UMAP Video Annotations

Rationale Behind Video Annotation

The mathematical foundations as discussed in the pure research papers for UMAP are difficult to understand, and rely on topological data analysis and category theory. The lecture I annotated is done by a pure mathematician who contributed to the UMAP algorithm.

Video:

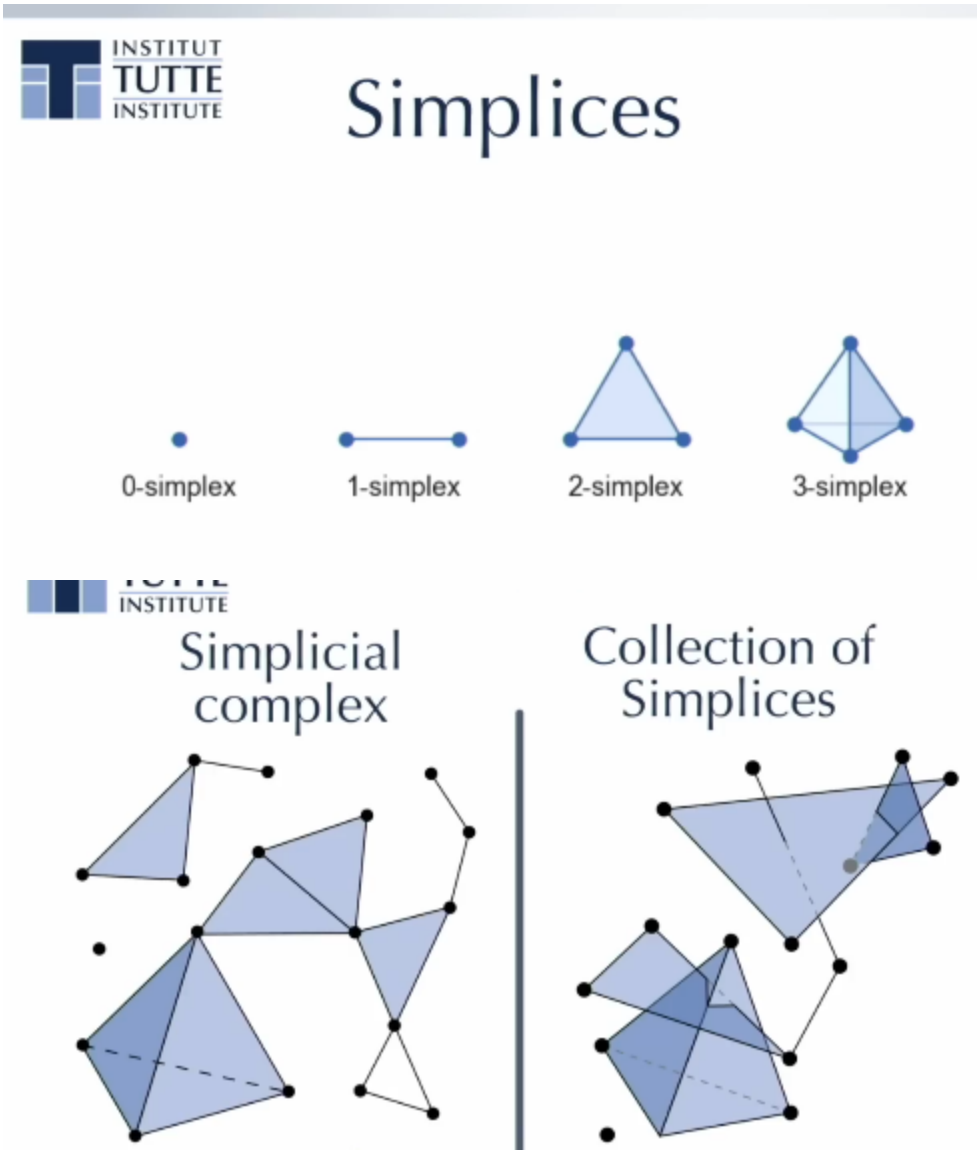
https://www.youtube.com/watch?v=nq6iPZVUxZU&t=395s

Background

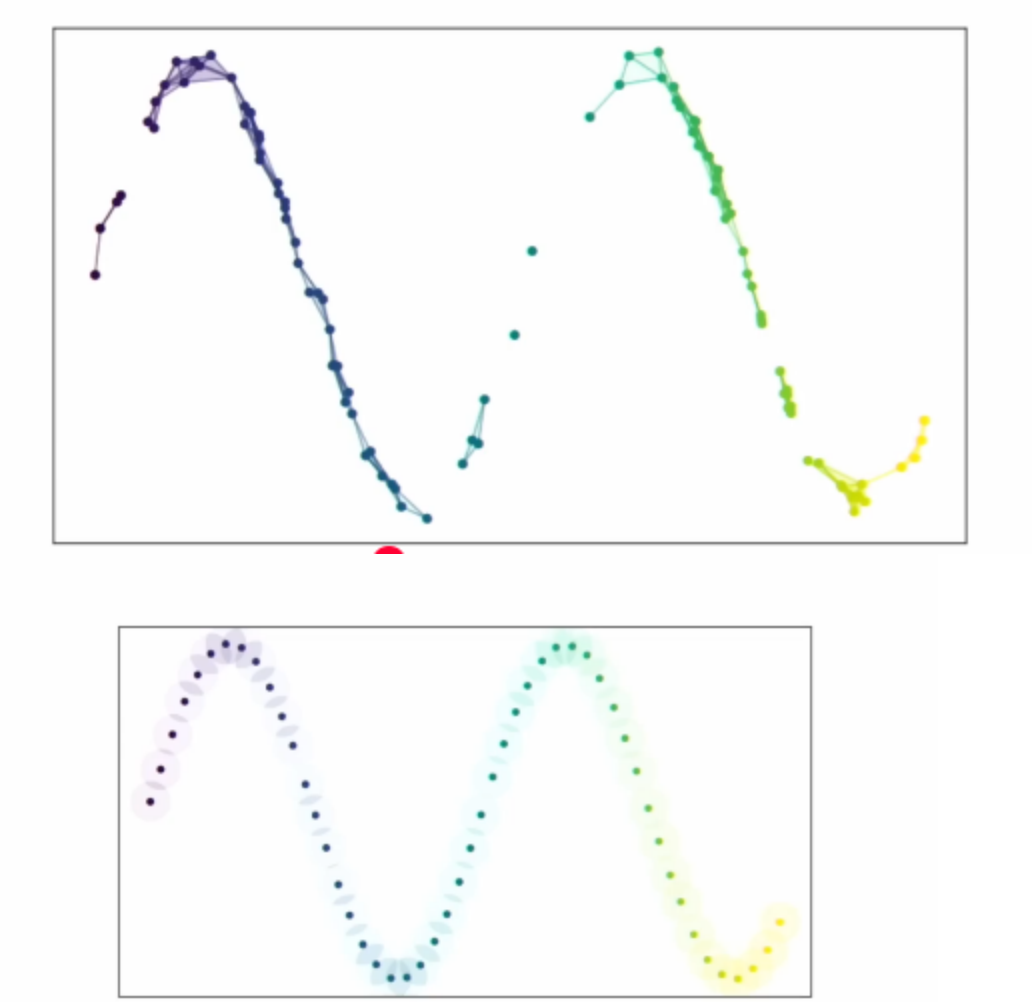
As of 2018, the time of the lecture, t-SNE was the standard since 2008 for dimension reduction. PCA was the ancestor to all of the dimension reduction algorithms used at that point. Some models of dimension reduction are based on nearest neighbors, which UMAP is based on as well.

Topological Data Analysis Simplicial Forms (4:00-7:00)

Topological data analysis refers to the structure of the data, and its general form, even if the data is very high dimensional and noisy. One way that we can represent a topological data space we are working with is by a combination of simplices:



Practically, pretty much topological set can be represented as a simplicial complex, as a region made by stitching together smaller simplicial forms. For instance consider the example with an approximation of $\sin(x)$. We want to construct a simplicial complex out of datapoints that are sampled from $\sin(x)$. We start by assigning at each data point an open ball, and then we take the "nerve" of the cover, which seems to just be simplicial form generated by taking the union of all of the subsets. The topology of the nerve should principally be the same as the topology of our original dataset.



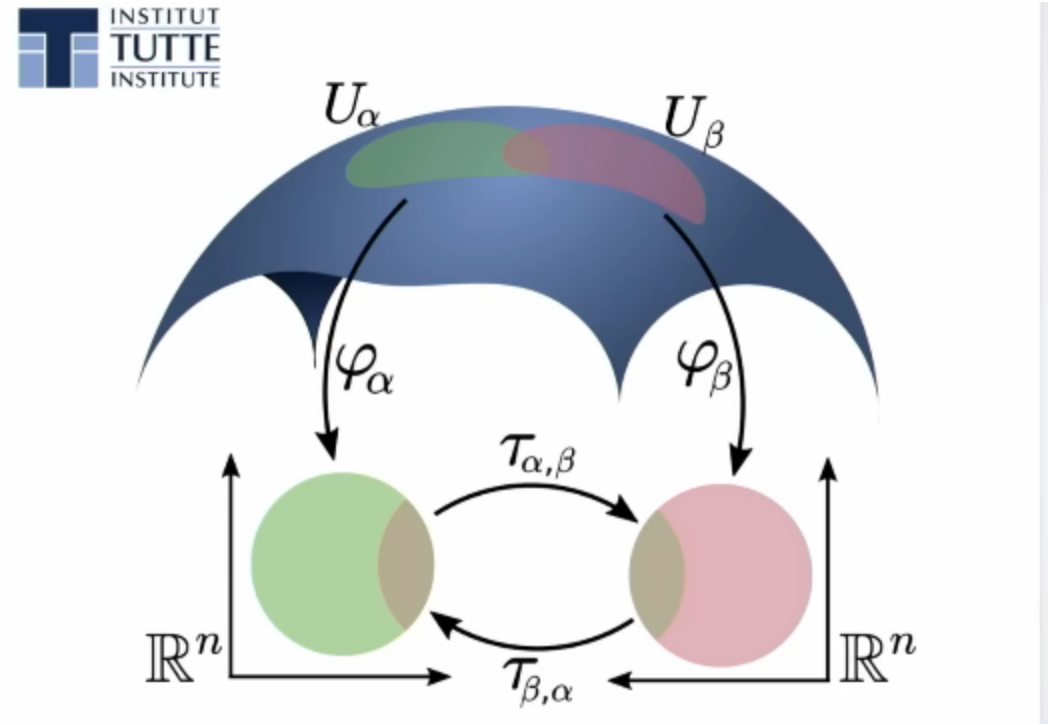
Here we have 2 differing sets that use the "nerve" to construct the topological space using simplices. They differ, even though both describe the same object, by having a different distribution of points. The bottom curve has no gaps and is much nicer since it is built using a uniform distribution, whereas the top is skewed and has gaps.

This is why we call UMAP uniform. Therefore, when using UMAP we want to assume that our data is "uniformly distributed".

Consequences of Assuming Uniform Data Distribution (7:00, eyedballed the timestamp)

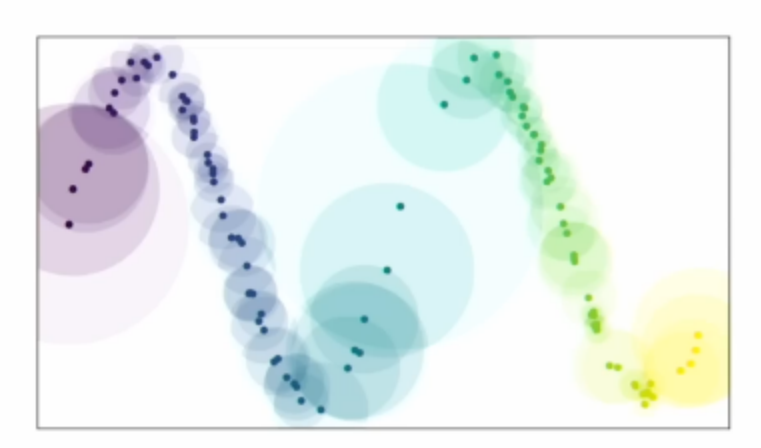
There are consequences from assuming data is uniform about the manifold we are working with. We define a "Riemannian metric" on the manifold in order verify our assumption of

uniformity.



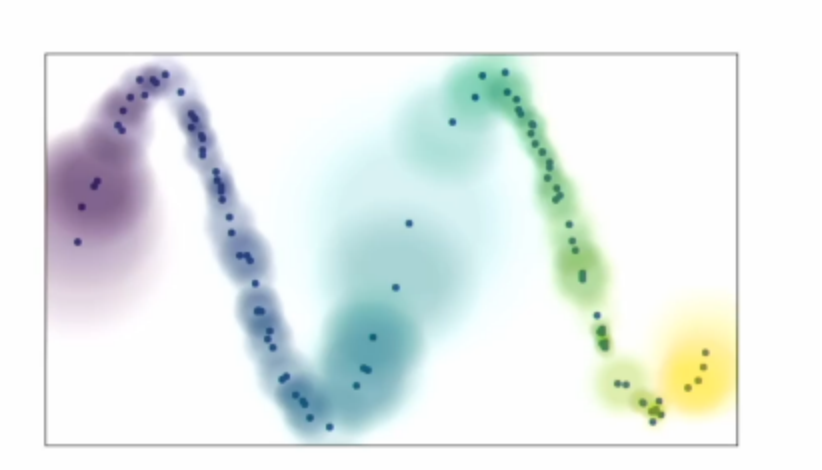
In this graphic, we have some manifold that our data is embedded in. Locally, distance along our metric behaves like distance in the Euclidean metric. But, it may be scaled by some amount when we feed in our information into our respective ϕ_a ,ϕ_b functions. We basically have Euclidean patches everywhere, and we just stitch them together to make our overall manifold. Patches vary in their compactness/density for distance.

For instance, in the sparse cosine curve graph above, we observed that data was not uniformly distributed in the Euclidean space, but when we are constructing our manifold for the data, our manifold tries to make the data uniform. The metric locally for each patch is Euclidean, but overall, using the metric with respect to the manifold the distance between the points there is equivalent.



(9:30) We may not need to use a fixed radius for this (I'm a bit unclear how this works in the video), but the main idea is that the radius sort of fades and away fizzles for each unit ball. This

is sort of like defining some probability around the center of a ball at a point. This is what we call a "fuzzy cover", which uses a lot of category theory/topology to justify so I won't go into it.



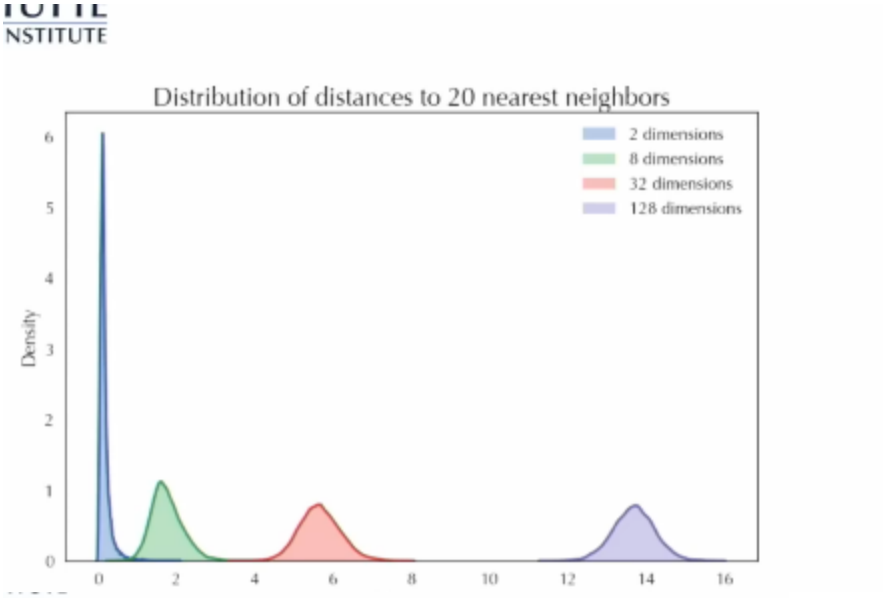
Notice that there's a bit of blur as we get further from a point.

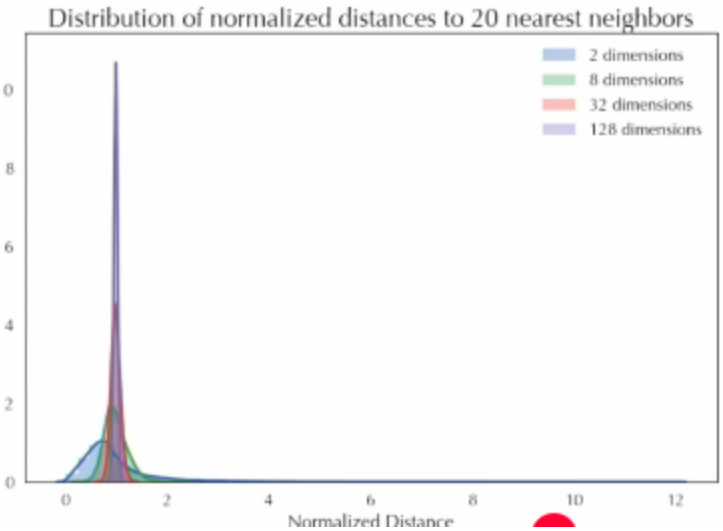
Second Assumption - Locally Connected (10:40)

We do not make the assumption that all parts of our manifold are connected, we instead suppose that there are no regions on our manifold that are isolated. Remember we are using a neighborhood type of model with our structure, and that we connect regions by using simplices. We will not have any point that has no neighbors, it will have at least 1 neighbor, which guarantees that we are locally connected.

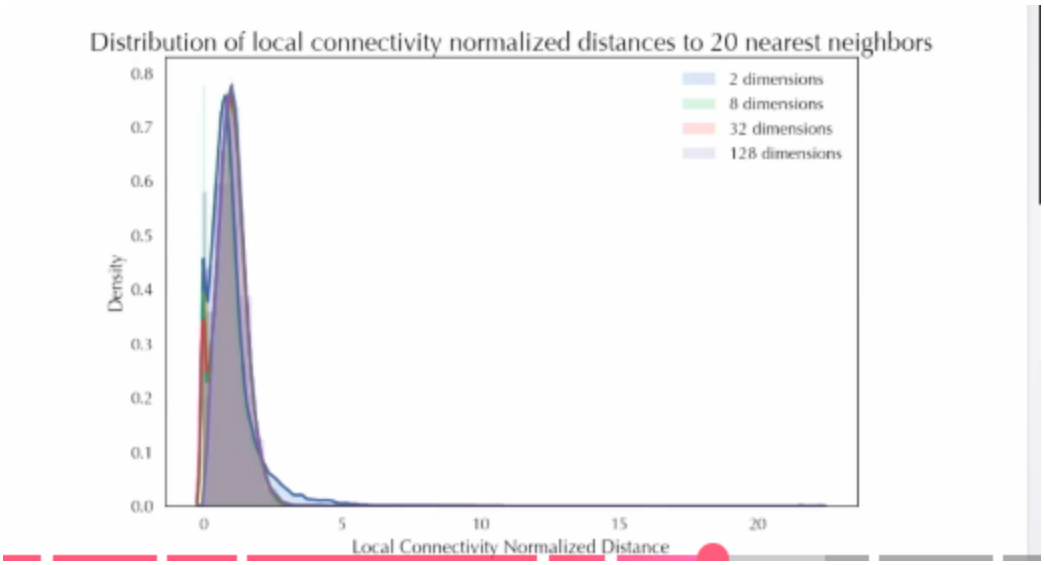
We should be completely confident about the distance from our starting point to our neighbor, and then after that we get a bit fuzzier. Why do we want to do this and why do we care? We can connect this idea with the curse of dimensionality.

For instance, if we take normally distributed data, and increase the dimensions, the overall density keeps getting smaller. Normalizing things doesn't really help, all it does is make the distribution tighter and tighter.



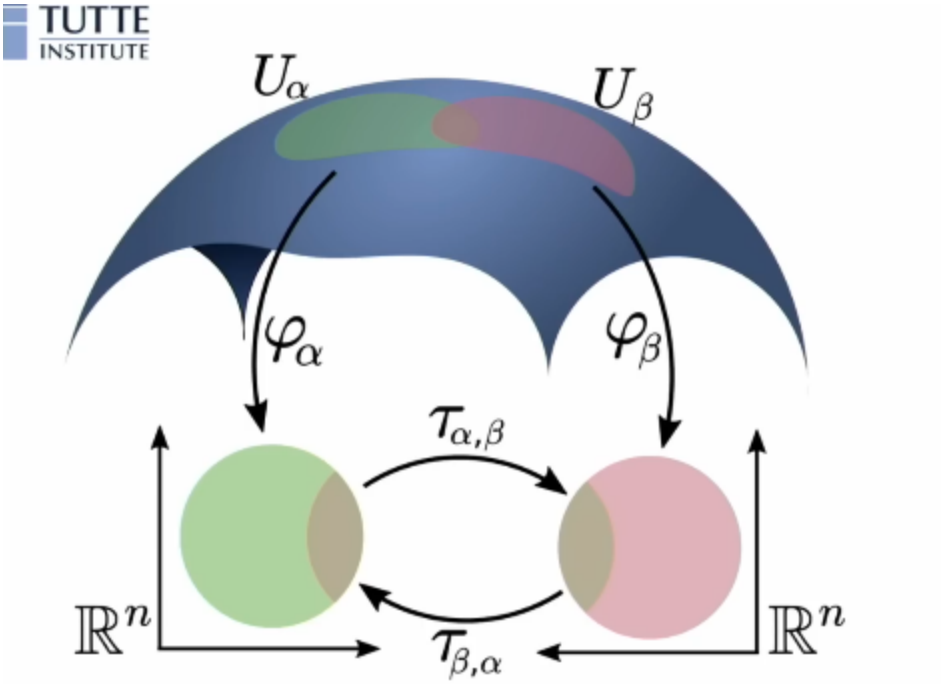


However (somehow) using our connectivity assumption and normalizing gives much nicer distributions that mimic the 2d case:

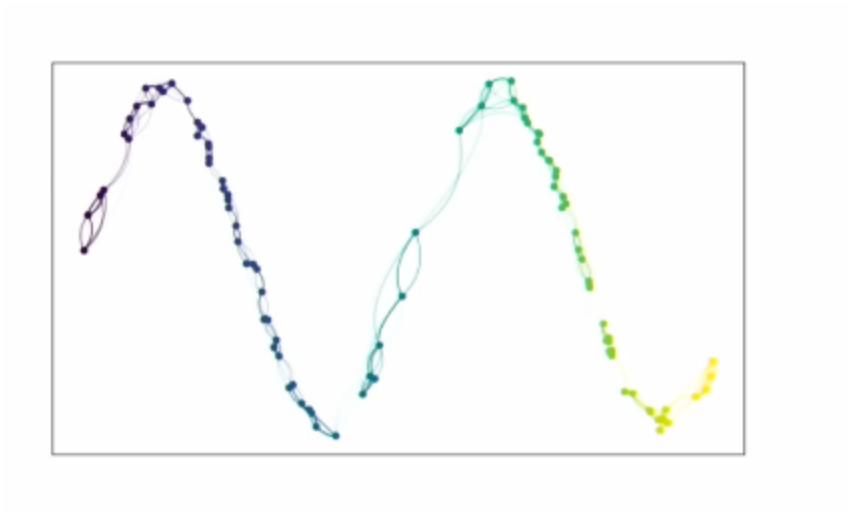


Another Issue With Our Manifold and Local Data, there is no way to convert between the local Euclidean distances (12:30)

With our manifold, the maps given by $T_{\alpha,\beta}$ and $T_{\beta,\alpha}$ are not yet defined because they come from finite data (ok but why shouldn't the others' also be undefined?)



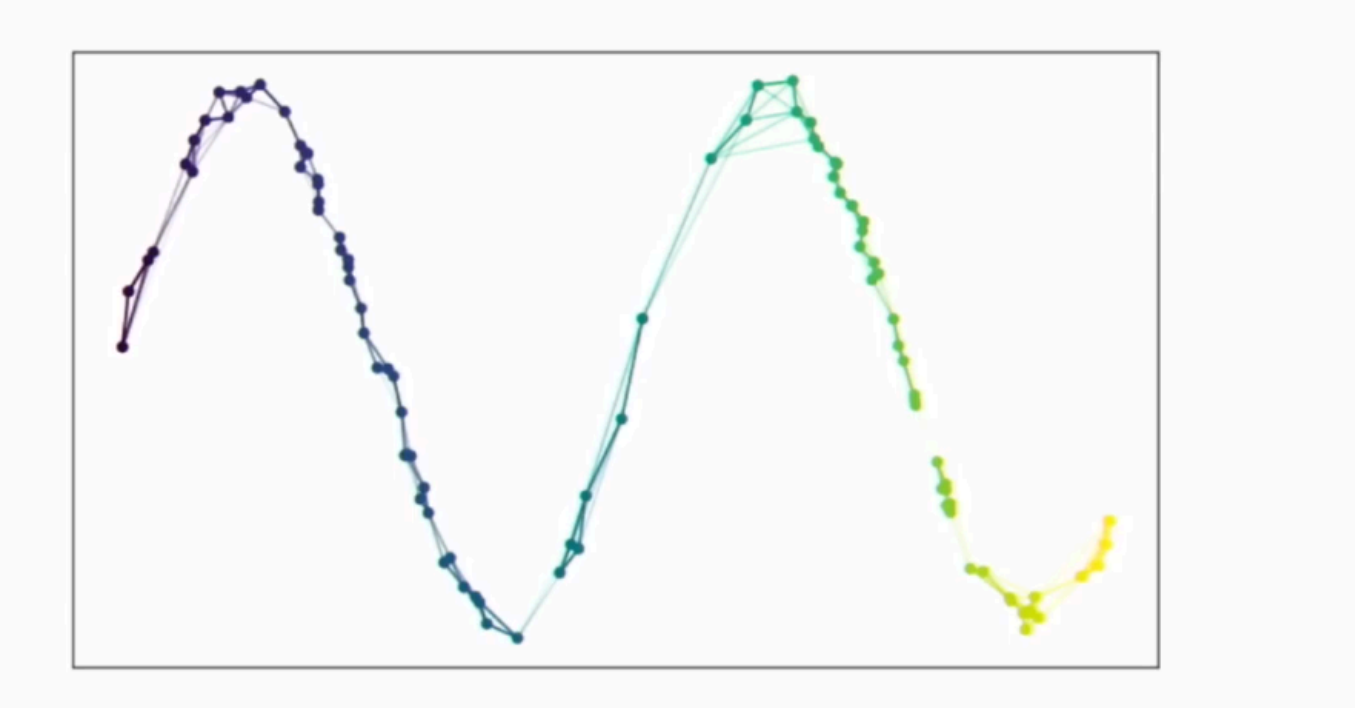
Part of this is because we have conflicting definitions for distance between points on our graph, there are multiple different edges between points, and we need to be able to determine which is the actual distance for coming and going (13:10, I assume by this he means that this is a directed graph, which means that we have potentially different weights for the path AB vs BA)



We have a way of taking a "fuzzy union" (whatever that is meant to mean), and a consistent way of combining the weights on the edges given by a formula. Recall in the graphic for the manifold coming down to Euclidean space is given by:

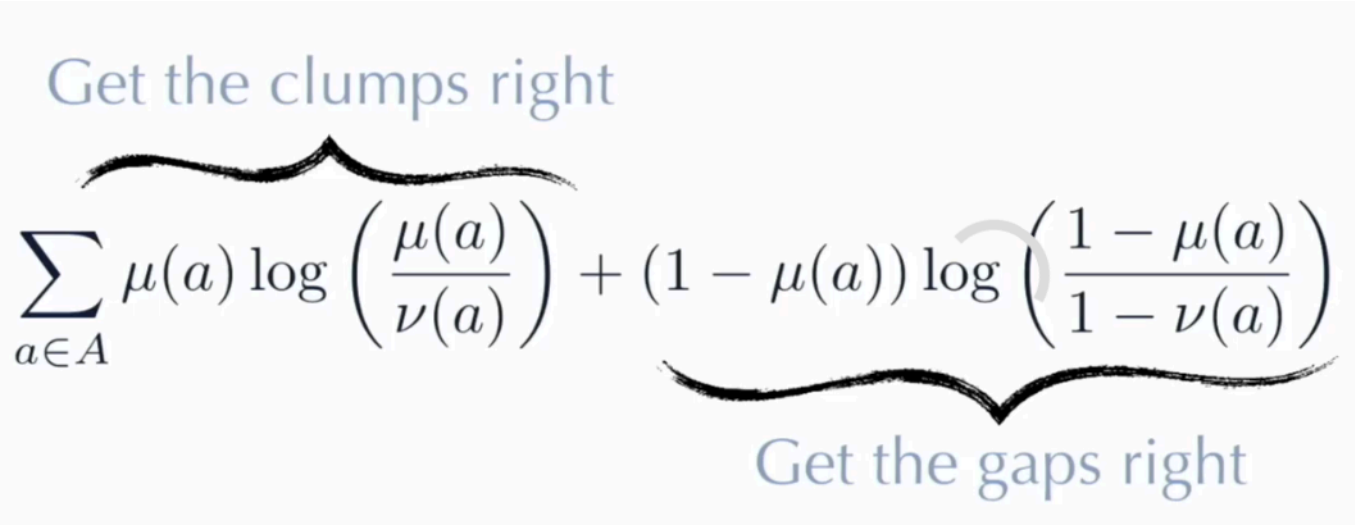
$$f(\alpha, \beta) = \alpha + \beta - \alpha \cdot \beta$$

when, "under a probabilistic fuzzy union, the combination of weights on edges is given by the above". We are essentially just combining the weights oft he edges between 2 points. If the weight on an edge is the probability that it exists, then we can consider the probability of a single edge between them the probability that either α exists or β exists, and then we take away the overlap, this looks just like the law of adding probabilities. The resulting graph is given by the following after doing the edge reweighting.



Measuring Distances Between Low Level Representation of Graph and Normal Graph

For this part of the problem, we are essentially doing some kind of optimization between the low level representation, and the higher level representation, I'm not really sure about the details of this part. But this is also where a lot of points of comparison with t-SNE come into play. We have a "cross entropy" function that we try and optimize over.



Which respectively is an attractive force, and a repulsive force, The attractive force is given by the term on the left, the "get the clumps right", which, piggy backing off of t-SNE is the Kullback Leiberman metric (a metric thats used to determine the distance between 2 different probability density functions), and the right term enforces that we have some repulsive force that makes sure that gaps form between the two.

Video Includes some Graphs on the MNIST set we are using + some stuff on a related fashion dataset that we could totally use as well

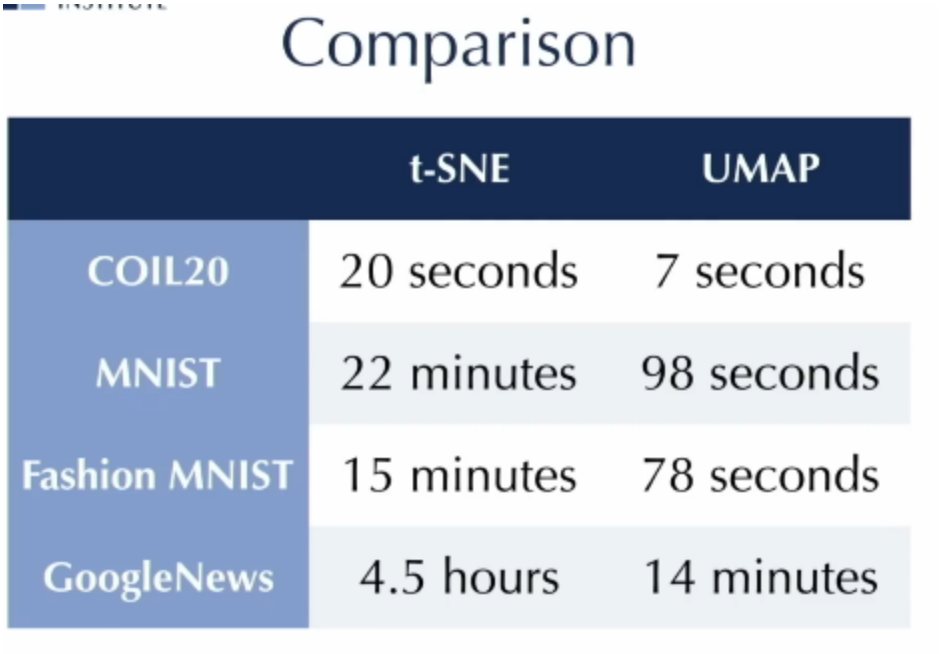
Implementation

We have some hurdles for this algorithm, we need an approximate near neighbors function that works efficiently, and one that is able to work in spite of the fact we are operating on very high dimensional data. We have two options here:

1. Random Projection Trees
2. NN Descent

We "optimize the layout", and use Stochastic Gradient Descent and "negative sampling". Somehow here, we are doing fine computationally.

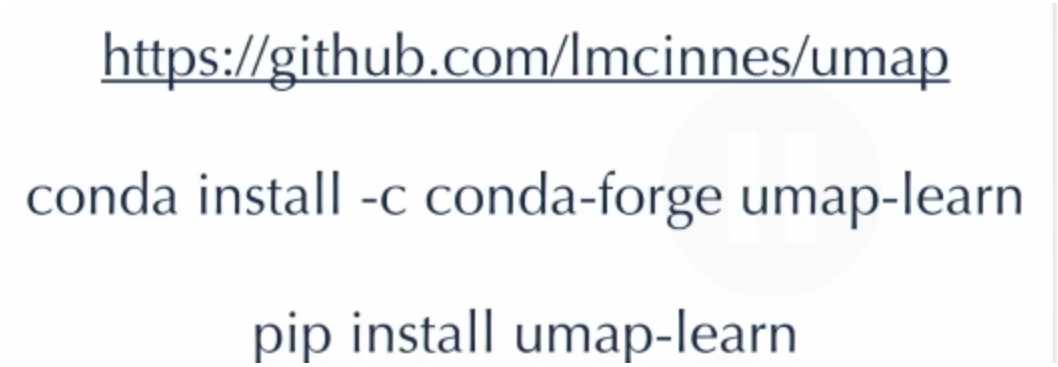
Maybe use NUMBA and python here? Get good high performance. We also get some custom distance metrics as well.



Multicore t-SNE vs UMAP
as dataset scales distance gets significantly larger

Handling New Points (tbh idk whats going on here that well i just added interesting things, i'll revisit when i understand the material better)

- We have a way of doing supervised dimension reduction as well while also maintaining internal structure/global structure?



If we want to compare our implementation vs theirs