Superposition in neural networks

By Elhage, Hume, Olsson, Schiefer, et al.

Background

- What are representations?
    - Features of the input represented as directions in activation space
- What do representations look like?
- Linear representation hypothesis:
    - Decomposability
    - Linearity
- How and why do we get good representations?
    - Privileged basis
    - Superposition

Goal

- Demonstrate how superposition interacts with privileged bases

- Importance can not be understated
  - Huge implications about what interpretability approaches make sense

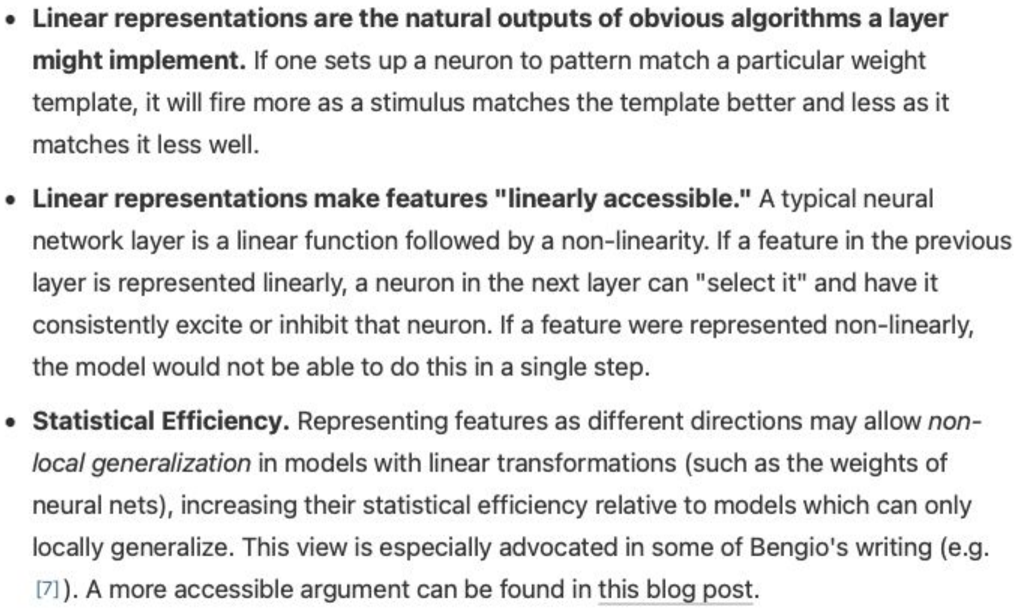
Empirical observations

- Word embedding arithmetic
- Latent spaces with interpretable directions
- Universality of neurons
- Polysmeantic neurons

Empirical observations

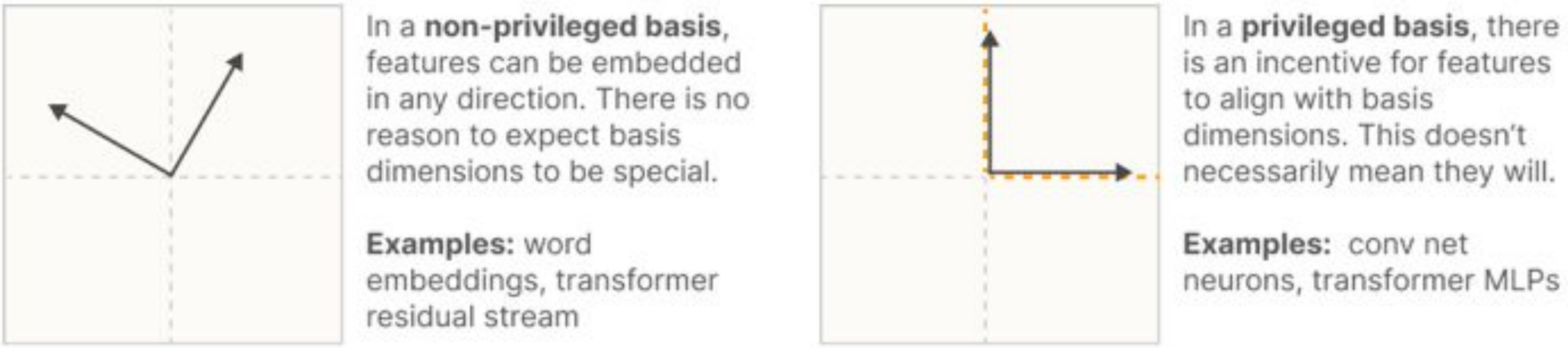
- Word embedding arithmetic
- Latent spaces with interpretable directions
- Universality of neurons
- Polysmeantic neurons

Linearity



- **Linear representations are the natural outputs of obvious algorithms a layer might implement.** If one sets up a neuron to pattern match a particular weight template, it will fire more as a stimulus matches the template better and less as it matches it less well.
- **Linear representations make features "linearly accessible."** A typical neural network layer is a linear function followed by a non-linearity. If a feature in the previous layer is represented linearly, a neuron in the next layer can "select it" and have it consistently excite or inhibit that neuron. If a feature were represented non-linearly, the model would not be able to do this in a single step.
- **Statistical Efficiency.** Representing features as different directions may allow *non-local generalization* in models with linear transformations (such as the weights of neural nets), increasing their statistical efficiency relative to models which can only locally generalize. This view is especially advocated in some of Bengio's writing (e.g. [7]). A more accessible argument can be found in this blog post.

Basis privilege



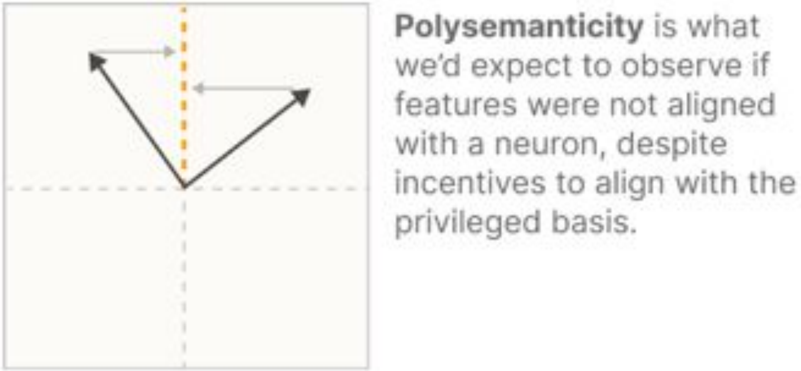
In a **non-privileged basis**, features can be embedded in any direction. There is no reason to expect basis dimensions to be special.

Examples: word embeddings, transformer residual stream

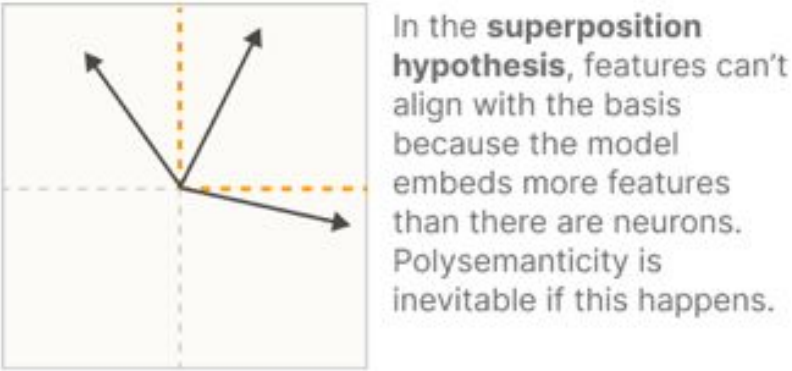
In a **privileged basis**, there is an incentive for features to align with basis dimensions. This doesn't necessarily mean they will.

Examples: conv net neurons, transformer MLPs

Superposition hypothesis



Polysemanticity is what we'd expect to observe if features were not aligned with a neuron, despite incentives to align with the privileged basis.

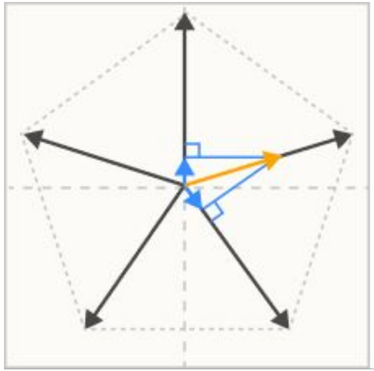


In the **superposition hypothesis**, features can't align with the basis because the model embeds more features than there are neurons. Polysemanticity is inevitable if this happens.

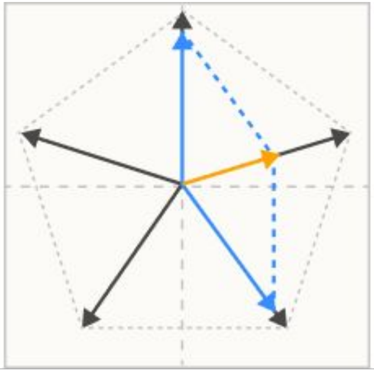
Superposition hypothesis

- Almost orthogonal vectors: exp(n) ‘almost orthogonal’ vectors in high-dimensional spaces. Compared to n orthogonal ones
- Compressed sensing: lower-dimensional mapping can not fully reconstruct, unless we know lower-dimensional space is sparse

Superposition hypothesis

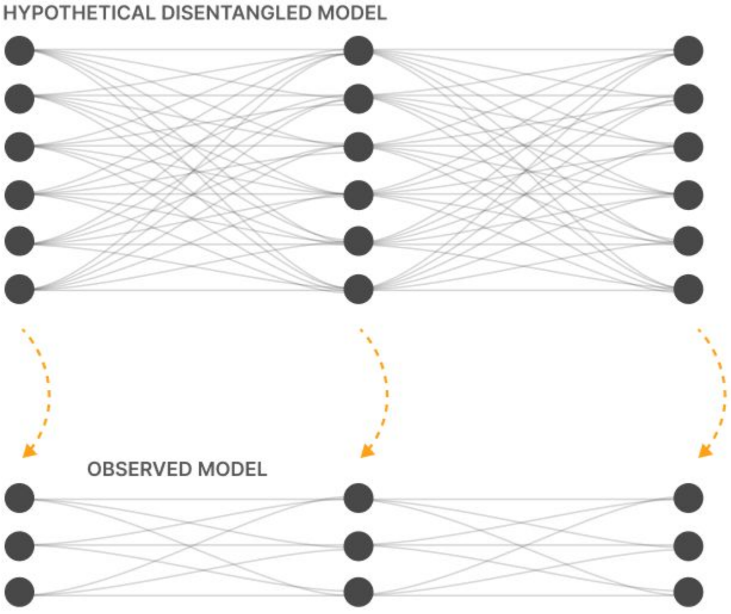


Even if only **one sparse feature** is active, using linear dot product projection on the superposition leads to **interference** which the model must tolerate or filter.



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**.

Simulating noisy larger model



Under the superposition hypothesis, the neural networks we observe are **simulations of larger networks** where every neuron is a disentangled feature.

These idealized neurons are **projected** on to the actual network as “almost orthogonal” vectors over the neurons.

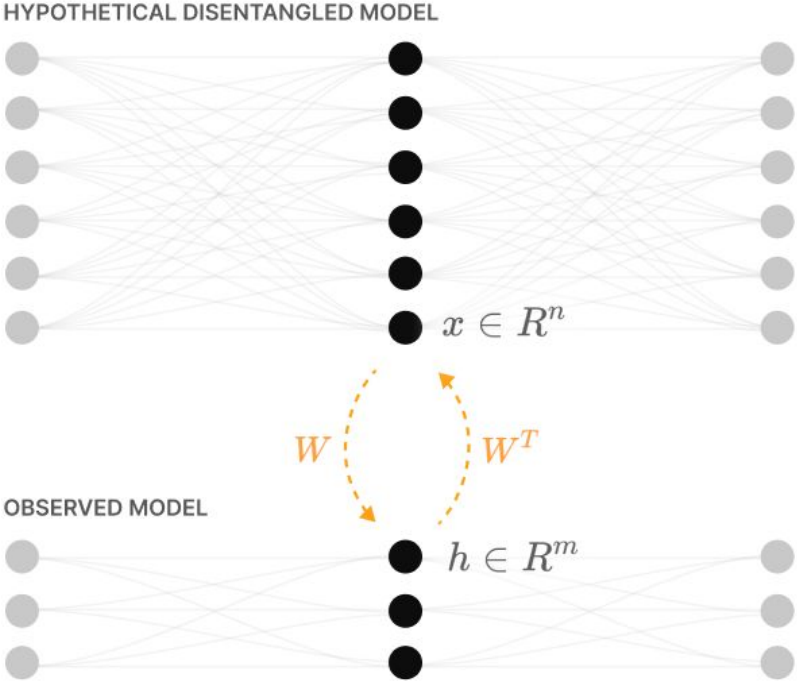
The network we observe is a **low-dimensional projection** of the larger network. From the perspective of individual neurons, this presents as polysemanticity.

Okay so what’s important about features

- Decomposability
- Linearity
- Superposition vs non-superposition
- Basis-aligned

Latter two occur sometimes, first two are widespread

Demonstration

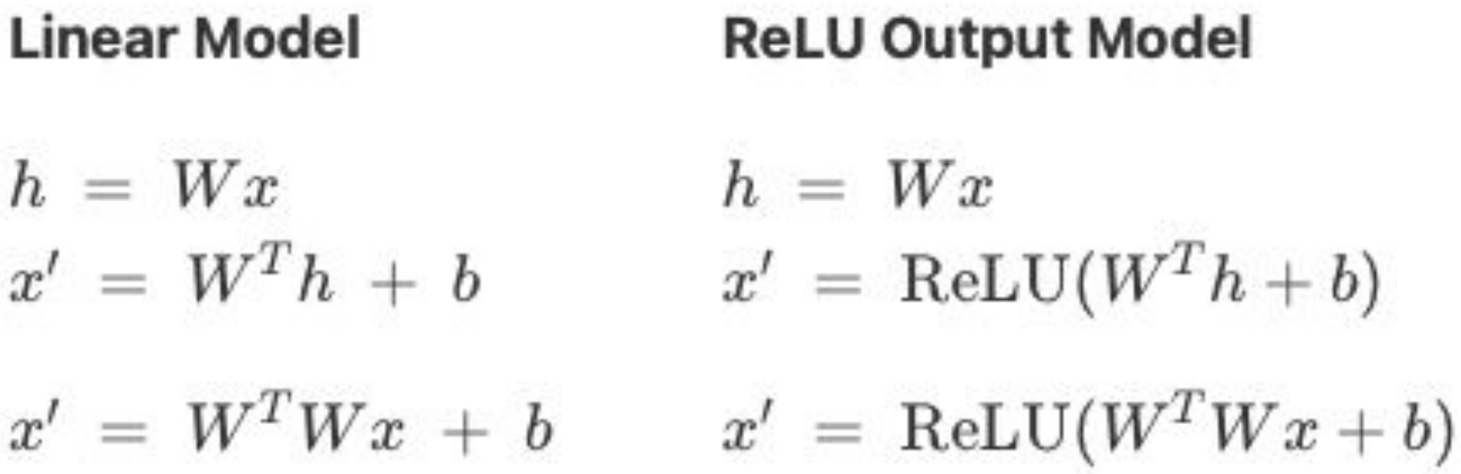


Our first experiments will test the extent to which the idealized activations of an imagined larger model can be **stored** and **recovered** from a lower-dimensional space.

Synthetic data

- Feature sparsity in the real world
- More (potential) features than neurons
- Features vary in importance

Model



Linear Model

$$h = Wx$$

$$x' = W^T h + b$$

$$x' = W^T W x + b$$

ReLU Output Model

$$h = Wx$$

$$x' = \mathrm{ReLU}(W^T h + b)$$

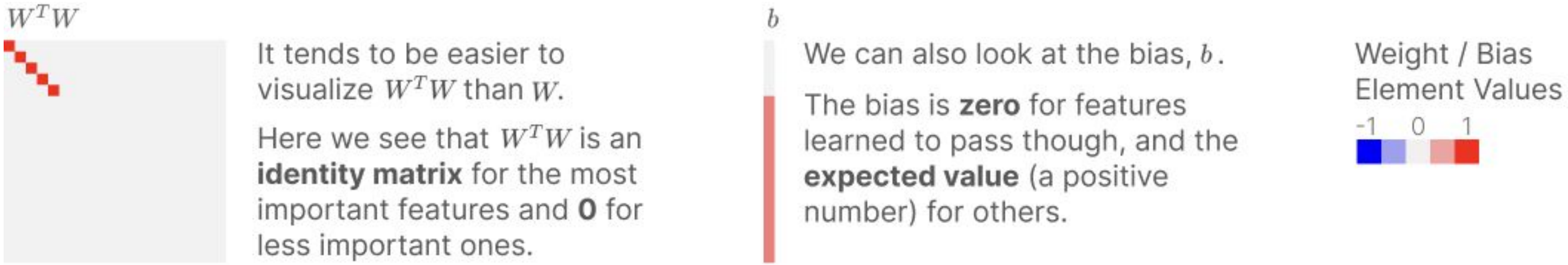
$$x' = \mathrm{ReLU}(W^T W x + b)$$

Model

- Autoencoder like, W \in R^{m, n}

$$L = \sum_{x} \sum_{i} I_i (x_i - x_i')^2$$

Result visualization



It tends to be easier to visualize W^TW than W.

Here we see that W^TW is an **identity matrix** for the most important features and **0** for less important ones.

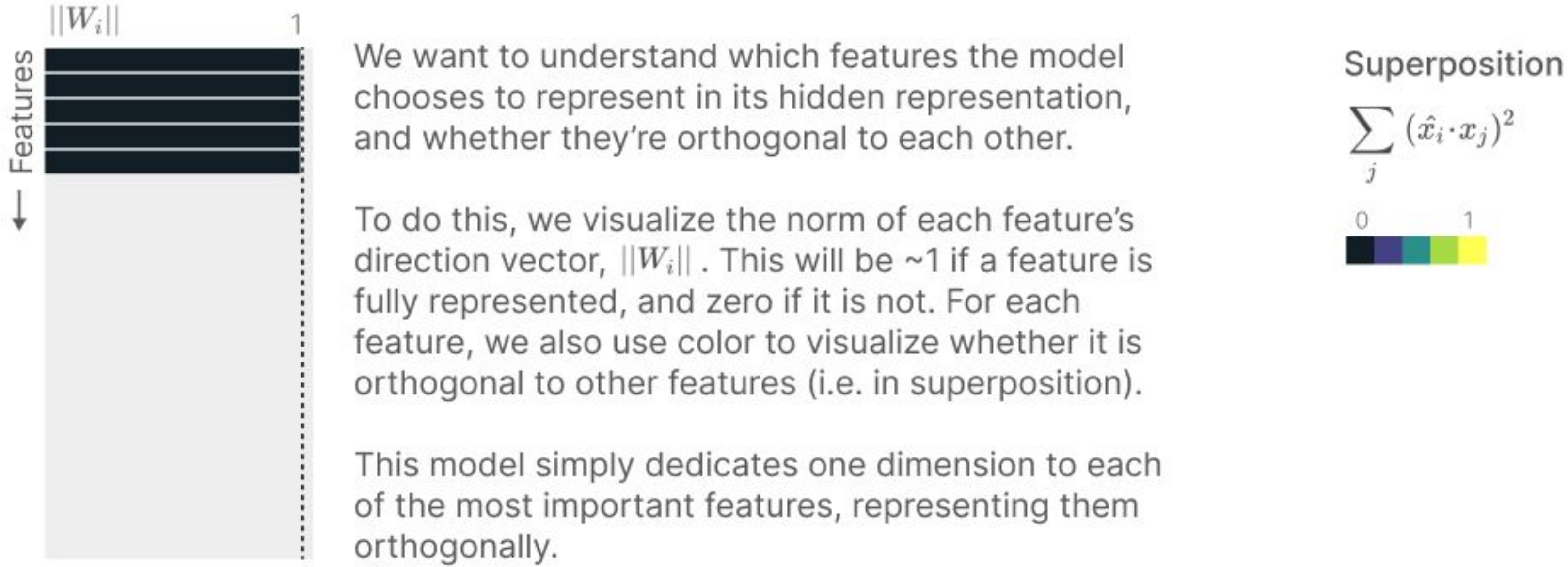
We can also look at the bias, b.

The bias is **zero** for features learned to pass though, and the **expected value** (a positive number) for others.

Result visualization

- If feature is represented in superposition is determined by ||Wi||, the norm of its embedding vector
- Calculate projection of all other features Wj onto Wi -> 0 if orthogonal, if >=1 there is some group of other features which can activate Wi as strongly as feature i itself

Result visualization

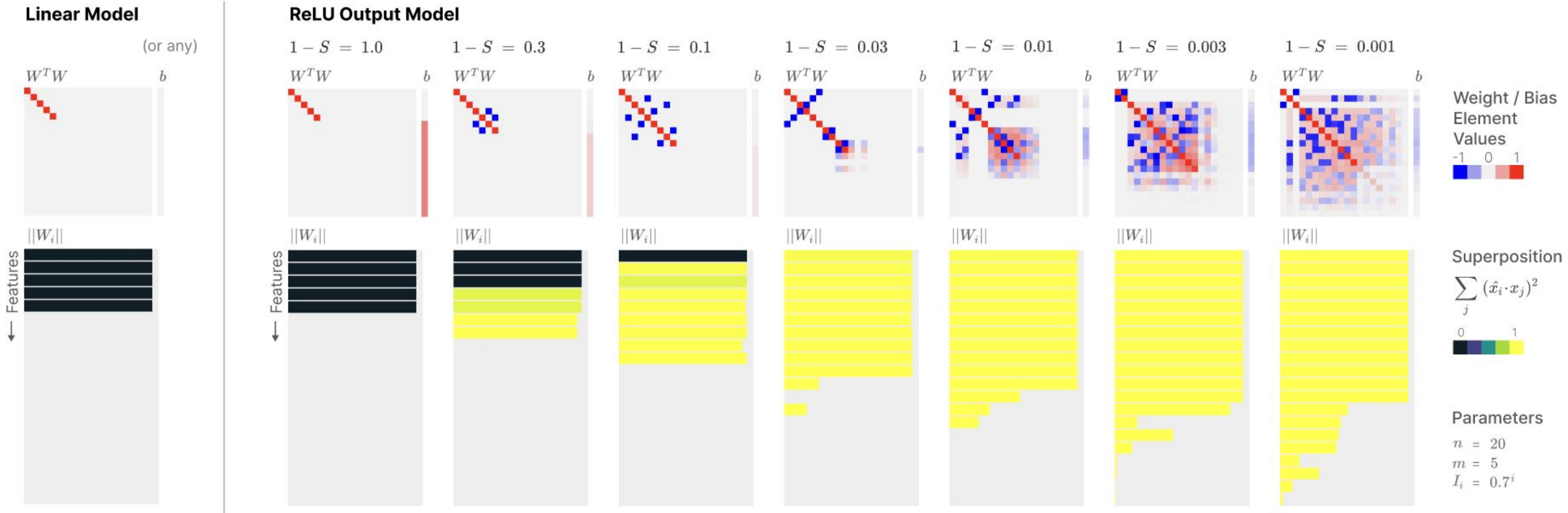


We want to understand which features the model chooses to represent in its hidden representation, and whether they're orthogonal to each other.

To do this, we visualize the norm of each feature's direction vector, $||W_i||$. This will be ~1 if a feature is fully represented, and zero if it is not. For each feature, we also use color to visualize whether it is orthogonal to other features (i.e. in superposition).

This model simply dedicates one dimension to each of the most important features, representing them orthogonally.

Results



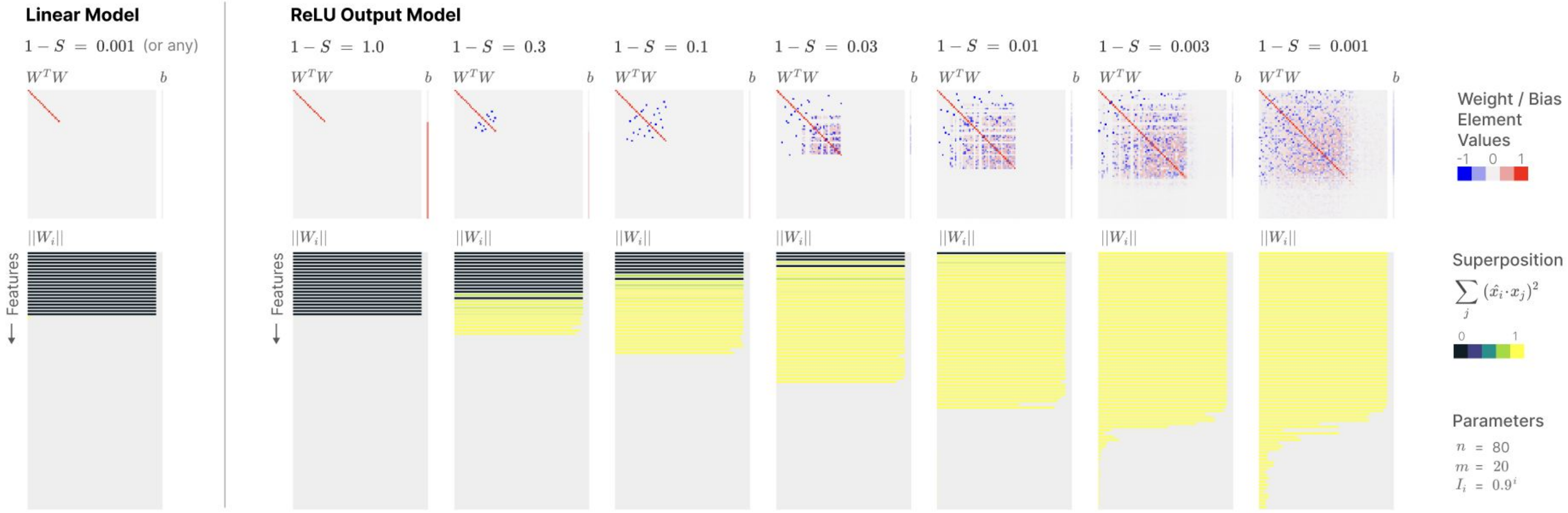
Linear models learn the top m features. $1 - S$ = 0.001 is shown, but others are similar.

In the **dense** regime, ReLU output models also learn the top m features.

As **sparsity increases**, superposition allows models to represent more features. The most important features are initially untouched. This early superposition is organized in antipodal pairs (more on this later).

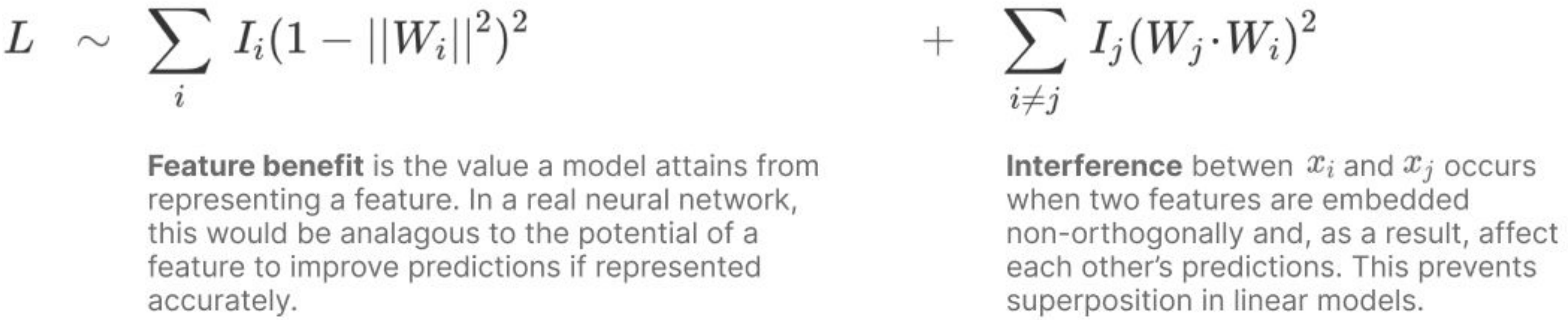
In the **high sparsity** regime, models put all features in superposition, and continue packing more. Note that at this point we begin to see positive interference and negative biases. We'll talk about this more later.

Results: more features/hidden dimensions



Why? Let's ask Saxe

- Neural networks can be thought of as optimizing a simple closed-form solution, tweaked formulas:



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

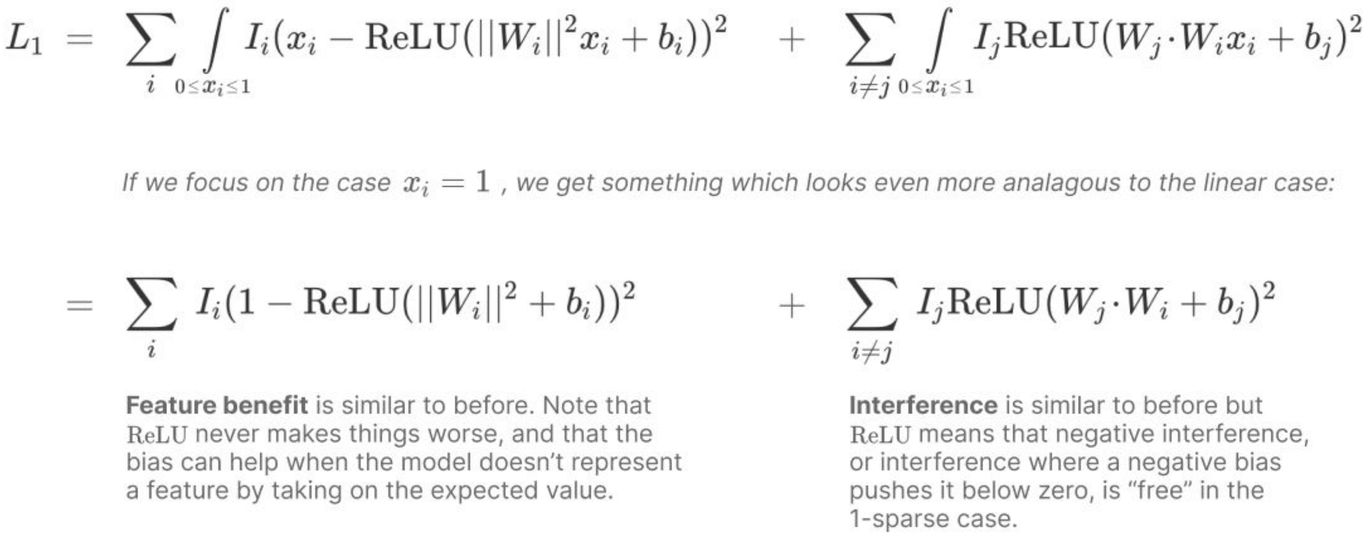
Feature benefit is the value a model attains from representing a feature. In a real neural network, this would be analagous to the potential of a feature to improve predictions if represented accurately.

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.

Why? Let’s ask Saxe

- Two main forces:
    - Feature benefit (more features to get a better loss)
    - Interference when it tries to represent too many features due to interference

A little math



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

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

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

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.

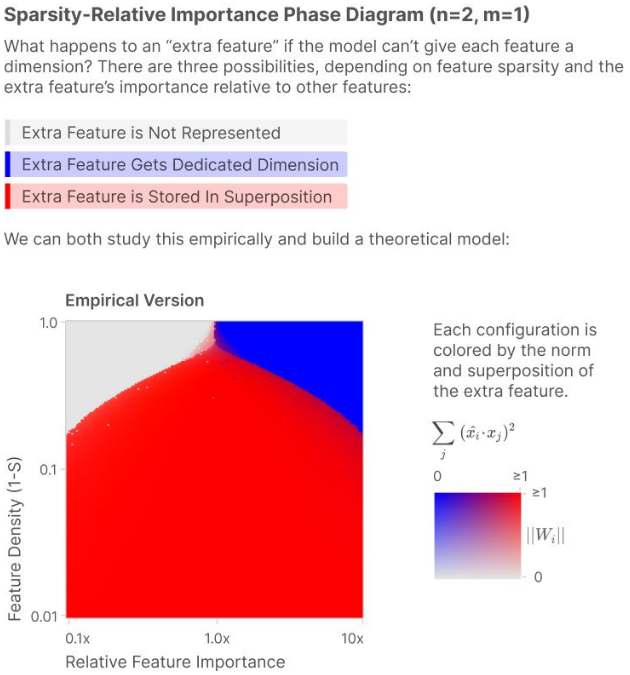
Interference is similar to before but ReLU means that negative interference, or interference where a negative bias pushes it below zero, is "free" in the 1-sparse case.

Summary

1) Features may simply not be learned
2) Feature may be learned and represented in superposition
3) The model may represent a feature with a dedicated dimension

There is some sort of transition between these three outcomes

Superposition as a phase change



Sparsity-Relative Importance Phase Diagram (n=2, m=1)

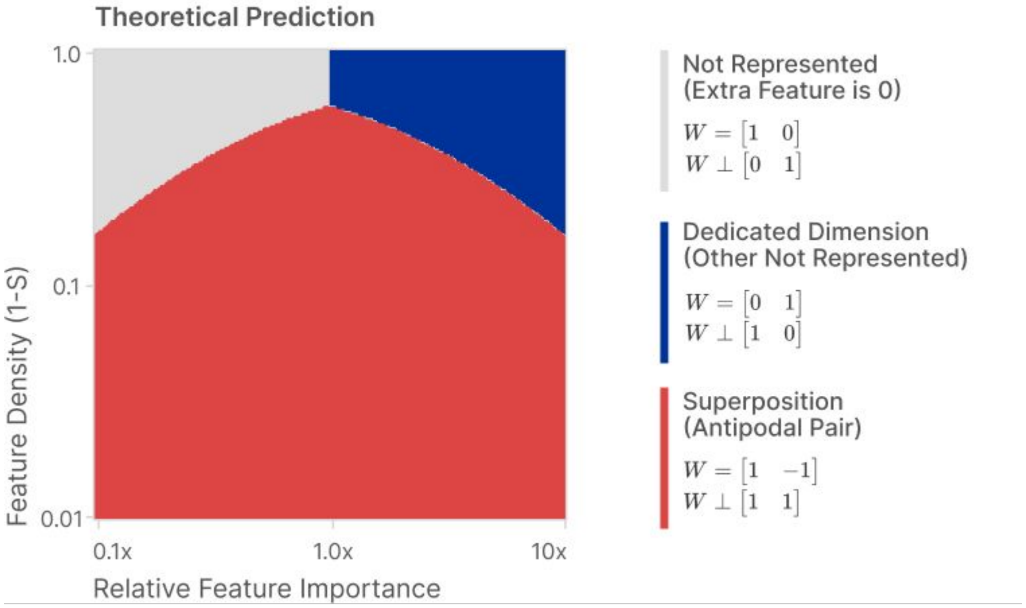
What happens to an "extra feature" if the model can't give each feature a dimension? There are three possibilities, depending on feature sparsity and the extra feature's importance relative to other features:

- ⬜ Extra Feature is Not Represented
- 🟦 Extra Feature Gets Dedicated Dimension
- 🟥 Extra Feature is Stored In Superposition

We can both study this empirically and build a theoretical model:

Empirical Version

Each configuration is colored by the norm and superposition of the extra feature.



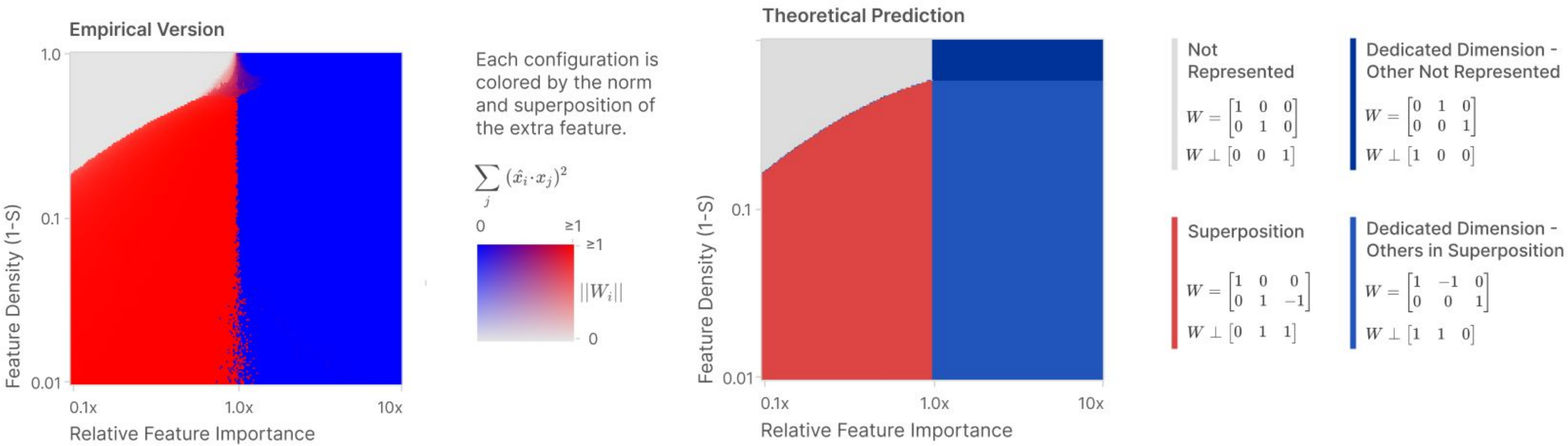
Theoretical Prediction

Conclusion

- Sparsity is necessary for superposition to occur
- There is a first-order phase change in the theoretical model, a crossover between functions causing a discontinuity in the derivative of the optimal loss

Phase change 3 features

Sparsity-Relative Importance Phase Diagram (n=3, m=2)



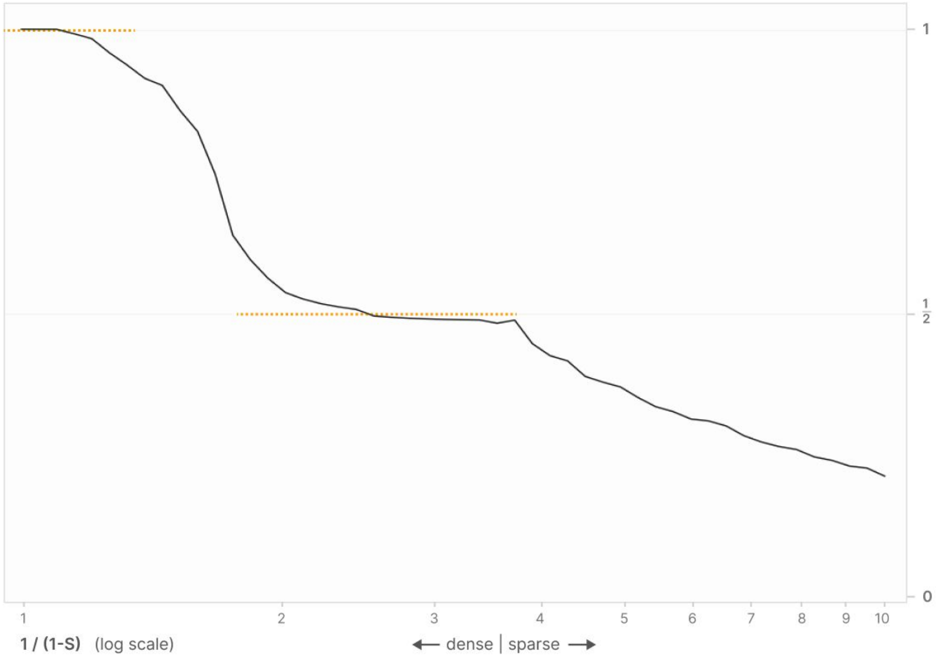
Empirical Version

Theoretical Prediction

Each configuration is colored by the norm and superposition of the extra feature.

We need to go deeper

Uniform superposition geometry



Number of Hidden Dimensions per Embedded Feature

$m/||W||_F^2$

Unexpectedly, we find that the number of dimensions per feature is “sticky” at 1 and 1/2. We denote this with ⋯ .

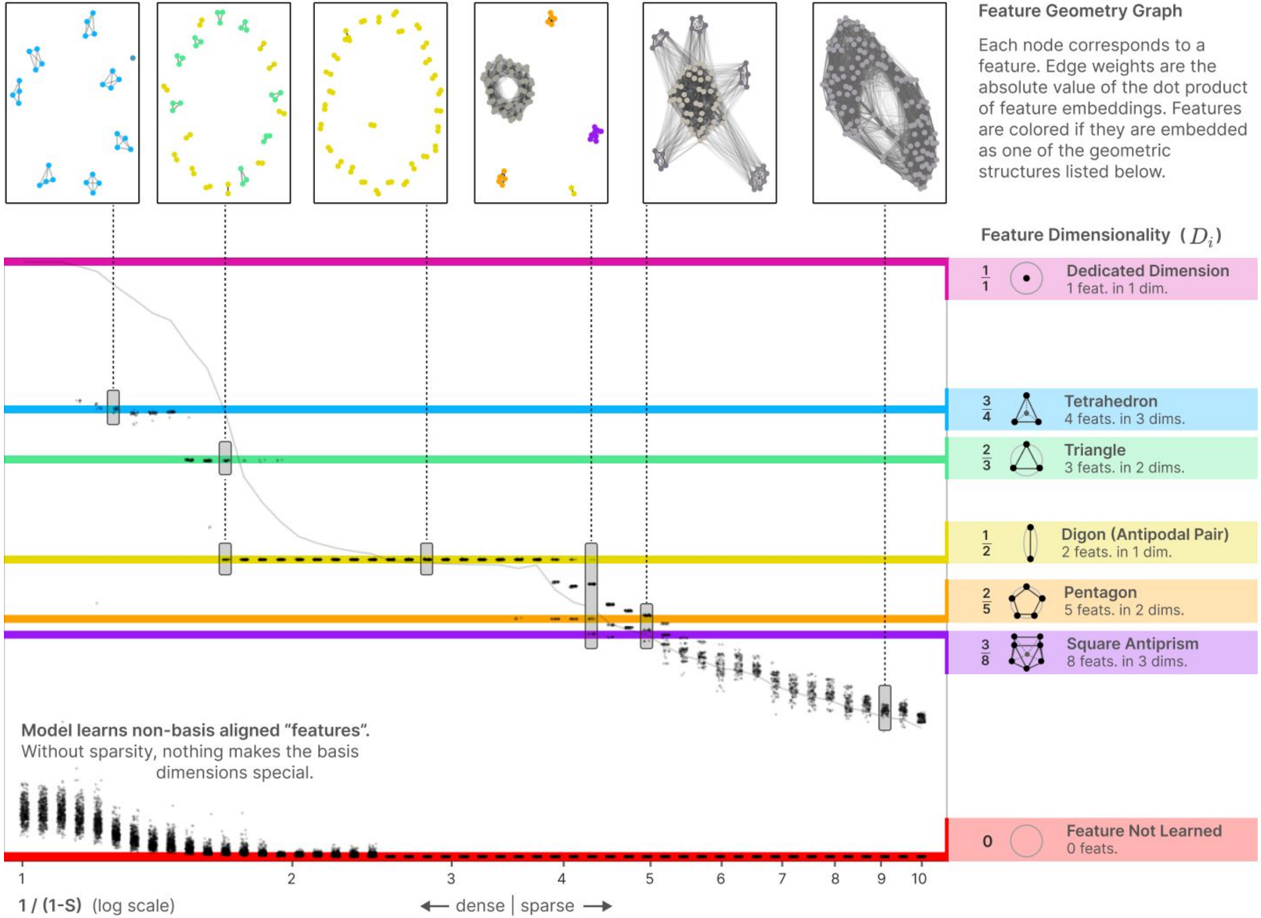
Conclusion

- There are two ‘sticky regions’ when the feature gets a full dimension and when it gets half a dimension on average -> measure how much dimension each feature gets

Feature dimensionality

$$D_i = \frac{||W_i||^2}{\sum_j (\hat{W}_i \cdot W_j)^2}$$

Feature dimensionality



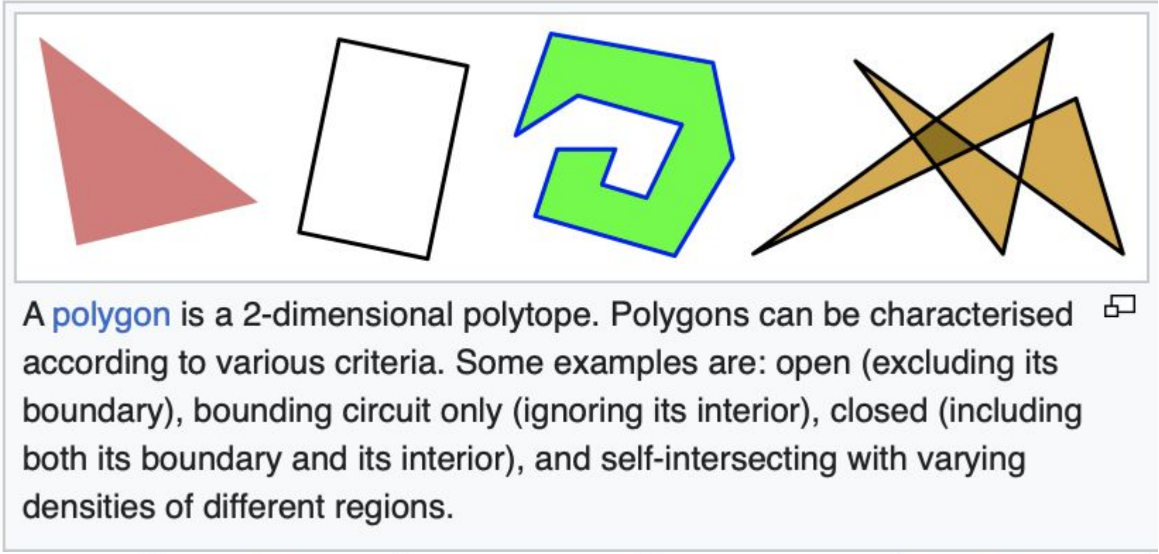
Feature Geometry Graph

Each node corresponds to a feature. Edge weights are the absolute value of the dot product of feature embeddings. Features are colored if they are embedded as one of the geometric structures listed below.

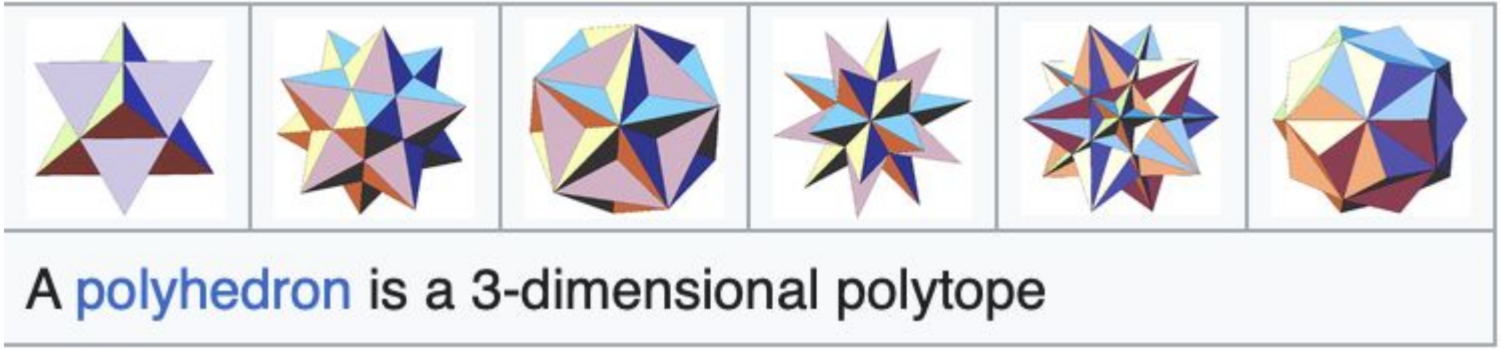
Conclusion

- Model likes to learn specific weight geometries and jumps between the different configurations. Moving from Digon to pentagon is ‘sticky’ region!
- They compare to this problem earlier, but a lot of these shapes are solutions to the Thompson problem.

Polytope?

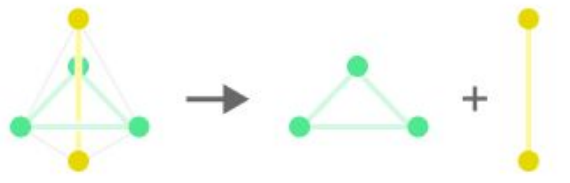


A polygon is a 2-dimensional polytope. Polygons can be characterised according to various criteria. Some examples are: open (excluding its boundary), bounding circuit only (ignoring its interior), closed (including both its boundary and its interior), and self-intersecting with varying densities of different regions.

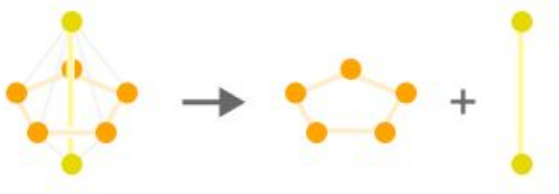


A polyhedron is a 3-dimensional polytope

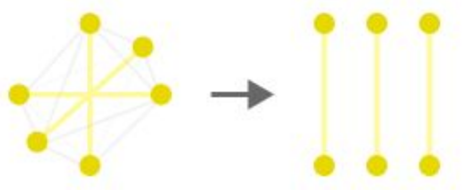
Tegum product (embedding two polytopes in orthogonal subspaces)



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.

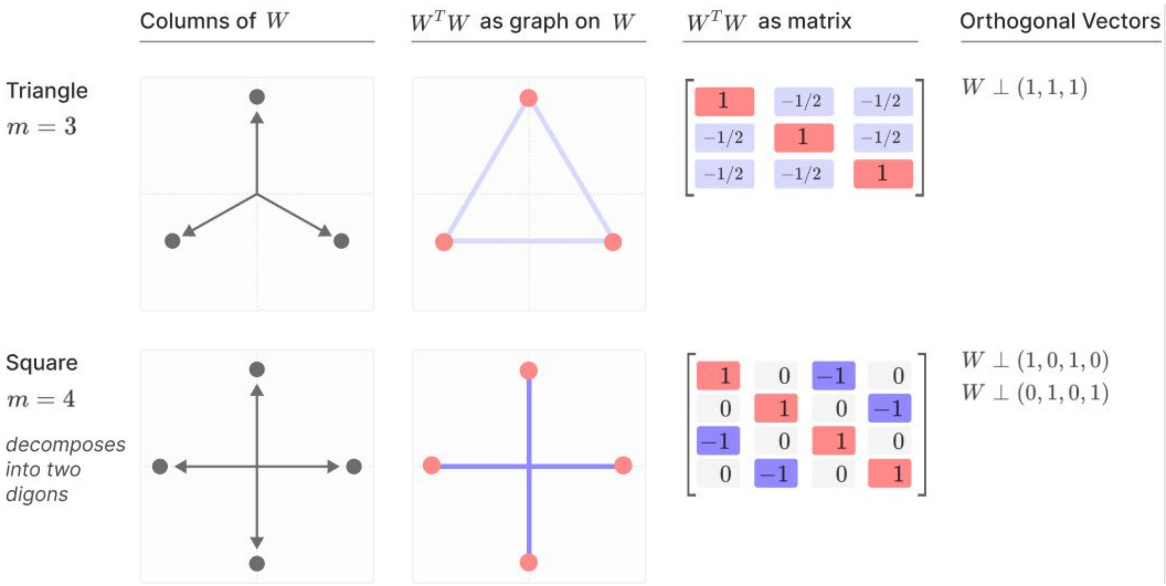


A pentagonal bipyramid is the tegum product of a pentagon and an antipode. As a result, we observe 5×2/5 features and 2×1/2 features, rather than 7×3/7 features.



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

Polytopes and low-rank matrices



	Columns of W	W^TW as graph on W	W^TW as matrix	Orthogonal Vectors
Triangle $m = 3$			$\begin{bmatrix} 1 & -1/2 & -1/2 \\ -1/2 & 1 & -1/2 \\ -1/2 & -1/2 & 1 \end{bmatrix}$	$W \perp (1,1,1)$
Square $m = 4$ *decomposes into two digons*			$\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$	$W \perp (1,0,1,0)$ $W \perp (0,1,0,1)$

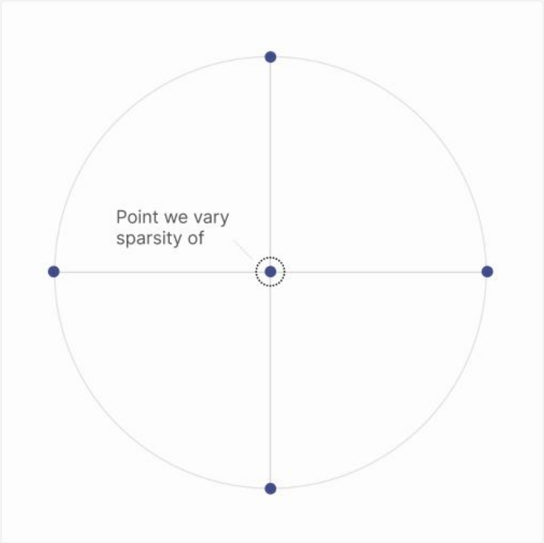
- Three features in 2-superposition corresponds to a triangle
- Three equally important features -> equilateral triangle

Non-uniform superposition

- Features varying in importance or sparsity
    - Polytopes smoothly deform until they snap into a new polytope
- Correlated features
    - May form an orthogonal locally correlated basis, or are side-by-side, or snap into one single feature -> PCA?
- Anti-correlated features
    - Prefer being in the same tegum factors, and antipodal

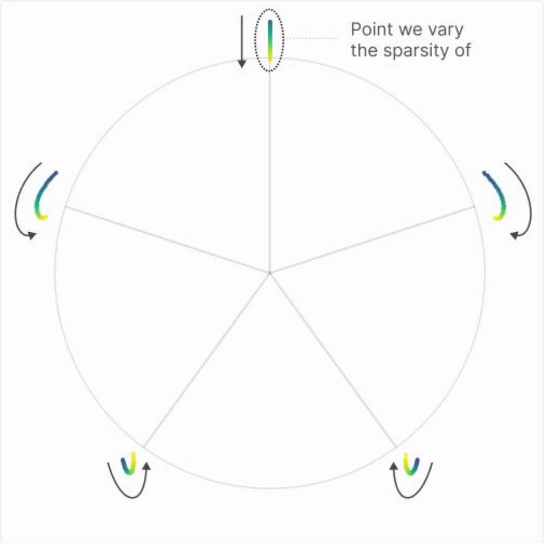
Experiment

Digon (Square) Solutions



When the sparsity of the varied point falls below a certain critical threshold (~2.5x less than others) the pentagon solution changes to two digons.

Pentagon Solutions



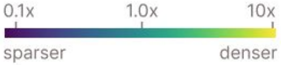
Note how vertices shift as sparsity changes

To study non-uniform sparsity, we consider models with five features, varying the sparsity of a single feature and observing how the resulting solutions change. We observe a mixture of continuous deformation and sharp phase changes.

Parameters

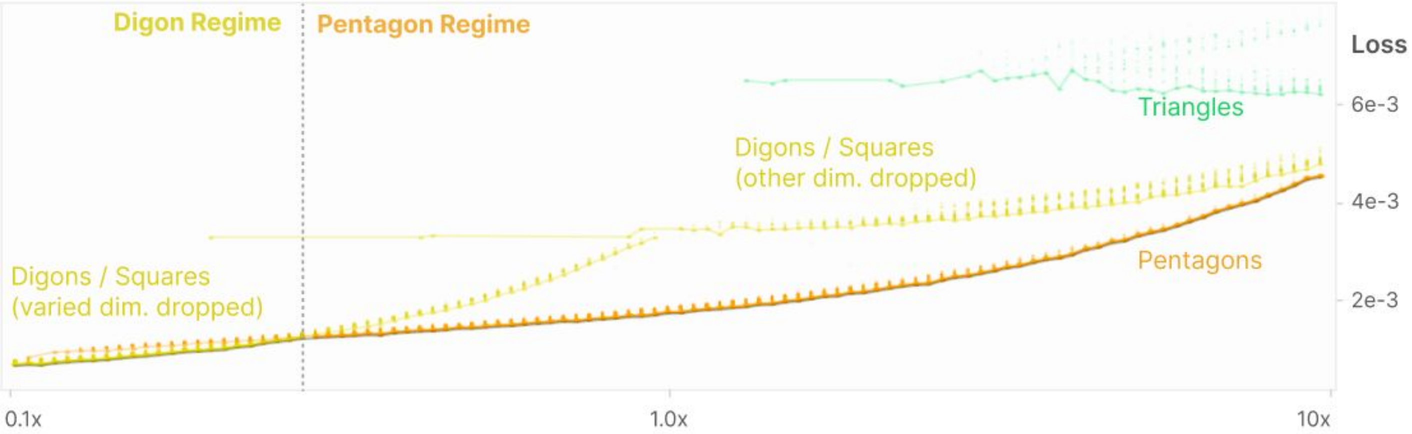
n = 5
m = 2
I_i = 1
$1\text{-}S$ = 0.05 (baseline)

Relative Feature Density (1-S)



Results

The Pentagon-Digon Phase Change Corresponds to a Loss Curve Crossover



Gradient descent has trouble moving between solutions associated with different geometries. As a result, fitting the model will often produce non-optimal solutions. By characterizing and plotting these, we can see that each geometry creates a different loss curve, and that the pentagon-digon phase change corresponds to a cross over between the curves.

Conclusion

- Jumping between uniform superpositions
- Gradient descent has issues with these jumps

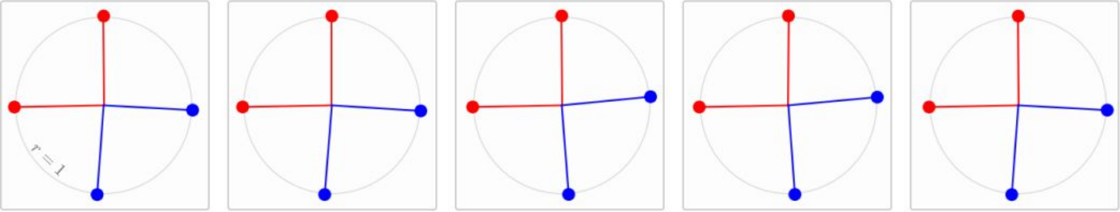
Correlated anti-correlated features

- Correlated feature sets: bundles of co-occurring features (whether all features in a bundle are one or zero with probability S).
- Anticorrelated feature sets: only one feature can be active at a time

Results

Models prefer to represent correlated features in orthogonal dimensions.

We train several models with 2 sets of 2 correlated features (n=4 total) and a m=2 hidden dimensions. We then visualize the weight column for each feature. For ease of comparison, we rotate and flip solutions to have a consistent orientation.

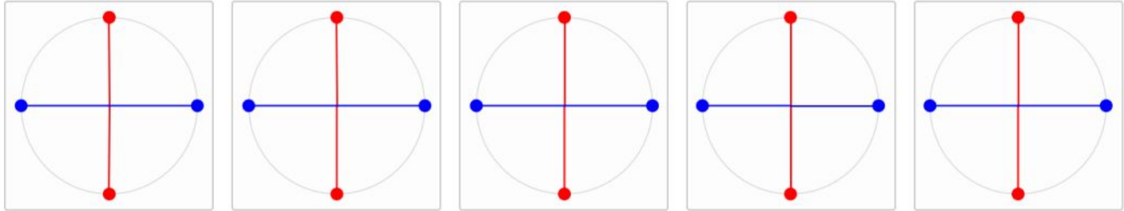


●● and ●● denote **correlated** feature sets.

Correlated feature sets are constructed by having them always co-occur (ie. be zero or not) at the same time.

Models prefer to represent anticorrelated features in opposite directions.

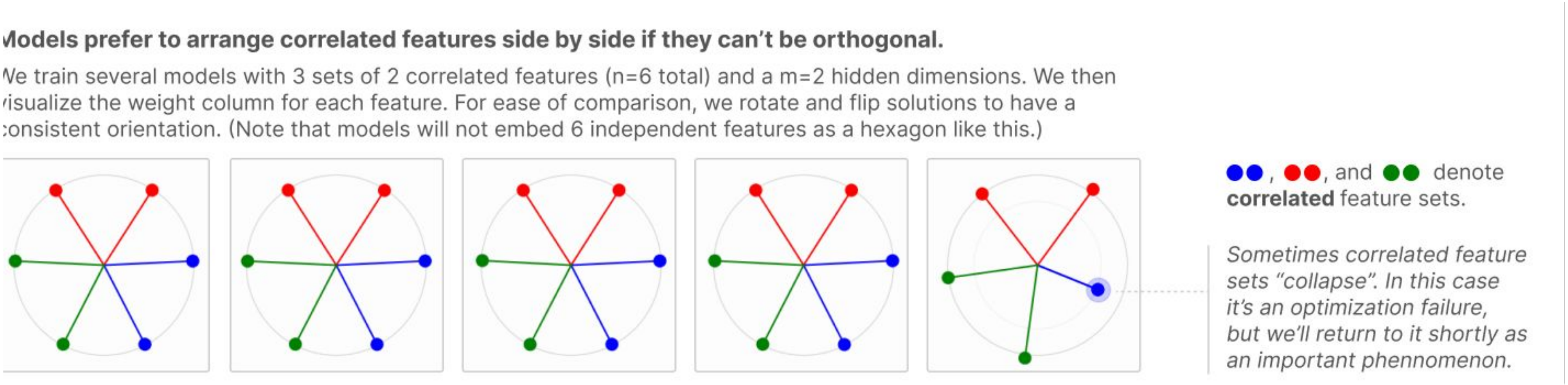
We train several models with 2 sets of 2 anticorrelated features (n=4 total) and a m=2 hidden dimensions. We then visualize the weight column for each feature. For ease of comparison, we rotate and flip solutions to have a consistent orientation.



●● and ●● denote **anticorrelated** feature sets.

Anticorrelated feature sets are constructed by having them never co-occur (ie. be zero or not) at the same time.

Results



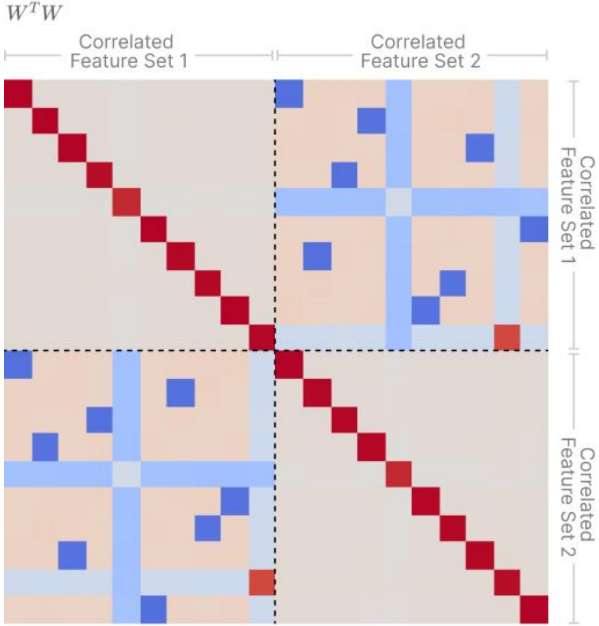
Models prefer to arrange correlated features side by side if they can't be orthogonal.

We train several models with 3 sets of 2 correlated features (n=6 total) and a m=2 hidden dimensions. We then visualize the weight column for each feature. For ease of comparison, we rotate and flip solutions to have a consistent orientation. (Note that models will not embed 6 independent features as a hexagon like this.)

●●, ●●, and ●● denote **correlated** feature sets.

Sometimes correlated feature sets "collapse". In this case it's an optimization failure, but we'll return to it shortly as an important phennomenon.

Locally orthogonal bases



Models prefer to represent correlated features in orthogonal dimensions, creating "local orthogonal bases".

We train a model with 2 sets of 10 correlated features (n=20 total) with m=10 hidden dimensions.

Within each set of correlated featuers, the model creates a *local orthogonal basis*, having each feature be represented orthogonally.

Weight Element Values

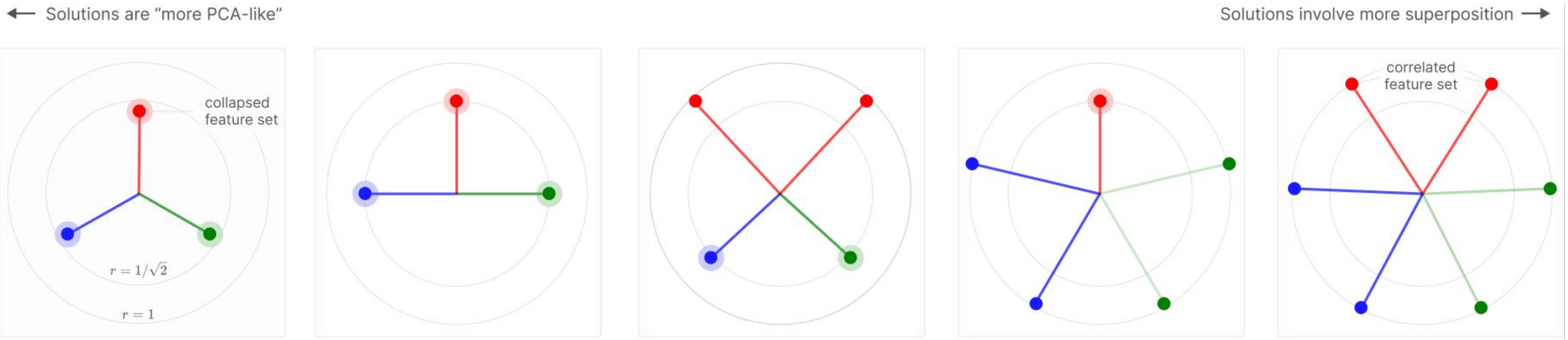
-1 0 1

🟦🟦⬜🟥🟥

Relation to PCA

- If there are two correlated features a and b, but the model only has capacity to represent one, the model will represent their principal component: (a + b) / sqrt(2), which is a relatively sparse variable and more impact on the loss than either individually
-

Results


← Solutions are "more PCA-like"
Solutions involve more superposition →
collapsed feature set
$r = 1/\sqrt{2}$
$r = 1$
correlated feature set

Most PCA-like Solution
Approximately 0.5 ≤ 1-S

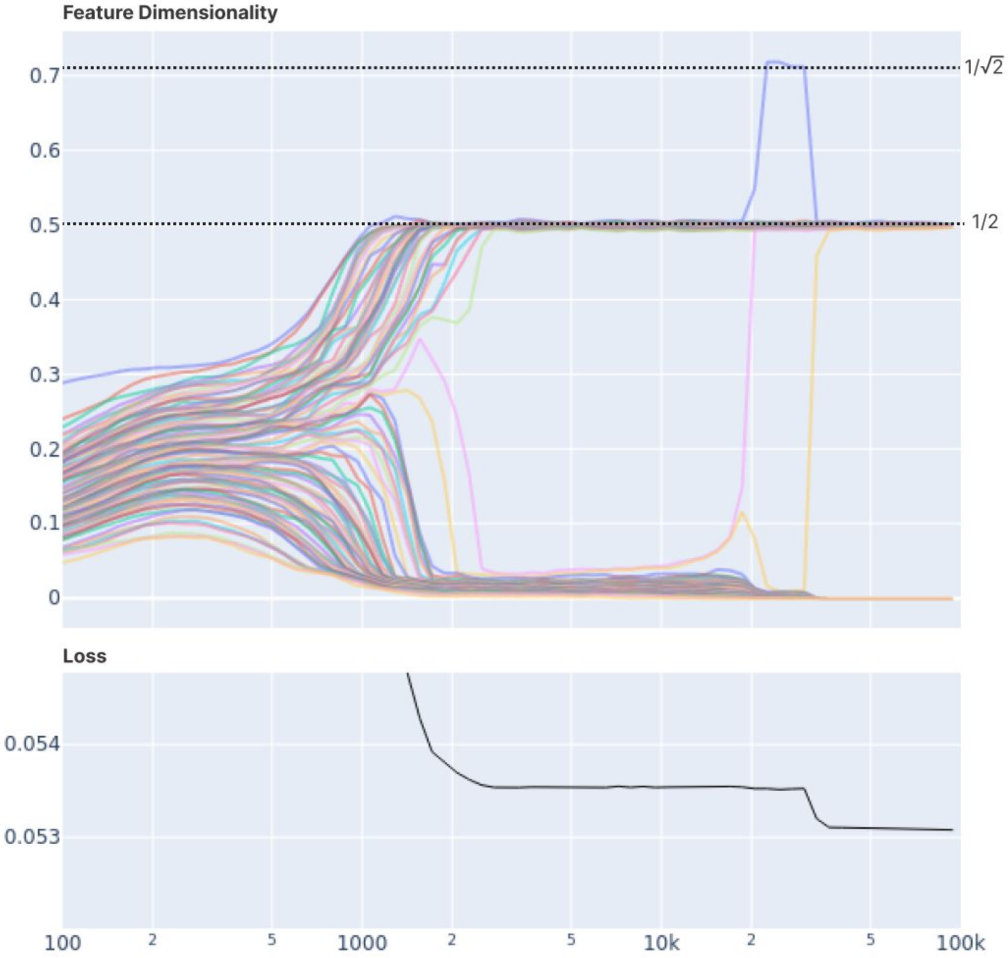
All Sets of Features Collapsed
Approximately 0.25 ≤ 1-S ≤ 0.5

Two Sets of Features Collapsed
Approximately 0.15 ≤ 1-S ≤ 0.2

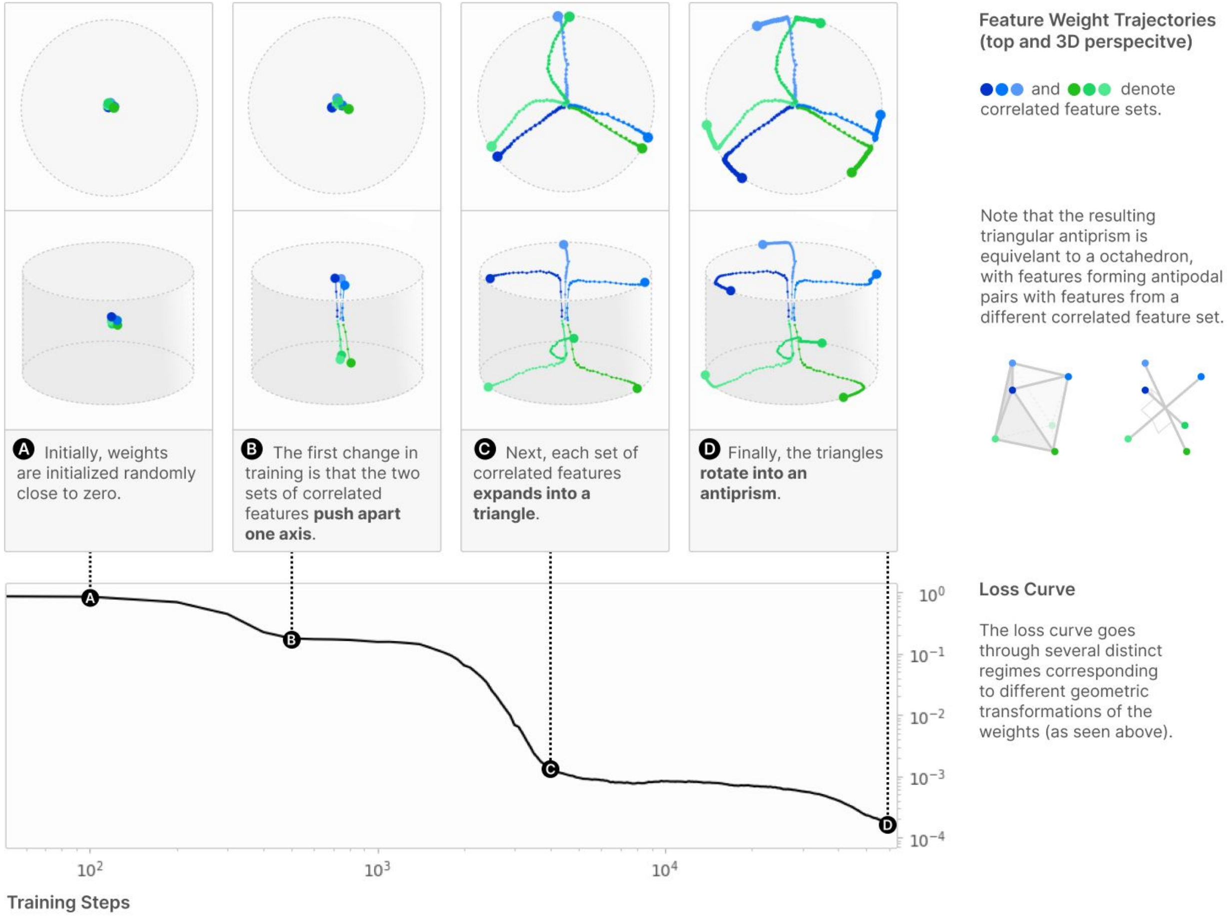
One Set of Features Collapsed
Approximately 0.05 ≤ 1-S ≤ 0.15

No Features Collapsed
Approximately 1-S ≤ 0.05

Teaser: dynamics



Teaser: dynamics



Feature Weight Trajectories (top and 3D perspecitve)

●●● and ●●● denote correlated feature sets.

Note that the resulting triangular antiprism is equivelant to a octahedron, with features forming antipodal pairs with features from a different correlated feature set.

A Initially, weights are initialized randomly close to zero.

B The first change in training is that the two sets of correlated features **push apart one axis**.

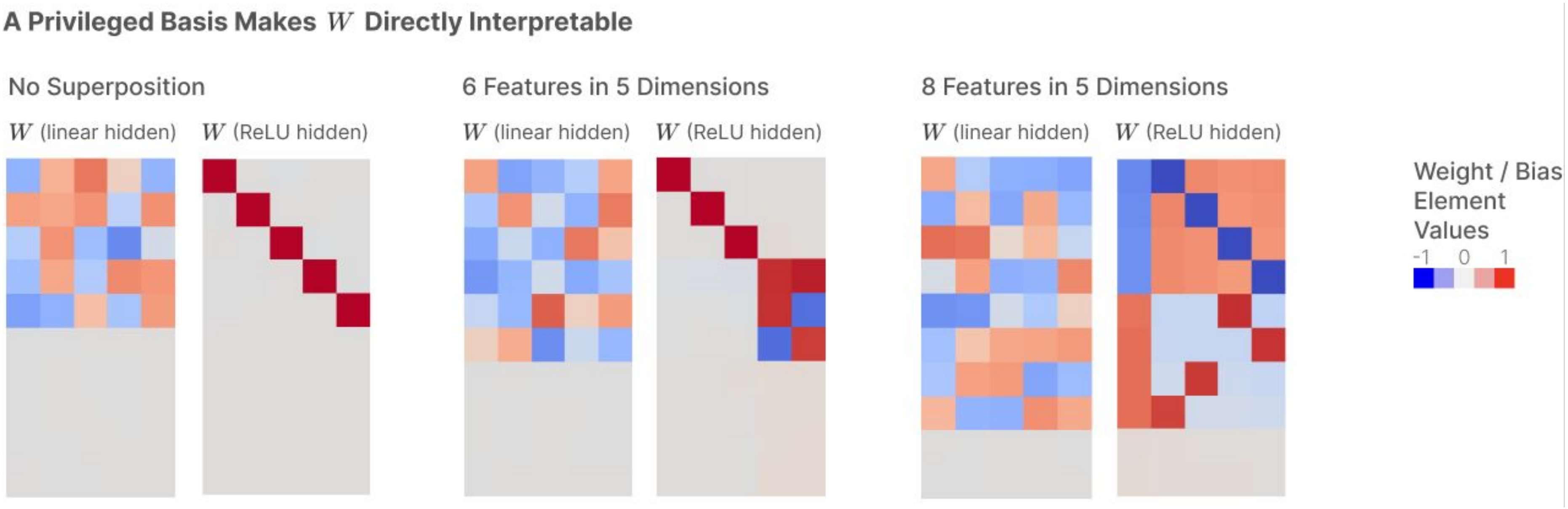
C Next, each set of correlated features **expands into a triangle**.

D Finally, the triangles **rotate into an antiprism**.

Loss Curve

The loss curve goes through several distinct regimes corresponding to different geometric transformations of the weights (as seen above).

Teaser: interpretability



Final conclusion

- Read the whole blog post, it is very accessible and detailed