Published version:
https://doi.org/10.1073/pnas.1916012117

A tractable latent variable model for nonlinear dimensionality reduction

Lawrence K. Saul[a,1]

[a]Department of Computer Science and Engineering, UC San Diego, La Jolla, CA 92093-0404, USA

This manuscript was compiled on XXXXXX

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-SNE (van der Maaten & Hinton, 2008), the model can produce two and three dimensional embeddings for visualization, but it can also learn higher dimensional embeddings for other uses. Like LargeVis (Tang et al., 2016) and UMAP (McInnes & Healy, 2018), the model's likelihood is optimized 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 non-adjacent 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-SNE (1), LargeVis (2), and 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 $\{\mathbf{x}_i\}_{i=1}^N$, infer a corresponding set of low dimensional (unobserved) outputs $\{\boldsymbol{\mu}_i\}_{i=1}^N$ 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 novel coarse-graining procedure to break the main logjam of learning in NLDR—the large number of pairwise interactions between non-neighboring 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.

Latent variable model

Our LVM is designed to learn a faithful embedding of the high dimensional inputs $\{\mathbf{x}_i\}_{i=1}^N$. 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 $\boldsymbol{\mu}_i \in \mathbb{R}^d$ 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 $(\mathbf{x}_i, \mathbf{x}_j)$ and ask whether they are associated to outputs $(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)$ in a neighborhood-preserving manner. To answer this question, we imagine an experiment in which latent (i.e.,

Significance Statement

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.

[1]To whom correspondence should be addressed. E-mail: saul@cs.ucsd.edu

hidden) random variables $\mathbf{h}, \mathbf{h}' \in \mathbb{R}^d$ are sampled from the multivariate Gaussian distributions

$$P(\mathbf{h}|\mathbf{x}_i) = (2\pi\sigma_i^2)^{-d/2} \exp\left(-\tfrac{1}{2}\|\mathbf{h}-\boldsymbol{\mu}_i\|^2/\sigma_i^2\right), \quad [1]$$

$$P(\mathbf{h}'|\mathbf{x}_j) = (2\pi\sigma_j^2)^{-d/2} \exp\left(-\tfrac{1}{2}\|\mathbf{h}'-\boldsymbol{\mu}_j\|^2/\sigma_j^2\right). \quad [2]$$

We note the following: if the embedding is faithful, and the variances σ_i and σ_j not too large, then with high probability the distance $\|\mathbf{h}-\mathbf{h}'\|$ should be commensurate with the distance $\|\mathbf{x}_i-\mathbf{x}_j\|$.

We should not expect too much from such a mapping, however. For example, distances are unlikely to be preserved: a neighborhood-preserving embedding may need considerable flexibility to situate the outputs in a space of much lower dimensionality. We can appeal, though, to some weaker properties of the desired mapping, which we again express in terms of the latent variables $\mathbf{h}$ and $\mathbf{h}'$. First, for any pair of inputs $(\mathbf{x}_i, \mathbf{x}_j)$ that we regard as nearby on some length scale of interest, there should be some readily computed distance δ_{ij}, reflecting this length scale, such that $\|\mathbf{h}-\mathbf{h}'\| < \delta_{ij}$ with high probability. Conversely, for any pair of inputs $(\mathbf{x}_i, \mathbf{x}_j)$ that we regard as dissimilar on some length scale of interest, there should be some readily computed distance Δ_{ij}, reflecting this length scale, such that $\|\mathbf{h}-\mathbf{h}'\| > \Delta_{ij}$ 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 $(\mathbf{x}_i, \mathbf{x}_j)$ 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 $(\mathbf{x}_i, \mathbf{x}_j)$. Let $c_\delta, c_\Delta \in \{0, 1\}$ be Bernoulli random variables with conditional distributions

$$P(c_\delta = 1|\mathbf{h}, \mathbf{h}', \mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\tfrac{1}{2}\|\mathbf{h}-\mathbf{h}'\|^2/\delta_{ij}^2\right), \quad [3]$$

$$P(c_\Delta = 1|\mathbf{h}, \mathbf{h}', \mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\tfrac{1}{2}\|\mathbf{h}-\mathbf{h}'\|^2/\Delta_{ij}^2\right). \quad [4]$$

The probabilities in eqs. (3–4) measure the degrees to which the Gaussian latent variables $\mathbf{h}$ and $\mathbf{h}'$ *coincide* on the length scales δ_{ij} and Δ_{ij}, respectively, in the lower dimensional space of outputs. Integrating out these latent variables, we find that

$$\begin{aligned} &P(c_\delta = 1|\mathbf{x}_i, \mathbf{x}_j) \\ &= \int_{\mathbf{h},\mathbf{h}'} P(c_\delta = 1|\mathbf{h}, \mathbf{h}', \mathbf{x}_i, \mathbf{x}_j)\, P(\mathbf{h}|\mathbf{x}_i)\, P(\mathbf{h}'|\mathbf{x}_j) \\ &= \left(\frac{\delta_{ij}^2}{\delta_{ij}^2 + \sigma_i^2 + \sigma_j^2}\right)^{\frac{d}{2}} \exp\left\{-\frac{1}{2}\left(\frac{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2}{\delta_{ij}^2 + \sigma_i^2 + \sigma_j^2}\right)\right\}, \quad [5] \end{aligned}$$

while an analogous equation holds for $P(c_\Delta = 1|\mathbf{x}_i, \mathbf{x}_j)$ in terms of the distance Δ_{ij}. If the inputs $(\mathbf{x}_i, \mathbf{x}_j)$ are nearby, then the value of $\log P(c_\delta = 1|\mathbf{x}_i, \mathbf{x}_j)$ measures how well the embedding preserves their nearness on the length scale of δ_{ij}; likewise, if the inputs $(\mathbf{x}_i, \mathbf{x}_j)$ are dissimilar, then the value of $\log P(c_\Delta = 0|\mathbf{x}_i, \mathbf{x}_j)$ measures how well the embedding preserves their dissimilarity on the length scale of Δ_{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

$$\mathcal{L} = \sum_{ij}\Big[\mathcal{S}_{ij}\log P(c_\delta = 1|\mathbf{x}_i, \mathbf{x}_j) + \mathcal{D}_{ij}\log P(c_\Delta = 0|\mathbf{x}_i, \mathbf{x}_j)\Big] \quad [6]$$

for the LVM; here $\mathcal{S}_{ij}$ and $\mathcal{D}_{ij}$ 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 $(\mathcal{S}_{ij}, \mathcal{D}_{ij})$, which appear in eq. (6) as constants, so that the remaining optimization over $\{(\boldsymbol{\mu}_i, \sigma_i)\}_{i=1}^n$ 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), though 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, though 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 $\{(\boldsymbol{\mu}_i, \sigma_i)\}_{i=1}^n$ 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. The Supplementary Material (SM) gives a full derivation of the steps 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 $(\mathbf{x}_i, \mathbf{x}_j)$ appearing in eq. (6), the relevant posterior distributions are given by Bayes rule:

$$P(\mathbf{h}, \mathbf{h}'|c_\delta = 1, \mathbf{x}_i, \mathbf{x}_j) = \frac{P(c_\delta = 1|\mathbf{h}, \mathbf{h}', \mathbf{x}_i, \mathbf{x}_j) P(\mathbf{h}|\mathbf{x}_i) P(\mathbf{h}'|\mathbf{x}_j)}{P(c_\delta = 1|\mathbf{x}_i, \mathbf{x}_j)}, \quad [7]$$

$$P(\mathbf{h}, \mathbf{h}'|c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j) = \frac{P(c_\Delta = 0|\mathbf{h}, \mathbf{h}', \mathbf{x}_i, \mathbf{x}_j) P(\mathbf{h}|\mathbf{x}_i) P(\mathbf{h}'|\mathbf{x}_j)}{P(c_\Delta = 0|\mathbf{x}_i, \mathbf{x}_j)}. \quad [8]$$

Substituting eqs. (1–5) into eq. (7), we see that the first of these distributions, $P(\mathbf{h}, \mathbf{h}'|c_\delta = 1, \mathbf{x}_i, \mathbf{x}_j)$, is multivariate Gaussian, though with different mean and covariance than the prior distribution $P(\mathbf{h}, \mathbf{h}'|\mathbf{x}_i, \mathbf{x}_j)$. The second of these distributions, $P(\mathbf{h}, \mathbf{h}'|c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j)$, is not multivariate Gaussian, but its statistics can still be efficiently computed. As shorthand, we define the odds ratio

$$\nu_{ij} = P(c_\Delta = 1|\mathbf{x}_i, \mathbf{x}_j) \big/ P(c_\Delta = 0|\mathbf{x}_i, \mathbf{x}_j), \quad [9]$$

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

$$\mathrm{E}[\mathbf{h}|c_\delta=1,\mathbf{x}_i,\mathbf{x}_j] \;=\; \boldsymbol{\mu}_i - \left(\frac{\sigma_i^2}{\delta_{ij}^2+\sigma_i^2+\sigma_j^2}\right)(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j), \quad [10]$$

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

The SM gives corresponding expressions for $\mathrm{E}[\mathbf{h}'|c_\delta=1,\mathbf{x}_i,\mathbf{x}_j]$ and $\mathrm{E}[\mathbf{h}'|c_\Delta=0,\mathbf{x}_i,\mathbf{x}_j]$. Note that in each of these expressions, the first term on the right hand side is the prior mean, $\mathrm{E}[\mathbf{h}|\mathbf{x}_i]=\boldsymbol{\mu}_i$, 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 $c_\delta=1$) 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

$$\phi_{ij} \;=\; \mathrm{E}\big[\|\mathbf{h}-\boldsymbol{\mu}_i\|^2 \,|\, c_\delta=1,\mathbf{x}_i,\mathbf{x}_j\big], \quad [12]$$

$$\phi'_{ij} \;=\; \mathrm{E}\big[\|\mathbf{h}'-\boldsymbol{\mu}_j\|^2 \,|\, c_\delta=1,\mathbf{x}_i,\mathbf{x}_j\big], \quad [13]$$

$$\psi_{ij} \;=\; \mathrm{E}\big[\|\mathbf{h}-\boldsymbol{\mu}_i\|^2 \,|\, c_\Delta=0,\mathbf{x}_i,\mathbf{x}_j\big], \quad [14]$$

$$\psi'_{ij} \;=\; \mathrm{E}\big[\|\mathbf{h}'-\boldsymbol{\mu}_j\|^2 \,|\, c_\Delta=0,\mathbf{x}_i,\mathbf{x}_j\big]. \quad [15]$$

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

$$\phi_{ij} \;=\; \sigma_i^2\left[d+\left(\frac{\sigma_i^2}{\delta_{ij}^2+\sigma_i^2+\sigma_j^2}\right)\left(\frac{\|\boldsymbol{\mu}_i-\boldsymbol{\mu}_j\|^2}{\delta_{ij}^2+\sigma_i^2+\sigma_j^2}-d\right)\right], \quad [16]$$

$$\psi_{ij} \;=\; \sigma_i^2\left[d-\left(\frac{\nu_{ij}\sigma_i^2}{\Delta_{ij}^2+\sigma_i^2+\sigma_j^2}\right)\left(\frac{\|\boldsymbol{\mu}_i-\boldsymbol{\mu}_j\|^2}{\Delta_{ij}^2+\sigma_i^2+\sigma_j^2}-d\right)\right]. \quad [17]$$

The SM gives the corresponding expressions for ϕ'_{ij} and ψ'_{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 $\{\boldsymbol{\mu}_i,\sigma_i^2\}_{i=1}^n$. 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 $\{\boldsymbol{\mu}_i\}_{i=1}^n$. From the similarity weights $\mathcal{S}_{ij}$, we define the matrix

$$W_{ij} \;=\; \frac{\mathcal{S}_{ij}}{\delta_{ij}^2+\sigma_i^2+\sigma_j^2} \;+\; \frac{\mathcal{S}_{ji}}{\delta_{ji}^2+\sigma_i^2+\sigma_j^2}, \quad [18]$$

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

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

The matrix $\mathbf{L}$ consists of the sum of two terms: the first is the weighted graph Laplacian that incorporates the similarity weights $\mathcal{S}_{ij}$ through eq. (18), while the second is a nonnegative diagonal matrix that incorporates the dissimilarity weights $\mathcal{D}_{ij}$. We re-estimate the outputs by solving the linear system:

$$(\mathbf{L}\boldsymbol{\mu}^{\mathrm{new}})_i = \frac{\sum_j\big(\mathcal{D}_{ij}\mathrm{E}[\mathbf{h}|c_\Delta=0,\mathbf{x}_i,\mathbf{x}_j]+\mathcal{D}_{ji}\mathrm{E}[\mathbf{h}'|c_\Delta=0,\mathbf{x}_j,\mathbf{x}_i]\big)}{\sigma_i^2}. \quad [20]$$

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:

$$(\sigma_i^2)^{\mathrm{new}} \;=\; \frac{1}{d}\,\frac{\sum_j\big(\mathcal{S}_{ij}\phi_{ij}+\mathcal{S}_{ji}\phi'_{ji}+\mathcal{D}_{ij}\psi_{ij}+\mathcal{D}_{ji}\psi'_{ji}\big)}{\sum_j(\mathcal{S}_{ij}+\mathcal{S}_{ji}+\mathcal{D}_{ij}+\mathcal{D}_{ji})}. \quad [21]$$

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, though, 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. (3–4) and the weights $(\mathcal{S}_{ij},\mathcal{D}_{ij})$ 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 (kNN) 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, though, it can be set to a default value of $s=1$.

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

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

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 $\mathbf{x}_i$ to $\mathbf{x}_j$ if *either* input is a kNN of the other). For this computation, we weight the graph's edges by the Euclidean distances $\|\mathbf{x}_i - \mathbf{x}_j\|$; 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 $\mathbf{T}$ to denote the adjacency matrix of this minimum weighted spanning tree. The matrix $\mathbf{T}$ 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 $\mathbf{K}$; the value of s (typically, small) can be regarded as the number of steps of a directed random walk. Let

$$\mathbf{R} \;=\; \mathbf{K} + \mathbf{K}^2 + \cdots + \mathbf{K}^s. \qquad [23]$$

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 $(\mathbf{x}_i, \mathbf{x}_j)$ 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-nearest neighbors.

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

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

By construction, this neighborhood graph connects only a subset of those pairs of inputs that are k nearest neighbors; 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 $\mathbf{x}_j$ as nearby to $\mathbf{x}_i$ (when $E_{ij} = 1$) but not vice versa (when $E_{ji} = 0$). This happens, most notably, when $\mathbf{x}_i$ is an outlier.

Coarse-graining. Finally we use the neighborhood graph E_{ij} to derive the length scales $(\delta_{ij}, \Delta_{ij})$ in eqs. (3–4) and the weights $(\mathcal{S}_{ij}, \mathcal{D}_{ij})$ 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 non-adjacent nodes in the graph. As an alternative, we describe a recursive, coarse-graining procedure where the number of levels (ℓ) 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 non-neighbors. In the base case, this is especially simple: we set the weights by $\mathcal{S}_{ij} = E_{ij}$ and $\mathcal{D}_{ij} = 1 - E_{ij}$ (but only keeping nonzero elements off the diagonal), and

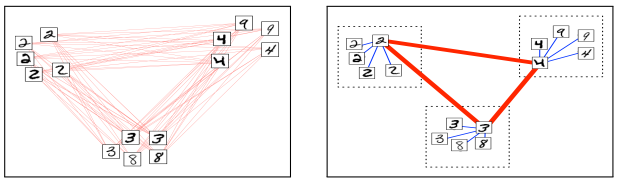

Fig. 1. Effect of the coarse-graining procedure, with red edges connecting inputs (e.g., images of handwritten digits) that are explicitly modeled as dissimilar. **Left:** before coarse-graining, many pairs of such inputs contribute a large number of terms to the model's log conditional likelihood. **Right:** 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—those inside the dashed squares—are modeled as *weakly* similar to the landmark (indicated by blue edges). During learning, the landmarks are widely separated from one other but kept relatively closer to the points they represent. Thus, by the triangle inequality, the coarse-grained model achieves a similar effect as the original one.

we set the length scales by $\delta_{ij}^2 = \|\mathbf{x}_i - \mathbf{x}_j\|^2 / (2 \log 2)$ and $\Delta_{ij}^2 = \max_k \left\{ E_{ik} \|\mathbf{x}_i - \mathbf{x}_k\|^2 \right\} / (2 \log 2)$. The constant factors of $2 \log 2$ in these denominators serve to calibrate the probabilities in eq. (3–4) so that $P(c_\delta = 1 | \mathbf{h}, \mathbf{h}', \mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2}$ when $\|\mathbf{h} - \mathbf{h}'\| = \delta_{ij}$. Before proceeding with EM, we also rescale the weights such that $\sum_{ij} \mathcal{S}_{ij} = \sum_{ij} \mathcal{D}_{ij}$.

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 $n_\ell < n$ 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 Ω_i denote the set of inputs assigned to the landmark $\mathbf{x}_i$ (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: (i) we add unit similarity weights between landmarks and the other inputs they represent; (ii) we zero the dissimilarity weights between inputs represented by different landmarks; and (iii) we add dissimilarity weights between different landmarks (say, $\mathbf{x}_i$ and $\mathbf{x}_j$), setting

$$\mathcal{D}_{ij} \;=\; |\Omega_i| \cdot |\Omega_j|, \qquad [25]$$

$$\Delta_{ij} \;=\; \max_{\alpha \in \Omega_i, \beta \in \Omega_j} \left(\Delta_{\alpha\beta}^{\text{prev}} + \delta_{i\alpha} + \delta_{j\beta} \right), \qquad [26]$$

where $\Delta_{\alpha\beta}^{\text{prev}}$ denotes the desired length scale of separation between $\mathbf{x}_\alpha$ and $\mathbf{x}_\beta$ 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; the SM provides further details.

It remains to specify how many landmarks to choose. In practice, we choose $n_\ell \propto n^{2/3}$; 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 $O(n_\ell^2)$ pairs of dissimilar landmarks and $O(n_\ell (\frac{n}{n_\ell})^2)$ pairs of dissimilar inputs belonging to the same landmarks. The overall number of such pairs is minimized by

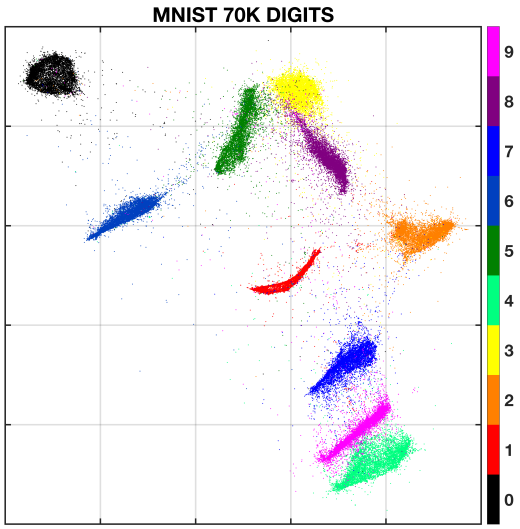

Fig. 2. Two dimensional visualization of 70K handwritten digits ($k=9, s=1, \ell=1$). Different clusters of digits are easily identified; in addition, the relative distances between clusters reveal more and less confusable classes of digits.

setting $n_\ell \propto n^{2/3}$, reducing the number of nonzero dissimilarity weights from $O(n^2)$ to $O(n^{4/3})$.

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 $n_\ell^{2/3}$ of them as *super-landmarks* and model their separation at even longer length scales; or, if there are $|\Omega_i|$ inputs assigned to the landmark $\mathbf{x}_i$, we can sample $|\Omega_i|^{2/3}$ of them as *sub-landmarks* and model their separation at a shorter length scale. In the idealized case of perfectly balanced assignments, a similar scaling argument (in the SM) shows that the next level of coarse-graining reduces the number of nonzero dissimilarity weights from $O(n^{4/3})$ 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 $\{\boldsymbol{\mu}_i\}_{i=1}^n$ from the bottom eigenvectors of a graph Laplacian matrix derived from the similarity weights $\mathcal{S}_{ij}$. Many further details and experiments are reported in the SM.

Fig. 2 shows a two dimensional visualization of the MNIST dataset of $n = 70000$ images of handwritten digits (24). Following earlier work (4), we preprocessed the 28×28 grayscale images by projecting them onto their first 50 principal components. The visualization by the LVM captures the different classes of digits with very little overlap; the results are similar to those from t-SNE (4) and UMAP (3). From start to finish, the embedding in Fig. 2 took 747 seconds to produce in MATLAB; this is comparable to the time of 751 seconds for the first benchmarked C++ implementation of tree-accelerated t-SNE (4), though not as fast as more recent methods (3, 5). We examine this embedding in more detail in the SM, which also explores the effects of different choices for k, s, and ℓ.

Our remaining experiments highlight the LVM's ability to learn higher dimensional embeddings. Fig. 3 shows the performance of a 9NN classifier on the LVM's embeddings ($k=9, s=2$) of varying dimensionality. The error rates were computed on the MNIST test set of 10K digits. The results show that most of the inter-class variation is captured in the first few dimensions learned by the LVM; by contrast, 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 coarse-graining incurs more slack from the triangle inequality. As a point of comparison, the 400 iterations of EM required 859 and 579 seconds, 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 three-million word vocabulary; the subset, containing $n = 207147$ word vectors, was obtained by filtering out words (such as misspellings and garbage phrases) not listed in WordNet (25). Fig. 4 compares the results of different embeddings by PCA and our LVM; the plot shows the percentage of words whose true nearest neighbor (in the full 300 dimensions) remained as one of the top r nearest neighbors after the embedding. In this regime, the results show that PCA requires 8-10 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 ten dimensions. The computation time for this embedding was dominated by the initial search for exact nearest neighbors, which took over two hours; the rest of the computation took roughly 30 minutes.

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. A similar EM algorithm as 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

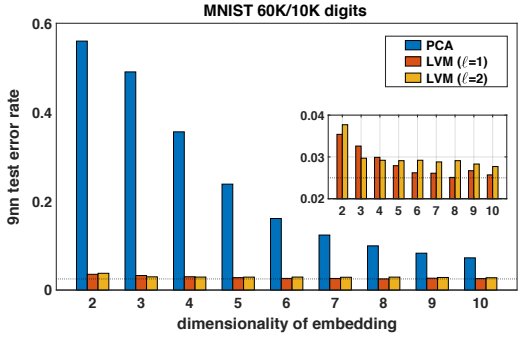

Fig. 3. Test error rates of 9nn classifiers on handwritten digits after dimensionality reduction by PCA versus latent variable modeling ($k = 9, s = 2$). The dotted line shows the test error rate when the digits are projected onto their top 50 principal components; these projections were the inputs to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

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

word	4nn (before)	4nn (after)
tractable	wilier, tractability, intractable, impulsion	tractability, Rube_Goldberg, effervescence, ingenious
latent	unexpressed, unarticulated, untapped, Latent	unexpressed, unvoiced, unarticulated, inchoate
variable	Variable, adjustable, quantize, variability	Variable, Fixed, fixed, assignable
model	models, Model, concept, paradigm	Model, models, Models, Modeling
nonlinear	linear, oscillatory, diffusive, fractal	linear, lineal, gravitational_attraction, gravitation
dimensionality	dimensional, translucency, tonal, spatiality	dimensional, Dimensional, Dimension, translucency
reduction	reductions, decrease, increase, increases	increases, decreases, increase, reductions

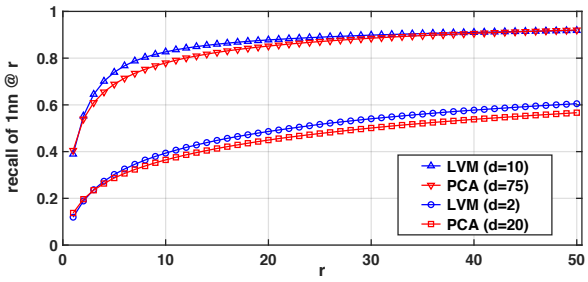

Fig. 4. Recall of 1st-nearest-neighbor at r, for the dataset of $n = 207147$ normalized word vectors, after linear dimensionality reduction by PCA and nonlinear dimensionality reduction by latent variable modeling $(k=4, s=3, \ell=2)$.

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 NN search (6, 27).

We emphasize, though, 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 data set of MNIST handwritten digits is available at http://yann.lecun.com/exdb/mnist/, and the Matlab word2vec data set is available at https://github.com/chrisjmccormick/word2vec_matlab.

ACKNOWLEDGMENTS. This paper is dedicated to the memory of Sam Roweis, who surely would have had many ideas for improving it. The Matlab experiments were made possible by the combinatorial multigrid solver (17) 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.

1. van der Maaten L, Hinton G (2008) Visualizing data using t-SNE. *Journal of Machine Learning Research* 9:2579–2605.
2. Tang J, Liu J, Zhang M, Mei Q (2016) Visualizing large-scale and high-dimensional data in *Proceedings of the 25th International Conference on World Wide Web*. pp. 287–297.
3. McInnes L, Healy J (2018) UMAP: Uniform manifold approximation and projection for dimension reduction. ArXiv e-prints 1802.03426.
4. van der Maaten L (2014) Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research* 15:3221–3245.
5. Linderman GC, Rachh M, Hoskins JG, Steinerberger S, Kluger Y (2019) Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data. *Nature Methods* 16:243–245.
6. Dong W, Charikar M, Li K (2011) Efficient k-nearest neighbor graph construction for generic similarity measures in *Proceedings of the 20th International Conference on World Wide Web (WWW-11)*. pp. 577–586.
7. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality in *Advances in Neural Information Processing Systems 26*, eds. Burges CJC, Bottou L, Welling M, Ghahramani Z, Weinberger KQ. (Curran Associates, Inc.), pp. 3111–3119.
8. Tenenbaum JB, de Silva V, Langford JC (2000) A global geometric framework for nonlinear dimensionality reduction. *Science* 290:2319–2323.
9. Roweis ST, Saul LK (2000) Nonlinear dimensionality reduction by locally linear embedding. *Science* 290:2323–2326.
10. Belkin M, Niyogi P (2003) Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation* 15(6):1373–1396.
11. Donoho DL, Grimes CE (2003) Hessian eigenmaps: locally linear embedding techniques for high-dimensional data. *Proc Natl Acad Sci USA* 100:5591–5596.
12. Coifman RR, et al. (2005) Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proc Natl Acad Sci USA* 102:7426–7431.
13. Rubin DB, Thayer DT (1982) EM algorithms for ML factor analysis. *Psychometrika* 47(1):69–76.
14. Redner R, Walker H (1984) Mixture densities, maximum likelihood, and the EM algorithm. *SIAM Review* 26:195–239.
15. Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B* 39(1):1–38.
16. Baum LE, Petrie T, Soules G, Weiss N (1970) A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics* 41(1):164–171.
17. Koutis I, Miller GL, Peng R (2012) A fast solver for a class of linear systems. *Commun. ACM* 55(10):99–107.
18. Spielman DA, Teng SH (2014) Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM Journal on Matrix Analysis and Applications* 35(3):835–885.
19. Carreira-Perpina MA, Zemel RS (2005) Proximity graphs for clustering and manifold learning in *Advances in Neural Information Processing Systems 17*, eds. Saul LK, Weiss Y, Bottou L. (MIT Press), pp. 225–232.
20. Silva VD, Tenenbaum JB (2003) Global versus local methods in nonlinear dimensionality reduction in *Advances in Neural Information Processing Systems 15*, eds. Becker S, Thrun S, Obermayer K. (MIT Press), pp. 721–728.
21. Platt JC (2004) Fast embedding of sparse similarity graphs in *Advances in Neural Information Processing Systems 16*, eds. Thrun S, Saul LK, Schölkopf B. (MIT Press), pp. 571–578.
22. Pezzotti N, Höllt T, Lelieveldt B, Eisemann E, Vilanova A (2016) Hierarchical stochastic neighbor embedding. *Computer Graphics Forum* 35(3):21–30.
23. van Unen V, et al. (2017) Visual analysis of mass cytometry data by hierarchical stochastic neighbour embedding reveals rare cell types. *Nature Communications* 8(1):1740.
24. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.
25. Miller GA (1995) WordNet: A lexical database for english. *Commun. ACM* 38(11):39–41.
26. Der M, Saul LK (2012) Latent coincidence analysis: A hidden variable model for distance metric learning in *Advances in Neural Information Processing Systems 25*, eds. Pereira F, Burges CJC, Bottou L, Weinberger KQ. (Curran Associates, Inc.), pp. 3230–3238.
27. Johnson J, Douze M, Jégou H (2017) Billion-scale similarity search with GPUs. ArXiv e-prints 1702.08734.
28. Blei DM, Kucukelbir A, McAuliffe JD (2017) Variational inference: A review for statisticians. *Journal of the American Statistical Association* 112(518):859–877.

PNAS

Supplementary Information for

A tractable latent variable model for nonlinear dimensionality reduction

Lawrence K. Saul

E-mail: saul@cs.ucsd.edu

This PDF file includes:

Supplementary text
Figs. S1 to S10
Tables S1 to S2
References for SI reference citations

Supporting Information Text

This supplement gives more details on the EM algorithm, the speed-ups from coarse-graining, the initialization of model parameters, and the experimental results on images and text. In particular, we provide many additional figures exploring the effects of the parameters k, s, and ℓ on the results of the algorithm. We also discuss potential applications that use metric (but not necessarily Euclidean) distances to assess which pairs of inputs should be regarded as similar examples.

EM Algorithm

Here we give a complete derivation of the EM updates for our latent variable model (LVM). First we discuss the E-step of the algorithm; in particular, we show how to calculate the statistics that we require of the LVM's posterior distributions. Then we discuss the M-step of the algorithm; in particular, we show how the statistics from the E-step are used to update the model parameters. For ease of implementation, we also collect all the key results for the EM algorithm in Table S1.

Inference (E-step). For the E-step of the EM algorithm, we must infer expected values of the model's latent variables. First we show how to calculate the statistics of the posterior distribution $P(\mathbf{h}, \mathbf{h}'|\mathbf{x}_i, \mathbf{x}_j, c_\delta = 1)$; these are the statistics that we require for pairs of nearby inputs. We start by computing the probability $P(c_\delta = 1|\mathbf{x}_i, \mathbf{x}_j)$, which is obtained from the Gaussian integral:

$$
\begin{aligned}
P(c_\delta = 1|\mathbf{x}_i, \mathbf{x}_j) &= \int_{\mathbf{h},\mathbf{h}'} P(c_\delta = 1, \mathbf{h}, \mathbf{h}'|\mathbf{x}_i, \mathbf{x}_j), && [1]\\
&= \int_{\mathbf{h},\mathbf{h}'} P(c_\delta = 1|\mathbf{h}, \mathbf{h}', \mathbf{x}_i, \mathbf{x}_j)\, P(\mathbf{h}|\mathbf{x}_i)\, P(\mathbf{h}'|\mathbf{x}_j), && [2]\\
&= \int_{\mathbf{h},\mathbf{h}'} \left[e^{-\frac{1}{2\delta_{ij}^2}\|\mathbf{h}-\mathbf{h}'\|^2} \right] \cdot \left[\frac{1}{(2\pi\sigma_i^2)^{\frac{d}{2}}}\, e^{-\frac{1}{2\sigma_i^2}\|\mathbf{h}-\boldsymbol{\mu}_i\|^2} \right] \cdot \left[\frac{1}{(2\pi\sigma_j^2)^{\frac{d}{2}}}\, e^{-\frac{1}{2\sigma_j^2}\|\mathbf{h}'-\boldsymbol{\mu}_j\|^2} \right], && [3]\\
&= \left(\frac{\delta_{ij}^2}{\delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right)^{\frac{d}{2}} \exp\left\{ -\frac{1}{2} \left(\frac{1}{\delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right) \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2 \right\}. && [4]
\end{aligned}
$$

From the intermediate result in eq. (3), we can also obtain some useful identities by the method of differentiating under the integral sign. In particular, we note that

$$
\begin{aligned}
\sigma_i^2 \frac{\partial}{\partial \boldsymbol{\mu}_i} \log P(c=1|\mathbf{x}_i, \mathbf{x}_j) &= \mathrm{E}[(\mathbf{h} - \boldsymbol{\mu}_i)|c=1, \mathbf{x}_i, \mathbf{x}_j], && [5]\\
\sigma_i^4 \frac{\partial^2}{\partial \boldsymbol{\mu}_i\, \partial \boldsymbol{\mu}_i^\top} \log P(c=1|\mathbf{x}_i, \mathbf{x}_j) &= -\sigma_i^2 \mathrm{I} \;+\; \mathrm{Cov}[\mathbf{h}|c=1, \mathbf{x}_i, \mathbf{x}_j]. && [6]
\end{aligned}
$$

We use the first of these identities, eq. (5), to compute the posterior means of the continuous latent variables $\mathbf{h}$ and $\mathbf{h}'$ for neighboring inputs $(\mathbf{x}_i, \mathbf{x}_j)$. For example, we have

$$
\begin{aligned}
\mathrm{E}[\mathbf{h}|c_\delta = 1, \mathbf{x}_i, \mathbf{x}_j] &= \int_{\mathbf{h}} P(\mathbf{h}\,|\,c_\delta = 1, \mathbf{x}_i, \mathbf{x}_j)\, \mathbf{h}, && [7]\\
&= \int_{\mathbf{h},\mathbf{h}'} P(\mathbf{h}, \mathbf{h}'|c_\delta = 1, \mathbf{x}_i, \mathbf{x}_j)\, (\boldsymbol{\mu}_i + \mathbf{h} - \boldsymbol{\mu}_i), && [8]\\
&= \boldsymbol{\mu}_i \;+\; \sigma_i^2 \frac{\partial}{\partial \boldsymbol{\mu}_i} \log P(c_\delta = 1|\mathbf{x}_i, \mathbf{x}_j), && [9]\\
&= \boldsymbol{\mu}_i \;-\; \left[\frac{\sigma_i^2}{\delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right] (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j). && [10]
\end{aligned}
$$

An analogous calculation gives the result for the posterior mean $\mathrm{E}[\mathbf{h}'|c_\delta = 1, \mathbf{x}_i, \mathbf{x}_j]$ of the latent variable $\mathbf{h}'$. Finally, we use the identity in eq. (6) to compute the second-order statistics that we require of this posterior distribution. What we need, in

particular, is the (posterior) expected squared Euclidean distance between each latent variable and its prior mean. Here we find

$$\begin{aligned}
\mathrm{E}\left[\left\|\mathbf{h}-\boldsymbol{\mu}_i\right\|^2 \,\middle|\, c_\delta=1,\mathbf{x}_i,\mathbf{x}_j\right] &= \mathrm{E}\left[\left\|\mathbf{h}-\mathrm{E}[\mathbf{h}|c_\delta=1,\mathbf{x}_i,\mathbf{x}_j]+\mathrm{E}[\mathbf{h}|c_\delta=1,\mathbf{x}_i,\mathbf{x}_j]-\boldsymbol{\mu}_i\right\|^2 \,\middle|\, c_\delta=1,\mathbf{x}_i,\mathbf{x}_j\right], & [11]\\
&= \mathrm{E}\left[\left\|\mathbf{h}-\mathrm{E}[\mathbf{h}|c_\delta=1,\mathbf{x}_i,\mathbf{x}_j]\right\|^2 \,\middle|\, c_\delta=1,\mathbf{x}_i,\mathbf{x}_j\right] + \left\|\mathrm{E}[\mathbf{h}|c_\delta=1,\mathbf{x}_i,\mathbf{x}_j]-\boldsymbol{\mu}_i\right\|^2, & [12]\\
&= \mathrm{trace}\big(\mathrm{Cov}[\mathbf{h}|c_\delta=1,\mathbf{x}_i,\mathbf{x}_j]\big) + \left\|\mathrm{E}[\mathbf{h}|c_\delta=1,\mathbf{x}_i,\mathbf{x}_j]-\boldsymbol{\mu}_i\right\|^2, & [13]\\
&= \mathrm{trace}\left[\sigma_i^2\mathrm{I}+\sigma_i^4\frac{\partial^2}{\partial\mathbf{h}\,\partial\mathbf{h}^\top}\log P(c_\delta=1|\mathbf{x}_i,\mathbf{x}_j)\right] + \left\|\mathrm{E}[\mathbf{h}|c_\delta=1,\mathbf{x}_i,\mathbf{x}_j]-\boldsymbol{\mu}_i\right\|^2, & [14]\\
&= d\sigma_i^2\left(1-\frac{\sigma_i^2}{\delta_{ij}^2+\sigma_i^2+\sigma_j^2}\right)+\left(\frac{\sigma_i^2}{\delta_{ij}^2+\sigma_i^2+\sigma_j^2}\right)^2\|\boldsymbol{\mu}_i-\boldsymbol{\mu}_j\|^2, & [15]\\
&= d\sigma_i^2+\sigma_i^4\left(\frac{1}{\delta_{ij}^2+\sigma_i^2+\sigma_j^2}\right)\left(\frac{\|\boldsymbol{\mu}_i-\boldsymbol{\mu}_j\|^2}{\delta_{ij}^2+\sigma_i^2+\sigma_j^2}-d\right). & [16]
\end{aligned}$$

Once again, an analogous calculation gives the result for $\mathrm{E}[\|\mathbf{h}'-\boldsymbol{\mu}_j\|^2 \,|\, c_\delta=1,\mathbf{x}_i,\mathbf{x}_j]$.

Note that the posterior distribution $P(\mathbf{h},\mathbf{h}'|c_\delta=1,\mathbf{x}_i,\mathbf{x}_j)$ is in fact multivariate Gaussian, which is one reason that its statistics are so readily computed. However, the statistics of this posterior distribution differ significantly from those of its prior counterpart. For example, in contrast to eqs. (10) and (16), the prior statistics are given simply by $\mathrm{E}[\mathbf{h}|\mathbf{x}_i]=\boldsymbol{\mu}_i$ and $\mathrm{E}[\|\mathbf{h}-\boldsymbol{\mu}_i\|^2|\mathbf{x}_i]=d\sigma_i^2$. In addition, the prior distribution factorizes as $P(\mathbf{h},\mathbf{h}|\mathbf{x}_i,\mathbf{x}_j)=P(\mathbf{h}|\mathbf{x}_i)\,P(\mathbf{h}|\mathbf{x}_j)$.

Next we show how to calculate the statistics of the posterior distribution $P(\mathbf{h},\mathbf{h}'|\mathbf{x}_i,\mathbf{x}_j,c_\Delta=0)$; these are the statistics that we require for pairs of dissimilar inputs. A similar result as eq. (4) holds for the probability $P(c_\Delta=1|\mathbf{h},\mathbf{h}',\mathbf{x}_i,\mathbf{x}_j)$. For pairs of dissimilar inputs, however, we will be mainly interested in the complementary probability

$$P(c_\Delta=0|\mathbf{x}_i,\mathbf{x}_j) \;=\; 1-\left(\frac{\Delta_{ij}^2}{\delta_{ij}^2+\sigma_i^2+\sigma_j^2}\right)^{\frac{d}{2}}\exp\left\{-\frac{1}{2}\left(\frac{1}{\Delta_{ij}^2+\sigma_i^2+\sigma_j^2}\right)\|\boldsymbol{\mu}_i-\boldsymbol{\mu}_j\|^2\right\}. \qquad [17]$$

The posterior distribution $P(\mathbf{h},\mathbf{h}'|\mathbf{x}_i,\mathbf{x}_j,c_\Delta=0)$ is not multivariate Gaussian, but the statistics we need can be derived from the ones we have already calculated. Let

$$\nu_{ij} \;=\; \frac{P(c_\Delta=1|\mathbf{x}_i,\mathbf{x}_j)}{P(c_\Delta=0|\mathbf{x}_i,\mathbf{x}_j)} \qquad [18]$$

denote the odds ratio when $\mathbf{x}_i$ and $\mathbf{x}_j$ are regarded as non-neighboring inputs; this notation will prove a useful shorthand in what follows. To compute the posterior mean $\mathrm{E}[\mathbf{h}|c_\Delta=0,\mathbf{x}_i,\mathbf{x}_j]$, we note that

$$\begin{aligned}
\mathrm{E}[\mathbf{h}|\mathbf{x}_i] &= \int_{\mathbf{h}} P(\mathbf{h}|\mathbf{x}_i)\,\mathbf{h}, & [19]\\
&= \int_{\mathbf{h}} P(\mathbf{h}|\mathbf{x}_i,\mathbf{x}_j)\,\mathbf{h}, & [20]\\
&= \int_{\mathbf{h}}\sum_{c\in\{0,1\}} P(\mathbf{h},c_\Delta=c|\mathbf{x}_i,\mathbf{x}_j)\,\mathbf{h}, & [21]\\
&= P(c_\Delta=0|\mathbf{x}_i,\mathbf{x}_j)\,\mathrm{E}[\mathbf{h}|c_\Delta=0,\mathbf{x}_i,\mathbf{x}_j] + P(c_\Delta=1|\mathbf{x}_i,\mathbf{x}_j)\,\mathrm{E}[\mathbf{h}|c_\Delta=1,\mathbf{x}_i,\mathbf{x}_j]. & [22]
\end{aligned}$$

Eq. (22) expresses the posterior mean $\mathrm{E}[\mathbf{h}|c_\Delta=0,\mathbf{x}_i,\mathbf{x}_j]$ in terms of the prior mean $\mathrm{E}[\mathbf{h}|\mathbf{x}_i]=\boldsymbol{\mu}_i$ and the posterior mean $\mathrm{E}[\mathbf{h}|c_\Delta=1,\mathbf{x}_i,\mathbf{x}_j]$, whose calculation is identical to that of $\mathrm{E}[\mathbf{h}|c_\delta=1,\mathbf{x}_i,\mathbf{x}_j]$ in eq. (10). Thus we can recover $\mathrm{E}[\mathbf{h}|c_\Delta=0,\mathbf{x}_i,\mathbf{x}_j]$ from our previous results. Rearranging terms and simplifying, we find

$$\mathrm{E}[\mathbf{h}|c_\Delta=0,\mathbf{x}_i,\mathbf{x}_j] \;=\; \boldsymbol{\mu}_i+\nu_{ij}\left[\frac{\sigma_i^2}{\Delta_{ij}^2+\sigma_i^2+\sigma_j^2}\right](\boldsymbol{\mu}_i-\boldsymbol{\mu}_j). \qquad [23]$$

An analogous calculation gives the result for the posterior mean of the latent variable $\mathbf{h}'$. We can compute the second-order statistics of the posterior distribution $P(\mathbf{h},\mathbf{h}'|\mathbf{x}_i,\mathbf{x}_j,c_\Delta=0)$ in a similar way. Once again, we appeal to the relation between prior and posterior statistics:

$$\mathrm{E}\left[\left\|\mathbf{h}-\boldsymbol{\mu}_i\right\|^2 \,\middle|\, \mathbf{x}_i\right] \;=\; P(c_\Delta=0|\mathbf{x}_i,\mathbf{x}_j)\,\mathrm{E}\left[\left\|\mathbf{h}-\boldsymbol{\mu}_i\right\|^2 \,\middle|\, c_\Delta=0,\mathbf{x}_i,\mathbf{x}_j\right] + P(c_\Delta=1|\mathbf{x}_i,\mathbf{x}_j)\,\mathrm{E}\left[\left\|\mathbf{h}-\boldsymbol{\mu}_i\right\|^2 \,\middle|\, c_\Delta=1,\mathbf{x}_i,\mathbf{x}_j\right]. \qquad [24]$$

As before, eq. (24) expresses $\mathrm{E}[\|\mathbf{h}-\boldsymbol{\mu}_i\|^2|c_\Delta=0,\mathbf{x}_i,\mathbf{x}_j]$ in terms of quantities that we have already shown how to compute. Rearranging terms and simplifying, we find

$$\mathrm{E}\left[\left\|\mathbf{h}-\boldsymbol{\mu}_i\right\|^2 \,\middle|\, c_\Delta=0,\mathbf{x}_i,\mathbf{x}_j\right] \;=\; d\sigma_i^2-\nu_{ij}\sigma_i^4\left(\frac{1}{\delta_{ij}^2+\sigma_i^2+\sigma_j^2}\right)\left(\frac{\|\boldsymbol{\mu}_i-\boldsymbol{\mu}_j\|^2}{\delta_{ij}^2+\sigma_i^2+\sigma_j^2}-d\right). \qquad [25]$$

An analogous calculation gives the result for $\mathrm{E}[\|\mathbf{h}' - \boldsymbol{\mu}_j\|^2 | c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j]$. Note that the posterior statistics in eq. (23) and eq. (25) have a fairly complicated dependence on the parameters of the latent variable model, but that much of this dependence is captured by the odds ratio ν_{ij} in eq. (18).

Learning (M-step). The EM updates for our LVM are derived by maximizing an iteratively constructed lower bound on its log conditional likelihood. The lower bound is used as a surrogate for the log conditional likelihood because it is easier to maximize. The log conditional likelihood for the LVM is given by

$$\mathcal{L} \;=\; \sum_{ij} \mathcal{S}_{ij} \log P(c_\delta = 1 | \mathbf{x}_i, \mathbf{x}_j) \;+\; \sum_{ij} \mathcal{D}_{ij} \log P(c_\Delta = 0 | \mathbf{x}_i, \mathbf{x}_j), \tag{26}$$

where the similarity weights $\mathcal{S}_{ij}$ and dissimilarity weights $\mathcal{D}_{ij}$ can be regarded as constants. As noted in the main paper, we employ one bound to derive the updates for the outputs $\{\boldsymbol{\mu}_i\}_{i=1}^n$ and another bound to derive the updates for the variances $\{\sigma_i^2\}_{i=1}^n$.

First we derive the updates for the outputs. Consider in particular the second term in eq. (26) involving pairs of dissimilar examples. Let $\{\tilde{\boldsymbol{\mu}}_i\}_{i=1}^n$ denote the updated values of the outputs that we seek to re-estimate, and let $\tilde{P}(c_\Delta = 0 | \mathbf{x}_i, \mathbf{x}_j)$ denote the probability computed from these updated values: i.e.,

$$\tilde{P}(c_\Delta = 0 | \mathbf{x}_i, \mathbf{x}_j) \;=\; 1 - \left(\frac{\Delta_{ij}^2}{\Delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right)^{\frac{d}{2}} \exp\left\{ -\frac{1}{2} \left(\frac{1}{\Delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right) \|\tilde{\boldsymbol{\mu}}_i - \tilde{\boldsymbol{\mu}}_j\|^2 \right\} \tag{27}$$

The EM algorithm exploits a lower bound due to Jensen's inequality. In particular, in the E-step of the algorithm, we compute

$$\log \tilde{P}(c_\Delta = 0 | \mathbf{x}_i, \mathbf{x}_j) \;\geq\; \mathrm{E}\left[\log \tilde{P}(\mathbf{h}, \mathbf{h}', c_\Delta | \mathbf{x}_i, \mathbf{x}_j) \,\middle|\, c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j \right] \;-\; \mathrm{E}\left[\log P(\mathbf{h}, \mathbf{h}' | c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j) \,\middle|\, c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j \right], \tag{28}$$

where the expectation is taken with respect to the posterior distribution $P(\mathbf{h}, \mathbf{h}' | c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j)$ using the current estimates of the outputs $\{\boldsymbol{\mu}_i\}_{i=1}^n$ and variances $\{\sigma_i\}_{i=1}^n$. Note that only the first term on the right hand side depends on the updated outputs $\{\tilde{\boldsymbol{\mu}}_i\}_{i=1}^n$. Expanding this term, we find

$$\begin{aligned} &\mathrm{E}\left[\log \tilde{P}(\mathbf{h}, \mathbf{h}', c_\Delta | \mathbf{x}_i, \mathbf{x}_j) \,\middle|\, c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j \right] \\ &\quad= \mathrm{E}\left[-\frac{d}{2} \log(2\pi\sigma_i^2) - \frac{\|\mathbf{h} - \tilde{\boldsymbol{\mu}}_i\|^2}{2\sigma_i^2} - \frac{d}{2} \log(2\pi\sigma_j^2) - \frac{\|\mathbf{h}' - \tilde{\boldsymbol{\mu}}_j\|^2}{2\sigma_j^2} + \log\left(1 - e^{-\frac{1}{2}\|\mathbf{h} - \mathbf{h}'\|^2 / \Delta_{ij}^2} \right) \,\middle|\, c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j \right]. \end{aligned} \tag{29}$$

The updates are derived by maximizing sums of terms of this form with respect to the re-estimated values of the outputs. For the purpose of this maximization, we only need to evaluate the terms on the right hand side of eq. (29) that depend on these values. In particular, we can write

$$\log \tilde{P}(c_\Delta = 0 | \mathbf{x}_i, \mathbf{x}_j) \;\geq\; -\frac{\|\tilde{\boldsymbol{\mu}}_i - \mathrm{E}[\mathbf{h} | c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j]\|^2}{2\sigma_i^2} \;-\; \frac{\left\|\tilde{\boldsymbol{\mu}}_j - \mathrm{E}[\mathbf{h}' | c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j]\right\|^2}{2\sigma_j^2} \;+\; \texttt{const}, \tag{30}$$

where `const` is a placeholder for terms that do not depend on $\tilde{\boldsymbol{\mu}}_i$ and $\tilde{\boldsymbol{\mu}}_j$. Note that the right hand side of eq. (30) is a quadratic form in the parameters $\tilde{\boldsymbol{\mu}}_i$ and $\tilde{\boldsymbol{\mu}}_j$. But this is also true for the terms involving neighboring examples in the log conditional likelihood, eq. (26); in particular, we have

$$\log \tilde{P}(c_\delta = 1 | \mathbf{x}_i, \mathbf{x}_j) \;=\; -\frac{1}{2} \left(\frac{1}{\delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right) \|\tilde{\boldsymbol{\mu}}_i - \tilde{\boldsymbol{\mu}}_j\|^2 \;+\; \texttt{const}. \tag{31}$$

To obtain a lower bound on the overall log conditional likelihood, we substitute the results from eqs. (30–31) into eq. (26). Making these substitutions, we find

$$\begin{aligned} \mathcal{L}\big(\tilde{\mathbf{h}}_1, \tilde{\mathbf{h}}_2, \ldots, \tilde{\mathbf{h}}_n\big) \;\geq\; &-\frac{1}{2} \sum_{ij} \mathcal{S}_{ij} \left(\frac{1}{\delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right) \|\tilde{\boldsymbol{\mu}}_i - \tilde{\boldsymbol{\mu}}_j\|^2 \\ &- \frac{1}{2} \sum_{ij} \mathcal{D}_{ij} \left(\frac{\|\tilde{\boldsymbol{\mu}}_i - \mathrm{E}[\mathbf{h} | c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j]\|^2}{\sigma_i^2} \;+\; \frac{\left\|\tilde{\boldsymbol{\mu}}_j - \mathrm{E}[\mathbf{h}' | c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j]\right\|^2}{\sigma_j^2} \right) \;+\; \texttt{const}. \end{aligned} \tag{32}$$

The lower bound on the right hand side is a quadratic form in the parameters $\{\tilde{\mathbf{h}}\}_{i=1}^n$ that we seek to re-estimate. It is maximized by solving the symmetric, diagonally dominant linear system given in the main paper.

Next we derive the updates for the variances. This derivation follows the standard recipe for the EM algorithm. Let $\{\tilde{\sigma}_i\}_{i=1}^n$ denote the updated values of the variances that we seek to re-estimate; now we assume that the values of the outputs $\{\boldsymbol{\mu}_i\}_{i=1}^n$ are fixed. To update the variances, we use Jensen's inequality to derive lower bounds on both types of probabilities—those

for pairs of neighboring examples, as well as for pairs of dissimilar examples—that appear in the log conditional likelihood, eq. (26). Then the same procedure as above shows that

$$\log \tilde{P}(c_\delta = 1 | \mathbf{x}_i, \mathbf{x}_j) \;\geq\; -d\,(\log\tilde{\sigma}_i + \log\tilde{\sigma}_j) - \mathrm{E}\left[\frac{\|\mathbf{h}-\boldsymbol{\mu}_i\|^2}{2\tilde{\sigma}_i^2} + \frac{\|\mathbf{h}'-\boldsymbol{\mu}_j\|^2}{2\tilde{\sigma}_j^2}\,\middle|\, c_\delta = 1, \mathbf{x}_i, \mathbf{x}_j\right] \;+\; \texttt{const}, \quad [33]$$

$$\log \tilde{P}(c_\Delta = 0 | \mathbf{x}_i, \mathbf{x}_j) \;\geq\; -d\,(\log\tilde{\sigma}_i + \log\tilde{\sigma}_j) - \mathrm{E}\left[\frac{\|\mathbf{h}-\boldsymbol{\mu}_i\|^2}{2\tilde{\sigma}_i^2} + \frac{\|\mathbf{h}'-\boldsymbol{\mu}_j\|^2}{2\tilde{\sigma}_j^2}\,\middle|\, c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j\right] \;+\; \texttt{const}, \quad [34]$$

where `const` is a placeholder for terms that do not depend on $\tilde{\sigma}_i$ or $\tilde{\sigma}_j$. As in the main paper, it is convenient to introduce the shorthand

$$\phi_{ij} \;=\; \mathrm{E}\big[\|\mathbf{h}-\boldsymbol{\mu}_i\|^2 \,|\, c_\delta = 1, \mathbf{x}_i, \mathbf{x}_j\big], \quad [35]$$

$$\phi'_{ij} \;=\; \mathrm{E}\big[\|\mathbf{h}'-\boldsymbol{\mu}_j\|^2 \,|\, c_\delta = 1, \mathbf{x}_i, \mathbf{x}_j\big], \quad [36]$$

$$\psi_{ij} \;=\; \mathrm{E}\big[\|\mathbf{h}-\boldsymbol{\mu}_i\|^2 \,|\, c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j\big], \quad [37]$$

$$\psi'_{ij} \;=\; \mathrm{E}\big[\|\mathbf{h}'-\boldsymbol{\mu}_j\|^2 \,|\, c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j\big]. \quad [38]$$

We obtain an overall lower bound on the log conditional likelihood by substituting the results from eqs. (33–34) into eq. (26). Making these substitutions with the above notation, we find

$$\mathcal{L}\big(\{\tilde{\sigma}_i\}_{i=1}^n\big) \;\geq\; -\sum_{ij} \mathcal{S}_{ij}\left(d\log\tilde{\sigma}_i + d\log\tilde{\sigma}_j + \frac{\phi_{ij}}{2\tilde{\sigma}_i^2} + \frac{\phi'_{ij}}{2\tilde{\sigma}_j^2}\right) - \sum_{ij}\mathcal{D}_{ij}\left(d\log\tilde{\sigma}_i + d\log\tilde{\sigma}_j + \frac{\psi_{ij}}{2\tilde{\sigma}_i^2} + \frac{\psi'_{ij}}{2\tilde{\sigma}_j^2}\right) + \texttt{const}. \quad [39]$$

The lower bound on the right hand side has a simple dependence on the parameters $\{\tilde{\sigma}_i\}_{i=1}^n$ that we seek to re-estimate. It is maximized by the update given in the main paper.

Table S1 collects all the key results needed to implement the EM algorithm. The derivation of these updates assumes that they are carried out in an alternating manner: the variances are held fixed while computing the statistics to update the outputs, and the outputs are held fixed while computing the statistics to update the variances. It is possible to derive joint updates for the outputs and variances, but we found that these updates—based on a looser overall bound of the log conditional likelihood—converge more slowly in practice.

Analysis of coarse-graining

Here we analyze the potential speedups obtained by the coarse-graining procedure. First we analyze the gains when one level ($\ell = 1$) of coarse-graining is applied. Then we use this result to analyze the gains from an additional round ($\ell = 2$) of coarse-graining.

The first level of coarse-graining aims to sparsify a dense $n \times n$ matrix whose nonzero elements indicate pairs of dissimilar examples. This is done by selecting $n_\ell < n$ landmarks and maximizing the conditional likelihood that the n_ℓ landmarks are widely separated by the embedding while examples belonging to the same landmark are relatively clustered. To this end, the matrix of dissimilarity weights in the coarse-grained model encodes only the dissimilarities between different landmarks (on larger length scales) and the dissimilarities between examples that belong to the same landmark (on smaller length scales). Suppose that the examples are evenly distributed with n/n_ℓ examples per landmark. Then to highest order, the dissimilarities after coarse-graining are modeled by a sparser matrix with

$$e_1 \;=\; \tfrac{1}{2}n_\ell^2 \;+\; \tfrac{1}{2}n_\ell\,(n/n_\ell)^2 \quad [40]$$

nonzero elements; the first term in this expression measures the number of pairs of landmarks, while the second measures the number of pairs of examples belonging to the same landmark. This overall number is minimized by setting $n_\ell = n^{2/3}/\sqrt[3]{2}$, as a result of which

$$e_1 \;\sim\; O(n^{4/3}). \quad [41]$$

In effect, the coarse-graining procedure replaces a dense $n \times n$ matrix by a sparser matrix with one dense $n^{2/3} \times n^{2/3}$ subblock (encoding dissimilarities between landmarks) and $n^{2/3}$ dense $n^{1/3} \times n^{1/3}$ subblocks (encoding dissimilarities within landmarks).

The second level of coarse-graining takes this one step further, applying this procedure to each of the dense subblocks obtained at the first level. In this way the dense subblock of size $n^{2/3} \times n^{2/3}$ is replaced by sparser subblock of $O(n^{8/9})$ elements, while each dense subblock of size $n^{1/3} \times n^{1/3}$ is replaced by a sparser subblock of $O(n^{4/9})$ elements. Since there are $n^{2/3}$ of the latter, the overall number of nonzero elements at this second level of coarse-graining scales as $n^{2/3+4/9}$, or

$$e_2 \;\sim\; O(n^{10/9}). \quad [42]$$

The above analysis is based on the simplifying assumption that at each level of coarse-graining, the examples are evenly distributed among landmarks. While this assumption does not hold in practice, we have observed that the gains are indeed substantial from one level to the next, and that the larger the number of examples, the more significant the gains from additional levels.

The coarse-graining procedure for our LVM has some high-level parallels to Barnes-Hut accelerated implementations of t-SNE. Inspired by many-body simulations from physics, the latter also involve some implicit form of coarse-graining, in which nearby inputs are grouped so as to aggregate their contributions to the gradient. These procedures for t-SNE differ from ours, however, in that rely on gridding a low dimensional space. As a result, they are perfectly suited for problems in visualization, but not as suitable for learning higher dimensional embeddings.

Our coarse-graining procedure also draws on ideas from renormalization-group (RG) methods in statistical mechanics (1). RG methods were developed to analyze physical systems at different length scales, but more recently they have been proposed as a framework for understanding the successes of deep learning (2, 3). This connection seems deserving of further study.

As noted in the main paper, many previous studies have used landmarks in some way to accelerate algorithms in NLDR. Specific usages, though, tend to vary, and it is hard to generalize across different algorithms about how landmark-based approximations affect the quality of embeddings. We return to this question later and present experimental results to illustrate not only the speed-ups from coarse-graining in our LVM, but also the effects on learned embeddings.

Implementation

The EM algorithm converges in general to a local maximum of the log conditional likelihood; therefore, the results of the algorithm depend on how the parameters of the LVM are initialized. We initialized the outputs $\{\boldsymbol{\mu}_i\}_{i=1}^n$ from the d bottom (non-trivial) eigenvectors of the graph Laplacian arising from the quadratic form $\sum_{ij} \mathcal{S}_{ij} \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2$. This initialization maps nearby inputs to nearby outputs, but in general it does not map dissimilar inputs to dissimilar outputs; thus, further learning is required to obtain a neighborhood-preserving mapping. We initialized the variances as $\sigma_i^2 = \frac{1}{2d} \max_j E_{ij} \|\mathbf{x}_i - \mathbf{x}_j\|^2$. This initialization ensures that the variances are on the same order as the distances between nearby inputs.

The results of the EM algorithm also depend on the relative weighting of terms in the log conditional likelihood. To balance these terms, we rescale the dissimilarity weights so that $\sum_{ij} \mathcal{D}_{ij} = \sum_{ij} \mathcal{S}_{ij}$. Before this, we also perform an internal rescaling of the similarity weights. Recall that there are two types of nonzero similarity weights. First, there are unit similarity weights between examples that are connected by edges E_{ij} of the neighborhood graph; the construction of the neighborhood graph ensures that $\sum_{ij} E_{ij} \leq kn$, since it consists of a subset of edges of the (directed) kNN graph. Second, there are similarity weights between examples that are connected by the coarse-graining procedure—for example, between examples and their landmarks after one round of coarse-graining, or between landmarks and their super-landmarks after two rounds of coarse-graining. We rescale the similarity weights arising from the coarse-graining procedure so that they sum to n. Intuitively, this rescaling calibrates the overall effect of the landmarks so that each output, on average, is attracted to one additional output besides its neighbors. Before these rescalings, the similarity and dissimilarity weights have nonnegative integer values; recall that weights arising from the coarse-graining procedure count the number of examples that they (implicitly) connect. After these rescalings, however, the similarity and dissimilarity weights may take on fractional values.

The astute reader may have noticed that before coarse-graining ($\ell = 0$), the length scale Δ_{ij} does not depend on the index j. In particular, for this base case of the LVM, we defined

$$\Delta_{ij}^2 = \left(\frac{1}{2 \log 2}\right) \max_k \left[E_{ik} \, \|\mathbf{x}_i - \mathbf{x}_k\|^2\right].$$

The above equation has a simple interpretation: before coarse-graining, for each input $\mathbf{x}_i$, the LVM attempts to keep the distance from $\mathbf{x}_i$ to other dissimilar inputs at least as large as the distance from $\mathbf{x}_i$ to any of its similar inputs. However, the length scale Δ_{ij} does acquire a dependence on j, as a result of eq. (26), after one or more rounds of coarse-graining ($\ell \geq 1$). Hence, the more general notation is needed.

EM algorithms do not require the tuning of learning rates. However, other gradient-based strategies can help to accelerate the convergence of EM. In our experiments, we obtained further speedups by incorporating a momentum term (4, 5) into the EM update for the outputs. More precisely, this was done as follows. Let $\boldsymbol{\mu}_i^{(t)}$ denote the estimate of the output $\boldsymbol{\mu}_i$ at the tth iteration of learning, and let $\boldsymbol{\mu}_i^{(t+1,\,\mathrm{EM})}$ denote the result of the EM update as described in the first section. Then the momentum-accelerated update is given by

$$\boldsymbol{\mu}_i^{(t+1)} \;=\; \boldsymbol{\mu}_i^{(t+1,\mathrm{EM})} + \beta \left(\boldsymbol{\mu}_i^{(t)} - \boldsymbol{\mu}_i^{(t-1)}\right), \qquad [43]$$

where β is the momentum parameter, which we set to a constant value of 0.9. Though this momentum-accelerated update does not preserve the guarantee of monotonic convergence in the log conditional likelihood, in practice we found it to be remarkably stable. There has been a great deal of work on strategies for accelerating EM algorithms (6–14), and it is likely that further speedups can be obtained beyond the simple heuristic considered above.

Results

In this section we examine the embeddings obtained by our LVM in more detail. We also explore the effect of the model's different tuning parameters—namely, the number of nearest neighbors (k), the number of steps (s) along the neighborhood graph, and the number of levels of coarse-graining (ℓ).

Detecting clusters. We begin by revisiting the two-dimensional embedding of MNIST handwritten digits in Fig. 2 of the main paper. Such embeddings have been used as a test case for many algorithms, including t-SNE and UMAP. For this dataset, the goal of visualization is to reveal the ten different classes of digits as easily identified clusters. Fig. S1 shows a logarithmic heatmap of the embedding in Fig. 2; though the heatmap does not color-code the digit classes, it nevertheless reveals ten clusters formed by the embedded images of nearby inputs.

Detecting outliers. Fig. S2 shows again the same embedding as Fig. 2 but omits the most *uncertain* outputs of the LVM. In particular, recall that for each high dimensional input $\mathbf{x}_i$, the LVM learns not only a low dimensional output $\boldsymbol{\mu}_i$, but also a corresponding variance σ_i^2. When the LVM has faithfully preserved the neighborhood of an input $\mathbf{x}_i$, its log conditional likelihood is maximizing by setting the variance σ_i^2 to a very small value. When this is not the case, however, the log conditional likelihood is maximized by setting the variance σ_i^2 to a larger value; otherwise, the latent variable for this input would be highly localized, and as such, it would have negligible probability to satisfy the many (and perhaps competing) demands of a neighborhood-preserving embedding. Fig. S2 reproduces the embedding of Fig. 2 but excludes the 3500 (out of $n = 70000$) outputs whose variances exceed the 95% quantile. The inset of the left subplot shows a histogram of the $n = 70000$ values of $\{\sigma_i^2\}_{i=1}^n$ for this embedding; here we observe that these variances range over several orders of magnitude.

The plot in Fig. S2 has much less speckle than the original one, suggesting that large variances provide a way of identifying outliers in the embedding (or perhaps even in the original space of inputs). We investigate this idea further in Fig. S3. The top plot of this figure highlights the 100 most uncertain outputs (i.e., those with the largest variances σ_i^2) in the embedding of Fig. 2. The bottom plot shows the digit images corresponding to these outputs. It appears that large variances do have some correlation with outliers in the original space of inputs: a sizable fraction of these images (roughly one-quarter) reveal highly atypical examples of the classes they represent. Presumably the other images correspond to outputs whose neighborhoods were distorted as a result of their embedding into a much lower dimensional space.

We also performed an experiment to understand the role of the variances during learning. Fig. S4 shows the much worse embedding (with the same parameters k, s, ℓ) that is obtained when all of the variances $\{\sigma_i^2\}_{i=1}^n$ are constrained to share the same value. When the variances are tied in this way, the outputs $\{\boldsymbol{\mu}_i\}_{i=1}^n$ do not have as much freedom to produce a faithful embedding that separates the different classes of digits. In addition, this embedding has a significantly lower likelihood than the previous one.

Effect of graph parameters. Next we explore the effect of the parameters (k, s) that determine the model's neighborhood graph. Fig. S5 shows the two-dimensional embeddings of MNIST digits obtained from different values of k and s. Of course, the number of neighbors k has a significant effect on the visual aspects of the embedding. If k is too small, then the neighborhood graph itself may be disconnected; if k is too large (as in the bottom row) then the neighborhood graph will connect more distant inputs, and the LVM will attempt to co-locate these inputs in the embedding. In general, there is a range of values of k (as shown in the top and middle rows) which yield meaningful embeddings. Within this range, one effect of k is also apparent—namely, higher values of k tend to yield more tightly observed clusters. This dependence on the neighborhood size is common to all algorithms for NLDR.

On the other hand, Fig. S5 also suggests that the value of s does not heavily affect the visual aspects of the embedding (at least, not when k is appropriately chosen). *Thus when the LVM is used for visualization, it seems simplest to use a default value of* $s = 1$. This default value also gives rise to a very simple similarity graph between inputs, whose edges correspond either to mutual k nearest neighbors (where $K_{ij} = K_{ji} = 1$) or to edges in the minimum spanning tree of the symmetrized k-nearest-neighbor graph (given by $K_{ij} + K_{ji} - K_{ij}K_{ji}$).

The utility of s becomes more apparent when we consider other (non-visual) aspects of the embedding. For example, Fig. S5 shows that the value of s does affect the 1-nearest-neighbor error rate of outputs in the MNIST test set. In general, our experiments suggest that other (small) values of s are worth exploring when other measures of the embedding are being optimized. For example, we found that the value $s = 2$ gave the best test error rates on MNIST digits (as shown in Fig. 3 of the main paper) and that the value $s = 3$ gave the best recall rates on the word2vec data (as shown in Fig. 4).

Effect of coarse-graining. Next we examine the effect on embeddings of the number of levels of coarse-graining (ℓ). In practice, the choice of ℓ is determined by the size of the dataset: one simply chooses the smallest value of ℓ such that the LVM can be trained in a reasonable amount of time and memory. Thus, for example, no coarse-graining ($\ell = 0$) is required for datasets with thousands of inputs, whereas multiple levels ($\ell > 1$) are likely to be required for datasets with hundreds of thousands of inputs. Even though ℓ is not tuned in the same way as the parameter k, however, it remains important to understand the effect of coarse-graining on the results of embeddings.

Figs S6–S8 illustrate the effect of coarse-graining on embeddings of MNIST digits. In particular, Figs. S6–S7 contrast the effect of one versus two levels of coarse-graining. In this case, the extra level of coarse-graining has the noticeable effect of magnifying the distances between clusters. Intuitively, this is to be expected: recall that each level of coarse-graining is based on an application of triangle inequalities, and the slack in these inequalities leads to over-estimates of the distances between faraway inputs. These figures also show close-up plots of the most confusable classes of digits (3/5/8 and 4/7/9) in these embeddings; the close-ups resolve some ambiguities when the visualizations are viewed at larger scales.

In a similar way, Fig. S8 illustrates the effect of the first level of coarse-graining on embeddings of MNIST digits. The experiments for this figure were of necessity performed on a smaller subset ($n = 7000$) of MNIST digits. Here we observe a similar trend: the effect of coarse-graining is to magnify the distances between clusters of inputs.

Given this effect of coarse-graining, it is natural to wonder if the procedure might artificially induce clusters in the embeddings of datasets where the inputs are not in fact tightly clustered. To examine this possibility, Fig. S9 shows the effect of coarse-graining on embeddings of $n=5000$ inputs sampled from the so-called Swiss roll. In this case, the inputs lie on a two-dimensional manifold, and the goal of NLDR is to discover a neighborhood-preserving mapping of the inputs into the plane. The figure shows that at least for this data set, where the underlying manifold is densely sampled, the effect of coarse-graining is mild; in particular, the embedding with one level of coarse-graining ($\ell=1$) does not exhibit spurious clusters.

Scaling with dimensionality. Fig. S10 shows how the learning algorithm for our LVM scales with the dimensionality of the embedding. The vertical axis of this plot shows the amount of time required for 400 iterations of the EM algorithm. As expected, the amount of time scales gracefully with the dimensionality of the embedding. This graceful scaling is shared by the algorithms for LargeVis and UMAP, which (as mentioned previously) are based on closely related objective functions. However, this behavior is quite different than leading algorithms for t-SNE, which are tailored specifically to embeddings in two and three dimensions.

Word vector embedding. Table S2 shows more nearest neighbors of word vectors before and after the LVM's embedding into ten dimensions. The words in this table were taken (including capitalization) from the preamble of the U.S. Constitution. The recall of nearest neighbors for words in this table is considerably higher than the recall averaged over the entire vocabulary of $n = 207147$ words. Anecdotally, we observe that the neighbors of more common words seem to be better preserved than those of less common words, but we do not have an explanation of why this is so.

Metric versus Euclidean distances

Throughout the main paper, we have assumed that the inputs $\{\mathbf{x}_i\}_{i=1}^n$ are high dimensional vectors and that the Euclidean distances $\|\mathbf{x}_i - \mathbf{x}_j\|$ best reflect which pairs of inputs $(\mathbf{x}_i, \mathbf{x}_j)$ should be regarded as nearby. These assumptions can be relaxed when other similarity metrics are more natural in the original space of inputs. We discuss briefly how this can be done.

Let $d(\mathbf{x}_i, \mathbf{x}_j)$ denote a measure of distance between the inputs $\mathbf{x}_i$ and $\mathbf{x}_j$. For example, if $\mathbf{x}_i$ and $\mathbf{x}_j$ are high dimensional vectors, then we might compute $d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_p$ from the ℓ_p-norm on the vector space, or if $\mathbf{x}_i$ and $\mathbf{x}_j$ are discrete sequences, we might compute $d(\mathbf{x}_i, \mathbf{x}_j)$ from their Levenshtein (edit) distance. These alternate distances can be substituted in the obvious way to derive the length scales δ_{ij} and Δ_{ij} that appear in our LVM. In particular, for the base model ($\ell = 0$) before coarse-graining, we can define

$$\delta_{ij}^2 = d(\mathbf{x}_i, \mathbf{x}_j)^2/(2\log 2), \quad [44]$$

$$\Delta_{ij}^2 = \max_k \left[E_{ik}\, d(\mathbf{x}_i, \mathbf{x}_k)^2\right]/(2\log 2), \quad [45]$$

where again the factors of $2\log 2$ are used to calibrate the model's probabilities so that, for example, $P(c_\delta = 1|\, \mathbf{h}, \mathbf{h}', \mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2}$ when $\|\mathbf{h} - \mathbf{h}'\| = \delta_{ij}$. These length scales for the base model ($\ell=0$) can then be used in the same way as before to derive the length scales at higher levels ($\ell \geq 1$) of coarse-graining.

It is possible that non-Euclidean measures of distance may lead to novel applications of NLDR. We note, however, that the coarse-graining procedures for our LVM are rooted in triangle inequalities; thus, implicitly, in the derivation of these procedures, we have assumed that the distance function between inputs satisfies the properties of a metric. For the purpose of NLDR, this does not seem a particularly restrictive assumption to impose on the distance function in the original space of inputs. We note, for example, that metric distances are obtained from both the ℓ_p norm (for $p \geq 1$) on vector spaces and the Levenshtein distance on sequences.

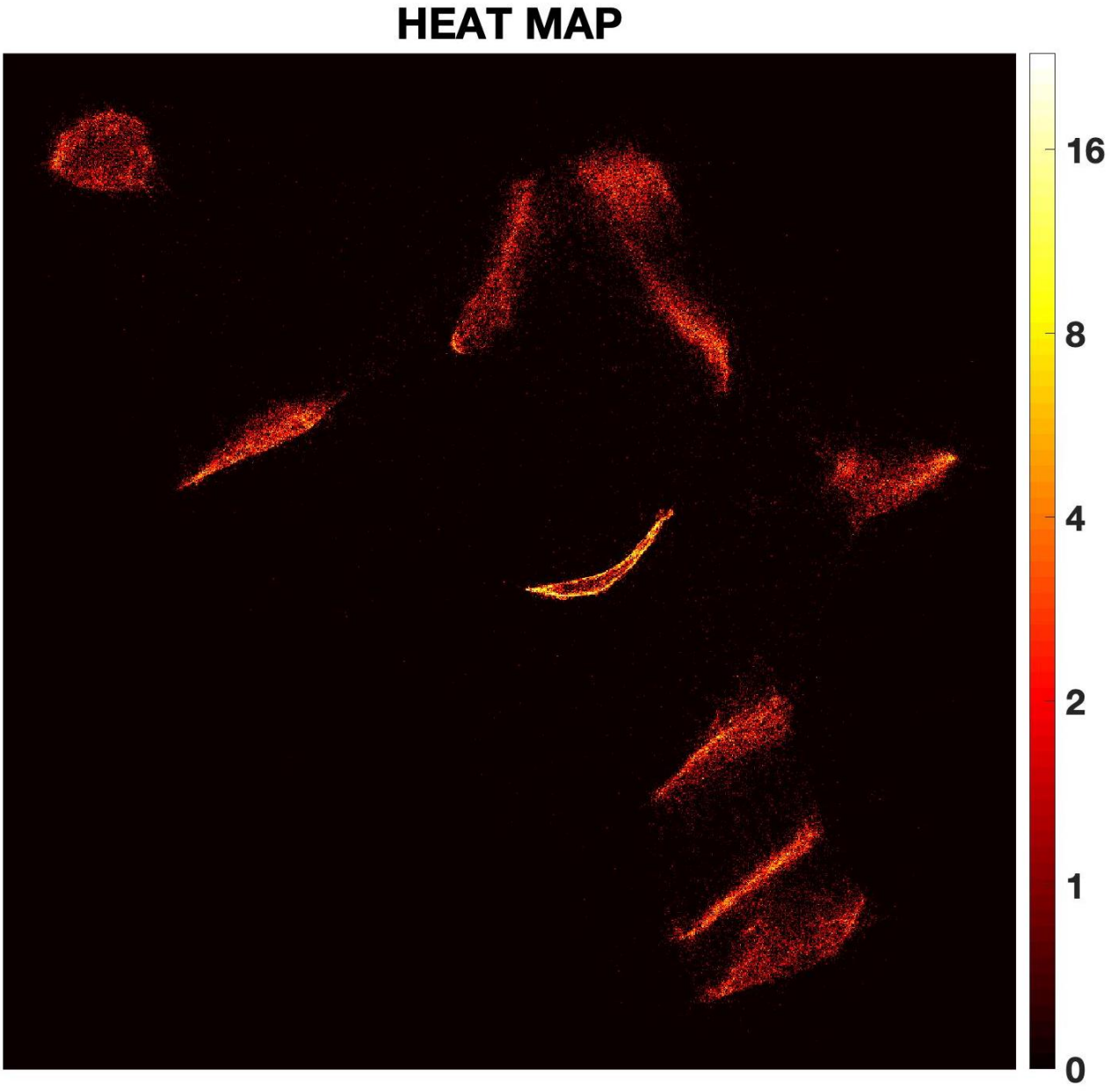

Fig. S1. Heatmap of the two dimensional embedding of $n = 70000$ MNIST handwritten digits in Fig. 2 of the main paper. The heatmap was obtained by binning the plane and computing the number of points within each bin. The heatmap's colors are coded on a logarithmic scale, revealing ten clusters of data, one for each class of digits.

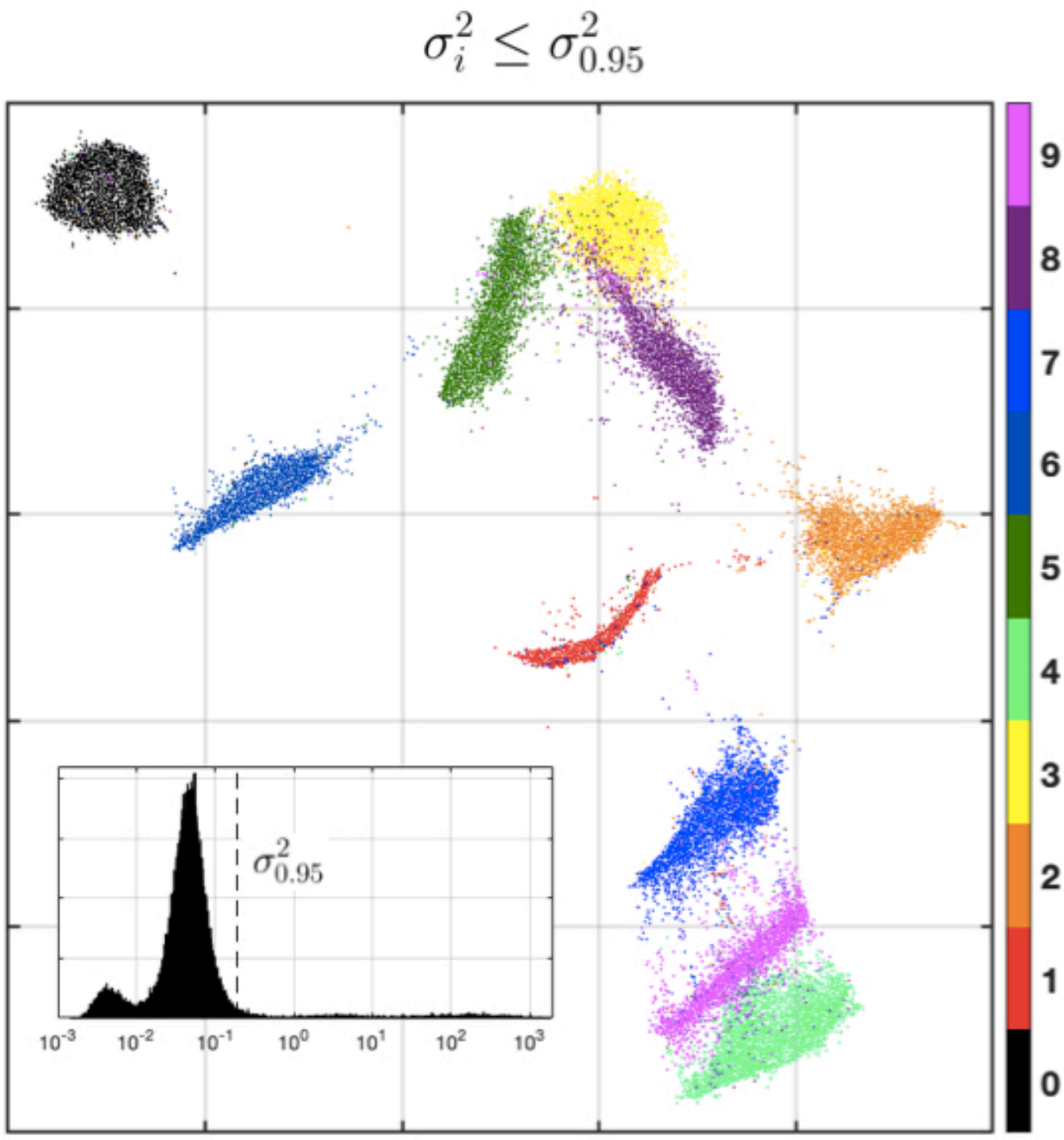

Fig. S2. For each high dimensional input $\mathbf{x}_i$, our LVM learns a low dimensional output $\boldsymbol{\mu}_i$ and a variance σ_i^2. Lower variances are attached to outputs that are situated with higher certainty in the low dimensional embedding. This plot shows the same embedding as Fig. 2, but excludes the most uncertain outputs—those whose variances exceed the 95% quantile. The resulting plot has much less speckle, showing that the outputs outside dense clusters correspond to outputs with larger amounts of uncertainty. The inset shows a histogram of all $n = 70000$ variances $\{\sigma_i^2\}_{i=1}^n$, which range over several orders of magnitude.

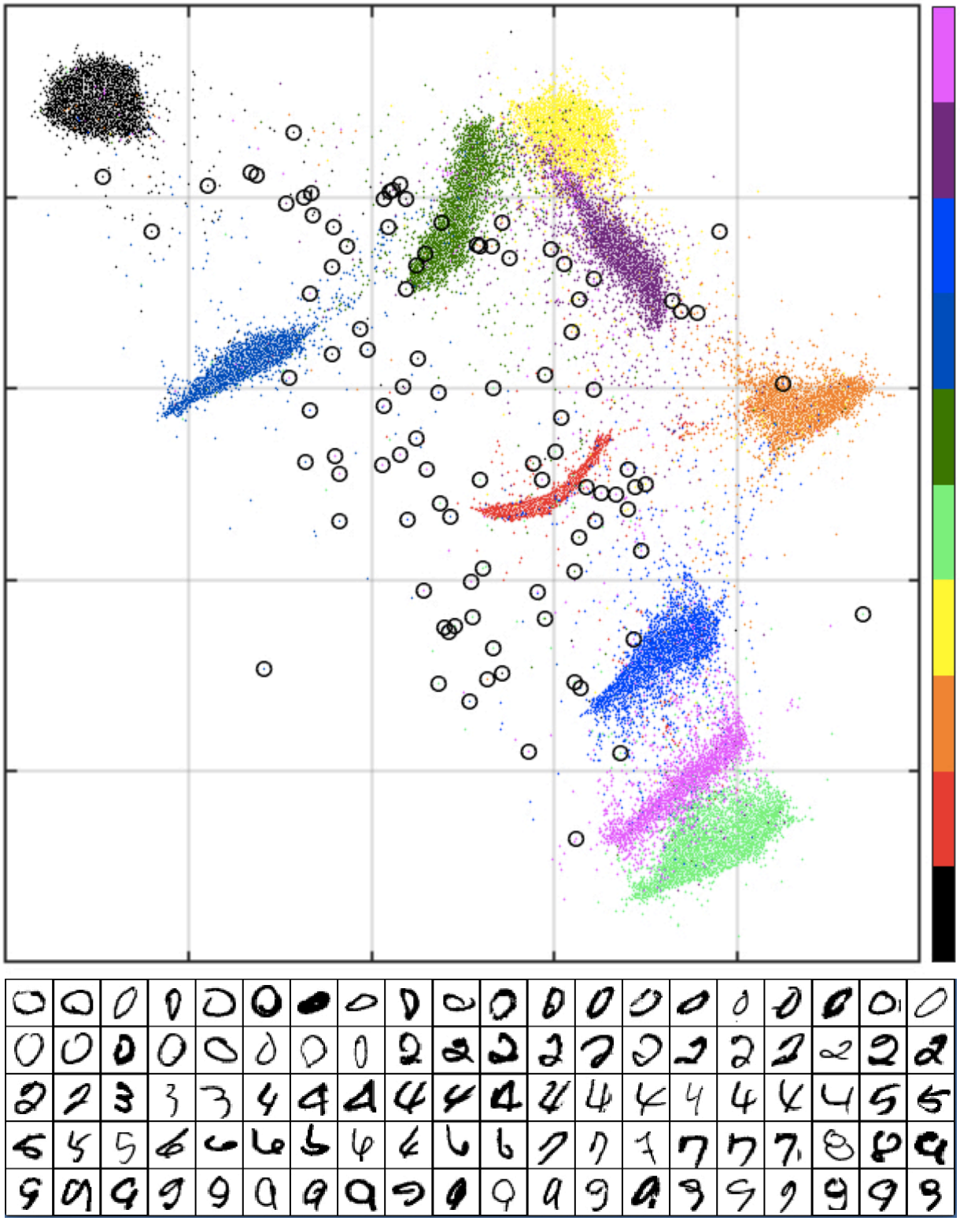

Fig. S3. The top plot shows the same embedding as Fig. 2, but circles the most 100 outputs with the highest variances. The bottom figure shows the digit images corresponding to the circled outputs.

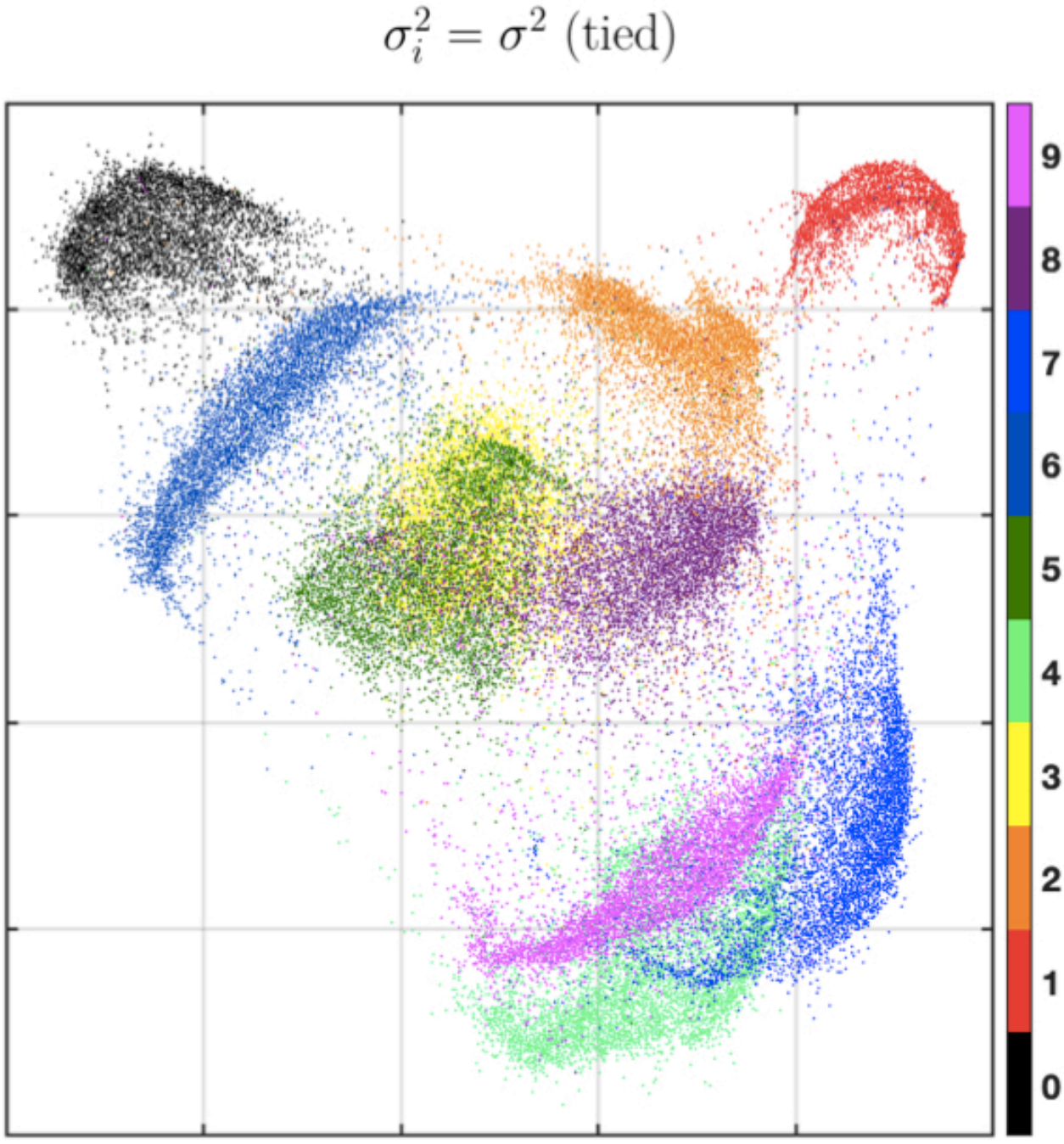

Fig. S4. Two dimensional embedding of 70K handwritten digits ($k=9, s=1, \ell=1$) when each output $\boldsymbol{\mu}_i$ is constrained to have the same variance $\sigma_i^2 = \sigma^2$. The resulting embedding does not separate the most similar classes of digits; it also has a considerably lower likelihood.

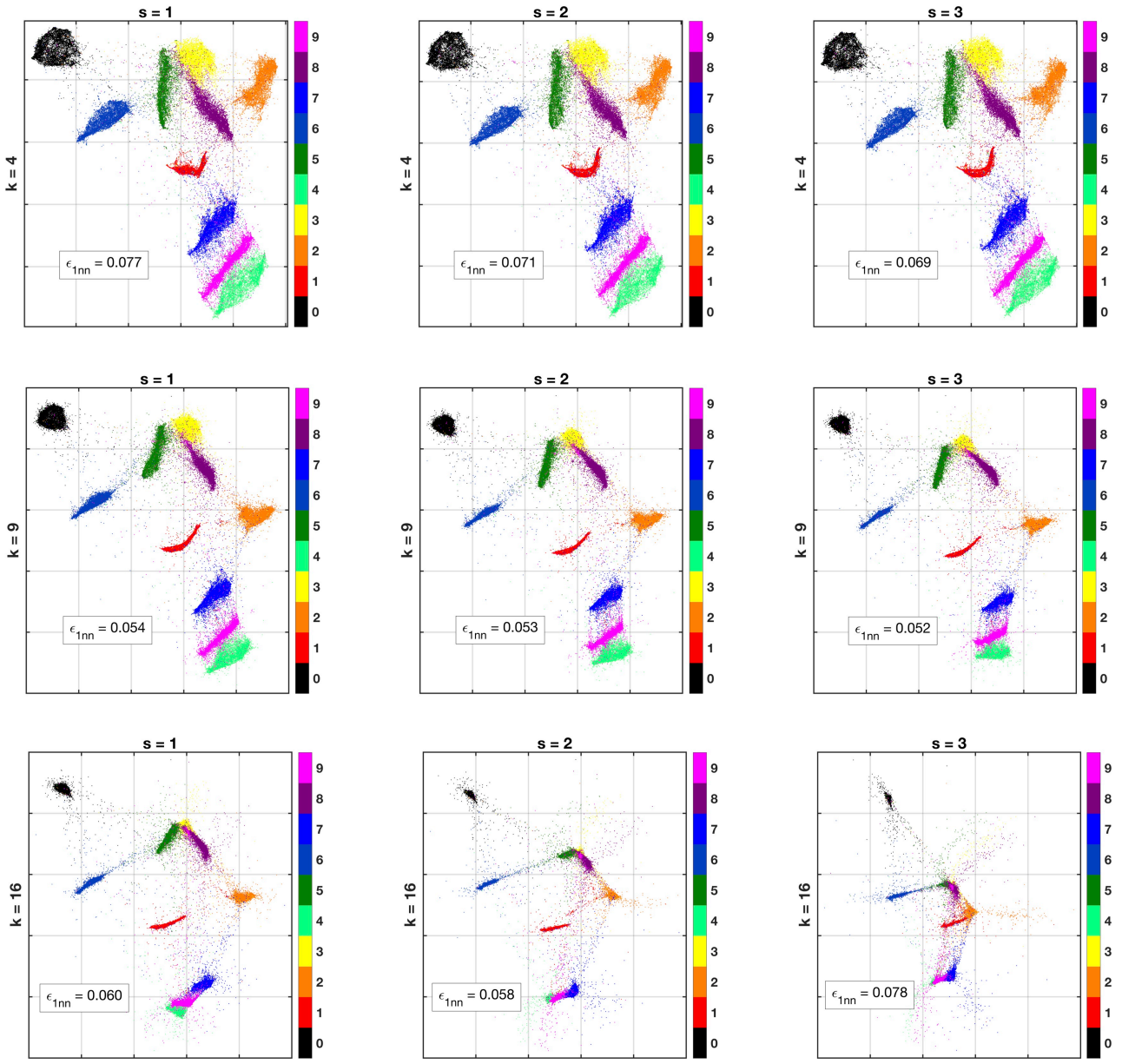

Fig. S5. Embeddings of MNIST digits with different values of k and s. Also shown is the 1-nearest-neighbor error on the MNIST test set in the latent space of lower dimensionality.

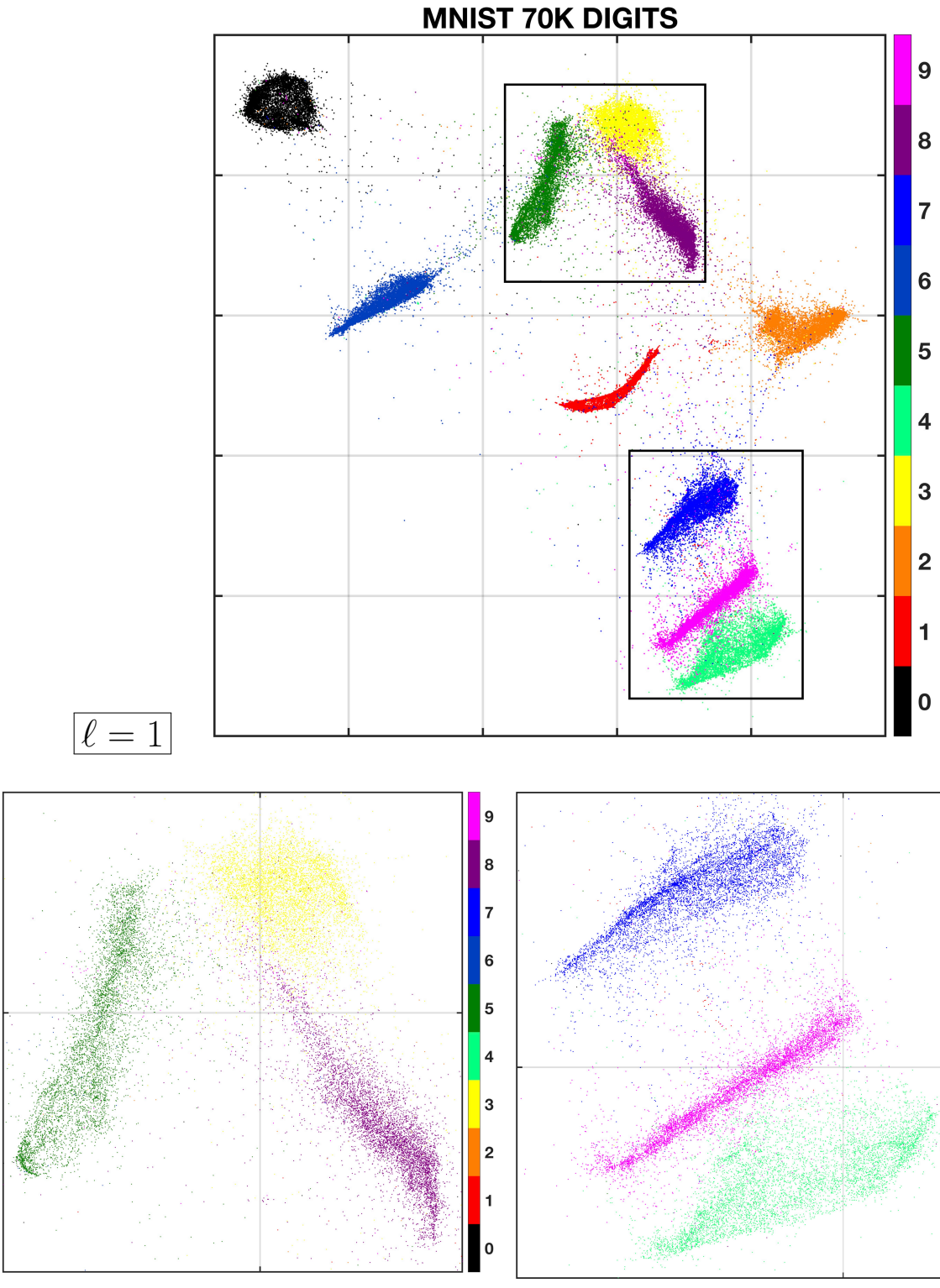

Fig. S6. *Top:* two dimensional visualization of $n = 70000$ MNIST handwritten digits by the LVM $(k = 9, s = 1, \ell = 1)$ with boxes around the most confusable classes of digits. The sheer number of points in these plots can produce some visual artifacts when viewing the visualization at larger scales. *Left:* closeup of the top cluster of threes, fives, and eights. *Right:* closeup of the bottom cluster of fours, sevens, and nines. Compare to the next figure to see the effect of further coarse-graining.

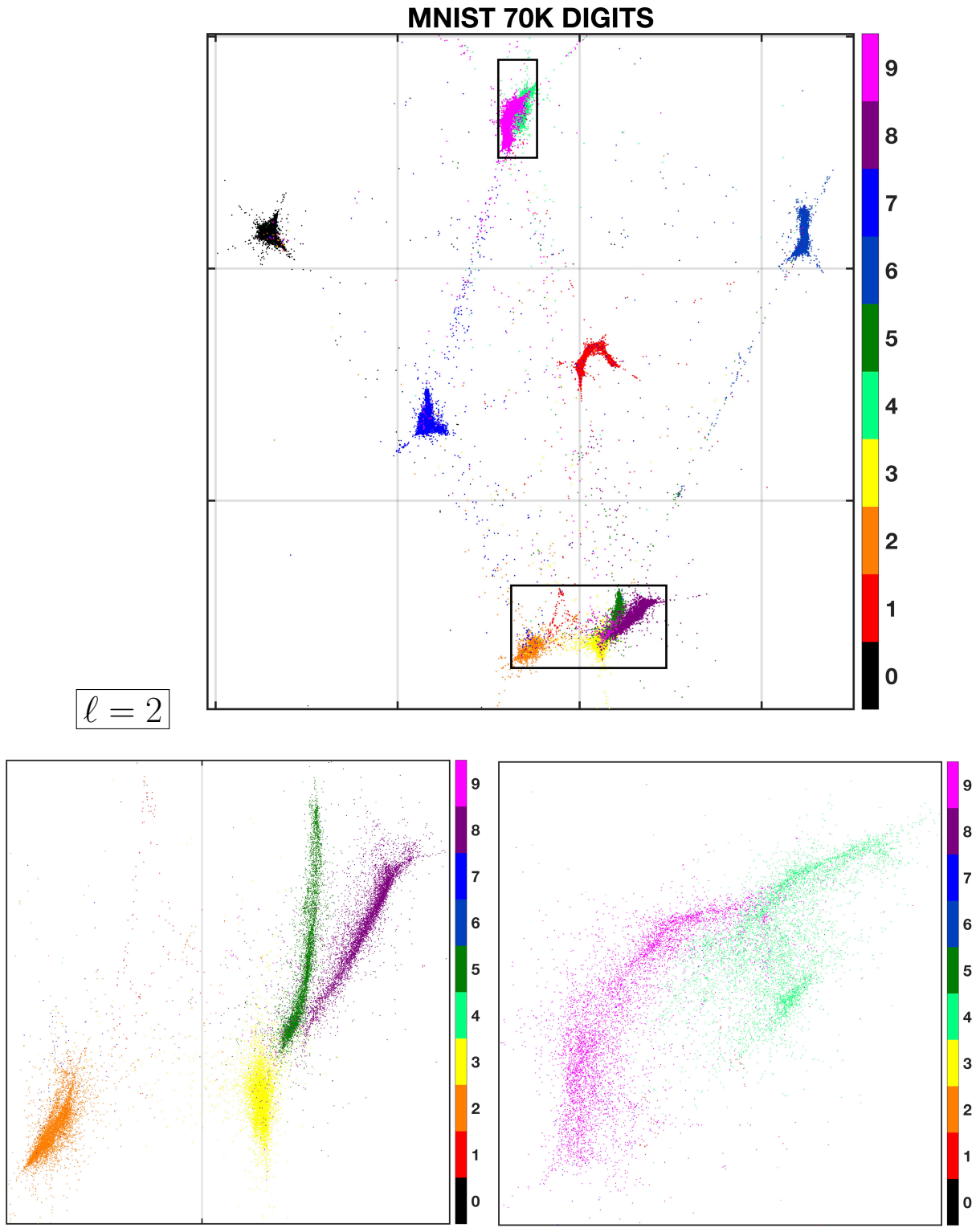

Fig. S7. *Top:* two dimensional visualization of $n = 70000$ MNIST handwritten digits by the LVM $(k = 9, s = 1, \ell = 2)$ with boxes around the most confusable classes of digits. The sheer number of points in these plots can produce some visual artifacts when viewing the visualization at larger scales. *Left:* closeup of the bottom cluster of twos, threes, fives, and eights. *Right:* closeup of the top cluster of fours and nines. This figure illustrates one side-effect of coarse-graining: distances between clusters are magnified.

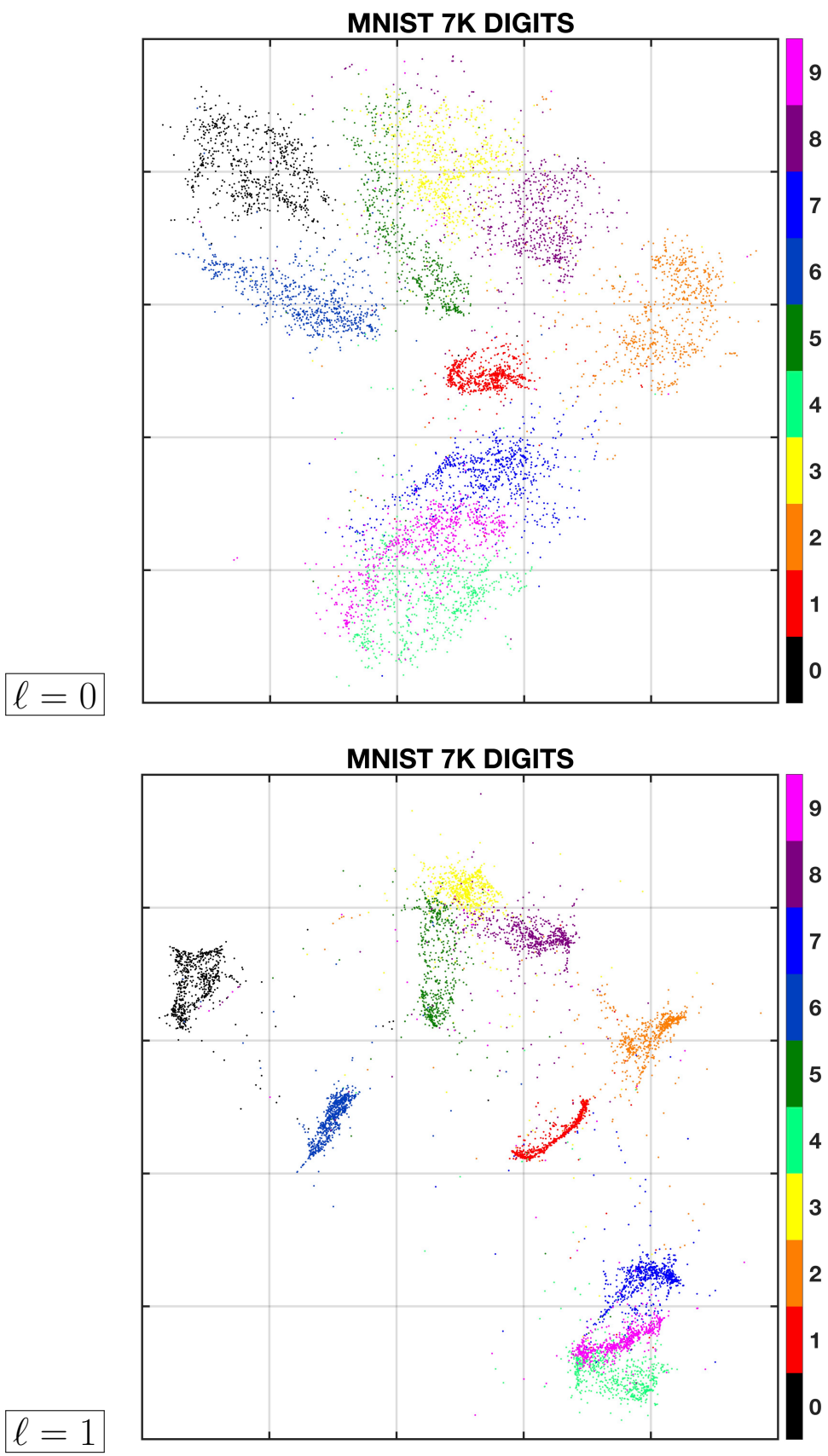

Fig. S8. Two dimensional visualizations of $n = 7000$ MNIST handwritten digits by LVMs ($k = 9$, $s = 1$) with different levels of coarse-graining ($\ell = 0$ versus $\ell = 1$). As in the previous figures, we observe that the coarse-graining tends to magnify the distance between clusters.

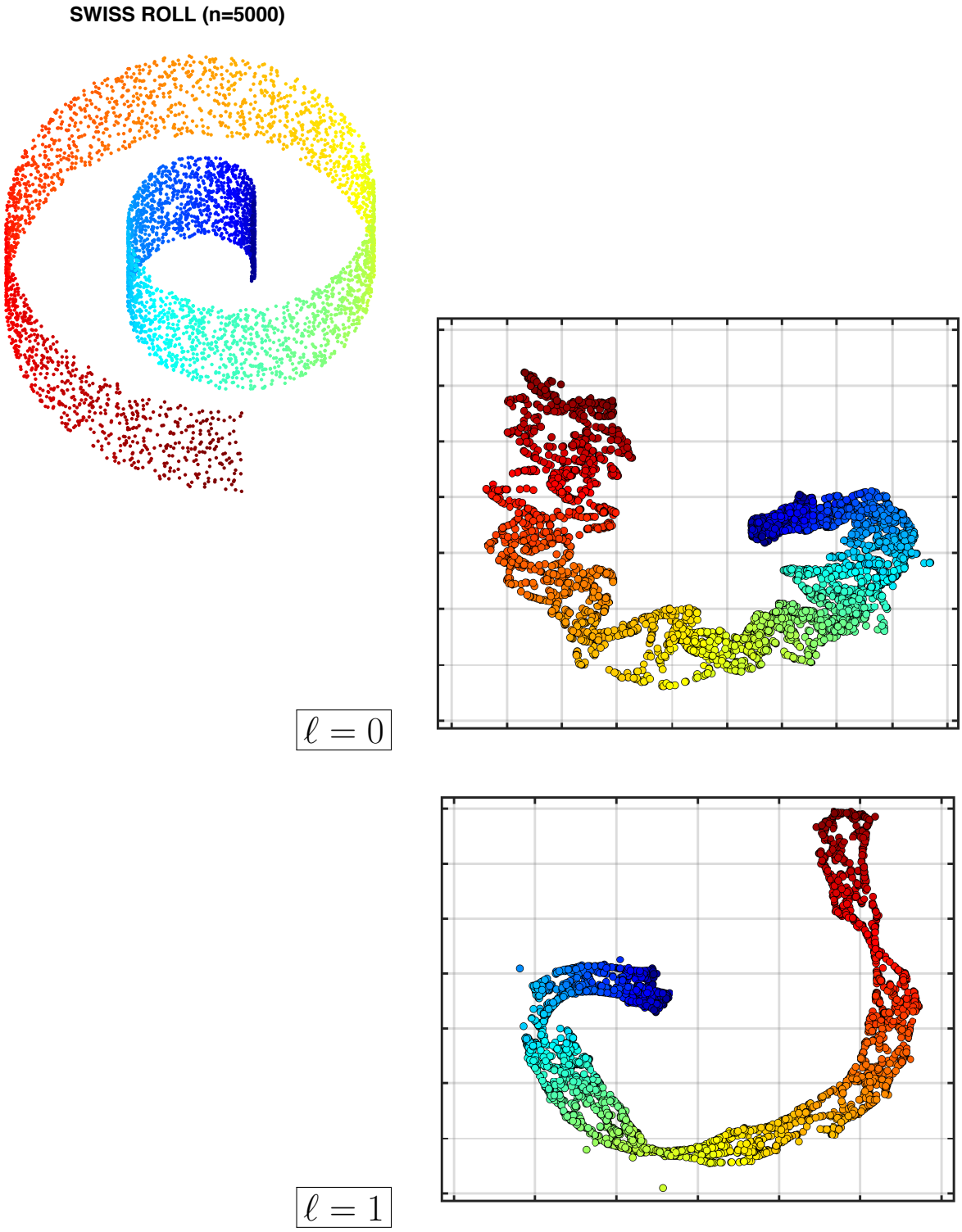

Fig. S9. Two dimensional visualizations of the Swiss roll data set ($n = 5000$) by LVMs ($k = 9, s = 1$) with different levels of coarse-graining ($\ell = 0$ versus $\ell = 1$).

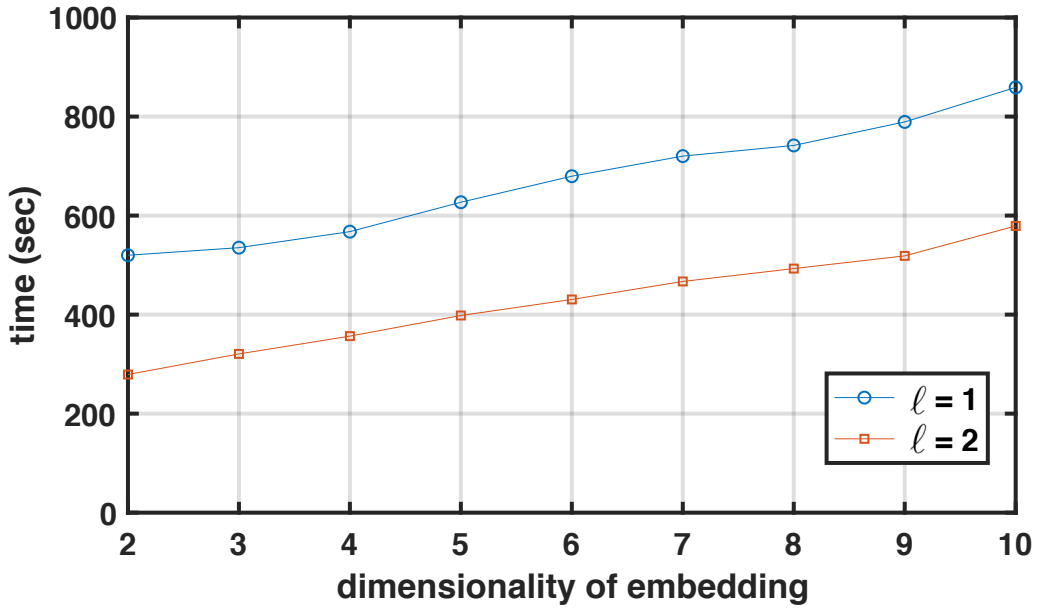

Fig. S10. Time in seconds to complete 400 iterations of EM versus the dimensionality of the embedding for the LVMs shown in Figure 3.

Log conditional likelihood:

$$\mathcal{L} = \sum_{ij} \mathcal{S}_{ij} \log P(c_\delta = 1 | \mathbf{x}_i, \mathbf{x}_j) + \sum_{ij} \mathcal{D}_{ij} \log P(c_\Delta = 0 | \mathbf{x}_i, \mathbf{x}_j)$$

$$P(c_\delta = 1 | \mathbf{x}_i, \mathbf{x}_j) = \left(\frac{\delta_{ij}^2}{\delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right)^{\frac{d}{2}} \exp\left\{ -\frac{1}{2} \left(\frac{1}{\delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right) \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2 \right\}$$

$$P(c_\Delta = 0 | \mathbf{x}_i, \mathbf{x}_j) = 1 - \left(\frac{\Delta_{ij}^2}{\Delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right)^{\frac{d}{2}} \exp\left\{ -\frac{1}{2} \left(\frac{1}{\Delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right) \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2 \right\}$$

Inference (E-step) for pairs of similar inputs:

$$\mathrm{E}[\mathbf{h} | c_\delta = 1, \mathbf{x}_i, \mathbf{x}_j] = \boldsymbol{\mu}_i - \left(\frac{\sigma_i^2}{\delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right) (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)$$

$$\mathrm{E}[\mathbf{h}' | c_\delta = 1, \mathbf{x}_i, \mathbf{x}_j] = \boldsymbol{\mu}_j - \left(\frac{\sigma_j^2}{\delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right) (\boldsymbol{\mu}_j - \boldsymbol{\mu}_i)$$

$$\mathrm{E}\left[\left\| \mathbf{h} - \boldsymbol{\mu}_i \right\|^2 \middle| c_\delta = 1, \mathbf{x}_i, \mathbf{x}_j \right] = d\sigma_i^2 + \sigma_i^4 \left(\frac{1}{\delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right) \left(\frac{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2}{\delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} - d \right).$$

$$\mathrm{E}\left[\left\| \mathbf{h}' - \boldsymbol{\mu}_j \right\|^2 \middle| c_\delta = 1, \mathbf{x}_i, \mathbf{x}_j \right] = d\sigma_j^2 + \sigma_j^4 \left(\frac{1}{\delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right) \left(\frac{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2}{\delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} - d \right).$$

Inference (E-step) for pairs of dissimilar inputs:

$$\mathrm{E}[\mathbf{h} | c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j] = \boldsymbol{\mu}_i + \nu_{ij} \left(\frac{\sigma_i^2}{\Delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right) (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)$$

$$\mathrm{E}[\mathbf{h}' | c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j] = \boldsymbol{\mu}_j + \nu_{ij} \left(\frac{\sigma_j^2}{\Delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right) (\boldsymbol{\mu}_j - \boldsymbol{\mu}_i)$$

$$\mathrm{E}\left[\left\| \mathbf{h} - \boldsymbol{\mu}_i \right\|^2 \middle| c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j \right] = d\sigma_i^2 - \nu_{ij}\sigma_i^4 \left(\frac{1}{\Delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right) \left(\frac{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2}{\Delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} - d \right).$$

$$\mathrm{E}\left[\left\| \mathbf{h}' - \boldsymbol{\mu}_j \right\|^2 \middle| c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j \right] = d\sigma_j^2 - \nu_{ij}\sigma_j^4 \left(\frac{1}{\Delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} \right) \left(\frac{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2}{\Delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} - d \right).$$

Update (M-step) for outputs:

$$W_{ij} = \frac{\mathcal{S}_{ij}}{\delta_{ij}^2 + \sigma_i^2 + \sigma_j^2} + \frac{\mathcal{S}_{ji}}{\delta_{ji}^2 + \sigma_i^2 + \sigma_j^2},$$

$$L_{ij} = \begin{cases} \sum_k W_{ik} + \frac{1}{\sigma_i^2} \sum_k (\mathcal{D}_{ik} + \mathcal{D}_{ki}) & \text{if } i = j, \\ -W_{ij} & \text{otherwise.} \end{cases}$$

$$\sum\nolimits_j L_{ij} \boldsymbol{\mu}_j^{\text{new}} = \frac{1}{\sigma_i^2} \sum_j \Big(\mathcal{D}_{ij} \mathrm{E}[\mathbf{h} | c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j] + \mathcal{D}_{ji} \mathrm{E}[\mathbf{h}' | c_\Delta = 0, \mathbf{x}_j, \mathbf{x}_i] \Big).$$

Update (M-step) for variances:

$$\begin{aligned} \phi_{ij} &= \mathrm{E}\big[\|\mathbf{h} - \boldsymbol{\mu}_i\|^2 \,|\, c_\delta = 1, \mathbf{x}_i, \mathbf{x}_j \big] \\ \phi'_{ij} &= \mathrm{E}\big[\|\mathbf{h}' - \boldsymbol{\mu}_j\|^2 \,|\, c_\delta = 1, \mathbf{x}_i, \mathbf{x}_j \big] \\ \psi_{ij} &= \mathrm{E}\big[\|\mathbf{h} - \boldsymbol{\mu}_i\|^2 \,|\, c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j \big] \\ \psi'_{ij} &= \mathrm{E}\big[\|\mathbf{h}' - \boldsymbol{\mu}_j\|^2 \,|\, c_\Delta = 0, \mathbf{x}_i, \mathbf{x}_j \big] \end{aligned}$$

$$(\sigma_i^2)^{\text{new}} = \frac{1}{d} \frac{\sum_j \left(\mathcal{S}_{ij} \phi_{ij} + \mathcal{S}_{ji} \phi'_{ji} + \mathcal{D}_{ij} \psi_{ij} + \mathcal{D}_{ji} \psi'_{ji} \right)}{\sum_j \left(\mathcal{S}_{ij} + \mathcal{S}_{ji} + \mathcal{D}_{ij} + \mathcal{D}_{ji} \right)}$$

Table S1. Summary of the EM algorithm.

Table S2. Four nearest neighbors of word vectors before and after the LVM's embedding $(k=4, s=3, \ell=2)$ into ten dimensions. The words (including capitalization) are from the preamble of the U.S. Constitution.

word	4nn (before)	4nn (after)
People	Folks, Individuals, Teenagers, people	Folks, folks, people, peeps
United_States	U.S., America, Europe, countries	U.S., aUS, American, America
Order	Orders, Edicts, Decree, order	Orders, Ordering, orders, ordering
form	forms, shape, forming, formation	forms, Forms, ray, silver_lining
more	less, More, fewer, rather	less, Fewer, Less, fewer
perfect	ideal, terrific, flawless, fabulous	ideal, Perfect, Ideal, Suited
Union	Federation, union, Confederation, Association	Federation, OCA, governing_body, association
establish	establishing, reestablish, established, develop	forge, forging, Forging, forges
insure	insuring, insures, insured, Insuring	insuring, insures, insured, Insuring
domestic	Domestic, domestically, DOMESTIC, overseas	Domestic, domestically, Domestically, Internationally
Tranquility	Tranquillity, Atlantis, Cupola, Endeavour	Tranquillity, Atlantis, Discovery, Endeavour
provide	providing, provides, provided, Providing	providing, Providing, offered, offer
common	Common, prevalent, commonest, ordinary	Common, Ordinary, ordinary, harmfulness
defense	defenses, defensive, defensively, offense	defenses, Defenses, Defences, defensive
promote	promoting, promotes, encourage, Promoting	promotes, Involve, promoting, encourages
Welfare	Social_Welfare, welfare, Invalids, Social_Affairs	Social_Welfare, Social_Affairs, Fruit_Fly, Fruit_Flies
secure	securing, secured, secures, Securing	inapplicable, securing, prescribes, prescribed
Blessings	blessings, Mercies, Thankfulness, Beatitude	blessings, bless, blessing, blesses
Liberty	LIberty, Freedom, Patriot, Independence	LIberty, Freedom, Independence, freedom
Posterity	Wherefore, Conceit, posterity, Absolutism	posterity, Wherefore, Founding_fathers, WHEREFORE
ordain	ordained, ordination, ordaining, ordains	ordained, ordaining, ordinations, Ordained
Constitution	constitution, constitutional, Constitutions, Constitutional	constitution, Constitutionally, instrumentalities, constitutionally
America	United_States, American, Europe, nation	American, U.S., United_States, aUS

References

1. Wilson KG (1983) The renormalization group and critical phenomena. *Reviews of Modern Physics* 55(3):583–600.
2. Mehta P, Schwab DJ (2014) An exact mapping between the variational renormalization group and deep learning. ArXiv e-prints 1410.3831.
3. Lin HW, Tegmark M, Rolnick D (2017) Why does deep and cheap learning work so well? *Journal of Statistical Physics* 168(6):1223–1247.
4. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representations by error propagation in *Parallel Distributed Processing*, eds. Rumelhart DE, McClelland JL. (MIT Press) Vol. 1, pp. 318–362.
5. Qian N (1999) On the momentum term in gradient descent learning algorithms. *Neural Networks* 12:145–151.
6. Jamshidian M, Jennrich RI (1993) Conjugate gradient acceleration of the EM algorithm. *Journal of the American Statistical Association* 88:221–228.
7. Liu C, Rubin DB (1994) The ECME algorithm: A simple extension of EM and ECM with faster monotone convergence. *Biometrika* 81:633–648.
8. Lange KL (1995) A quasi-Newton acceleration of the EM algorithm. *Statistica Sinica* 5:1–18.
9. Jamshidian M, Jennrich RI (1997) Acceleration of the EM algorithm by using quasi-Newton methods. *Journal of the Royal Statistical Society, Series B* 59:569–587.
10. Liu C, Rubin DB, Wu YN (1998) Parameter expansion to accelerate EM: The PX-EM algorithm. *Biometrika* 85:755–770.
11. Salakhutdinov RR, Roweis ST, Ghahramani Z (2003) Optimization with EM and expectation-conjugate-gradient in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. pp. 672–679.
12. Varadhan R, Roland C (2008) Simple and globally convergent methods for accelerating the convergence of any EM algorithm. *Scandinavian Journal of Statistics* 35:335–353.
13. Yu Y (2012) Monotonically overrelaxed EM algorithms. *Journal of Computational and Graphical Statistics* 21:518–537.
14. Henderson NC, Varadhan R (2019) Damped anderson acceleration with restarts and monotonicity control for accelerating EM and EM-like algorithms. *Journal of Computational and Graphical Statistics* 28(4):834–846.