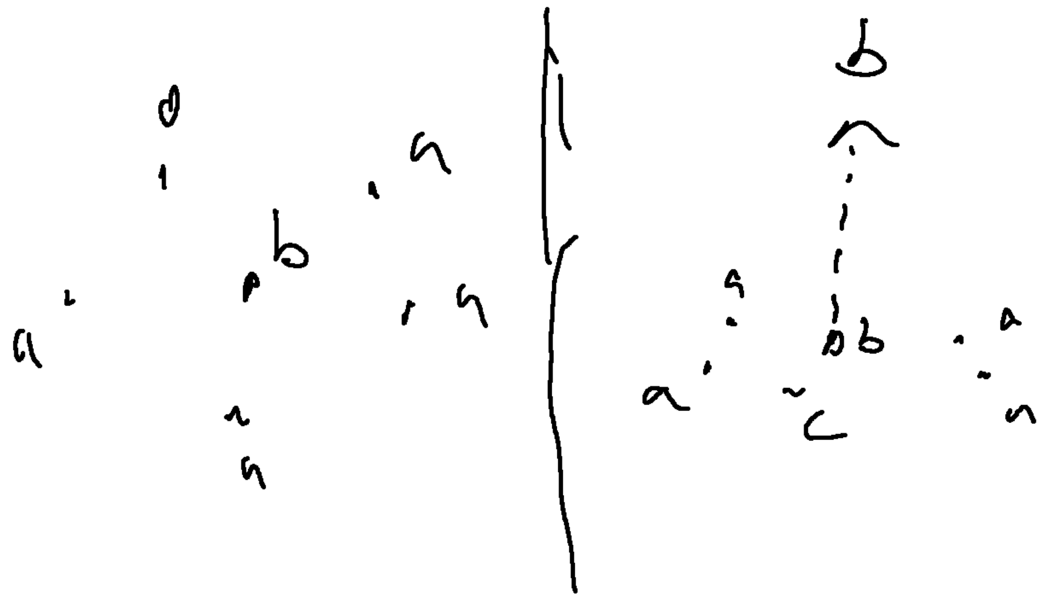
When we consider the two ring problem if we look at a knn interpretation with distance weighting it essentially creates a new dimension for a point and pushes it out on that dimension

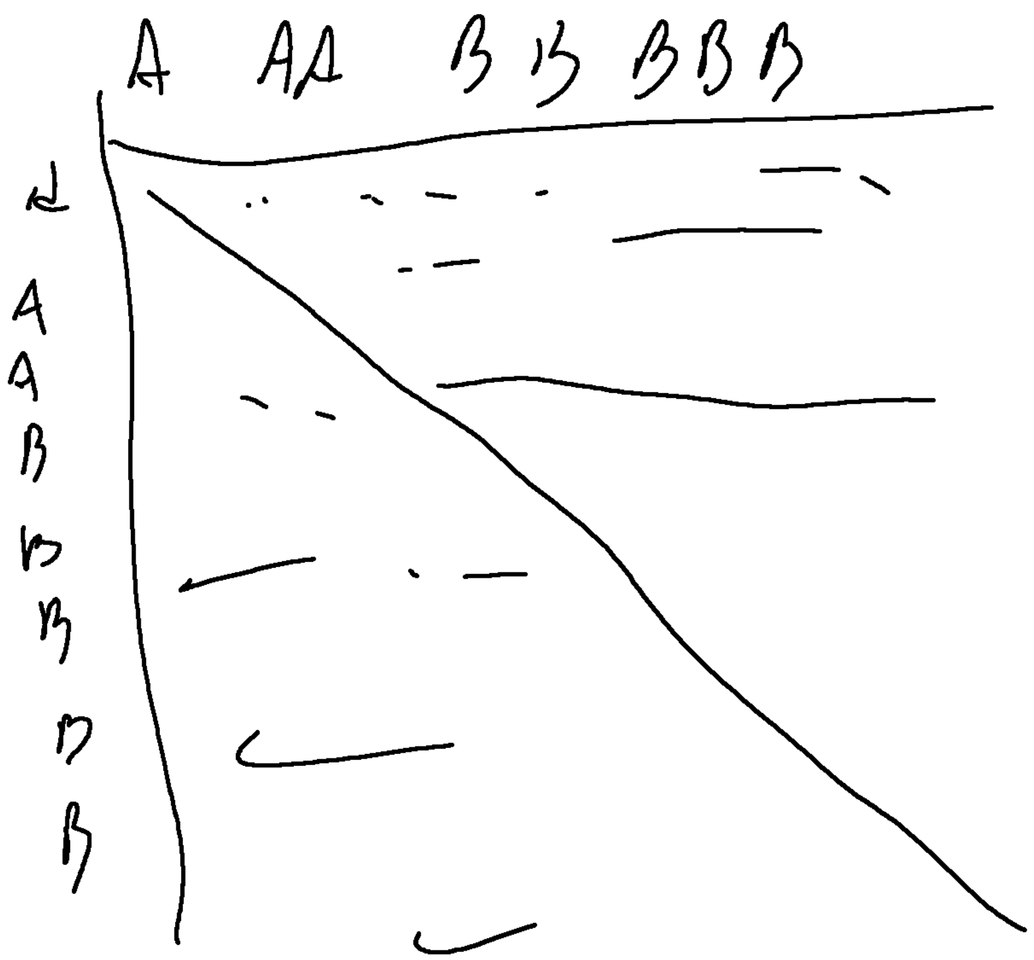


In effect it almost switches off that datapoint, which although can be useful, if a datapoint is overly noisy or non representative, or adversarial in some way. Really what we want to do in the ring problem is move the inner samples closer together, and away from the outer samples.

And herein lies the crux of why layers or sequential transforms are required.

we have to move the central points together to a basin of attraction with a radius that doesn't touch the outer points, then we want to push those points up and away, then we want to pull the ring together. My question for a while now has been what is the interstitial value that we are trying to predict and I think it is the inter class proximity ☺

If we consider the cross corr of distances



we know in the output dimension that the A-classes have zero distance between them and the B-classes have 0 distance between them. Therefore in the input dimension we want to make the distance between A-A and B-B and A-B pairs all equal to their distance in the output dimension. So the trick is coming up with a series of transforms that moves towards that equilibrium. This is probably somewhat analogous to an embedding.

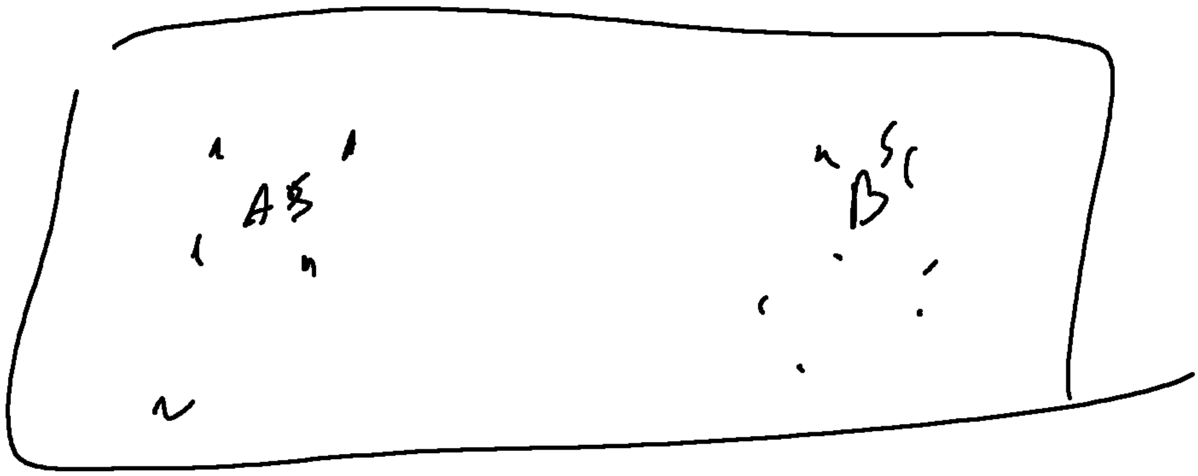
Once we achieve this interstitial goal of the input dimensional distances equalling the output dimension distances, the goal becomes moving the input space to the output space. So if we have 784 input dimensions, and only 10 output dimensions, we may find a transform that satisfies the distances at the output, but in the wrong place, so we may need to migrate points. This may be an ↑ up dimensionality or a ↓ down dimensionally. i.e. compression or rarefaction.

Because we aren't using gradient descent we can invent whatever transforms we want, such as choosing a point in space, choosing a hard radius and a level of attraction, giving us a sphere of influence.

It may take many such transforms to rearrange the space adequately, which really is the premise of a neural network. The problem as well is being too selective and missing the generalization.

Okay so first principle is that given an input and output dimension, whichever has more components is the space you are actually working in. So if you have $784 \to 10$ you are operating in a 784 dimensional space, then as a result just ignoring the components of the other 774 dimensions, kind of clipping them out. Similarly if you are doing $10 \to 784$ you are starting with 784 dimensions but 774 of them just have no value in them.

lets consider space as a fabric and each point is a pin



The regularization is that each pin has to pull the fabric from where dst is, but points also compete to be the correct distance apart in the output dimension. Gravity equation?

So then how do we know where a interstitial point has ended up?

Further we don't have to move to the appropriate dimension, kan will do that for free, we just need to improve the kan fabric.

If we can do a ~~fabric~~ fabric transform why not just move it into the correct dimensional position entirely?