Neural Networks, Manifolds, and Topology

Posted on April 6, 2014

topology, neural networks, deep learning, manifold hypothesis

Recently, there's been a great deal of excitement and interest in deep neural networks because they've achieved breakthrough results in areas such as computer vision.[1]

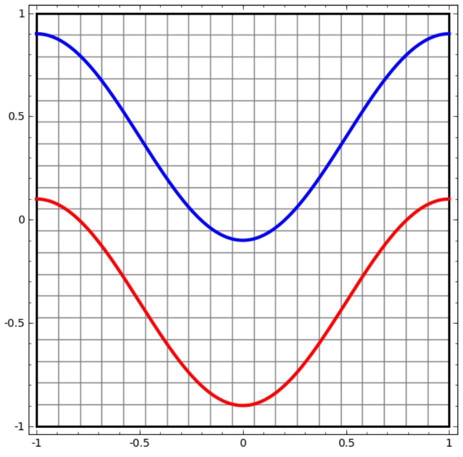
However, there remain a number of concerns about them. One is that it can be quite challenging to understand *what* a neural network is really doing. If one trains it well, it achieves high quality results, but it is challenging to understand how it is doing so. If the network fails, it is hard to understand what went wrong.

While it is challenging to understand the behavior of deep neural networks in general, it turns out to be much easier to explore low-dimensional deep neural networks – networks that only have a few neurons in each layer. In fact, we can create visualizations to completely understand the behavior and training of such networks. This perspective will allow us to gain deeper intuition about the behavior of neural networks and observe a connection linking neural networks to an area of mathematics called topology.

A number of interesting things follow from this, including fundamental lower-bounds on the complexity of a neural network capable of classifying certain datasets.

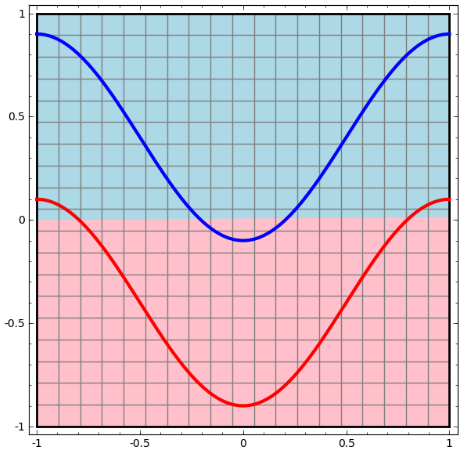
A Simple Example

Let's begin with a very simple dataset, two curves on a plane. The network will learn to classify points as belonging to one or the other.



The obvious way to visualize the behavior of a neural network – or any classification algorithm, for that matter – is to simply look at how it classifies every possible data point.

We'll start with the simplest possible class of neural network, one with only an input layer and an output layer. Such a network simply tries to separate the two classes of data by dividing them with a line.



That sort of network isn't very interesting. Modern neural networks generally have multiple layers between their input and output, called "hidden" layers. At the very least, they have one.

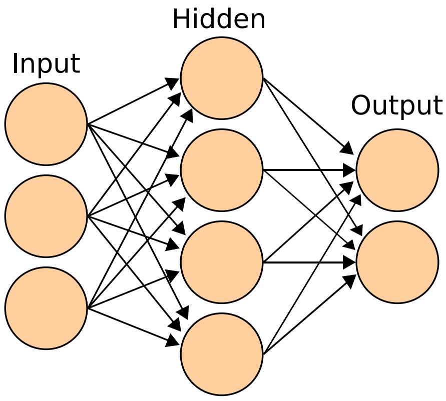
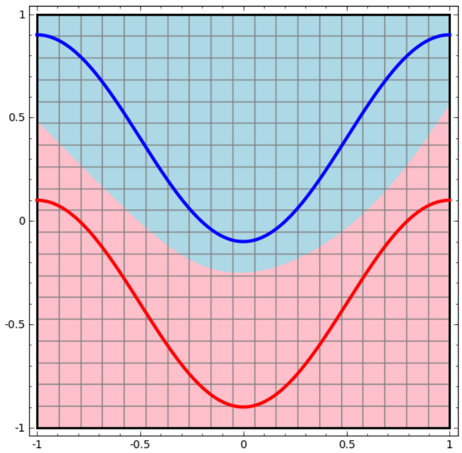


Diagram of a simple network from Wikipedia

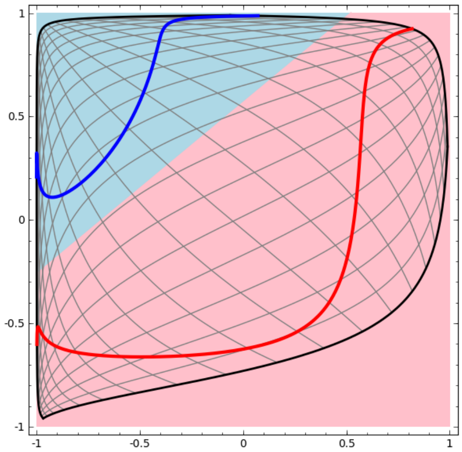
As before, we can visualize the behavior of this network by looking at what it does to different points in its domain. It separates the data with a more complicated curve than a line.



With each layer, the network transforms the data, creating a new *representation*.[2] We can look at the data in each of these representations and how the network classifies them. When we get to the final representation, the network will just draw a line through the data (or, in higher dimensions, a hyperplane).

In the previous visualization, we looked at the data in its “raw” representation. You can think of that as us looking at the input layer. Now we will look at it after it is transformed by the first layer. You can think of this as us looking at the hidden layer.

Each dimension corresponds to the firing of a neuron in the layer.



The hidden layer learns a representation so that the data is linearly separable

Continuous Visualization of Layers

In the approach outlined in the previous section, we learn to understand networks by looking at the representation corresponding to each layer. This gives us a discrete list of representations.

The tricky part is in understanding how we go from one to another. Thankfully, neural network layers have nice properties that make this very easy.

There are a variety of different kinds of layers used in neural networks. We will talk about tanh layers for a concrete example. A tanh layer $\tanh(Wx + b)$ consists of:

1. A linear transformation by the “weight” matrix W
2. A translation by the vector b
3. Point-wise application of tanh.

We can visualize this as a continuous transformation, as follows:

Gradually applying a neural network layer

The story is much the same for other standard layers, consisting of an affine transformation followed by pointwise application of a monotone activation function.

We can apply this technique to understand more complicated networks. For example, the following network classifies two spirals that are slightly entangled, using four hidden layers. Over time, we can see it shift from the “raw” representation to higher level ones it has learned in order to classify the data. While the spirals are originally entangled, by the end they are linearly separable.

On the other hand, the following network, also using multiple layers, fails to classify two spirals that are more entangled.

It is worth explicitly noting here that these tasks are only somewhat challenging because we are using low-dimensional neural networks. If we were using wider networks, all this would be quite easy.

(Andrej Karpathy has made a nice demo (http://cs.stanford.edu/people/karpathy/convnetjs//demo/classify2d.html) based on ConvnetJS that allows you to interactively explore networks with this sort of visualization of training!)

Topology of tanh Layers

Each layer stretches and squishes space, but it never cuts, breaks, or folds it. Intuitively, we can see that it preserves topological properties. For example, a set will be connected afterwards if it was before (and vice versa).

Transformations like this, which don't affect topology, are called homeomorphisms. Formally, they are bijections that are continuous functions both ways.

Theorem: Layers with N inputs and N outputs are homeomorphisms, if the weight matrix, W, is non-singular. (Though one needs to be careful about domain and range.)

Proof: Let's consider this step by step:

1. Let's assume W has a non-zero determinant. Then it is a bijective linear function with a linear inverse. Linear functions are continuous. So, multiplying by W is a homeomorphism.
2. Translations are homeomorphisms
3. tanh (and sigmoid and softplus but not ReLU) are continuous functions with continuous inverses. They are bijections if we are careful about the domain and range we consider. Applying them pointwise is a homemorphism

Thus, if W has a non-zero determinant, our layer is a homeomorphism. ∎

This result continues to hold if we compose arbitrarily many of these layers together.

Topology and Classification

Consider a two dimensional dataset with two classes $A, B \subset \mathbb{R}^2$:

A is red, B is blue

$$A = \{x | d(x, 0) < 1/3\}$$

$$B = \{x | 2/3 < d(x, 0) < 1\}$$

Claim: It is impossible for a neural network to classify this dataset without having a layer that has 3 or more hidden units, regardless of depth.

As mentioned previously, classification with a sigmoid unit or a softmax layer is equivalent to trying to find a hyperplane (or in this case a line) that separates A and B in the final represenation. With only two hidden units, a network is topologically incapable of separating the data in this way, and doomed to failure on this dataset.

In the following visualization, we observe a hidden representation while a network trains, along with the classification line. As we watch, it struggles and flounders trying to learn a way to do this.

For this network, hard work isn't enough.

In the end it gets pulled into a rather unproductive local minimum. Although, it's actually able to achieve $\sim 80\%$ classification accuracy.

This example only had one hidden layer, but it would fail regardless.

Proof: Either each layer is a homeomorphism, or the layer's weight matrix has determinant 0. If it is a homemorphism, A is still surrounded by B, and a line can't separate them. But suppose it has a determinant of 0: then the dataset gets collapsed on some axis. Since we're dealing with something homeomorphic to the original dataset, A is surrounded by B, and collapsing on any axis means we will have some points of A and B mix and become impossible to distinguish between. ∎

If we add a third hidden unit, the problem becomes trivial. The neural network learns the following representation:

With this representation, we can separate the datasets with a hyperplane.

To get a better sense of what's going on, let's consider an even simpler dataset that's 1-dimensional:

$$A = [-\frac{1}{3}, \frac{1}{3}]$$

$$B = [-1, -\frac{2}{3}] \cup [\frac{2}{3}, 1]$$

Without using a layer of two or more hidden units, we can't classify this dataset. But if we use one with two units, we learn to represent the data as a nice curve that allows us to separate the classes with a line:

What's happening? One hidden unit learns to fire when $x > -\frac{1}{2}$ and one learns to fire when $x > \frac{1}{2}$. When the first one fires, but not the second, we know that we are in A.

The Manifold Hypothesis

Is this relevant to real world data sets, like image data? If you take the manifold hypothesis really seriously, I think it bears consideration.

The manifold hypothesis is that natural data forms lower-dimensional manifolds in its embedding space. There are both theoretical[3] and experimental[4] reasons to believe this to be true. If you believe this, then the task of a classification algorithm is fundamentally to separate a bunch of tangled manifolds.

In the previous examples, one class completely surrounded another. However, it doesn't seem very likely that the dog image manifold is completely surrounded by the cat image manifold. But there are other, more plausible topological situations that could still pose an issue, as we will see in the next section.

Links And Homotopy

Another interesting dataset to consider is two linked tori, A and B.

Much like the previous datasets we considered, this dataset can't be separated without using $n + 1$ dimensions, namely a 4th dimension.

Links are studied in knot theory, an area of topology. Sometimes when we see a link, it isn't immediately obvious whether it's an unlink (a bunch of things that are tangled together, but can be separated by continuous deformation) or not.

A relatively simple unlink.

If a neural network using layers with only 3 units can classify it, then it is an unlink. (Question: Can all unlinks be classified by a network with only 3 units, theoretically?)

From this knot perspective, our continuous visualization of the representations produced by a neural network isn't just a nice animation, it's a procedure for untangling links. In topology, we would call it an *ambient isotopy* between the original link and the separated ones.

Formally, an ambient isotopy between manifolds A and B is a continuous function $F : [0,1] \times X \to Y$ such that each F_t is a homeomorphism from X to its range, F_0 is the identity function, and F_1 maps A to B. That is, F_t continuously transitions from mapping A to itself to mapping A to B.

Theorem: There is an ambient isotopy between the input and a network layer's representation if: a) W isn't singular, b) we are willing to permute the neurons in the hidden layer, and c) there is more than 1 hidden unit.

Proof: Again, we consider each stage of the network individually:

1. The hardest part is the linear transformation. In order for this to be possible, we need W to have a positive determinant. Our premise is that it isn't zero, and we can flip the sign if it is negative by switching two of the hidden neurons, and so we can guarantee the determinant is positive. The space of positive determinant matrices is path-connected (http://en.wikipedia.org/wiki/Connected_space#Path_connectedness), so there exists $p : [0,1] \to GL_n(\mathbb{R})$[5] such that $p(0) = Id$ and $p(1) = W$. We can continually transition from the identity function to the W transformation with the function $x \to p(t)x$, multiplying x at each point in time t by the continuously transitioning matrix $p(t)$.
2. We can continually transition from the identity function to the b translation with the function $x \to x + tb$.
3. We can continually transition from the identity function to the pointwise use of σ with the function: $x \to (1 - t)x + t\sigma(x)$. ∎

I imagine there is probably interest in programs automatically discovering such ambient isotopies and automatically proving the equivalence of certain links, or that certain links are separable. It would be interesting to know if neural networks can beat whatever the state of the art is there.

(Apparently determining if knots are trivial is NP. This doesn't bode well for neural networks.)

The sort of links we've talked about so far don't seem likely to turn up in real world data, but there are higher dimensional generalizations. It seems plausible such things could exist in real world data.

Links and knots are 1-dimensional manifolds, but we need 4 dimensions to be able to untangle all of them. Similarly, one can need yet higher dimensional space to be able to unknot n-dimensional manifolds. All n-dimensional manifolds can be untangled in $2n + 2$ dimensions.[6]

(I know very little about knot theory and really need to learn more about what's known regarding dimensionality and links. If we know a manifold can be embedded in n-dimensional space, instead of the dimensionality of the manifold, what limit do we have?)

The Easy Way Out

The natural thing for a neural net to do, the very easy route, is to try and pull the manifolds apart naively and stretch the parts that are tangled as thin as possible. While this won't be anywhere close to a genuine solution, it can achieve relatively high classification accuracy and be a tempting local minimum.

It would present itself as very high derivatives on the regions it is trying to stretch, and sharp near-discontinuities. We know these things happen.[7] Contractive penalties, penalizing the derivatives of the layers at data points, are the natural way to fight this.[8]

Since these sort of local minima are absolutely useless from the perspective of trying to solve topological problems, topological problems may provide a nice motivation to explore fighting these issues.

On the other hand, if we only care about achieving good classification results, it seems like we might not care. If a tiny bit of the data manifold is snagged on another manifold, is that a problem for us? It seems like we should be able to get arbitrarily good classification results despite this issue.

(My intuition is that trying to cheat the problem like this is a bad idea: it's hard to imagine that it won't be a dead end. In particular, in an optimization problem where local minima are a big problem, picking an architecture that can't genuinely solve the problem seems like a recipe for bad performance.)

Better Layers for Manipulating Manifolds?

The more I think about standard neural network layers – that is, with an affine transformation followed by a point-wise activation function – the more disenchanted I feel. It's hard to imagine that these are really very good for manipulating manifolds.

Perhaps it might make sense to have a very different kind of layer that we can use in composition with more traditional ones?

The thing that feels natural to me is to learn a vector field with the direction we want to shift the manifold:

And then deform space based on it:

One could learn the vector field at fixed points (just take some fixed points from the training set to use as anchors) and interpolate in some manner. The vector field above is of the form:

$$F(x) = \frac{v_0 f_0(x) + v_1 f_1(x)}{1 + f_0(x) + f_1(x)}$$

Where v_0 and v_1 are vectors and $f_0(x)$ and $f_1(x)$ are n-dimensional gaussians. This is inspired a bit by radial basis functions (http://en.wikipedia.org/wiki/Radial_basis_function).

K-Nearest Neighbor Layers

I've also begun to think that linear separability may be a huge, and possibly unreasonable, amount to demand of a neural network. In some ways, it feels like the natural thing to do would be to use k-nearest neighbors (knn) (k-NN). However, k-NN's success is greatly dependent on the representation it classifies data from, so one needs a good representation before k-NN can work well.

As a first experiment, I trained some MNIST networks (two-layer convolutional nets, no dropout) that achieved $\sim 1\%$ test error. I then dropped the final softmax layer and used the k-NN algorithm. I was able to consistently achieve a reduction in test error of 0.1-0.2%.

Still, this doesn't quite feel like the right thing. The network is still trying to do linear classification, but since we use k-NN at test time, it's able to recover a bit from mistakes it made.

k-NN is differentiable with respect to the representation it's acting on, because of the 1/distance weighting. As such, we can train a network directly for k-NN classification. This can be thought of as a kind of "nearest neighbor" layer that acts as an alternative to softmax.

We don't want to feedforward our entire training set for each mini-batch because that would be very computationally expensive. I think a nice approach is to classify each element of the mini-batch based on the classes of other elements of the mini-batch, giving each one a weight of 1/(distance from classification target).[9]

Sadly, even with sophisticated architecture, using k-NN only gets down to 5-4% test error – and using simpler architectures gets worse results. However, I've put very little effort into playing with hyper-parameters.

Still, I really aesthetically like this approach, because it seems like what we're "asking" the network to do is much more reasonable. We want points of the same manifold to be closer than points of others, as opposed to the manifolds being separable by a hyperplane. This should correspond to inflating the space between manifolds for different categories and contracting the individual manifolds. It feels like simplification.

Conclusion

Topological properties of data, such as links, may make it impossible to linearly separate classes using low-dimensional networks, regardless of depth. Even in cases where it is technically possible, such as spirals, it can be very challenging to do so.

To accurately classify data with neural networks, wide layers are sometimes necessary. Further, traditional neural network layers do not seem to be very good at representing important manipulations of manifolds; even if we were to cleverly set weights by hand, it would be challenging to compactly represent the transformations we want. New layers, specifically motivated by the manifold perspective of machine learning, may be useful supplements.

(This is a developing research project. It's posted as an experiment in doing research openly. I would be delighted to have your feedback on these ideas: you can comment inline or at the end. For typos, technical errors, or clarifications you would like to see added, you are encouraged to make a pull request on github (https://github.com/colah/NN-Topology-Post).)

Acknowledgments

Thank you to Yoshua Bengio, Michael Nielsen, Dario Amodei, Eliana Lorch, Jacob Steinhardt, and Tamsyn Waterhouse for their comments and encouragement.

---

1. This seems to have really kicked off with Krizhevsky *et al.*, (2012) (http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf), who put together a lot of different pieces to achieve outstanding results. Since then there's been a lot of other exciting work.↩
2. These representations, hopefully, make the data "nicer" for the network to classify. There has been a lot of work exploring representations recently. Perhaps the most fascinating has been in Natural Language Processing: the representations we learn of words, called word embeddings, have interesting properties. See Mikolov *et al.* (2013) (http://research.microsoft.com/pubs/189726/rvecs.pdf), Turian *et al.* (2010) (http://www.iro.umontreal.ca/~lisa/pointeurs/turian-wordrepresentations-acl10.pdf), and, Richard Socher's work

(http://www.socher.org/). To give you a quick flavor, there is a very nice visualization (http://metaoptimize.s3.amazonaws.com/cw-embeddings-ACL2010/embeddings-mostcommon.EMBEDDING_SIZE=50.png) associated with the Turian paper.↩

3. A lot of the natural transformations you might want to perform on an image, like translating or scaling an object in it, or changing the lighting, would form continuous curves in image space if you performed them continuously.↩
4. Carlsson *et al.* (http://comptop.stanford.edu/u/preprints/mumford.pdf) found that local patches of images form a klein bottle.↩
5. $GL_n(\mathbb{R})$ is the set of invertible $n \times n$ matrices on the reals, formally called the general linear group (http://en.wikipedia.org/wiki/General_linear_group) of degree n.↩
6. This result is mentioned in Wikipedia's subsection on Isotopy versions (http://en.wikipedia.org/wiki/Whitney_embedding_theorem#Isotopy_versions).↩
7. See Szegedy *et al.* (http://cs.nyu.edu/~zaremba/docs/understanding.pdf), where they are able to modify data samples and find slight modifications that cause some of the best image classification neural networks to misclasify the data. It's quite troubling.↩
8. Contractive penalties were introduced in contractive autoencoders. See Rifai *et al.* (2011) (http://www.iro.umontreal.ca/~lisa/pointeurs/ICML2011_explicit_invariance.pdf).↩
9. I used a slightly less elegant, but roughly equivalent algorithm because it was more practical to implement in Theano: feedforward two different batches at the same time, and classify them based on each other.↩

80 Comments (/posts/2014-03-NN-Manifolds-Topology/#disqus_thread)

80 Comments **colah's blog** 1 **Login**

Recommend 28 **Share** **Sort by Best**

Join the discussion…

Jan Rudy • 3 years ago

Your experiments with k-NN remind me of Neighbourhood Components Analysis. NCA attempts to learn a projection of the data space such that k-NN performs well. The original paper examined learning a linear projection, but this was followed up with a non-linear version.

https://papers.nips.cc/pape...
http://machinelearning.wust...

11 • Reply • Share ›

chrisolah Mod Jan Rudy • 3 years ago

That seems like a very similar -- or the same -- thing! I will need to look at that more. :)

Thanks for the references!

1 • Reply • Share ›

Brian Hayes • 3 years ago

"(Apparently determining if knots are trivial is NP. This doesn't bode well for neural networks.)"

Wolfgang Haken's algorithm for detecting the unknot is indeed exponential (in the number of crossings). [Haken, W. 1961. Theorie der Normalflachen. Acta Mathematica 105:245–375.] As far as I know, that's the only complete algorithm--proved to give a correct result in every case. But there are practical methods that work just fine on every knot you've ever seen. In particular, there's the Jones polynomial. [Jones, V.F.R. 1985. A polynomial invariant for knots via von Neumann algebra. Bull. Amer. Math. Soc. 12:103–111.] No one has yet found a knot that has the same Jones polynomial as the unknot. (On the other hand, there are links that fool the Jones polynomial.)

7 ^ | v • Reply • Share ›

Timothy O'Leary ➦ Brian Hayes • 3 years ago
But calculating the Jones Polynomial is #P hard
2 ^ | v • Reply • Share ›

Brian Hayes ➦ Timothy O'Leary • 3 years ago
Thanks for bringing that fact to my attention. I was unaware of it.
1 ^ | v • Reply • Share ›

Martin Beyerle • 3 years ago
very interesting read. I would be Interested in the Code for the Graphics, are they done in mathematica?
5 ^ | v • Reply • Share ›

mkrawier ➦ Martin Beyerle • 2 years ago
python with theano and matplotlib i'd guess
^ | v • Reply • Share ›

frankster • 3 years ago
Very interesting writeup - it looks like I'm going to have to study some topology!
5 ^ | v • Reply • Share ›

chrisolah Mod ➦ frankster • 3 years ago
Thanks!
^ | v • Reply • Share ›

Timothy Scharf • 3 years ago
Great stuff, but I am really going to need to see a visualization of the two tori unlinked in the 4th dimension. :)

Just really well done and the visualizations are outstanding
4 ^ | v • Reply • Share ›

Igor • 3 years ago
you might be interested in looking at this presentation; Avoiding pathologies in very deep networks by David Duvenaud, Oren Rippel,Ryan Adams, Zoubin Ghahramani at http://mlg.eng.cam.ac.uk/du...
4 ^ | v • Reply • Share ›

kram1032 ➦ Igor • a year ago
That link is dead but here is a working one instead: http://arxiv.org/pdf/1402.5...
^ | v • Reply • Share ›

geomblog • 3 years ago
I'm thinking about your points regarding increasing the number of hidden elements in a layer. Essentially this boils down to lifting the space to a higher dimension, and your particular example of the two concentric rings is one that's particularly easy to explain this way: for example, by mapping each point (x,y) to the paraboloid (x,y, x^2 + y^2) we can linearly separate the two rings.

More generally, kernels do precisely this kind of thing at grand scale, by effectively lifting to an infinite dimensional space (or in practice n dimensions). This corresponds to a really wide single hidden layer.

What this might suggest is that the places where NNs might have extra power is in the increased depth rather than width, which reinforces what you said in the post.
3 ^ | v • Reply • Share ›

chrisolah Mod ➦ geomblog • 3 years ago
One thing I think might be interesting would be to explore a "kernel" layer before softmax, that applies kernel methods in a learned representation...
^ | v • Reply • Share ›

geomblog ➦ chrisolah • 3 years ago
Yes. and while I can't now remember who did it. I thought there has been some work on this.

^ | v • Reply • Share ›

chrisolah Mod ➦ geomblog • 3 years ago
It's very possible! (If anyone has a reference handy, that would be lovely. Or I'll try to dig them up later. :))
^ | v • Reply • Share ›

geomblog ➦ chrisolah • 3 years ago
Here's at least one paper from NIPS 2009: http://cseweb.ucsd.edu/~sau...
^ | v • Reply • Share ›

geomblog ➦ geomblog • 3 years ago
Turns out there are at least a few more:

http://jmlr.org/papers/volu...

http://machinelearning.wust...
^ | v • Reply • Share ›

Jeremy Kun • 3 years ago
One thing I've always found suspicious about neural networks is the restriction of the underlying graph to layers, and the funneling and expanding through the nodes in those layers. It enables your analysis here to say every layer is a homeomorphism and the composition of homeomorphisms produce homeomorphisms, but a priori it appears to restrict the expressive power of the network.

Is there some theorem that says, "Every neural network is equivalent to one in layers with only a small number of extra neurons (say constant or logarithmic)"?

In the same vein, is there a theorem that classifies the family of homeomorphisms expressible by some neural network in some more tractable way?
3 ^ | v • Reply • Share ›

Tom Larkworthy • 3 years ago
I have not had much luck with homology methods on real data, they seem to assume the manifold is evenly sampled which is rarely the case. Really I think you need another measure on top of the manifold, like the sampling density (so a manifold field, or possibly a flow). Also in real systems the manifold dimension changes, so its really a complex. Urgh, so many problems...
3 ^ | v • Reply • Share ›

anthony_bak ➦ Tom Larkworthy • 3 years ago
Persistent homology doesn't have an assumption that the underlying space is a manifold - although there are theorems that guarantee recovering the correct homology given constraints on the sampling (recover in the sense that he most persistent cycles are the correct ones). However, there are examples where there is _no_ scale (for the rips filtration) at which all correct bars exist simultaneously. This says that persistence is the "right" way to do homology on point clouds and that trying to fix a scale is misguided.

A technical point for those delving into persistence. In my experience to get the information you want out of your data you should consider filtrations other than the rips (or nearest neighbor) filtration.

More importantly though when thinking about data problems geometrically is to realize that real world data is almost _never_ sampled from a manifold. While there are local constraints that give the data "shape" the shapes are more wild then an underlying manifold structure.

To me this says that topology (as opposed to geometry) is the "right" way to understand the shape of data. In particular, the "Mapper" algorithm produces topological summaries of the data that are very useful. It has been used to do "introspection" of neural networks as a way to open the "black box" to see what's going on.

Ok, so I'm tooting my own horn here. Here's a talk I gave at Stanford about how the topological summaries

see more

3 ^ | v • Reply • Share ›

Tom Larkworthy ↪ anthony_bak • 3 years ago
oh yeah exactly. It was the rips filtration that was giving me hassle. I did this work a few years back but there did not seem to be much off-the-shelf alternatives. I ruminated at the time that something with parametrized Guassians (e.g. LWPR) instead of hyperspheres would have worked better but I was not in the position for basic research and I moved on.

Thanks for the vid. I did come across Ayasdi before, its small world probably in the persister homology sphere. I'll watch it fully later as its biggie.
1 • Reply • Share ›

Tommy Levi ↪ anthony_bak • 3 years ago
Hi Anthony! I should mention that I used to sit next to him in a class on the Atiyah-Singer index theorem and algebraic topology. He's pretty damn smart.
• Reply • Share ›

chrisolah Mod ↪ Tom Larkworthy • 3 years ago
Yeah, I haven't tried to apply techniques like persistent homology yet, but that sounds like a pretty frustrating issue.

I'm hoping one might be able to do clever things by applying these techniques to the representations of neural networks. You can think of a neural network as learning a change of coordinates that makes the manifold "nicer". Hopefully, the data will be denser and in a lower-dimensional embedding space... (An autoencoder might be the right thing here.)
2 • Reply • Share ›

Tom Larkworthy ↪ chrisolah • 3 years ago
yeah you can probably simplify the neural network and have a better summery of the data ... in theory. Still, will be nigh impossible to visualise, damn this 3d universe. Also I recommend this book if you have not seen it ... Algebraic Geometry and Statistical Learning Theory
1 • Reply • Share ›

Horia • 3 years ago
I wanted to point out that, computationally, it is feasible to have very wide layers in such a feed-forward network. Essentially the hidden layer can be millions of units.

From your excellent write-up, I understand that specific edge cases, such as the linked tori, cannot be unlinked simply by geometric rotations and scaling because of the property of homeomorphism. At the same time, arbitrarily increasing the dimensionality of the space in which the hidden layers operate will result in a scenario similar to that which you describe as "the easy way out."

Pragmatically speaking, we are only interested in an epsilon-bounded approximation to the truly unlinked transformation. Could one use topology to prove a minimum number of needed dimensions to guarantee that the epsilon-bounds are able to be met?

I guess you'd have to know the manifold on which your data-set resides?
2 • Reply • Share ›

DanielM_113 ↪ Horia • 3 years ago
I suspect some methods can estimate the data dimensionality. This would be the first step.
• Reply • Share ›

catphive • 3 years ago
Do you have an RSS or atom feed? There doesn't seem to be the appropriate meta tag, or link on site.
2 • Reply • Share ›

chrisolah Mod ↪ catphive • 3 years ago
Not yet, it's coming! In the mean time, you can subscribe to my old blogs RSS feed: http://christopherolah.word... (I'll be cross-posting to it.)
2 • Reply • Share ›

chrisolah Mod ↪ catphive • 3 years ago
I now have an RSS feed! http://colah.github.io/rss.xml

• Reply • Share ›

Igor • 3 years ago
slides are here: http://mlg.eng.cam.ac.uk/du...
2 • Reply • Share ›

chrisolah Mod Igor • 3 years ago
Thanks, Igor!
• Reply • Share ›

rdrighetto • 3 years ago
Great stuff! I'd like to learn more about the relationship between manifolds and topology in the unsupervised case (i.e. detecting two or more low-d manifolds underlying your high-d data if they are unlabeled). If anyone can point me references on this, it would be nice :-)
1 • Reply • Share ›

chrisolah Mod rdrighetto • 3 years ago
You probably want to look into computational topology. In particular, there's a technique called persistent homology that allows one to extract topological structures.

You may also wish to look in to the field of manifold learning.
3 • Reply • Share ›

FU Yanwei • 3 years ago
Nice blog! Some random ideas: probably the parameter space of each layer is Riemann manifold. And there are some ideas in information geometry probably similar to yours.
http://link.springer.com/ar...
http://en.wikipedia.org/wik...

Not sure, I just start to learn deep learning.
1 • Reply • Share ›

Christopher Chatham • 3 years ago
Beautiful work.

Deep learning approaches are in some ways inspired by quite old observations about real neural microcircuits - namely the stochasticity of neurons that comprise the local (deep, 5 or 6 layer) structure.

But what about some of the more recent advances in computational neuroscience, namely the kind of universal macrocircuit structure we know exists between cortex and BG/thalamus, and are consistently disrupted in pathologies of higher-level cognition? These macrocircuits are widely acknowledged in the neuroscience community to implement discontinuities, often called "gating" in the motor and working memory literatures. It seems such mechanisms might prove useful for the more complex kinds of manifold surgeries that may be necessary to untangle knots or highly-intertwined geometries. But, as far as I know, these macrostructures have not been explored in artificial neural networks for modeling anything except for some of our most challenging cognitive tasks for humans (e.g., n-back).

Can anyone speak even in theory to their applicability to some of the manifold problems that prove challenging for even state-of-the-art maxout/drednet/whatever-nets?
1 • Reply • Share ›

saijanai • 3 years ago
You've been "reddited" and apparently a link was posted to hacker news so you can expect a lot of attention: http://www.reddit.com/r/sta...
1 • Reply • Share ›

Dylan Robert Cope • 11 days ago
Is there any equivalent intuition behind convolutional networks?
• Reply • Share ›

Dylan Robert Cope • 11 days ago
Is there a preferred way that I can properly cite this blog?

^ | v • Reply • Share ›

Jordan Micah Bennett • a month ago

This partially inspired me to explore something I call the 'super-manifold hypothesis' in deep causal reinforcement learning:

https://www.quora.com/What-...

https://www.academia.edu/31...

^ | v • Reply • Share ›

Andy T • 10 months ago

Hoping you've seen this :)

http://arxiv.org/pdf/1606.0...

^ | v • Reply • Share ›

Vida Vakilotojar • a year ago

Great article! I first discovered your excellent blog posts through links on TensorFlow's website, and I have learned a lot from them; very well written and quite thought provoking. Would appreciate your thoughts/feedback on something that I just posted at the following link, about my thoughts on how deep neural networks might be working, perhaps another way of looking at it.
http://vidavakil.github.io
Thanks in advance.

^ | v • Reply • Share ›

chrisolah Mod → Vida Vakilotojar • a year ago

Hi Vida,

I can't leave comments on your site, so I'll leave it here:

This is a nice way to think about neural networks. You might enjoy Michael Nielsen's visual proof of the universality of neural networks, which takes a similar approach: http://neuralnetworksanddee...

I'd be careful with your use of the word gradient. In the context of neural networks, people usually mean it in the sense of derivatives: https://en.wikipedia.org/wi...

^ | v • Reply • Share ›

Vida Vakilotojar → chrisolah • a year ago

Hi Christopher,

Thank you very much for responding. I took a quick look at Michael Nielsen's page, and yes indeed he has the same approach, and great visualizations that spare me form trying to replicate them.
I thought that it was interesting to make a connection between all the different, yet related, meanings of the word gradient (color gradient, slope, derivative), and that is why I used it, but I see your point.
Thanks again.

^ | v • Reply • Share ›

Ahmad El Sallab • a year ago

Is it the same thing as the SLM described here:

https://www.researchgate.ne...

^ | v • Reply • Share ›

Ahmad El Sallab • a year ago

Regarding the last section of K-NN, is it the same thing as this paper: https://www.researchgate.ne...
?

^ | v • Reply • Share ›

Nick Kersting • a year ago

Thank you! You've revolutionized my thinking with this : "an affine transformation followed by a point-wise activation function."

activation function

^ | v • Reply • Share ›

Lance Legel • 2 years ago

PS was just reading a publication from 2009 by Stanford Prof. Gunnar Carlsson, "Topology and Data", from the American Mathematical Society, and I thought you might find some inspiration in it: http://www.ams.org/journals...

Carlsson is a co-founder of Ayasdi (http://www.ayasdi.com), which just raised $55m driven by their "topological data analytics". They seem to be having great success in using topological pre-processing for machine learning: http://www.ayasdi.com/wp-co....

^ | v • Reply • Share ›

leonard Strnad • 2 years ago

Do you remember how many nodes at each hidden unit you used when separating the spiral example? It would be nice to know the exact structure for these examples. I am trying to duplicate results. Using three hidden layers, my network is very picky to properly learn the boundaries for the "target" data. Also, do you have any suggestions of how to visualize these representations?

^ | v • Reply • Share ›

chrisolah Mod → leonard Strnad • 2 years ago

For the spirals, I used two tanh units for each layer. That's why you can explicitly visualize it in two dimensions. I had to fiddle with regularization a bit. As I recall, a small about of L2 regularizing the weights towards the identity makes the mappings much nicer.

I did the visualizations with sage (see sagemath.org). In particular, you want to look at the functions: animate, contour_plot and parametric_plot.

^ | v • Reply • Share ›

Load more comments

Subscribe   Add Disqus to your site Add Disqus Add   Privacy