Locally Linear Landmarks for Large-Scale Manifold Learning

Max Vladymyrov and Miguel Á. Carreira-Perpiñán

Electrical Engineering and Computer Science, University of California, Merced, USA
{mvladymyrov,mcarreira-perpinan}@ucmerced.edu

Abstract. Spectral methods for manifold learning and clustering typically construct a graph weighted with affinities from a dataset and compute eigenvectors of a graph Laplacian. With large datasets, the eigendecomposition is too expensive, and is usually approximated by solving for a smaller graph defined on a subset of the points (landmarks) and then applying the Nyström formula to estimate the eigenvectors over all points. This has the problem that the affinities between landmarks do not benefit from the remaining points and may poorly represent the data if using few landmarks. We introduce a modified spectral problem that uses all data points by constraining the latent projection of each point to be a local linear function of the landmarks' latent projections. This constructs a new affinity matrix between landmarks that preserves manifold structure even with few landmarks, allows one to reduce the eigenproblem size, and defines a fast, nonlinear out-of-sample mapping.

Keywords: manifold learning, spectral methods, optimization.

1 Introduction

Manifold learning algorithms have long been used for exploratory analysis of a high-dimensional dataset, to reveal structure such as clustering, or as a preprocessing step to extract some low-dimensional features that are useful for classification or other tasks. Here we focus on the well-known class of spectral manifold learning algorithms [1]. The input to these algorithms is a symmetric positive (semi)definite matrix $\mathbf{A}_{N\times N}$ (affinity matrix, graph Laplacian, etc.) that contains information about the similarity between pairs of data points $\mathbf{Y} \in \mathbb{R}^{D\times N}$, and a symmetric positive definite matrix $\mathbf{B}_{N\times N}$ that usually sets the scale of the solution. Given these two matrices, we seek a solution $\mathbf{X} \in \mathbb{R}^{d\times N}$ to the following *generalized spectral problem*:

$$\min_{\mathbf{X}} \operatorname{tr}\left(\mathbf{X}\mathbf{A}\mathbf{X}^T\right) \text{ s.t. } \mathbf{X}\mathbf{B}\mathbf{X}^T = \mathbf{I}. \tag{1}$$

Within this framework it is possible to represent manifold learning methods such as Laplacian Eigenmaps (LE) [2], Kernel PCA [3], MDS [4], ISOMAP [5] and LLE [6], as well as spectral clustering [7].

H. Blockeel et al. (Eds.): ECML PKDD 2013, Part III, LNAI 8190, pp. 256–271, 2013.

The solution of the spectral problem (1) is given by $\mathbf{X} = \mathbf{U}_d^T \mathbf{B}^{-\frac{1}{2}}$, where $\mathbf{U}_d = (\mathbf{u}_1, \ldots, \mathbf{u}_d)$ are the d trailing eigenvectors of the matrix $\mathbf{C} = \mathbf{B}^{-\frac{1}{2}} \mathbf{A} \mathbf{B}^{-\frac{1}{2}}$. In large problems (large N), the computational cost means the matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ have to be sparse, and these eigenvectors are found with numerical linear algebra techniques such as restarted Arnoldi iterations [8]. The resulting cost is still large when N and d are large. The primary goal of this paper is to find fast, approximate solutions to the spectral problem (1) (and thus to LE, spectral clustering, etc.). We propose a method we call *Locally Linear Landmarks (LLL)*, based on the idea of selecting a subset of $L \ll N$ landmarks $\widetilde{\mathbf{Y}}_{L \times N}$ from the data, approximating the data manifold by a globally nonlinear but locally linear manifold around these landmarks, and then constraining the solution $\mathbf{X}$ to follow this locally linear structure. The locally linear mapping is given by a projection matrix $\mathbf{Z} \in \mathbb{R}^{L \times N}$ that satisfies

$$\mathbf{Y} \approx \widetilde{\mathbf{Y}} \mathbf{Z} \tag{2}$$

in the high-dimensional space, and by enforcing it in the low-dimensional space, we can re-express the problem (1) as a new spectral problem on a smaller number of variables L. This reduces the cost of the eigendecomposition dramatically and, as we will show, constructs affinity matrices that preserve much manifold information because the problem still involves the entire dataset. Note that LLL is not a new manifold learning method, but a fast, approximate way to solve an existing method of the form (1).

The LLL algorithm can be used for purposes beyond fast solutions of spectral problems. First, it is particularly useful for model selection. The similarity matrices $\mathbf{A}$ and $\mathbf{B}$ are usually constructed using some meta-parameters, such as a bandwidth σ of Gaussian affinities and a sparsity level K_W (number of neighbors). In practice, a user has to tune these parameters to the dataset by hand by solving the spectral problem for each parameter value. This is extremely costly with large datasets. As we will show, we can run LLL with very few landmarks so that the *shape* of the model selection curve (especially its minimum) is preserved well. This way we can identify the optimal meta-parameters much faster and then solve the spectral problem (possibly using more landmarks). Second, LLL solves the out-of-sample problem for the spectral problem (1) (which projects only the training points) by providing a natural, explicit mapping to project new points, which does not exist in the original spectral problem. Finally, we observe that the gain of LLL is much bigger when the number of eigenvectors d is large, which makes it very attractive as a preprocessing step for classification to other machine learning tasks.

Related Work. The most widespread method to find an approximate, fast solution of the spectral problem is the Nyström method [9,10,11,12]. It approximates the eigendecomposition of a large positive semidefinite matrix using the eigendecomposition of a much smaller matrix of landmarks. It can be seen as an out-of-sample extension where we first solve for the landmarks separately from the non-landmark points, and then use it to project the non-landmark points. Since, during the projection of the landmarks, the Nyström method does not use

the data from the non-landmark points, which is available from the beginning, it can result in large approximation errors if the number of landmarks is low.

It is possible to redefine the affinities between landmarks so that they use information from all points, for example by using a commute distance (the expected time it takes a random walk to travel from the first to the second node and back). Besides the fact that this solves a different spectral problem, computing these distances is costly, it provides no out-of-sample mapping, and commute distances have been shown to be problematic with large datasets in high dimensions [13]. As we will show, in LLL the affinities between landmarks use naturally the information in non-landmarks without us having to define new affinities.

Other landmark-based methods can be seen as forms of a Nyström approach. De Silva and Tenenbaum [14] suggested to run the metric MDS algorithm on a subset of the data, while the rest of the points can be located through a distance-based triangulation process. The same idea can be applied to a graph of geodesic distances (instead of Euclidean ones) which leads to the Landmark Isomap algorithm [15]. This algorithm is able to give better results due to its ability to deal with nonlinear manifolds. These approaches have been shown [10,16] to be a Nyström approximation combined with classical MDS or Isomap.

The idea of representing points by linear coding as in eq. (2) has been used in many different domains of machine learning, such as image classification [17,18], manifold learning [19,20,21], supervised [22] and semi-superwised [23] learning. In addition to linearity, many of above algorithms try to find local, sparse representations of the data, so that points are reconstructed using only nearby landmarks. An early work is the LLE method for manifold learning [19], which computes the matrix $\mathbf{Z}$ that best reconstructs each data point from a set of nearby points. Variations exist, such as using multiple local weight vectors in constructing $\mathbf{Z}$ in the MLLE algorithm [24]. However, these works use local linear mappings to define a new spectral problem, while LLL uses them to approximate an existing spectral problem. The AnchorGraph algorithm [23] uses local coding in the graph Laplacian regularization term of a semi-supervised learning problem. The problem it solves is different from (1), and does not generalize beyond the Laplacian regularizer, compared to the more general approach that we propose here. Chen and Cai [25] propose a landmarks-based approximation for spectral clustering. However, the affinities they construct entirely ignore the original affinity matrix and thus cannot be seen as approximating the target problem. Landmark SDE [20] proposes to reconstruct kernel matrix using much smaller matrix of inner products between the landmarks only. This problem is also different to ours.

Two approaches exist to construct out-of-sample mappings for spectral problems such as Laplacian eigenmaps: Bengio el al. [26] apply the Nyström formula using the affinity kernel that defined the problem. Carreira-Perpiñán and Lu [27] augment the spectral problem with the test point and solve it subject to not changing the points already embedded, which results in a kernel regression mapping. In LLL, the out-of-sample mapping is a natural subproduct of assuming each low-dimensional point to be a local linear mapping of the landmark projections associated with it.

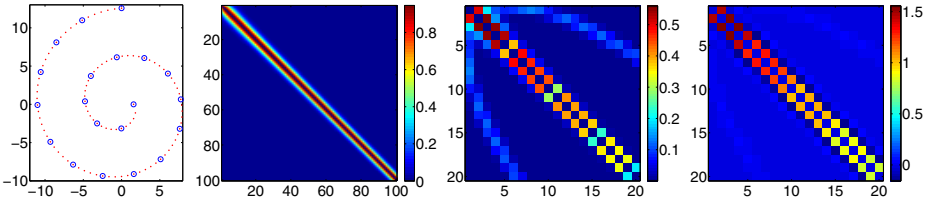

Fig. 1. Affinity matrices for landmarks in a spiral dataset. *From left to right*: 100 points along the spiral (in red) with 20 landmarks selected uniformly (in blue); the affinity matrix $\mathbf{W}$ used by LE constructed using all the points; the affinity matrix $\mathbf{W}$ built using just landmarks; the learned affinity matrix $\mathbf{C}$ of LLL using the whole dataset.

2 Solving Spectral Problems with Locally Linear Landmarks

The fundamental assumption in LLL is that the local dependence of points on landmarks that occurs in high-dimensional space, eq. (2), is preserved in the low-dimensional space:

$$\mathbf{X} \approx \widetilde{\mathbf{X}}\mathbf{Z}. \tag{3}$$

Substituting this into the spectral problem (1) gives the following *reduced spectral problem* (on dL parameters):

$$\min_{\widetilde{\mathbf{X}}} \operatorname{tr}\left(\widetilde{\mathbf{X}}\widetilde{\mathbf{A}}\widetilde{\mathbf{X}}^T\right) \text{ s.t. } \widetilde{\mathbf{X}}\widetilde{\mathbf{B}}\widetilde{\mathbf{X}}^T = \mathbf{I}, \tag{4}$$

where the matrices

$$\widetilde{\mathbf{A}} = \mathbf{Z}\mathbf{A}\mathbf{Z}^T, \qquad \widetilde{\mathbf{B}} = \mathbf{Z}\mathbf{B}\mathbf{Z}^T. \tag{5}$$

are of $L \times L$. The solution for the reduced problem is given by $\widetilde{\mathbf{X}} = \widetilde{\mathbf{U}}_d^T \widetilde{\mathbf{B}}^{-\frac{1}{2}}$, where $\widetilde{\mathbf{U}}_d$ are d trailing eigenvectors of the matrix $\widetilde{\mathbf{C}} = \widetilde{\mathbf{B}}^{-\frac{1}{2}} \widetilde{\mathbf{A}} \widetilde{\mathbf{B}}^{-\frac{1}{2}}$. After the solution for the landmarks is found, the values of $\mathbf{X}$ can be recovered by applying the formula (3) once again.

We can see the reduced problem (4) as a spectral problem for just the landmark points using a similarity matrix $\widetilde{\mathbf{A}}$ that incorporates information from the whole dataset. For example, in Laplacian Eigenmaps (see Section 4) the matrix $\mathbf{A}$ is given by the graph Laplacian of a matrix $\mathbf{W}$ of affinities (typically Gaussian). Using LLL we can dramatically improve the quality of $\mathbf{W}$ over that of constructing $\mathbf{W}$ using only the landmarks, by including additional information from the whole dataset. Fig. 1 shows the affinity matrix constructed in the usual way for a spiral dataset in the full case (using all 100 points) and using 20 landmark points versus the affinity matrix learned using LLL. The latter one (right plot) is almost perfectly banded with uniform entries. This means the connectivity pattern proceeds along the spiral, respecting its geometry, rather than across it. However, when affinities are constructed directly on landmarks that are quite distant from each other, undesirable interactions across the spiral occur.

Out-of-sample Extension. Given a new point $\mathbf{y}_0 \in \mathbb{R}^D$ that is not a part of the original dataset, we find its projection on the low-dimensional space by computing a new projection vector $\mathbf{z}_0$ for that point using K_Z landmarks around $\mathbf{y}_0$. The embedding of $\mathbf{y}_0$ is found from a linear combination of the landmark projections $\mathbf{x}_0 = \widetilde{\mathbf{X}}\mathbf{z}_0$. The cost of the out-of-sample is $\mathcal{O}(DK_Z^2 + Ld)$, which is linear for all the parameters except for K_Z, which is usually low.

Construction of the Projection Matrix $\mathbf{Z}$. Let us define the landmarks as a set $\widetilde{\mathbf{Y}} = (\widetilde{\mathbf{y}}_1, \dots, \widetilde{\mathbf{y}}_L) \in \mathbb{R}^{D \times L}$ of L points in the same space as the high-dimensional input $\mathbf{Y}$. Now each datapoint $\mathbf{y}_n$ can be expressed as a linear combination of nearby landmark points: $\mathbf{y}_n = \sum_{k=1}^{L} \widetilde{\mathbf{y}}_k z_{nk}$ where $\mathbf{z}_n$ is a local projection vector for the point $\mathbf{y}_n$. There are multiple ways to make this projection local. One can consider choosing K_Z landmarks closest to $\mathbf{y}_n$ or ϵ-balls centered around $\mathbf{y}_n$. Moreover, the choice of landmarks can be different for every n. In our experiments, we keep only the K_Z landmarks that are closest to $\mathbf{y}_n$ and use the same value of K_Z for all the points. Therefore, the projection matrix $\mathbf{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_N) \in \mathbb{R}^{L \times N}$ has only K_Z nonzero elements for every column. This matrix intuitively corresponds to the proximity of the points in the dataset to the nearby landmarks and it should be invariant to rotation, rescaling and translation. The invariance to rotation and rescaling is given by the linearity of the reconstructing matrix $\widetilde{\mathbf{Y}}\mathbf{Z}$ with respect to $\widetilde{\mathbf{Y}}$, whereas translation invariance must be enforced by constraining columns of $\mathbf{Z}$ to sum to one. This leads to the following optimization problem:

$$\min_{\mathbf{Z}} \|\mathbf{Y} - \widetilde{\mathbf{Y}}\mathbf{Z}\|^2 \text{ s.t. } \mathbf{1}^T\mathbf{Z} = \mathbf{1}^T. \tag{6}$$

Following the approach proposed in the LLE algorithm [19] we introduce a pointwise Gram matrix $\mathbf{G} \in \mathbb{R}^{L \times L}$ with elements

$$g_{ij} = (\mathbf{y}_n - \widetilde{\mathbf{y}}_i)^T(\mathbf{y}_n - \widetilde{\mathbf{y}}_j) \tag{7}$$

for every $n = 1, \dots, N$. Now, the solution to problem (6) is found by solving a linear system $\sum_{k=1}^{L} g_{jk} z_{nk} = \mathbf{1}$ and rescaling the weights so they sum to one.

Computational Complexity. This algorithm reduces the number of computations for the eigendecomposition in the solution to the problem (1). However, we also need to perform extra computations to evaluate $\mathbf{Z}$, compute auxiliary matrices (5) and perform the final multiplication (3) to recover the full embedding.

The computation of $\mathbf{Z}$ consists of computing the pointwise Gram matrix $\mathbf{G}$ and solving the linear system. $\mathbf{G}$ is sparse and has only K_Z nonzero elements in each row, so it takes $\mathcal{O}(NDK_Z^2)$ to compute it. The linear system also should be solved just for K_Z unknowns, so it takes $\mathcal{O}(NK_Z^3)$. Among the two, the computation of $\mathbf{G}$ matrix dominates because $K_Z < D$, as we will show below. Note this step is independent of the number of landmarks L. The cost of computing $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ is $\mathcal{O}(K_Z N^2)$ with dense matrices and $\mathcal{O}(K_Z Nc)$ with sparse matrices, where c is some constant that depends on the sparsity of the matrices $\mathbf{A}$ and $\mathbf{B}$ and on

the particular location of the nonzero elements in $\mathbf{Z}$. Computing $\widetilde{\mathbf{C}}$ and performing the eigendecomposition both take $\mathcal{O}(L^3)$, and recovering the final embedding takes $\mathcal{O}(NLd)$. Overall, the complexity of LLL is $\mathcal{O}\big(K_Z N^2 + N(Ld + DK_Z^2) + L^3)\big)$ with dense inputs and $\mathcal{O}\big(N(K_Z c + Ld + DK_Z^2) + L^3)\big)$ with sparse inputs, which is asymptotically much faster than the cost of the eigendecomposition if $L \ll N$.

The computational cost of the out-of-sample mapping is $\mathcal{O}(DK_Z^2)$ to find a projection vector $\mathbf{z}_0$ and $\mathcal{O}(Ld)$ for a reprojection in the low-dimensional space, hence $\mathcal{O}(DK_Z^2 + Ld)$ overall.

3 Choice of Parameters

Number of Landmarks L. One should use as many landmarks as one can afford computationally, because the more landmarks the better the approximation. As L increases, the results look more and more similar to the solution of the original spectral problem, which is recovered when $L = N$.

Number of Landmarks K_Z around Each Point. Each point should be a local linear reconstruction of nearby landmarks. Thus it is important that there are enough landmarks around each point so that its nearest landmarks are chosen along the manifold. These landmarks will have nonzero weights in the reconstruction, thus achieving locally and linearity. Non-local weights may not work unless the manifold is globally linear.

Using weights that are nonzero only for the nearest K_Z landmarks implies that the low-dimensional space is partitioned into regions where $\mathbf{X}$ is piecewise linear as a function of the corresponding subset of landmarks. If the K_Z landmarks are in general position, they span a linear manifold of dimension $K_Z - 1$. Therefore, we need no more than $D + 1$ landmarks, since $K_Z = D + 1$ of them reconstruct any point in D dimensions perfectly. On the other hand, using $K_Z > D + 1$ makes the weights non-unique and we need to add a regularization term to (7) to penalize the weight norm by adding a small positive amount to the diagonal of the linear system. However, the manifold learning assumption implies that the intrinsic dimensionality of the manifold is lower than D. For example, if the manifold is linear with dimension $\hat{d}$ then the number of landmarks needed to reconstruct any point is $K_Z = \hat{d} + 1$ by the same argument as above. However, if the manifold is nonlinear with local dimension $\hat{d}$, then $K_Z = \hat{d} + 1$ landmarks reconstruct the point approximately (near its projection on the tangent plane). Thus, overall the number of landmarks around each point should be between $\hat{d}+1$ (which may have a certain reconstruction error, particularly if the landmarks are not in general position) and $D+1$ (which achieves perfect reconstruction). If the reconstruction is imperfect, we introduce an additional error on the embedding, by implicitly replacing each original data point with its projection on landmarks. Thus, K_Z is a user parameter with values in $[\hat{d} + 1, D + 1]$: the larger K_Z the smaller the error and the larger the computational cost. In practice, K_Z can be estimated so a desired reconstruction error $\|\mathbf{Y} - \widetilde{\mathbf{Y}}\mathbf{Z}\|$ is achieved, but it should not be much bigger than $\hat{d} + 1$. Note $\hat{d}$ in this context is an intrinsic local

dimensionality of the manifold and not the dimensionality of the low-dimensional output d, which may or many not match $\hat{d}$.

The Location of Landmarks. Kumar et al. [28] provide a formal analysis of different types of sampling and show that, at least for the Nyström approximation, uniform sampling works best. Our experiments confirm this as well. However, we should not spend much computation on selecting landmarks, so as to introduce as little computational overhead as possible. Based on this, we investigated three general methods on how to compute the location of the landmarks.

First, we can always choose the landmarks at random from a set of existing points. This method requires almost no computational resources. However, the result can vary dramatically, especially when only a small number of landmarks is available. We can apply an additional heuristic to make the landmark location as close to uniform as possible: we select $K + M$ landmarks at random, find M pairs of closest landmarks and then discard one landmark from each pair. This heuristic is also useful because the distances are already given to us from the adjacency matrix. Even when the adjacency matrix is sparse, it is usually the largest distances that are missing. Thus, we can always identify closest landmarks to each other.

Second, we can select the landmarks by running a clustering algorithm with L clusters and choose each landmark in the middle of the clusters. For instance, one can run k-means and set the landmarks to the points that are closest to the centroids of the clusters. One problem with this approach is that the clustering is usually quite expensive. Another is that, for data with a nonconvex manifold structure, the landmarks can end up in between branches of the manifold (although this could be avoided with a k-modes algorithm that places landmarks in high-density regions of the data [29]). In our experiments we avoid dealing with landmarks that are not part of the dataset.

Finally, the landmarks can be also selected using other heuristics so they span the manifold as uniformly as possible. It has been proposed [14] to use a MinMax algorithm which chooses landmarks one by one by maximizing the mutual distance between the new landmark and the existing set of landmarks. However, this requires having the mutual distances between all the points, which in case of a large number of points N is unavailable.

4 Locally Linear Landmarks for Laplacian Eigenmaps

A particular case of the spectral method for which we can apply LLL is the Laplacian Eigenmaps (LE) algorithm [2]. The general embedding formulation is recovered using $\mathbf{A}$ as a graph Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$ defined on a symmetric affinity matrix $\mathbf{W}$ with degree matrix $\mathbf{D} = \operatorname{diag}\left(\sum_{m=1}^{N} w_{nm}\right)$ and $\mathbf{B} = \mathbf{D}$. The objective function is thus

$$\min_{\mathbf{X}} \operatorname{tr}\left(\mathbf{X}\mathbf{L}\mathbf{X}^T\right) \text{ s.t. } \mathbf{X}\mathbf{D}\mathbf{X}^T = \mathbf{I}, \mathbf{X}\mathbf{D}\mathbf{1} = \mathbf{0}. \tag{8}$$

Note that adding the second constraint does not alter the general formulation of the spectral solution, but just removes the first eigenvector, which is constant and equal to $\mathbf{D}^{-\frac{1}{2}}\mathbf{1}$ with eigenvalue 1.

The matrices in the reduced spectral problem (4) are then:

$$\widetilde{\mathbf{A}} = \mathbf{Z}\mathbf{L}\mathbf{Z}^T, \qquad \widetilde{\mathbf{B}} = \mathbf{Z}\mathbf{D}\mathbf{Z}^T. \tag{9}$$

Similarly to the case of the original LE, the second constraint is satisfied by discarding the first eigenvector. We can see this by noticing that $\widetilde{\mathbf{A}}\mathbf{1} = \mathbf{0}$ and looking at the eigendecomposition of $\widetilde{\mathbf{C}}$:

$$\widetilde{\mathbf{B}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{B}}^{-\frac{1}{2}}\widetilde{\mathbf{u}}_1 = \widetilde{\mathbf{B}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{x}}^T = \lambda_1\widetilde{\mathbf{u}}_1.$$

Therefore, the solution corresponding to the eigenvalue $\lambda_1 = 0$ is trivial.

The affinity matrix $\mathbf{W}$ for LE is usually computed using a Gaussian kernel with a bandwidth parameter σ (or a separate bandwidth per point [30])). The affinities are also sparsified by retaining only the K_W biggest values for every row. The performance of LE depends crucially on the choice of those parameters and they have to be tuned quite carefully in order to achieve good results. Unfortunately, in most cases there is no procedure to check the quality of the affinity matrix without running LE itself. However, instead of solving multiple, expensive LE problems, we can tune those parameters by running LLL. This gives a much cheaper runtime, especially considering that the matrix $\mathbf{Z}$ is independent of the choice of σ and K_W and, thus, is computed only once.

5 Experimental Evaluation

We compared LLL for LE to three natural baselines. (1) "Exact LE" runs LE on the full dataset and gives the optimal embedding by definition, but the runtime is large. (2) "LE ($\mathbf{Z}$)" runs LE only on a set of landmark points and then projects non-landmark points using the projection matrix $\mathbf{Z}$, which gives a locally linear (but globally nonlinear) out-of-sample mapping. (3) "LE (Nys.)" runs LE only on a set of landmark points and uses the Nyström out-of-sample formula. The latter two Landmark LE baselines give faster performance, but the embedding quality can be worse because non-landmark points are completely ignored in solving the spectral problem. For all our experiments we used Matlab's `eigs` function to compute the partial eigendecomposition of a sparse matrix.

Role of the Number of Landmarks. We used 60 000 MNIST digits with sparsity $K_W = 200$ and bandwidth $\sigma = 200$ to build the affinity matrix and reduced the dimensionality to $d = 50$. For LLL, we set $K_Z = 50$ and increased the number of landmarks logarithmically from 50 to 60 000. We chose landmarks at random and repeated the experiment 5 times for different random initialization to see the sensibility of the results to the random choice of the landmarks. To quantify the error with respect to Exact LE we first used Procrustes alignment [4, ch. 5] to align the embeddings of the methods and then computed the relative error between the aligned embeddings.

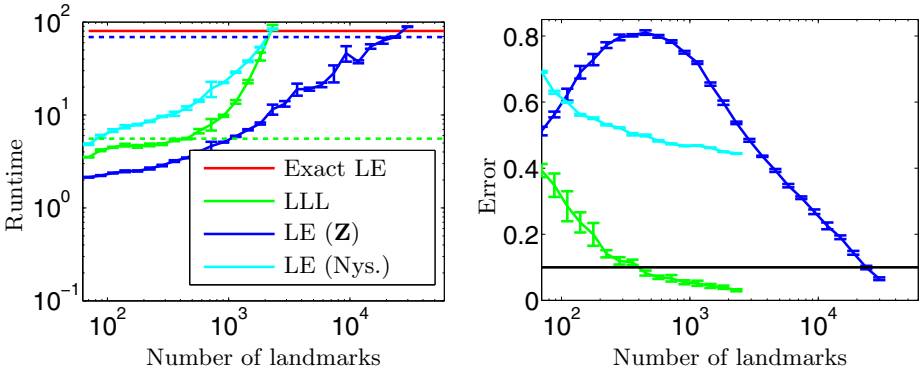

Fig. 2. Performance of LLL (green), Landmark LE with $\mathbf{Z}$ as an out-of-sample (blue) and Landmark LE with Nyström as an out-of-sample (cyan). *Left*: runtime as the number of landmarks changes. The green and blue dashed lines correspond to the runtime that gives 10% error with respect to Exact LE for LLL and Landmark LE using $\mathbf{Z}$, respectively. *Right*: error with respect to Exact LE. The black line corresponds to 10% error. Note the log scale in most of the axes.

Fig. 2 shows the error as well as the overall runtime for different algorithms as the number of landmarks increases. Our first indicator of performance is to see which algorithm can attain an error of 10% faster. LLL needed 451 landmarks and 5.5 seconds (shown by a dashed green line in the left plot). This is 14 times faster compared to Exact LE, which takes 80 seconds. Landmark LE with $\mathbf{Z}$ as out-of-sample mapping attains the same error with 23 636 landmarks and the runtime of 69 seconds (1.15 speedup, blue dashed line in the right plot). Landmark LE with Nyström is not able to attain an error smaller than 50% with any number of landmarks. Note the deviation from the mean for 5 runs of randomly chosen landmarks is rather small, suggesting the algorithm is robust to different locations of landmarks. Fig. 3 shows the embedding of Exact LE and the embedding of LLL with 451 randomly selected landmarks. The embedding of LLL is very similar to the one of Exact LE, but the runtime is 15 times faster (5.5 seconds compared to 80 seconds). Using more landmarks only decreases the error further and for 3 000 landmarks, where the runtime of LLL matches the runtime of Exact LE, the mean error among 5 runs drops to 3%. Landmark LE with $\mathbf{Z}$ as an out-of-sample attained the same error only by using 23 636 landmarks and a runtime of 69 seconds (1.15 speedup, blue dashed line in the right plot). Landmark LE with Nyström is not able to attain an error smaller than 50% for any number of landmarks. Note the deviation from the mean for 5 runs of randomly chosen landmarks is rather small, suggesting the algorithm is relatively robust to different locations of landmarks.

Model Selection. We evaluated the use of LLL to select the parameters of the affinity matrix. We used 4 000 points from the Swissroll dataset and ran the

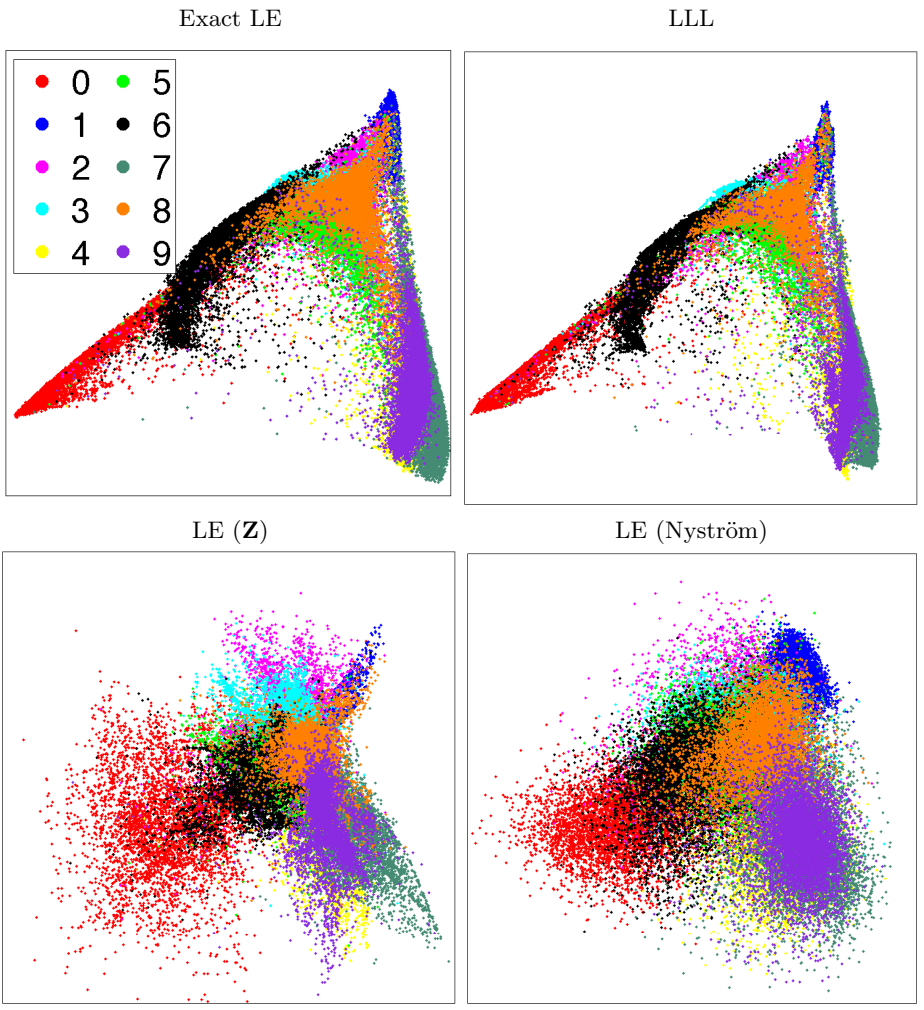

Fig. 3. Embedding 60 000 MNIST digits using the first two dimensions. *Left to right*: Exact LE ($t = 80$ s), LLL ($t = 5.5$ s, 451 landmarks), Landmark LE with $\mathbf{Z}$ as out-of-sample mapping ($t = 5.5$ s, 1 144 landmarks), and LE with Nyström as out-of-sample mapping ($t = 5.5$ s, 88 landmarks).

methods varying different parameters of the algorithm. We ran LLL and Landmark LE 5 times using different random initializations to show the general behavior of the algorithm. Experimentally we discovered that the best results are obtained with a bandwidth $\sigma = 1.6$, a number of landmarks L no less than 300 and a sparsity level K_W around 150. We then fixed two out of these three parameters and changed the third one to see how the error curve changes. Fig. 4 shows the results. First, for different σ values the error curve of Exact LE is

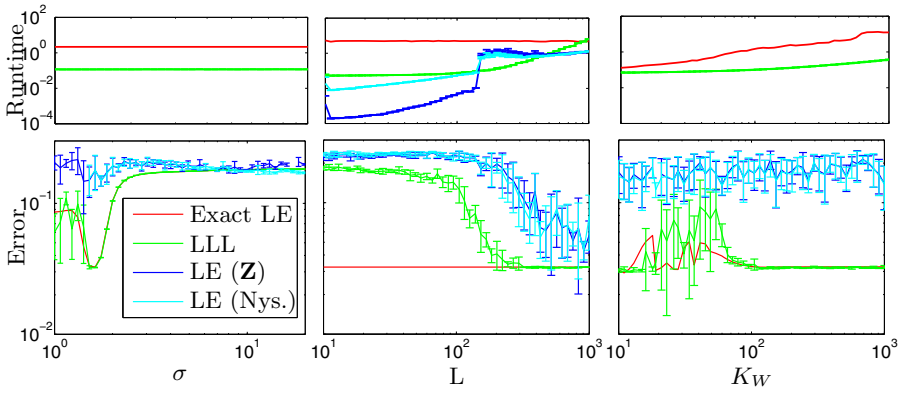

Fig. 4. Quality of the embedding with respect to the ground truth for different values of the bandwidth σ, number of landmarks L, and sparsity level K_W. The dataset contains 4 000 points from a Swissroll. From left to right: vary σ for fixed $L = 300$, $K_W = 150$; vary L for fixed $\sigma = 1.6$, $K_W = 150$; vary K_W for fixed $L = 300$, $\sigma = 1.6$. *Top row*: runtime for different values of the parameters. *Bottom row*: error.

much more similar to the one from LLL, but LLL is able to achieve it about 18× faster (top plot). Compared to that, Landmark LE definitely needs more landmarks in order to show a similar behavior. Second, the number of landmarks needed to achieve the same error as Exact LE is much lower for LLL than for Landmark LE. Using 300 landmarks the error of LLL is about 3% and it is also 18 times faster than Exact LE. Landmark LE is never able to achieve a 10% error for any set of landmarks up to 1 000. Third, changing the sparsity level parameter K_W, the error curve is again very similar between Exact LE and LLL, but very different between Exact LE and Landmark LE. The speedup of LLL compared to Exact LE varies between 2 for small values of K_W to 40 for large K_W. Note that, although LLL is not able to reproduce an error curve identical to that of Exact LE, *it does match the minima of these curves* (for σ and K_W), and of course the minima correspond to the parameter values we are interested in. That is, LLL can be used as a fast way to find good parameter values for Exact LE. This suggests a practical procedure to set the parameters of Exact LE: we run LLL to obtain the values of σ and K_W that give the (approximately) minimum error and then run Exact LE using those values.

Classification. Here our goal was to find a good set of parameters to achieve a low 1-nearest neighbor classification error for the full 70 000 MNIST digits dataset. We first split the dataset into three independent sets: 50 000 digits as a training set, 10 000 digits as a test set and 10 000 digits for out-of-sample mapping. We then projected training and test sets (overall 60 000 points) to 500 dimensions using LLL with 1 000 landmarks selected using k-means with $K_Z = 50$. We did this a number of times for different values of K_W from 1 to 200 and σ from 4.6 to 1000. Note the $\mathbf{Z}$ matrix is independent from the affinities and depends only on

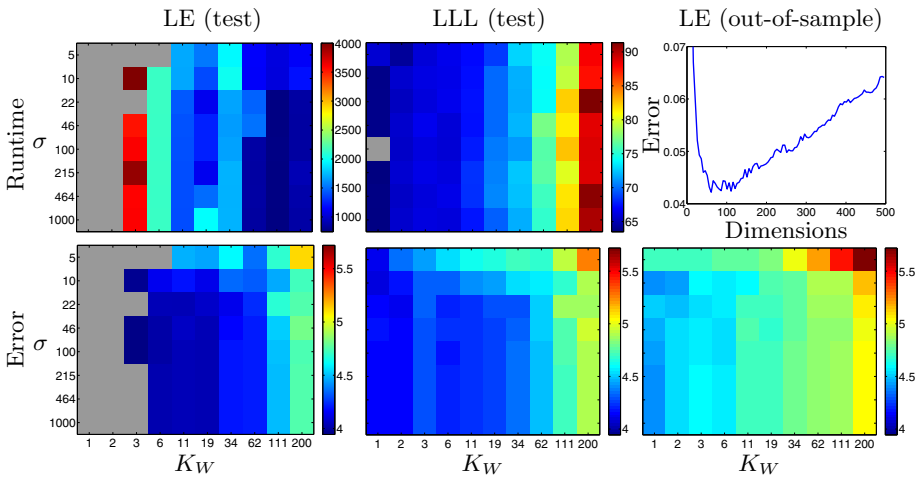

Fig. 5. 1-nearest neighbor classification error of MNIST digits after applying LLL, for different values of σ and K_W. See the main text for details. Gray areas corresponds to Matlab's `eigs` routine not converging. *Top two plots*: runtime for Exact LE and LLL. The color corresponds to the runtime in seconds. *Bottom three plots*: error of Exact LE, LLL and out-of-sample set, respectively. The color corresponds to percent classification error. The out-of-sample runtime is constant and equal to 30 seconds for all values of K_W and σ. *Top right plot*: 1-nearest neighbor classification error for different dimensions for the test subset with $\sigma = 10$ and $K_W = 1$.

the choice of landmark points, so we can save 30 seconds' runtime for each point by precomputing that matrix and using it for all variations of the parameters. Given the location of the embedding points $\widetilde{\mathbf{X}}$, we also computed the out-of-sample projection matrix $\mathbf{Z}_{\text{OOS}}$ to find the embedding of the out-of-sample set as well. We computed the 1-nearest neighbor classification for different number of dimensions separately and reported the smallest error, for both the test and the out-of-sample sets. Fig. 5 shows the results. The smallest error is achieved for very small values of K_W. There is also little discrepancy between the error for test and out-of-sample sets, which indicate our out-of-sample mapping is accurate. The top right corner shows the variation of the error as we change the dimensionality. The results are shown for $\sigma = 10$ and $K_W = 1$, but the curve is very similar for other sets of parameters as well. Note the runtime of LLL is less than two minutes for the embedding of as many as 60 000 MNIST points.

We also tried to repeat the same experiment for Exact LE to compare the results with LLL, but we found many complications. First of all, it turns out that for small values of K_W the graph Laplacian is not connected and Matlab's `eigs` routine does not converge (at least not all 500 requested eigenvalues). For larger values of K_W `eigs` converges, but takes many iterations, which increase the runtime dramatically to almost 4 000 seconds. Note it is exactly for those values that both Exact LE and LLL give the smallest error (in fact the smallest

Fig. 6. Example of the elastic deformation of MNIST digits in the infinite MNIST dataset. *Top*: original. *Bottom*: one of the 16 deformations we applied to each digit.

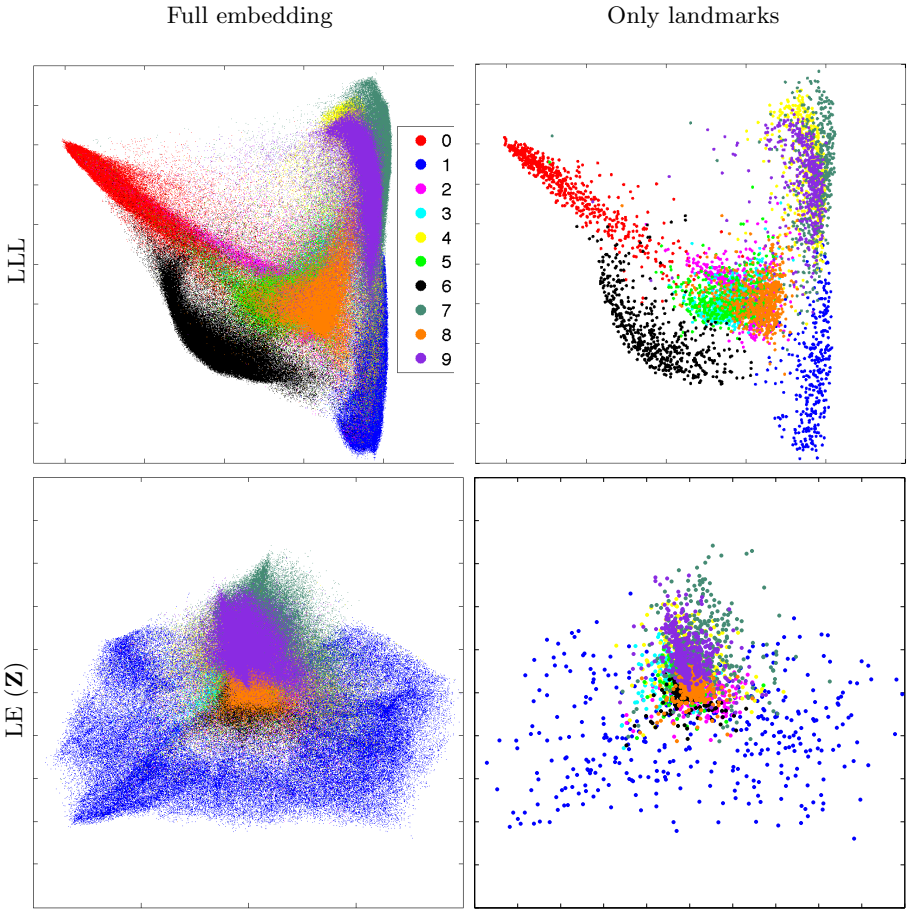

Fig. 7. Embedding of 1 020 000 points from the infinite MNIST dataset using 10 000 landmarks. *Top*: LLL, *bottom*: LE ($\mathbf{Z}$). *Left*: full dataset $\mathbf{X}$, *right*: landmarks $\tilde{\mathbf{X}}$ only.

error for LLL is achieved for $K_W = 1$, for which Exact LE did not even converge). Increasing K_W improves the connectivity of the graph Laplacian, but the runtime of `eigs` did not decrease much below 1 000 seconds, which means LLL is 15–40× faster depending on the particular set of parameters. Finally, the general pattern of variation and values of the error is almost the same for Exact LE, LLL and the out-of-sample set. The error gradually increases from the lower left corner to the upper right in all three cases.

Large-Scale Experiment. We used the infinite MNIST dataset [31], where we generated 1 020 000 handwritten digits using elastic deformations of the original MNIST dataset (see examples of the deformations in Fig. 6). We reduce the dimensionality to two with 10 000 randomly selected landmarks and $K_Z = 5$ nearest landmarks. LLL took 4.2 minutes to compute the projection matrix $\mathbf{Z}$ and 14 minutes to compute the embedding. We also run LE ($\mathbf{Z}$) on the same 10 000 landmarks. Fig. 7 shows the resulting embeddings. In the embedding of LLL, zeros, sixes and ones are separated from the rest of the digits, and nines, fours and sevens form their own group (all those digits contain in them a straight vertical line). The embedding for For LE ($\mathbf{Z}$) shows far ess structure. Only ones and a group containing sevens and nines can be separated. The rest of the points are trapped in the center of the figure.

6 Conclusion

Spectral methods for manifold learning and clustering often give good solutions to problems involving nonlinear manifolds or complex clusters, and are in widespread use. However, scaling them up to large datasets (large N) and nontrivial numbers of eigenvectors (d) requires approximations. The Locally Linear Landmarks (LLL) method proposes a reduced formulation of the original spectral problem that optimizes only over a small set of landmarks, while retaining structure of the whole data. The algorithm is well defined theoretically and has better performance than the Nyström method, allowing users to scale up applications to larger dataset sizes. LLL also defines a natural out-of-sample extension that is cheaper and better than the Nyström method. This paper has focused on the case of Laplacian eigenmaps, where LLL was able to achieve 10×–20× speedups with small approximation error.

The basic framework of LLL, where we replace the low-dimensional projections by a fixed linear function of only a few of the projections, is applicable to any spectral method. However, the best choice of the linear function is an interesting topic of future research. In particular for spectral clustering, the input data need not have manifold structure, but the cluster label of a point may be well approximated by a function of some of its neighboring landmarks.

Acknowledgment. Work funded in part by NSF CAREER award IIS–0754089.

References

1. Saul, L.K., Weinberger, K.Q., Ham, J.H., Sha, F., Lee, D.D.: Spectral methods for dimensionality reduction. In: Chapelle, O., Schölkopf, B., Zien, A. (eds.) Semi-Supervised Learning. Adaptive Computation and Machine Learning Series, pp. 293–308. MIT Press (2006)
2. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation 15(6), 1373–1396 (2003)
3. Schölkopf, B., Smola, A., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation 10(5), 1299–1319 (1998)
4. Cox, T.F., Cox, M.A.A.: Multidimensional Scaling. Chapman & Hall, London (1994)
5. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290(5500), 2319–2323 (2000)
6. Saul, L.K., Roweis, S.T.: Think globally, fit locally: Unsupervised learning of low dimensional manifolds. J. Machine Learning Research 4, 119–155 (2003)
7. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Analysis and Machine Intelligence 22(8), 888–905 (2000)
8. Lehoucq, R.B., Sorensen, D.C.: Deflation techniques for an implicitly restarted Arnoldi iteration. SIAM J. Matrix Anal. and Apps. 17(4), 789–821 (1996)
9. Williams, C.K.I., Seeger, M.: Using the Nyström method to speed up kernel machines. In: Leen, T.K., Dietterich, T.G., Tresp, V. (eds.) Advances in Neural Information Processing Systems (NIPS), vol. 13, pp. 682–688. MIT Press, Cambridge (2001)
10. Bengio, Y., Paiement, J.F., Vincent, P., Delalleau, O., Le Roux, N., Ouimet, M.: Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and spectral clustering. In: Thrun, S., Saul, L.K., Schölkopf, B. (eds.) Advances in Neural Information Processing Systems (NIPS), vol. 16. MIT Press, Cambridge (2004)
11. Drineas, P., Mahoney, M.W.: On the Nyström method for approximating a Gram matrix for improved kernel-based learning. J. Machine Learning Research 6, 2153–2175 (2005)
12. Talwalkar, A., Kumar, S., Rowley, H.: Large-scale manifold learning. In: Proc. of the 2008 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR 2008), Anchorage, AK, June 23-28 (2008)
13. von Luxburg, U., Radl, A., Hein, M.: Getting lost in space: Large sample analysis of the resistance distance. In: Lafferty, J., Williams, C.K.I., Shawe-Taylor, J., Zemel, R., Culotta, A. (eds.) Advances in Neural Information Processing Systems (NIPS), vol. 23, pp. 2622–2630. MIT Press, Cambridge (2010)
14. de Silva, V., Tenenbaum, J.B.: Sparse multidimensional scaling using landmark points (June 30, 2004)
15. de Silva, V., Tenenbaum, J.B.: Global versus local approaches to nonlinear dimensionality reduction. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in Neural Information Processing Systems (NIPS), vol. 15, pp. 721–728. MIT Press, Cambridge (2003)
16. Platt, J.: FastMap, MetricMap, and landmark MDS are all Nyström algorithms. In: Cowell, R.G., Ghahramani, Z. (eds.) Proc. of the 10th Int. Workshop on Artificial Intelligence and Statistics (AISTATS 2005), Barbados, January 6-8, pp. 261–268 (2005)

17. Gao, S., Tsang, I.W.H., Chia, L.T., Zhao, P.: Local features are not lonely — Laplacian sparse coding for image classification. In: Proc. of the 2010 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR 2010), San Francisco, CA, June 13-18, pp. 3555–3561 (2010)
18. Wang, F.Y., Chi, C.Y., Chan, T.H., Wang, Y.: Nonnegative least-correlated component analysis for separation of dependent sources by volume maximization. IEEE Trans. Pattern Analysis and Machine Intelligence 32(5), 875–888 (2010)
19. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science 290(5500), 2323–2326 (2000)
20. Weinberger, K., Packer, B., Saul, L.: Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In: Cowell, R.G., Ghahramani, Z. (eds.) Proc. of the 10th Int. Workshop on Artificial Intelligence and Statistics (AISTATS 2005), Barbados, January 6-8, pp. 381–388 (2005)
21. Yu, K., Zhang, T., Gong, Y.: Nonlinear learning using local coordinate coding. In: Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C.K.I., Culotta, A. (eds.) Advances in Neural Information Processing Systems (NIPS), vol. 22. MIT Press, Cambridge (2009)
22. Ladický, Ľ., Torr, P.H.S.: Locally linear support vector machines. In: Getoor, L., Scheffer, T. (eds.) Proc. of the 28th Int. Conf. Machine Learning (ICML 2011), Bellevue, WA, June 28-July 2, pp. 985–992 (2011)
23. Liu, W., He, J., Chang, S.F.: Large graph construction for scalable semi-supervised learning. In: Fürnkranz, J., Joachims, T. (eds.) Proc. of the 27th Int. Conf. Machine Learning (ICML 2010), Haifa, Israel, June 21-25 (2010)
24. Zhang, Z., Wang, J.: MLLE: Modified locally linear embedding using multiple weights. In: Schölkopf, B., Platt, J., Hofmann, T. (eds.) Advances in Neural Information Processing Systems (NIPS), vol. 19, pp. 1593–1600. MIT Press, Cambridge (2007)
25. Chen, X., Cai, D.: Large scale spectral clustering with landmark-based representation. In: Proc. of the 25th National Conference on Artificial Intelligence (AAAI 2011), San Francisco, CA, August 7-11, pp. 313–318 (2011)
26. Bengio, Y., Delalleau, O., Le Roux, N., Paiement, J.F., Vincent, P., Ouimet, M.: Learning eigenfunctions links spectral embedding and kernel PCA. Neural Computation 16(10), 2197–2219 (2004)
27. Carreira-Perpiñán, M.Á., Lu, Z.: The Laplacian Eigenmaps Latent Variable Model. In: Meilă, M., Shen, X. (eds.) Proc. of the 11th Int. Workshop on Artificial Intelligence and Statistics (AISTATS 2007), San Juan, Puerto Rico, March 21-24, pp. 59–66 (2007)
28. Kumar, S., Mohri, M., Talwalkar, A.: Sampling methods for the Nyström method. J. Machine Learning Research (2012)
29. Carreira-Perpiñán, M.Á., Wang, W.: The K-Modes algorithm for clustering. arXiv:1304.6478 (April 23, 2013) (unpublished manuscript)
30. Vladymyrov, M., Carreira-Perpiñán, M.Á.: Entropic affinities: Properties and efficient numerical computation. In: Proc. of the 30th Int. Conf. Machine Learning (ICML 2013), Atlanta, GA, June 16-21, pp. 477–485 (2013)
31. Loosli, G., Canu, S., Bottou, L.: Training invariant support vector machines using selective sampling. In: Bottou, L., Chapelle, O., DeCoste, D., Weston, J. (eds.) Large Scale Kernel Machines. Neural Information Processing Series, pp. 301–320. MIT Press (2007)