THE MEANING OF UMAP: TALK SCRIPT

NICHOLAS LINES

1. Script

(1) Hi there, I'm Nick Lines. Thanks for joining me today as we discuss the meaning of the algorithm known as Uniform Manifold Approximation and Projection, or UMAP. Throughout this talk I'll cite a handful of references, which will be listed at the end so you can look them up. More material is available at my GitHub repo for this talk, which is linked below and through the QR code on the far left.

(2) UMAP is in many ways the new industry standard for non-linear dimensionality reduction. It's particularly useful for projecting high-dimensional data down into two or three dimensions for visualization. If you've used t-SNE before, you'll see some distinct similarities. UMAP is in every way an upgrade from t-SNE, and best of all it comes with a solid mathematical justification that explains why it works so well in specific situations. UMAP comes from Leland McInnes and John Healy et al, the same guys that brought us the HDBSCAN clustering technique.

(3) This talk is intended as a technical introduction to UMAP, but it is itself at best an oversimplification for the purposes of conceptualization. Also, please note that nothing I'll tell you about here is original, it all comes from the original paper and materials the authors and users have provided for documentation and clarification. If you like what you see here, go read the real authorities to find out how it all really works. I'll start off here by motivating the need for UMAP-style dimensionality reduction, and then give the "meaning" of the algorithm, after which we'll discuss some practical things to keep in mind, and wrap it up.

(4) So what is it that UMAP does for us?

(5) This is a painting many of you may know, it's called *Guernica*, by Pablo Picasso. Guernica is a great example of the cubist school of art, which represents real objects, but in a very different way from traditional realism.

(6) Take for example this horse, which we'll enlarge. Cubists often attempt to show more than a 2D snapshot of a scene, and instead provide all the details of significance that might be observed in a 3D survey of the subjects. In this case, the terrified horse is painted in such a way that we can see both eyes, all of its teeth, it's nose, and both ears, as well as the inside of its mouth. No single 2D "photo" of a horse could capture all of these features, yet Picasso has made them all visible in this piece.

Note that this is not an abstraction, it really does portray the results of the Nazi bombing of Guernica. When people asked Picasso what elements in this piece represented, he famously said that the bull was a bull, the horse

Date: November 29, 2021.

was a horse, and "I make the painting for the painting. I paint objects for what they are." Whatever your artistic tastes, we can all appreciate as data scientists that cubism manages to pack into two dimensions much of the experience of a 3D visualization.

(7) In Mathematics this is called projection. Dimensionality reduction of data often takes on this role, projecting high-dimensional data into a lower-dimensional space for viewing, clustering, etc. Here's some real-world examples of prehistorical animal 3D models projected into two dimensions by UMAP. Like the Cubists approach, these resulting images are NOT simply snapshots of the 3D model (i.e. linear projections), but instead show us ALL important components of the animals. Local structures are highlighted (e.g. the legs and tusks of the mammoth) but also kept appropriately distant from each other.

(8) So now that you've seen it done, let's talk about the steps it takes to get to a projection with that kind of integrity.

(9) You can check out the many complicated taxonomies that have been created and shared across the machine learning community trying to organize the many varieties of dimensionality reduction techniques, but for our purposes I'm going to stick with Leland McInnes's standby and simply say that these approaches fall into two broad categories, Matrix Factorizations, and Neighborhood-Graph techniques.

(10) Matrix Factorization techniques include Principle Component Analysis, or Non-negative Matrix Factorization. These all represent the data's high-dimensional structure with a matrix showing relationships between data points, and then through linear algebra generate a lower-dimensioned matrix representation of the data's structure.

In contrast, Neighborhood Graph-building techniques like t-SNE and UMAP start by creating a graphical representation of the data's structure, and then optimizing a lower-dimensioned graph to preserve local neighborhood structures.

(11) In t-SNE we do this by computing the pairwise (probably euclidean) distances between data points, then from these we form Gaussian probability distributions representing the similarity between each two points. Then in the lower dimensional projection space we create Students t-distributions representing the similarity between each point pair, and optimize these by minimizing the Kullback-Leibler divergence between the two distributions. The Students t-distributions that give t-SNE its "t" are chosen because they have heavier tails, so we can put less emphasis on preserving the most distant structures in the neighborhood graph.

(12) So t-SNE basically consists of two phases. You find some representation of the topological structure of your data in it's original space, and then you generate an optimal lower-dimensional projection of that structure.

(13) There is one parameter fed into t-SNE, and that is perplexity. Perplexity is an expression of entropy in the data, and can be thought of as a continuous analog proportionate to the number k in k-nearest-neighbors. If you know your data is more densely packed, you will want to choose a larger perplexity, thereby accepting more neighbors in the local structures to be preserved.

(14) So how does UMAP compare? Well, here's an overview. You can also divide UMAP into the same two phases, one that learns a representation of the data's topology, and one that optimizes a lower dimensional representation of that graph.

UMAP begins with the U, justifying a uniform representation of the data. Then it represents that data with combinatorial structures, and finds the neighborhood distances. It then initializes a spectral embedding in a lower dimension, and preserves the integrity of the original distance structure using a force-directed algorithm.

(15) UMAP takes in two parameters, which act as knobs allowing you to focus more on local structure integrity or more on global structure integrity. The first parameter is a K-Neighbors value, which is comparable to the perplexity parameter in t-SNE. Again, lower values mean you are interested in smaller neighborhood structures. The second parameter is the minimum distance between points in the lower-dimensional embedding space. This controls the density of the output embedding. Lower values will produce tighter structural representations in the projected graph.

From just what we've shown here, you can probably guess that UMAP has the potential to do a better job balancing global vs. local structure preservation in the dimensionality reduction, compared to t–SNE.

(16) Now let's look at the novel components of the UMAP algorithm in slightly more detail. We'll start with the uniformity assumption. Let's think about this toy data here, where each point x_i lives in the two-dimensional manifold shown here. The overall goal of UMAP (building off of earlier work by David Spivak) is to describe the data's topological structure in a way that can be simplified. To do that, we want to create neighborhood balls of a given radius around the points (in math terms, an open cover) and then preserve those neighborhood relationships. But even in the toy data this might not work, since the data is not uniformly distributed. In general, finite data samples in high dimensions never *will* be uniformly distributed. So what now? Well, the UMAP paper solves this problem by creating a distance function specific to each point x_i in the data. Each of these distances is designed so that a unit ball contains the k-nearest neighbors of that point. They then prove that this allows you to treat the data as if it is uniformly distributed across the manifold by using the correct distance functions at each step!

(17) Now let's talk about the topological representation of data employed by UMAP. Here's the building blocks they use. A simplex is just a triangle generalized to n-dimensions. Each k-simplex is made up of "faces" that are themselves lower-dimensional simplices. We can glue simplices together by their faces to create a simplicial complex. So a simplicial complex is basically a graph with nice combinatorial properties. We can build a simplicial complex that represents our data, called a Čech complex ("chesch"), by making each point a 0-simplex, then drawing edges between each point and the other points in its unit ball neighborhood to create 1-simplices, and so on until we've pulled every data point into a single complex. Our entire motivation for doing this is that now we can use a theorem called

the Nerve Theorem to guarantee that this combinatorial object has stored all important information about the data's topology.

It turns out that the majority of the structure in this representation is a Vietoris-Rips Complex, composed out of only zero and one simplices, which is just a graph, and therefore computationally convenient.

(18) But remember, we said each data point possesses its own custom-made distance function. So how do we connect points in these structures when their distances are asymmetric? The sneaky thing UMAP does here is it steps away from hard connections and switches to fuzzy connections by using the distance functions to create "probabilities of connections." That means that each two points have a probability of being connected given by their mutual distances. This data gets stored as a weighted complex (basically a weighted graph). Note that we force each point to be connected to its nearest neighbor with probability 1.

(19) Let's visualize these steps with the toy data I showed before.

(20) Setting the k-neighbors parameter to 2, we get these solid and fuzzy balls at each point. Note that the radius of each data point's neighborhood ball is entirely dependent on the local structure of the data.

(21) If we crank the k-neighbors parameter up to 5, we get much wider neighborhoods.

(22) And bumping the parameter all the way up to 11 gives us even broader neighborhoods. This open cover, the Čech complex, is still doing a good job of highlighting the global structure of the data, at the expense of ignore some of the local structure.

(23) The second phase of the UMAP algorithm is much less novel. Suppose we initialize an embedding in the lower-dimensional embedding space. It's going to include the same points, so the 0-simplices are the same in both simplicial complexes. The real differences of interest are in the graph edges, the 1-simplices. Since we represented these with probabilities, we can compare the two embeddings by representing the connection between each pair of points as a Bernoulli random variable drawn from a distribution whose parameter is the edge probability in that embedding.

Practically, for each edge $e \in E$, the set of all possible edges, we'll say $w_h(e)$ is the weight or probability of the edge e in the original embedding and $w_l(e)$ its analog in the new lower-dimensional space. The cross-entropy is

$$\sum_{e \in E} w_h\left(e\right) \log\left(\frac{w_h\left(e\right)}{w_l\left(e\right)}\right) + \left(1 - w_h\left(e\right)\right) \log\left(\frac{1 - w_h\left(e\right)}{1 - w_l\left(e\right)}\right)$$

where the first term represents an attractive force along e if the original probability of connection was large, and the second term is a repulsive force along e if the original probability of connection was small. By minimizing the cross-entropy, we will reach an optimal low-dimensional representation of the data.

(24) That wraps up our discussion about the theory behind UMAP. But before we finish I want to say a few words about using UMAP in practice.

(25) If you are trying UMAP for the first time, your instincts from using t-SNE may work against you in a few ways. The first thing to keep in mind is your choice of K-neighbors and minimum distance can significantly affect

the time it takes the algorithm to run. If you have a huge number of data points, you may want to spend some time thinking about sensible parameter choices before diving in.

It can also be tempting to read more into UMAP plots than is really there. If you are using UMAP to visualize data, it might be a good exercise to generate a few different plots so you can get used to identifying which parts of the plots are stochastic. The orientation and sizes of the clusters are essentially meaningless. The distances between clusters, while more meaningful than in t-SNE plots, are not always easy to interpret.

If you have used t-SNE before, you probably didn't stress too much about the perplexity parameter. This is because it doesn't make nearly as big an impact in the algorithm's output as the k-neighbors parameter does in the UMAP output. Again, you may need to spend time thinking through parameter choices for big data sets or simply running a bunch of UMAP projections with different parameter choices for small data sets. Since UMAP is much faster than t-SNE, this is practical.

Lastly, anytime you get a new tool like UMAP it's tempting to use it everywhere. I've seen a handful of cases where a scientist used UMAP in situations where they are projecting a HUGE amount of data in a high dimension to another slightly lower dimension. In these cases, UMAP is going to be slow compared to simpler factorization techniques like NMF, and may not perform as well. Don't get me wrong, it is an amazing tool and my favorite dimensionality reduction algorithm, but it's not a panacea.

(26) Now it's time to sum up.

(27) We've seen UMAP rocks. It provides a fast neighborhood-graph approach for dimensionality reduction that comes with theoretical guarantees to preserve the structure of your data, and it's now the tool of choice for visualizing high-dimensional data.

(28) Thank you for listening! On the left is a list of references for further reading if you are interested. If you have questions please reach out to me by email or through the GitHub project for this talk. Happy UMAPping!

References

[1] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[2] L. McInnes and J. Healy, "How umap works," *How UMAP Works - umap 0.5 documentation*. [Online]. Available: https://umap-learn.readthedocs.io/en/latest/how_umap_works.html

[3] D. Angelov, "Top2vec: Distributed representations of topics," 2020.

[4] Wikipedia contributors, "Guernica (picasso) — Wikipedia, the free encyclopedia," 2021, [Online; accessed 17-November-2021]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Guernica_(Picasso)\&oldid=1054068902

[5] D. I. Spivak, "Metric realization of fuzzy simplicial sets," *Self published notes*, 2012.

[6] M. Noichl, "Examples for umap reduction using 3d models of prehistoric animals," *GitHub repository*, 2019. [Online]. Available: https://github.com/MNoichl/UMAP-examples-mammoth\-

[7] A. Coenen and A. Pearce, "Understanding umap," *Understanding UMAP*. [Online]. Available: https://pair-code.github.io/understanding-umap/

E-mail address: nicholasalines@gmail.com