Available online at www.sciencedirect.com
SCIENCE DIRECT
Neural Networks 18 (2005) 702–710
Neural Networks
www.elsevier.com/locate/neunet

2005 Special Issue

Improving dimensionality reduction with spectral gradient descent

Roland Memisevic*, Geoffrey Hinton

Department of Computer Science, University of Toronto, Toronto, Ont., Canada M5S 3G4

Abstract

We introduce spectral gradient descent, a way of improving iterative dimensionality reduction techniques.[1] The method uses information contained in the leading eigenvalues of a data affinity matrix to modify the steps taken during a gradient-based optimization procedure. We show that the approach is able to speed up the optimization and to help dimensionality reduction methods find better local minima of their objective functions. We also provide an interpretation of our approach in terms of the power method for finding the leading eigenvalues of a symmetric matrix and verify the usefulness of the approach in some simple experiments.

1. Introduction

Many problems in machine learning and data analysis involve high-dimensional data that is distributed along some low-dimensional manifold in the input space. Finding this manifold and representing the data in a low-dimensional coordinate system is an important step in many applications since it can help to identify the true underlying causes of variability in the data, provide more efficient and therefore more manageable representations of the data, and help to overcome the 'curse of dimensionality'.

PCA is the standard linear method for this task, but recently there has been a lot of research on nonlinear dimensionality reduction (NLDR), driven by the awareness that real world data is usually distributed along nonlinear manifolds rather than along linear subspaces of the input space. As a result of this increased interest in NLDR a lot of methods have been proposed in the last few years, and they are now increasingly being used in applications.

Interestingly, the basic computational procedures used by the different methods that have been proposed (such as kernel PCA (Schölkopf, Smola, & Muller, 1998), stochastic neighbor embedding (Hinton & Roweis), locally linear embedding (Roweis & Saul, 2000), unsupervised kernel regression (Meinicke, Klanke, Memisevic, & Ritter, in press), Laplacian eigenmaps (Belkin & Niyogi, 2003), Gaussian process latent variable models (Lawrence, 2004), and many others) are very similar to one another: First, a matrix is constructed that encodes pair-wise similarities between input data-points. Then, an analogous matrix is defined for a set of low-dimensional data representatives. Finally, some form of mismatch between these similarity matrices is minimized. Since the low-dimensional similarity matrix is 'parameterized' by the set of low-dimensional or 'latent' data representatives, this minimization can be performed directly with respect to these latent space elements.

Depending on the definition of the similarity matrices and the mismatch, these methods can be categorized into two classes: Iterative methods and spectral methods. The first rely on iterative, gradient-based optimization, to find a local minimum of their objective function, while the latter can find their (globally optimal) solutions by an eigen-decomposition of a (usually symmetric) matrix.

Besides their computational differences, these two kinds of method have also very different practical drawbacks and benefits. Further advantages of spectral methods, for example, are their ability to discover very complicated manifold structure and their independence of any parameter initializations. An advantage of many iterative methods on the other hand is that they are based on more principled, often probabilistic, theoretical foundations. As a result they can be more easily extended or combined with other methods. Consequently, there are iterative methods (i) that provide mappings from the high-dimensional input

* Corresponding author.
E-mail addresses: roland@cs.toronto.edu (R. Memisevic), hinton@cs.toronto.edu (G. Hinton).

[1] An abbreviated version of some portions of this article appeared in Memisevic and Hinton (2005a), published under the IEEE copyright.

0893-6080/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2005.06.034

to the low-dimensional embedding space and back (Meinicke et al., in press), (ii) that are based on data-driven error criteria that allow for principled ways to perform model selection (Meinicke et al., in press), (iii) that provide ways to include side-information into the learning process (Memisevic & Hinton, 2005b), and (iv) that make it possible to associate more than a single latent representative with each high-dimensional data-point (Hinton & Roweis).

In practice, the exclusiveness of the benefits of the two approaches often poses a dilemma. In some situations where we would like to use a spectral method for its efficiency we might actually have to resort to an iterative method, because the problem setting might demand a suitably defined mapping from latent to data-space for example. In other situations we might prefer the objective function that is optimized by an iterative method, but the data manifold we are dealing with might turn out to be so complicated that the only way to avoid poor local minima is to use a spectral method.

In this paper we suggest a way of partially resolving this dilemma by showing how to use spectral information in iterative methods. We derive our approach from a simple intuition from the area of spectral clustering. Our idea is to aggregate the movements of latent space elements during the gradient-based optimization of an iterative method, such that latent representatives of nearby points move in similar directions. The required information about nearness is extracted beforehand from the eigenvectors of a data similarity matrix. Experiments show that our approach can significantly speed up iterative methods and can reduce their dependence on latent variable initializations thus allowing them to find better local minima of their objective functions.

The remainder of this paper is structured as follows: In Section 2 we review the problem of dimensionality reduction and give an overview of different methods. In Section 3 we describe our approach of using spectral methods in the iterative optimization procedure of iterative methods. The original approach relies on the extraction of eigenvectors of some similarity matrix prior to the actual gradient based optimization. We discuss a way to circumvent altogether the need for an eigen-decomposition in Section 4. In Section 5 we provide an interpretation of this approach in terms of the power method for finding the leading eigenvectors of a matrix. We present simple experiments that support the usefulness of our method in Section 6 and conclude in Section 7.

2. Dimensionality reduction

We can formalize the problem of dimensionality reduction as follows: For a set of N real vectors $y_1,\ldots, y_N$ of dimensionality d find a corresponding set of 'latent' vectors $x_1, \ldots, x_N$ of dimensionality $q \ll d$ such that these latent representatives preserve the similarity structure inherent in the input data set as well as possible. For convenience, we will frequently stack the data points column-wise in a $(d\times N)$-matrix Y and the latent space elements in a corresponding $(q\times N)$-matrix X. Practically all nonlinear methods that have been proposed in the last few years solve this task in a nonparametric fashion: They first construct an $(N\times N)$-matrix P that encodes pairwise similarities between data-points and a corresponding matrix $Q(X)$ that encodes the similarities between the low-dimensional data representatives. Since the latent similarity matrix depends on the set X of latent elements, dimensionality reduction can then be performed by minimizing wrt. X a measure $E(X)$ of the *mismatch* between these two matrices.

Practically, the methods that have been proposed differ only in the definition of P, $Q(X)$ and $E(X)$. The motivations for the many ways to define the similarities and the mismatch differ greatly across methods and range from probabilistic considerations (Hinton & Roweis) to the desire to fit principal subspaces in kernel feature spaces (Schölkopf et al., 1998). Computationally, the approaches usually lead to one of two possible optimization problems. If $E(X)$ is quadratic in the latent variables, then under suitable constraints on X the solution can be found by an eigen-decomposition, with the advantages discussed above. If not, we have to resort to iterative optimization. Unfortunately, since the different motivations yield methods with rather different practical advantages, computational considerations often have to stand back and the actual task at hand dictates the choice of method.

To provide two examples of iterative methods, which we will also use in the experiments, we briefly review the methods stochastic neighbor embedding (SNE) (Hinton and Roweis) and unsupervised kernel regression (UKR) (Meinicke et al., in press). We then contrast these with the computational procedures used in spectral methods, which are closely related to the speed-up strategy that we discuss in Section 2.1.

2.1. Stochastic neighbor embedding

SNE is a probabilistic approach to embedding that tries to preserve stochastically defined neighborhood relations between data-points. It defines each of the matrices P and $Q(X)$ as a set of transition probabilities (row-wise) and the mismatch $E(X)$ as the sum of Kullback–Leibler divergences between corresponding probability distributions (that is between corresponding rows of P and $Q(X)$). The transition probabilities are defined by centering Gaussian kernel functions with bandwidth h on the data-points and normalizing. Formally, we have

$$P_{ij} := \frac{\exp\left(-\frac{1}{h}||y_i - y_j||^2\right)}{\sum_k \exp\left(-\frac{1}{h}||y_i - y_k||\right)^2} \tag{1}$$

as the transition probability from the *i*th to the *j*th data-point and analogously for the latent space elements

$$Q_{ij}(X) := \frac{\exp(-||x_i - x_j||^2)}{\sum_k \exp(-||x_i - x_k||)^2}. \quad (2)$$

Note that, since the latent space elements are free to move during optimization, the bandwidth for the latent space is arbitrary and set to 1.0 here. The mismatch is defined as

$$E(X) = \frac{1}{N} \sum_i \sum_j P_{ij} \log\left(\frac{P_{ij}}{Q_{ij}(X)}\right), \quad (3)$$

and can be minimized by using the gradient, given by

$$\frac{\partial E(X)}{\partial x_l} = \frac{2}{N} \sum_i (P_{il} + P_{li} - Q_{li} - Q_{il})(x_l - x_i). \quad (4)$$

2.2. Unsupervised kernel regression

UKR is based on rather different considerations and tries to minimize the point-wise reconstruction errors resulting from mapping the latent data representatives into the data-space. The mapping, usually referred to as 'forward' mapping, is defined as the Nadaraya–Watson regression estimator (Härdle, 1990) with Gaussian kernels:

$$f(x) = \sum_j \frac{\exp(-||x - x_j||^2)}{\sum_k \exp(-||x - x_k||^2)} y_j. \quad (5)$$

Note again the arbitrariness of the latent space kernel bandwidth, which we again set to 1.0 for convenience.

The sum of the reconstruction errors under this mapping can now be shown (Meinicke et al., in press) to be equal to

$$E(X) = \frac{1}{N} \mathrm{tr}(PQ(X)), \quad (6)$$

using the definitions

$$P = Y^{\mathrm{T}} Y \quad (7)$$

and

$$Q(X) = (I - B(X))(I - B(X))^{\mathrm{T}} \quad (8)$$

with

$$B_{ij}(X) = \frac{\exp(-||x_i - x_j||^2)}{\sum_k \exp(-||x_k - x_j||^2)}. \quad (9)$$

As with SNE, any gradient-based optimization method can be used to minimize the loss defined in Eq. (6). However, in practice an explicit regularization needs to be enforced for the solution to be well-defined. One way to realize this is to restrict the latent representatives to a finite interval of the latent space. Another way is to replace the regression estimator by a leave-one-out version. ((See Meinicke et al., in press) for details.)

2.3. Spectral methods

If the mismatch $E(X)$ is defined as a quadratic form in X, that is, if it is of the form

$$E(X) = \mathrm{tr}(\mathrm{XPX}^{\mathrm{T}}) \quad (10)$$

(where tr denotes the trace operator), then by the Rayleigh–Ritz theorem (Horn & Johnson, 1986) minimizer of $E(X)$ under the constraints

$$\frac{1}{N} \mathrm{XX}^{\mathrm{T}} = I_q \quad \text{and} \quad X\mathbf{1} = \mathbf{0} \quad (11)$$

(with I_q the $q \times q$ identity matrix, $\mathbf{1}$ the N-vector of all ones, and $\mathbf{0}$ the q-vector of all zeros) is the matrix X whose rows contain the leading eigenvectors of P. As with iterative methods, a lot of different objectives have been formulated for spectral methods, leading to a lot of different error measures-which obviously differ only in the definition of the data-space similarity matrix P. Kernel PCA (Schölkopf et al., 1998), e.g. is based on the eigen-decomposition of a kernel matrix defined on the data points; Laplacian eigenmaps (Belkin & Niyogi, 2003) set P to be a normalized kernel matrix; etc.

The foremost advantage of spectral methods is that the optimum is *global*. Furthermore, it is well-known from practice that spectral methods are able to fit far more complicated manifolds than iterative methods (see, e.g. (Meinicke et al., in press)). Spectral methods are also widely appreciated for their superior efficiency. Usually, performing a single eigen-decomposition is orders of magnitude faster than running an iterative optimization procedure for many iterations. This advantage, however, might vanish as larger and larger datasets are considered, since the eigen-decomposition generally needs $O(N^3)$ operations whereas it is clear from the previous considerations that the number of operations needed for a single iteration of an iterative methods usually scales quadratically with dataset size. Unfortunately, using a quadratic objective usually implies that important features of iterative methods have to be abandoned, as discussed previously.

In Section 3 we describe a way of gaining some of the advantages of spectral methods without restricting the objective function to be quadratic. The approach is based on the idea of intertwining the two different kinds of method to obtain a single optimization routine in which spectral information is used to improve a gradient-based search.

3. Aggregating the movements

3.1. An intuition from spectral clustering

We derive our approach from a simple intuition from spectral clustering. Spectral clustering methods are similar to spectral NLDR methods: They also construct a similarity matrix P from the data and compute

its eigen-decomposition. In contrast to NLDR, they use the resulting low-dimensional data representation as the input for a subsequent standard clustering step. Usually standard k-means clustering is applied in this step. As opposed to using k-means on the raw input data, which finds groups of 'convex blobs' in the data, this approach has been shown to group together points that are arranged in more structured formations.

Although in general we may view spectral clustering as a form of spectral NLDR followed by clustering, the construction of the similarity matrix usually differs. In both cases, the matrix is formed by centering (usually Gaussian) kernels on the data points, followed by some form of normalization. But for spectral clustering the normalizations that have been suggested are usually different from that suggested for NLDR.

An approach that is commonly used in spectral clustering is to set

$$M = D^{-\frac{1}{2}}\mathrm{K}D^{-\frac{1}{2}}, \tag{12}$$

with $K_{ij} = \exp(-(1/h)||y_i - y_j||^2)$ and where D is diagonal with entries $D_{ii} = \sum_j K_{ij}$ (see Ng, Jordan, & Weiss, 2002).

A key observation that justifies the final (clustering) step is that the entries of the leading eigenvectors of the normalized similarity matrix reflect the clustering of the input data. That is, the eigenvectors are approximately piecewise constant, where each constant level corresponds to a cluster in the original data. (See, e.g. Maila & Shi, 2001 or Weiss, 1999) for a more detailed account of this observation). An illustration is given in Fig. 1 (three upper-most plots): Plot (a) shows a two-dimensional dataset that consists of three clusters taking the shape of circular arcs. Plot (b) depicts the corresponding matrix M (Eq. (12)). The data has been sorted beforehand, such that points that belong to the same cluster are grouped together in the display. Plot (c) shows the entries of the three leading eigenvalues of M. The entries of the first eigenvalue are approximately constant and could be discarded. The entries of the second and third eigenvectors both reflect the clustering of the data-points. Both take on values that are approximately piecewise constant and are concentrated around three distinct levels corresponding to the three clusters.

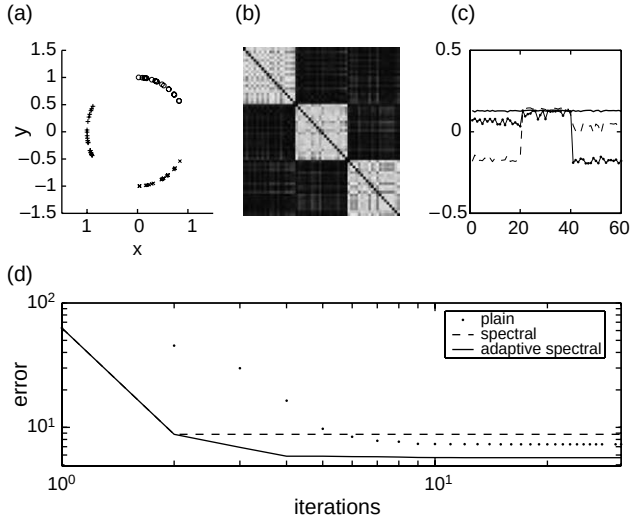

Fig. 1. (a) 2d-dataset; (b) normalized similarity matrix M; (c) Entries of the three leading eigenvectors; (d) learning curves (see text).

3.2. Altering the gradient

The question that we address in this paper is this: Can we use the clustering property of the eigenvectors to improve an *iterative* dimensionality reduction method? The approach we take is to use the entries of the eigenvectors to modify the gradient of the objective function, such that the movements of latent space elements resulting from following this gradient better respect the similarity structure of the input data. In particular, we will use the knowledge about cluster-membership provided by the eigenvectors to aggregate the movements of latent space elements. This allows us to encourage latent representatives that are known to belong together, in the sense that the data-points they represent reside in the same cluster, to move in similar directions.

Fig. 2 shows a simple illustration using two-dimensional latent representations of a set of hypothetical data-points consisting of three clusters. Latent elements whose corresponding data-points belong to the same cluster are depicted using the same symbol. The leftmost plot shows a possible random initialization of the latent space elements. Also indicated are the gradients with respect to the latent space elements at the current time-step, symbolized by arrows. As shown, the gradient directions do not necessarily reflect the clustering of the original data. As a result, latent representatives that should actually end up in nearby locations, because their data-space correspondents belong to the same cluster, may move in different directions during the initial phase of the gradient-based optimization, which will result in unnecessarily slow convergence.

In order to overcome this problem we first determine a set of l aggregated directions (where $l < N$) as linear combinations of the original ones, using the eigenvalue-entries as coefficients. Note that as a consequence directions belonging to similar data-points will have similar contributions (center subplot). Then we build a linear combination of these aggregated directions using each data-point's contribution as coefficients in order to obtain a new gradient

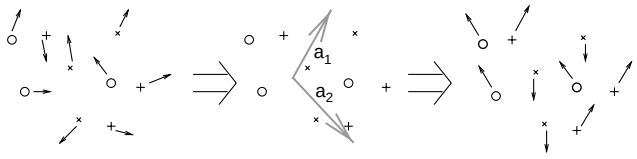

Fig. 2. Illustration of the aggregation strategy.

that better complies with the clustering of the input data. An update of the latent space elements based on this modified gradient respects this clustering and makes elements that belong together actually move together (rightmost subplot).

More formally, let g_i denote the gradient of E with respect to only the ith latent representative x_i:

$$g_i := \frac{\partial E}{\partial x_i} \tag{13}$$

That is, g_i contains the partial derivatives of E w.r.t. the entries of x_i. Furthermore we define the 'gradient matrix' G as the matrix containing the g_i column-wise.

Consider the eigen-decomposition $M := VDV^{\mathrm{T}}$, with $V = (v_1, v_2, \ldots, v_N)$ the matrix of (normalized) eigenvectors arranged column-wise and corresponding eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$ (residing on the diagonal of D). We now compute for each of the l leading eigenvectors v_i, $i = 1, \ldots, l$ an 'aggregated gradient' a_i as a linear combination of the original (point-wise) gradients $g_1, \ldots, g_N$, where we use the corresponding entries of v_i as the coefficients. That is, with v_{ij} denoting the jth entry of the ith eigenvector, we define $a_i := \sum_{j=1}^{N} v_{ij} g_j$, or in matrix form:

$$A := (a_1, \ldots, a_l) = \mathrm{G}\mathrm{V}_l, \tag{14}$$

where matrix V_l contains the l leading eigenvectors as columns. Note, that this results in similar weights for gradients corresponding to nearby data-points. The a_i span a 'gradient-subspace' of possible movements in latent space. Since the a_i are aggregations of the original gradients with similar coefficient for directions that belong together they are likely to reflect possible grouping structure in the data better than the original gradients.

We can now construct (for each $i = 1, \ldots, N$) a new point-wise gradient as a linear combination of these aggregated directions, where we use the data-point's original contributions in the aggregation as coefficients. That is, we define the reconstruction $\hat{g}_i$ for the gradient w.r.t. the ith data-point as $\hat{g}_i = \sum_{j=1}^{l} v_{ji} a_j$. The reconstruction $\hat{G}$ of the whole gradient matrix can then be written as

$$\hat{G} := (\hat{g}_1, \ldots, \hat{g}_{\mathrm{N}}) = \mathrm{A}\mathrm{V}_l^{\mathrm{T}} = \mathrm{G}\mathrm{V}_l V_l^{\mathrm{T}}. \tag{15}$$

3.3. Practical issues

It is obvious that the objective function that is minimized by following the modified gradient $\hat{G}$ is not the same as $E(X)$ anymore. A local minimum that we might reach using the aggregation strategy therefore does not necessarily coincide with an optimal solution from the original, unmodified method. In fact, there is no reason to assume, that it would. This means that in practice we should use the approach primarily to improve the optimization process in its initial phase and decrease its influence (or continue with the unmodified gradient) as the optimization continues.

One way to decrease the influence of the aggregation strategy immediately suggests itself: From Eq. (15) it is clear, that for $l = N$ we get $\hat{G} = G$ as implied by the orthogonality of V, which shows that it is in fact necessary to cut off the expansion for A and $\hat{G}$ after $l < N$ terms. The smaller the value for l that we choose the stronger the influence of the clustering. This suggests gradually *increasing* l during the course of optimization until, when it reaches N, the function being optimized is the original objective function itself.

3.4. Example

To find out if this aggregation strategy works, we tested it on the 'partitioned-circle' dataset discussed above, using SNE (Eqs. (3) and (4)) as the reference method to be improved. The results are depicted in Fig. 1(d). Here and in the other experiments, we used a simple gradient descent line-search to allow for a fair comparison between the different optimization approaches. We used M as defined in Eq. (12) and obtained K by centering a Gaussian kernel on each data-point. The plot shows the learning curves obtained from using a constant value for l (we set $l = 2$, after discarding the first eigenvalue—dashed line) vs. adapting l (solid line). For the adaptation, we started with $l = 1$ and increased it by one after each 1d-search.

The plot shows that the error is indeed going down faster in the beginning of the optimization. The problem of reaching a local minimum of the wrong objective function is reflected in the plot. It results in a final error that is larger than the error that we get from running SNE unmodified. A significant improvement is obtained when using the adaptive strategy. Here, the result is not only a speed-up, but also a better local minimum (note the use of a logarithmic scale on both axes; 200 iterations are depicted.)

4. A computational shortcut

It is possible to derive a computational shortcut that allows us to completely drop the eigenvalue decomposition as well as the necessity to choose l. Instead of cutting off the expansion after l terms, we might keep the entire expansion, but *weight* the contribution of each a_i with a factor that corresponds to the degree to which we believe that the corresponding eigenvector tells us something about the underlying clustering of the data. When we use the eigenvalues λ_i themselves as the weights, we obtain the particularly convenient form

$$\hat{G} = \mathrm{G}\mathrm{V}\mathrm{D}\mathrm{V}^{\mathrm{T}} = \mathrm{G}\mathrm{M}. \tag{16}$$

That is, we can get the optimized gradient $\hat{G}$ from G simply by post-multiplication with M.

It is also possible to obtain an 'annealing'-strategy analogous to the scheme of gradually increasing l that we used before. We can for example use a spectral transformation in order to control the steepness of the spectrum of M. By gradually decreasing the steepness we can make sure that the influence of M gets gradually less biased towards the leading eigenvalues.

A way to control the steepness of the spectrum would be to replace D by (renormalized) powers of itself:

$$D \leftarrow \frac{D^\nu}{\mathrm{tr}(D^\nu)},$$

where large ν will enhance the steepness and a choice $0<\nu<1$ will 'flatten' the spectrum. For M positive definite and symmetric we can achieve this implicitly, i.e. without actually needing to explicitly perform the eigen-decomposition, by replacing

$$M \leftarrow \frac{M^\nu}{\mathrm{tr}(M^\nu)}, \tag{17}$$

which simply follows from $M^\nu = VD^\nu V^{\mathrm{T}}$ and $\mathrm{tr}(VD^\nu V^{\mathrm{T}} = \mathrm{tr}(D^\nu)$ for orthogonal V.

Note that the shortcut, possibly in conjunction with the spectral transformation based annealing strategy, allows us to use the spectral speedup strategy without ever needing to explicitly compute an eigen-decomposition. This result is rather convenient. The spectral guidance strategy in general allows us to enjoy the advantages of spectral methods without having to give up on important properties of iterative methods. It does so by embedding a spectral algorithm in the optimization routine of an iterative method. The implicit approach makes it possible to do that without ever making use of any form of spectral algorithm at all.

5. Interpretation as power method

As pointed out in (Memisevic & Hinton, 2005a), it is possible to interpret the shortcut-approach as an implicit use of the power method for finding eigenvectors (Golub & Van Loan, 1991). According to this interpretation, the post-multiplication of the gradient—in particular for large ν—may be viewed as a projection onto an approximation to the leading eigenvector of the original similarity matrix. In other words, we project (each latent component of) the gradient onto the approximate solution of a spectral method.

The similarity matrices used for spectral clustering (e.g. Eq. (12)) are, as pointed out above, generally not the same as those used in dimensionality reduction. However, since the procedures for the construction of these matrices are very similar, it is reasonable to expect that the low-dimensional coordinates obtained from the eigen-decomposition of these matrices give reasonable representations for the purpose of NLDR. The converse, i.e. the fact that dimensionality reduction methods such as kernel PCA might give reasonable clustering results if the low-dimensional representations are clustered using for example k-means, has actually been pointed out in the literature before (see, e.g. Ng et al., 2002 and references therein).

Some support for the power-method view is given by the fact that the approach works best for low-dimensional (in particular $q=1$) latent spaces. Since according to the power-method interpretation we project *each row* of the gradient onto (the same) solution of a spectral method, this result might actually be expected under this view. For $q>1$ a positive effect could then still be achieved by this projection, since it might help 'untangle' the random initialization to some degree and provide at least a starting point from which a better global optimum may be found.

An interesting question in this context would be how other 'projection-matrices' than the clustering matrix defined in Eq. (12) would perform when used in the spectral gradient approach. In particular the data-space similarity matrix of an NLDR method might—according to the power-method perspective—be suitable. We report on some comparisons to the kernel PCA similarity matrix in Section 6, in which the results are clearly in favor of the clustering matrix. These results however might be biased and could depend on several free parameters in the problem setting, such as problem size (number of data points), choice of 'annealing' strategy, choice of kernel bandwidths, etc. To give a general answer to this question is therefore still an open problem—and it is probably not even a well-posed one.

5.1. Power method for multi-dimensional latent spaces

Given the power method interpretation, a more reasonable strategy than projecting each latent component of the gradient onto the approximate leading eigenvector of M would be to project the components onto separate eigenvectors. We could achieve this practically by projecting out a leading eigenvector after (approximately) extracting it, as would be done when explicitly computing multiple leading eigenvectors using the power method. That is, if we let g^j denote the jth row of the gradient matrix G, a possible strategy would be to iterate over the rows of G, using the updates

$$g^j \leftarrow g^j M^\nu \tag{18}$$

$$M \leftarrow M - \lambda_j g^{j^{\mathrm{T}}} g^j, \tag{19}$$

with

$$\lambda_j = \frac{g^j \mathrm{M} g^{j^{\mathrm{T}}}}{g^j g^{j^{\mathrm{T}}}}, \tag{20}$$

where in Eq. (19) the current eigenvector projected out. Note that applying such an approach from the beginning of the optimization process would encourage each latent component to specialize on a particular eigenvector, leading

to an ordering of the latent components according to eigenvalues. A well-known problem with the power method arises in the presence of multiple eigenvalues. This is particularly problematic when using the annealing strategy where we approach the identity matrix by decreasing ν with increasing iteration count. (In fact, when ν gets small, we are actually not even using a power-method-like algorithm at all anymore.) The simplest way to counter this problem would be to simply stop the process of projecting out eigenvectors as ν gets too small.

As is the case for the spectral gradient approach in general, we can afford more 'sloppiness' than in the context of an actual eigenvalue problem. Since a suitable annealing strategy will finally hand over all the responsibility to the standard gradient-based optimization anyway, the worst that can happen in the case of theoretical singularities, such as multiple eigenvalues, is that we do not achieve an improvement over the plain, unguided method. Regarding the speed of convergence, the method can in the worst case cause a slow-down rather than a speed-up during the initial optimization phase. As for local minima, we do not expect those found by the spectral gradient strategy to be worse in general than those that the unguided optimization would find based on some random initialization.

6. Experiments

6.1. Noisy helix

We applied the spectral guidance including the shortcut approach (Section 4) on the 'noisy helix' dataset (containing 300 points in 3d) shown in Fig. 3. In this experiment we used unsupervised kernel regression with leave-one-out regularization (Meinicke et al., in press) to find a one-dimensional embedding. For the spectral guidance, we computed M using Eq. (12) with K obtained from a Gaussian kernel with bandwidth 1.0 centered on every data-point.

We achieved the strongest improvement by using the spectral transformation based annealing. We started with $\nu = 10$ and decreased it by 1 after each iteration (after 10 iterations we simply continued with $\nu = 1$). The learning curves are shown in the upper-most plot of Fig. 3. They indicate that using the aggregation strategy here not only leads to an acceleration, but also mitigates the initialization problem. The final error achieved after 200 iterations using the spectral approach is close to zero and thereby significantly smaller than the one obtained from running UKR unmodified (again note the use of a logarithmic scale on both axes.)

A convenient property of the UKR method is that it defines a mapping from latent to data-space (Eq. (5)), which cannot be achieved using any (nonlinear) spectral method. Such a mapping is crucial for purposes such as noise reduction, since it allows us to define an orthogonal

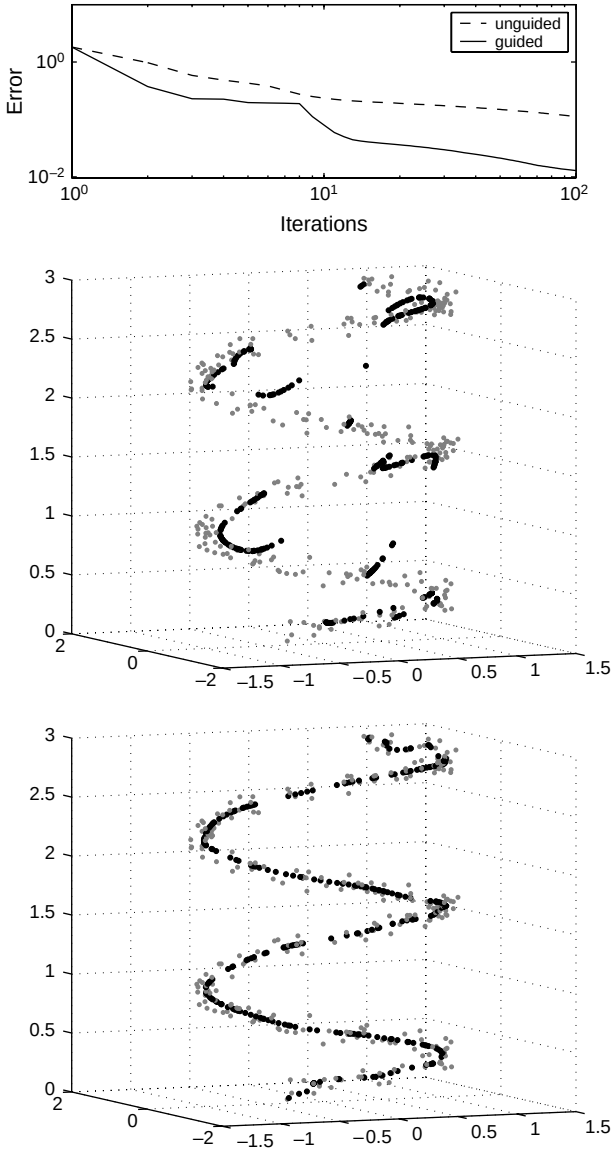

Fig. 3. Top plot: learning curves (see text). Bottom plots: training data (in gray) and UKR-manifolds (in black). The manifolds were produced by projecting the latent data representatives back into the data-space using the UKR 'forward' mapping.

projection onto the manifold based on the reconstruction error in data-space (Meinicke et al., in press). For up to three-dimensional data-spaces, we can also use the forward mapping to 'draw' the learned manifold in data-space by applying it to a set of densely sampled latent space elements (or the latent representatives of the training data themselves) in order to project these into the data-space. The two bottom plots of Fig. 3 show the learned helix manifolds obtained in this way. The superiority of the local minimum found by the spectral guidance becomes obvious: The manifold obtained from using standard UKR is partly disrupted and does not lie within the noise-range of the input data. The reason is that the poor local minimum found by the method does not reflect the correct one-dimensional structure of the latent

variables. The one obtained from applying the spectral modification, on the other hand, smoothly approximates the data.

6.2. 'S-curve' data and relation to power method

We used the 's-curve'-dataset depicted in Fig. 4 to obtain some empirical insight into the relation of the spectral gradient approach to the power method, that we discussed in Section 5. We computed a two-dimensional embedding using SNE with $h=(1/2)$. We compared the previously used matrix (Eq. (12)) to the kernel PCA similarity matrix, defined by

$$M^{\text{kpca}} = (I_{\text{N}} - \text{ee}^{\text{T}})K(I_{\text{N}} - \text{ee}^{\text{T}}) \tag{21}$$

where K is defined in Section 3.1 and e is the constant N-vector with entries $e_i = N^{-\frac{1}{2}}$.

We also compared the plain spectral gradient strategy with the 'simulated' power method for multiple eigenvectors (Section 5.1). As an annealing approach we set $\nu=20$ in the beginning and decreased by 1 after each iteration, until we reached 0. To deal with the degeneracy of multiple eigenvectors in the end of the annealing schedule, we simply stop the eigenvector extraction when $\nu=0$.

Figs. 5 and 6 show the learning curves for $N=200$ and $N=500$ data-points, respectively. For the spectral clustering matrix there is a significant speed-up at the beginning of the optimization in both cases and an improvement with respect to the local minimum that is found for $N=200$. Interestingly, when using the kernel PCA matrix, convergence is slower at the beginning of the optimization as compared to unguided SNE, but the method catches up later and also results in an improved local optimum for $N=200$. Interestingly, the multiple-eigenvector approach is actually able to significantly improve the local minimum in the $N=200$ case, but slightly degrades the performance for $N=500$. Overall, the performance is comparable to the plain spectral gradient approach.

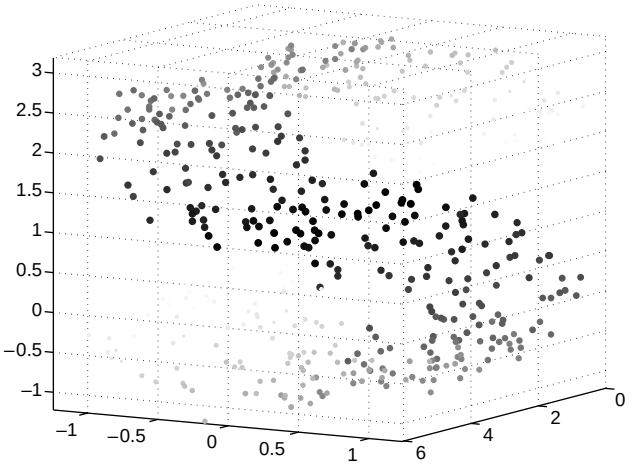

Fig. 4. 'Noisy s-curve' data ($N=500$).

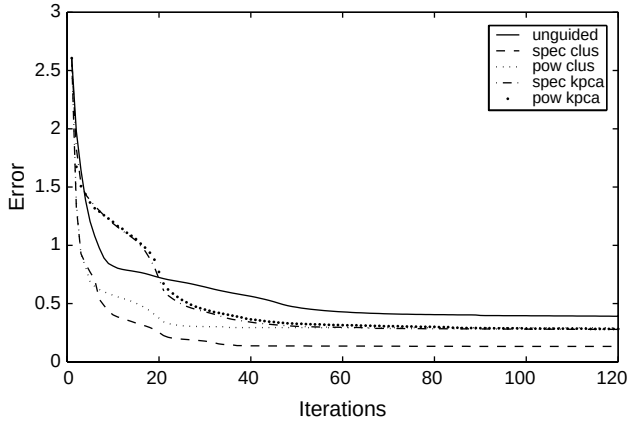

Fig. 5. Learning curves for $N=200$ points. Keys: 'spec-clus': Spectral gradient descent, using spectral clustering matrix; 'pow-clus': Spectral gradient descent, using spectral clustering matrix with simulated power method; 'spec-kpca': Spectral gradient descent, using kernel PCA matrix; 'pow-kpca': Spectral gradient descent, using kernel PCA matrix with simulated power method.

7. Conclusions and future work

We showed how using information contained in the eigenvectors of an affinity matrix can help improve iterative dimensionality reduction methods. When we require properties such as the presence of a 'forward' mapping, that are not provided by spectral methods, our approach can help to make iterative methods more efficient and can improve the quality of the local minima that they find.

Further experimentation is required to assess the usefulness of the approach in larger scale problems and to determine the influence of experimental parameters, such as dataset size, kernel bandwidths, etc. on the performance. A more principled way to set such parameters than currently available is of great importance for dimensionality reduction methods in general and is especially important for the applicability of the approach discussed in this paper.

Also, gradient descent line-search as an optimization method is well-known for its very poor convergence rate and is in general not recommended for use in actual real

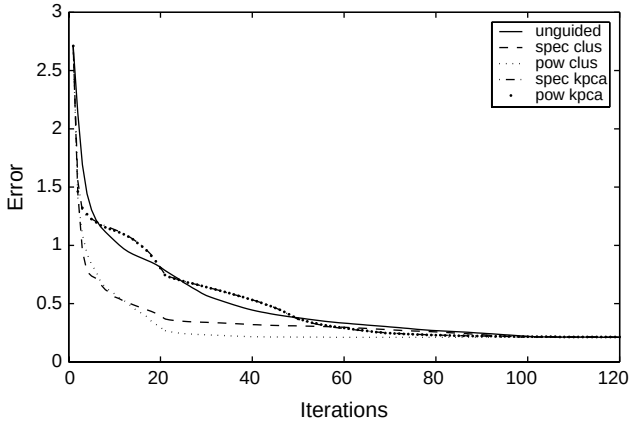

Fig. 6. Learning curves for $N=500$ points. See caption of Fig. 5 for keys.

world problems. Future work will examine ways to use the spectral gradient approach in the context of other optimization approaches, such as quasi-Newton or conjugate gradient methods.

Acknowledgements

This research was funded by a Government of Canada Award, NSERC, CFI, OIT. GH is a fellow of the Canadian Institute for Advanced Research and holds a Canada Research Chair in machine learning.

References

Belkin, M., & Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, *15*(6), 1373–1396.

Golub, G. H., & Van Loan, C. F. (1991). *Matrix computation* (2nd ed.). Baltimore, MD: John Hopkins University Press.

Härdle, W. (1990). *Applied nonparametric regression*. Cambridge: Cambridge University Press.

Hinton, G., Roweis, S. (2003). Stochastic neighbor embedding. In S. Becker, S.T. Obermayer, K. (Eds.), *Advances in Neural Information Processing Systems 15*. MIT Press.

Horn, R. A., & Johnson, C. R. (1986). *Matrix analysis*. New York: Cambridge University Press.

Lawrence, N. D. (2004). Gaussian process latent variable models for visualisation of high dimensional data. In S. Thrun, L. Saul, & B. Schölkopf (Eds.), *Advances in neural information processing systems 16*. Cambridge, MA: MIT Press.

Maila, M., & Shi, J. (2001). *A random walks view of spectral segmentation AI and STATISTICS (AISTATS) 2001*.

Meinicke, P., Klanke, S., Memisevic, R., Ritter, H. (2005). Principal surfaces from unsupervised kernel regression. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 27(9), 1379–1391.

Memisevic, R., & Hinton, G. (2005a). Embedding via clustering: Using spectral information to guide dimensionality reduction. *Proceedings of the International Joint Conference on Neural Networks, July 31–August 4, 2005, Montreal, Canada*.

Memisevic, R., & Hinton, G. (2005b). Multiple relational embedding. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems 17*. Cambridge, MA: MIT Press.

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in neural information processing systems 14*. Cambridge, MA: MIT Press.

Roweis, S., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science* , 290.

Schölkopf, B., Smola, A., & Müller, K. R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*(5), 1299–1319.

Weiss, Y. (1999). Segmentation using eigenvectors: A unifying view *ICCV '99: Proceedings of the international conference on computer vision-volume 2*. Silver Spring, MD: IEEE Computer Society p. 975.