EFFICIENT REPRESENTATION OF LOW-DIMENSIONAL MANIFOLDS USING DEEP NETWORKS

Ronen Basri
Dept. of Computer Science and Applied Math
Weizmann Institute of Science
Rehovot, 76100 Israel
ronen.basri@weizmann.co.il

David W. Jacobs
Dept. of Computer Science
University of Maryland
College Park, MD
djacobs@cs.umd.edu

ABSTRACT

We consider the ability of deep neural networks to represent data that lies near a low-dimensional manifold in a high-dimensional space. We show that deep networks can efficiently extract the intrinsic, low-dimensional coordinates of such data. Specifically we show that the first two layers of a deep network can exactly embed points lying on a *monotonic chain*, a special type of piecewise linear manifold, mapping them to a low-dimensional Euclidean space. Remarkably, the network can do this using an almost optimal number of parameters. We also show that this network projects nearby points onto the manifold and then embeds them with little error. Experiments demonstrate that training with stochastic gradient descent can indeed find efficient representations similar to the one presented in this paper.

1 INTRODUCTION

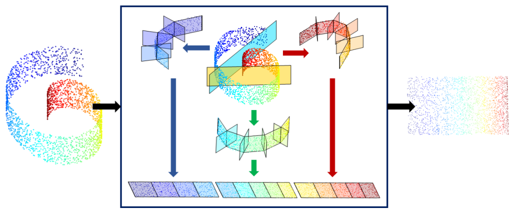

Figure 1: We illustrate the embedding of a manifold by a deep network using the famous Swiss Roll example (left). Dots represent color coded input data, with color indicating one of the intrinsic coordinates of each input point. In the center, the data is divided into three parts using hidden units represented by the yellow and cyan planes. Each part is then approximated by a monotonic chain of linear segments. Additional hidden units, also depicted as planes, control the orientation of the next segments in the chain. A second layer of the network then flattens each chain into a 2D Euclidean plane, and assembles these into a common 2D representation (right).

Deep neural networks have achieved state-of-the-art results in a variety of tasks. One possible reason for this remarkable success is that their hierarchical, layered structure may allow them to capture the geometric regularities of commonplace data. We support this hypothesis by exploring ways that networks can handle input data that lie on or near a low-dimenisonal manifold. In many problems, for example face recognition, data lie on or near manifolds that are of much lower dimension than the input space (Turk & Pentland, 1991; Basri & Jacobs, 2003; Lee et al., 2003), and that represent the intrinsic degrees of variation in the data.

We study the ability of deep networks to represent manifold data. We show that the initial layers of networks can approximate data that lies on high-dimensional manifolds using piecewise linear functions, and economically output their coordinates embedded in a low-dimensional Euclidean space. In fact, each new linear segment approximating the manifold can be represented by a single additional hidden unit, leading to a representation of manifold data that in some cases is nearly optimal in the number of parameters of the system. Subsequent layers of a deep network could build upon these early layers, operating in lower dimensional spaces that more naturally represent the input data. We further show empirical results that suggest that training with stochastic gradient descent can find efficient representations akin to the one suggested in this paper.

We first show how this embedding can be done efficiently for manifolds consisting of *monotonic chains* of linear segments. We then show how these primitives can be combined to form linear approximations for more complex manifolds. This process is illustrated in Figure 1. We further show that when the data lies sufficiently close to their linear approximation, the error in the embedding will be small. Our constructions will use a feed-forward network with rectified linear unit (RELU) activation. We consider fully connected layers, although the treatment of complex manifolds that are divided into pieces (e.g., of monotonic chains) will be modular, resulting in many zero weights.

2 Prior Work

Realistic learning problems, e.g., in vision and speech processing, involve high dimensional data. Such data is often governed by many fewer variables, producing manifold-like sub-structures in a high dimensional ambient space. A large number of dimensionality reduction techniques, such as principle component analysis and multi-dimensional scaling (Duda et al., 2012), Isomap (Tenenbaum et al., 2000), and local linear embedding (LLE) (Roweis & Saul, 2000), have been introduced. An underlying *manifold assumption*, which states that different classes lie in separate manifolds, has also guided the design of clustering and semi-supervised learning algorithms (Nadler et al., 2005; Belkin & Niyogi, 2003; Weston et al., 2008; Mobahi et al., 2009).

A number of recent papers examine properties of neural nets in light of this manifold assumption. Brahma et al. (2015) show empirically that the layers of deep networks trained with data that lies on a manifold progressively unfold that data into Euclidean spaces. They do not consider the mechanisms used to perform this unfolding. Rifai et al. (2011) trained a contractive auto-encoder to represent an atlas of manifold charts. Shaham et al. (2015) demonstrate that a 4-layer network can efficiently represent any function on a manifold through a trapezoidal wavelet decomposition. In both, each chart is represented independently, requiring an independent projection for each chart. Likewise, (Chui & Mhaskar, 2016) consider methods by which a neural network can map points on a manifold to a low-dimensional, Euclidean space, although they do not consider the efficiency of this representation in terms of hidden units or weights. We show that for monotonic chains we can reduce the size of the representation to near optimal by exploiting geometric relations between neighboring projection matrices, so an additional chart requires only a single hidden unit.

Another family of networks attempt to learn a “semantic” distance metric for training pairs, often by using a siamese network (Salakhutdinov & Hinton, 2007; Chopra et al., 2005; R. Hadsell & LeCun, 2006; Yi et al., 2014; Huang et al., 2015). These assume that the input space can be mapped non-linearly by a network to produce the desired distances in a lower dimensional feature space. Giryes et al. (2016) shows that even a feed-forward neural network with random Gaussian weights embeds the input data in an output space while preserving distances between input items.

Another outstanding question is to what extent deep networks can be more efficient than shallow networks with a single hidden layer. Shallow networks are universal approximators (Cybenko, 1989). However, recent work demonstrates that deep networks can be exponentially more efficient in representing certain functions (Bianchini & Scarselli, 2014; Telgarsky, 2015; Eldan & Shamir, 2015; Delalleau & Bengio, 2011; Montufar et al., 2014; Cohen et al., 2015). On the other hand, (Ba & Caruana, 2014) shows empirically that in many practical cases a shallow network can be trained to mimic the behavior of a deep network. Our construction does not produce exponential gains, but does show that the early layers of a network can efficiently reduce the dimensionality of data that feeds into later layers.

3 Monotonic Chains of Linear Segments

We construct networks that perform dimensionality reduction on data that lies on or near a manifold. We focus on feed-forward networks with RELU activation, i.e., $\max(x, 0)$. Clearly the output of such networks are continuous, piecewise linear functions of their input. It is therefore natural to ask whether they can embed piecewise-linear manifolds in a low-dimensional Euclidean space both ac-

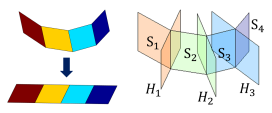

Figure 2: Left: A continuous chain of linear segments (above) that can be flattened to lie in a single low-dimensional linear subspace (bottom). Right: A monotonic chain. S_k denotes the k'th segment in the chain. H_k is a hyperplane bounding the half-space that separates $S_1, ..., S_k$ from $S_{k+1}, ..., S_K$.

curately and efficiently. In this section we construct such efficient networks for a class of manifolds that we call *monotonic chains of linear segments*, which are defined shortly. These will serve as building blocks for handling more general data that can be decomposed into monotonic chains.

We will consider data lying in a chain of linear segments, denoted $\mathcal{C} = S_1 \cup ... \cup S_K$. Each segment S_k ($1 \leq k \leq K$) in the chain is a portion of some m-dimensional affine subspace of $\mathbb{R}^d$, and the segments are connected to form a chain (Figure 2). We suppose that every two consecutive segments S_{k-1} and S_k intersect, and that the intersection lies in an $(m-1)$-dimensional affine subspace. We further assume that these chains can be flattened by isometry so that they may be represented in $\mathbb{R}^m$. Note that any curve on $\mathcal{C}$ will be mapped to a curve of the same length in $\mathbb{R}^m$ on the flattened chain.

Each unit in the first hidden layer of a neural network will have a response of zero to input points that lie on a hyperplane, defined by its weights and bias term. This hyperplane bounds a half-space in which the output of the unit is positive; when the output is negative, RELU turns the output to zero. We say that a unit is *active* over the half-space in which its output is positive. There is a close connection between these hyperplanes and the embedding of a manifold, which we begin to develop with the following definition.

Definition: We say that a chain of K linear segments is *monotonic* (see Figure 2) when there exist a set of hyperplanes such that the k'th hyperplane separates the first k segments from the rest. Denoting the positive half-spaces associated with these hyperplanes as $H_1, H_2, ..., H_{K-1}$, then H_k is bounded by a hyperplane that contains the intersection of S_k and S_{k+1}, and $S_{k+1}, S_{k+2}, ..., S_K \subset H_k$ while $S_1, S_2, ..., S_k \subset H_k^C$, where H_k^C is the complement of H_k. We can consider each half-space to represent a hidden unit that is *active* (i.e., non-zero) over a subset of the regions. With a monotonic chain, the set of active units grows monotonically, so that, $(H_{k+1} \cap \mathcal{C}) \subseteq (H_k \cap \mathcal{C})$. We can also define some additional units that are active over all the regions.

Below we show that monotonic chains can be embedded efficiently by networks with two layers of weights. These networks have d units in the input layer, a hidden layer with $\kappa = K + m - 1$ units that encodes the structure of the manifold, and an output layer with m units. Denote the weights in the first layer by a $\kappa \times d$ matrix A and further use a bias vector $\mathbf{a}_0 \in \mathbb{R}^\kappa$. The second layer of weights is captured by a $m \times \kappa$ matrix B. The total number of weights in these two layers is $(d + m + 1)(K + m - 1)$. This network maps a point $\mathbf{x} \in \mathbb{R}^d$ to the embedding space $\mathbb{R}^m$ through

$$\mathbf{u} = B[A\mathbf{x} + \mathbf{a}_0]_+$$

where $[.]_+$ denotes the RELU operation. For now we do not use a bias or RELU in the second level, but those will be used later when we discuss more complex manifolds.

A simple example of a manifold that can be represented efficiently with a neural network occurs when the data lies in a single m-dimensional affine subspace of $\mathbb{R}^d$. Embedding can be done in this case with just one layer, with the matrix A of size $m \times d$ containing in its rows a basis parallel to the affine space. One way to extend this example to handle chains is by encoding each linear segment separately. Such encoding will require mK units in addition to units that use RELU to separate each segment from the rest of the segments. A related representation was used, e.g., in (Shaham et al., 2015). Below we show that monotonic chains can be encoded much more efficiently.

We next show how to construct the network (i.e., set the weights in A, $\mathbf{a}_0$, and B) to encode monotonic chains. Below we use the notation $A^{(k)}$ to denote the matrix formed by the first k *rows* of A, $\mathbf{a}_0{}^{(k)}$ is the vector containing the first k entries of $\mathbf{a}_0$, and $B^{(k)}$ the matrix including the first k *columns* of B. Therefore $B^{(k)}[A^{(k)}\mathbf{x} + \mathbf{a}_0{}^{(k)}]_+$ will express the output of the network when only the first k hidden units are used. These will be set to recover the intrinsic coordinates of points in the first k segments in $\mathcal{C}$; RELU ensures that subsequent hidden units do not affect the output for points in these segments.

For the construction we consider the pull-back of the standard basis of $\mathbb{R}^m$ onto the chain, producing a geodesic basis to the manifold. Note that to produce a local basis for the intrinsic coordinates of points on the manifold, we only need a basis for each linear segment. This basis is expressed by a collection of $d \times m$ column-orthogonal matrices $X^{(1)}, X^{(2)}, ..., X^{(K)}$. Each matrix provides an orthogonal basis for one of the segments.

We will construct the network inductively. Suppose $k = 1$. We set $A^{(1)} = X^{(1)\,T}$, $B^{(1)} = I$, and set $\mathbf{a}_0{}^{(1)}$ so that for all $\mathbf{x} \in \mathcal{C}$ all the components of $A^{(1)}\mathbf{x} + \mathbf{a}_0{}^{(1)}$ are non-negative. Clearly, $B^{(1)}A^{(1)} = X^{(1)\,T}$ is an orthogonal projection matrix and $B^{(1)}A^{(1)}X^{(1)} = I$. This shows that the network projects the orthonormal basis for the first segment into I, an orthonormal basis in $\mathcal{R}^m$. Next we will show that $B^{(k)}A^{(k)}X^{(k)} = I$ for all k. This implies that $B^{(k)}A^{(k)}x = X^{(k)\,T}x$, so there is no distortion in the projection. This will show that the network extends this basis throughout the monotonic chain in a consistent way.

Suppose we used $m + k - 2$ units to construct $A^{(k-1)}$, $\mathbf{a}_0{}^{(k-1)}$, and $B^{(k-1)}$ for the first $k - 1 \geq 1$ segments. (For notational convenience we will next omit the superscript $k - 1$ for these matrices and vectors, so $A = A^{(k-1)}$, etc.) We will now use those to construct $A^{(k)}$, $\mathbf{a}_0{}^{(k)}$, and $B^{(k)}$. We do so by adding a node to the first hidden layer. The weights on the incoming edges to this node will be encoded by appending a row vector $\mathbf{a}^T \in \mathbb{R}^d$ to A and a scalar a_0 to $\mathbf{a}_0$, and the weights on the outgoing edges will be encoded by appending a column vector $\mathbf{b} \in \mathbb{R}^m$ to B. Our aim is to assign values to these vectors and scalar to extend the embedding to S_k.

By induction we assume that any $\tilde{\mathbf{x}} \in S_1 \cup ... \cup S_{k-1}$ is embedded with no distortion to $\mathbb{R}^m$ by

$$\tilde{\mathbf{u}} = B[A\tilde{\mathbf{x}} + \mathbf{a}_0]_+,$$

and that $BAX = I$. By monotonicity we further assume that $S_{k-1} \cap S_k$ is $m - 1$ dimensional and there exists a hyperplane H with normal $\mathbf{h} \in \mathbb{R}^d$ that contains this intersection with $C - (S_1 \cup ... \cup S_{k-1})$ lying completely on the side of H in the direction of $\mathbf{h}$, while $S_1 \cup ... \cup S_{k-1}$ lies on the opposite side of H. We then set $\mathbf{a} = \mathbf{h}$ and set a_0 so that $\mathbf{a}^T\bar{\mathbf{x}} + a_0 = 0$ for any point $\bar{x} \in S_{k-1} \cap S_k$. (This is well defined since $\mathbf{h}$ is orthogonal to $S_{k-1} \cap S_k$.)

To determine $\mathbf{b}$, we first rotate the bases $X^{(k-1)}$ (referred to as X below) and $X^{(k)}$ by a common, $m \times m$ matrix R, i.e., $Y = XR$ and $Y^{(k)} = X^{(k)}R$ so that $Y = [\mathbf{w}, \mathbf{y}_2, ..., \mathbf{y}_m]$ and $Y^{(k)} = [\mathbf{v}, \mathbf{y}_2, ..., \mathbf{y}_m]$ with $\mathbf{y}_2, ..., \mathbf{y}_m$ providing an orthogonal basis parallel to $S_{k-1} \cap S_k$. (This is equivalent to rotating the coordinate system in the embedded space and then pulling-back to the manifold.) Note that by the induction assumption $BAYR^T = I$. We next aim to set $\mathbf{b}$ so that $B^{(k)}A^{(k)}X^{(k)} = I$. We note that

$$B^{(k)}A^{(k)}X^{(k)} = B^{(k)}A^{(k)}Y^{(k)}R^T = (BA + \mathbf{b}\mathbf{a}^T)Y^{(k)}R^T.$$

We aim to set $\mathbf{b}$ so that $(BA + \mathbf{b}\mathbf{a}^T)Y^{(k)}R^T = I = BAYR^T$. Consider this equality first for the common columns $\mathbf{y}_2, ..., \mathbf{y}_m$ of Y and $Y^{(k)}$. These columns are parallel to $S_{k-1} \cap S_k$, so that $\mathbf{a}^T\mathbf{y}_j = 0$ for $2 \leq j \leq m$, implying equality for any choice of $\mathbf{b}$. Consider next the left-most column of Y and $Y^{(k)}$, denoted respectively $\mathbf{w}$ and $\mathbf{v}$, we get

$$(BA + \mathbf{b}\mathbf{a}^T)\mathbf{v} = BA\mathbf{w}.$$

This is satisfied if we set

$$\mathbf{b} = \frac{1}{\mathbf{a}^T\mathbf{v}}BA(\mathbf{w} - \mathbf{v}).$$

We have constructed $\mathbf{b}$ so that the segments are embedded with consistent orientations. In Appendix A we show that they are also translated properly by $\mathbf{a}_0$, to create a continuous embedding. Note that by construction $\mathbf{a}^T\mathbf{y} + a_0 \leq 0$ for all $\mathbf{y} \in S_1 \cup ... \cup S_{k-1}$ so RELU ensures that the embedding of the these segments will not be affected by the additional unit.

Finally, we note that the proposed representation of monotonic chains with a neural network is very efficient and uses only a few parameters beyond the degrees of freedom needed to define such chains. In particular, the definition of a chain requires specifying m basis vectors in $\mathbb{R}^d$ for one linear segment (exploiting orthonormality these require $m(d - (m + 1)/2)$ parameters), with each additional segment specified by a 1D direction for the new segment (a unit vector in R^d specified by $d - m - 1$ parameters) and a direction in the previous segment to be replaced (specified by a unit vector in $\mathbb{R}^m$, i.e. $m - 1$ parameters). The total number of degrees of freedom of a chain is therefore $N = m(d - (m + 1)/2) + (K - 1)(d - 2)$. This is the number of parameters required to specify a monotonic chain. Our construction requires $N' = (K + m + 1)(d + m + 1)$ parameters. Specifically, note that for any choice of parameters $K, d, m > 0$, $N \geq (K + m - 1)(d - m - 2)$. We therefore obtain that

$$\frac{N'}{N} \leq \left(1 + \frac{2}{K + m - 1}\right)\left(1 + \frac{2m + 3}{d - m - 2}\right).$$

Assuming $d, K + m >> 1$ we get

$$\frac{N'}{N} \lessapprox 1 + \frac{2m}{d - m}.$$

Since we normally expect that the dimension of the input space will be much greater than the dimension of the manifold, this ratio will be close to 1.

4 Error Analysis

We now consider points that do not lie exactly on the monotonic chain, due to noise, or because we are approximating a non-linear manifold with piece-wise linear segments. Let $\mathbf{p}_0$ be a point on the segment S_j that is then perturbed by some small noise vector, δ, that is perpendicular to S_j, to produce the point $\mathbf{p} = \mathbf{p}_0 + \delta$. Ideally, the network would represent $\mathbf{p}$ using the coordinates of $\mathbf{p}_0$. In effect, the network would project all points onto the monotonic chain. If the network embeds $\mathbf{p}$ and $\mathbf{p}_0$ with coordinates $\hat{\mathbf{p}}$ and $\hat{\mathbf{p}}_0$ we define the *relative error* of the embedding as $\frac{\|\hat{\mathbf{p}} - \hat{\mathbf{p}_0}\|}{\delta}$. We now analyze this relative error. Our analysis assumes that $\|\delta\|$ is small enough that $\mathbf{p}$ and $\mathbf{p}_0$ lie in the same region so that they are both on the same side of all hyperplanes defined by the hidden units.

We note that given sufficient data that lies on the manifold, it is possible to learn local linear projections of the manifold that will embed it with zero relative error. This can be done with traditional manifold learning methods or by neural networks that contain a sufficiently large number of units. Zhang & Zha (2004) provides an error analysis that shows how the error of their approach depends on the noisiness and number of points in the training data, and the magnitude of the difference between the manifold and its linear approximation. Our contribution here is to analyze the error that can occur when a network learns the embedding very efficiently using a small number of units.

In Appendix B we show that in the worst case, the relative error of the embedding can be unbounded. This occurs when the monotonic chain has very high curvature, so that a separating hyperplane has to be nearly parallel to the segment that follows it. In this section we show that for more typical cases, the relative error will be a small constant.

We will consider a class of monotonic chains in which the total curvature between all segments is less than or equal to some angle T, and in each separating hyperplane is not too close to parallel to the next segment. We denote the angle between S_{k-1} and S_k as θ_{k-1}. (This angle is well defined since S_{k-1} and S_k intersect in an $m-1$-dimensional affine space.) As before, we will drop the subscript when it is $k-1$, and just write θ. Specifically, we define θ so that $\cos\theta = \mathbf{v}^T\mathbf{w}$ (where $\mathbf{v}$ and $\mathbf{w}$ are defined as in Sec. 3, as vectors perpendicular to $S_{k-1} \cap S_k$, and parallel to S_{k-1} and S_k, respectively), defining θ_k similarly for any k. We then express our constraint on the curvature as $\sum_{k=1}^{K-1} |\theta_k| \leq T$.

Now let c be a constant such that we can bound $\mathbf{a}^T\mathbf{v} \geq 1/c$ for any $k-1$. c is a bound on the cosine of the angle between the normal to a separating hyperplane and a vector in the direction of the next segment. To understand this, recall that $\mathbf{a}$ is a unit vector normal to the hyperplane separating S_{k-1} and S_k. By saying this bound holds for all $k-1$, we mean that we are able to choose the hyperplanes that divide the chain into segments so that the angle between the normal to each hyperplane and the following segment is not too big. We next bound the error in terms of c and $\|\delta\|$.

Let $\mathbf{p} = \mathbf{p}_0 + \delta$ be as in the last section. We define the embedding error of $\mathbf{p}$ by $E(\mathbf{p}) = \left(B^{(k)}A^{(k)} - X^{(k)T}\right)\mathbf{p}$, where $X^{(k)}$ denotes the orthogonal projection to S_k, as in Sec. 3. Noting that, by the construction of our network, $B^{(k)}A^{(k)}\mathbf{p}_0 = X^{(k)T}\mathbf{p}_0$ (since $\mathbf{p}_0$ is on S_k) and that $X^{(k)T}\delta = 0$ (due to the orthonormality of $X^{(k)}$), we obtain $E(\mathbf{p}) = B^{(k)}A^{(k)}\delta$. The magnitude of the error therefore is scaled at most by the maximal singular value of $B^{(k)}A^{(k)}$, denoted σ_k.

To bound σ_k we note that $B^{(k)}A^{(k)} = BA + \mathbf{b}\mathbf{a}^T$ for $k \geq 2$ (where, as before, we drop superscripts so that B denotes $B^{(k-1)}$). Therefore, $\sigma_k \leq \sigma_{k-1} + |\mathbf{a}^T\mathbf{b}|$, where σ_{k-1} denotes the largest singular

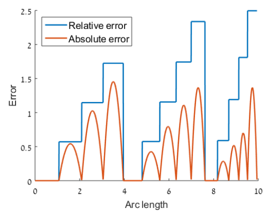

Figure 3: This plot shows the error in flattening the Swiss Roll. Relative error is constant in every segment, starting from zero for each monotonic chain and increasing with each segment. The absolute error (for display purposes this error is normalized by the maximal distance from the Swiss Roll to its linear approximation) behaves similarly, but vanishes at the end points of each segment where the Swiss Roll and its linear approximation coincide.

value of BA. Recall that $\|\mathbf{a}\| = 1$ and $\mathbf{b} = \frac{1}{\mathbf{a}^T\mathbf{v}} BA(\mathbf{w} - \mathbf{v})$. Note that $\mathbf{w} - \mathbf{v} \leq \theta_{k-1}$. Therefore, $|\mathbf{a}^T\mathbf{b}| \leq c\sigma_{k-1}\theta_{k-1}$, from which we conclude that $\sigma_k \leq \sigma_{k-1}(1 + c\theta_{k-1})$.

Finally, note that $B^{(1)}A^{(1)} = X^{(1)T}$, implying that $\sigma_1 = 1$. We therefore obtain $\sigma_k \leq \prod_{j=1}^{k-1}(1 + c\theta_j)$. Note that $\sum_{j=1}^{k-1}\theta_j \leq T$ and so $\prod_{j=1}^{k-1}(1 + c\theta_j) \leq (1 + \frac{cT}{k-1})^{k-1}$. Therefore, $\sigma_k \leq \left(1 + \frac{cT}{k-1}\right)^{k-1} \leq e^{cT}$. We conclude that $\|E(\mathbf{p}_0 + \delta)\| \leq e^{cT}\|\delta\|$.

Many segments of many monotonic chains can be divided using hyperplanes in which c is not too big, and may be as low as 1. For such manifolds, when a point is perturbed away from the manifold, its coordinates will not be changed by more than the magnitude of the perturbation times a small constant factor. For example, if $T = \pi/4$ and $c = 1$ then $e_k \leq e^{\frac{\pi}{4}} \approx 2.19$. Note that rather than beginning at the start of the monotonic chain, we could ”begin” in the middle, and work our way out. That is, provide an orthonormal basis for the middle segment and add hidden units to represent the chain from the central segment toward either ends of the chain. This can reduce the total curvature from the starting point to either end by up to half. We further emphasize that this bound is not tight.

We conclude this section by showing the error obtained in using our construction in the ”Swiss Roll” example. To represent this data we use hidden units and their corresponding hyperplanes to divide the Roll into three monotonic chains (see Section 5 below for further details). We then divide each chain into segments, obtaining a total of 14 segments. Figure 1 shows the points that are input into the network, and the 2D representation that the network outputs. The points are color coded to allow the reader to identify corresponding points. In Figure 3 we further plot the absolute and relative error in embedding every point of the Swiss Roll due to the linear approximation used by the network. One can see that the Swiss Roll is unrolled almost perfectly. In fact, despite the relatively large angular extent of each monotonic chain (the three chains range between 126 to 166.5 degrees each in total curvature), the relative error does not exceed 2.5. (In fact, our bound for this case is very loose, amounting to 18.3 for $166.5°$.) The mean relative error is 0.98, indicating that the magnitude of the error is approximately the same as the distance of points to the approximating monotonic chains.

5 Combinations of Monotonic Chains

To handle non-monotonic chains and more general piecewise linear manifolds that can be flattened we show that we can use a network to divide the manifold into monotonic chains, embed each of these separately, and then stitch these embeddings together. Suppose we wish to flatten a non-monotonic chain that can be divided into L monotonic chains, $M_1, M_2, ...M_L$. Let A_l, $\mathbf{a}_{0l}$ and B_l denote the matrices and bias used to represent the hidden units that flatten M_l, which has K_l segments. We suppose that a set of J_l hyperplanes (that is, a convex polytope) can be found that separate M_l from the other chains. Let N_l denote a matrix in which the rows represent the normals to these hyperplanes, oriented to point away from M_l. We can concatenate these vertically, letting $A'_l = [A_l; N_l]$. We next let $\Upsilon = -n\mathbf{1}_{m\times J_l}$ where $\mathbf{1}_{m\times J_l}$ denotes an $m \times J_l$ matrix containing all ones and n is a very large constant. Note that B_l has m rows. So we can define $B'_l = [B_l, \Upsilon]$, where the matrices are concatenated horizontally.

We now note that if $\mathbf{u} = B'_l[A'_l\mathbf{x} + \mathbf{a}_{0l}]_+$ then when $\mathbf{x}$ lies on M_l, $\mathbf{u}$ will contain the coordinates of $\mathbf{x}$ embedded in $\mathcal{R}^m$, as before. When $\mathbf{x}$ lies on a different monotonic chain, $\mathbf{u}$ will be a vector with very small negative numbers. Applying RELU will therefore eliminate these numbers.

A'_l and B'_l therefore represent a module consisting of a two layer network that embeds one monotonic chain in $\mathcal{R}^m$ while producing zero for other chains. We can then stitch these values together. First, we must rotate and translate each embedded chain so that each chain picks up where the previous one left off. Let R_l denote the rotation of each chain, and let $\mathbf{b}_{0l}$ denote its appropriate translation. Then, for each chain, the appropriate coordinates are produced by

$$[R_l B_l'[A_l'\mathbf{x} + \mathbf{a}_{0l}]_+ + \mathbf{b}_{0l}]_+.$$

We can now concatenate these for all chains to produce the final network. We let A, $\mathbf{a}_0$ and $\mathbf{b}_0$ be the vertical concatenation of all A_l' and $\mathbf{a}_{0l}$ and $\mathbf{b}_{0l}$ respectively, and let B be the block-diagonal concatenation of all $R_l B_l'$. The application of $[B[A\mathbf{x} + \mathbf{a}_0]_+ + \mathbf{b}_0]_+$ to $\mathbf{x} \in M_l$ will produce a vector with mL entries in which the $m(l-1)+1, ..., ml$ entries give the embedded coordinates of $\mathbf{x}$ and the rest of the entries are zero. We can now construct a third layer of the network to then stitch these monotonic chains together. Let C denote a matrix of size $m \times mL$ obtained by concatenating horizontally L identity matrices of size $m \times m$. Then the output of the network is:

$$\mathbf{u} = C[B[A\mathbf{x} + \mathbf{a}_0]_+ + \mathbf{b}_0]_+.$$

Note, for example, that the first element of $\mathbf{u}$ is the sum of the first coordinates produced by each module in the first two layers. Each of these modules produces the appropriate coordinates for points in one monotonic chain, while producing 0 for points in all other monotonic chains.

We note that this summation may result in wrong values if there is overlap between the regions (which will generally be of zero measure). This can be rectified by replacing the summation due to C by max pooling, which allows overlap of any size. Together, all three layers will require $\left(\sum_{l=1}^{L} J_l + m + K_l - 1\right) + (L+1)m$ units. If the network is fully connected, this requires $\left(\sum_{l=1}^{L} J_l + m + K_l - 1\right)(d + Lm) + Lm^2$ weights.

Note that the size of this network depends on how many regions are required (L) and how many hyperplanes each region needs to separate it from the rest of the manifold (L_l). In the worst case, this can be quite large. Consider, for example, a 1D manifold that is a polyline that passes through every point with integer coordinates in $\mathcal{R}^d$. To separate any portion of this polyline from the rest will require regions that are not unbounded, and so $L_l = O(d)$ for all l. We expect that many manifolds can be divided appropriately using many fewer hyperplanes. We have shown this for the example of a Swiss rolls (Figure 1).

6 EXPERIMENTS

Up to this point we have theoretically analyzed the representational capacity of a deep network. Our primary result is to show that data lying on a monotonic chain can be efficiently flattened by a network with two hidden layers, using $m+k-1$ hidden units in the first layer, and m units in the second layer. An important question is whether real networks trained with stochastic gradient descent can uncover such efficient representations. In this section we address that question experimentally.

We do not expect that a trained network will always produce the constructions developed in this paper. First,we note that our constructions provide an upper bound; more efficient representations possible. So we predict that $m + k - 1$ *or fewer* hidden units are needed. Second, a trained network may settle in a local minimum, and not produce an efficient embedding, even though one might be possible. To determine whether a particular architecture can produce a good embedding, we train networks with multiple random starting points, and select the solutions that produce very low error.

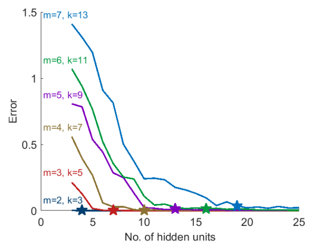

Figure 4: This graph shows error in the embedding produced by a trained network. Each curve represents a manifold of different dimension, with a different number of segments. Each curve shows how error in the embedding on validation points changes as the number of hidden units increases. Stars indicate the validation error at the point of each curve in which $h = m + k - 1$. As our theory predicts, the error has reached an asymptote close to zero at these points.

To determine the number of hidden units needed to create effective embeddings, we generate data on monotonic chains in which we vary the dimension of the manifold, m, and the number of segments,

k. An example in which $m = 2$ and $k = 7$ is shown in Figure 5. Note that there is some skew in the chain, so that none of the dimensions can be trivially embedded by a single linear projection. We sample 40,000 points on the manifold. We then train a regressor, with a varying number of hidden units, using the squared difference between the ground truth distance between pairs of embedded points and the distance computed by the network as a loss function. This simulates non-linear metric learning. For each condition, we repeat training 15 times, and report the minimum error in the objective (see Figure 4). We can see that for each curve the error has dropped to an asymptote near zero when $h = m + k - 1$, just as our theory predicts.

In Figure 5 we show a typical example produced for a 2D manifold with seven segments, shown in a 3D space. Portions of hyperplanes correspond to six hidden units. This solution resembles our constructions in several ways. One hyperplane is active over the entire chain, while the other hyperplanes intersect the manifold at the intersection of consecutive segments. The solution differs from our construction in that some hyperplanes are used to handle two segments of the manifold; it is even more efficient than our construction. And two hyperplanes, at the top, intersect the manifold in the same location. These hidden units have weights with opposite signs, producing positive outputs for different segments. For reasons of space and simplicity, we do not discuss these constructions theoretically, but it is straightforward to show that they can also produce efficient embeddings.

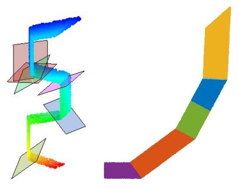
Figure 5: A network trained with $m = 2, k = 7, h = 6$. Left: Colored dots represent points on the manifold. Their ground truth coordinates are encoded by the size and hue of the dots. Colored rectangles represent the hyperplanes associated with the six hidden units. Right: We show the labels generated for each point, in 2D. Points are colored to indicate their segment. The embedding is near perfect.

We perform a final experiment to get a sense of whether such embeddings can occur with more realistic data. We generate images of a face with azimuth ranging from 0 to 50 degrees, and with elevation ranging from 0 to 8 degrees. As a loss function, we use an L2 norm between the m output units and the true azimuth and elevation. Because the images have many pixels, and the amount of training data is limited, a fully connected network would overfit the data if we use each pixel as an input dimension. Consequently, we perform PCA before training to reduce the faces to a 3D space, which also allows us to visualize the input and resulting network (see Figure 6). We can see that the data forms an approximately 2D manifold, but that it is much messier than with our previous, synthetic data. The resulting embedding captures the azimuth and elevation reasonably well, but with some noise (eg., it does not form a perfect grid). We can also see that the hyperplanes associated with the first hidden layer of the network also resemble our construction, with individual units periodically intersecting the manifold as it curves.

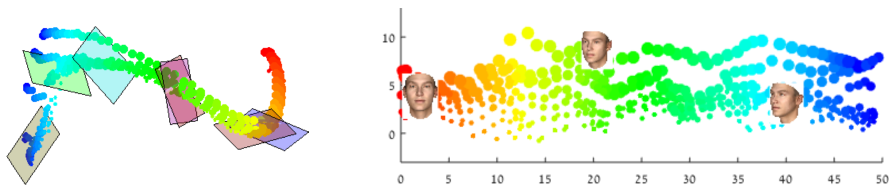

Figure 6: We train a regression network to learn the azimuth and elevation of face images. Left: the face images projected to 3D and the hyperplanes learned in the first network layer. Dot size encodes elevation and hue encodes azimuth. Right: We show the images embedded in a 2D space by the trained network.

7 DISCUSSION

We show that deep networks can represent data that lies on a low-dimensional manifold with great efficiency. In particular, when using a monotonic chain to approximate some component of the data, the addition of only a single neural unit can produce a new linear segment to approximate a region of the data. This suggests that deep networks may be very effective devices for such dimensionality reduction. It also may suggest new architectures for deep networks that encourage this type of dimensionality reduction.

We also feel that our work makes a larger point about the nature of deep networks. It has been shown by Montufar et al. (2014) that a deep network can divide the input space into a large number of regions in which the network computes piecewise linear functions. Indeed, the number of regions can be exponential in the number of parameters of the network. While this suggests a source of great power, it also suggests that there are very strong constraints on the set of regions that can be constructed, and the set of functions that can be computed. Our work shows one way in which a single hidden unit can control the variation in the linear function that a network computes in two neighboring regions; it can shape this function to follow a manifold that contains the data.

ACKNOWLEDGEMENTS

This research is based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA R&D Contract No. 2014-14071600012. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

This research is also based upon work supported by the Israel Binational Science Foundation Grant No. 2010331 and Israel Science Foundation Grants No. 1265/14.

The authors thank Angjoo Kanazawa and Shahar Kovalsky for their helpful comments.

A CONTINUITY OF EMBEDDING

In Section 3 of our paper we defined the weight matrices $A^{(k)}$ and $B^{(k)}$ and the bias vector $\mathbf{a}_0{}^{(k)}$ that map an input vector $\mathbf{x}$ to its geodesic coordinates on the manifold. We showed that this construction indeed maps points on S_k to their geodesic coordinates, so that this coordinate system is consistent in orientation with the coordinates assigned to the previous segments S_1, ..., S_{k-1}. It is now left to show that the bias $\mathbf{a}_0{}^{(k)}$ is chosen properly to create a continuous embedding.

Consider a point $\mathbf{x} \in S_k$. Denote by $\bar{\mathbf{x}}$ its projection onto $S_{k-1} \cap S_k$, so that $\mathbf{x} = \bar{\mathbf{x}} + \beta\mathbf{v}$ for a scalar β. Denoting the embedded coordinates of $\mathbf{x}$ by $\mathbf{u}$,

$$\mathbf{u} = B^{(k)}(A^{(k)}\mathbf{x} + \mathbf{a}_0{}^{(k)}).$$

We want to verify that as β tends to 0 $\mathbf{u}$ will coincide with the embedding of $\bar{\mathbf{x}}$ due to S_{k-1}, i.e.,

$$\bar{\mathbf{u}} = B(A\bar{\mathbf{x}} + \mathbf{a}_0).$$

In our construction, $B^{(k)}$ is obtained from B by appending the column vector $\mathbf{b}$ to its right side, and $A^{(k)}$ is obtained from A by appending the row vector $\mathbf{a}^T$ to its bottom, so that $B^{(k)}A^{(k)} = BA + \mathbf{b}\mathbf{a}^T$. Recall further that $\mathbf{a}_0{}^{(k)}$ is obtained from $\mathbf{a}_0$ by appending the scalar a_0 at its end. We therefore obtain

$$\mathbf{u} = (BA + \mathbf{b}\mathbf{a}^T)\mathbf{x} + B\mathbf{a}_0 + a_0\mathbf{b}.$$

Replacing $\mathbf{x} = \bar{\mathbf{x}} + \beta\mathbf{v}$ we obtain

$$\mathbf{u} = (BA + \mathbf{b}\mathbf{a}^T)\bar{\mathbf{x}} + \beta(BA + \mathbf{b}\mathbf{a}^T)\mathbf{v} + B\mathbf{a}_0 + a_0\mathbf{b}.$$

Since $\mathbf{a} = \mathbf{h}$, $\mathbf{a}^T\bar{\mathbf{x}} + a_o = 0$ and we get

$$\mathbf{u} = B(A\bar{\mathbf{x}} + \mathbf{a}_0) + \beta(BA + \mathbf{b}\mathbf{a}^T)\mathbf{v},$$

which coincides with $\bar{\mathbf{u}}$ when $\beta \to 0$, implying that the embedding is extended continuously to S_k. Note that by construction $\mathbf{a}^T\mathbf{y} + a_0 \leq 0$ for all $\mathbf{y} \in S_1 \cup ... \cup S_{k-1}$ so RELU ensures that the embedding of these segments will not be affected by the additional unit.

B WORST-CASE ERROR

In this section we show that the error obtained while embedding noisy points using our construction can in principle be unbounded. As we show below, this happens when we are forced to choose hyperplanes that are almost parallel to the segments they represent. In contrast, Section 4.1 of our paper shows that we can bound the error in many reasonable scenarios.

To show that the error can be unbounded, we consider a simple case in which the piecewise linear manifold consists of three connected 1D line segments, S_1, S_2 and S_3, with 2D vertices respectively of $(0, 0)$ and $(N, 0)$, $(N, 0)$ and (N, ϵ), and (N, ϵ) and $(0, \epsilon)$. N is very large, and ϵ is very small (see Figure 7). Since three segments compose a 1D manifold, three hidden units defining three hyperplanes, H_1, H_2 and H_3 (lines) will be needed to represent the manifold. In addition, a single output unit will sum the results of these units to produce the geodesic distance from the origin to any point on the three segments.

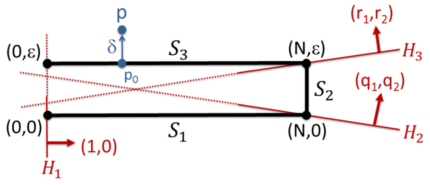

Figure 7: In black, we show a 1D monotonic chain with three segments. In red, we show three hidden units that flatten this chain into a line. Note that each hidden unit corresponds to a hyperplane (in this case, a line) that separates the segments into two connected components. The third hyperplane must be almost parallel to the third segment. This leads to large errors for noisy points near S_3.

Using our construction in Section 3 of the paper we get the embedding $f(\mathbf{p}) = B[A\mathbf{p} + \mathbf{a}_0]_+$ with

$$B = \left(1, \frac{1}{q_2}, -\frac{1}{r_1}\left(2 + \frac{q_1}{q_2}\right)\right), \quad A = \begin{pmatrix} 1 & 0 \\ q_1 & q_2 \\ r_1 & r_2 \end{pmatrix}, \quad \mathbf{a}_0 = \begin{pmatrix} 0 \\ q_3 \\ r_3 \end{pmatrix}.$$

Note that the first row of A uses the standard orthogonal projection $(x, y) \rightarrow x$; the two other rows of A and $\mathbf{a}_0$ separate the three segments with (1) $q_1, q_2 > 0$ and $q_1/q_2 \leq \epsilon/N$ and $q_3 = -q_1 N$ set so that the separator H_2 goes through $(N, 0)$, and (2) $r_1 < 0$, $r_2 > 0$ and $r_1/r_2 \geq -\epsilon/N$, and $r_3 = -r_1 N - r_2 \epsilon$ set so that the separator H_3 goes through (N, ϵ). It can be easily verified that in this setup points on the first segment $(x, 0)$, $0 \leq x \leq N$ are mapped to x, points (N, y), $0 \leq y \leq \epsilon$ on the second segment are mapped to $N + y$, and points (x, ϵ), $0 \leq x \leq N$ on the third segment are mapped to $N + \epsilon + (N - x)$.

Ideally, we would want $\mathbf{p}$ to be embedded to the same point as $\mathbf{p}_0$. Let $E(\mathbf{p}) = f(\mathbf{p}) - f(\mathbf{p}_0)$. Clearly $E(\mathbf{p}) = B^{(k)} A^{(k)} \delta$. It can be readily verified that, under these conditions, when $\mathbf{p}_0 \in S_1$ then $E(\mathbf{p}) = 0$; when $\mathbf{p}_0 \in S_2$ then $E(\mathbf{p}) = (1 + q_1/q_2)\delta$, and when $\mathbf{p}_0 \in S_3$ then $E(\mathbf{p}) = (1 - (r_2/r_1)(2 + q_2/q_1))\delta$. Therefore, there is no error in embedding $\mathbf{p}$ for $\mathbf{p}_0 \in S_1$. The error in embedding $\mathbf{p}$ with $\mathbf{p}_0 \in S_2$ is small and bounded (since $q_1/q_2 \leq \epsilon/N$, assuming ϵ is small and N is large), while the error in embedding $\mathbf{p}$ when $\mathbf{p}_0 \in S_3$ can be huge since $-r_2/r_1 \geq N/\epsilon$. In the next section we show that this can only happen when there is a large angle between a segment and the normal to the previous separating hyperplane.

C CLASSIFICATION

In experiments in the body of this paper we have demonstrated that the theoretical constructions that we analyze can arise when networks are trained to solve regression problems that map points on the manifold to their low-dimensional embeddings. An interesting question is whether similar embeddings may be learned by a network that is trained to classify points that lie on a low-dimensional manifold when it is more efficient to represent the boundaries of these classes in the embedded space than it is in the ambient space. In this Appendix, we describe some very preliminary experiments that address this question.

First we note that the embeddings that arise in solving classification problems may be much less constrained and therefore more complex than those that arise in regression problems. The regression loss function directs the network to learn the known, ground truth coordinates of the embedded manifold. Only an isometric unfolding of the manifold will satisfy this condition. While this isometric embedding will facilitate classification as well, there may be many non-isometric unfoldings that will be equally useful in classification.

As a simple example of this, suppose a monotonic chain contains two classes that are linearly separable, once the chain is isometrically embedded in a low-dimensional space. If instead of an isometric embedding, we allow a related embedding in which each segment of the chain undergoes a different linear transformation that stretches it in the direction of the linear separator, or orthogonal to the separator, the classes will still be linearly separable in the transformed, non-isometric embedding.

As another example, no mapping of the manifold to a low-dimensional space will allow for correct classification if it maps two points from different classes to the same point in the low-dimensional space. However, classification may not be affected if two points from the same class are mapped to the same point. So when points from only one class appear near the boundary between two segments, a network may learn a mapping in which the points from two segments overlap in the low-dimensional space.

It is an open and rather complex problem to determine which mappings of the input to low-dimension may be suitable for classification of a particular set of labeled points. However, we stress that the main point of our paper is to show that when isometric embeddings can be used to solve a problem, a deep network can efficiently represent such embeddings. It is certainly possible that the network can also efficiently find alternate embeddings that are equally useful.

Bearing this in mind, we have designed some simple classification tasks and examined the embeddings that they give rise to in a neural network. We stress that these experiments are quite preliminary, and should be taken as intriguing examples that can help motivate future work.

In our experiments we created monotonic chains with seven segments, similar to those used in our earlier experiments. We generate 20,000 points that lie on each chain. To label these points with classes, we unfolded the chain and intersected it with several lines, varying the number. These lines form an arrangement on the 2D unfolded manifold; we labeled each region of the arrangement, which is a convex polygon, as a separate class. We did this randomly, selecting arrangements in which classes tended to span multiple segments.

We then trained a network to perform classification. After the input layer, the next layer contained between five and eight hidden units. This was followed by a layer containing two hidden units. This was followed by another layer with 10-30 units, and an output layer with a unit for each class. Relu was used between layers, with softmax for the loss function. The layer containing two units essentially represents a two-dimensional embedding of the input. The previous layer could be used to represent the constructions developed in this paper, while the subsequent layer can be used to classify the data in the low-dimensional space. This architecture allows us to easily extract the embedding that the network has learned.

Figure 8 shows a typical example of the results. On the left we plot the input points, color coded to indicate their class. On the right, we plot each point at its embedded location, color coded to indicate to which segment it belongs. The embedding preserves the order and continuity of the segments. In several cases each segment has been approximately transformed by a different linear transformation. In the case of the red and green colored segments on the right, there is some overlap. Looking at the left-hand figure we can see that in this case, points near the boundary between the two segments belong to the same class. So this folding over of the segments in the embedding does not interfere with the network's ability to correctly classify the points.

In general, this embedding meets our expectations, showing that the monotonic chain can be very efficiently mapped to a low-dimensional space using very few units, in a way that enables accurate classification. It would be interesting in future work to determine the class of mappings that can be instantiated efficiently by a network, and to understand how these relate to different classification problems. It would also be interesting to design classification problems that can only be solved using isometric embeddings, and to determine whether these embeddings can be found by neural networks.

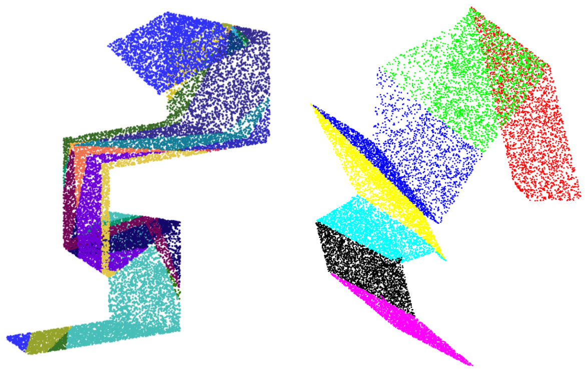

Figure 8: We train a network on a classification problem in which the points lie on a low-dimensional manifold. We show the points on the left, color coded to indicate their class. We then extract the embedding learned by the network. Here we show the input mapped to this embedding, with points colored to indicate which of the seven segments of the monotonic chain they lie on.

D Deeper Networks

We also note that the previously developed constructions can be applied recursively, producing a deeper network that progressively approximates data using linear subspaces of decreasing dimension. That is, we may first divide the data into a set of segments that each lie in a low dimensional subspace whose dimension is higher than the intrinsic dimension of the data. Then we may subdivide each segment into a set of subsegments of lower dimension, using a similar construction, and deeper layers of the network. These subsegments may represent the original data, or they be further subdivided by additional layers, until we ultimately produce subsegments that represent the data.

We first illustrate this hierarchical approach with a simple example that requires only one extra layer in the hierarchy. Consider a monotonic chain of K, m_2-dimensional linear segments that collectively lie in a m_1-dimensional linear subspace, $\mathcal{L}$, of a d-dimensional space, with $m_2 < m_1$. We can construct the first hidden layer with m_1 units that are active over the entire monotonic chain, so that their gradient directions form an orthonormal basis for $\mathcal{L}$. The output of this layer will contain the coordinates in $\mathcal{L}$ of points on the monotonic chain. These can form the input to two layers that then flatten the chain, as described in Section 3.

In Section 3 we had already shown how to flatten the manifold with two layers that take their input directly from the input space. Here we accomplish the same end with an extra layer. However, this construction, while using more layers, may also use fewer parameters. The construction in Section 3 required $d(m_2 + K - 1)$ parameters. Our new construction will require $dm_1 + m_1(m_2 + K - 1)$ parameters. Note that as K increases, the number of parameters used in the first construction increases in proportion to d, while in the second construction the parameters increase only in proportion to m_1. Consequently, the second construction can be much more economical when K is large and m_1 is small.

In much the same way, we could represent a manifold using a hierarchy of chains. The first layers can map a m_1-dimensional chain to a linear m_1-dimensional output space. The next layers can select an m_2-dimensional chain that lies in this m_1-dimensional space, and map it to an m_2-dimensional space. This process can repeat indefinitely, but whether it is economical will depend on the structure of the manifold.

References

J. Ba and R. Caruana. Do deep nets really need to be deep? In *NIPS*, pp. 2654–2662, 2014.

R. Basri and D. W. Jacobs. Lambertian reflectance and linear subspaces. *PAMI*, 25(2):218–233, 2003.

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

M. Bianchini and F. Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Trans. on Neural Networks and Learning Systems*, 25(8), 2014.

P. P. Brahma, D. Wu, and Y. She. Why deep learning works: A manifold disentanglement perspective. 2015.

S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005.

C. K. Chui and H. N. Mhaskar. Deep nets for local manifold learning. *ArXiv preprint:1607.07110*, 2016.

N. Cohen, O. Sharir, and A. Shashua. On the expressive power of deep learning: A tensor analysis, 2015.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

O. Delalleau and Y. Bengio. Shallow vs. deep sum-product networks. In *NIPS*, pp. 666674, 2011.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. John Wiley & Sons, 2012.

R. Eldan and O. Shamir. The power of depth for feedforward neural networks. *ArXiv preprint: 1512.03965*, 2015.

R. Giryes, G. Sapiro, and A. M. Bronstein. Deep neural networks with random gaussian weights: A universal classification strategy? *ArXiv preprint: 1504.08291*, 2016.

R. Huang, F. Lang, and C. Shu. Nonlinear metric learning with deep convolutional neural network for face verification. In J. et al. Yang (ed.), *Biometric Recognition*, volume 9428 of *Lecture Notes in Computer Science*, pp. 78–87. Springer, 2015.

K. C. Lee, J. Ho, M. H. Yang, and D. Kriegman. Video-based face recognition using probabilistic appearance manifolds. In *CVPR*, volume 1, pp. I–313. IEEE, 2003.

H. Mobahi, J. Weston, and R. Collobert. Deep learning from temporal coherence in video. In *ICML*, 2009.

G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *NIPS*, pp. 2924–2932, 2014.

B. Nadler, S. Lafon, R. R. Coifman, and I. G. Kevrekidis. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. In *NIPS*, volume 18, 2005.

S. Chopra R. Hadsell and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006.

S. Rifai, Y. N. Dauphin, P. Vincent, Y. Bengio, and X. Muller. The manifold tangent classifier. In *NIPS*, pp. 2294–2302, 2011.

S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500): 2323–2326, 2000.

R. Salakhutdinov and G. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *AISTATS*, 2007.

U. Shaham, A. Cloninger, and R. R. Coifman. Provable approximation properties for deep neural networks. *ArXiv preprint: 1509.07385*, 2015.

M. Telgarsky. Representation benefits of deep feedforward networks. *ArXiv preprint: 1509.08101*, 2015.

J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:23192323, 2000.

M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

J. Weston, F. Ratle, and R. Collobert. Deep learning via semi-supervised embedding. In *ICML*, 2008.

D. Yi, Z. Lei, S. Liao, and S. Z. Li. Deep metric learning for person re-identification. In *ICPR*, 2014.

Zhen-yue Zhang and Hong-yuan Zha. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *Journal of Shanghai University (English Edition)*, 8(4):406–424, 2004.