Principles of Riemannian Geometry in Neural Networks

Michael Hauser & Asok Ray, Pennsylvania State University

Helen Ngo
TDLS

Based on the paper by Michael Hauser & Asok Ray (Pennsylvania State University)
Presented at NIPS 2017

August 13 2018

Why formalize the theory behind neural networks?

i.e. why should you care?

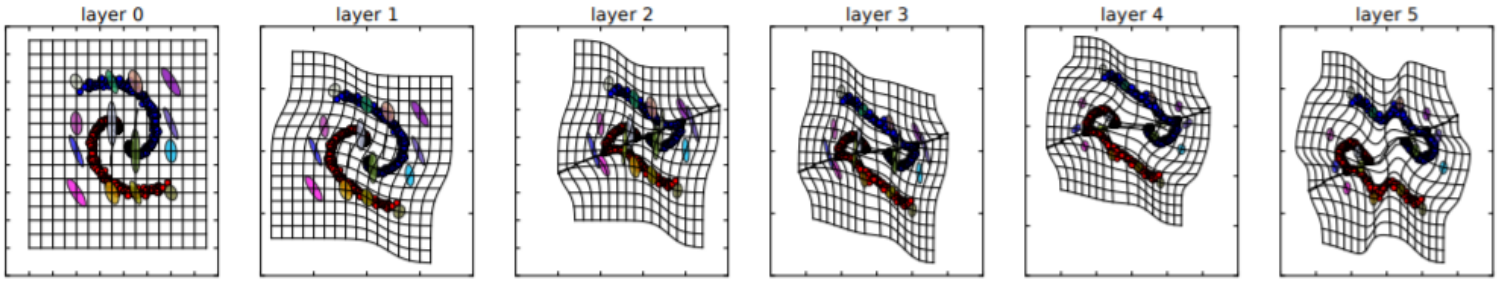
- Lots of industry machine learning is empirical
- Lack of mathematical backing for empirical findings
- Data scientists often fall into the trap of fitting their findings or hypotheses to a narrative

Neural networks as latent variable representations

- Canonical view of neural networks is formalized as a learning representations of "uncovering latent variables"
- Convolutional neural networks are explained as "learning hierarchical representations of images"
- Long-short term memory machines have memory cells which manage conflicts with long-term dependencies and new data

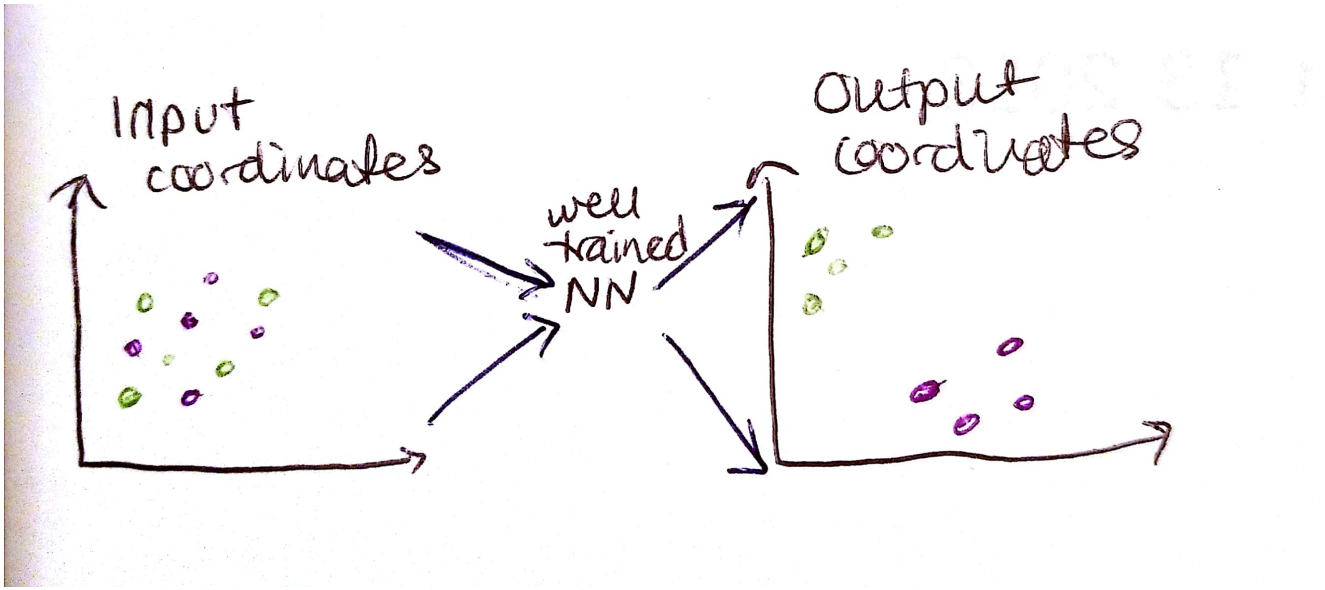
Geometric representations for deep learning

- **Goal:** Learn an optimal coordinate representation system of the underlying data manifold where different target classes are linearly separable by hyperplanes
- Iteratively transform the coordinate representations (network layers) of the data manifold (input data) with non-linear transformations (activation functions) into a representation where the classes (targets) are linearly separable by hyperplanes



Geometric representations for deep learning (2)

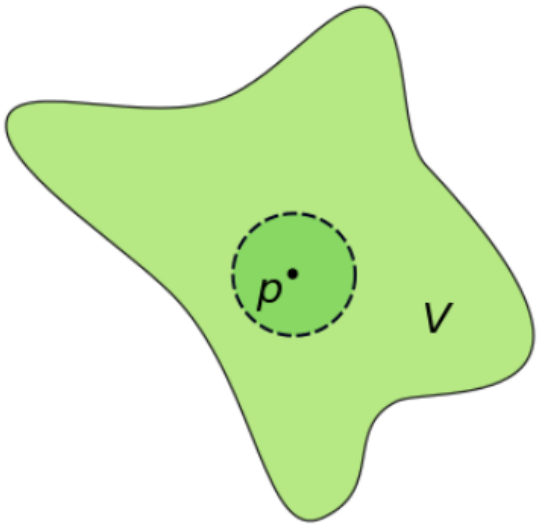
- If the network is well-trained, then the Euclidean distance between two input points of the same class may be far apart when represented by the input coordinates but close together in the output representation
- Similarly, two points from different classes could appear close in the input space (as measured by Euclidean distance) but far apart in the output coordinates



What's a manifold?

first, some preliminary definitions

- A **topological space** is a set of points (each with a set of neighbourhoods)
- A **neighbourhood** of a point is a set of points containing that point where one can move some amount in any direction away from that point without leaving the set
- Or, given a topological space X and some point $p \in X$, a **neighbourhood** of p is a subset V of X that includes an open set U containing p. That is, $p \in U \subseteq V$



What's a manifold? (2)

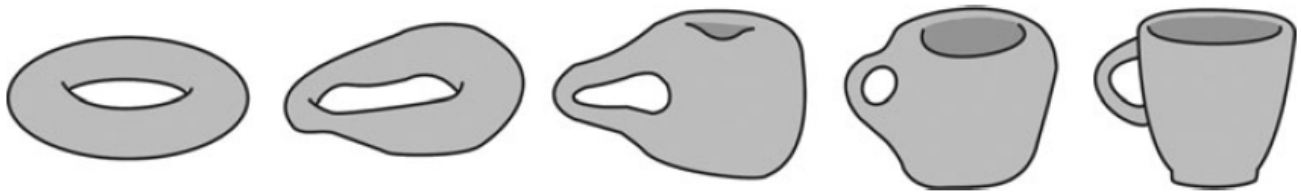
- A **manifold** is topological space which is locally "equivalent" to Euclidean space at each point
- Equivalence can be expressed via n coordinate functions, $c_i : M \longrightarrow \mathbb{R}$, which together form a "structure-preserving" function, $c : M \longrightarrow \mathbb{R}^n$, called a chart.
- The Earth seems flat if you stand on it and look around
- "Flatness" enables familiar concepts such as distance and gives geometric meaning

Okay, but really, what's a manifold?

- One dimensional manifolds: lines, circles
- Two dimensional manifolds (surfaces): the plane, sphere, torus
- Three dimensional manifolds (surfaces): ball, Earth, cube
- We can embed manifolds into higher dimensional spaces
- An **embedding** is when an instance of one mathematical structure is contained within another instance (i.e. embed two-dimensional manifolds in three-dimensional space)

Okay, but really, what's a manifold? (2)

- A **homeomorphism** is a continuous function between topological spaces that has a continuous inverse function
- Two topological spaces with a homeomorphism between them are called homeomorphic and we view them as the same
- A manifold **locally resembles Euclidean space**; that is, every point has a neighbourhood homeomorphic to an open subset of Euclidean space while globally it may not



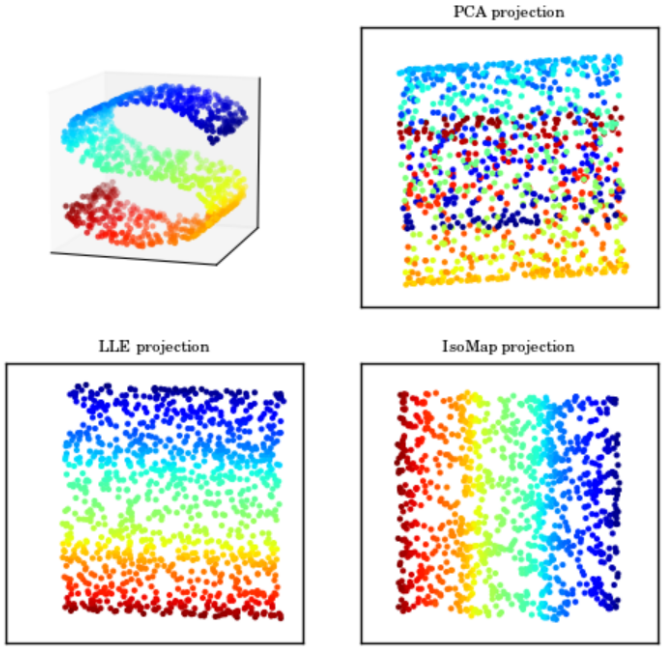
To a topologist, a doughnut is the same as a coffee cup!

Principal components analysis and manifold learning

- PCA identifies three principal components within the data. Projection onto the first two PCA components may not perfectly separate the data
- After PCA, only points on some embedded subspace inside in the original space space will be attainable
- The embedded subspace is a linear subspace and also manifold (e.g. hyperplane)
- For a non-linear dimensional reduction technique, that subspace may be more complex (e.g. curved hyper-surface)

Principal components analysis and manifold learning (2)

- Take a two-dimensional manifold in a three-dimensional space
- Manifold learning (Locally linearly embeddings (LLE) and IsoMap) preserves the local structure when projecting the data, preventing the mixing of the colors
- `http://www.astroml.org/book_figures/chapter7/fig_S_manifold_PCA.html`



Non-linear dimensionality reduction

in the context of locally linear embeddings

- We often want to map high-dimensional space to a low-dimensional embedding, assuming the high-dimensional data lies on an embedded non-linear manifold within the higher-dimensional space
- Locally linear embeddings find a set of the nearest neighbors of each point
- We then compute a set of weights for each point that best describes it as a linear combination of its neighbors
- The weights that reconstructs the ith data point in the higher dimensional space will be used to reconstruct the same point in the lower dimensional space, creating a neighborhood preserving map

Non-linear dimensionality reduction (2)

in the context of locally linear embeddings

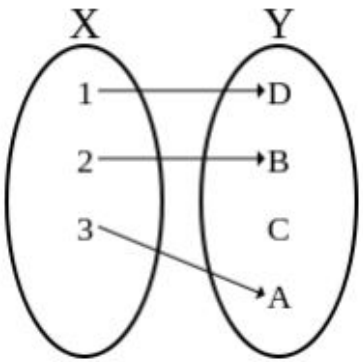
- Each point X_i in the higher dimensional space is mapped onto a point Y_i in the lower dimensional space by minimizing the cost function $C(Y) = \sum_i \mid Y_i - \sum_j W_{ij} Y_j \mid^2$
- We fix the weights from the previous step and minimize the points Y to find a new coordinate representation
- This is equivalent to finding the eigendecomposition of an NxN matrix (for N data points) where the non-zero eigenvectors make up an orthogonal set of coordinates

Locally linear embeddings & relations to manifold calculus

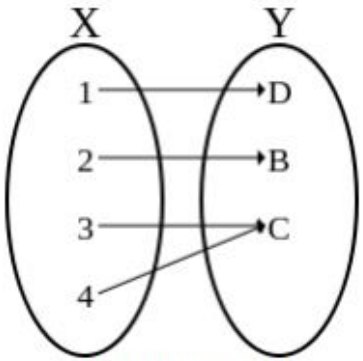
- We will formalize neural networks as coordinate representations of the data manifold
- In locally linear embeddings, Euclidean distance can be used on the newly learned coordinate system, which is ideal for us to understand how "close" points are to each other
- Coordinate representation of the metric tensor (loss) can be backpropagated (pull-backed) through to the input to measure distance in the input coordinates

One more word on homeomorphisms

- Recall that we establish homeomorphisms as continuous maps between topological spaces
- A homeomorphism f is continuous, one-to-one, onto, and has a continuous inverse f^{-1}
- They "preserve topological integrity"



one-to-one (injective), but not onto



onto (surjective), but not one-to-one

Feedforward networks as coordinate transformations

- Consider a homeomorphism as a coordinate system contained in the manifold M
- Our homeomorphism-as-a-coordinate-system will be $x : M \longrightarrow \mathrm{x}\ (\mathrm{M}) \subseteq \mathbb{R}^{dimM}$
- A feedforward neural network learns coordinate transformations as $\varphi^{(l)} :\ x^{(l)}\ (M) \longrightarrow (\varphi^{(l)} \circ x^{(l)})\ (M)$
- The feedforward network is initialized with Cartesian coordinates as $x^{(0)} :\ M \longrightarrow \mathrm{x}^{(0)} :\ M$
- The next iteration of the coordinate transformation is defined as $x^{(l+1)} := \varphi^{(l)}\ (x^{(l)}) :\ (M) \longrightarrow \mathrm{x}^{(l+1)}\ (M)$

Feedforward networks as coordinate transformations (2)

- A data point $q \in M$ can be represented w.r.t. a coordinate system
- For the coordinates at layer $l+1$, our point q is represented as layerwise composition
  $x^{(l+1)}(q) := (\varphi^l \circ ... \circ \varphi^1 \circ \varphi^0 \circ x^0)(q)$
- Each layer of the neural network is represented by some function φ
- A **bijection** is a one-to-one (injective) and onto (surjective) mapping of a set X to a set Y
- Coordinate transformations are bijective (ex. ReLU is not a "proper" coordinate transformation since it is not 1:1)

Feedforward networks as coordinate transformations (3)

- How do activation functions fit into this paradigm?
- For some activation function f, a feedforward network transforms coordinates as
  $x^{(l+1)} := \varphi^l\left(x^l\right) := f(x^{(l)}; l)$
- Here, φ represents layer l of the neural network (coordinate transformation) and f specifically refers to the activation function
- "The $l+1$-th coordinate system is iteratively defined as the l-th coordinate transformation on the previous (l-th) coordinate system"

Feedforward networks as coordinate transformations (3)

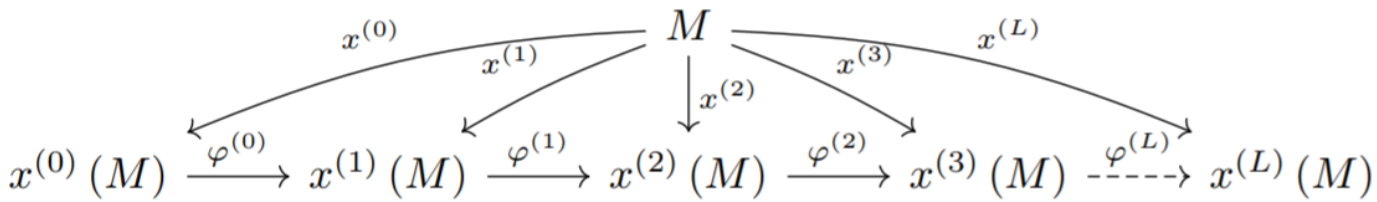
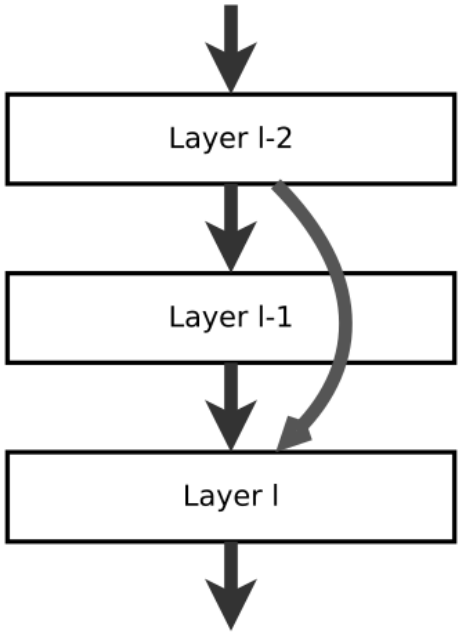


Figure 1: Coordinate systems $x^{(l+1)} := \varphi^{(l)} \circ ... \circ \varphi^{(1)} \circ \varphi^{(0)} \circ x^{(0)}$ induced by the coordinate transformations $\varphi^{(l)} : x^{(l)}(M) \to \left(\varphi^{(l)} \circ x^{(l)}\right)(M)$ learned by the neural network. The pullback metric $g_{x^{(l)}(M)}(X, Y) := g_{\left(\varphi^{(l)} \circ x^{(l)}\right)(M)}\left(\varphi_*^{(l)} X, \varphi_*^{(l)} Y\right)$ pulls-back (i.e. backpropagates) the coordinate representation of the metric tensor from layer $l+1$ to layer l, via the pushforward map $\varphi_*^{(l)} : Tx^{(l)}(M) \to T\left(\varphi^{(l)} \circ x^{(l)}\right)(M)$ between tangent spaces.

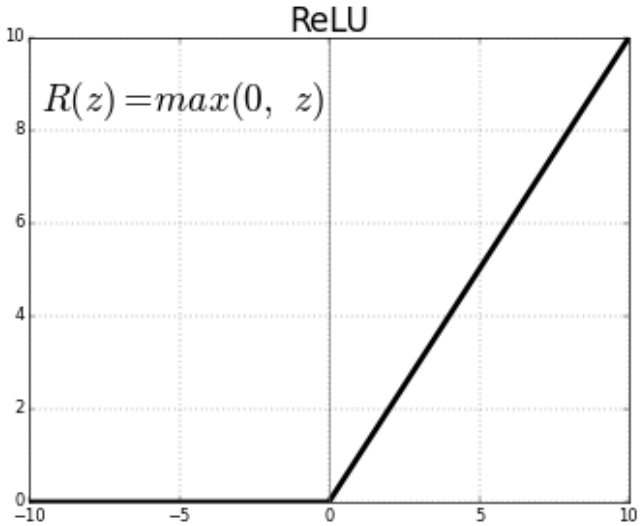
Residual networks as coordinate transformations

- A **residual network** is a sequential model which includes skip connections between layers
- With skip layers we avoid vanishing gradients by using activations from previous layers; layers are gradually expanded as opposed to forcibly connected



Residual networks as coordinate transformations (2)

- A residual network transforms coordinates as $x^{(l+1)} := \varphi^{(l)}(x^l) := x^{(l)} + f(x^{(l)}; l)$
- A residual network with ReLU activation is always piecewise linear with kinks of infinite curvature
- By piecing the patches together, we find a global coordinate system and thus the coordinate transformation is a bijection



Softmax output layer

How does the output layer fit into this paradigm?

- We can define the softmax coordinate transformation as $\text{softmax}(W^{(L)} \cdot x^{(L)})^j := e^{W^{(L)}{}_j x^{(L)}} \; / \; \sum_{k=1}^{K} e^{W^{(L)}{}_k x^{(L)}}$
- The probability that $q \in M$ being from class j is $P(Y = j | X = q) = \text{softmax}\,(W^{(L)} \cdot x^{(L)}(q))^j$

The importance of the metric tensor

- The **metric tensor** g is a function which takes as input a pair of tangent vectors v and w at a point on a manifold and produces a real scalar $g(v, w)$
- A **tangent vector** is a vector that is tangent to a surface at some point p
- The metric tensor generalizes the dot product for non-Euclidean space, and defines the length and angle between tangent vectors
- Now we can define and compute the length of curves on the manifold!

The importance of the metric tensor (2)

- A **positive-definite** metric tensor assigns a positive value $g(v, v) > 0$ to every nonzero vector v
- A manifold with a positive-definite metric tensor is a **Riemannian manifold**
- A Riemannian manifold is a **metric space** (i.e. it has a distance function $d(p, q)$ which defines the distance from p to q)
- The metric tensor is the derivative of the distance function
- The metric tensor in a coordinate basis is a symmetric matrix with entries that transform covariantly under changes to the coordinate system (i.e. change of basis)

The metric tensor as a loss function

- From the l-th layer to the $l+1$-th layer, we iterate through coordinate transformations:

$$g(x^{(l)})_{a_l b_l} = \left(\frac{\partial x^{(l+1)}}{\partial x^{(l)}}\right)^{a_{l+1}\cdot}_{\cdot a_l} \left(\frac{\partial x^{(l+1)}}{\partial x^{(l)}}\right)^{b_{l+1}\cdot}_{\cdot b_l} g(x^{(l+1)})_{a_{l+1} b_{l+1}}$$

- In the output coordinates we use Euclidean distance on a "flattened" representation of the data manifold
- The above is solved output-to-input; the coordinate representation of the metric tensor is **backpropagated** through the network

$$g(x^{(l)})_{a_l b_l} = \prod_{l'=L-1}^{l} \left[\left(\frac{\partial x^{(l'+1)}}{\partial x^{(l')}}\right)^{a_{l'+1}\cdot}_{\cdot a_{l'}} \left(\frac{\partial x^{(l'+1)}}{\partial x^{(l')}}\right)^{b_{l'+1}\cdot}_{\cdot b_{l'}}\right] \eta_{a_L b_L}$$

The metric tensor as a loss function (2)

- Let $\delta^{a^{l+1}\cdot}_{\cdot a_l}$ denote the Kronecker delta: $\delta_{\alpha\beta} := \begin{cases} 1 & : \alpha = \beta \\ 0 & : \alpha \neq \beta \end{cases}$
- For a residual network, we find the Jacobian of the coordinate transformation

$$\left(\frac{\partial x^{(l+1)}}{\partial x^{(l)}}\right)^{a_{l+1}\cdot}_{\cdot a_l} = \delta^{a_{l+1}\cdot}_{\cdot a_l} + \left(\frac{\partial f\left(x^{(l)}; l\right)}{\partial x^{(l)}}\right)^{a_{l+1}\cdot}_{\cdot a_l} \Delta l$$

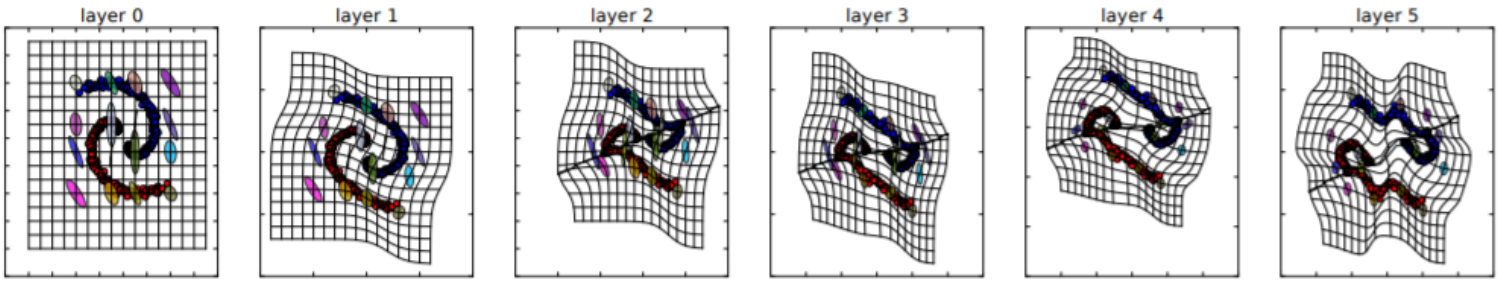
The metric tensor as a loss function (3)

- For a layer l, we take the sequence of matrix products from output to input to backpropagate the coordinate representation of the metric tensor, where $z^{(l+1)} := W^{(l)} \cdot x^{(l)} + b(l)$

$$P_{.a_l}^{a_L \cdot} := \prod_{l'=l}^{L-1} \left[\delta_{.a_{l'}}^{a_{l'+1} \cdot} + \left(\frac{\partial f(z^{(l'+1)}; l')}{\partial z^{(l'+1)}} \right)_{.e_{l'+1}}^{a_{l'+1} \cdot} \left(\frac{\partial z^{(l'+1)}}{\partial x^{(l')}} \right)_{.a_{l'}}^{e_{l'+1} \cdot} \Delta l \right]$$

The metric tensor as a loss function (4)

- With the output metric as the Euclidean distance η_{ab}, the linear element in the coordinate space for some layer l is $ds^2 = \eta_{ab} P^{a.}_{.a_l} P^{b.}_{.b_l} dx^{a_l} dx^{b_l}$
- This defines a δ-ball at layer l corresponding to the ϵ-ball in the output where the metric is Euclidean distance
- These balls form an $\epsilon - \delta$ relationship over the layers of the network, where the δ-ball at one layer maps to the ϵ-ball of the next layer

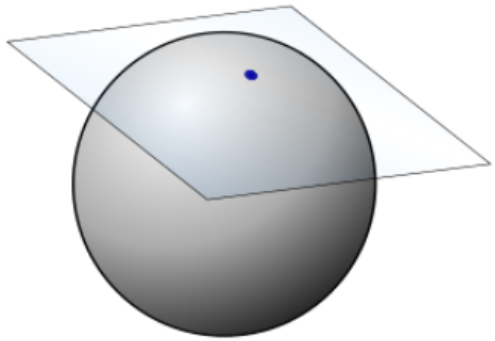


The pullback metric

- So far we have assumed constant layerwise dimension, but in practice the number of nodes per layer often changes
- We say the dimension of the data manifold is the dimension of the smallest layer of the neural network (i.e. $dimM := min_l dimx^{(l)}(M)$)
- Then all higher dimensional layers are immersion or embedding representations of this lowest dimensional representation
- Recall that embedding refers to a one structure contained within another
- Immersions are just embeddings where the derivative is injective on the tangent space at each point (i.e. the manifold is allowed to self-intersect)

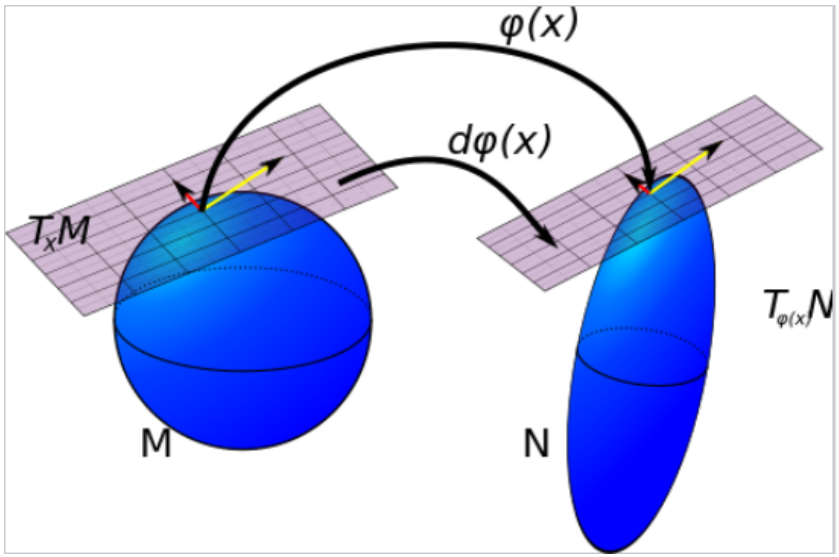
Tangent spaces

- A **tangent space** is a vector space that contains the directions in which one can tangentially pass through a point $x \in M$
- For topological manifolds M, N, $\varphi^{(I)} : M \longrightarrow N$ is a smooth map. Let TM, TN denote their tangent spaces
- Take $X \in TM$ where $X : \mathcal{C}^\infty(M) \longrightarrow \mathbb{R}$ and let $f \in \mathcal{C}^\infty(N)$



The pushforward map

- The **pushforward map** is the linear map $\varphi_*^{(I)} : TM \longrightarrow TN$ which takes an element $X \mapsto \varphi_*^{(I)}$
- Its action on f is $(\varphi_*^{(I)} X)(f) := X(f \circ \varphi(I))$
- Intuitively, it pushes tangent vectors on M forward to tangent vectors on N
- This can be viewed as the (total) derivative of φ



The pullback metric

- For some Riemannian manifolds M, N, $\varphi^{(l)} : M \longrightarrow N$ is a smooth map and $\varphi_*^{(l)} : TM \longrightarrow TN$ is the pushforward map between their tangent spaces
- The **pullback metric** g is defined as $g_M(X, Y) := g_N(\varphi_*^{(l)} X, \varphi_*^{(l)} Y)$ for all $X, Y \in TM$
- This is the linear map from the space of 1-forms on N (vector space of sections of the cotangent bundle) to the space of 1-forms on M
- This is backpropagation

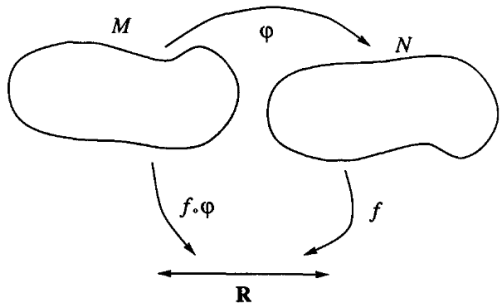
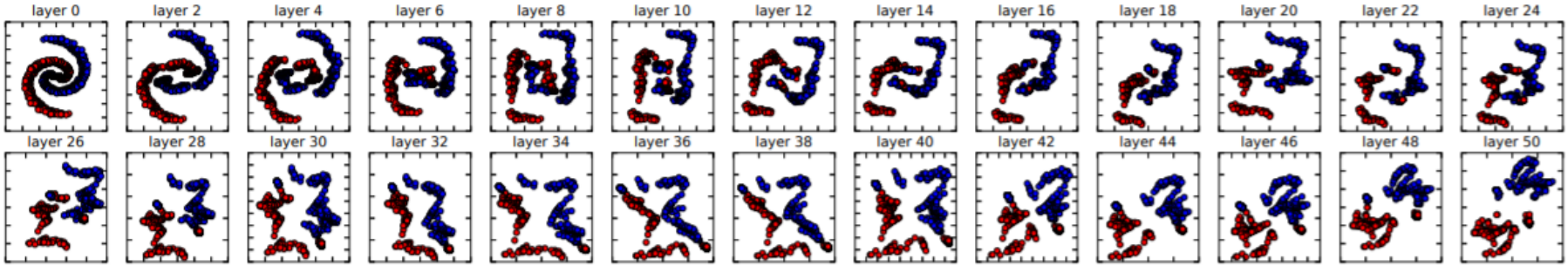


Fig. 6. Pulling back f from N to M

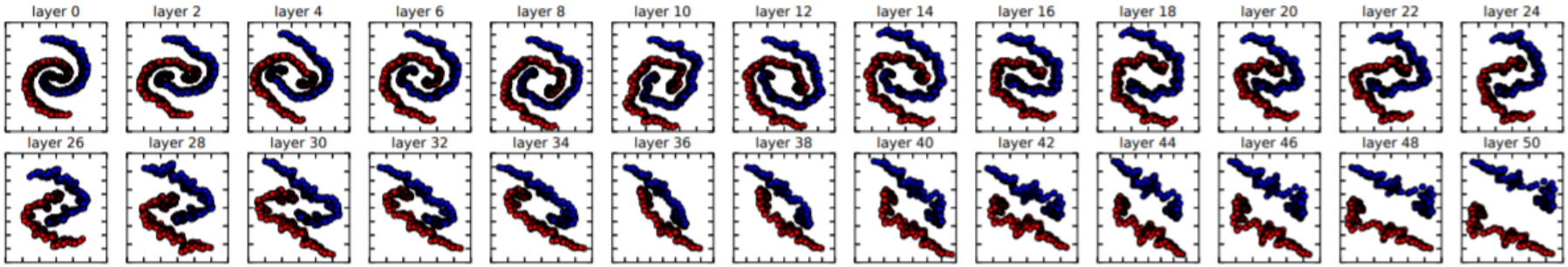
The importance of changing dimensions

- Since activation functions are limited, transforming the coordinate systems aids in discovering the optimal coordinate charts (systems) for the data manifold
- Higher dimensions make it easier to separate complex data

Empirical results

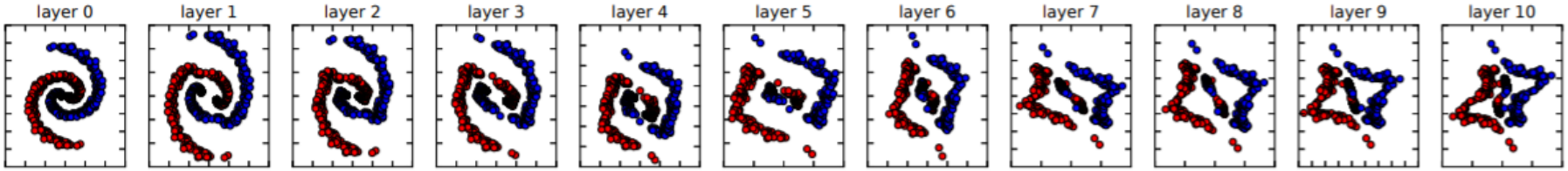


(a) A batch size of 300 for untangling data. As early as layer 4 the input connected sets have been disconnected and the data are untangled in an unintuitive way. This means a more complex coordinate representation of the data manifold was learned.

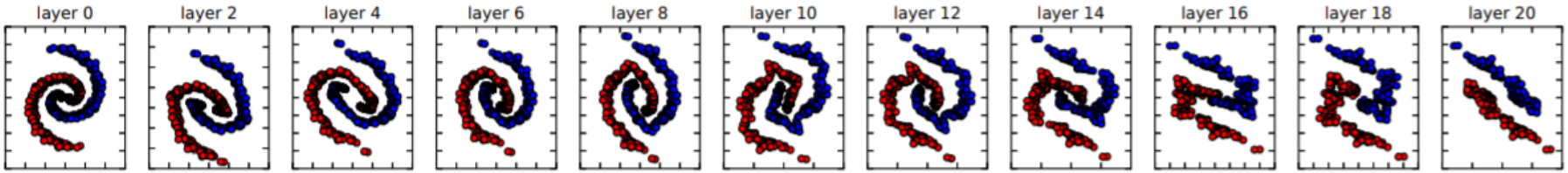


(b) A batch size of 1000 for untangling data. Because the large batch size can well-sample the data manifold, the spiral sets stay connected and are untangled in an intuitive way. This means a simple coordinate representation of the data manifold was learned.

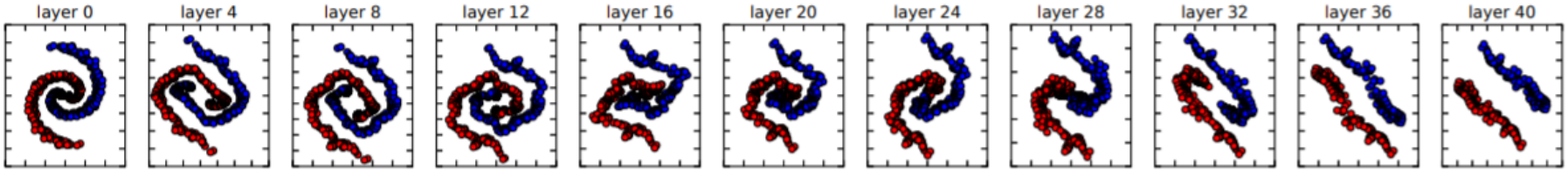
Empirical results (2)



(a) A 10 layer $\mathcal{C}^1$ network struggles to separate the spirals and has 1% error rate.



(b) A 20 layer $\mathcal{C}^1$ network is able to separate the spirals and has 0% error rate.



(c) A 40 layer $\mathcal{C}^1$ network is able to separate the spirals and has 0% error rate.

In conclusion,

at long last...

- A neural network starts with Cartesian coordinates
- It learns a sequence of non-linear coordinate transformations to find a coordinate representation of the data manifold which is representative
- This representation tends to be flat (experimentally)

In conclusion,

finally

- In the forward pass, we begin with Cartesian coordinates and apply differentiable coordinate transformations to find a nonlinear coordinate representation of the data manifold so the classes of the output coordinates satisfy the cost function
- In the backwards pass, we start with the Euclidean metric at the output and backpropagate the coordinate representation of the metric tensor (loss) through the network to find the metric tensor representation in the input Cartesian coordinates
- This defines an $\epsilon - \delta$ relationship between the input and output data

References

if you liked this, you may also like...

- `http://geometricdeeplearning.com/`
- An introduction to locally linear embeddings
  `https://cs.nyu.edu/~roweis/lle/papers/lleintro.pdf`
- Geometric deep learning: going beyond Euclidean data
  `https://arxiv.org/abs/1611.08097`