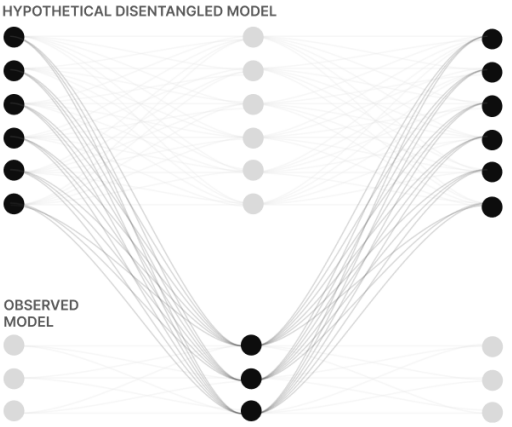

Can neural networks do useful computation in superposition, in addition to just representing features?

We replace a single layer of an imagined larger network with a lower dimensional model and study what happens.

Specifically, we'll have the model compute $y = \text{abs}(x)$. Absolute value is an appealing function to study because there's a very simple way to compute it with ReLU neurons: $\text{abs}(x) = \text{ReLU}(x) + \text{ReLU}(-x)$. This simple structure will make it easy for us to study the geometry of how the hidden layer is leveraged to do computation.

Since this model *needs* ReLU to compute absolute value, it doesn't have the issues the model in the previous section had with trying to avoid the activation function.

Experiment Setup

The input feature vector, x, is still sparse, with each feature x_i having probability S_i of being 0. However, since we want to have the model compute absolute value, we need to allow it to take on non-positive values for this to be a non-trivial task. As a result, if it is non-zero, its value is now sampled uniformly from $[-1, 1]$. The target output y is $y = \text{abs}(x)$.

Following the previous section, we'll consider the "ReLU hidden layer" toy model variant, but no longer tie the two weights to be identical:

$$h = \text{ReLU}(W_1 x)$$

$$y' = \text{ReLU}(W_2 h + b)$$

The loss is still the mean squared error weighted by feature importances I_i as before.

Basic Results

With this model, it's a bit less straightforward to study how individual features get embedded; because of the ReLU on the hidden layer, we can't just study $W_2^T W_1$. And because W_2 and W_1 are now learned independently, we can't just study columns of W_1. We believe that with some manipulation we could recover much of the simplicity of the earlier model by considering "positive features" and "negative features" independently, but we're going to focus on another perspective instead.

As we saw in the previous section, having a hidden layer activation function means that it makes sense to visualize the weights in terms of neurons. We can visualize W directly or as a neuron stack plot as we did before. We can also visualize it as a graph, which can sometimes be helpful for understanding computation.