004.

A. Frieze, R. Kannan, and S. Vempala. Fast Monte Carlo algorithms for finding low-rank approximations. In *Foundation of Computer Science*, 1998.

G. Golub and C. V. Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 2nd edition, 1983. ISBN 0-8018-3772-3 (hardcover), 0-8018-3739-1 (paperback).

N. Halko, P. Martinsson, and J. Tropp. Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. `arXiv:0909.4061v1[math.NA]`, 2009.

J. Ham, D. D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *International Conference on Machine Learning*, 2004.

X. He, S. Yan, Y. Hu, and P. Niyogi. Face recognition using Laplacianfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):328–340, 2005.

T. Joachims. Making large-scale support vector machine learning practical. In *Neural Information Processing Systems*, 1999.

S. Kumar and H. Rowley. People Hopper. `http://googleresearch.blogspot.com/2010/03/hopping-on-face-manifold-via-people.html`, 2010.

S. Kumar, M. Mohri, and A. Talwalkar. On sampling-based approximate spectral decomposition. In *International Conference on Machine Learning*, 2009.

S. Kumar, M. Mohri, and A. Talwalkar. Sampling methods for the Nyström method. In *Journal of Machine Learning Research*, 2012.

Y. LeCun and C. Cortes. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 1998.

T. Liu, A. W. Moore, A. G. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. In *Neural Information Processing Systems*, 2004.

L. Mackey, A. Talwalkar, and M. I. Jordan. Divide-and-conquer matrix factorization. In *Neural Information Processing Systems*, 2011.

M. W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3(2):123–224, 2011.

G. Mann, R. McDonald, M. Mohri, N. Silberman, and D. Walker. Efficient large-scale distributed training of conditional maximum entropy models. In *Neural Information Processing Systems*, 2009.

E. Nyström. Über die praktische auflösung von linearen integralgleichungen mit anwendungen auf randwertaufgaben der potentialtheorie. *Commentationes Physico-Mathematicae*, 4(15):1–52, 1928.

J. Platt. Fast training of Support Vector Machines using sequential minimal optimization. In *Neural Information Processing Systems*, 1999.

J. Platt. Fast embedding of sparse similarity graphs. In *Neural Information Processing Systems*, 2004.

S. Roweis and L. Saul. Nonlinear dimensionality reduction by Locally Linear Embedding. *Science*, 290(5500), 2000.

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press: Cambridge, MA, 2002.

B. Shaw and T. Jebara. Structure preserving embedding. In *International Conference on Machine Learning*, 2009.

T. Sim, S. Baker, and M. Bsat. The CMU pose, illumination, and expression database. In *Conference on Automatic Face and Gesture Recognition*, 2002.

A. Talwalkar. *Matrix Approximation for Large-scale Learning*. Ph.D. thesis, Computer Science Department, Courant Institute, New York University, New York, NY, 2010.

A. Talwalkar and A. Rostamizadeh. Matrix coherence and the Nyström method. In *Conference on Uncertainty in Artificial Intelligence*, 2010.

A. Talwalkar, S. Kumar, and H. Rowley. Large-scale manifold learning. In *Conference on Vision and Pattern Recognition*, 2008.

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

K. Weinberger and L. Saul. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *AAAI Conference on Artificial Intelligence*, 2006.

C. K. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Neural Information Processing Systems*, 2000.