A tractable latent variable model for nonlinear dimensionality reduction

Lawrence K. Saul

*Department of Computer Science and Engineering, University of California San Diego, La Jolla, CA 92093-0404

Edited by Nathan Kallus, Cornell University, New York, NY, and accepted by Editorial Board Member Donald B. Rubin May 6, 2020 (received for review September 15, 2019)

We propose a latent variable model to discover faithful low-dimensional representations of high-dimensional data. The model computes a low-dimensional embedding that aims to preserve neighborhood relationships encoded by a sparse graph. The model both leverages and extends current leading approaches to this problem. Like t-distributed Stochastic Neighborhood Embedding, the model can produce two- and three-dimensional embeddings for visualization, but it can also learn higher-dimensional embeddings for other uses. Like LargeVis and Uniform Manifold Approximation and Projection, the model produces embeddings by balancing two goals—pulling nearby examples closer together and pushing distant examples further apart. Unlike these approaches, however, the latent variables in our model provide additional structure that can be exploited for learning. We derive an Expectation–Maximization procedure with closed-form updates that monotonically improve the model’s likelihood: In this procedure, embeddings are iteratively adapted by solving sparse, diagonally dominant systems of linear equations that arise from a discrete graph Laplacian. For large problems, we also develop an approximate coarse-graining procedure that avoids the need for negative sampling of nonadjacent nodes in the graph. We demonstrate the model’s effectiveness on datasets of images and text.

unsupervised learning | nonlinear dimensionality reduction

A common problem arises in many fields of science and engineering—how to discover low-dimensional representations of high-dimensional data. These representations can help to visualize complex patterns; they can also support the faster indexing and querying of large data collections.

Current leading approaches in this area have demonstrated remarkable successes. Algorithms such as t-distributed Stochastic Neighborhood Embedding (t-SNE) (1), LargeVis (2), and Uniform Manifold Approximation and Projection (UMAP) (3) have produced stunning visualizations of high-dimensional datasets; they have also tackled extremely large problems with efficient data structures (4, 5) and clever randomizations (6, 7). In practical applications, they have largely supplanted earlier approaches, based on spectral methods (8–12), for manifold learning and nonlinear dimensionality reduction (NLDR).

The models for t-SNE, LargeVis, and UMAP share a common element: They all appeal to probabilistic notions of similarity and dissimilarity. Despite this shared grounding, however, these approaches bear little resemblance to other canonical probabilistic models for unsupervised learning. Problems such as factor analysis (13) and clustering (14) are amenable to especially tractable latent variable models (LVMs). The problem of NLDR is equally simple to state: Given a set of high-dimensional (observed) inputs \( \{x_i\}_{i=1}^N \), infer a corresponding set of low-dimensional (unobserved) outputs \( \{\mu_i\}_{i=1}^N \) such that only nearby inputs are mapped to nearby outputs. The formulation in these terms seems to lend itself to latent variable modeling.

In this paper, we make this connection explicit. We develop an LVM that provides another way of conceptualizing the recent breakthroughs in NLDR. We also describe a coarse-graining procedure to break the main logjam of learning in NLDR—the large number of pairwise interactions between nonneighboring inputs. The rest of the paper presents our LVM, evaluates its performance on datasets of images and text, and places it more fully in the context of previous work.

LVM

Our LVM is designed to learn a faithful embedding of the high-dimensional inputs \( \{x_i\}_{i=1}^N \). In a nutshell, the model succeeds by mapping these inputs stochastically into a latent space of lower dimensionality, then adjusting the parameters of this mapping so that, with high probability, only the images of nearby inputs remain nearby. Specifically, the model’s parameters are estimated by maximizing an objective function that consists of two terms: The first measures the likelihood that nearby inputs remain nearby, while the second measures the likelihood that distant inputs remain distant. Within the model, these likelihoods are computed by integrating over the (stochastically assigned) locations of inputs in the latent space of lower dimensionality. The rest of this section is devoted to filling out this description.

To begin, we associate, to each input, a low-dimensional output \( \mu_i \in \mathbb{R}^d \) and a variance \( \sigma_i^2 > 0 \); these outputs and variances are the parameters of the LVM that we will estimate to learn an embedding. Next, we focus on a particular pair of inputs \((x_i, x_j)\) and ask whether they are associated with outputs \((\mu_i, \mu_j)\) in a

Significance

Latent variable models (LVMs) are powerful tools for discovering hidden structure in data. Canonical LVMs include factor analysis, which explains the correlation of a large number of observed variables in terms of a smaller number of unobserved ones, and Gaussian mixture models, which reveal clusters of data arising from an underlying multimodal distribution. In this paper, we describe a conceptually simple and equally effective LVM for nonlinear dimensionality reduction (NLDR), where the goal is to discover faithful, neighborhood-preserving embeddings of high-dimensional data. Tools for NLDR can help researchers across all areas of science and engineering to better understand and visualize their data. Our approach elevates NLDR into the family of problems that can be studied by especially tractable LVMs.

Author contributions: L.K.S. designed research, performed research, contributed new reagents/analytic tools, analyzed data, and wrote the paper. The author declares no competing interest.

This article is a PNAS Direct Submission. N.K. is a guest editor invited by the Editorial Board.

Published under the PNAS license.

Data deposition: Code and scripts are available in Open Science Framework at https://osf.io/wy793/.

1Email: saul@cs.ucsd.edu.

This article contains supporting information online at https://www.pnas.org/lookup/suppl/doi:10.1073/pnas.1916012117/DCSupplemental.

neighborhood-preserving manner. To answer this question, we imagine an experiment in which latent (i.e., hidden) random variables \( h, h' \in \mathbb{R}^d \) are sampled from the multivariate Gaussian distributions

\[
P(h| x) = (2\pi \sigma_i^2)^{-d/2} \exp \left( -\frac{1}{2} ||h - \mu_i||^2 / \sigma_i^2 \right),
\]

\[
P(h'| x) = (2\pi \sigma_j^2)^{-d/2} \exp \left( -\frac{1}{2} ||h' - \mu_j||^2 / \sigma_j^2 \right).
\]

We note the following: If the embedding is faithful, and the variances \( \sigma_i \) and \( \sigma_j \) are not too large, then, with high probability, the distance \( ||h - h'|| \) should be commensurate with the distance \( ||x_i - x_j|| \).

We should not expect too much from such a mapping, however. For example, distances are unlikely to be preserved:

\[
\text{A neighborhood-preserving embedding may need considerable flexibility to situate the outputs in a space of much lower dimensionality. We can appeal, however, to some weaker properties of the desired mapping, which we again express in terms of the latent variables \( h \) and \( h' \). First, for any pair of inputs \( (x_i, x_j) \) that we regard as nearly on some length scale of interest, there should be some readily computed distance \( \delta_{ij} \), reflecting this length scale, such that \( ||h - h'|| < \delta_{ij} \) with high probability. Conversely, for any pair of inputs \( (x_i, x_j) \) that we regard as dissimilar on some length scale of interest, there should be some readily computed distance \( \Delta_{ij} \), reflecting this length scale, such that \( ||h - h'|| > \Delta_{ij} \) with high probability.}

For now, let us stipulate that we can formalize these judgments of nearness and specify the distances \( \delta_{ij}, \Delta_{ij} \) for any pair of inputs \( (x_i, x_j) \) as needed; we will fill in these details later.

These criteria suggest a natural way to measure the quality of an embedding, which can then be iteratively optimized. We start by considering how well the embedding behaves with respect to a particular pair of inputs \( (x_i, x_j) \). Let \( c_{ij} \in \{0, 1\} \) be Bernoulli random variables with conditional distributions

\[
P(c_{ij} = 1| h, h', x_i, x_j) = \exp \left( -\frac{1}{2} ||h - h'||^2 / \delta_{ij}^2 \right),
\]

\[
P(c_{ij} = 0| h, h', x_i, x_j) = \exp \left( -\frac{1}{2} ||h - h'||^2 / \Delta_{ij}^2 \right).
\]

The probabilities in Eqs. 3 and 4 measure the degrees to which the Gaussian latent variables \( h \) and \( h' \) coincide on the length scales \( \delta_{ij} \) and \( \Delta_{ij} \), respectively, in the lower-dimensional space of outputs. Integrating out these latent variables, we find that

\[
P(c_{ij} = 1| x_i, x_j) = \int_{h,h'} P(c_{ij} = 1| h, h', x_i, x_j) P(h|x_i) P(h'|x_j) = \left( \frac{\delta_{ij}^2}{\delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right)^{\frac{d}{2}} \exp \left\{ -1 - \frac{1}{2} \left( \frac{||\mu_i - \mu_j||^2}{\delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right) \right\},
\]

while an analogous equation holds for \( P(c_{ij} = 1| x_i, x_j) \) in terms of the distance \( \Delta_{ij} \). If the inputs \( (x_i, x_j) \) are nearby, then the value of \( \log P(c_{ij} = 1| x_i, x_j) \) measures how well the embedding preserves their nearness on the length scale of \( \delta_{ij} \); likewise, if the inputs \( (x_i, x_j) \) are dissimilar, then the value of \( \log P(c_{ij} = 0| x_i, x_j) \) measures how well the embedding preserves their dissimilarity on the length scale of \( \Delta_{ij} \).

We can measure the overall quality of the LVM’s embedding by accumulating these scores over multiple pairs of inputs. As we shall see, it will be convenient to invest some pairs of inputs with a larger role in this process than others. Allowing for a weighted sum of scores, we obtain an overall log (conditional) likelihood

\[
L = \sum_{ij} \left[ S_{ij} \log P(c_{ij} = 1| x_i, x_j) + D_{ij} \log P(c_{ij} = 0| x_i, x_j) \right]
\]

for the LVM; here \( S_{ij} \) and \( D_{ij} \) are nonnegative weights attached to pairs of inputs that we regard as similar at one length scale and/or dissimilar at another. We shall discuss later how to choose the distances \( \delta_{ij}, \Delta_{ij} \) and weights \( (S_{ij}, D_{ij}) \), which appear in Eq. 6 as constants, so that the remaining optimization over \( (\{\mu_i, \sigma_i\})_{i=1}^n \) yields a sensible embedding. For now, however, we shall assume that these constant values in Eq. 6 have been fixed, and focus on the more computationally intensive problem of maximum likelihood (ML) estimation. As a final aside, we note that LargeVis (2) and UMAP (3), although not formulated as LVMs, are based on objective functions of this same general form (involving sums over pairs of similar and dissimilar examples) but that differ in the weighting of individual terms and the way they assess similarity.

**EM Algorithm**

To proceed, we avail ourselves of the Expectation–Maximization (EM) algorithm (15) for ML estimation in LVMs. When applied to maximum effect, EM algorithms yield closed-form parameter updates that monotonically improve the likelihood in these models. EM algorithms have been widely used for ML estimation in Gaussian mixture models (14), factor analysis (13), and hidden Markov models (16), where the posterior statistics can be efficiently computed. Inference in these LVMs (as required by the E-step of the EM algorithm) is especially tractable because either the latent variables are discrete, with dependencies that are represented by a trivial or sparsely connected graph, or they are continuous and their posterior distributions are multivariate Gaussian.

The LVM in this paper is interesting in two respects. First, although its latent variables are continuous, with posterior distributions that are not always multivariate Gaussian, exact inference remains tractable. Second, unlike most LVMs for unsupervised learning, ours does not attempt to learn a so-called generative model of the data; instead, the LVM’s parameters \( \{\mu_i, \sigma_i\}_{i=1}^n \) are estimated to maximize the log conditional likelihood in Eq. 6, and it is these parameters that provide a low-dimensional representation of the data. SI Appendix gives a full derivation of the expectation step (E-step) and the maximization step (M-step) of the EM algorithm; here we present only the key results.

**Inference (E-step).** We can express the parameter updates for the EM algorithm most simply in terms of the posterior statistics of the LVM’s continuous latent variables. For each pair of inputs \( (x_i, x_j) \) appearing in Eq. 6, the relevant posterior distributions are given by Bayes rule,

\[
P(h, h'| c_{ij} = 1, x_i, x_j) = \frac{P(c_{ij} = 1| h, h', x_i, x_j) P(h|x_i) P(h'|x_j)}{P(c_{ij} = 1| x_i, x_j)},
\]

\[
P(h, h'| c_{ij} = 0, x_i, x_j) = \frac{P(c_{ij} = 0| h, h', x_i, x_j) P(h|x_i) P(h'|x_j)}{P(c_{ij} = 0| x_i, x_j)}.
\]

Substituting Eqs. 1–5 into Eq. 7, we see that the first of these distributions, \( P(h, h'| c_{ij} = 1, x_i, x_j) \), is multivariate Gaussian, although with different mean and covariance than the prior distribution \( P(h, h'| x_i, x_j) \). The second of these distributions, \( P(h, h'| c_{ij} = 0, x_i, x_j) \), is not multivariate Gaussian, but its statistics can still be efficiently computed. As shorthand, we define the odds ratio,

\[
\nu_{ij} = \frac{P(c_{ij} = 1| x_i, x_j)}{P(c_{ij} = 0| x_i, x_j)},
\]

for the Bernoulli distribution in Eq. 4; this ratio will simplify the expressions for many posterior statistics of interest.
The most important statistics to compute for the latent variables are their posterior means. For example, we have

\[
E[\mathbf{h}|c_5 = 1, \mathbf{x}_i, \mathbf{x}_j] = \mu_i - \left( \frac{\sigma_i^2}{\delta_i^2 + \sigma_i^2 + \sigma_j^2} \right) (\mu_i - \mu_j), \quad [10]
\]

\[
E[\mathbf{h}|c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j] = \mu_i + \left( \frac{\nu_{ij} \sigma_i^2}{\Delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right) (\mu_i - \mu_j). \quad [11]
\]

**SI Appendix** gives corresponding expressions for \(E[\mathbf{h}'|c_5 = 1, \mathbf{x}_i, \mathbf{x}_j]\) and \(E[\mathbf{h}'|c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j]\). Note that, in each of these expressions, the first term on the right-hand side is the prior mean, \(E[\mathbf{h}|c_5] = \mu_i\), while the remaining term gives a correction. The different signs of these corrections show, intuitively, that the posterior means for \(\mathbf{h}\) and \(\mathbf{h}'\) are pulled together when \(\mathbf{x}_i\) and \(\mathbf{x}_j\) are modeled as nearby inputs (with \(c_5 = 1\)) and pushed apart when they are not (with \(c_\Delta = 0\)).

The other posterior statistics we need for the EM algorithm are the expected squared Euclidean distances between the latent variables and their prior means. As shorthand, let

\[
\phi_{ij} = E \left[ \| \mathbf{h} - \mu_i \|^2 \bigg| c_5 = 1, \mathbf{x}_i, \mathbf{x}_j \right], \quad [12]
\]

\[
\phi'_{ij} = E \left[ \| \mathbf{h}' - \mu_i \|^2 \bigg| c_5 = 1, \mathbf{x}_i, \mathbf{x}_j \right], \quad [13]
\]

\[
\psi_{ij} = E \left[ \| \mathbf{h} - \mu_i \|^2 \bigg| c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j \right], \quad [14]
\]

\[
\psi'_{ij} = E \left[ \| \mathbf{h}' - \mu_i \|^2 \bigg| c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j \right]. \quad [15]
\]

These second-order statistics can also be efficiently computed. For example, we have

\[
\phi_{ij} = \sigma_i^2 \left[ d + \left( \frac{\sigma_j^2}{\delta_i^2 + \sigma_i^2 + \sigma_j^2} \right) \left( \frac{\| \mu_i - \mu_j \|^2}{\delta_i^2 + \sigma_i^2 + \sigma_j^2} - d \right) \right]. \quad [16]
\]

\[
\psi_{ij} = \sigma_i^2 \left[ d - \left( \frac{\nu_{ij} \sigma_i^2}{\Delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right) \left( \frac{\| \mu_i - \mu_j \|^2}{\Delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} - d \right) \right]. \quad [17]
\]

**SI Appendix** gives the corresponding expressions for \(\phi'_{ij}\) and \(\psi'_{ij}\).

**Learning (M-step).** The EM algorithm for LVMs is based on maximizing a lower bound on the log-likelihood (15). There are two ways to derive EM updates for our LVM. The first (standard) approach employs a single lower bound on Eq. 6 to derive joint updates for the model parameters \(\{\mu_i, \sigma_i^2\}_{i=1}^n\). The second approach employs separate (and tighter) lower bounds to derive alternating updates for these parameters. Here we present the results of this second approach, which converged significantly faster in practice.

We begin by giving the update for the outputs \(\{\mu_i\}_{i=1}^n\). From the similarity weights \(S_{ij}\), we define the matrix

\[
W_{ij} = \frac{S_{ij}}{\delta_i^2 + \sigma_i^2 + \sigma_j^2} + \frac{S_{ji}}{\delta_j^2 + \sigma_i^2 + \sigma_j^2}, \quad [18]
\]

which is generally sparse and symmetric. Next, we form the symmetric diagonally dominant (SDD) matrix

\[
L_{ij} = \begin{cases} 
\sum_k W_{ik} + \frac{1}{\sigma_i^2} \sum_k (D_{ik} + D_{ki}) & \text{if } i = j, \\
-W_{ij} & \text{otherwise}.
\end{cases} \quad [19]
\]

The matrix \(L\) consists of the sum of two terms: The first is the weighted graph Laplacian that incorporates the similarity weights \(S_{ij}\) through Eq. 18, while the second is a nonnegative diagonal matrix that incorporates the dissimilarity weights \(D_{ij}\). We reestimate the outputs by solving the linear system

\[
(L\mu_{\text{new}})^{ij} = \frac{1}{\sigma_i^2} \sum_j (D_{ij} E[\mathbf{h}|c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j] + D_{ji} E[\mathbf{h}'|c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j]). \quad [20]
\]

To implement this update, we can avail ourselves of extremely efficient solvers for SDD linear systems (17). In practice, the update exploits some second-order structure of the objective function while avoiding the complexity of a full Newton–Raphson update. It also exemplifies the so-called Laplacian paradigm (18), in which fast solvers for SDD systems are used as algorithmic primitives in problems involving large graphs.

Next, we give the update for the variances \(\{\sigma_i^2\}_{i=1}^n\). After computing the expected squared distances in Eqs. 12–15, this update takes an especially simple form:

\[
(\sigma_i^2)_{\text{new}} = \frac{1}{d} \sum_j \left( S_{ij} \phi_{ij} + S_{ji} \phi'_{ij} + D_{ij} \psi_{ij} + D_{ji} \psi'_{ij} \right). \quad [21]
\]

An appealing property of EM (15) is that both this update and the one in Eq. 20 are guaranteed to increase the log conditional likelihood in Eq. 6 (except at stationary points).

**Length Scales and Pairwise Similarities**

The goal of our LVM is to learn an embedding that maps nearby inputs to nearby outputs and distant inputs to distant outputs. To begin, however, we must formalize which pairs of inputs we regard as nearby in the first place. As is common, we encode these neighborhood relationships by a sparse graph. Then we use this graph to derive the length scales \((\delta_{ij}, \Delta_{ij})\) in Eqs. 3 and 4 and the weights \((S_{ij}, D_{ij})\) in Eq. 6.

**Neighborhood Graph.** It would be simplest to regard the inputs \(\mathbf{x}_i\) and \(\mathbf{x}_j\) as nearby when either is among the \(k\) nearest neighbors (\(k\NN\)) of the other, as judged by Euclidean distance or some other metric. Here, \(k\) is a single user-specified parameter that controls the size of desired neighborhoods. This prescription generates a sparse symmetric graph, with edges connecting neighboring inputs, but, in practice, it often connects inputs that should not be regarded as nearby. Problems arise especially when the data contains outliers, when the inputs are not densely sampled, or when the underlying density is variable (19). Many failure modes of manifold learning can be traced to graphs that were formed in this way.

For our LVMs, we have used a simple two-parameter procedure to construct sparse neighborhood graphs, where both parameters \((k, s)\) are small positive integers. The parameter \(k\), which largely determines the neighborhood size, is analogous to the main tuning parameter of other algorithms for data visualization, such as t-SNE (1), LargeVis (2), and UMAP (3). The parameter \(s\) is mainly useful to optimize embeddings for purposes other than visualization (e.g., when \(d > 3\)): for visualizations, however, it can be set to a default value of \(s = 1\).

Our procedure starts by computing the \(k\NN\) of each input and encoding these neighborhood relationships by a binary matrix \(\mathbf{K}\), where

\[
K_{ij} = \begin{cases} 
1 & \text{if } \mathbf{x}_i \text{ is a } k\NN \text{ of } \mathbf{x}_i, \\
0 & \text{otherwise (and on the diagonal).}
\end{cases} \quad [22]
\]

Note that this matrix is not, in general, symmetric. In practice, we find that, even for small values of \(k\), this graph may be too permissive as a representation of pairs of inputs that should be mapped to nearby outputs. Therefore, the rest of our procedure is designed to extract a subset of edges in this graph that represent more robust neighborhood relationships (19). We do this in
three steps: first, by identifying pairs of inputs that yield a minimally connected subgraph; second, by identifying pairs of inputs that unambiguously deserve to be regarded as nearby; and third, by taking the union of these findings. We now describe each of these steps in more detail.

In the first step, we compute the minimum spanning tree of the undirected graph with adjacency matrix $K_{ij} + K_{ji} - K_{ij}K_{ji}$ (i.e., whose edges connect $x_i$ to $x_j$ if either input is a $k$NN of the other). For this computation, we weight the graph’s edges by the Euclidean distances $||x_i - x_j||$; if the graph is not connected, we consider only its largest connected component, which we assume (for a suitable value of $k$) to contain all but a few outliers. We use $T$ to denote the adjacency matrix of this minimum weighted spanning tree. The matrix $T$ encodes a minimal set of edges that preserve the large-scale connectivity of the inputs.

In the second step, we accumulate the first $s$ powers of the matrix $T$; the value of $s$ (typically, small) can be regarded as the number of steps of a directed random walk. Let

$$R = K + K^2 + \cdots + K^s.$$  \[23\]

Clearly, the larger the value of $s$, the further the reach of this $s$-step random walk. Of particular interest to us are the pairs of inputs $(x_i, x_j)$ that are mutually reachable from each other by this random walk: These are the pairs for which $R_{ij}R_{ji} > 0$, and they tend not to include outliers. We have observed that mutual reachability in this directed random walk reveals more meaningful neighborhoods than simple $k$NN.

The final step of our procedure combines the results of the previous two. We define a neighborhood graph with edges

$$E_{ij} = \begin{cases} 1 & \text{if } K_{ij}(T^s_{ij} + R_{ij}R_{ji}) > 0, \\ 0 & \text{otherwise.} \end{cases}$$  \[24\]

By construction, this neighborhood graph connects only a subset of those pairs of inputs that are $k$NN; in particular, it retains those pairs needed to keep the graph connected, as well as those whose mutuality suggests an extra degree of robustness. Note that these neighborhood relations are not, in general, symmetric, so that we may sometimes regard $x_i$ as nearby to $x_j$ (when $E_{ij} = 1$) but not vice versa (when $E_{ji} = 0$). This happens, most notably, when $x_i$ is an outlier.

**Coarse Graining.** Finally, we use the neighborhood graph $E_{ij}$ to derive the length scales $(\delta_{ij}, \Delta_{ij})$ in Eqs. 3 and 4 and the weights $(S_{ij}, D_{ij})$ in Eq. 6. The edges that are present in this graph indicate the pairs of inputs that should be mapped to nearby outputs, while the edges that are absent indicate the pairs that should not. When the neighborhood graph is sparse, the number of pairs in the latter category will be prohibitively large to enumerate for all but the smallest datasets. Algorithms such as LargeVis (2) and UMAP (3) overcome this difficulty by a negative sampling procedure (7) for nonadjacent nodes in the graph. As an alternative, we describe a recursive, coarse-graining procedure where the number of levels ($\ell$) of recursion can be adjusted to learn from larger numbers of inputs.

**Case $\ell = 0$.** We choose the weights and length scales to achieve two complementary goals—first, to shrink the distances between neighbors, and second, to ensure that neighbors remain closer than nonneighbors. In the base case, this is especially simple: We set the weights by $S_{ij} = E_{ij}$ and $D_{ij} = 1 - E_{ij}$ (but only keeping nonzero elements off the diagonal), and we set the length scales by $\delta_{ij} = ||x_i - x_j||^2/(2\log 2)$ and $\Delta_{ij}^2 = \max_k \{E_{ik}||x_i - x_k||^2\}/(2\log 2)$. The constant factors of $2\log 2$ in these denominators serve to calibrate the probabilities in Eqs. 3 and 4 so that $P(c_i = 1 | h, \mathbf{x}_i, x_j) = 1/2$ when $||h - \mathbf{x}_j|| = \delta_{ij}$.

Before proceeding with EM, we also rescale the weights such that $\sum_j S_{ij} = \sum_j D_{ij}$.

**Case $\ell > 1$.** Fig. 1 shows the basic intuition behind coarse graining. Suppose that $n$ is too large to model $O(n^2)$ pairs of dissimilar inputs. Instead, we randomly designate $n_u < n$ of the inputs as landmarks and assign each input to its closest landmark; similar ideas (20–23) have been used in many algorithms for NLDR. Let $\Omega_i$ denote the set of inputs assigned to the landmark $x_i$ (including itself), and let $|\Omega_i|$ denote the size of this set. Coarse graining does not change the dissimilarity weights between inputs assigned to the same landmark. But now we make three changes: 1) We add unit similarity weights between landmarks and the

![Fig. 1. Effect of the coarse-graining procedure, with red edges connecting inputs (e.g., images of handwritten digits) that are explicitly modeled as dissimilar. (Left) Before coarse graining, many pairs of such inputs contribute a large number of terms to the model’s log conditional likelihood. (Right) After coarse graining, only the inputs designated as landmarks are explicitly modeled as dissimilar, but their contributions to the log conditional likelihood are weighted by the number of inputs they represent; in addition, inputs represented by the same landmark—are those inside the dashed squares—are modeled as weakly similar to the landmark (indicated by blue edges in the new neighborhood graph). During learning, the landmarks are widely separated from one another but kept relatively closer to the points they represent. Thus, by the triangle inequality, the coarse-grained model achieves an effect similar to the original one.](image-url)

![Fig. 2. The 2D visualization of 70,000 handwritten digits ($k = 9$, $s = 1$, $\ell = 1$). Different clusters of digits are easily identified; in addition, the relative distances between clusters reveal more and less confusable classes of digits.](image-url)
other inputs they represent; 2) we zero the dissimilarity weights between inputs represented by different landmarks; and 3) we add dissimilarity weights between different landmarks (say, \( x_i \) and \( x_j \)), setting

\[
D_{ij} = |\Omega_i| \cdot |\Omega_j|,
\]

\[
\Delta_{ij} = \max_{\alpha \in \Omega_i, \beta \in \Omega_j} (\Delta_{\alpha\beta}^{\text{prev}} + \delta_{\alpha\beta} + \delta_{ij}),
\]

where \( \Delta_{\alpha\beta}^{\text{prev}} \) denotes the desired length scale of separation between \( x_{\alpha} \) and \( x_{\beta} \) before this level of coarse graining. Note that, if the landmarks are widely separated while the inputs they represent remain relatively clustered, then, by the triangle inequality, the inputs represented by different landmarks will also be well separated. The length scales in Eq. 26 were chosen with this in mind—so that ML estimation after coarse graining will separate inputs belonging to different landmarks by at least the amount as before coarse graining. As in the base case, we also rescale the weights before proceeding with EM; SI Appendix provides further details.

It remains to specify how many landmarks to choose. In practice, we choose \( n_l \propto n^{2/3} \); this scaling is motivated by the idealized scenario in which, as a result of coarse graining, the \( n \) inputs of the dataset are evenly assigned to the \( n_l \) landmarks. In this case, the \( O(n^2) \) pairs of dissimilar inputs before coarse graining are replaced by \( O(n_l^2) \) pairs of dissimilar landmarks and \( O(n(n_l)^2) \) pairs of dissimilar inputs belonging to the same landmarks. The overall number of such pairs is minimized by setting \( n_l \propto n^{2/3} \), reducing the number of nonzero dissimilarity weights from \( O(n^2) \) to \( O(n^{8/3}) \).

**Case \( \ell \geq 2 \).** For large \( n \), it may remain prohibitive to explicitly model the dissimilarity between pairs of landmarks and/or between pairs of inputs assigned to the same landmark. In this case, the coarse-graining procedure can be applied recursively. For example, if there are \( n_l \) landmarks after one level of coarse graining, we can sample \( n_l^{2/3} \) of them as superlandmarks and model their separation at even longer length scales; or, if there are \( |\Omega_i| \) inputs assigned to the landmark \( x_i \), we can sample \( |\Omega_i|^{2/3} \) of them as sublandmarks and model their separation at a shorter length scale. In the idealized case of perfectly balanced assignments, a similar scaling argument (in SI Appendix) shows that the next level of coarse graining reduces the number of nonzero dissimilarity weights from \( O(n^{8/3}) \) to \( O(n^{10/9}) \).

**Results**

We have implemented the EM algorithm for this LVM in MATLAB. All results were obtained on an Intel Core i5 CPU running at 1.3 GHz. In each experiment, we ran the EM algorithm for 400 iterations and used a momentum term to accelerate the update in Eq. 20. We also initialized the outputs \( \{\mu_k\}_{k=1}^n \) from the bottom eigenvectors of a graph Laplacian matrix derived from the similarity weights \( S_{ij} \). Many further details and experiments are reported in SI Appendix.

Fig. 2 shows a two-dimensional (2D) visualization of the MNIST test set of 10,000 digits. The results show that most of the interclass variation is captured in the first few dimensions learned by the LVM; by contrast, principal component analysis (PCA) requires many more dimensions to obtain comparable error rates. The figure also illustrates the effect of one versus two levels of coarse graining. The error rates are somewhat higher when \( \ell = 2 \), presumably because the extra level of coarse graining incurs more slack from the triangle inequality.

As a point of comparison, the 400 iterations of EM required 859 and 579 s, respectively, for the \( d = 10 \) embeddings with \( \ell = 1 \) and \( \ell = 2 \) levels of coarse graining.

We also experimented on a subset of normalized word vectors from the word2vec model trained on the GoogleNews corpus (7). The original model produced a vector of dimensionality 300 for each word in a 3-million-word vocabulary; the subset, containing \( n = 207,147 \) word vectors, was obtained by filtering out words (such as misspellings and garbage phrases) not listed in WordNet (25). Fig. 3 compares the test error rates of 9NN classifiers on handwritten digits after dimensionality reduction by PCA and LVM. The overall number of such pairs is minimized by setting \( n_l \propto n^{2/3} \), reducing the number of nonzero dissimilarity weights from \( O(n^2) \) to \( O(n^{8/3}) \).

**Fig. 3.** Test error rates of 9NN classifiers on handwritten digits after dimensionality reduction by PCA versus latent variable modeling \((k = 9, s = 2)\). The dotted line shows the test error rate when the digits are projected onto their top 50 principal components; these projections were the inputs to the LVM.

**Fig. 4.** Recall of first nearest neighbor at \( r \), for the dataset of \( n = 207,147 \) normalized word vectors, after linear dimensionality reduction by PCA and NLDL by latent variable modeling \((k = 4, s = 3, \ell = 2)\).
times as many dimensions as NLDR to obtain the same degree of recall. Table 1 shows the four nearest neighbors of word vectors before and after the embedding by our LVM \((k = 4, s = 3, \ell = 2)\) into 10 dimensions. The computation time for this embedding was dominated by the initial search for exact nearest neighbors, which took over 2 h; the rest of the computation took roughly 30 min.

**Perspective**

Our model builds directly on earlier approaches. LargeVis (2) and UMAP (3) optimize a likelihood of the same general form as Eq. 6, but with a different specific form for the probability in Eq. 5 that pairs of outputs are colocated. An EM algorithm similar to the one in this paper was also investigated for an earlier model of distance metric learning (26).

Our work was heavily inspired by the success of previous visualizations, as well as a desire to replicate them with longstanding methodologies. Visualizations from t-SNE (1, 4), LargeVis, and UMAP have emerged as core tools in exploratory data analysis; in addition, current implementations of LargeVis and UMAP scale very well to large datasets, as do leading implementations of t-SNE (5). While the main bottleneck of our approach is currently the search for exact nearest neighbors, this is not an intrinsic bottleneck: Like these other methods, we could also turn to approximate nearest neighbor search (6, 27).

We emphasize, however, that the general problem of NLDR has larger goals than visualization (which focuses only on embeddings in three or fewer dimensions). This general problem is where more work remains to be done, and also where our formulation is likely to yield the most benefits. LVMs can be extended in many directions—most notably, by incorporating additional latent variables—and tractable LVMs can serve as a basis for approximate inference in less tractable ones (28). Notwithstanding the power of current approaches, we anticipate many further developments along these lines.

**Data Availability.** Code and scripts are available at https://osf.io/wy793/. The dataset of MNIST handwritten digits is available at http://yann.lecun.com/exdb/mnist/, and the Matlab word2vec dataset is available at https://github.com/chrisjmccormick/word2vec_matlab.

**ACKNOWLEDGMENTS.** The Matlab experiments were made possible by the combinatorial multigrid solver distributed by J. Koutis and the abridged word2vec dataset distributed by C. McCormick on Github. A similar debt is owed to the anonymous reviewers, whose suggestions improved the paper in numerous ways.

---

**Table 1.** Four nearest neighbors of word vectors before and after the LVM's embedding \((k = 4, s = 3, \ell = 2)\) into 10 dimensions

<table>
<thead>
<tr>
<th>Word</th>
<th>4NN (before)</th>
<th>4NN (after)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tractable</td>
<td>willer, tractability, intractable</td>
<td>tractability, Rube, Goldberg, effervescence, ingenious</td>
</tr>
<tr>
<td>Latent Variable</td>
<td>unexpressed, unarticulated, untapped</td>
<td>unexpressed, unvoiced, unarticulated, inchoate</td>
</tr>
<tr>
<td>Model</td>
<td>Variable, adjustable, quantize, variability</td>
<td>Variable, Fixed, fixed, assignable</td>
</tr>
<tr>
<td>Dimensionality Reduction</td>
<td>linear, oscillatory, diffuse, fractal</td>
<td>linear, linear, gravitational_attraction, gravitation</td>
</tr>
<tr>
<td></td>
<td>dimensional, translucency, tonal, spatiality</td>
<td>dimensional, Dimensional, Dimension, translucency</td>
</tr>
<tr>
<td></td>
<td>reductions, decrease, increase, increases</td>
<td>increases, decreases, increase, reductions</td>
</tr>
</tbody>
</table>

---