Dimensionality Reduction by Learning an Invariant Mapping

Raia Hadsell, Sumit Chopra, Yann LeCun
The Courant Institute of Mathematical Sciences
New York University, 719 Broadway, New York, NY 1003, USA.
http://www.cs.nyu.edu/~yann

(November 2005. To appear in CVPR 2006)

Abstract

Dimensionality reduction involves mapping a set of high dimensional input points onto a low dimensional manifold so that "similar" points in input space are mapped to nearby points on the manifold. Most existing techniques for solving the problem suffer from two drawbacks. First, most of them depend on a meaningful and computable distance metric in input space. Second, they do not compute a "function" that can accurately map new input samples whose relationship to the training data is unknown. We present a method - called Dimensionality Reduction by Learning an Invariant Mapping (DrLIM) - for learning a globally coherent non-linear function that maps the data evenly to the output manifold. The learning relies solely on neighborhood relationships and does not require any distance measure in the input space. The method can learn mappings that are invariant to certain transformations of the inputs, as is demonstrated with a number of experiments. Comparisons are made to other techniques, in particular LLE.

1. Introduction

Modern applications have steadily expanded their use of complex, high dimensional data. The massive, high dimensional image datasets generated by biology, earth science, astronomy, robotics, modern manufacturing, and other domains of science and industry demand new techniques for analysis, feature extraction, dimensionality reduction, and visualization.

Dimensionality reduction aims to translate high dimensional data to a low dimensional representation such that similar input objects are mapped to nearby points on a manifold. Most existing dimensionality reduction techniques have two shortcomings. First, they do not produce a *function* (or a mapping) from input to manifold that can be applied to new points whose relationship to the training points is unknown. Second, many methods presuppose the existence of a meaningful (and computable) distance metric in the input space.

For example, Locally Linear Embedding (LLE) [15] linearly combines input vectors that are identified as neighbors. The applicability of LLE and similar methods to image data is limited because linearly combining images only makes sense for images that are perfectly registered and very similar. Laplacian Eigenmap [2] and Hessian LLE [8] do not require a meaningful metric in input space (they merely require a list of neighbors for every sample), but as with LLE, new points whose relationships with training samples are unknown cannot be processed. Out-of-sample extensions to several dimensionality reduction techniques have been proposed that allow for consistent embedding of new data samples without recomputation of all samples [3]. These extensions, however, assume the existence of a computable kernel function that is used to generate the neighborhood matrix. This dependence is reducible to the dependence on a computable distance metric in input space.

Another limitation of current methods is that they tend to cluster points in output space, sometimes densely enough to be considered degenerate solutions. Rather, it is sometimes desirable to find manifolds that are uniformly covered by samples.

The method proposed in the present paper, called Dimensionality Reduction by Learning an Invariant Mapping (DrLIM), provides a solution to the above problems. DrLIM is a method for learning a globally coherent non-linear function that maps the data to a low dimensional manifold. The method presents four essential characteristics:

- It only needs neighborhood relationships between training samples. These relationships could come from prior knowledge, or manual labeling, and be independent of any distance metric.
- It may learn functions that are invariant to complicated non-linear trnasformations of the inputs such as lighting changes and geometric distortions.
- The learned function can be used to map new samples not seen during training, with no prior knowledge.

- The mapping generated by the function is in some sense “smooth” and coherent in the output space.

A contrastive loss function is employed to learn the parameters W of a parameterized function G_W, in such a way that neighbors are pulled together and non-neighbors are pushed apart. Prior knowledge can be used to identify the neighbors for each training data point.

The method uses an energy based model that uses the given neighborhood relationships to learn the mapping function. For a family of functions G, parameterized by W, the objective is to find a value of W that maps a set of high dimensional inputs to the manifold such that the euclidean distance between points on the manifold, $D_W(\vec{X}_1, \vec{X}_2) = ||G_W(\vec{X}_1) - G_W(\vec{X}_2)||_2$ approximates the “semantic similarity”of the inputs in input space, as provided by a set of neighborhood relationships. No assumption is made about the function G_W except that it is differentiable with respect to W.

1.1. Previous Work

The problem of mapping a set of high dimensional points onto a low dimensional manifold has a long history. The two classical methods for the problem are Principal Component Analysis (PCA) [7] and Multi-Dimensional Scaling (MDS) [6]. PCA involves the projection of inputs to a low dimensional subspace that maximizes the variance. In MDS, one computes the projection that best preserves the pairwise distances between input points. However both the methods - PCA in general and MDS in the classical scaling case (when the distances are euclidean distances) - generate a linear embedding.

In recent years there has been a lot of activity in designing non-linear *spectral methods* for the problem. These methods involve solving the eigenvalue problem for a particular matrix. Recently proposed algorithms include ISOMAP (2000) by Tenenbaum *et al*. [1], Local Linear Embedding - LLE (2000) by Roweis and Saul [15], Laplacian Eigenmaps (2003) due to Belkin and Niyogi [2] and Hessian LLE (2003) by Donoho and Grimes [8]. All the above methods have three main steps. The first is to identify a list of neighbors of each point. Second, a gram matrix is computed using this information. Third, the eigenvalue problem is solved for this matrix. The methods differ in how the gram matrix is computed. None of these methods attempt to compute a *function* that could map a new, unknown data point without recomputing the entire embedding and without knowing its relationships to the training points. Out-of-sample extensions to the above methods have been proposed by Bengio *et al*. in [3], but they too rely on a predetermined computable distance metric.

Along a somewhat different line Schöelkopf *et al*. in 1998 [13] proposed a non-linear extension of PCA, called Kernel PCA. The idea is to non-linearly map the inputs to a high dimensional feature space and then extract the principal components. The algorithm first expresses the PCA computation solely in terms of dot products and then exploits the kernel trick to implicitly compute the high dimensional mapping. The choice of kernels is crucial: different kernels yield dramatically different embeddings. In recent work, Weinberger *et al*. in [11, 12] attempt to learn the kernel matrix when the high dimensional input lies on a low dimensional manifold by formulating the problem as a semidefinite program. There are also related algorithms for clustering due to Shi and Malik [14] and Ng *et al*. [17].

The proposed approach is different from these methods; it learns a function that is capable of consistently mapping new points unseen during training. In addition, this function is not constrained by simple distance measures in the input space. The learning architecture is somewhat similar to the one discussed in [4, 5].

Section 2 describes the general framework, the loss function, and draws an analogy with a mechanical spring system. The ideas in this section are made concrete in section 3. Here various experimental results are given and comparisons to LLE are made.

2. Learning the Low Dimensional Mapping

The problem is to find a function that maps high dimensional input patterns to lower dimensional outputs, given neighborhood relationships between samples in input space. The graph of neighborhood relationships may come from information source that may not be available for test points, such as prior knowledge, manual labeling, etc. More precisely, given a set of input vectors $\mathcal{I} = \{\vec{X}_1, \ldots, \vec{X}_P\}$, where $\vec{X}_i \in \Re^D, \forall i = 1, \ldots, n$, find a parametric function $G_W : \Re^D \longrightarrow \Re^d$ with $d \ll D$, such that it has the following properties:

1. Simple distance measures in the output space (such as euclidean distance) should approximate the *neighborhood relationships* in the input space.

2. The mapping should not be constrained to implementing simple distance measures in the input space and should be able to learn invariances to complex transformations.

3. It should be *faithful* even for samples whose neighborhood relationships are unknown.

2.1. The Contrastive Loss Function

Consider the set $\mathcal{I}$ of high dimensional training vectors $\vec{X}_i$. Assume that for each $\vec{X}_i \in \mathcal{I}$ there is a set $\mathcal{S}_{\vec{X}_i}$ of training vectors that are deemed similar to $\vec{X}_i$. This set can be computed by some prior knowledge - invariance to distortions or temporal proximity, for instance - which does not depend on a simple distance. A meaningful mapping from high to low dimensional space maps similar input vectors to nearby points on the output manifold and dissimilar vectors to distant points. A new loss function whose minimization can produce such a function is now introduced. Unlike conventional learning systems where the loss function is a sum over samples, the loss function here runs over *pairs of samples*. Let $\vec{X}_1, \vec{X}_2 \in \mathcal{I}$ be a pair of input vectors shown to the system. Let Y be a binary label assigned to this pair. $Y = 0$ if $\vec{X}_1$ and $\vec{X}_2$ are deemd similar, and $Y = 1$ if they are deemed dissimilar. Define the parameterized distance function to be learned D_W between $\vec{X}_1, \vec{X}_2$ as the euclidean distance between the outputs of G_W. That is,

$$D_W(\vec{X}_1, \vec{X}_2) = \|G_W(\vec{X}_1) - G_W(\vec{X}_2)\|_2 \tag{1}$$

To shorten notation, $D_W(\vec{X}_1, \vec{X}_2)$ is written D_W. Then the loss function in its most general form is

$$\mathcal{L}(W) = \sum_{i=1}^{P} L(W, (Y, \vec{X}_1, \vec{X}_2)^i) \tag{2}$$

$$L(W, (Y, \vec{X}_1, \vec{X}_2)^i) = (1 - Y)L_S\left(D_W^i\right) + YL_D\left(D_W^i\right) \tag{3}$$

where $(Y, \vec{X}_1, \vec{X}_2)^i$ is the i-th labeled sample pair, L_S is the partial loss function for a pair of similar points, L_D the partial loss function for a pair of dissimilar points, and P the number of training pairs (which may be as large as the square of the number of samples).

L_S and L_D must be designed such that minimizing L with respect to W would result in low values of D_W for similar pairs and high values of D_W for dissimilar pairs.

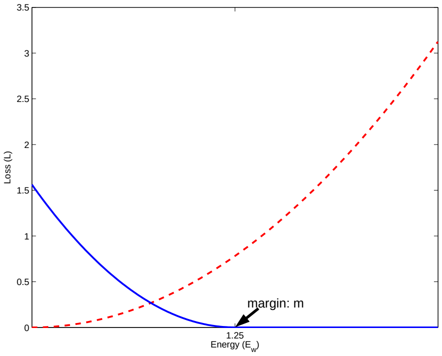

Figure 1. Graph of the loss function L against the energy D_W. The dashed (red) line is the loss function for the similar pairs and the solid (blue) line is for the dissimilar pairs.

The exact loss function is

$$L(W, Y, \vec{X}_1, \vec{X}_2) =$$
$$(1 - Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}\{max(0, m - D_W)\}^2 \tag{4}$$

where $m > 0$ is a margin. The margin defines a radius around $G_W(\vec{X})$. Dissimilar pairs contribute to the loss function only if their distance is within this radius (See figure 1). The contrastive term involving dissimilar pairs, L_D, is crucial. Simply minimizing $D_W(\vec{X}_1, \vec{X}_2)$ over the set of all similar pairs will usually lead to a collapsed solution, since D_W and the loss L could then be made zero by setting G_W to a constant. Most energy-based models require the use of an explicit contrastive term in the loss function.

2.2. Spring Model Analogy

An analogy to a particular mechanical spring system is given to provide an intuition of what is happening when the loss function is minimized. The outputs of G_W can be thought of as masses attracting and repelling each other with springs. Consider the equation of a spring

$$F = -KX \tag{5}$$

where F is the force, K is the spring constant and X is the displacement of the spring from its rest length. A spring is *attract-only* if its rest length is equal to zero. Thus any positive displacement X will result in an attractive force between its ends. A spring is said to be *m-repulse-only* if its rest length is equal to m. Thus two points that are connected with a *m-repulse-only* spring will be pushed apart if X is less than m. However this spring has a special property that if the spring is stretched by a length $X > m$, then no attractive force brings it back to rest length. Each point is connected to other points using these two kinds of springs. Seen in the light of the loss function, each point is connected by *attract-only* springs to *similar* points, and is connected by *m-repulse-only* spring to *dissimilar* points. See figure 2.

Consider the loss function $L_S(W, \vec{X}_1, \vec{X}_2)$ associated with similar pairs.

$$L_S(W, \vec{X}_1, \vec{X}_2) = \frac{1}{2}(D_W)^2 \tag{6}$$

The loss function L is minimized using the stochastic gradient descent algorithm. The gradient of L_S is

$$\frac{\partial L_S}{\partial W} = D_W \frac{\partial D_W}{\partial W} \tag{7}$$

Comparing equations 5 and 7, it is clear that the gradient $\frac{\partial L_S}{\partial W}$ of L_S gives the attractive force between the two points. $\frac{\partial D_W}{\partial W}$ defines the spring constant K of the spring and D_W, which is the distance between the two points, gives the perturbation X of the spring from its rest length. Clearly, even a small value of D_W will generate a gradient (force) to decrease D_W. Thus the similar loss function corresponds to the *attract-only* spring (figure 2).

Now consider the partial loss function L_D.

$$L_D(W, \vec{X}_1, \vec{X}_2) = \frac{1}{2}(max\{0, m - D_W\})^2 \tag{8}$$

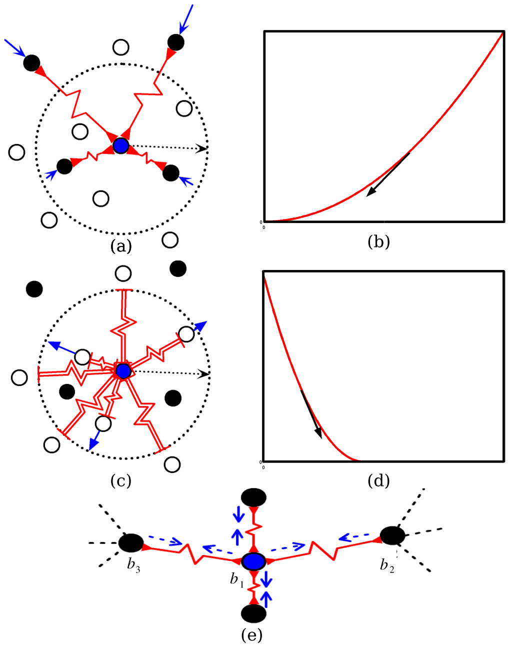

Figure 2. Figure showing the spring system. The solid circles represent points that are similar to the point in the center. The hollow circles represent dissimilar points. The springs are shown as red zigzag lines. The forces acting on the points are shown in blue arrows. The length of the arrows approximately gives the strength of the force. In the two plots on the right side, the x-axis is the distance D_W and the y-axis is the value of the loss function. (a). Shows the points connected to similar points with *attract-only* springs. (b). The loss function and its gradient associated with similar pairs. (c) The point connected only with dissimilar points inside the circle of radius m with *m-repulse-only* springs. (d) Shows the loss function and its gradient associated with dissimilar pairs. (e) Shows the situation where a point is pulled by other points in different directions, creating equilibrium.

When $D_W > m$, $\frac{\partial L_D}{\partial W} = 0$. Thus there is no gradient (force) on the two points that are dissimilar and are at a distance $D_W > m$. If $D_W < m$ then

$$\frac{\partial L_D}{\partial W} = -(m - D_W)\frac{\partial D_W}{\partial W} \qquad (9)$$

Again, comparing equations 5 and 9 it is clear that the dissimilar loss function L_D corresponds to the *m-repulse-only* spring; its gradient gives the force of the spring, $\frac{\partial D_W}{\partial W}$ gives the spring constant K and $(m - D_W)$ gives the perturbation X. The negative sign denotes the fact that the force is repulsive only. Clearly the force is maximum when $D_W = 0$ and absent when $D_W = m$. See figure 2.

Here, especially in the case of L_S, one might think that simply making $D_W = 0$ for all *attract-only* springs would put the system in equilibrium. Consider, however, figure 2e. Suppose b_1 is connected to b_2 and b_3 with *attract-only* springs. Then decreasing D_W between b_1 and b_2 will increase D_W between b_1 and b_3. Thus by minimizing the global loss function L over all springs, one would ultimately drive the system to its equilibrium state.

2.3. The Algorithm

The algorithm first generates the training set, then trains the machine.

Step 1: For each input sample $\vec{X}_i$, do the following:

(a) Using prior knowledge find the set of samples $\mathcal{S}_{\vec{X}_i} = \{\vec{X}_j\}_{j=1}^{p}$, such that $\vec{X}_j$ is deemed similar to $\vec{X}_i$.

(b) Pair the sample $\vec{X}_i$ with all the other training samples and label the pairs so that:
$Y_{ij} = 0$ if $\vec{X}_j \in \mathcal{S}_{\vec{X}_i}$, and $Y_{ij} = 1$ otherwise.

Combine all the pairs to form the labeled training set.

Step 2: Repeat until convergence:

(a) For each pair $(\vec{X}_i, \vec{X}_j)$ in the training set, do

i. If $Y_{ij} = 0$, then update W to decrease
$D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

ii. If $Y_{ij} = 1$, then update W to increase
$D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

This increase and decrease of euclidean distances in the output space is done by minimizing the above loss function.

3. Experiments

The experiments presented in this section demonstrate the invariances afforded by our approach and also clarify the limitations of techniques such as LLE. First we give details of the parameterized machine G_W that learns the mapping function.

3.1. Training Architecture

The learning architecture is similar to the one used in [4] and [5]. Called a *siamese* architecture, it consists of two copies of the function G_W which share the same set of parameters W, and a cost module. A loss module whose input is the output of this architecture is placed on top of it. The input to the entire system is a pair of images $(\vec{X}_1, \vec{X}_2)$ and a label Y. The images are passed through the functions, yielding two outputs $G(\vec{X}_1)$ and $G(\vec{X}_2)$. The cost module then generates the distance $D_W(G_W(\vec{X}_1), G_W(\vec{X}_2))$. The loss function combines D_W with label Y to produce the scalar loss L_S or L_D, depending on the label Y. The parameter W is updated using stochastic gradient. The gradients can be computed by back-propagation through the loss, the cost, and the two instances of G_W. The total gradient is the sum of the contributions from the two instances.

The experiments involving airplane images from the NORB dataset [10] use a 2-layer fully connected neural network as G_W. The number of hidden and output units used was 20 and 3 respectively. Experiments on the MNIST dataset [9] used a convolutional network as G_W (figure 3). Convolutional networks are trainable, non-linear learning machines that operate at pixel level and learn low-level features and high-level representations in an integrated manner. They are trained *end-to-end* to map images to outputs. Because of a structure of shared weights and multiple layers, they can learn optimal shift-invariant local feature detectors while maintaining invariance to geometric distortions of the input image.

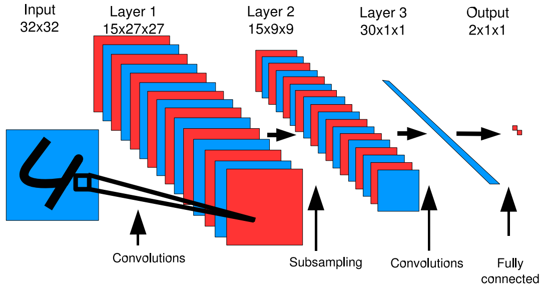

Figure 3. Architecture of the function G_W (a convolutional network) which was learned to map the MNIST data to a low dimensional manifold with invariance to shifts.

The layers of the convolutional network comprise a convolutional layer C_1 with 15 feature maps, a subsampling layer S_2, a second convolutional layer C_3 with 30 feature maps, and fully connected layer F_3 with 2 units. The sizes of the kernels for the C_1 and C_3 were 6x6 and 9x9 respectively.

3.2. Learned Mapping of MNIST samples

The first experiment is designed to establish the basic functionality of the DrLIM approach. The neighborhood graph is generated with euclidean distances and no prior knowledge.

The training set is built from 3000 images of the handwritten digit 4 and 3000 images of the handwritten digit 9 chosen randomly from the MNIST dataset [9]. Approximately 1000 images of each digit comprised the test set. These images were shuffled, paired, and labeled according to a simple euclidean distance measure: each sample $\vec{X}_i$ was paired with its 5 nearest neighbors, producing the set S_{X_i}. All other possible pairs were labeled dissimilar, producing 30,000 similar pairs and on the order of 18 million dissimilar pairs.

The mapping of the test set to a 2D manifold is shown in figure 4. The lighter-colored blue dots are 9's and the darker-colored red dots are 4's. Several input test samples are shown next to their manifold positions. The 4's and 9's are in two somewhat overlapping regions, with an overall organization that is primarily determined by the slant angle of the samples. The samples are spread rather uniformly in the populated region.

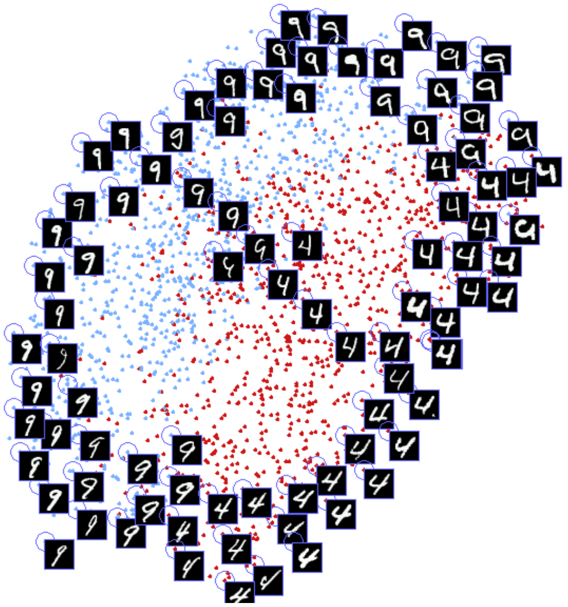

Figure 4. Experiment demonstrating the effectiveness of the DrLIM in a trivial situation with MNIST digits. A Euclidean nearest neighbor metric is used to create the local neighborhood relationships among the training samples, and a mapping function is learned with a convolutional network. Figure shows the placement of the *test* samples in output space. Even though the neighborhood relationships among these samples are unknown, they are well organized and evenly distributed on the 2D manifold.

3.3. Learning a Shift-Invariant Mapping of MNIST samples

In this experiment, the DrLIM approach is evaluated using 2 categories of MNIST, distorted by adding samples that have been horizontally translated. The objective is to learn a 2D mapping that is invariant to horizontal translations.

In the distorted set, 3000 images of 4's and 3000 images of 9's are horizontally translated by -6, -3, 3, and 6 pixels and combined with the originals, producing a total of 30,000 samples. The 2000 samples in the test set were distorted in the same way.

First the system was trained using pairs from a euclidean distance neighborhood graph (5 nearest neighbors per sample), as in experiment 1. The large distances between translated samples creates a disjoint neighborhood relationship graph and the resulting mapping is disjoint as well. The output points are clustered according to the translated position of the input sample (figure 5). Within each cluster, however, the samples are well organized and evenly distributed.

For comparison, the LLE algorithm was used to map the distorted MNIST using the same euclidean distance neighborhood graph. The result was a degenerate embedding in which differently registered samples were completely separated (figure 6). Although there is sporadic local organiza-

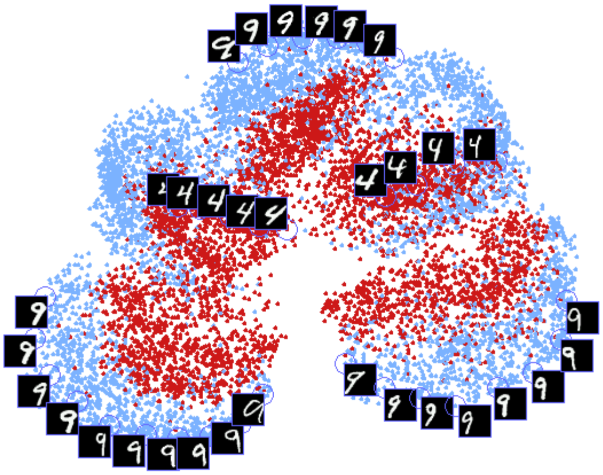

Figure 5. This experiment shows the effect of a simple distance-based mapping on MNIST data with horizontal translations added (-6, -3, +3, and +6 pixels). Since translated samples are far apart, the manifold has 5 distinct clusters of samples corresponding to the 5 translations. Note that the clusters are individually well-organized, however. Results are on test samples, unseen during training.

tion, there is no global coherence in the embedding.

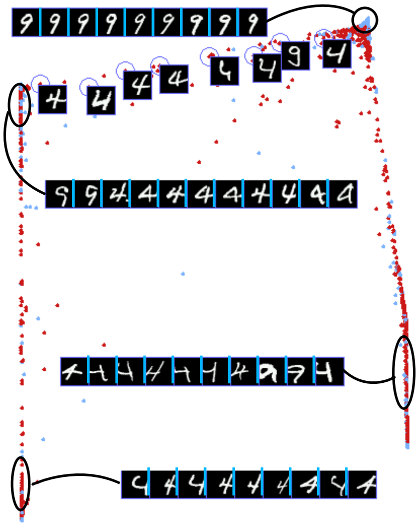

Figure 6. LLE's embedding of the distorted MNIST set with horizontal translations added. Most of the untranslated samples are tightly clustered at the top right corner, and the translated samples are grouped at the sides of the output.

In order to make the mapping function invariant to translation, the euclidean nearest neighbors were supplemented with pairs created using *prior knowledge*. Each sample was paired with *(a)* its 5 nearest neighbors, *(b)* its 4 translations, and *(c)* the 4 translations of each of its 5 nearest neighbors. Additionally, each of the sample's 4 translations was paired with *(d)* all the above nearest neighbors and translated samples. All other possible pairs are labeled as dissimilar.

The mapping of the test set samples is shown in figure 7. The lighter-colored blue dots are 4's and the darker-colored red dots are 9's. As desired, there is no organization on the basis of translation; in fact, translated versions of a given character are all tighlty packed in small regions on the manifold.

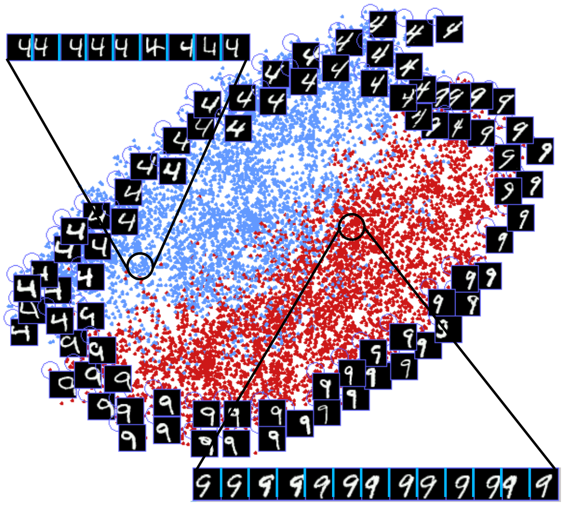

Figure 7. This experiment measured DrLIM's success at learning a mapping from high-dimensional, shifted digit images to a 2D manifold. The mapping is invariant to translations of the input images. The mapping is well-organized and globally coherent. Results shown are the test samples, whose neighborhood relations are unknown. Similar characters are mapped to nearby areas, regardless of their shift.

3.4. Mapping Learned with Temporal Neighborhoods and Lighting Invariance

The final experiment demonstrates dimensionality reduction on a set of images of a single object. The object is an airplane from the NORB [10] dataset with uniform backgrounds. There are a total of 972 images of the airplane under various poses around the viewing half-sphere, and under various illuminations. The views have 18 azimuths (every 20 degrees around the circle), 9 elevations (from 30 to 70 degrees every 5 degrees), and 6 lighting conditions (4 lights in various on-off combinations). The objective is to learn a globally coherent mapping to a 3D manifold that is invariant to lighting conditions. A pattern based on temporal continuity of the camera was used to construct a neighborhood graph; images are similar if they were taken from contiguous elevation or azimuth regardless of lighting. Images may be neighbors even if they are very distant in terms of Eucliden distance in pixel space, due to different lighting.

The dataset was split into 660 training images and a 312 test images. The result of training on all 10989 similar pairs and 206481 dissimilar pairs is a 3-dimensional manifold in the shape of a cylinder (see figure 8). The circumference of the cylinder corresponds to change in azimuth in input space, while the height of the cylinder corresponds to elevation in input space. The mapping is completely invariant to lighting. This outcome is quite remarkable. Using only local neighborhood relationships, the learned manifold corresponds globally to the positions of the camera as it produced the dataset.

Viewing the weights of the network helps explain how the mapping learned illumination invariance (see figure 9). The concentric rings match edges on the airplanes to a particular azimuth and elevation, and the rest of the weights are close to 0. The dark edges and shadow of the wings, for example, are relatively consistent regardless of lighting.

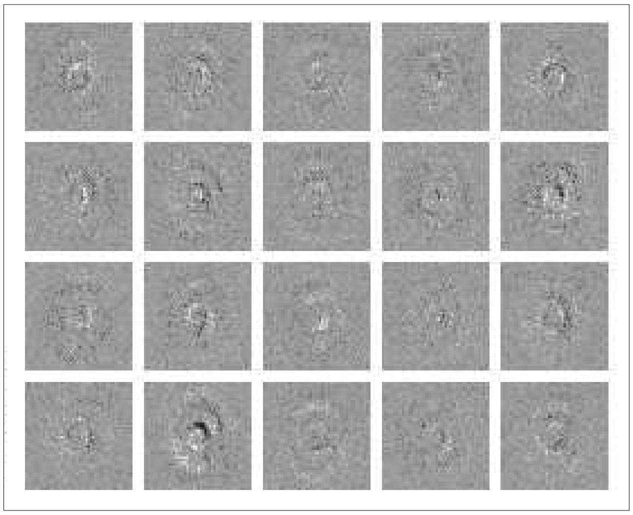

Figure 9. The weights of the 20 hidden units of a fully-connected neural network trained with DrLIM on airplane images from the NORB dataset. Since the camera rotates 360^o around the airplane and the mapping must be invariant to lighting, the weights are zero except to detect edges at each azimuth and elevation; thus the concentric patterns.

For comparison, the same neighborhood relationships defined by the prior knowledge in this experiment were used to create an embedding using LLE. Although arbitrary neighborhoods can be used in the LLE algorithm, the algorithm computes linear reconstruction weights to embed the samples, which severely limits the desired effect of using distant neighbors. The embedding produced by LLE is shown (see figure 10). Clearly, the 3D embedding is not invariant to lighting, and the organization of azimuth and elevation does not reflect the real topology neighborhood graph.

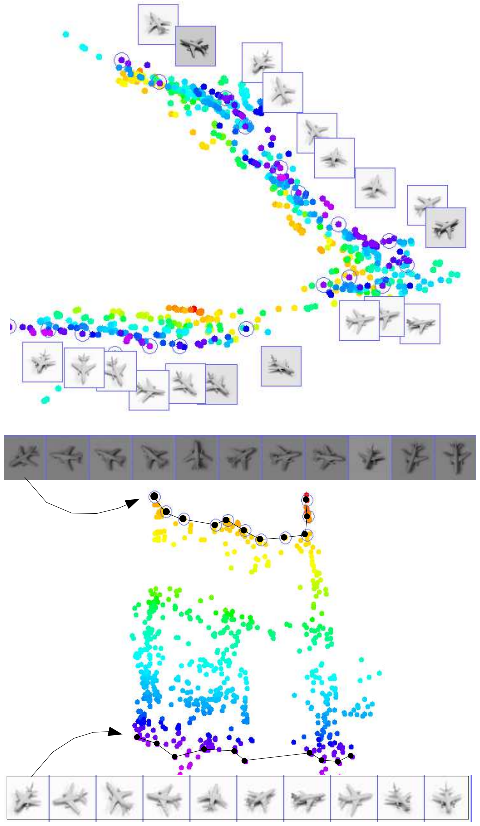

Figure 10. 3d embedding of NORB images by LLE algorithm. The neighborhood graph was constructed to create invariance to lighting, but the linear reconstruction weights of LLE force it organize the embedding by lighting. The shape of the embedding resembles a folded paper. The top image shows the ’v’ shape of the fold and the lower image looks into the valley of the fold.

4. Discussion and Future Work

The experiments presented here demonstrate that, unless prior knowledge is used to create invariance, variabilities such as lighting and registration can dominate and distort the outcome of dimensionality reduction. The proposed approach, DrLIM, offers a solution: it is able to learn an invariant mapping to a low dimensional manifold using prior knowledge. The complexity of the invariances that can be learned are only limited by the power of the parameterized function G_W. The function maps inputs that evenly cover a manifold, as can be seen by the experimental results. It also faithfully maps new, unseen samples to meaningful locations on the manifold.

The strength of DrLIM lies in the contrastive loss function. By using a separate loss function for similar and dis-

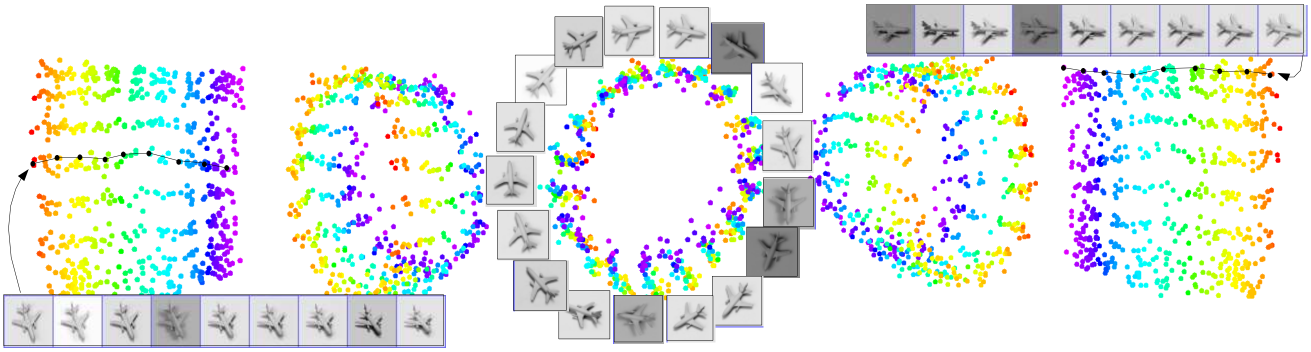

Figure 8. Test set results: the DrLIM approach learned a mapping to 3d space for images of a single airplane (extracted from NORB dataset). The output manifold is shown under five different viewing angles. The manifold is roughly cylindrical with a systematic organization: along the circumference varies azimuth of camera in the viewing half-sphere. Along the height varies the camera elevation in the viewing sphere. The mapping is invariant to the lighting condition, thanks to the prior knowledge built into the neighborhood relationships.

similar pairs, the system avoids collapse to a constant function and maintains an equilibrium in output space, much as a mechanical system of interconnected springs does.

The experiments with LLE show that LLE is most useful where the input samples are locally very similar and well-registered. If this is not the case, then LLE may give degenerate results. Although it is possible to run LLE with arbitrary neighborhood relationships, the linear reconstruction of the samples negates the effect of very distant neighbors. Other dimensionality reduction methods have avoided this limitation, but none produces a function that can accept new samples without recomputation or prior knowledge.

Creating a dimensionality reduction mapping using prior knowledge has other uses. Given the success of the NORB experiment, in which the positions of the camera were learned from prior knowledge of the temporal connections between images, it may be feasible to learn a robot's position and heading from image sequences.

References

[1] J. B. Tenenbaum, V. DeSilva, and J. C. Langford. A global geometric framework for non linear dimensionality reduction. *Science*, 290:2319–2323, 2000. 2

[2] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in Neural Information Processing Systems*, 15(6):1373–1396, 2003. 1, 2

[3] Y. Bengio, J. F. Paiement, and P. Vincent. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. *Advances in Neural Information Processing Systems*, 16, 2004. In S. Thrun, L.K. Saul and B. Scholkopf, editors. Cambrige, MA. MIT Press. 1, 2

[4] J. Bromley, I. Guyon, Y. LeCun, E. Sackinger, and R. Shah. Signature verification using a siamese time delay neural network. J. Cowan and G. Tesauro (eds) *Advances in Neural Information Processing Systems*, 1993. 2, 4

[5] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with applications to face verificaton. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-05)*, 1:539–546, 2005. 2, 4

[6] T. Cox and M. Cox. Multidimensional scaling. *London: Chapman and Hill*, 1994. 2

[7] T. I. Jolliffe. Principal component analysis. *New York: Springer-Verlag*, 1986. 2

[8] D. L. Donoho and C. E. Grimes. Hessian eigenmap: Locally linear embedding techniques for high dimensional data. *Proceedings of the National Academy of Arts and Sciences*, 100:5591–5596, 2003. 1, 2

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 5

[10] Y. LeCun, F. J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-04)*, 2:97–104, 2004. 4, 6

[11] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-04)*, 2:988–995, 2004. 2

[12] K. Q. Weinberger, F. Sha, and L. K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. *In Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)*, pages 839–846, 2004. 2

[13] B. Schőelkopf, A. J. Smola, and K. R. Muller. Nonlinear component analysis as a kernel eigen-value problem. *Neural Computation*, 10:1299–1219, 1998. 2

[14] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pages 888–905, 2000. 2

[15] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290. 1, 2

[16] P. Vincent and Y. Bengio. A neural support vector network architecture with adaptive kernels. *In Proc. of the International Joint Conference on Neural Networks*, 5, July 2000.

[17] A. Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 14:849–856, 2002. 2