Validating Superpositions in Neural Networks

2022-10-04, Topological Machine Learning Seminar, U Barcelona

Julian Pfeifle

Universitat Politècnica de Catalunya

julian.pfeifle@upc.edu

Outline

1. What is the problem?

   How do neural networks learn?

2. What is the new input?

   The paper *"Toy Models of Superposition"* (*) (September 14, 2022) claims that

   > *"features come in direct sums"*

3. What will we talk about today?

   - How to evaluate these claims
     (Clustering using the Grassmannian, ...)
   - How to go beyond

(*) https://transformer-circuits.pub/2022/toy_model/index.html

How it began

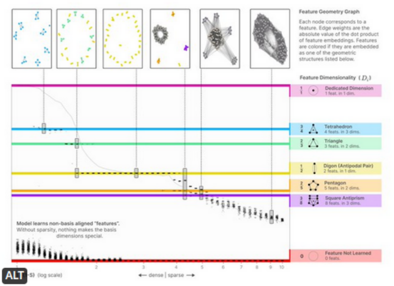
Chris Olah @ch402 · Sep 14

OK, I can buy that. But... oh dear.

Why is there a *tetrahedron* in my neural net???

What is going on???

> **Anthropic** @AnthropicAI · Sep 14
>
> Amazingly, features in superposition are organized into subspaces with geometry corresponding to regular polytopes (pentagons, tetrahedra) and other solutions to the Thomson problem. This creates discretized "energy levels" in the amount of dimensionality allocated to features.
>
> Show this thread



About this talk

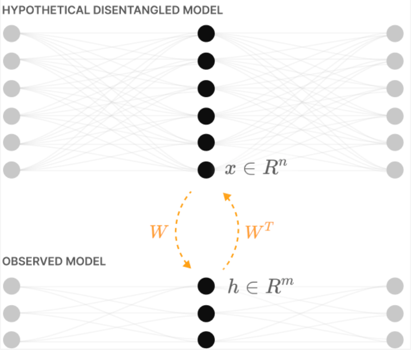
- I'm a discrete geometer
- I know very little about neural networks / ML

- I found this paper fascinating
- I would like to learn about neural networks / ML from you!

Setup

- ONB $x_1, \ldots, x_n$ of n "features" of sparsity S_i and importance I_i
- ONB $h_1, \ldots, h_m$ of $m \leq n$ hidden dimensions
- $m \times n$ projection matrix W: features $\mapsto$ hidden dimensions

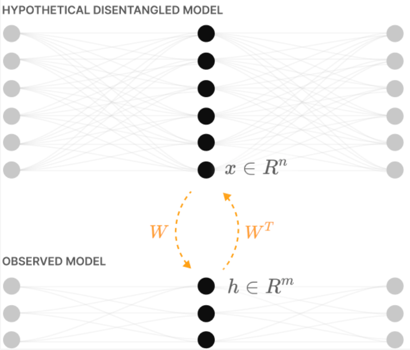
$$Wx = h$$



Setup

- ONB $x_1, \ldots, x_n$ of n "features" of sparsity S_i and importance I_i
- ONB $h_1, \ldots, h_m$ of $m \leq n$ hidden dimensions
- $m \times n$ projection matrix W: features $\mapsto$ hidden dimensions

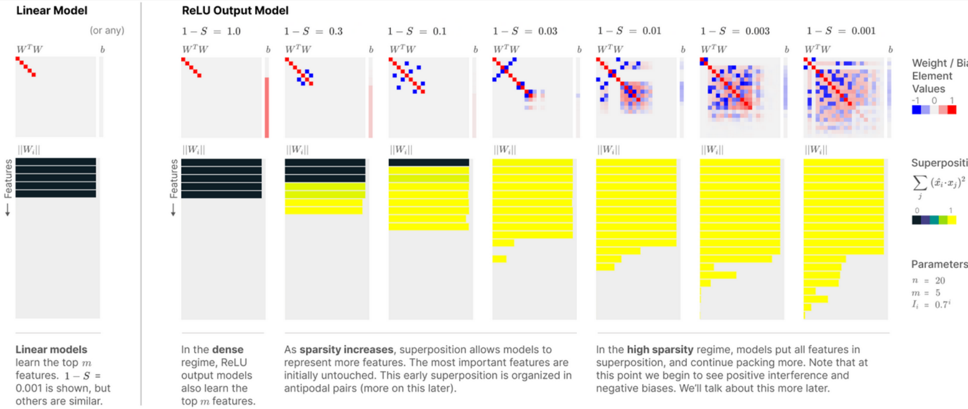
$$Wx = h$$



Each column $w_i \in \mathbb{R}^m$ of W is therefore a feature direction:

- w_i represents the feature $x_i \in \mathbb{R}^n$ in hidden-dimension space $\mathbb{R}^m$
- $|w_i|$ says how well feature is represented

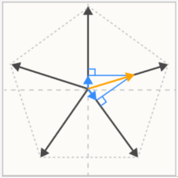
Sparsity



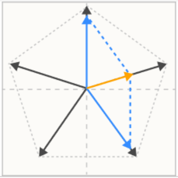
Sparsity S:

- probability that $x_i = 0$ when generating a random input vector.
- If x_i should not be zero, draw x_i uniformly from $[0, 1]$ *a bit weird...*

Features appear to come in direct sums



Even if only **one sparse feature** is active, using linear dot product projection on the superposition leads to **interference** which the model must tolerate or filter.



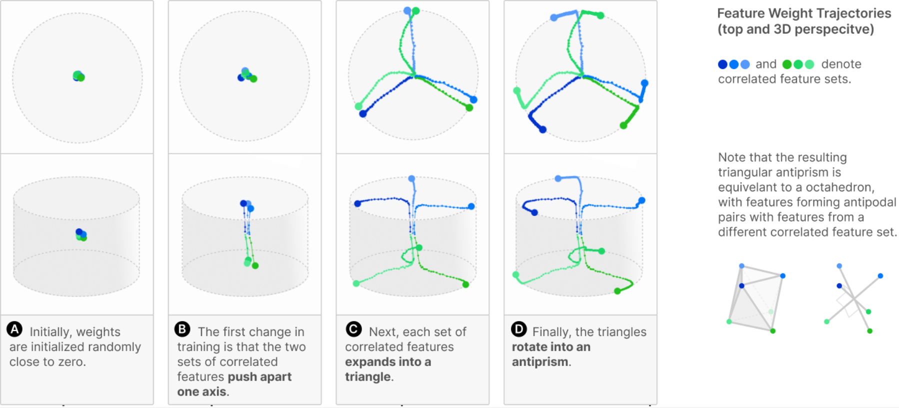
If the features aren't as sparse as a superposition is expecting, **multiple present features** can additively interfere such that there are multiple possible nonlinear reconstructions of an **activation vector**.

A triangular bipyramid is the tegum product of a triangle and an antipode. As a result, we observe 3×2/3 features and 2×1/2 features, rather than 6×3/5 featurs.

A pentagonal bipyramid is the tegum product of a pentagon and an antipode. As a result, we observe 5×2/5 features and 2×1/2 features, rather than 7×3/7 features.

An octahedron is the tegum product of three antipodes. This doesn't change the observed lines since 3/6=1/2.

Dynamics during training



Finding feature bundles in weight space, I

Summands of direct-sum subconfigurations

- Each summand consists of vectors in a k-plane, for some k
- We are looking for k-planes in $W = (w_1, w_2, \ldots, w_n) \subset \mathbb{R}^m$.

$(**)$

Finding feature bundles in weight space, I

Summands of direct-sum subconfigurations

- Each summand consists of vectors in a k-plane, for some k
- We are looking for k-planes in $W = (w_1, w_2, \ldots, w_n) \subset \mathbb{R}^m$.
- Hough Transform:(**)

for all $k = 1, \ldots, K$ **do**
  $L_k \leftarrow ()$
  for all $S \in \binom{W}{k}$ **do**
    find unique rep $\rho(S)$ of k-plane through S
    $L_k \leftarrow \mathrm{append\_to}(L_k, \rho(S))$
  Cluster the $\rho(S)$ in L_k
  Output best clusters

(**) https://en.wikipedia.org/wiki/Hough_transform

The Grassmannian

The moduli space of all k-dimensional subspaces of $\mathbb{R}^m$ is the Grassmannian $G(m,k)$.

- $\dim G(m,k) = k(m-k)$
- Plücker embedding in $\mathbb{RP}^{\binom{m}{k}}$, cut out by

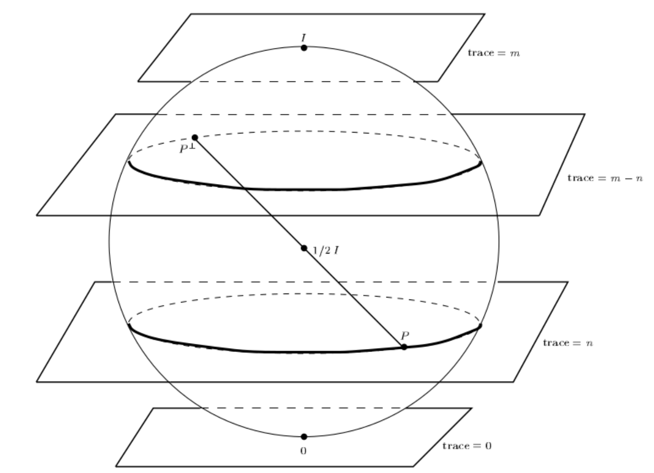
$$\sum_{j\in J} \operatorname{sgn}(j,I,J)\,[I\cup j]\,[J\setminus j] = 0 \quad \text{for } I \in \binom{[m]}{k-1},\ J \in \binom{[m]}{k+1}$$

- Projector matrix embedding in $\mathbb{R}^{D=\binom{m+1}{2}} \subset \mathbb{R}^{m\times m}$, cut out by

$$P^T = P, \qquad P^2 = P, \qquad \operatorname{trace} P = k.$$

  - Idea: P projects to the given subspace
  - $P = AA^\top$ from <u>any</u> $(m\times k)$ column-ONB A of the subspace
  - must column-reduce A for uniqueness of P!

The projector matrix embedding of $G(m,k)$



- ambient dim $D = \binom{m+1}{2}$
- $\mathbb{R}^D \subset \mathbb{R}^{m^2}$ $(P^\top = P)$
- not full-dim: $P^2 = P$
- $r_k = \sqrt{k(m-k)/m}$
- $R = \frac{1}{2}\sqrt{m}$
- trace $P = k$

Distance in this embedding ∼ "chordal distance"
good for clustering!

Conway-Hardin-Sloane 1996

How to cluster k-planes

- Start with $W = (w_1, \dots, w_n) \in \mathbb{R}^{m \times n}$.

  ▷ *Collect all projectors onto subspaces* ◁

  $L = ()$

  for all $k = 2, 3, \dots, K$ **do**

      for all $S \in \binom{[n]}{k}$ **do**

          $A = \text{col-red}(W_{\star,S})$ ▷ *make projector matrix unique*

          $L \leftarrow L \cup \text{vec}(AA^\top)$

  ▷ *Cluster and post-process them* ◁

  $C \leftarrow \text{db-scan}(L)$

  discard 1-element clusters

  discard pyramid clusters

- implemented in julia
  at https://gitlab.com/julian-upc/superpositions

How to cluster k-planes

The implementation

- works on synthetic examples
- needs to be hardened against perturbed examples.

Perturbation can bring about qualitatively different behavior:

- $\mathrm{perturb}_2\left(\text{n-gon}(r)\right) \oplus \mathrm{perturb}_2\left(\text{n-gon}(s)\right)$ works
- $\mathrm{perturb}_4\left(\text{n-gon}(r) \oplus \text{n-gon}(s)\right)$ has numerical stability issues

How to cluster k-planes

The implementation

- works on synthetic examples
- needs to be hardened against perturbed examples.

Perturbation can bring about qualitatively different behavior:

- $\mathrm{perturb}_2\left(\text{n-gon}(r)\right) \oplus \mathrm{perturb}_2\left(\text{n-gon}(s)\right)$ works
- $\mathrm{perturb}_4\left(\text{n-gon}(r) \oplus \text{n-gon}(s)\right)$ has numerical stability issues

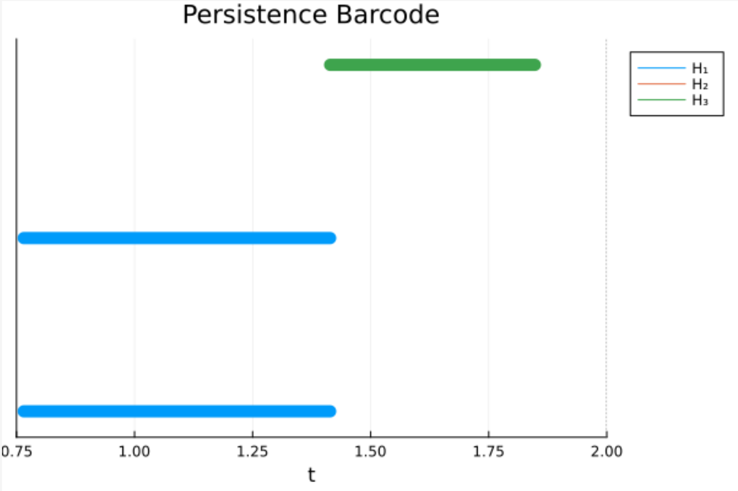
Reason: unique choice of representative!

- appearance of small non-zero entries
- brings about discrete change in pivot structure
- and discrete jumps in distance between representatives

Finding feature bundles in weight space, II

Let's use our favorite tool — TDA!!!

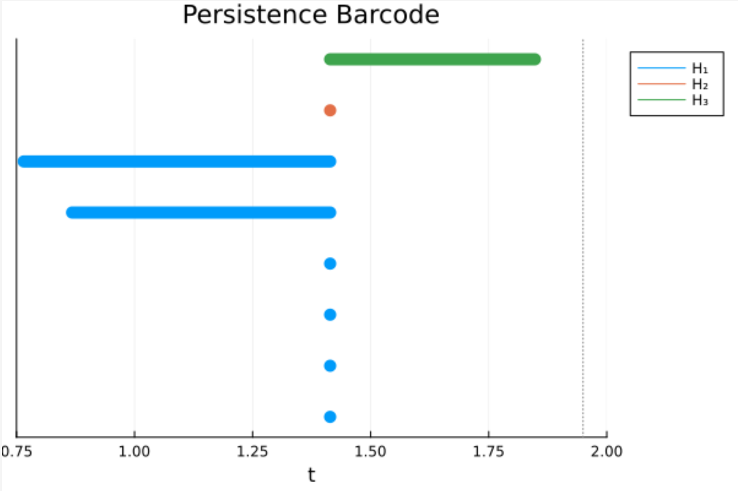
We are looking for spheres that are direct sums of smaller spheres!

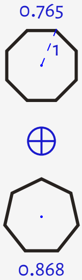


Finding feature bundles in weight space, II

Let's use our favorite tool — TDA!!!

We are looking for spheres that are direct sums of smaller spheres!

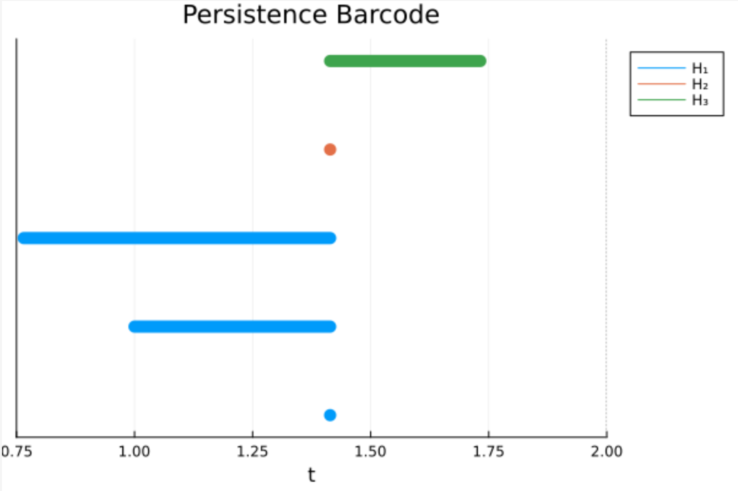


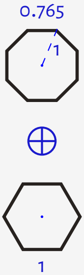


Finding feature bundles in weight space, II

Let's use our favorite tool — TDA!!!

We are looking for spheres that are direct sums of smaller spheres!

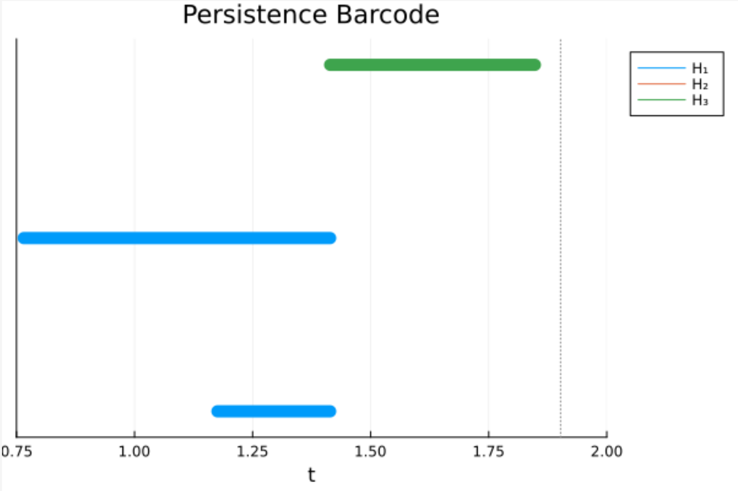




Finding feature bundles in weight space, II

Let's use our favorite tool — TDA!!!

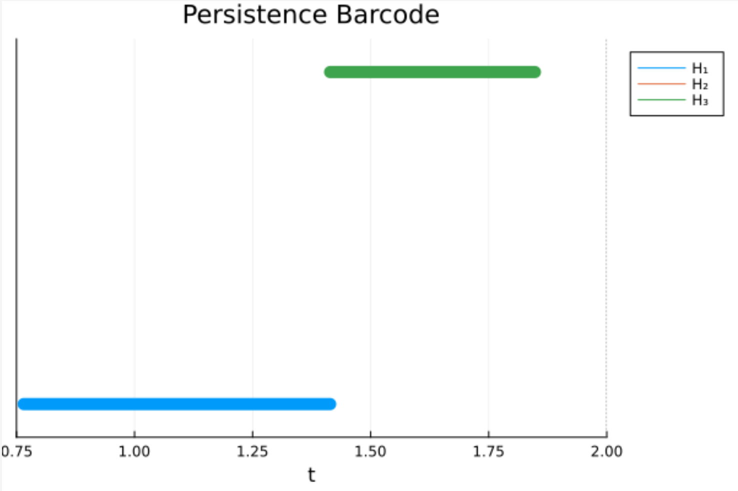
We are looking for spheres that are direct sums of smaller spheres!



Finding feature bundles in weight space, II

Let's use our favorite tool — TDA!!!

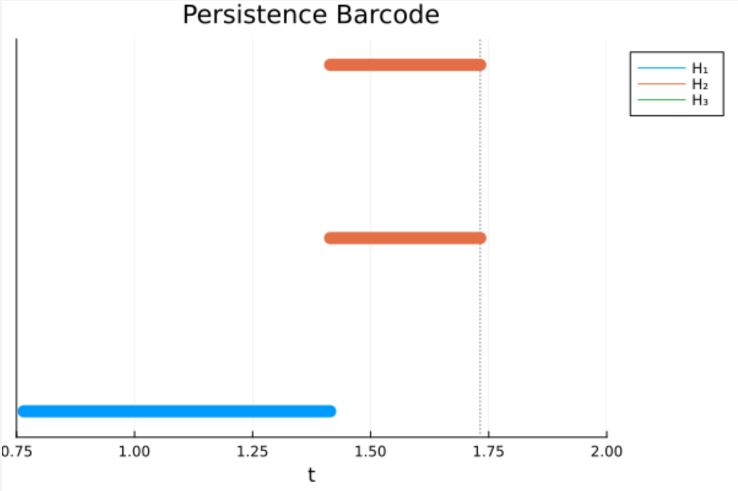
We are looking for spheres that are direct sums of smaller spheres!



Finding feature bundles in weight space, II

Let's use our favorite tool — TDA!!!

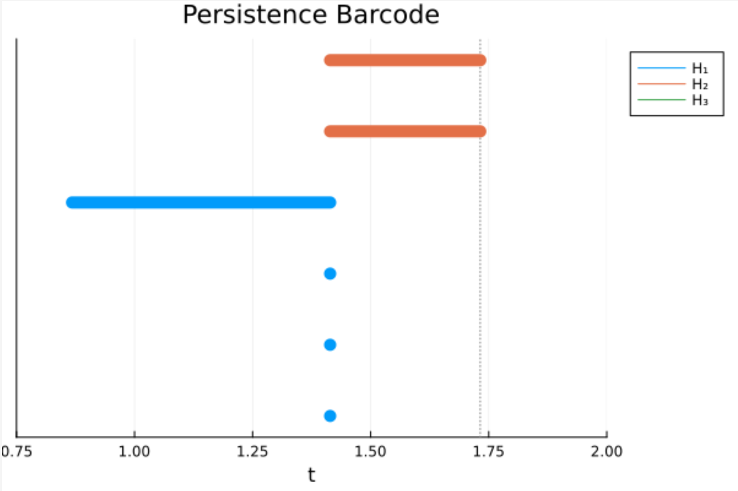
We are looking for spheres that are direct sums of smaller spheres!



Finding feature bundles in weight space, II

Let's use our favorite tool — TDA!!!

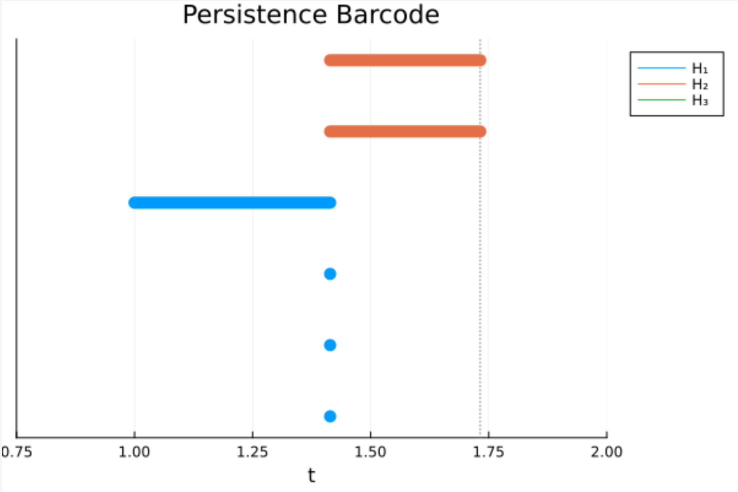
We are looking for spheres that are direct sums of smaller spheres!



Finding feature bundles in weight space, II

Let's use our favorite tool — TDA!!!

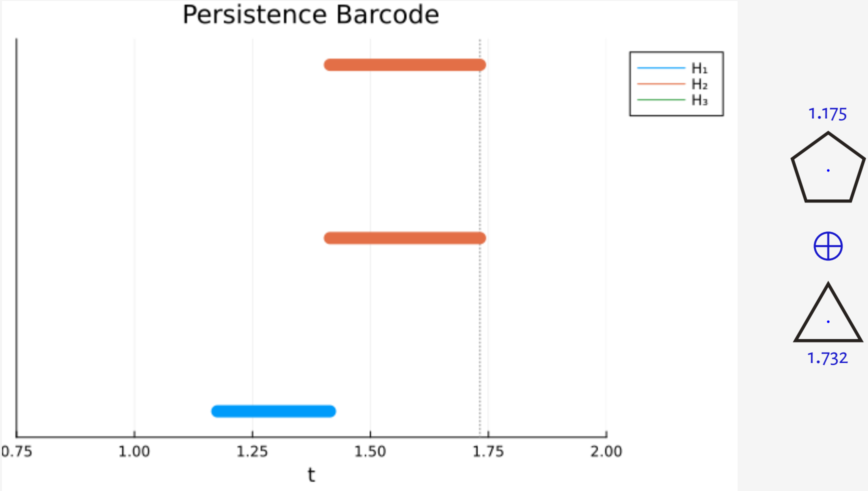
We are looking for spheres that are direct sums of smaller spheres!



Finding feature bundles in weight space, II

Let's use our favorite tool — TDA!!!

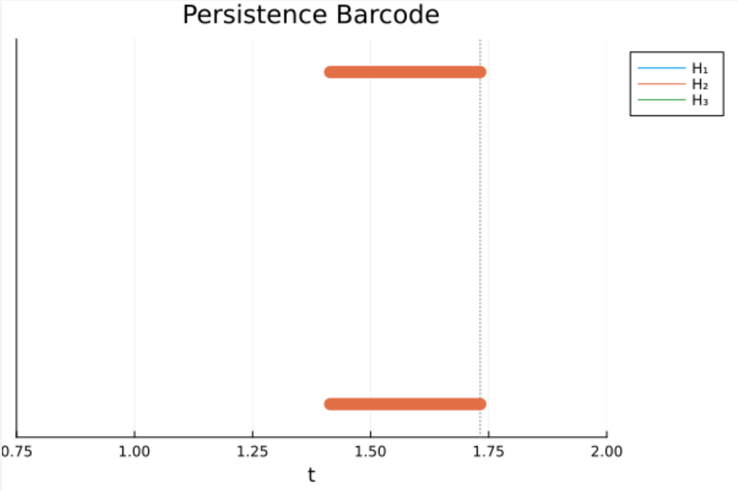
We are looking for spheres that are direct sums of smaller spheres!



Finding feature bundles in weight space, II

Let's use our favorite tool — TDA!!!

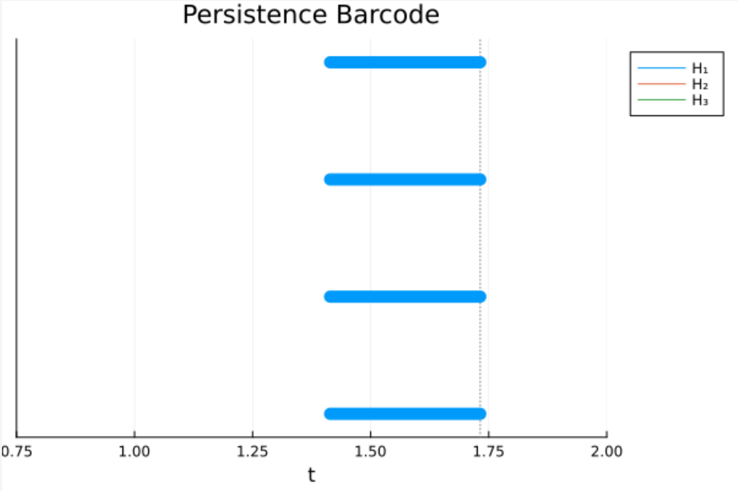
We are looking for spheres that are direct sums of smaller spheres!



Finding feature bundles in weight space, II

Let's use our favorite tool — TDA!!!

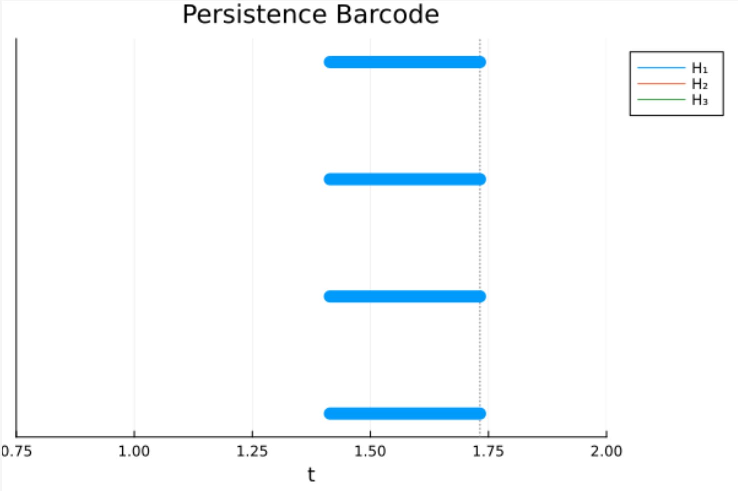
We are looking for spheres that are direct sums of smaller spheres!



Finding feature bundles in weight space, II

Let's use our favorite tool — TDA!!!

We are looking for spheres that are direct sums of smaller spheres!



. . . MEH

Finding feature bundles in weight space, III

Perhaps spheres are a better goal

- The authors of [Toy] actually think that feature vectors make spherical codes, i.e., points maximally apart on a fixed low-dimensional sphere
- Sometimes, these codes decompose into direct sums

Finding feature bundles in weight space, III

Perhaps spheres are a better goal

- The authors of [Toy] actually think that feature vectors make spherical codes, i.e., points maximally apart on a fixed low-dimensional sphere
- Sometimes, these codes decompose into direct sums

Example

2-sphere through 4 points: Expand the first row of

$$\begin{vmatrix} x^2+y^2+z^2 & x & y & z & 1 \\ x_1^2+y_1^2+z_1^2 & x_1 & y_1 & z_1 & 1 \\ x_2^2+y_2^2+z_2^2 & x_2 & y_2 & z_2 & 1 \\ x_3^2+y_3^2+z_3^2 & x_3 & y_3 & z_3 & 1 \\ x_4^2+y_4^2+z_4^2 & x_4 & y_4 & z_4 & 1 \end{vmatrix} = 0$$

Question

Find a good distance measure to represent & cluster these spheres

Correlation, anti-correlation, …and then?

In [Toy], the authors observe that

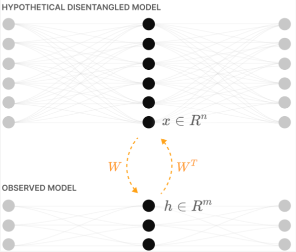
- correlated features combine in one summand
- anti-correlated features combine in the other summand

of a 2-component direct sum.

Question

What happens for sums with 3 or more components?

Revisiting $W^\top$



To recover the original vector, we'll use the transpose of the same matrix W^T. This has the advantage of avoiding any ambiguity regarding what direction in the lower-dimensional space really corresponds to a feature. It also seems relatively mathematically principled[9], and empirically works.

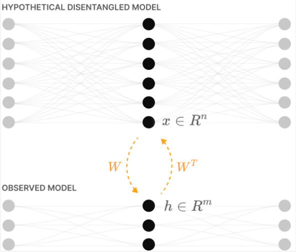
Recall that $W^T = W^{-1}$ if W is orthonormal. Although W can't be literally orthonormal, our intuition from compressed sensing is that it will be "almost orthonormal" in the sense of Candes & Tao [25]. [↩]

This is a very weak excuse:

- W is very far from being even square (necessary for orthogonality)
- Even the columns of W are very far from being orthogonal
  (That's the whole point of superposition)

So...why do they use $W^\top$?

Revisiting $W^\top$



To recover the original vector, we'll use the transpose of the same matrix W^T. This has the advantage of avoiding any ambiguity regarding what direction in the lower-dimensional space really corresponds to a feature. It also seems relatively mathematically principled [9], and empirically works.

Recall that $W^T = W^{-1}$ if W is orthonormal. Although W can't be literally orthonormal, our intuition from compressed sensing is that it will be "almost orthonormal" in the sense of Candes & Tao [25]. [↩]

This is a very weak excuse:

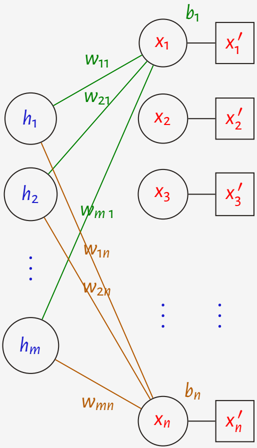
- W is very far from being even square (necessary for orthogonality)
- Even the columns of W are very far from being orthogonal
  (That's the whole point of superposition)

So...why do they use $W^\top$?

Because $W^\top$ encodes a neural net that reconstructs x given h!

The affine setting: incorporating bias and activation function

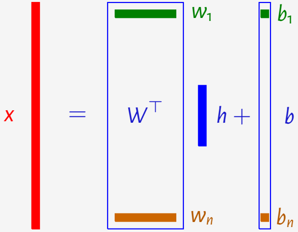
The model in [Toy] is $x' = \mathrm{ReLU}(\underbrace{W^\top h + b}_{x})$



$$x_1 = w_{11}h_1 + w_{21}h_2 + \cdots + w_{m1}h_m + b_1 = \langle w_1, h\rangle + b_1 = \langle \overline{w_1}, \overline{h}\rangle$$

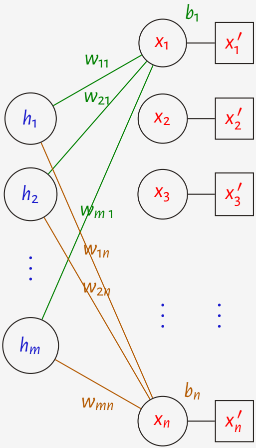
$$\vdots$$

$$x_n = w_{1n}h_1 + w_{2n}h_2 + \cdots + w_{mn}h_m + b_1 = \langle w_n, h\rangle + b_n = \langle \overline{w_n}, \overline{h}\rangle$$

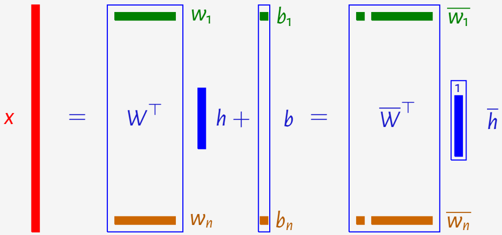


The affine setting: incorporating bias and activation function

The model in [Toy] is $x' = \mathrm{ReLU}(\underbrace{W^\top h + b}_{x}) = \mathrm{ReLU}(\overline{W}^\top \overline{h})$



$$x_1 = w_{11}h_1 + w_{21}h_2 + \cdots + w_{m1}h_m + b_1 = \langle w_1, h\rangle + b_1 = \langle \overline{w_1}, \overline{h}\rangle$$
$$\vdots$$
$$x_n = w_{1n}h_1 + w_{2n}h_2 + \cdots + w_{mn}h_m + b_1 = \langle w_n, h\rangle + b_n = \langle \overline{w_n}, \overline{h}\rangle$$



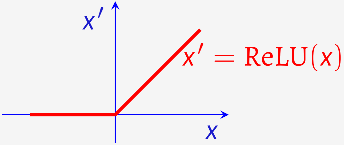
Decision boundaries

The final step is whether to add an activation function. This turns out to be critical to whether superposition occurs. In a real neural network, when features are actually used by the model to do computation, there will be an activation function, so it seems principled to include one at the end.

$$x' = \mathrm{ReLU}(W^\top h + b) = \mathrm{ReLU}\left(\overline{W}^\top \overline{h}\right)$$

$$x_1 = \langle w_1, h\rangle + b_1 = \langle \overline{w_1}, \overline{h}\rangle$$

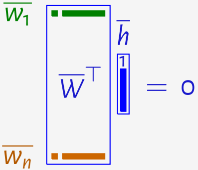
$$x_n = \langle w_n, h\rangle + b_n = \langle \overline{w_n}, \overline{h}\rangle$$



The decision boundaries

$$\{\overline{h} \in \mathbb{R}^{m+1} : \langle \overline{w_i}, \overline{h}\rangle = 0\}$$

form an affine hyperplane arrangement



Understanding the loss function: $L = \sum_x \sum_i I_i (x_i - x_i')^2$

linear: $$L \sim \sum_i I_i(1 - ||W_i||^2)^2 + \sum_{i \neq j} I_j (W_j \cdot W_i)^2$$

Feature benefit is the value a model attains from representing a feature. In a real neural network, this would be analagous to the potential of a feature to improve predictions if represented accurately.

Interference between x_i and x_j occurs when two features are embedded non-orthogonally and, as a result, affect each other's predictions. This prevents superposition in linear models.

L_1 : $$L_1 = \sum_i \int_{0 \leq x_i \leq 1} I_i (x_i - \mathrm{ReLU}(||W_i||^2 x_i + b_i))^2 + \sum_{i \neq j} \int_{0 \leq x_i \leq 1} I_j \mathrm{ReLU}(W_j \cdot W_i x_i + b_j)$$

If we focus on the case $x_i = 1$ *, we get something which looks even more analagous to the linear case:*

$$= \sum_i I_i (1 - \mathrm{ReLU}(||W_i||^2 + b_i))^2 + \sum_{i \neq j} I_j \mathrm{ReLU}(W_j \cdot W_i + b_j)^2$$

Feature benefit is similar to before. Note that ReLU never makes things worse, and that the bias can help when the model doesn't represent a feature by taking on the expected value.

Interference is similar to before but ReLU means that negative interference, or interference where a negative bias pushes it below zero, is "free" in the 1-sparse case.

Understanding the loss function

L_1 :

$$L_1 = \sum_i \int_{0 \le x_i \le 1} I_i (x_i - \mathrm{ReLU}(||W_i||^2 x_i + b_i))^2 \quad + \sum_{i \neq j} \int_{0 \le x_i \le 1} I_j \mathrm{ReLU}(W_j \cdot W_i x_i + b_j)$$

If we focus on the case $x_i = 1$ *, we get something which looks even more analagous to the linear case:*

$$= \sum_i I_i (1 - \mathrm{ReLU}(||W_i||^2 + b_i))^2 \quad + \sum_{i \neq j} I_j \mathrm{ReLU}(W_j \cdot W_i + b_j)^2$$

Feature benefit is similar to before. Note that ReLU never makes things worse, and that the bias can help when the model doesn't represent a feature by taking on the expected value.

Interference is similar to before but ReLU means that negative interference, or interference where a negative bias pushes it below zero, is "free" in the 1-sparse case.

- If $I_i = 1$, $|w_i| = 1$ and $b_i = 0$ for all i: Thomson problem (minimizing the potential energy of charged particles on a sphere)

Understanding the loss function

L_1 :

$$L_1 = \sum_i \int_{0 \le x_i \le 1} I_i(x_i - \mathrm{ReLU}(||W_i||^2 x_i + b_i))^2 + \sum_{i \neq j} \int_{0 \le x_i \le 1} I_j \mathrm{ReLU}(W_j \cdot W_i x_i + b_j)$$

If we focus on the case $x_i = 1$, we get something which looks even more analagous to the linear case:

$$= \sum_i I_i(1 - \mathrm{ReLU}(||W_i||^2 + b_i))^2 + \sum_{i \neq j} I_j \mathrm{ReLU}(W_j \cdot W_i + b_j)^2$$

Feature benefit is similar to before. Note that ReLU never makes things worse, and that the bias can help when the model doesn't represent a feature by taking on the expected value.

Interference is similar to before but ReLU means that negative interference, or interference where a negative bias pushes it below zero, is "free" in the 1-sparse case.

- If $I_i = 1, |w_i| = 1$ and $b_i = 0$ for all i: Thomson problem (minimizing the potential energy of charged particles on a sphere)

Question

Why do spherical codes apparently also appear in the general case?

Wrap-up

- Dynamics of learning! Haven't said anything
- Finding direct sums in existing networks:
  - Harden the Grassmannian reconstruction against $O_k(\mathbb{R})$-action
  - TDA probably not a good fit
  - Find a good distance measure to represent and cluster spheres
- Analyze large direct sums in terms of anti/correlation
- Figure out what makes direct-sum hyperplane arrangements special
  - Are they minima for training?
  - Role of “sparsity”?
- Compose two or more layers of such components
  - for example, adding two nodes for binary classification
  - adding another whole component

The loss function

The loss function

linear: $$L \sim \sum_i I_i(1 - ||W_i||^2)^2 + \sum_{i \neq j} I_j(W_j \cdot W_i)^2$$

Feature benefit is the value a model attains from representing a feature. In a real neural network, this would be analagous to the potential of a feature to improve predictions if represented accurately.

Interference betwen x_i and x_j occurs when two features are embedded non-orthogonally and, as a result, affect each other's predictions. This prevents superposition in linear models.

ReLU: $L = \int_x ||I(x - \mathrm{ReLU}(W^T W x + b))||^2 d\mathbf{p}(x)$ where x is distributed such that $x_i = 0$ with probability S.

The integral over x decomposes into a term for each sparsity pattern according to the binomial expansion of $((1-S) + S)^n$. We can group terms of the sparsity together, rewriting the loss as $L = (1-S)^n L_n + \ldots + (1-S)S^{n-1}L_1 + S^n L_0$, with each L_k corresponding to the loss when the input is a k-sparse vector. Note that as $S \to 1$, L_1 and L_0 dominate. The L_0 term, corresponding to the loss on a zero vector, is just a penalty on positive biases, $\sum_i \mathrm{ReLU}(b_i)^2$. So the interesting term is L_1, the loss on 1-sparse vectors: