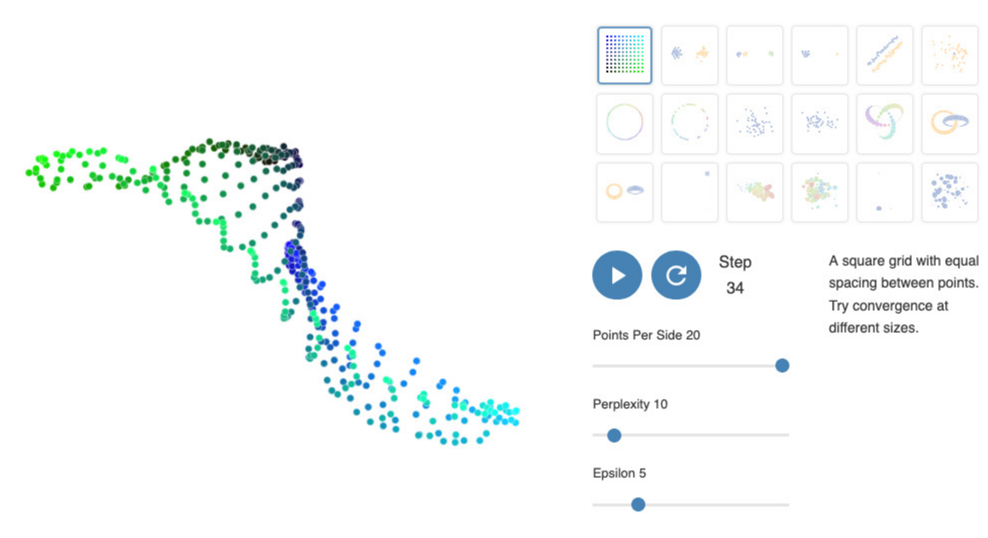
How to Use t-SNE Effectively

Although extremely useful for visualizing high-dimensional data, t-SNE plots can sometimes be mysterious or misleading. By exploring how it behaves in simple cases, we can learn to use it more effectively.



MARTIN WATTENBERG
Google Brain
FERNANDA VIÉGAS
Google Brain
IAN JOHNSON
Google Cloud
Oct. 13
2016
Citation:
Wattenberg, et al., 2016

A popular method for exploring high-dimensional data is something called t-SNE, introduced by van der Maaten and Hinton in 2008 [1]. The technique has become widespread in the field of machine learning, since it has an almost magical ability to create compelling two-dimensonal "maps" from data with hundreds or even thousands of dimensions. Although impressive, these images can be tempting to misread. The purpose of this note is to prevent some common misreadings.

We'll walk through a series of simple examples to illustrate what t-SNE diagrams can and cannot show. The t-SNE technique really is useful—but only if you know how to interpret it.

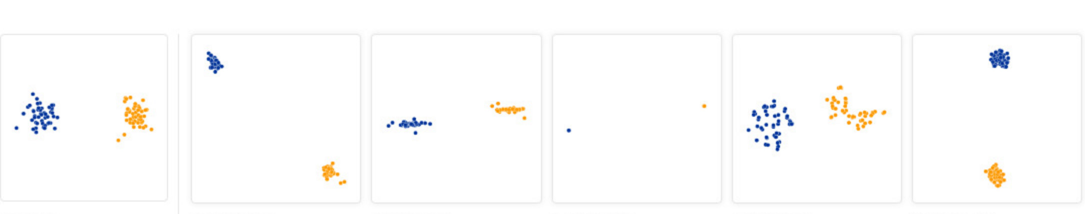
Before diving in: if you haven't encountered t-SNE before, here's what you need to know about the math behind it. The goal is to take a set of points in a high-dimensional space and find a faithful representation of those points in a lower-dimensional space, typically the 2D plane. The algorithm is non-linear and adapts to the underlying data, performing different transformations on different regions. Those differences can be a major source of confusion.

A second feature of t-SNE is a tuneable parameter, "perplexity," which says (loosely) how to balance attention between local and global aspects of your data. The parameter is, in a sense, a guess about the number of close neighbors each point has. The perplexity value has a complex effect on the resulting pictures. The original paper says, *"The performance of SNE is fairly robust to changes in the perplexity, and typical values are between 5 and 50."* But the story is more nuanced than that. Getting the most from t-SNE may mean analyzing multiple plots with different perplexities.

That's not the end of the complications. The t-SNE algorithm doesn't always produce similar output on successive runs, for example, and there are additional hyperparameters related to the optimization process.

1. Those hyperparameters really matter

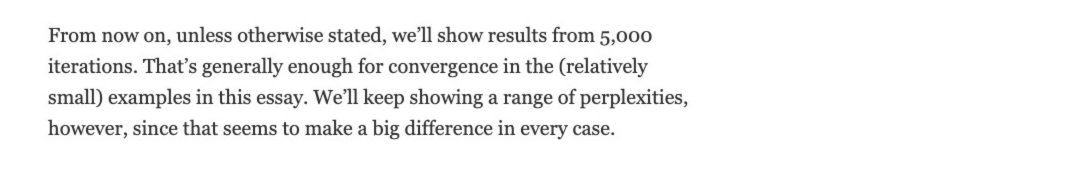
Let's start with the "hello world" of t-SNE: a data set of two widely separated clusters. To make things as simple as possible, we'll consider clusters in a 2D plane, as shown in the lefthand diagram. (For clarity, the two clusters are color coded.) The diagrams at right show t-SNE plots for five different perplexity values.



Original | Perplexity: 2 Step: 5,000 | Perplexity: 5 Step: 5,000 | Perplexity: 30 Step: 5,000 | Perplexity: 50 Step: 5,000 | Perplexity: 100 Step: 5,000

With perplexity values in the range (5 - 50) suggested by van der Maaten & Hinton, the diagrams do show these clusters, although with very different shapes. Outside that range, things get a little weird. With perplexity 2, local variations dominate. The image for perplexity 100, with merged clusters, illustrates a pitfall: for the algorithm to operate properly, the perplexity really should be smaller than the number of points. Implementations can give unexpected behavior otherwise.

Each of the plots above was made with 5,000 iterations with a learning rate (often called "epsilon") of 10, and had reached a point of stability by step 5,000. How much of a difference do those values make? In our experience, the most important thing is to iterate until reaching a stable configuration.



Original | Perplexity: 30 Step: 10 | Perplexity: 30 Step: 20 | Perplexity: 30 Step: 60 | Perplexity: 30 Step: 120 | Perplexity: 30 Step: 1,000

The images above show five different runs at perplexity 30. The first four were stopped before stability. After 10, 20, 60, and 120 steps you can see layouts with seeming 1-dimensional and even pointlike images of the clusters. If you see a t-SNE plot with strange "pinched" shapes, chances are the process was stopped too early. Unfortunately, there's no fixed number of steps that yields a stable result. Different data sets can require different numbers of iterations to converge.

Another natural question is whether different runs with the same hyperparameters produce the same results. In this simple two-cluster example, and most of the others we discuss, multiple runs give the same global shape. Certain data sets, however, yield markedly different diagrams on different runs; we'll give an example of one of these later.

From now on, unless otherwise stated, we'll show results from 5,000 iterations. That's generally enough for convergence in the (relatively small) examples in this essay. We'll keep showing a range of perplexities, however, since that seems to make a big difference in every case.

2. Cluster sizes in a t-SNE plot mean nothing

So far, so good. But what if the two clusters have different standard deviations, and so different sizes? (By size we mean bounding box measurements, not number of points.) Below are t-SNE plots for a mixture of Gaussians in plane, where one is 10 times as dispersed as the other.

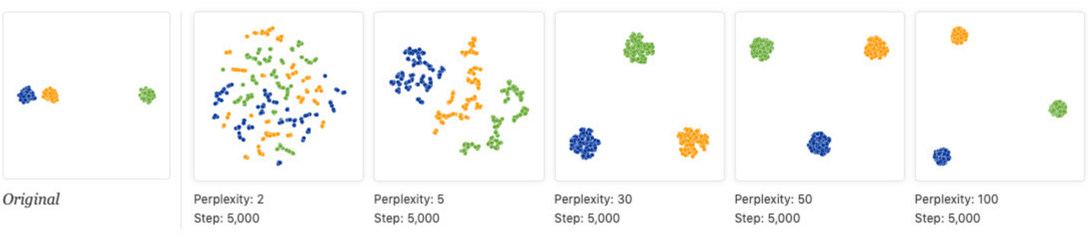
Original | Perplexity: 2 Step: 5,000 | Perplexity: 5 Step: 5,000 | Perplexity: 30 Step: 5,000 | Perplexity: 50 Step: 5,000 | Perplexity: 100 Step: 5,000

Surprisingly, the two clusters look about same size in the t-SNE plots. What's going on? The t-SNE algorithm adapts its notion of "distance" to regional density variations in the data set. As a result, it naturally expands dense clusters, and contracts sparse ones, evening out cluster sizes. To be clear, this is a different effect than the run-of-the-mill fact that any dimensionality reduction technique will distort distances. (After all, in this example all data was two-dimensional to begin with.) Rather, density equalization happens by design and is a predictable feature of t-SNE.

The bottom line, however, is that you cannot see relative sizes of clusters in a t-SNE plot.

3. Distances between clusters might not mean anything

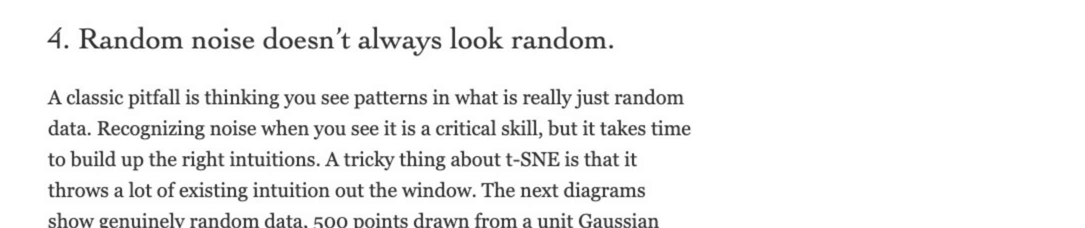
What about distances *between* clusters? The next diagrams show three Gaussians of 50 points each, one pair being 5 times as far apart as another pair.



Original | Perplexity: 2 Step: 5,000 | Perplexity: 5 Step: 5,000 | Perplexity: 30 Step: 5,000 | Perplexity: 50 Step: 5,000 | Perplexity: 100 Step: 5,000

At perplexity 50, the diagram gives a good sense of the global geometry. For lower perplexity values the clusters look equidistant. When the perplexity is 100, we see the global geometry fine, but one of the cluster appears, falsely, much smaller than the others. Since perplexity 50 gave us a good picture in this example, can we always set perplexity to 50 if we want to see global geometry?

Sadly, no. If we add more points to each cluster, the perplexity has to increase to compensate. Here are the t-SNE diagrams for three Gaussian clusters with 200 points each, instead of 50. Now none of the trial perplexity values gives a good result.



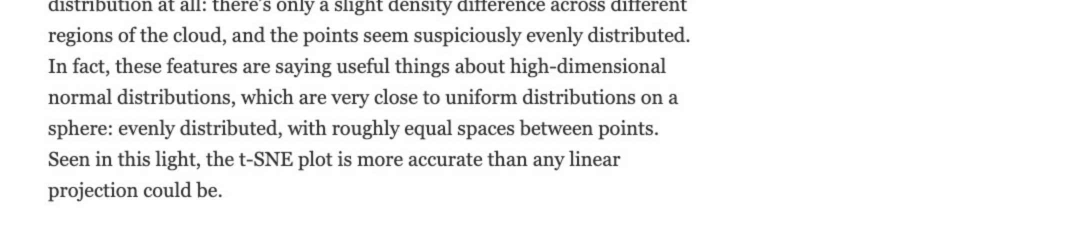
Original | Perplexity: 2 Step: 5,000 | Perplexity: 5 Step: 5,000 | Perplexity: 30 Step: 5,000 | Perplexity: 50 Step: 5,000 | Perplexity: 100 Step: 5,000

It's bad news that seeing global geometry requires fine-tuning perplexity. Real-world data would probably have multiple clusters with different numbers of elements. There may not be one perplexity value that will capture distances across all clusters—and sadly perplexity is a global parameter. Fixing this problem might be an interesting area for future research.

The basic message is that distances between well-separated clusters in a t-SNE plot may mean nothing.

4. Random noise doesn't always look random.

A classic pitfall is thinking you see patterns in what is really just random data. Recognizing noise when you see it is a critical skill, but it takes time to build up the right intuitions. A tricky thing about t-SNE is that it throws a lot of existing intuition out the window. The next diagrams show genuinely random data, 500 points drawn from a unit Gaussian distribution in 100 dimensions. The left image is a projection onto the first two coordinates.



Original | Perplexity: 2 Step: 5,000 | Perplexity: 5 Step: 5,000 | Perplexity: 30 Step: 5,000 | Perplexity: 50 Step: 5,000 | Perplexity: 100 Step: 5,000

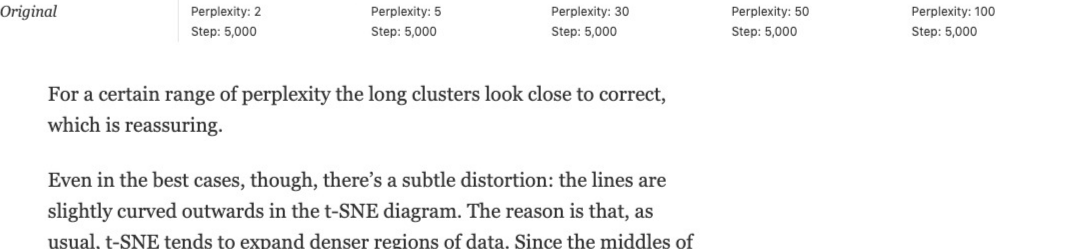
The plot with perplexity 2 seems to show dramatic clusters. If you were tuning perplexity to bring out structure in the data, you might think you'd hit the jackpot.

Of course, since we know the cloud of points was generated randomly, it has no statistically interesting clusters: those "clumps" aren't meaningful. If you look back at previous examples, low perplexity values often lead to this kind of distribution. Recognizing these clumps as random noise is an important part of reading t-SNE plots.

There's something else interesting, though, which may be a win for t-SNE. At first the perplexity 30 plot doesn't look like a Gaussian distribution at all: there's only a slight density difference across different regions of the cloud, and the points seem suspiciously evenly distributed. In fact, these features are saying useful things about high-dimensional normal distributions, which are very close to uniform distributions on a sphere: evenly distributed, with roughly equal spaces between points. Seen in this light, the t-SNE plot is more accurate than any linear projection could be.

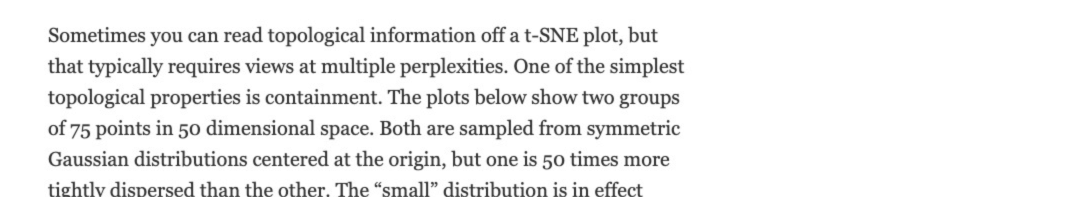
5. You can see some shapes, sometimes

It's rare for data to be distributed in a perfectly symmetric way. Let's take a look at an axis-aligned Gaussian distribution in 50 dimensions, where the standard deviation in coordinate i is 1/i. That is, we're looking at a long-ish ellipsoidal cloud of points.



Original | Perplexity: 2 Step: 5,000 | Perplexity: 5 Step: 5,000 | Perplexity: 30 Step: 5,000 | Perplexity: 50 Step: 5,000 | Perplexity: 100 Step: 5,000

For high enough perplexity values, the elongated shapes are easy to read. On the other hand, at low perplexity, local effects and meaningless "clumping" take center stage. More extreme shapes also come through, but again only at the right perplexity. For example, here are two clusters of 75 points each in 2D, arranged in parallel lines with a bit of noise.



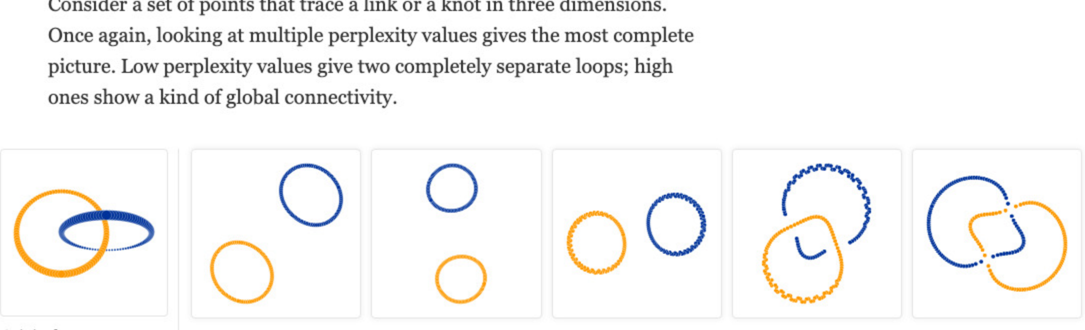
Original | Perplexity: 2 Step: 5,000 | Perplexity: 5 Step: 5,000 | Perplexity: 30 Step: 5,000 | Perplexity: 50 Step: 5,000 | Perplexity: 100 Step: 5,000

For a certain range of perplexity the long clusters look close to correct, which is reassuring.

Even in the best cases, though, there's a subtle distortion: the lines are slightly curved outwards in the t-SNE diagram. The reason is that, as usual, t-SNE tends to expand denser regions of data. Since the middles of the clusters have less empty space around them than the ends, the algorithm magnifies them.

6. For topology, you may need more than one plot

Sometimes you can read topological information off a t-SNE plot, but that typically requires views at multiple perplexities. One of the simplest topological properties is containment. The plots below show two groups of 75 points in 50 dimensional space. Both are sampled from symmetric Gaussian distributions centered at the origin, but one is 50 times more tightly dispersed than the other. The "small" distribution is in effect contained in the large one.

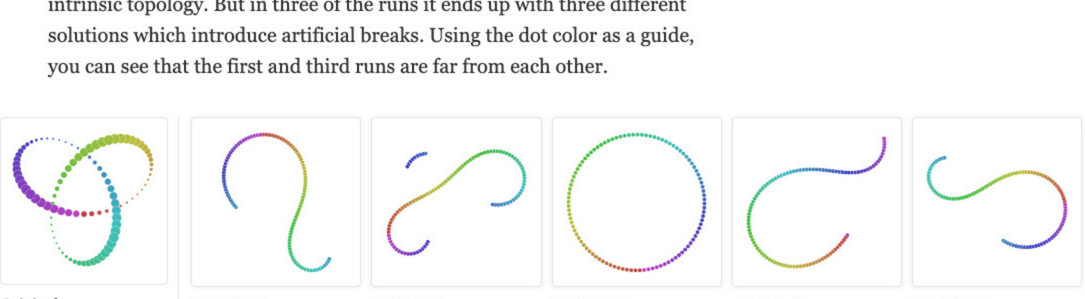


Original | Perplexity: 2 Step: 5,000 | Perplexity: 5 Step: 5,000 | Perplexity: 30 Step: 5,000 | Perplexity: 50 Step: 5,000 | Perplexity: 100 Step: 5,000

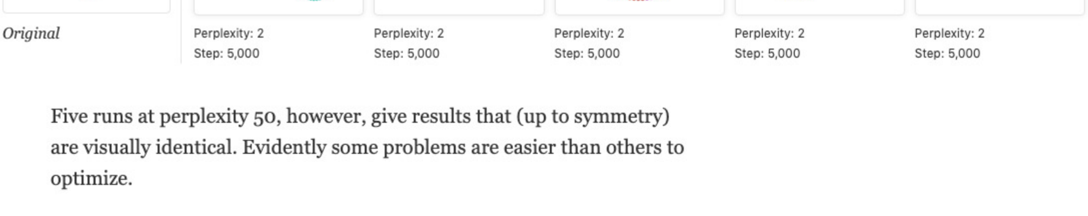
The perplexity 30 view shows the basic topology correctly, but again t-SNE greatly exaggerates the size of the smaller group of points. At perplexity 50, there's a new phenomenon: the outer group becomes a circle, as the plot tries to depict the fact that all its points are about the same distance from the inner group. If you looked at this image alone, it would be easy to misread these outer points as a one-dimensional structure.

What about more complicated types of topology? This may be a subject dearer to mathematicians than to practical data analysts, but interesting low-dimensional structures are occasionally found in the wild.

Consider a set of points that trace a link or a knot in three dimensions. Once again, looking at multiple perplexity values gives the most complete picture. Low perplexity values give two completely separate loops; high ones show a kind of global connectivity.



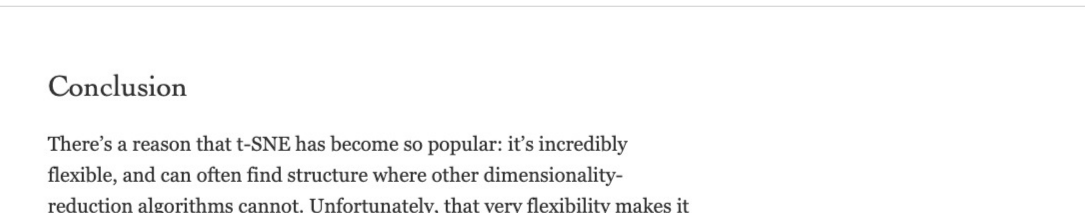
Original | Perplexity: 2 Step: 5,000 | Perplexity: 5 Step: 5,000 | Perplexity: 30 Step: 5,000 | Perplexity: 50 Step: 5,000 | Perplexity: 100 Step: 5,000



Original | Perplexity: 2 Step: 5,000 | Perplexity: 5 Step: 5,000 | Perplexity: 30 Step: 5,000 | Perplexity: 50 Step: 5,000 | Perplexity: 100 Step: 5,000

The trefoil knot is an interesting example of how multiple runs affect the outcome of t-SNE. Below are five runs of the perplexity-2 view.

The algorithm settles twice on a circle, which at least preserves the intrinsic topology. But in three of the runs it ends up with three different solutions which introduce artificial breaks. Using the dot color as a guide, you can see that the first and third runs are far from each other.



Original | Perplexity: 2 Step: 5,000 | Perplexity: 2 Step: 5,000 | Perplexity: 2 Step: 5,000 | Perplexity: 2 Step: 5,000 | Perplexity: 2 Step: 5,000

Five runs at perplexity 50, however, give results that (up to symmetry) are visually identical. Evidently some problems are easier than others to optimize.

Original | Perplexity: 50 Step: 5,000 | Perplexity: 50 Step: 5,000 | Perplexity: 50 Step: 5,000 | Perplexity: 50 Step: 5,000 | Perplexity: 50 Step: 5,000

Conclusion

There's a reason that t-SNE has become so popular: it's incredibly flexible, and can often find structure where other dimensionality-reduction algorithms cannot. Unfortunately, that very flexibility makes it tricky to interpret. Out of sight from the user, the algorithm makes all sorts of adjustments that tidy up its visualizations. Don't let the hidden "magic" scare you away from the whole technique, though. The good news is that by studying how t-SNE behaves in simple cases, it's possible to develop an intuition for what's going on.

Acknowledgments

We are grateful to Chris Olah and Shan Carter for creating this platform, and for excellent design and editorial help from Shan Carter. Daniel Smilkov, James Wexler, and Chi Zeng provided many helpful comments. We also thank Andrej Karpathy for creating the tsnejs library used in the interactive diagrams.

This work was made possible by the support of the Google Brain team.

Edited on Oct. 18, 2016 to describe and correct issues when perplexity is defined to be larger than the number of points. Thanks to Laurens van der Maaten for pointing this out.

References

1. **Visualizing data using t-SNE** [PDF]
Maaten, L.v.d. and Hinton, G., 2008. Journal of Machine Learning Research, Vol 9(Nov), pp. 2579—2605.

Updates and Corrections

View all changes to this article since it was first published. If you see a mistake or want to suggest a change, please create an issue on GitHub.

Citations and Reuse

For attribution in academic contexts, please cite this work as

```
Wattenberg, et al., "How to Use t-SNE Effectively", Distill, 2016. http://doi.org/10.23915/dist
```

BibTeX citation

```
@article{wattenberg2016how,
  author = {Wattenberg, Martin and Viégas, Fernanda and Johnson, Ian},
  title = {How to Use t-SNE Effectively},
  journal = {Distill},
  year = {2016},
  url = {http://distill.pub/2016/misread-tsne},
  doi = {10.23915/distill.00002}
}
```