Toy Models of Superposition: A discussion of the seminal paper

Tristan Thomas
Department of Mechanical Engineering
Clemson University
Clemson, SC 29631
tthoma7@clemson.edu

Abstract

In deep learning, both for LLMs and for other neural network based models, there is a noticed effect where neurons may be activated for vastly distinct ideas/objects. In the context of LLMs, this could be having a neuron with high activation for text about Clemson Universtiy, and 1980s Western movies. Researchers in the field now known as mechanistic interpretability have become interested in trying to understand this phenomenon on the path to better understanding how these models really work. To do so, the authors in [2] build what they call toy models of superposition as a hypothesis for how this polysemanticity of neurons happens.

1 Introduction

As the success of deep learning based models built on neural network architectures have taken off for a large range of tasks, researchers have sought to understand these models better. While we have vast experimental results suggesting that these models work very well, understanding them better may allow us to create new architectures that work even better. There is still a large element of mystery of exactly how these models work and why we see certain emergent behaviors.

Here enters the field of mechanistic interpretability. This field can in some ways be traced back to the work of [3]. In this work, the authors study the way that neural networks learn images, and in particular if neurons themselves are a way to gain information about this interpretability, among other things.

For the course project, I instead study (some) of [2]. This paper was published in 2022 by a group of authors at Anthropic. The idea of the paper is to explore the phenomenon knows as polysemanticity, where a neuron tries to learn multiple features. The authors present a way to understand this idea using their proposed concept called superposition. Since this work is more on the conceputal/theoretical side, my goal in this project was to mainly understand this paper. The demonstrating examples in the paper are intentionally not complicated. After the idea is built up, they show its application to small networks on the order of 10s of neurons rather than hundreds or thousands. The code behind this work is publically available (`https://github.com/anthropics/toy-models-of-superposition`). While I did run this code myself, I will likely borrow figures from the original paper as they are rendered better than they are inline in the notebook. I will however, provide my own explanations of (some) of the code.

2 Preliminaries

The authors in the paper use a number of definitions throughout the work. These are taken from [2] verbatim:

Preprint. Under review.

- Decomposability: Network representations can be described in terms of independently understandable features.
- Linearity: Features are represented by direction.
- Privileged Basis: Only some representations have a *privileged basis* which encourages features to align with basis directions (i.e. to correspond to neurons).
- Superposition: Linear representations can represent more features than dimensions, using a strategy we call *superposition*. This can be seen as neural networks *simulating larger networks*. This pushes features *away* from corresponding to neurons.

The authors also make a point to discuss how they think of features. They define features under three working definitions that are summarized here. First they describe features as arbitrary functions. In other words, a feature is any function of the input. Another notion they work with is features as interpretable properties. This would be something that could be described as understandable to a human. The last notion that they use is that of neurons in sufficiently large models. The idea is that if there are enough neurons in a model, said succinctly as "if the model is sufficiently large," than a neuron will be dedicated to each feature. They state that in this definition the hope is that polysemantic neurons which represent multiple features as once, would instead dedicate a neuron to each feature if the network size was increased enough. This idea of feature is ultimately what the authors use as their way of thinking for this work.

As far as mathematical preliminaries go, standard vector/matrix notation is used where $x \in \mathbb{R}^n$ would represent a n-dimensional vector of real numbers. Similarly, $y \in \mathbb{R}^{n \times m}$ represents a matrix with n rows and m columns, where each entry is a real number.

3 More Important Concepts

This section will go into detail on a couple of important concepts for the discussion of the experiments.

3.1 Features as Directions

The paper gives the example of word embeddings corresponding to directions as we see $V(\text{"king"}) - V(\text{"man"}) + V(\text{"woman"}) = V(\text{"queen"})$. Another example being interpretable neurons, as we already know that neurons have a basis direction in the activation space, and when these neurons are interpretable, each feature aligns with a particular neuron direction, thus it also has a direction. Then, we can talk about a neural network representation being linear if features do really correspond to having a direction in activation space (meaning they have a direction that aligns with neuron directions). Specifically, each feature f_i will have a direction W_i. If you have multiple features, all activating with a value x_{f_i}, then we might have something like $x_{f_1} * W_1 + x_{f_2} * W_2 \ldots$. Thus, we have a linear map from features to activation vectors, despite the features are likely nonlinear functions of the input.

The authors posit that neural networks empirically have linear representations. They specifically discuss three benefits of having linear representations. The first being that a linear representation is naturally what comes from obvious algorithms. For example, a neuron will activate more if the thing it is trying to represent is present, and less otherwise. Another reason for linear representations is that linear representations allow for easily making accessible features in subsequent layers. Finally, these representations allow for non-local generalization, as unseen features may be simple linear combinations of learned features.

3.2 Privileged vs Non-privileged Bases

When we study things like word embeddings, basis directions don't matter as the relative distance between the vectors is how meaning is defined. This is what is meant by non-privileged basis. A privileged basis, as simply defined earlier, has incentive for features to align with basis directions. This can be due to something like an activation function. Because of this, it only makes sense to talk about neuron interpretability if the layer has a privileged basis. There is something special in those directions because they are implicitly encouraged. These basis dimensions the authors call neurons, as it is sensible that each neuron will have it's own direction in the activation space. This can be seen

in the following thought experiment: take a layer with 100 layers so that your activation space is $\mathbb{R}^{100}$. The space R^{100} has a basis dimension of 100. Thus, it is sensible that each basis coordinate would simply be the activation of the corresponding neuron. Of course the authors note that having a privileged basis does not actually guarantee that features will be aligned with the basis. This leads into the superposition hypothesis as we might ask when features are aligned with the basis directions (neurons). However, this question only makes sense to ask if we are dealing with a privileged basis.

4 The Superposition Hypothesis

So far we have discussed privileged bases, talked about when neurons being polysemantic in certain cases where they seem to represent unrelated things at times. How do we tie all this together?

That is where we enter the superposition hypothesis. The rough description of this is that "neural networks want to represent more features than they have neurons," as it is put in the paper. The hypothesis is that when this is the case, they do so by effectively simulating a model with m ore neurons.

The authors argue that this is at least plausible because of the Johnson-Lindenstrauss Lemma. This lemma is not proved here, but a proof is given in [1] as well as in the original paper. We also do not state the formal statement of the lemma here, as it is a bit cumbersome for how it is used in this work, but in essence the lemma states that in high dimensional spaces, it is possible to have $exp(n)$ almost orthogonal vectors in n-dimensional space. So while we can not have more than n orthogonal vectors in n-dimensional space, we can have an extremely large number of almost orthogonal vectors where almost orthogonal is defined in terms of having $< \epsilon$ cosine similarity. Additionally the authors provide the example of compressed sensing, where projecting a vector into low dimensional space and then trying to recover it is difficult. However, if the vector is sparse, it becomes possible. This ideas is very relevant to the conclusion of some of the experiments in this paper.

Based off of this theorem, we can loosely conclude that if the features outnumber the neurons, we will have these features being represented as almost orthogonal directions in the activation space. This of course has the consequence that the activation of a particular feature means that other features may look weakly activated. However, if features are sparse, meaning that if one feature is active it is likely that all others are not active, then it may be "worth it" for the neural network to still embed these features in the activation space and allow the interference. This is because the representation of extra features may outweigh the downside of having these extra representations look like false presence of other features. We demonstrate how this interference happens with a basic example.

- Setup: To understand the left figure, we define a concrete example with 3 features (Universities) packed into 2 neurons.
- Feature Directions: We assign a vector in 2D space for each university:
  - Clemson: $v_1 = (\frac{\sqrt{3}}{2}, \frac{1}{2})$
  - Duke: $v_2 = (-\frac{\sqrt{3}}{2}, \frac{1}{2})$
  - NC State: $v_3 = (0, -1)$
- Weight Matrix (W): Stacking these as columns gives us our encoding matrix:

$$W = \begin{pmatrix} \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} & 0 \\ \frac{1}{2} & \frac{1}{2} & -1 \end{pmatrix}$$

- In low dimensional space, we can't have all three vectors be even close to mutually orthogonal.
- We project the Duke vector onto the Clemson vector:

$$\text{proj}_{\text{C}}(\text{D}) = \underbrace{\left(\begin{pmatrix} -\frac{\sqrt{3}}{2} \\ \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} \frac{\sqrt{3}}{2} \\ \frac{1}{2} \end{pmatrix}\right)}_{\text{Dot Product}} \underbrace{\begin{pmatrix} \frac{\sqrt{3}}{2} \\ \frac{1}{2} \end{pmatrix}}_{\text{Direction}}$$

$$= \left(-\frac{3}{4} + \frac{1}{4}\right) \text{Clemson} = -0.5\text{Clemson}$$

- Result: It is obvious from this derivation that activating Duke will have an effect on the apparent activation of Clemson.

Another issue that can happen with representing more features is that along with intereference, it can be ambiguous as to what feature(s) was truly present. We can also build off the previous example to show this:

- Activate Clemson and Duke both at **0.9**.
- $\begin{pmatrix} \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} & 0 \\ \frac{1}{2} & \frac{1}{2} & -1 \end{pmatrix} \begin{pmatrix} 0.9 \\ 0.9 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0.9 \end{pmatrix}$
- From previous results, if you de-embed this vector you will get half the Clemson feature value (because the Duke projects on and cancels half).
- NC State Result: How does this look to the NC State feature $\begin{pmatrix} 0 \\ -1 \end{pmatrix}$?

$$\text{Activation} = \begin{pmatrix} 0 \\ 0.9 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ -1 \end{pmatrix} = -0.9$$

- Conclusion: The presence of Clemson and Duke creates an ambiguous signal of -0.9 on the NC State channel.
- Recovery ($x' = W^T h$): We get the activation of each feature by multiplying our activation vector by the transpose of the feature matrix.

$$\begin{bmatrix} x'_{\text{Clemson}} \\ x'_{\text{Duke}} \\ x'_{\text{NC State}} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} \\ 0 & -1 \end{bmatrix}}_{W^T \text{ (Feature Rows)}} \underbrace{\begin{bmatrix} 0 \\ 0.9 \end{bmatrix}}_{\text{Activation } h} = \begin{bmatrix} 0.45 \\ 0.45 \\ -0.9 \end{bmatrix}$$

- Results:
  - Clemson & Duke: Recovered at 0.45
  - NC State: Introduced a nonzero value (-0.9).
- Interpretation: We now have no idea what features were really present at the input. A problem due to not having sparsity.

Note that if sparsity is present, then both of these issues diminish. Another way of stating the importance of sparsity is that if sparsity increases, the effects of interference or reconstruction issues will be diminished.

5 Demonstrating Superposition

5.1 Setup

Now that the idea of superposition has been described, and potential impacts of its presence have been analyzed, we now want to illustrate it actually happening. Finally, we have arrived at the "toy models of superposition" mentioned in the title of this work.

The authors mention that for this hypothesis to be at all valid, it must be possible to show that a neural network can noisily represent more features than there are neurons.

The experiment details are as follows:

- Can we project a vector from high dimensional space to low dimensional space and then recover it?
- Data is based on each x_i having a sparsity S_i and important I_i. The sparsity is defined as $:x_i = 0$ with probability S_i, otherwise uniformly sampled from $[0, 1]$. For the importance, these will be used as weighting terms in a cost function.

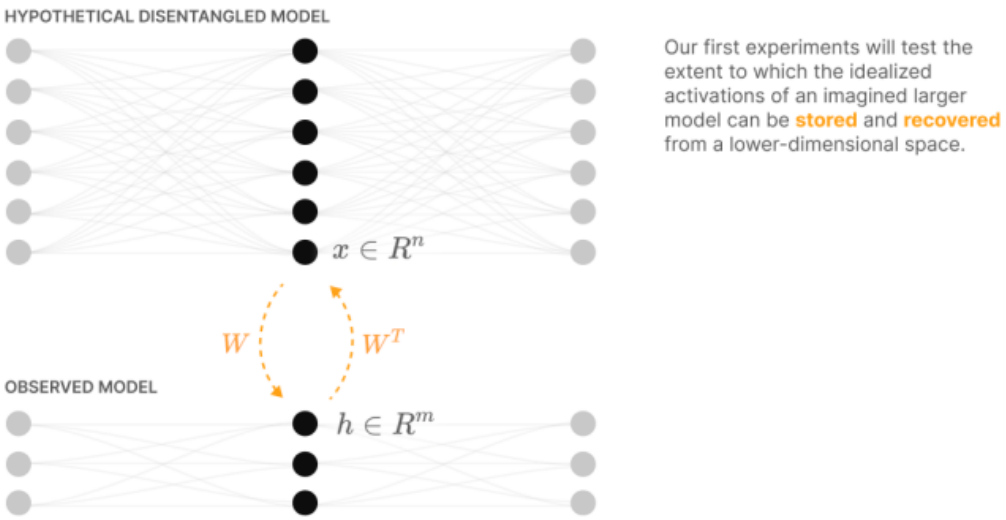

Figure 1: Figure borrowed from [2] giving a visual depiction of the goal

To discover when superposition might emerge, the following two models are used in the experiments:

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

The loss function is defined as:

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

5.2 Results

First, we shall discuss how results are visualized in this work. These include the following ideas:

- Visualizing $W^T W$. This matrix consists of all pairwise dot products of feature vectors. Thus, if the features all have a distinct direction, we expect that the diagonal entries will be equal to 1 (they will always be 1), and the other entries should be close to 0. Off diagonal entries not being close to 0 implies superposition, as this means the network is starting to give more features and embedding than there are neurons.
- We visualize $\sum_{j \neq i} (\hat{W}_i \cdot W_j)^2$ where $\hat{W}$ is the matrix with normalized features. This sum is essentially summing up for a fixed i, all dot products of feature i with all other (normalized) features and summing them up. This effectively sums up the intereferences of all other features with feature i. If this sum is greater or equal to 1, that means a group of features can activate the ith feature as strongly as the feature itself.
- The length of the bars in a particular plot will also represent the norm or length of the feature vectors.

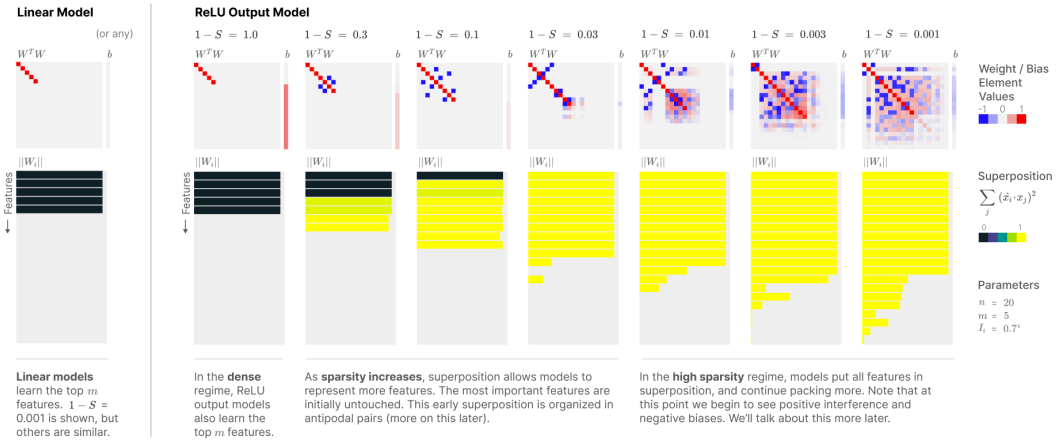

Figure 2: Figure borrowed from [2].This shows all of the visualizations described in this section for both model options at different sparsity levels.

With this in mind, we look at the results in Figure 2. Note that results were reimplemented from the given code. We slightly modified the given code to better match the paper in terms of the number of features and hidden dimension. Those results do not have the nice labeling and formatting, but similar things can be seen as this figure from the paper.

In terms of the results of this experiment, the results are quite clear. We first notice that a purely linear network will simply act like a principle component analyis (PCA). Once ReLu is added, we see that with no sparsity, the PCA behavior is maintained. However, as sparsity increases, more and more features are being represented as can be seen with both visualizations. This clearly demonstrates that with enough sparsity, a neural network is able to see the benefit of representing more features despite the possible problems of interference and recovery. Thus, in this work, we have successfully understood the hypothesis made by the authors. Namely, that nonlinearity and sparsity in neural networks is able to allow superposition to occur.

6 Conclusion

In this work, we summarize some of the key ideas and results of part of [2]. Note, that what was touched on here, is a small fraction of the full results in the original paper. In that work, they also show superposition as a phase change, and talk about the geometry of feature embeddings in a much more extensive way than what was discussed here. We thank the authors of [2], as much of the content in this document was adapted from that work, and at times used directly. This project was a great experience to learn about a field within LLMs called mechanistic interpretability of which I was not previously familiar.

References

[1] Sanjoy Dasgupta and Anupam Gupta. "An elementary proof of a theorem of Johnson and Lindenstrauss". In: *Random Structures Algorithms* 22.1 (2003), pp. 60–65. ISSN: 1098-2418. DOI: 10.1002/rsa.10073. URL: http://dx.doi.org/10.1002/rsa.10073.

[2] Nelson Elhage et al. *Toy Models of Superposition*. 2022. arXiv: 2209.10652 [cs.LG]. URL: https://arxiv.org/abs/2209.10652.

[3] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. "Feature Visualization". In: *Distill* (2017). https://distill.pub/2017/feature-visualization. DOI: 10.23915/distill.00007.