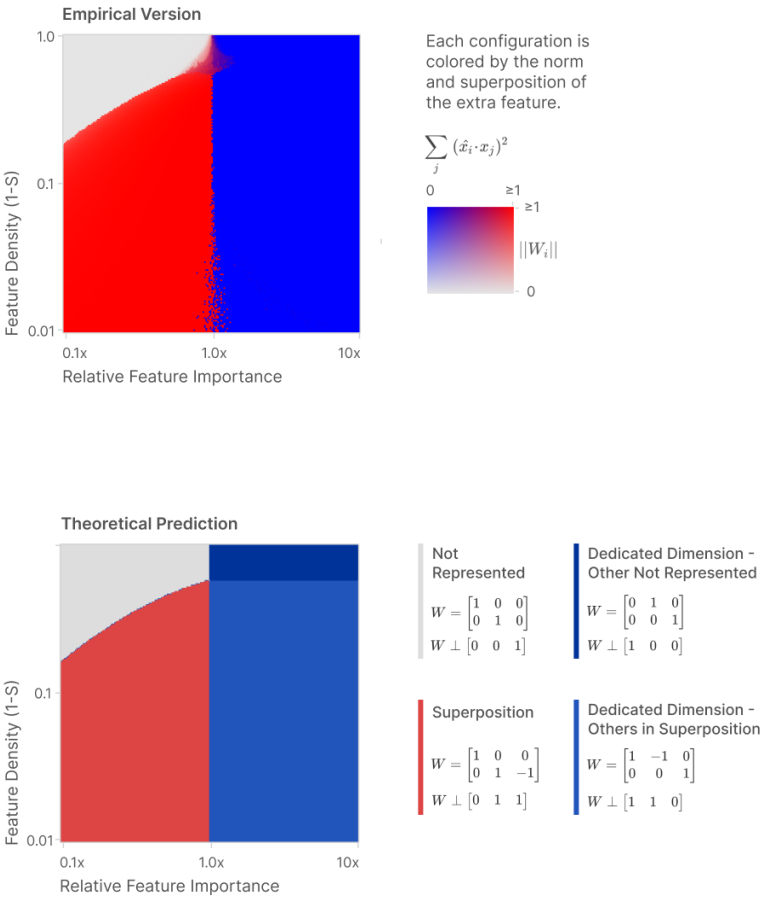
Sparsity-Relative Importance Phase Diagram (n=3, m=2)



These diagrams suggest that there really is a phase change between different strategies for encoding features. However, we'll see in the next section that there's much more complex structure this preliminary view doesn't capture.

The Geometry of Superposition

We've seen that superposition can allow a model to represent extra features, and that the number of extra features increases as we increase sparsity. In this section, we'll investigate this relationship in more detail, discovering an unexpected geometric story: features seem to organize themselves into geometric structures such as pentagons and tetrahedrons! In some ways, the structure described in this section seems "too elegant to be true" and we think there's a good chance it's at least partly idiosyncratic to the toy model we're investigating. But it seems worth investigating because if anything about this generalizes to real models, it may give us a lot of leverage in understanding their representations.

We'll start by investigating ***uniform superposition***, where all features are identical: independent, equally important and equally sparse. It turns out that uniform superposition has a surprising connection to the geometry of uniform polytopes! Later, we'll move on to investigate ***non-uniform superposition***, where features are not identical. It turns out that this can be understood, at least to some extent, as a deformation of uniform superposition.

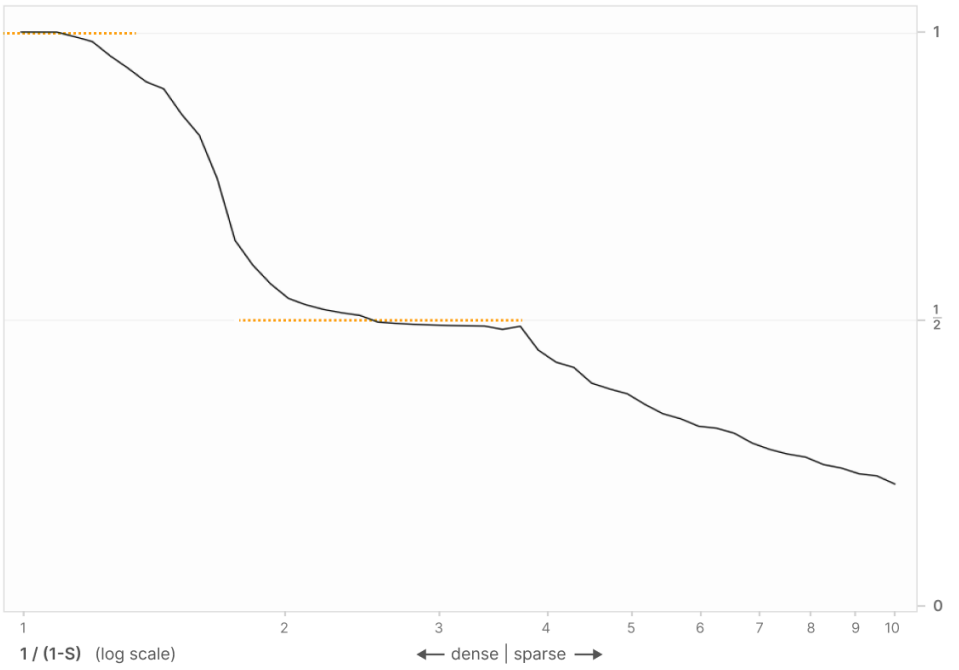
Uniform Superposition

As mentioned above, we begin our investigation with uniform superposition, where all features have the same importance and sparsity. We'll see later that this case has some unexpected structure, but there's also a much more basic reason to study it: it's much easier to reason about than the non-uniform case, and has fewer variables we need to worry about in our experiments.

We'd like to understand what happens as we change feature sparsity, S. Since all features are equally important, we will assume without loss of generality[14] that each feature has importance $I_i = 1$. We'll study a model with $n = 400$ features and $m = 30$ hidden dimensions, but it turns out the number of features and hidden dimensions doesn't matter very much. In particular, it turns out that the number of input features n doesn't matter as long as it's much larger than the number of hidden dimensions, $n \gg m$. And it also turns out that the number of hidden dimensions doesn't really matter as long as we're interested in the ratio of features learned to hidden features. Doubling the number of hidden dimensions just doubles the number of features the model learns.

A convenient way to measure the number of features the model has learned is to look at the Frobenius norm, $||W||_F^2$. Since $||W_i||^2 \simeq 1$ if a feature is represented and $||W_i||^2 \simeq 0$ if it is not, this is roughly the number of features the model has learned to represent. Conveniently, this norm is basis-independent, so it still behaves nicely in the dense regime $S = 0$ where the feature basis isn't privileged by anything and the model represents features with arbitrary directions instead.

We'll plot $D^* = m/||W||_F^2$, which we can think of as the "dimensions per feature":



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 ····· .

Surprisingly, we find that this graph is "sticky" at 1 and $1/2$. (This very vaguely resembles the fractional quantum Hall effect – see e.g. this diagram.) Why is this? On inspection, the $1/2$ "sticky point" seems to correspond to a precise geometric arrangement where features come in "antipodal pairs", each being exactly the negative of the other, allowing two features to be packed into each hidden dimension. It appears that antipodal pairs are so effective that the model preferentially uses them over a wide range of the sparsity regime.

It turns out that antipodal pairs are just the tip of the iceberg. Hiding underneath this curve are a number of extremely specific geometric configurations of features.

FEATURE DIMENSIONALITY

In the previous section, we saw that there's a sticky regime where the model has "half a dimension per feature" in some sense. This is an average statistical property of the features the model represents, but it seems to hint at something interesting. Is there a way we could understand what "fraction of a dimension" a specific feature gets?

We'll define the *dimensionality* of the ith feature, D_i, as:

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

where W_i is the weight vector column associated with the ith feature, and $\hat{W}_i$ is the unit version of that vector.

Intuitively, the numerator represents the extent to which a given feature is represented, while the denominator is "how many features share the dimension it is embedded in" by projecting each feature onto its dimension. In the antipodal case, each feature participating in an antipodal pair will have a dimensionality of $D = 1/(1+1) = 1/2$ while features which are not learned will have a dimensionality of 0. Empirically, it seems that the dimensionality of all features add up to the number of embedding dimensions when the features are "packed efficiently" in some sense.

We can now break the above plot down on a per-feature basis. This reveals many more of these "sticky points"! To help us understand this better, we're going to create a scatter plot annotated with some additional information:

- We start with the line plot we had in the previous section.
- We overlay this with a scatter plot of the individual feature dimensionalities for each feature in the models at each sparsity level.
- The feature dimensionalities cluster at certain fractions, so we draw lines for those. (It turns out that each fraction corresponds to a specific weight geometry – we'll discuss this shortly.)
- We visualize the weight geometries for a few models with a "feature geometry graph" where each feature is a node and edge weights are based on the absolute value of the dot product feature embedding vectors. So features are connected if they aren't orthogonal.

Let's look at the resulting plot, and then we'll try to figure out what it's showing us: