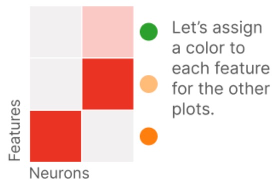
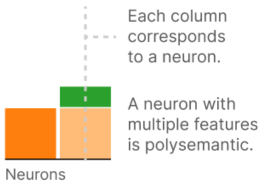
W as Matrix

Since the hidden layer now has a priviliged basis can visualize the raw weight matrix.



W as Stack Plot

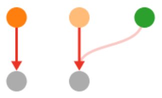
Instead of showing a matrix, we can map features to colors and stack the weights per neuron.



Each column corresponds to a neuron.
A neuron with multiple features is polysemantic.
Neurons

W as Graph

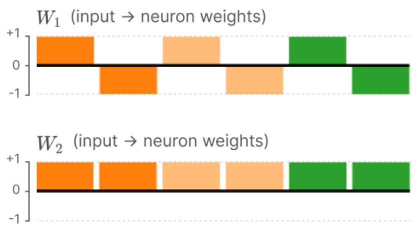
We can also visualize weights as the edges of a graph, as is often done for neural nets.



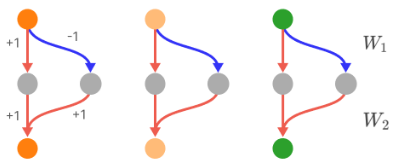
Let's look at what happens when we train a model with $n = 3$ features to perform absolute value on $m = 6$ hidden layer neurons. Without superposition, the model needs two hidden layer neurons to implement absolute value on one feature.

The resulting model – modulo a subtle issue about rescaling input and output weights[15] – performs absolute value exactly as one might expect. For each input feature x_i, it constructs a "positive side" neuron $\mathrm{ReLU}(x_i)$ and a "negative side" neuron $\mathrm{ReLU}(-x_i)$. It then adds these together to compute absolute value:

Raw Weights Visualization



Graph Visualization



Superposition vs Sparsity

We've seen that – as expected – our toy model can learn to implement absolute value. But can it use superposition to compute absolute value for more features? To test this, we train models with $n = 100$ features and $m = 40$ neurons and a feature importance curve $I_i = 0.8^i$, varying feature sparsity.[16]

A couple of notes on visualization: Since we're primarily interested in understanding superposition and polysemantic neurons, we'll show a stacked weight plot of the absolute values of weights. The features are colored by superposition. To make the diagrams easier to read, neurons are faintly colored based on how polysemantic they are (as judged by eye based on the plots). Neuron order is sorted by the importance of the largest feature.