Effects of Superposition on Latent Space Visualization

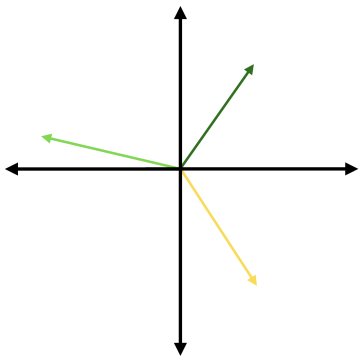
Chris Hamblin

2070 midterm – 2022-10-06

This project is a direct extension of this paper on 'Superposition'. It's a long paper, so first I will give a brief synopsis of what they did, and re-implement some of their experiments. This involves training small neural networks on simulated data, which I did in Pytorch. This code can be run without any setup using this colab notebook. All the code/data for this project can be found in this github repo.Visualizations of the latent spaces (saved in the data file) of these models are then performed in R (Hamblin.R).

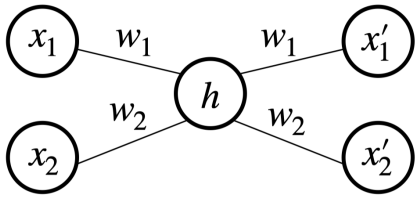
What is Superposition?

Latent Features: Superposition is a proposed mechanism by which a neural network layer can represent more 'features' than it has dimensions. While this could be a quite general phenomena, let us, for the sake of concreteness, consider a neural network that takes in pictures and outputs object labels. What are 'features' in this case? In the output 'label' space, features correspond to semantic labels selected by the modeler. By design, each label feature is encoded by a unique dimension of the output space in a performant model. In an Imagenet trained model for example, output dimensions encode features of the input image like 'stingray', 'Tibetan mastiff', and 'oil filter'. The features encoded by the dimensions in the input pixel space are also unproblematic; they represent the luminous intensity of light passing through each cell in 2D grid. The features encoded by the hidden layers of the neural network are more mysterious. There are many interesting things an algorithm might need to represent en route when transforming light into stingrays and oil filters, like 'shiny', or 'eye', or 'sky', or 'wooden' etc. etc. Of course when designing the model architecture, the modeler likely chose arbitrary, large dimensionalities for the hidden layers, say 4000 for a given layer. Does that mean the layer represents 4000 features, one along each dimension of the hidden layer? Surely not, the layer may not have an ideal basis and the meaningful features correspond to non-basis directions in the space. But then a simple change of basis would suffice to find the 4000 features. Can the layer encode *more* than 4000 features? It would seem to require some sort of interference, as not all of the feature directions could be orthogonal with each other. I cannot for example, draw 3 lines in 2 dimensions such that they are all 90 degrees to each other.



Superpostion is the way a neural network represents m features with n dimensions when $m > n$. It turns out that optimal solutions (the m directions for each feature that minimize the loss of the encoding) correspond to very particular geometric arrangements. When certain conditions are met (conditions that are very reasonable to grant for a neural network doing a task like visual object recognition), latent layers will put disparate features in *superposition*, such that they are encoded in the same direction of the latent space. I reason that this could be problematic for common latent space visualization techniques deployed on deep nets, like MDS, t-SNE, and UMAP, as even the high dimension space being projected maybe putting the *true*, disparate features close together in superposition. But before we get to my experiments, lets build up a little intuition for why superposition happens by probing the simplest case; 2 features in 1 dimension.

2 Features in 1 Dimension: We can think of optimally representing 2 features in 1 dimension as constructing a mapping function that first collapses the 2 features ($\mathbf{x} = [x_1, x_2]$) into 1 number (h), then expands h into 2 numbers again $\mathbf{x}' = [x'_1, x'_2]$ with minimal loss ($L = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2}$). We could image a simple linear model that does this by first taking the dot product with weights W to get h, then multiplying by W^T to get $\mathbf{x}'$;



as network

$$h = Wx$$
$$x' = W^T h$$
$$x' = W^T W x$$

linear model

$$x'_1 = w_1 w_1 x_1 + w_1 w_2 x_1$$
$$x'_2 = w_2 w_1 x_1 + w_2 w_2 x_2$$

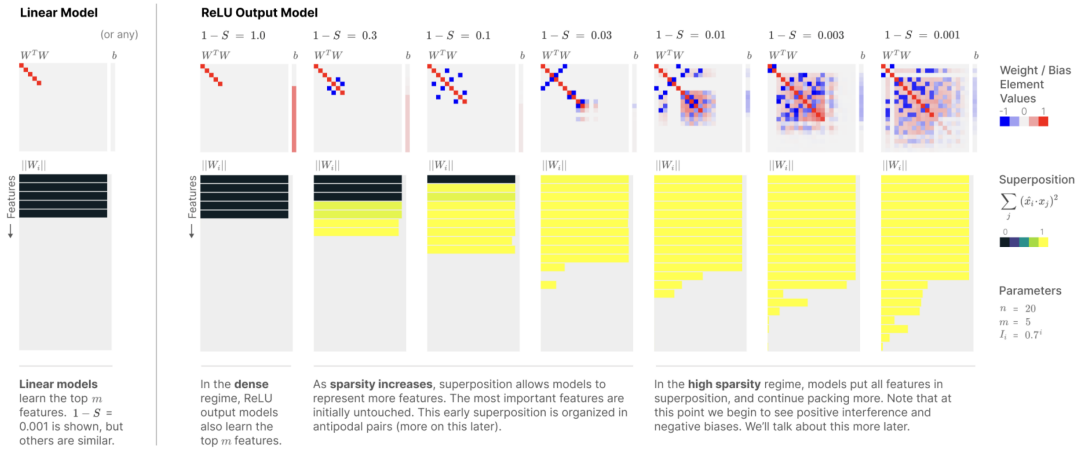
expanded

Let's consider some possible settings for W. We can choose to represent one feature, but not the other by setting $W = [1, 0]$, or $W = [0, 1]$. This will pass one feature through to the output without interference, but completely ignore the other feature. Perhaps we could represent both features with $W = [1, -1]$. Unfortunately this doesn't work, as then x_2 negatively interferes with x'_1, and visa-versa. What if we make some presumptions about the distribution of our features? For high-level visual features, we expect sparseness (the feature is usually 0) as a given image usually does not contain a high level semantic concept. Lets consider the superposition solution ($W = [1, -1]$) under the premise of sparsity. If x_1 is present we can now *usually* represent it, as though x_2 negatively interferes, its probably not simultaneously present. However, we are still not all the way there, as x_2, the *absent* feature, is still interfered with; $x'_2 = w_2 w_1 x_1 + w_2 w_2 x_2 = -1 * 1 * 1 + -1 * -1 * 0 = -1$. The final consideration is the effects of including ReLU's in our network, suppose we used the encoding model $x' = ReLU(W^T W x)$ instead. Then the absent feature is *not* interfered with as it was above, as the -1 goes to 0! This ReLU model still cannot encode the simultaneous presence of features ($x = [1, 1]$) with superposition ($W = [1, -1]$); it will return $x' = [0, 0]$. However, the simultaneous presence of features is least likely if the features are sparsely distributed, in which case we should expect superposition to be the optimal solution.

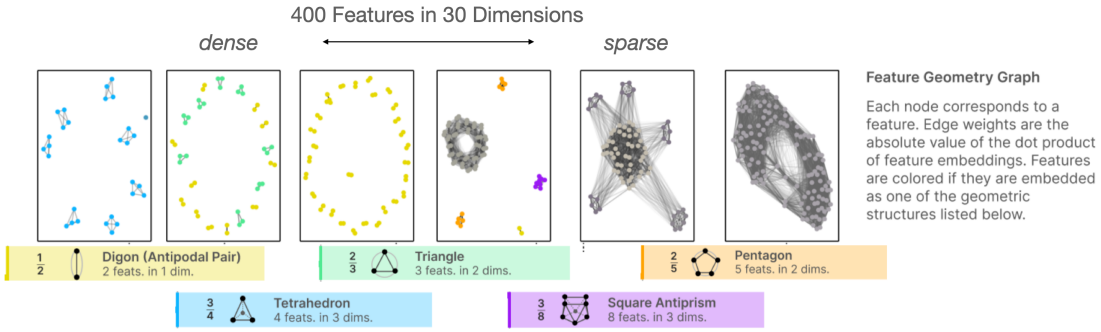
$$W = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \quad W^T W = \begin{bmatrix} w_1 w_1 & w_1 w_2 \\ w_2 w_1 & w_2 w_2 \end{bmatrix}$$

$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ -1 \end{bmatrix}$
$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$
x_1 not x_2	*x_2 not x_1*	*Superposition: x_1 or x_2, but not both at once*

So the superposition solution for representing 2 features on 1 dimensional is to assign each feature to one pole of the dimension. We would expect this solution to be optimal when the 2 features are sparsely distributed (features are usually 0, a reasonable assumption for things like visual features). Note that a hallmark of superposition is values on the off-diagonal of $W^T W$. The original authors tested this empirically (in a higher dimensional setting than 2 -> 1) , by tuning the W with gradient descent to minimize $L = \sum_{i=1}^{n} \sqrt{I_i (x_i - x_i')^2}$, where I is an importance weight assigned to each x. They used simulated data for x, and found exactly what we'd expect, that as sparsity increases, the ReLU model packs and more features into the low dimensional space, allowing for more and more interference/superposition (large inner product) between features. Here's a visualization from their paper of this phenomena;



2 features on 1 dimension results in the particular geometry of an 'antipole' when superposition occurs, with one feature on each pole of a single axis. But for different numbers of features and dimensions, many different geometries can emerge as optimal (again from their paper);

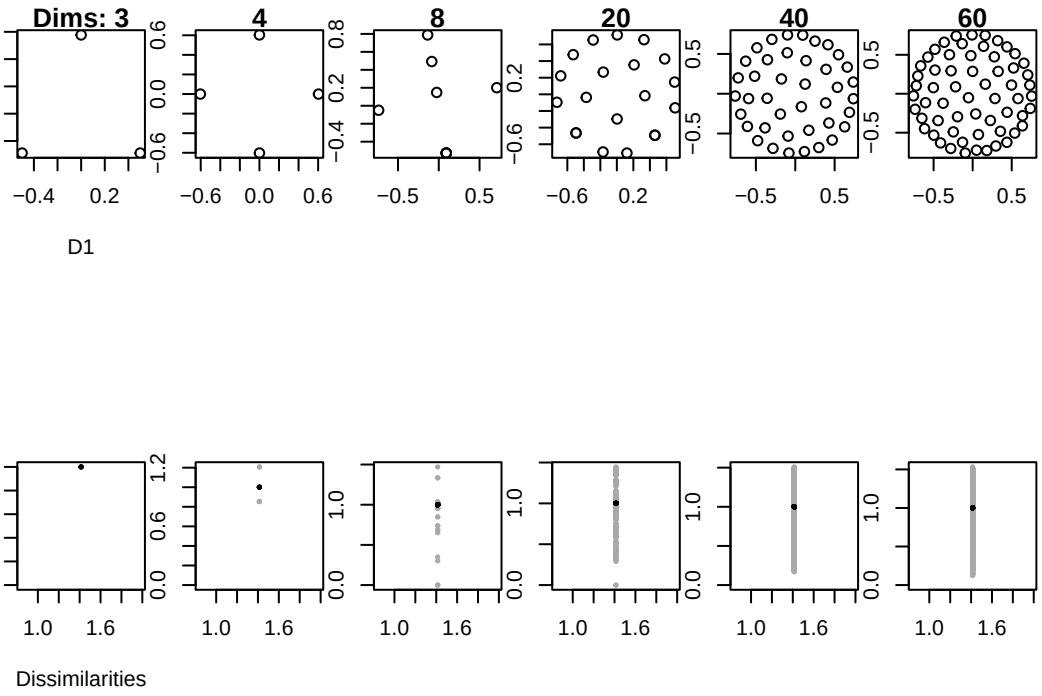


The original paper has a lot more substance, and I am starting to seriously gloss over things, but hopefully the general idea is clear. Let's move on and do some experiments of our own.

Dimensionality Reduction of Latent Spaces with Superposition

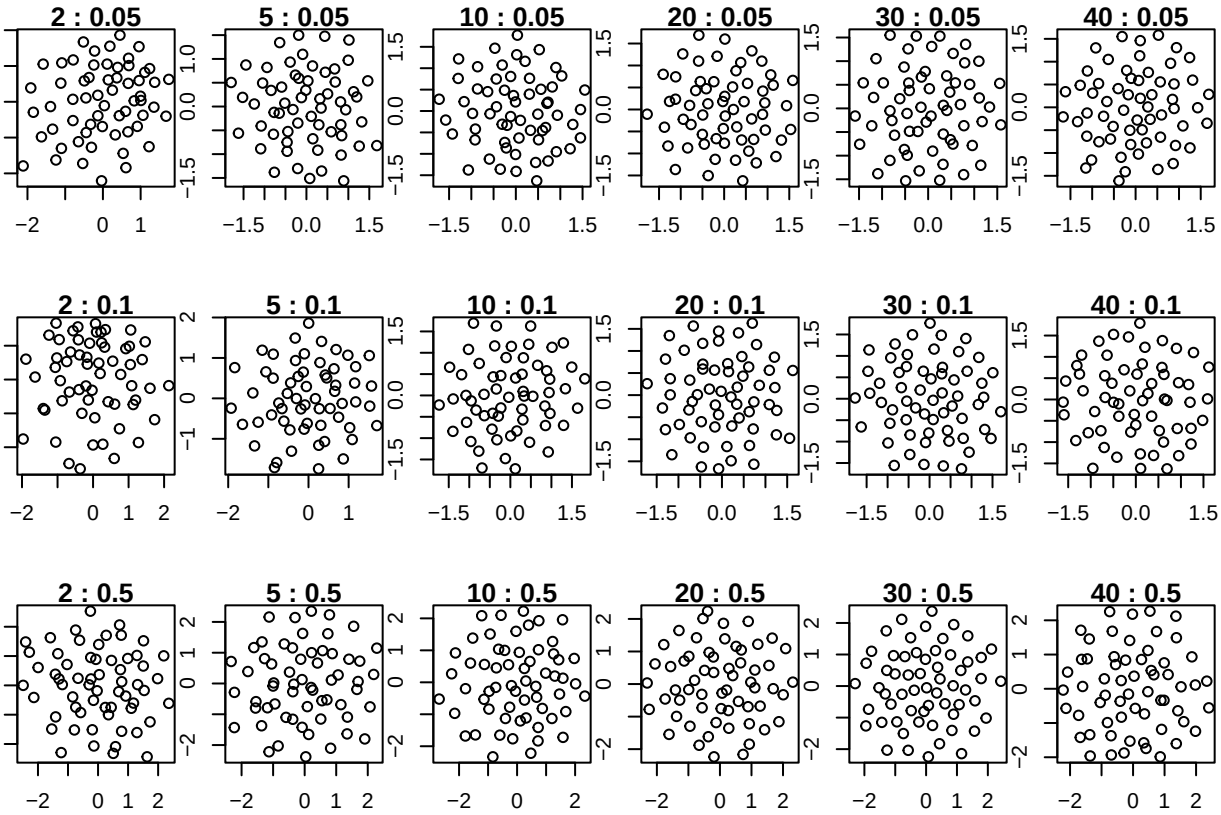
It is quite common in deep learning to 'visualize' data projected in latent spaces of a neural network with dimensionality reduction techniques like MDS (actually people usually use nonlinear variants like t-SNE and UMAP, which can be biased in various ways to preserve 'local' geometries, points close to each other). Superposition seems like it could present a huge problem here; ostensibly the modeler wants to visualize how data points relate to each other when loaded onto the 'real' (high-dimensional) features encoded by a latent layer, not the literal dimensions of the latent space, which might being compressing the 'real' features in superposition. Geometric structure in the MDS (UMAP, t-SNE) might simply be reflecting the superposition geometry, an artifact of compression. Here we will see if there are particular visual signatures of superposition in such plots.

Whats 'Correct' Geometry? No superposition We will take a similar approach to that employed in 'Toy Models of Superposition'. We are going to embed features (orthogonal , high-dimensional unit vectors) in superposition in a lower dimensional latent space. If we were to look at a distance preserving plot (MDS) of data in that low-dimensional embedding we want it to reflect the geometry of the *high-dimensional* features, not the embedding. What then, is the 'correct' MDS projection for orthogonal high-dimensional features? Let's just look at MDS plots of high dimensional orthogonal vectors. That will act as our target 'correct' MDS plot. Deviations from this plot when we run MDS on superposition embeddings of the high-dimensional data will correspond to artifacts introduced by superposition. So, we'll first run MDS on orthogonal data of dimension N, diag(N). The pairwise distance between each data point will all be $\sqrt{2}$ (if we use unit vectors for our data, follows from pythagorean theorem). Let's look at the MDS solution across N = [3,4,8,40,60]. Well plot shepard plots under each plot to see how good the fit is.

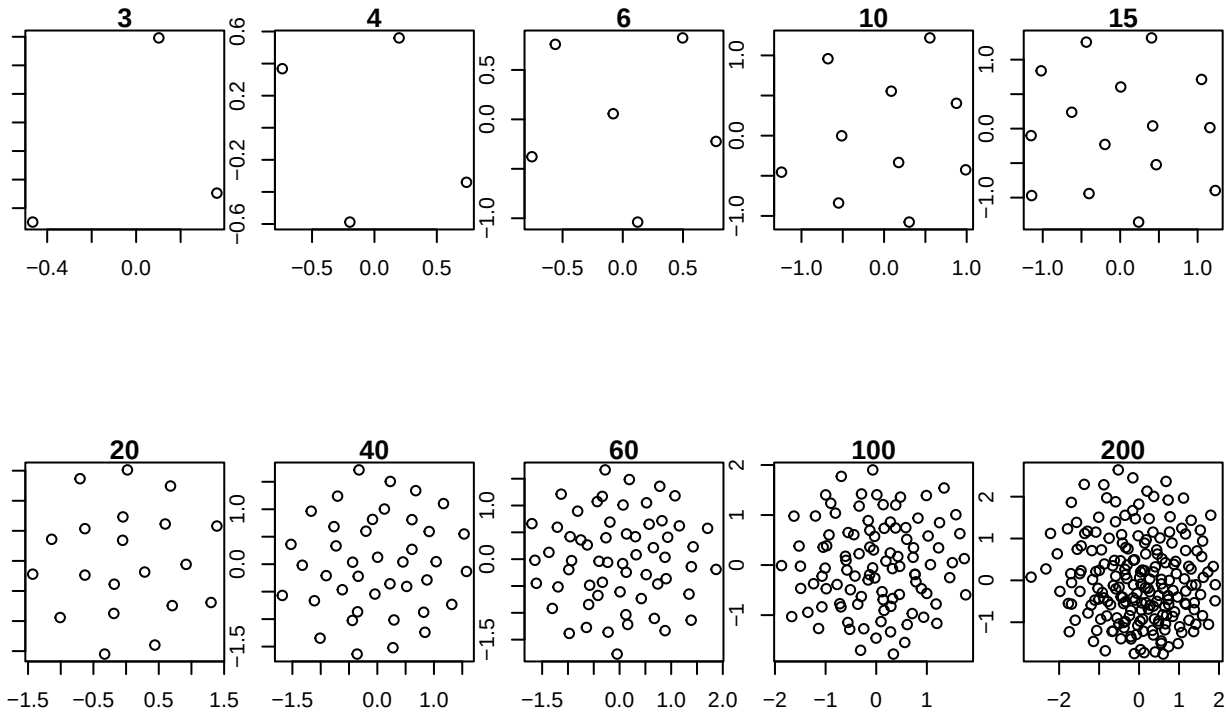


We see as N increases, the optimal distance preserving projection is a homogenous density circle (we talked about a related phenomena before after class one time). Here we also see some of the difference between pairwise distance and inner product; where 3 dimensions cant be projected into 2 and preserve inner product (as shown in the intro diagram), the equilateral triangle solution *does* preserve pairwise distances. The Shepard plots are a little silly here, as there is no disperion of the original pairwise distance, but in any case stress increases dramatically as we increase dimensionality, as we would expect.

We can do the same exact experiment with other commonly used dimensionality techniques for latent space visualization: tSNE and UMAP. These nonlinear methods should introduce some sort of biased prior, Well want to isolate the effect of what the prior might be first. So before we look at different dimension projects, lets do a hyperparameter sweep on $N = 60$ dimensions first. UMAP has two main hyper-parameters, n-neighbors and min-distance, which relate to the number of data points to include in a 'local' neighborhood and minimal distance allowed between data points in the output plot. We'll sweep across both parameters and label each plot as ''n_neighbors : min_distance'.

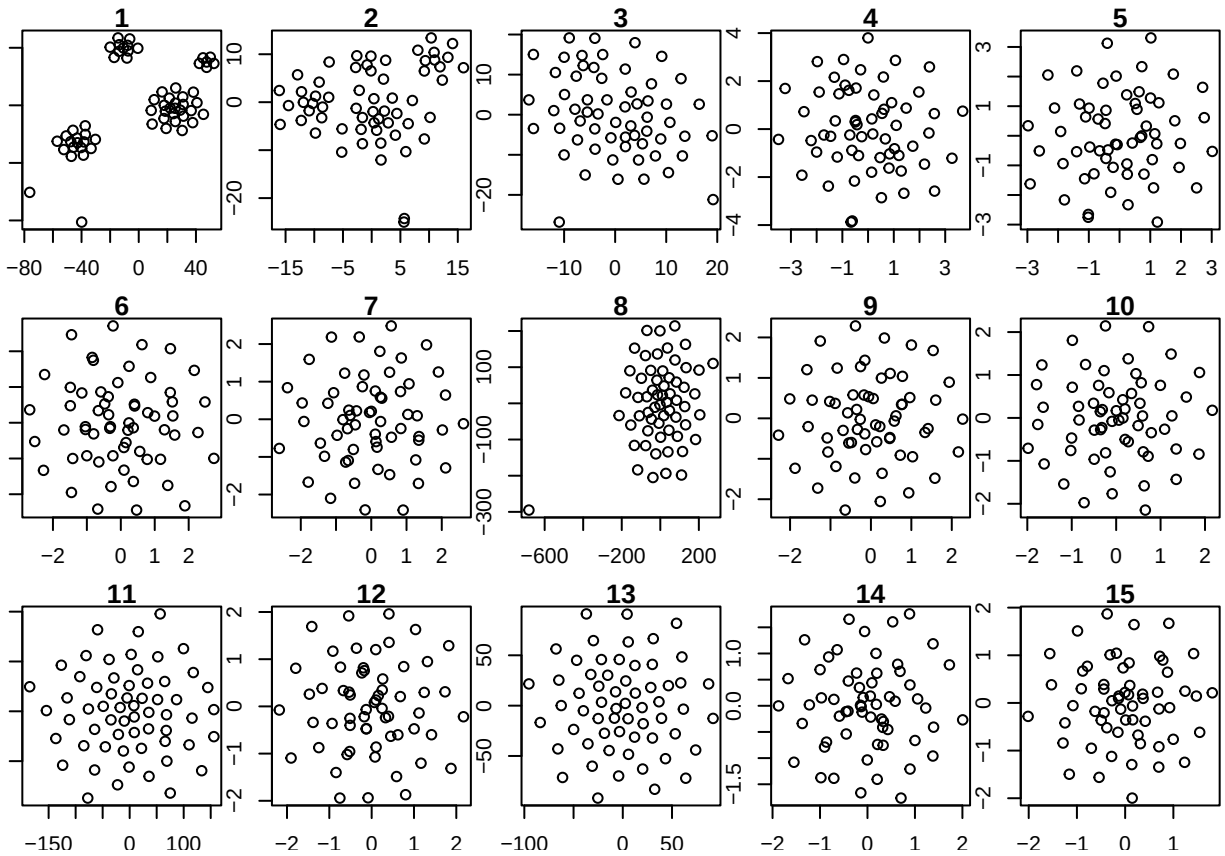


It seems like the relevant parameter is n_neighbors for our purposes, which decreases bias as it increases, which is too be expected (many neighbors means preserve global structure). Let's see the UMAP solution across dimensions;

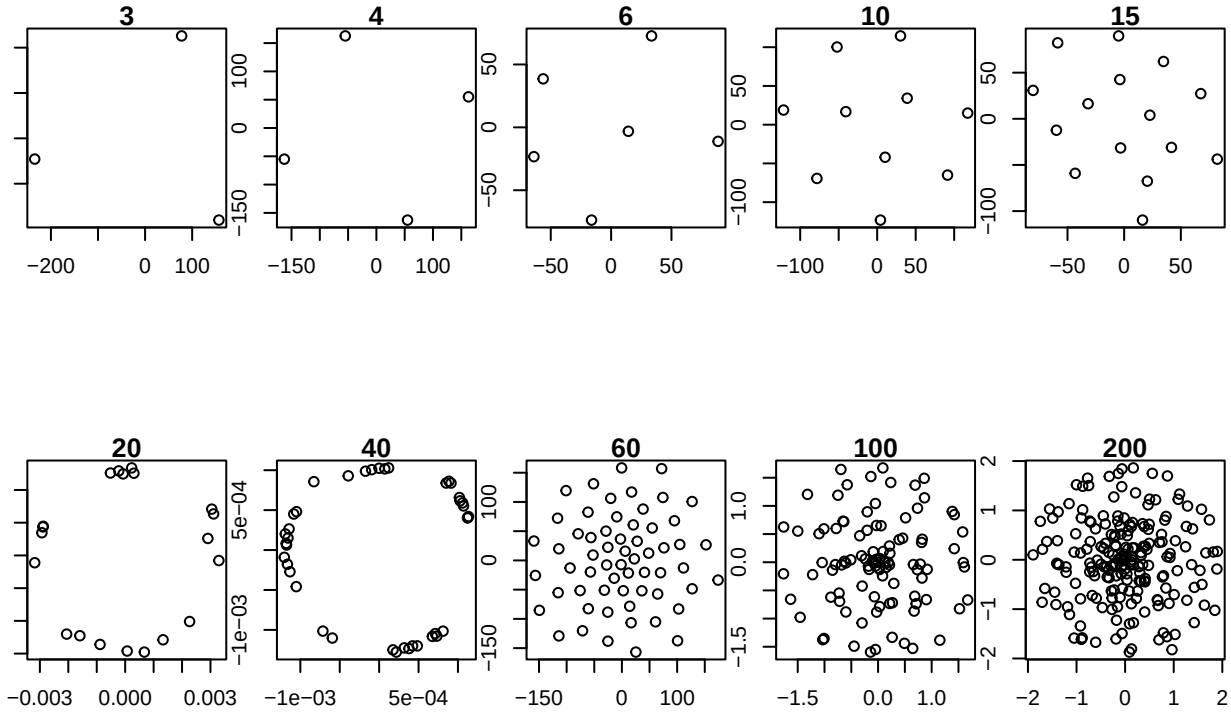


Looks very similar to MDS, allow it's introducing some noise in higher dimensions.

What about t-SNE? it has one relevant hyperparameter, perplexity:



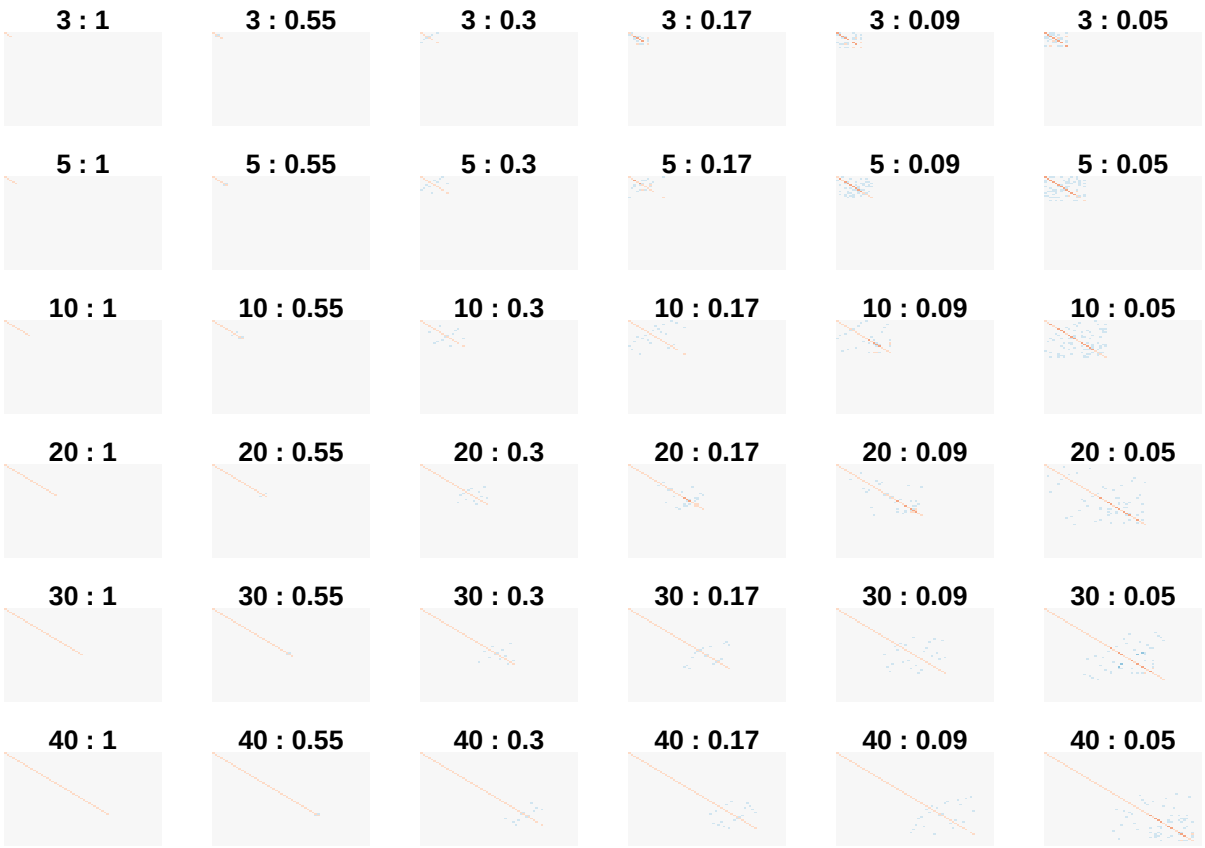
Low perplexity introduces substantial artifacts. Let's looks at t-SNE across dimensions (perpexity cannot be too high relative to dimensionality for t-SNE to run, so we keep $perplexity = floor(\frac{N-1}{3})$):



t-SNE clearly introduces major artifacts, lets avoid it going forward with our superposition experiments.

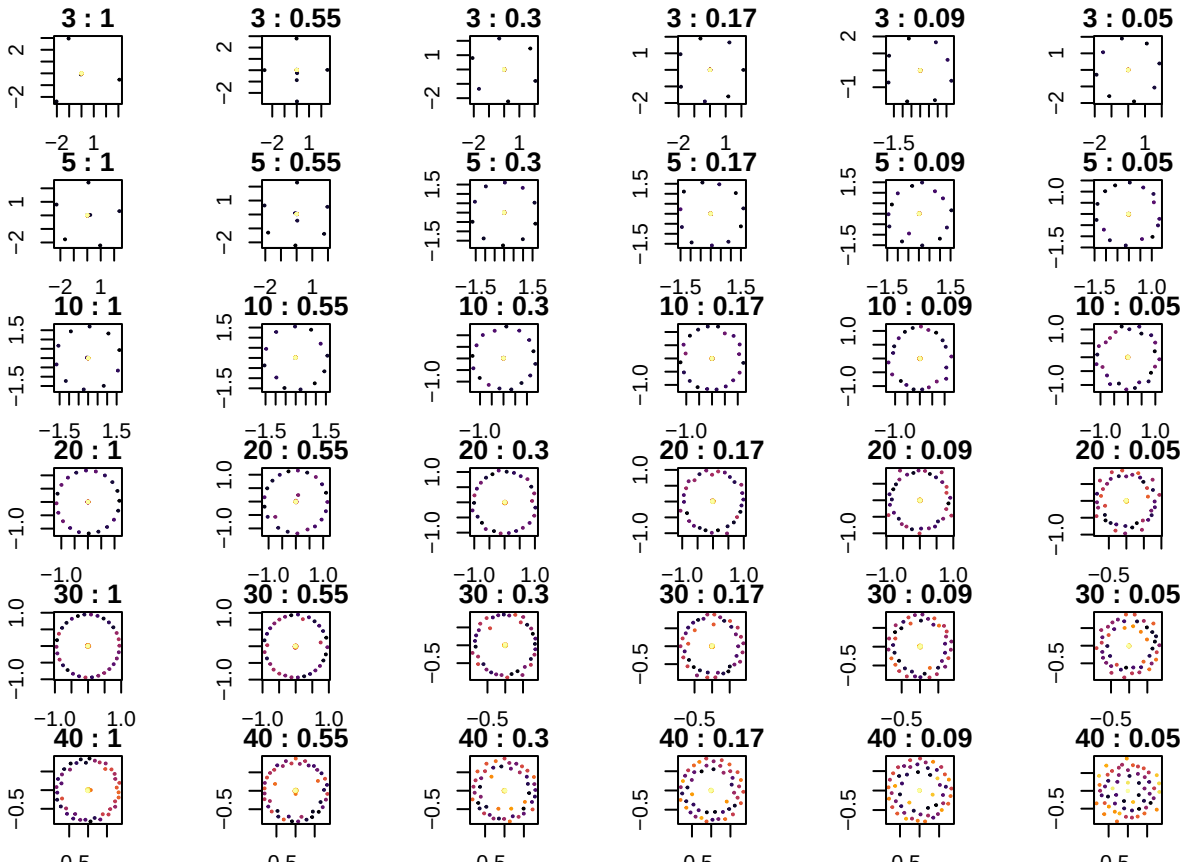
Solution on superposition data Okay, we now know how we want our latent space data to look under a distance preserving dimensionality reduction; as a homogenous density circle. But what does MDS (or UMAP) return when our features are first embedded in superposition before visualization? We will consider the case of $N = 60$ features, projected into several lower-dimensional embeddings $M = [50, 40, 30, 20, 10, 5]$. We will generate these embeddings using the 'ReLU' encoder/decoder model $x' = ReLU(W^T W x)$ where $W \in \mathbb{R}^{n \times m}$. Data point x in the original N dimensional space is encoded as the point Wx in the embedding space. The original N unit features are simply embedded onto the direction corresponding to each row of W. We will tune W with gradient descent to minimize $L = \sum_{i=1}^{n} \sqrt{I_i(x_i - x_i')^2}$, where I is an importance weight assigned to each x, where each I are between 1 and $1/20$, spaced exponentially.

We will sanity check our work by looking at plots of $W^T W$, they should have a positive diagonal, which gets longer and has more off-diagonal elements as sparsity increases.As the embedding dimension increases we should also expect the diagonal to get longer, as we can pack more dimensions. Each $W^T W$ plot will be labeled as 'embedding dimensionality : sparsity'.

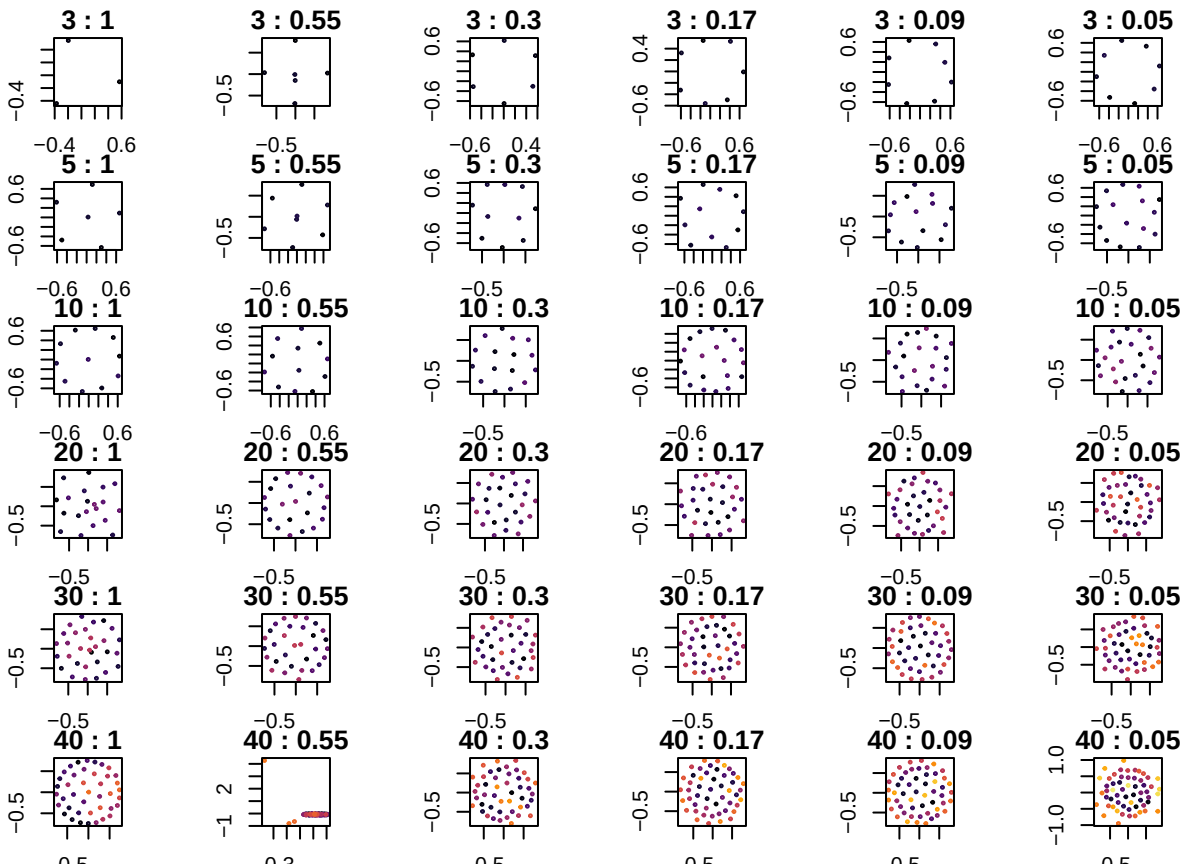


These look good (apologize for the small hard to read plots that are mostly gray). Interesting to note, as the hidden dimensions get very large (40) we can pack a lot of the 60 dimensions in without needing superposition at all, and the model super imposes the unimportant features first.

Now we'll see what MDS configurations of these embedded features are, by treating the rows of W as our data points. We will color each point by its importance (black most important, yellow least.)

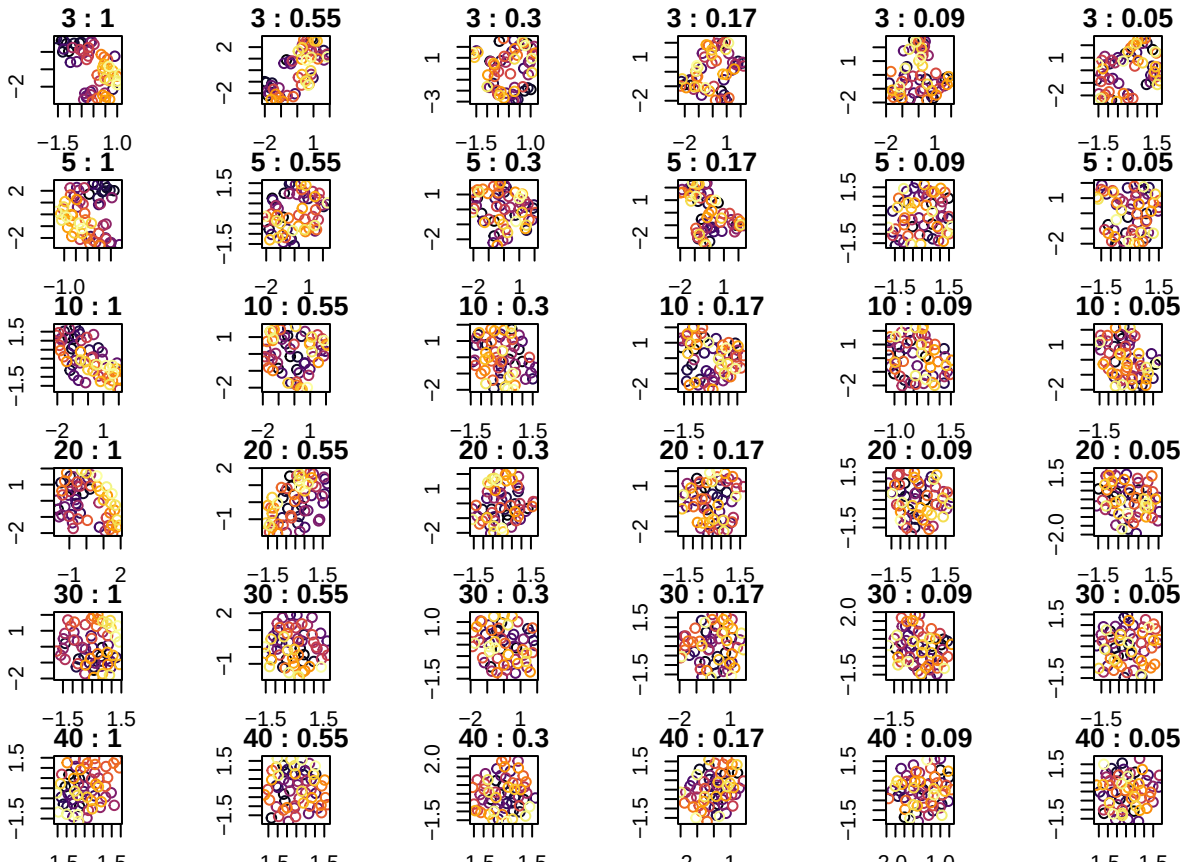


To be honest this result is not what I was expecting (or rather was hoping for?), although it suggests superposition might not be as large a problem for latent space visualization as I originally thought. The only signature in these plots different from the original MDS solution is the tight cluster of the unimportant features at the center of circle. This corresponds to rows *beyond* the diagonal in the corresponding W^tW plots above, features which get discarded from the embedding and are all mapped onto 0. I was expecting this behavior. What I was honestly not expecting was for all the points that are represented in the embedding, points that are not mapped to the origin in the MDS configuration plot, to be consistently mapped onto the perimeter of a circle. As we increase sparsity, if we are to believe the original paper, these points correspond to particular superposition geometries in the embedding space. But it would seem, these geometries are such that they lead to no obvious artifacts in the MDS. This is could probably be explored analytically, but I would have to think about it more. One possibility is that the circular geometry emerges only because of the presence of all the points at the origin. We could remove this points from consideration and see how it affects the configuration plot;



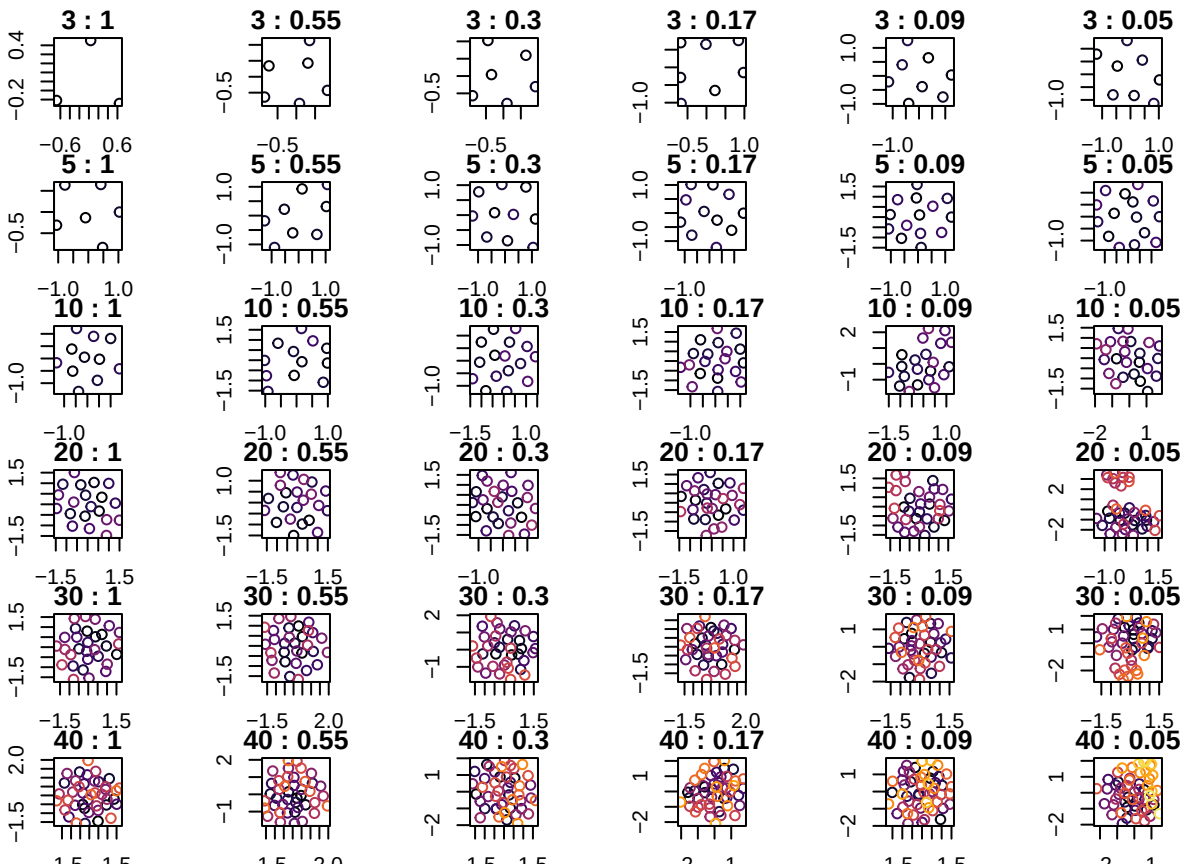
I say there is still no obvious artifact structure here. M=40 sparsity=.55 is a case of deviation, but its got just 3 big outlier points messing everything up, this could be an instance of our original optimization procedure that generated the embedding failing. Unclear, we certainly dont want to claim 'outliers + a uniform is ellipse is a signature of superposition'.

Finally let's check the same plots using UMAP. We'll keep the minimum distance very small (.00001), and the n_neighbors at the default of 15. We'll again plot across sparsities and embedding dimensions, including the points that map to zero, and then excluding them.



Looks like there are substantial artifacts with respect to the lower embedding dimensions, but without a consistent pattern. The low importance features, which should be mapped to 0 are smeared out close to each other, despite the very small min_dist hyperparameter. I think if I knew a little more about the UMAP algorithm I could make sense of this.

Now let's check the configuration excluding the origin points, as before. If there are less than 15 points to plot, we will decrease the number of neighbors to the number of points -1;



As with MDS, we only see obvious artifacts in particular instances (20 dims : .05 sparsity).

In conclusion, I dont think there are obvious worries we should have with respect to visualizing latent spaces that are in superposition. It would be nice to do some experiments like this with real data, that was my original idea, but is very tricky to define what the 'real' high-dimensional features are in such a case. I think that is why the authors choose to do their experiments by presupposing the high-dimensional features a priori. Look forwarding to chatting with you about this.