THE MODEL ($X \to X'$)

We will actually consider two models, which we motivate below. The first "linear model" is a well understood baseline which does not exhibit superposition. The second "ReLU output model" is a very simple model which does exhibit superposition. The two models vary only in the final activation function.

Linear Model	ReLU Output Model
$h = Wx$ $x' = W^T h + b$	$h = Wx$ $x' = \text{ReLU}(W^T h + b)$
$x' = W^T W x + b$	$x' = \text{ReLU}(W^T W x + b)$

Why these models?

The superposition hypothesis suggests that each feature in the higher-dimensional model corresponds to a direction in the lower-dimensional space. This means we can represent the down projection as a linear map $h = Wx$. Note that each column W_i corresponds to the direction in the lower-dimensional space that represents a feature x_i.

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.

We also add a bias. One motivation for this is that it allows the model to set features it doesn't represent to their expected value. But we'll see later that the ability to set a negative bias is important for superposition for a second set of reasons – roughly, it allows models to discard small amounts of noise.

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.

THE LOSS

Our loss is weighted mean squared error weighted by the feature importances, I_i, described above:

$$L = \sum_x \sum_i I_i (x_i - x'_i)^2$$

Basic Results

Our first experiment will simply be to train a few ReLU output models with different sparsity levels and visualize the results. (We'll also train a linear model – if optimized well enough, the linear model solution does not depend on sparsity level.)

The main question is how to visualize the results. The simplest way is to visualize $W^T W$ (a features by features matrix) and b (a feature length vector). Note that features are arranged from most important to least, so the results have a fairly nice structure. Here's an example of what this type of visualization might look like, for a small model model ($n = 20;\ m = 5;$) which behaves in the "expected linear model-like" way, only representing as many features as it has dimensions: