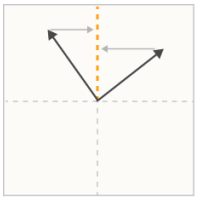
From this perspective, it only makes sense to ask if a *neuron* is interpretable when it is in a privileged basis. In fact, we typically reserve the word "neuron" for basis directions which are in a privileged basis. (See longer discussion here.)

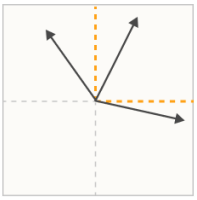
Note that having a privileged basis doesn't guarantee that features will be basis-aligned – we'll see that they often aren't! But it's a minimal condition for the question to even make sense.

The Superposition Hypothesis

Even when there is a privileged basis, it's often the case that neurons are "polysemantic", responding to several unrelated features. One explanation for this is the *superposition hypothesis*. Roughly, the idea of superposition is that neural networks "want to represent more features than they have neurons", so they exploit a property of high-dimensional spaces to simulate a model with many more neurons.



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

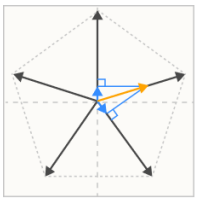


In the **superposition hypothesis**, features can't align with the basis because the model embeds more features than there are neurons. Polysemanticity is inevitable if this happens.

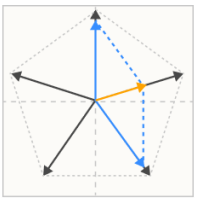
Several results from mathematics suggest that something like this might be plausible:

- **Almost Orthogonal Vectors.** Although it's only possible to have n orthogonal vectors in an n-dimensional space, it's possible to have $\exp(n)$ many "almost orthogonal" ($< \epsilon$ cosine similarity) vectors in high-dimensional spaces. See the Johnson–Lindenstrauss lemma.
- **Compressed sensing.** In general, if one projects a vector into a lower-dimensional space, one can't reconstruct the original vector. However, this changes if one knows that the original vector is sparse. In this case, it is often possible to recover the original vector.

Concretely, in the superposition hypothesis, features are represented as almost-orthogonal directions in the vector space of neuron outputs. Since the features are only almost-orthogonal, one feature activating looks like other features slightly activating. Tolerating this "noise" or "interference" comes at a cost. But for neural networks with highly sparse features, this cost may be outweighed by the benefit of being able to represent more features! (Crucially, sparsity greatly reduces the costs since sparse features are rarely active to interfere with each other, and non-linear activation functions create opportunities to filter out small amounts of noise.)

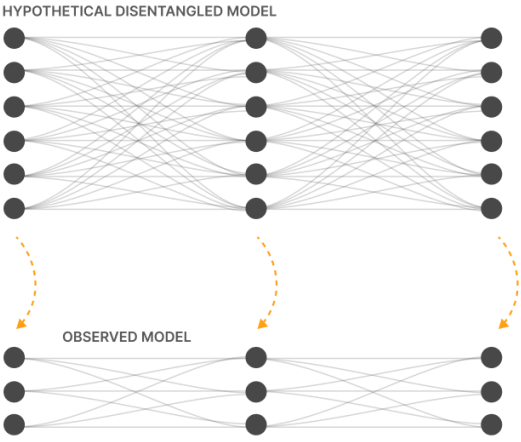


Even if only **one sparse feature** is active, using linear dot product projection on the superposition leads to **interference** which the model must tolerate or filter.



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**.

One way to think of this is that a small neural network may be able to noisily "simulate" a sparse 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.

Although we've described superposition with respect to neurons, it can also occur in representations with an unprivileged basis, such as a word embedding. Superposition simply means that there are more features than dimensions.

Summary: A Hierarchy of Feature Properties

The ideas in this section might be thought of in terms of four progressively more strict properties that neural network representations might have.

- **Decomposability:** Neural network activations which are *decomposable* can be decomposed into features, the meaning of which is not dependent on the value of other features. (This property is ultimately the most important – see the role of decomposition in defeating the curse of dimensionality.)
- **Linearity:** Features correspond to directions. Each feature f_i has a corresponding representation direction W_i. The presence of multiple features $f_1, f_2 \ldots$ activating with values $x_{f_1}, x_{f_2} \ldots$ is represented by $x_{f_1} W_{f_1} + x_{f_2} W_{f_2} \ldots$.
- **Superposition vs Non-Superposition:** A linear representation exhibits superposition if $W^T W$ is not invertible. If $W^T W$ is invertible, it does not exhibit superposition.
- **Basis-Aligned:** A representation is basis aligned if all W_i are one-hot basis vectors. A representation is partially basis aligned if all W_i are sparse. This requires a privileged basis.

The first two (decomposability and linearity) are properties we hypothesize to be widespread, while the latter (non-superposition and basis-aligned) are properties we believe only sometimes occur.

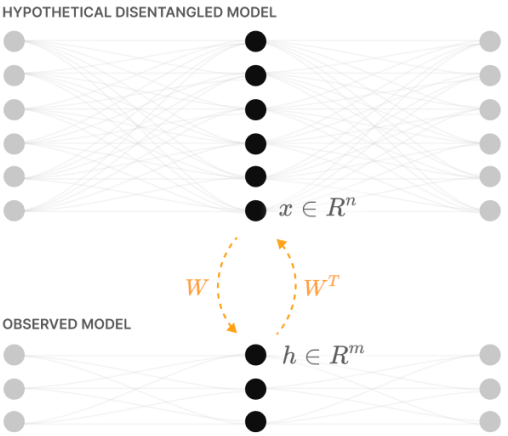
Demonstrating Superposition

If one takes the superposition hypothesis seriously, a natural first question is whether neural networks can actually noisily represent more features than they have neurons. If they can't, the superposition hypothesis may be comfortably dismissed.

The intuition from linear models would be that this isn't possible: the best a linear model can do is to store the principal components. But we'll see that adding just a slight nonlinearity can make models behave in a radically different way! This will be our first demonstration of superposition. (It will also be an object lesson in the complexity of even very simple neural networks.)

Experiment Setup

Our goal is to explore whether a neural network can project a high dimensional vector $x \in R^n$ into a lower dimensional vector $h \in R^m$ and then recover it.[5]



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.

THE FEATURE VECTOR (X)

We begin by describing the high-dimensional vector x: the activations of our idealized, disentangled larger model. We call each element x_i a "feature" because we're imagining features to be perfectly aligned with neurons in the hypothetical larger model. In a vision model, this might be a Gabor filter, a curve detector, or a floppy ear detector. In a language model, it might correspond to a token referring to a specific famous person, or a clause being a particular kind of description.

Since we don't have any ground truth for features, we need to create *synthetic data* for x which simulates any important properties we believe features have from the perspective of modeling them. We make three major assumptions:

- **Feature Sparsity:** In the natural world, many features seem to be sparse in the sense that they only rarely occur. For example, in vision, most positions in an image don't contain a horizontal edge, or a curve, or a dog head [1]. In language, most tokens don't refer to Martin Luther King or aren't part of a clause describing music [2]. This idea goes back to classical work on vision and the statistics of natural images (see e.g. Olshausen, 1997, the section "Why Sparseness?" [24]). For this reason, we will choose a sparse distribution for our features.
- **More Features Than Neurons:** There are an enormous number of potentially useful features a model might represent.[6] This imbalance between features and neurons in real models seems like it must be a central tension in neural network representations.
- **Features Vary in Importance:** Not all features are equally useful to a given task. Some can reduce the loss more than others. For an ImageNet model, where classifying different species of dogs is a central task, a floppy ear detector might be one of the most important features it can have. In contrast, another feature might only very slightly improve performance.[7]

Concretely, our synthetic data is defined as follows: The input vectors x are synthetic data intended to simulate the properties we believe the true underlying features of our task have. We consider each dimension x_i to be a "feature". Each one has an associated sparsity S_i and importance I_i. We let $x_i = 0$ with probability S_i, but is otherwise uniformly distributed between $[0, 1]$.[8] In practice, we focus on the case where all features have the same sparsity, $S_i = S$.