# Dimension Reduction for Data Visualization Duke Course Notes Cynthia Rudin 

Credits: Principal Components Analysis I.T. Joliffe, notes of Frank Wood (Columbia), and the PaCMAP paper (Wang u. a., 2021).

The first thing that we usually want to do with a new dataset is to "look" at it. But that's not easy in high dimensions. The goal of dimension reduction for data visualization is to take high dimensional data and project it down to 2 or 3 dimensions so that humans can understand its structure. Usually people are interested in seeing whether the data has cluster structures or curving manifolds. I'll cover two techniques for doing this, an old one (PCA), which is from 1901 and a new one (PaCMAP), which is from 2021.

As an aside, dimension reduction techniques are used for other purposes besides dimension reduction. Scientists often use PCA to reduce an extremely high dimensional dataset (thousands of dimensions perhaps) to a much lower dimensional space (say 10 or 100 dimensions) so they can produce meaningful learning results. I will just cover dimension reduction for visualization here. I typically do not use PCA for dimension reduction in my projects because it destroys interpretability for tabular data, and it doesn't make sense to do it for image data. Some people find it useful in applications that are different than the ones I work on.

## 1 Principal Components Analysis

The first technique people think of using for dimension reduction is principal components analysis (Pearson, 1901). You have probably already seen this technique before reading this, but embarrassingly, most ML scientists do not know its derivation.

PCA projects the data onto the several orthogonal vectors that capture the most variation in the data. Those several vectors will be the several eigenvectors of the data's covariance matrix $\hat{\boldsymbol{\Sigma}}=\frac{1}{n-1} \mathbf{X X}^{T}$ that correspond to the largest eigenvalues. It will take me a few steps to show this.

Let us consider a random variable $\mathbf{x}$ with mean $0(\mathbb{E}(\mathbf{x})=\mathbf{0})$. If the variable is 2-dimensional, its covariance matrix would look like:

$$
\boldsymbol{\Sigma}_{2 \times 2}=\left[\begin{array}{cc}
\operatorname{Var}\left(x_{1}\right) & \operatorname{Cov}\left(x_{1}, x_{2}\right) \\
\operatorname{Cov}\left(x_{2}, x_{1}\right) & \operatorname{Var}\left(x_{1}\right)
\end{array}\right] .
$$

In higher dimensions, we have variances on the diagonals and pairwise covariances at other positions.

First Principal Component: First, let us just try to find the direction with maximum variance. (We will see that this is the eigenvector for the largest eigenvalue.) The direction we want is $\boldsymbol{\alpha}^{\prime} \mathbf{x}=\sum_{j} \alpha_{j} x_{j}$ that maximizes $\operatorname{Var}\left(\boldsymbol{\alpha}^{\prime} \mathbf{x}\right)$. For instance, if only the first feature has variance and all the others are essentially constant, then $\alpha_{1}$ should be 1 and the rest should be zero. By definition, the variance is $\mathbb{E}\left[\left(\boldsymbol{\alpha}^{\prime} \mathbf{x}-\mathbb{E}\left(\boldsymbol{\alpha}^{\prime} \mathbf{x}\right)\right)^{2}\right]$ but $\mathbb{E}\left(\boldsymbol{\alpha}^{\prime} \mathbf{x}\right)=0$ since we mean-centered our random variable so that $\mathbb{E}(\mathbf{x})=\mathbf{0}$. So the variance is just $\mathbb{E}\left(\left(\boldsymbol{\alpha}^{\prime} \mathbf{x}\right)^{2}\right)$.

$$
\operatorname{Var}\left(\boldsymbol{\alpha}^{\prime} \mathbf{x}\right)=\mathbb{E}\left(\left(\boldsymbol{\alpha}^{\prime} \mathbf{x}\right)^{2}\right)=\mathbb{E}\left(\boldsymbol{\alpha}^{\prime} \mathbf{x x}^{\prime} \boldsymbol{\alpha}\right)=\boldsymbol{\alpha}^{\prime} \mathbb{E}\left(\mathbf{x x}^{\prime}\right) \boldsymbol{\alpha}=\boldsymbol{\alpha}^{\prime} \boldsymbol{\Sigma} \boldsymbol{\alpha}
$$

Thus we need to find $\boldsymbol{\alpha}$ that maximizes this, subject to $\boldsymbol{\alpha}$ being a unit vector along a direction, meaning that $\boldsymbol{\alpha}$ must have norm 1 , that is, $\boldsymbol{\alpha}^{\prime} \boldsymbol{\alpha}=1$.

Using Lagrange multipliers, we thus want to maximize:

$$
\begin{equation*}
\boldsymbol{\alpha}^{\prime} \boldsymbol{\Sigma} \boldsymbol{\alpha}-\nu\left(\boldsymbol{\alpha}^{\prime} \boldsymbol{\alpha}-1\right) \tag{1}
\end{equation*}
$$

using $-\nu$ as the Lagrange multiplier (I used a negative sign for convenience later on). Differentiating with respect to $\boldsymbol{\alpha}$ and setting the result to 0 yields:

$$
\boldsymbol{\Sigma} \boldsymbol{\alpha}-\nu \boldsymbol{\alpha}=0 \Longrightarrow \boldsymbol{\Sigma} \boldsymbol{\alpha}=\nu \boldsymbol{\alpha}
$$

which is an eigenvalue equation. So we know that $\boldsymbol{\alpha}$ must be an eigenvector and $\nu$ must be its eigenvalue. But which one? Substituting $\boldsymbol{\Sigma} \boldsymbol{\alpha}=\nu \boldsymbol{\alpha}$ into the first term of our objective (1), we find $\boldsymbol{\alpha}^{\prime} \boldsymbol{\Sigma} \boldsymbol{\alpha}=\boldsymbol{\alpha}^{\prime} \nu \boldsymbol{\alpha}=\nu$, which means we want $\nu$ to be as large as possible. So $\nu$ is the largest eigenvalue and $\boldsymbol{\alpha}$ is its eigenvector.

So we now know why the first principal component is $\boldsymbol{\alpha}_{1}^{\prime} \mathbf{x}$ where $\boldsymbol{\alpha}_{1}$ is the eigenvector corresponding to the largest eigenvalue. We will now call them $\boldsymbol{\alpha}_{1}$ and $\nu_{1}$.

Second Principal Component: The second principal component, which will be denoted $\boldsymbol{\alpha}_{2}^{\prime} \mathbf{x}$, needs to maximize the same objective, $\boldsymbol{\alpha}_{2}^{\prime} \boldsymbol{\Sigma} \boldsymbol{\alpha}_{2}$, but is constrained to be orthogonal/uncorrelated with the first principal component $\boldsymbol{\alpha}_{1}^{\prime} \mathbf{x}$. We thus add a constraint as follows:

$$
\begin{align*}
0 & =\operatorname{Cov}\left(\boldsymbol{\alpha}_{1}^{\prime} \mathbf{x}, \boldsymbol{\alpha}_{2}^{\prime} \mathbf{x}\right)=\mathbb{E} \boldsymbol{\alpha}_{1}^{\prime} \mathbf{x x}^{\prime} \boldsymbol{\alpha}_{2}=\boldsymbol{\alpha}_{1}^{\prime} \mathbb{E}\left(\mathbf{x x}^{\prime}\right) \boldsymbol{\alpha}_{2}=\boldsymbol{\alpha}_{1}^{\prime} \boldsymbol{\Sigma} \boldsymbol{\alpha}_{2}=\boldsymbol{\alpha}_{2}^{\prime} \boldsymbol{\Sigma} \boldsymbol{\alpha}_{1}  \tag{2}\\
& =\boldsymbol{\alpha}_{2}^{\prime} \nu_{1} \boldsymbol{\alpha}_{1}=\nu_{1}\left(\boldsymbol{\alpha}_{2}^{\prime} \boldsymbol{\alpha}_{1}\right) . \tag{3}
\end{align*}
$$

Thus, $\boldsymbol{\alpha}_{2}^{\prime} \boldsymbol{\alpha}_{1}=0$ would be sufficient to force the covariance to 0 . Creating a Lagrangian, we want to maximize:

$$
\begin{equation*}
\boldsymbol{\alpha}_{2}^{\prime} \boldsymbol{\Sigma} \boldsymbol{\alpha}_{2}-\nu\left(\boldsymbol{\alpha}_{2}^{\prime} \boldsymbol{\alpha}_{2}-1\right)-\gamma\left(\boldsymbol{\alpha}_{2}^{\prime} \boldsymbol{\alpha}_{1}\right) \tag{4}
\end{equation*}
$$

where $-\nu$ and $-\gamma$ are Lagrange multipliers (again the negation is for convenience later). Differentiating with respect to $\boldsymbol{\alpha}_{2}$ and setting the result to 0 yields:

$$
\begin{equation*}
\boldsymbol{\Sigma} \boldsymbol{\alpha}_{2}-\nu \boldsymbol{\alpha}_{2}-\gamma \boldsymbol{\alpha}_{1}=\mathbf{0} \tag{5}
\end{equation*}
$$

Now here is a trick, where we multiply both sides on the left by $\boldsymbol{\alpha}_{1}^{\prime}$ :

$$
\boldsymbol{\alpha}_{1}^{\prime} \boldsymbol{\Sigma} \boldsymbol{\alpha}_{2}-\nu \boldsymbol{\alpha}_{1}^{\prime} \boldsymbol{\alpha}_{2}-\gamma \boldsymbol{\alpha}_{1}^{\prime} \boldsymbol{\alpha}_{1}=0
$$

We know already that the first term is 0 from (2) and the second term is 0 from (3). The last term has $\boldsymbol{\alpha}_{1}^{\prime} \boldsymbol{\alpha}_{1}$ which is 1 . Thus, we must have $\gamma=0$. Plugging that back into (5), we have:

$$
\boldsymbol{\Sigma} \boldsymbol{\alpha}_{2}-\nu \boldsymbol{\alpha}_{2}=\mathbf{0} \longrightarrow \boldsymbol{\Sigma} \boldsymbol{\alpha}_{2}=\nu \boldsymbol{\alpha}_{2}
$$

which is again an eigenvalue equation, where $\nu$ is an eigenvalue, and $\boldsymbol{\alpha}_{2}$ is its eigenvector. Using the eigenvalue equation,

$$
\boldsymbol{\Sigma} \boldsymbol{\alpha}_{2}=\nu_{2} \boldsymbol{\alpha}_{2} \longrightarrow \nu=\boldsymbol{\alpha}_{2}^{\prime} \Sigma \boldsymbol{\alpha}_{2}
$$

and since the term we aim to maximize is $\boldsymbol{\alpha}_{2}^{\prime} \Sigma \boldsymbol{\alpha}_{2}$, we again want to have $\nu$ as large as possible. Since it needs to be an eigenvalue, and since it can't be the largest one (or else $\boldsymbol{\alpha}_{2}=\boldsymbol{\alpha}_{1}$ which violates our constraint $\boldsymbol{\alpha}_{1}^{\prime} \boldsymbol{\alpha}_{2}=0$ ), it must be the second largest one. Thus, $\boldsymbol{\alpha}_{2}$ must be the eigenvector corresponding to the second largest eigenvalue. The second principal component (PC) is thus $\boldsymbol{\alpha}_{2}^{\prime} \mathbf{x}$.

Remaining principal components: The pattern from the first two PCs continues, so that the remaining principal components are $\boldsymbol{\alpha}_{k} \mathbf{x}$, where $\boldsymbol{\alpha}_{k}$ is the eigenvalue
corresponding to the $k$ th eigenvalue.
We refer to the $\boldsymbol{\alpha}_{k}^{\prime} \mathbf{x}$ 's as principal components, and the $\boldsymbol{\alpha}_{k}$ vectors as "loadings." The loadings for PC $k$ tell us how each original variable is weighted in PC $k$. For instance, if $\alpha_{5}$ is large, it means that variable 5 is an important contributor to PC 5.

Back to Data: Now that we showed that we need to consider eigenvectors to derive principal components, let's go back to the data and do it. Again assume that the data has already been mean-centered, so that we have subtracted the mean of each covariate to form the $n \times p$ data matrix $\mathbf{X}$, whose entries are thus: $x_{i j}=x_{i j}-\bar{x}_{j}$. Define the covariance matrix:

$$
\hat{\boldsymbol{\Sigma}}=\frac{1}{n-1} \mathbf{X}^{\prime} \mathbf{X}
$$

Now, let us take its top $K$ eigenvalues (assuming we want $K$ PCs) and form their eigenvectors into a $p \times K$ matrix $\mathbf{A}$. Each column is an eigenvector.

Now, we project the data onto those first $K$ components:

$$
\mathbf{X}^{\text {projected }}=\mathbf{X A}
$$

$\mathbf{X}$ is $n \times p$ and $\mathbf{A}$ is $p \times K$, so the result is $n \times K$. That is, we projected $n$ points from $p$ dimensions to $K$ dimensions. The graphic below shows an example of two principal components for 2D data. The longer axis corresponds to the first principal component because that is the direction with the most variation. The second PC is always perpendicular to it.


If desired, we could project the data down to the first PC, and ignore the second PC. In that case, all the data would be projected to lie along the first PC and we would lose some information.


But we didn't lose too much information because we only lost information where the variance was small!

## Discussion

PCA is straightforward and preserves global structure. The benefits of PCA are that it is straightforward and does not involve optimization, and that it really does allow us to keep the dimensions where the interesting variation in the data lies. PCA nicely preserves the global structure of high-dimensional data, in that the points at the extremes of the projection are at the extremes of the high-dimensional space.

PCA does not preserve neighborhoods or local structure. There are disadvantages to PCA. If the data live along manifolds or in clusters, that information will not necessarily be preserved when projecting to only 2 or 3 PCs. PCA does not preserve graph structure - in other words, it does not aim to preserve which points are neighbors in the high-dimensional space. So we should not expect to see neighborhoods or clusters preserved when we project to 2D using PCA.

PCA for preprocessing does not preserve interpretability. PCA is typically used as a preprocessing step to other methods (though here we're discussing dimension reduction). When using PCA as a preprocessing step, we often lose interpretability of the models we create from the PCA-transformed data. For instance, let us say we did PCA and then created a sparse model after that. This new model is not sparse in the original variables, it is only sparse in the PCs. But each PC is generally created from all of the other variables. So even though the transformed model is sparse in PCs, it is still a combination of all of the original variables.

PCA can be kernelized. Here we would replace the inner product between x's with the inner product in the new space.

$$
\operatorname{Var}\left(\boldsymbol{\alpha}^{\prime} \mathbf{x}\right)=\mathbb{E}\left(\left(\boldsymbol{\alpha}^{\prime} \mathbf{x}\right)^{2}\right)=\mathbb{E}\left(\boldsymbol{\alpha}^{\prime} \mathbf{x x}^{\prime} \boldsymbol{\alpha}\right) \rightarrow \mathbb{E}\left(\boldsymbol{\alpha}^{\prime} k(\mathbf{x}, \mathbf{x}) \boldsymbol{\alpha}\right)
$$

Since I have already discussed kernels in depth, I'll refrain from it again here.

## Demonstration

Mammoth. Here is an example of PCA on the mammoth dataset (The Smithsonian Institute, 2020) which is shown on the left. The whole dataset is 3 dimensional. The image on the right is the mammoth dataset after preserving 2 of the 3 PCs. PCA captures the dominant two dimensions of the mammoth, and ignores the final (lower variance dimension).

Original mammoth dataset


PCA result


S-Curve with a hole. This is another 3D dataset projected onto the first two PCs.


While 3D datasets are good for demonstration, most datasets are not 3D, so there is often significant information lost when projecting down to 2D.

HIV dataset. This dataset contains gene expression levels of 17 K genes for 59 K cells (Kazer u. a., 2020).


I think this dataset may actually have some cluster structure that is not visible from the PCA results, as we will see when we look at PaCMAP's results.

MNIST handwritten digits. The number of dimensions is the number of pixels in the images. It is not possible to display the full dataset without some kind of dimension reduction, so I will show some samples on the left, and PCA's result on the right, projecting onto 2 PCs .

Samples from MNIST dataset MNIST PCA result


After projecting to the first two PCs, the global structure is preserved, but you would not be able to tell from this image that the clusters of handwritten digits are fairly well separated in the high-dimensional space. In order to see this when projecting to 2D, we need a method that preserves neighborhood structure.

## 2 Dimension Reduction with PaCMAP

There are a huge number of algorithms for dimension reduction - hundreds or thousands of them. They have usually fallen into two categories: local and global structure-preservation algorithms. There is no strict definition of local or global structure preservation, but I typically think of a global structure-preservation method (like PCA) as preserving overall relative placement of large clusters. I think of local structure-preservation methods as preserving neighborhoods, so that neighbors in the high-dimensional space are still neighbors in the lowdimensional space.

For global structure preservation, I typically think of older methods like PCA or multidimensional scaling (MDS) (Torgerson, 1952). There is a lot of recent work on local structure preservation including LLE (Roweis und Saul, 2000), Isomap (Tenenbaum u. a., 2000), Laplacian Eigenmaps (Belkin und Niyogi, 2001), t-SNE (van der Maaten und Hinton, 2008), LargeVis (Tang u. a., 2016), and UMAP (McInnes u. a., 2018) (see van der Maaten u. a., 2009; Bellet u. a., 2013; Belkina u.a., 2019, for useful surveys.).

A typical introductory lecture on dimension reduction might feature t-SNE and UMAP, which are really popular in biology, but those methods tend to be nearsighted, meaning that they don't preserve global structure, tend to be unstable with initialization (giving totally different results depending on how it is ini-
tialized). They are also difficult to understand and explain. Local dimension reduction methods often make some critical errors, for instance:

- They try to maintain the actual distance values between points, but in high dimensions, the distances between points are all similar. These types of techniques don't seem to work. It seems to be better to maintain relative distances: which points are near and which points are far.
- They do not optimize global structure. An algorithm that has a zero loss whenever the high-dimensional neighbors are close and the high-dimensional further points are far is not good enough. For instance, in the figure below, the original dataset is on the left (which are points along a curve in 2 D , so the algorithm just needs to leave the points where they are for an optimal solution), but the UMAP result on the right moves the points around in a way that does not preserve global structure. But, it does keep nearby points near, and it does keep farther points far.


So, I chose to focus here on a more recent (and much simpler) method, PaCMAP (Wang u. a., 2021), which aims to preserve both local and global structure. The figure below shows the projected Mammoth dataset using t-SNE with several choices for its parameters, LargeVis with several parameter values, UMAP with several parameter values, and PaCMAP with its default parameters (it shouldn't need tuning of the parameters).


In the cases on the left where the mammoth seems to have been run over by a steam roller, the Mammoth's global structure was not preserved. In cases where the Mammoth's legs have been split off from its body, global structure has not been preserved.

PacMAP (Pairwise Controlled Manifold Approximation Projection) incorporates several principles in its design, and handles local and global preservation in different ways.

First, some notation. Define the scaled distances between pairs of observations $i$ and $j$ :

$$
d_{i j}^{2, \text { select }}=\frac{\left\|\mathbf{x}_{i}-\mathbf{x}_{j}\right\|^{2}}{\sigma_{i j}}
$$

and $\sigma_{i j}=\sigma_{i} \sigma_{j}$, where $\sigma_{i}$ is the average distance between $i$ and its Euclidean nearest fourth to sixth neighbors. Here we just define the distances, we do not precompute these distances, as we will select only a small subset of them and can compute the distances after selecting them. These distances are scaled to take into account the local density of points nearby, so that smaller distances within dense areas appear to be larger, for instance.

## PaCMAP Algorithm Outline

Input: Data points $\mathbf{X}$, which is $n \times p$.
Output: Projected data points $\mathbf{X}^{\text {projected }}$, which is $n \times 2$ (or 3 ).

1. Collect all of the information we will keep from the high-dimensional data. To do this, compile a set of pairs of data points as follows:

- Near pairs NB: Pair each point $i$ with its nearest $n_{N B}$ neighbors according to distance $d_{i j}^{2 \text { sselect }}$. Default $n_{N B}$ value is 10 . These points will be strongly attracted.
- Mid-near pairs MN: For each $i$, sample 6 observations, choose the second closest of the 6 , and pair it with $i$. The number of mid-near pairs to compute is $n_{M N}=\left\lfloor n_{N B} \times M N \_\right.$ratio $\rfloor$. Default $M N \_$ratio is 0.5 . These points will be weakly attracted.
- Further pairs FP: Sample non-neighbors by using random sampling (here we assume most points are not neighbors). The number of such pairs is $n_{F P}=\left\lfloor n_{N B} \times F P\right.$ ratio $\rfloor$, where default $F P$ _ratio $=2$. These points will be repulsed.

The illustration below shows these three types of points, green for neighbors, purple for mid-near points, and blue for further points.

2. Initialize $\mathbf{X}^{\text {projected }}$ using PCA to the first 2 PCs.

Now that we have all information we need from the high-dimensional data, we will start optimizing the values of $\mathbf{X}^{\text {projected }}$.
3. Adjust $\mathbf{X}^{\text {projected }}$ to minimize the loss below in three stages, where $\tilde{d}_{a b}=$

$$
\begin{align*}
& \left\|\mathbf{x}_{a}^{\text {projected }}-\mathbf{x}_{b}^{\text {projected }}\right\|^{2}+1: \\
& \operatorname{Loss}^{\text {PaCMAP }}= \\
& w_{N B} \cdot \sum_{i, j \text { are neighbors }} \frac{\tilde{d}_{i j}}{10+\tilde{d}_{i j}} \text { (strong attraction) }  \tag{6}\\
& +w_{M N} \cdot \sum_{i, k \text { are mid-near pairs }} \frac{\tilde{d}_{i k}}{10000+\tilde{d}_{i k}} \text { (weak attraction) }  \tag{7}\\
& +w_{F P} \cdot \sum_{i, l \text { are further points }} \frac{1}{1+\tilde{d}_{i l}} . \quad \text { (repulsion) } \tag{8}
\end{align*}
$$

These terms might look arbitrary, and they are. But it does not matter exactly what these functions are, as long as we choose functions that obey a general set of principles (Wang u. a., 2021), it will probably work. For instance, the NB term is large if the neighbors in high dimensions are not close in the 2D projected space, which is what causes attraction. As long as we strongly attract neighbors, weakly attract mid-near points, and repulse further points, it should generally work, with the caution that once the neighbors get close enough we would like to stop pulling them towards us to avoid problems with crowding. We change the parameters over 3 stages.
(a) Global Structure Optimization. In the first stage (i.e., the first 100 iterations of an optimizer such as Adam), the mid-near points are weighted very highly, and they help maintain global structure. Set $w_{N B}=2, w_{M N}(t)=1000 \cdot\left(1-\frac{t-1}{100}\right)+3 \cdot \frac{t-1}{100}, w_{F P}=1 . \quad\left(\right.$ Here, $w_{M N}$ decreases linearly from 1000 to 3.)
(b) Local and Global Structure Optimization. In the second stage (iterations 101 to 200), set $w_{N B}=3, w_{M N}=3, w_{F P}=1$.
(c) Refine Local Structure. In the third stage (iteration 201 to 450), $w_{N B}=$ $1, w_{M N}=0, w_{F P}=1$.

Again, the parameters don't matter too much, it's the spirit of what they are doing that matters.

## 4. Return $\mathbf{X}^{\text {projected }}$.

Several important choices made in PaCMAP allow it to maintain both local and global structure.

- Notice that in the loss function, near pairs have a strong attractive force: if $\tilde{d}_{i j}$ is large, then neighbors are too far away from each other in the projected space and the the loss term in (6) is large. This loss term ensures that local structure is maintained: neighbors in the high-dimensional space should be neighbors in the low-dimensional space.
- Note that when $\tilde{d}_{i j}$ is small for neighbor pairs, the force goes to 0 . This helps to avoid "crowding" of points close to one spot.
- The repulsive forces between further points in (8) are necessary to maintain both local and global structure, since points that are far away in highdimensional space should be far in the projected space.
- The weakly attractive force on mid-near pairs in (7) are essential for maintaining global structure. Mid-near pairs ensure that clusters that are near each other in high dimensions are still near each other in the projected space. The force is weak so that its attractive forces do not interfere with the neighbors term (6), but it is strong enough so that nearby clusters are pulled towards each other. (Think of weak attractive forces between many points within two nearby clusters.)

In its three stages, PaCMAP first emphasizes fixing global structure, then gradually eases its global structure optimization and starts optimization of the local structure. In the third stage, it just refines the local structure. Let us show what happens over iterations of PaCMAP.

For the mammoth, when we start from the PCA solution, the algorithm needs to separate the tusks and the ribs. Stage 1 forces move the points around without consideration of the local structure, until they are roughly in the right areas around iteration 100. In Stage 2, the local and global structure are nicely optimized, and the result looks quite good by iteration 170 .


If we had instead started from a random initialization rather than PCA, you can see the global structure form more clearly.


For MNIST, a similar pattern arises, where by the end of Stage 1 at iteration 100, the global structure is in place thanks to the attractive forces on the mid-near points. Here it is also easier to see the improvement from the refinement in the third stage. We will start from PCA again:


This time, we will start from random initialization rather than PCA so we can see the global structure form.


It is hard to resist including more pictures. I will include some pictures from the S-curve dataset.


The last one is the HIV dataset, where cluster structures appear that were not there previously.


## References

[Belkin und Niyogi 2001] Belkin, Mikhail ; Niyogi, Partha: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in Neural Information Processing Systems Bd. 14, MIT Press, 2001, S. 585-591
[Belkina u. a. 2019] Belkina, Anna C.; Ciccolella, Christopher O. ; Anno, Rina ; Halpert, Richard ; Spidlen, Josef ; Snyder-Cappione, Jennifer E.: Automated optimized parameters for T-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. In: Nature Communications 10 (2019), Nr. 5415
[Bellet u. a. 2013] Bellet, Aurélien ; Habrard, Amaury ; Sebban, Marc: A Survey on Metric Learning for Feature Vectors and Structured Data. In: arXiv e-prints (2013), Juni, S. arXiv:1306.6709
[Kazer u. a. 2020] Kazer, Samuel W.; Aicher, Toby P. ; Muema, Daniel M. ;

Carroll, Shaina L. ; Ordovas-Montanes, Jose ; Miao, Vincent N. ; Tu, Ang A. ; Ziegler, Carly G. ; Nyquist, Sarah K. ; Wong, Emily B. u. a.: Integrated single-cell analysis of multicellular immune dynamics during hyperacute HIV-1 infection. In: Nature Medicine 26 (2020), Nr. 4, S. 511-518
[van der Maaten und Hinton 2008] Maaten, Laurens van der ; Hinton, Geoffrey: Visualizing data using t-SNE. In: Journal of Machine Learning Research 9 (2008), S. 2579-2605
[van der Maaten u. a. 2009] Maten, Laurens van der ; Postma, Eric O. ; Herik, Jaap van den: Dimensionality Reduction: A Comparative Review / Tilburg University. The Netherlands, 2009. - Forschungsbericht
[McInnes u.a. 2018] McInnes, Leland ; Healy, John ; Melville, James: UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. In: arXiv e-prints (2018), Februar, S. arXiv:1802.03426
[Pearson 1901] Pearson, Karl: On Lines and Planes of Closest Fit to Systems of Points in Space. In: Philosophical Magazine 2 (1901), Nr. 11, S. 559-572
[Roweis und Saul 2000] Roweis, Sam T. ; Saul, Lawrence K.: Nonlinear Dimensionality Reduction by Locally Linear Embedding. In: Science 290 (2000), Nr. 5500, S. 2323-2326
[Tang u. a. 2016] Tang, Jian ; Liu, Jingzhou ; Zhang, Ming ; Mei, Qiaozhu: Visualizing large-scale and high-dimensional data. In: Proceedings of the 25th International Conference on the World Wide Web, 2016, S. 287-297
[Tenenbaum u. a. 2000] Tenenbaum, Joshua B. ; Silva, Vin de ; Langford, John C.: A Global Geometric Framework for Nonlinear Dimensionality Reduction. In: Science 290 (2000), Nr. 5500, S. 2319-2323

| $[$ The Smithsonian | Institute 2020] | The | Smithsonian |
| :---: | :---: | :---: | :---: | ---: |
| Institute: | Mammuthus | primigenius | (Blumbach). | https://3d.si.edu/object/3d/mammuthus-primigenius-blumbach:341c96cd-f967-4 2020

[Torgerson 1952] Torgerson, Warren: Multidimensional scaling: I Theory and Method. In: Psychometrika 17 (1952), Nr. 4, S. 401-419
[Wang u.a. 2021] Wang, Yingfan ; Huang, Haiyang ; Rudin, Cynthia ; Shaposhnik, Yaron: Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMap, and PaCMAP for Data Visualization. In: Journal of Machine Learning Research 22 (2021), Nr. 201, S. 1-73. - URL http://jmlr.org/papers/v22/20-1061.html

