TOWARDS A COMMON DIMENSIONALITY REDUCTION APPROACH;
COMPARING PCA, TSNE, AND UMAP THROUGH A COHESIVE FRAMEWORK

by

Andrew Draganov
A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Master of Science
Mathematics

Committee:

_____	Dr. Tyrus Berry, Thesis Director
_____	Dr. Tyrus Berry, Committee Member
_____	Dr. Matthew Holzer, Committee Member
_____	Dr. Timothy Sauer, Committee Member
_____	Dr. David Walnut, Department Chairperson
_____	Dr. Donna M. Fox, Associate Dean, Office of Student Affairs & Special Programs College of Science
_____	Dr. Fernando Miralles-Wilhelm, Dean, College of Science

Date: _____ Spring Semester 2021
George Mason University
Fairfax, VA

Towards a Common Dimensionality Reduction Approach; Unifying PCA, tSNE, and UMAP through a Cohesive Framework

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

By

Andrew Draganov
Bachelor of Science
University of Virginia, 2017

Director: Dr. Tyrus Berry, Professor
Department of Mathematics

Spring Semester 2021
George Mason University
Fairfax, VA

Dedication

I dedicate this thesis to my family, coworkers and mentors.

Acknowledgments

I would like to thank the following people who made this possible:

1. Tyrus Berry, whose patient mentoring and willingness to hear out my thoughts proved invaluable as I struggled from one derivation to the next.
2. My father, who consistently volunteered to listen to my latest thoughts on the subject.
3. My coworkers, who encouraged me to pursue interesting subjects that ultimately became the topic of this thesis.

Table of Contents

Abstract

TOWARDS A COMMON DIMENSIONALITY REDUCTION APPROACH; UNIFYING PCA, TSNE, AND UMAP THROUGH A COHESIVE FRAMEWORK

Andrew Draganov

George Mason University, 2021

Thesis Director: Dr. Tyrus Berry

Dimensionality reduction is a widely studied field that is used to visualize data, cluster samples, and extract insights from high-dimensional distributions. The classical approaches such as PCA, Isomap, and Laplacian eigenmaps rely on clear optimization strategies while more modern approaches such at tSNE and UMAP define gradient descent search spaces through disparities between the high- and low-dimensional datasets.

In this work, we notice that all of these approaches can be interpreted as minimizing the difference between two kernel functions – one for the high dimensional space and one for the low dimensional space. In particular, once we abstract the kernel functions, we can develop a common framework for any dimensionality reduction problem. Namely, one needs to identify their high-dimensional distance kernel, the low-dimensional distance kernel, and the method used for minimization.

With this in mind, we identify the relevant general framework and then proceed to discuss the ways in which PCA, tSNE, and UMAP all fit into it. For each, we discuss insights that were obtained during the process. We lastly highlight next steps and directions for future work.

Chapter 1: Introduction

Many disciplines rely on high-dimensional data representations for the purposes of analysis. However, these datasets are usually impossible to visualize due to residing in > 3 dimensions. To circumvent this, the field of dimensionality reduction studies how to *faithfully* represent high-dimensional data in a low-dimensional space. 'Faithfully' here means two things:

1. Samples that are close in the high-dimensional space should remain close in the low-dimensional space
2. Samples that are far in the high-dimensional space should remain far in the low-dimensional space

There are multiple approaches for achieving this. PCA and other linear decomposition problems simply try to find the optimal linear projection for representing the data in a low-dimensional space according to some metrics. Manifold learning algorithms such as Isomap [1], Laplacian eigenmaps [2], Parallel Transport Unfolding [3], and others attempt to define a graph representation in the high-dimensional space and to make a similar graph of points in the low-dimensional space. Then others such as t-SNE [4] and UMAP [5] rely on gradient descent to minimize an objective between the high-dimensional probability distributions and low-dimensional probability distributions.

Similarly to [6], we posit that all of these approaches can be viewed through a graph-theoretic framework, which then extends the gradient descent optimization to be applicable to any of the algorithms. Specifically, each of the above approaches first defines kernels on similarity metrics in the high- and low-dimensional spaces before proceeding to find points in the low-dimensional space that minimize some dissimilarity.

In the case of gradient-descent based optimization for t-SNE and UMAP, they minimize the KL divergence between probability distributions. Thus, the edge weights on the graph

are defined by $E_{ij}^X = p_{ij}(x_i, x_j)$ and $E_{ij}^Y = q_{ij}(y_i, y_j)$, where p is a Gaussian and q is a student T distribution. Then minimizing the KL divergence $KL_i = \sum_j p(x_i, x_j) \cdot log(\frac{p(x_i, x_j)}{q(y_i, y_j)})$ can be reframed in terms of the above reparameterization.

Given this intuition, we proceed by

1. Establishing the algorithms that we intend to analyze
2. Defining the analysis framework with sufficient generality
3. Applying the framework to each algorithm and highlighting insights

We finally conclude by summarizing our work and identifying next steps that we believe are worth further investigation.

Chapter 2: Introduction to Various Dimensionality Reduction Algorithms

We begin by describing PCA, t-SNE, and UMAP while attempting to draw parallels between them.

2.1 Principal Component Analysis

Principal Component Analysis (PCA) is likely the most famous dimensionality reduction algorithm. It can be interpreted in many ways, but is most commonly thought of as a linear optimization that removes the axes of lowest variance. More formally, we remove the axes w^* defined by:

$$w^* = \underset{w^T w=1}{argmin}\ w^T C w \tag{2.1}$$

In the above equation, the covariance matrix C for the high-dimensional points X is defined by:

$$C = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T \rightarrow$$

$$\rightarrow = XX^T$$

. Due to the outer-product definition of the covariance, we note that this definition of PCA relies on optimizing the low-dimensional representation according to distance functions between the high- and low-dimensional distributions. Specifically, we are maintaining the dimensions of highest variance, which means that the sum of squares of our points' distance to the origin remains maximized.

This preservation of axes of highest variance can also be interpreted in other ways. Euclidean Multi-Dimensional Scaling (MDS), in which one uses linear operations to find the points that minimize the difference in pairwise dot products, can be shown to be identical to PCA. Said otherwise, PCA can be thought of as finding the points in a low-dimensional space such that their dot products are closest to the high-dimensional space's dot products according to the sum of squared differences. Additionally, in section 4.1, we show a Laplacian reinterpretation of PCA as a graph-embedding algorithm.

All of these definitions of PCA share the same basic foundation. They attempt to find a low-dimensional set of points that linearly minimize an metric function with respect to the original high-dimensional points.

2.2 t-distributed Stochastic Neighborhood Embedding

In contrast to PCA's distance-based methodology, the t-SNE algorithm attempts to match probability distributions between the high- and low-dimensional spaces. These probability distributions define the likelihood that a point x_i will choose point x_j as its neighbor. In the high-dimensional space, the authors chose the Gaussian based on Euclidean distance to define the probability that point x_i chooses point x_j as its neighbor:

$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_k \exp(-||x_i - x_k||^2/2\sigma_i^2)} \tag{2.2}$$

The σ_i in this equation is chosen through binary search to standardize the spread of the probability distributions with respect to the varying density across the high-dimensional space. To further standardize the distributions, the authors average the probabilities of point x_i given x_j and point x_j given x_i by setting $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2}$.

The authors then define their low-dimensional probability distribution using the student-t distribution:

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}. \tag{2.3}$$

. This choice was made to undo the effects of the *crowding problem*, in which lower-dimensional representations have much less "space" to distribute points than high-dimensional spaces do. Through using the student-t distribution's heavy-tailed kernel, the authors find that far-away points are more likely to maintain nearest-neighbor relationships, despite the fact that they may have less space on the manifold to choose from.

We note that the t-SNE probability kernels are both defined with respect to Euclidean distance and create a full matrix of high- and low-dimensional pairwise similarities between the points. The t-SNE objective, then, can be interpreted as a weighted minimization over the pairwise similarities that are defined through these kernels.

2.3 UMAP

Uniform Manifold Approximation and Projection (UMAP) is a similar algorithm to t-SNE in that it minimizes pairwise similarities using gradient descent. However, UMAP was motivated by a theoretical approach wherein the authors attempt to find a manifold-based representation of the high- and low-dimensional spaces. Specifically, they define the pseudo-distance metric $D(x_i, x_j) = d(x_i, x_j) - \rho(x_i)$, where d is some regular distance metric and ρ is the minimum of this distance to any of the other points. This pseudo-metric is particularly useful for alleviating the issues with Euclidean nearest neighbors in high-dimensional spaces as noted in [7]. Using this pseudo-metric, they claim that they obtain a uniform distribution across the space and can then utilize topological tools for defining the minimization criterion.

However, the authors of UMAP arrive at a very similar set of functions to t-SNE's high- and low-dimensional probability distributions. Namely, they define

$$v_{j|i} = exp[(-d(x_i, x_j) - \rho_i)/\sigma_i]; v_{ij} = v_{j|i} + v_{i|j} - v_{j|i}v_{i|j} \tag{2.4}$$

$$w_{ij} = (1 + a||y_i - y_j||_2^{2b})^{-1} \tag{2.5}$$

where v_{ij} is the high-dimensional likelihood of a nearest-neighbor relationship and w_{ij} is the low-dimensional counterpart. Rather than representing probability distributions across the entire set of points as t-SNE does, UMAP instead represents every nearest-neighbor relationship between two points as a Bernoulli random variable. This has the inherent benefit of making the edge values independent of the vantage point. Thus, they avoid requiring the normalization terms across the entire dataset, instead opting to keep each edge's probability independent of the others (in this respect).

2.4 Commonalities

Notice that each of these algorithms, in a general sense, attempts to minimize the discrepancy between functionals in the high- and low-dimensional spaces. Each of these functionals can be defined in terms of kernels that are acting on the distances between pairs of points. We can further generalize this to define connected graphs for the high- and low-dimensional spaces, wherein vertices represent the points and edges represent the functional values between points. Using this interpretation, we observe that a dimensionality reduction algorithm is ultimately attempting to find a set of points that would minimize some similarity metric between the two graphs.

Chapter 3: Uniting These Algorithms Under a Common Framework

3.1 Identifying a common framework

As mentioned, all of the above algorithms can be interpreted as consolidating the connected graphs between high- and low-dimensional spaces. We consider the connected graphs as the probabilities that each point will choose another point as its nearest neighbor. In this formulation, the points are represented as vertices and their nearest neighbor probabilities are the values on the edges.

Given a data set $X = \{x_i\}_{i=1}^N \subset \mathbb{R}^n$ an optimal embedding is another data set $Y = \{y_i\}_{i=1}^N \subset \mathbb{R}^m$ that solves an optimization problem,

$$\min_Y F(X, Y)$$

for some functional F. Typically F only depends on the distances between points, thus we define

$$D_{ij}^X = d_X(x_i, x_j)^2, \qquad D_{ij}^Y = d_Y(y_i, y_j)^2$$

for pointwise similarity measures d_X and d_Y. Moreover, F typically has the form,

$$F(X, Y) = \tilde{F}(D^X, D^Y),$$

where $\tilde{F}$ represents some similarity measure between the high- and low-dimensional pairwise distance matrices D^X and D^Y. Lastly, we choose to instead interpret $\tilde{F}$ in terms of symmetrized kernels on the distances D^X and D^Y, which we define as v and w respectively.

Therefore, our similarity measure between high- and low-dimensional datasets X and Y can be written as:

$$F(X,Y) = \tilde{F}(D^X, D^Y) = \sum_{i \neq j}^{N} \hat{F}\left(S^X\left(v(D_i^X), v(D_j^X)\right), S^Y\left(w(D_i^Y), w(D_j^Y)\right)\right)$$

where D_i^X and D_i^Y represent rows of the pairwise distance matrices. Notice that each of the above algorithms from section 2 defines a kernel on distances in the high- and low-dimensional spaces. Thus, we aim to identify the above algorithms through their kernels while fixing $\hat{F}$. $\hat{F}$ then becomes a similarity measure on graphs, where the high- and low-dimensional graphs share their vertices (the points) but we define edge weights through the separate kernel functions.

3.2 Choosing a probabilistic interpretation

Given the high-dimensional graph

$$G^X = \left(X, \{S^X\left(v(D_i^X), v(D_j^X)\right) \mid 0 < i, j < N\}\right) = \left(X, E^X\right)$$

of points and their kernel distances, we can interpret the edges as probability distributions in two ways. The first is to define a probability distribution for every point x_i to all of its nearest neighbors. This would mean that our kernel must normalize x_i's outgoing edges across the entire dataset, giving us

$$E_{ij}^X = \frac{v(D(x_i, x_j))}{\sum\limits_{\substack{k=1\ldots n \\ k \neq i}} v(D(x_i, x_k))}$$

We refer to this as the neighborhood probabilistic interpretation, and it is the option that t-SNE uses for its kernel functions in the high- and low-dimensional spaces. Intuitively, it defines a probability distribution for each point to its set of neighbors in the space, however one defines neighbor relationships. At this time we think it's important to note that t-SNE actually defines its high-dimensional kernel as a probability distribution across rows of the distance matrix while the low-dimensional kernel defines a probability distribution over the entire set of pairwise distances. More on this in section 5.1.

However, we can choose to instead treat every edge independently. Indeed, by choosing to not normalize we obtain the simpler kernel function

$$E_{ij}^X = S^X\left(v(D_i^X), v(D_j^X)\right),$$

wherein every edge is simply defined as having some probability of existing. This interpretation is the one that UMAP uses, and it allows us to treat the edges as Bernoulli random variables. Thus, our kernel function in this case defines the likelihood that a nearest-neighbor relationship exists, implicitly defining the additional state of it not existing.

The reader may notice that we are using both the i-th row and the j-th row to define the probability of an edge existing. Indeed, these may not be symmetrical. To remedy this, we can naturally take the mean and define

$$E_{ij}^X = \frac{E_{j|i}^X + E_{i|j}^X}{2}.$$

However, we have an additional option available when we use the Bernoulli interpretation. Specifically, if we consider the edges $e_{j|i}^X$ and $e_{i|j}^X$ as two separate events, then we can define each edge as the probability that at least one of the two will exist. This is defined as

$$E_{ij}^X = E_{j|i}^X + E_{i|j}^X - E_{j|i}^X E_{i|j}^X.$$

Due to using the neighborhood probabilistic interpretation, t-SNE is confined to using the mean to obtain symmetry. However, UMAP utilizes the Bernoulli interpretation and thus chooses to use the or-probability to symmetrize its edges. Both define their low-dimensional kernels to be inherently symmetric, and therefore don't utilize any symmetrization function in the low-dimensional space.

Going forward, we attempt to define our graphs using the Bernoulli interpretation and generalize the symmetrization function as S.

3.3 Optimizing the low-dimensional embeddings

In the formulation defined in 3.1, $\hat{F}$ depends on the kernel v of the high-dimensional dataset, where the points are given, and the kernel w of the low-dimensional dataset, where the points are learned. As established, we interpret these kernel functions as the likelihood that a nearest-neighbor relationship exists. Thus our goal in performing dimensionality reduction is to try to learn a low-dimensional set of points that would have the same nearest-neighbor likelihoods as their high-dimensional counterparts (subject to the kernel functions).

More formally, for edges E_{ij}^X and E_{ij}^Y defining the symmetrized nearest-neighbor likelihoods between points i and j in the high- and low-dimensional spaces respectively, we may choose to minimize the KL divergence $\mathrm{KL}(E_{ij}^X \mid\mid E_{ij}^Y)$. For example, this is necessary when we are recreating tSNE and UMAP.

Recall that our edges have probability of existing defined by $E_{ij}^X = S^X(v(D_i^X), v(D_j^X))$ and $E_{ij}^Y = S^Y(w(D_i^Y), w(D_j^Y))$. Then the application of the KL divergence is appropriate since we have $N \times N$ Bernoulli distributions – one for each edge. Note that in this context, minimizing the KL divergence is equivalent to minimizing the cross-entropy. Let us first define the KL divergence and cross-entropy H from some high-dimensional kernel to a low-dimensional kernel.

$$KL(v||w) = -\sum_{x \in \mathcal{X}} v(x)log(w(x)) - v(x)log(v(x))$$
$$H(v,w) = -\sum_{x \in \mathcal{X}} v(x)log(w(x)) \tag{3.1}$$

where $\mathcal{X}$ represents the set of discrete states that the probability distribution can occupy. In our case, this is just the two options of the edge between two points existing or not existing. Notice that the first term in the KL divergence is exactly the cross-entropy and that the second term is a constant when we assume that our high-dimensional points are given. Thus, minimizing one of the above equations inherently minimizes the other. For this reason, we will be discussing minimizing the cross entropy, which is the simpler calculation, under the acknowledgement that this simultaneously optimizes the KL divergence between probability distributions, which is perhaps a more intuitive measure of probabilistic similarity.

Given this context, the cross entropy on a Bernoulli random variable can be disassembled into its two discrete states to obtain

$$H(E_{ij}^X, E_{ij}^Y) = -E_{ij}^X log(E_{ij}^Y) - (1 - E_{ij}^X)log(1 - E_{ij}^Y) \tag{3.2}$$

Since we are optimizing over all of the points in the dataset, we must then do this calculation for each point.

Given this framework and optimization criterion, we proceed by highlighting which algorithms can fit within it and identifying their strengths or weaknesses.

Chapter 4: Applying the Framework to PCA

4.1 PCA as a Graph Embedding Method

Given some background on Laplacian matrices, we can reinterpret PCA in terms of a graph embedding approach. We will first reintroduce PCA, then show how it naturally relates to modeling the data as a fully-connected graph.

First note that for a dataset X, the covariance matrix C is equal to

$$C = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T \rightarrow$$

$$\rightarrow = XX^T$$

We intend to show that PCA, in removing the directions with minimal variance, is actually optimizing the following objective:

$$w^* = \underset{w^T w=1}{\operatorname{argmin}}\ w^T C w \tag{4.1}$$

Why might this be the case? Recall that PCA is operating on the squared errors between x_i and its projections on the principal components. So it is finding w^* such that

$$
\begin{aligned}
w^* &= \arg\min_w \sum_{i=1}^{N} ||x_i - (w^T x_i)w||^2 \\
&= \arg\min_w \sum_{i=1}^{N} ||x_i - ww^T x_i||^2 \\
&= \arg\min_w \sum_{i=1}^{N} [x_i - ww^T x_i]^T [x_i - ww^T x_i] \\
&= \arg\min_w \sum_{i=1}^{N} [x_i^T x_i - x_i^T ww^T x_i - x_i^T ww^T x_i + x_i^T ww^T ww^T x_i]
\end{aligned}
$$

But $w^T w = 1$, so the last two terms cancel, giving us

$$
\begin{aligned}
w^* &= \arg\min_w \sum_{i=1}^{N} [x_i^T x_i - w^T x_i x_i^T w] = \\
&= \arg\min_w \sum_{i=1}^{N} x_i^T x_i - w^T (\sum_{i=1}^{N} x_i x_i^T) w
\end{aligned}
$$

The first term here doesn't depend on w, giving us

$$
w^* = \underset{w}{argmax}\ w^T (XX^T) w
$$

where $X = [x_1, x_2, ..., x_N]$.

This, though, is just an eigendecomposition problem. For further clarification, note that $w^T(XX^T)w$ will be maximized when $(XX^T)w$ is parallel to w, so it must be that w is the first eigenvector of XX^T and $\lambda = w^T(XX^T)w$ is the first eigenvalue. Thus, applying a constraint that our principal components w are of unit length, we obtain the initially stated formulation: $w^* = \underset{w^T w=1}{\text{argmin}}\ w^T C w$

With this derivation out of the way, we can show that PCA is really just linearizing a fully connected graph where all data pairs have equal weights.

$$\begin{aligned} w^* &= \underset{w^T w=1}{\operatorname{argmin}}\ w^T C w \\ &= \underset{w^T w=1}{\operatorname{argmin}}\ w^T X X^T w \end{aligned}$$

If we may find a graph-Laplacian matrix L such that $XLX^T = XX^T$, then we can apply the following reformulation:

$$\begin{aligned} w^* &= \underset{w^T w=1}{\operatorname{argmin}}\ w^T X X^T w \\ &= \underset{w^T w=1}{\operatorname{argmin}}\ w^T X L X^T w \\ &= \underset{w^T w=1}{\operatorname{argmin}}\ (w^T X) L (w^T X)^T, \end{aligned}$$

which is just a graph eigendecomposition problem akin to the ones done in Laplacian eigenmaps and spectral clustering.

Such an L can be found by making a fully connected graph on the N data points with edge weights $1/N$:

$$L = D - A = \begin{bmatrix} \frac{N-1}{N} & \cdots & -\frac{1}{N} \\ \vdots & \ddots & \vdots \\ -\frac{1}{N} & \cdots & \frac{N-1}{N} \end{bmatrix} \tag{4.2}$$

This is exactly the centering matrix as defined in [8], which, when multiplied with a vector, has the effect of subtracting the mean of the components of a vector from every component of that vector. Centering matrices have convenient properties; namely they are symmetric positive semi-definite, singular with a single eigenvalue of 0, and all of the rows and columns sum to 0. We point out that these properties are consistent with those of graph Laplacian

matrices.

Due to the fact that the centering matrix's operation translates a vector to the origin, it will have the property that $LX^T = X^T \rightarrow XLX^T = XX^T$ for all matrices X that are already centered. This is because, for N samples with zero mean ($\sum_i x_i = 0$), we get

$$\begin{aligned}\frac{N-1}{N}x_1 - \sum_{i=2}^{N}\frac{x_i}{N} &= x_1 - \frac{x_1}{N} - \sum_{i=2}^{N}\frac{x_i}{N} \\ &= x_1 - \sum_{i=1}^{N}\frac{x_i}{N} \\ &= x_1\end{aligned}$$

This naturally generalizes to any index, not just the first. So by defining L as a fully connected adjacency matrix with edge weights $\frac{1}{N}$, we have a graph embedding that fully represents PCA's optimization criterion.

4.2 Deriving PCA's kernels and optimization criteria

Given a data set $\{x_i\}_{i=1}^N \subset \mathbb{R}^n$ represented as a matrix $X \in \mathbb{R}^{n\times N}$, we can interpret multidimensional scaling (MDS) [9] as finding a data set $Y \in \mathbb{R}^{m\times N}$ that minimizes the functional,

$$F_{\text{MDS}}(X,Y) = ||X^\top X - Y^\top Y||_F^2 = \sum_{i,j=1}^{N}((X^\top X)_{ij} - (Y^\top Y)_{ij})^2 = \sum_{i,j=1}^{N}(x_i^\top x_j - y_i^\top y_j)^2$$

Notice that F_{MDS} compares the inner products in the original data set, X, to those in the lower-dimensional data set Y. These inner products are invariant to orthogonal transformations but they are not invariant to translations. The following theorem shows that optimally preserving the inner products requires applying PCA *without* centering. Later

we will see that PCA *with* centering optimally preserves the pairwise distances, which are invariant to both orthogonal transformations and translations.

Theorem 1 (Multi-Dimensional Scaling (MDS))**.** *Let* $\{x_i\}_{i=1}^N \subset \mathbb{R}^n$ *have Gram matrix* $G_{ij} = x_i^\top x_j$ *with orthogonal eigendecomposition* $G = QDQ^\top$*. The point set* $\{y_i\}_{i=1}^N \subset \mathbb{R}^m$*, that minimizes* $F_{\text{MDS}}(X, Y)$ *is given by* $Y = SV^\top$ *where* $S_{ii} = \sqrt{D_{ii}}$ *for* $i = 1, ..., m$ *and* V *is given by the first* m *columns of* Q*.*

Proof. First, expand the cost functional as,

$$F_{\text{MDS}}(X, Y) = \sum_{i,j=1}^{N} x_i^\top x_j x_i^\top x_j - 2x_i^\top x_j y_i^\top y_j + y_i^\top y_j y_i^\top y_j$$

Computing the gradient with respect to $y_k \in \mathbb{R}^m$ we find,

$$\begin{aligned}
\nabla_{y_k} F_{\text{MDS}}(X, Y) &= \sum_{i,j=1}^{N} -\delta_{ik} 2x_i^\top x_j y_j - \delta_{jk} 2x_i^\top x_j y_i + 2\delta_{ik} y_i^\top y_j y_j + 2\delta_{jk} y_i^\top y_j y_i \\
&= \sum_{i=1}^{N} -2x_k^\top x_i y_i - 2x_i^\top x_k y_i + 2y_k^\top y_i y_i + 2y_i^\top y_k y_i \\
&= 4\sum_{i=1}^{N} y_i y_i^\top y_k - y_i x_i^\top x_k \\
&= 4YY^\top y_k - 4YX^\top x_k.
\end{aligned}$$

Setting these gradients equal to zero we find $YY^\top y_k = YX^\top x_k$ for $k = 1, ..., N$ and combining all these equations we have

$$YY^\top Y = YX^\top X \tag{4.3}$$

as a necessary condition for an optimum.

Notice that $F_{\text{MDS}}(X, Y)$ is invariant to orthogonal transformations of X since when

$U^\top U = I$ then,

$$F_{\text{MDS}}(UX, Y) = ||(UX)^\top UX - Y^\top Y||_F^2 = ||X^\top X - Y^\top Y||_F^2 = F_{\text{MDS}}(X, Y)$$

and similarly for orthogonal transformations of Y. This means that if $Y = USV^\top$ is the singular value decomposition of Y we can redefine $Y \to U^\top Y = SV^\top$. Thus, without loss of generality, we can assume that Y has SVD $Y = SV^\top$. Thus, we can rewrite (4.3) as,

$$SV^\top VS^2V^\top = SV^\top QDQ^\top$$

so that,

$$S^2V^\top = V^\top QDQ^\top$$

and,

$$S^2 = V^\top QDQ^\top V \tag{4.4}$$

which implies that V selects m of the N eigenvalues of $X^\top X$. Returning to the cost functional, we have,

$$\begin{aligned} F_{\text{MDS}}(UX, Y) &= ||X^\top X - Y^\top Y||_F^2 = ||QDQ^\top - VS^2V^\top||_F^2 \\ &= ||QDQ^\top - VV^\top QDQ^\top VV^\top||_F^2 \\ &= ||D - Q^\top VV^\top QDQ^\top VV^\top Q||_F^2 \end{aligned}$$

where the last step follows from the invariance of the Frobenius norm to orthogonal transformations. Notice that $Q^\top VV^\top Q$ is an $N \times N$ matrix with rank-m and minimizing F_{MDS} requires that $Q^\top VV^\top Q = \begin{pmatrix} I_{m\times m} & 0_{m\times(N-m)} \\ 0_{(N-m)\times m} & 0_{(N-m)\times(N-m)} \end{pmatrix}$ so that $V^\top Q = [I_{m\times m} \quad 0_{m\times(N-m)}]$

and thus V contains the first m columns of Q and so by (4.4) we have $S_{ii}^2 = D_{ii}$ for $i = 1, ..., m$. □

Next we show that the MDS solution is identical to the PCA projection but without the centering step.

Corollary 1 (Uncentered Principal Component Analysis (PCA))**.** *Let* $\{x_i\}_{i=1}^N \subset \mathbb{R}^n$ *have uncentered covariance matrix* $C = XX^\top = \sum_{i=1}^N x_i x_i^\top$ *with orthogonal eigendecomposition* $C = W\Lambda W^\top$ *with* $\Lambda_{11} \geq \cdots \geq \Lambda_{nn}$*. The point set* $\{y_i\}_{i=1}^N \subset \mathbb{R}^m$*, that minimizes* $F_{\text{MDS}}(X, Y)$ *is given by the PCA projection,* $Y = [I_{m\times m} \quad 0_{m\times(n-m)}]W^\top X$*.*

Proof. Let X have SVD given by $X = \tilde{U}\tilde{S}\tilde{V}^\top$ so that $XX^\top = \tilde{U}\tilde{S}^2\tilde{U}^\top = W\Lambda W^\top$ and $X^\top X = \tilde{V}\tilde{S}^2\tilde{V}^\top = QDQ^\top$. Together with uniqueness of the eigendecomposition, these facts imply that $X = [W \quad \tilde{W}]\sqrt{D}Q^\top$ for some orthogonal matrix $\tilde{W} \in \mathbb{R}^{n\times(N-n)}$ and $D_{ii} = \tilde{S}_{ii}^2 = \Lambda_{ii}$ for $i = 1, ..., m$. The PCA projection is given by,

$$\begin{aligned}
[I_{m\times m} \quad 0_{m\times(n-m)}]W^\top X &= [I_{m\times m} \quad 0_{m\times(n-m)}]W^\top[W \quad \tilde{W}]\sqrt{D}Q^\top \\
&= [I_{m\times m} \quad 0_{m\times(n-m)}][I_{n\times n} \quad 0_{n\times(N-n)}]\sqrt{D}Q^\top \\
&= [I_{m\times m} \quad 0_{m\times(N-m)}]\sqrt{D}Q^\top \\
&= S[I_{m\times m} \quad 0_{m\times(N-m)}]Q^\top \\
&= SV^\top
\end{aligned}$$

which is identical to the solution from Theorem 1. □

The previous theorem shows that PCA without centering optimally preserves the inner products (in the sum-of-squares sense). Our goal is to interpret PCA/MDS *with* centering as finding the embedding Y that optimally preserves pairwise distances. In other words, we

will show that PCA with centering minimizes the functional,

$$F_{\text{PCA}}(X,Y) = ||L(E^X - E^Y)||_F^2 = \sum_{i,j=1}^{N} (||x_i - x_j||^2 - ||y_i - y_j||^2)^2 - 2(||x_i||^2 - ||y_i||^2)^2$$

where $E_{ij}^X = ||x_i - x_j||^2$ and $E_{ij}^Y = ||y_i - y_j||^2$ are the matrices of pairwise squared distances and L is the Laplacian weight matrix described below.

In order to understand why centered PCA minimizes F_{PCA}, we need to consider the following Laplacian matrix (also called the centering matrix),

$$L = D - A = \begin{bmatrix} \frac{N-1}{N} & \cdots & -\frac{1}{N} \\ \vdots & \ddots & \vdots \\ -\frac{1}{N} & \cdots & \frac{N-1}{N} \end{bmatrix} \tag{4.5}$$

where $A = \frac{1}{N}\vec{1}\vec{1}^\top$ so that $A_{ij} = 1/N$ is the adjacency matrix of an all-to-all weighted graph with equal graph weights and $D = I$ is the associated degree matrix since $D_{ii} = \sum_{j=1}^{N} A_{ij}$. We call L the centering matrix because it 'centers' vectors by subtracting their mean, namely,

$$L\vec{v} = D\vec{v} - A\vec{v} = \vec{v} - \left(\frac{1}{N}\vec{1}^\top \vec{v}\right)\vec{1} = \vec{v} - \left(\frac{1}{N}\sum_{i=1}^{N} \vec{v}_i\right)\vec{1}.$$

This means that $L\vec{1} = \vec{0}$ and L will yield zero when applied to any constant vector. Since L sends non-zero vectors to zero, it cannot be invertible, however, if $\vec{v}$ is orthogonal to the constant vector then $\vec{1}^\top \vec{v} = \vec{0}$ so, $L\vec{v} = \vec{v}$. Thus, L acts as the identity on the subspace orthogonal to the constant vector, and it is invertible on this subspace.

The importance of L is due to the connection between distances and inner products, namely,

$$||\vec{v} - \vec{u}||^2 = \langle \vec{v} - \vec{u}, \vec{v} - \vec{u} \rangle = ||\vec{v}||^2 + ||\vec{u}||^2 - 2\langle \vec{v}, \vec{u} \rangle .$$

The following lemma connects the Gram matrix to the matrix of squared pairwise distances using L through a formula/process known as *double centering*.

Lemma 1. *Let* $\{x_i\}_{i=1}^N$ *be centered, so* $\sum_{i=1}^N x_i = 0$*, with Gram matrix* $G_{ij} = x_i^\top x_j$ *and squared pairwise distance matrix* $E_{ij} = ||x_i - x_j||^2$ *then,*

$$G = -\frac{1}{2}LEL$$

Proof. Note that $E_{ij} = ||x_i||^2 + ||x_j||^2 - 2\langle x_i, x_j\rangle = G_{ii} + G_{jj} - 2G_{ij}$ so

$$E = \vec{g}\vec{1}^\top + \vec{1}\vec{g}^\top - 2G$$

where $\vec{g} = \text{diag}(G)$ so $\vec{g}_i = G_{ii}$. Thus,

$$LE = L\vec{g}\vec{1}^\top + L\vec{1}\vec{g}^\top - 2LG = L\vec{g}\vec{1}^\top - 2LG$$

since $L\vec{1} = \vec{0}$. Moreover, since the data is centered, each column of G is centered, $\sum_{i=1}^N G_{ij} = \left(\sum_{i=1}^N x_i\right)^\top x_j = \vec{0}$. Thus, $LG = G$ so we have $LE = L\vec{g}\vec{1}^\top - 2G$. Multiplying on the right by L we have,

$$LEL = L\vec{g}\vec{1}^\top L - 2GL = -2G$$

since $\vec{1}^\top L = \vec{0}^\top$ and $GL = G$ since the data is centered. Finally, $G = -\frac{1}{2}LEL$. □

Starting from a matrix E, the process of computing LEL is equivalent to subtracting the row average from each row, then subtracting the column average from each column, and then adding the total average of E to each entry. This process is called *double centering* and as shown above it recovers the Gram matrix from the matrix of pairwise distances when the underlying data is centered. Moreover, since the matrix of pairwise distances is invariant to translations, even when the original data set is not centered, the process of constructing

E and then double centering will construct the Gram matrix *of the centered data*. Thus, constructing the Gram matrix from the matrix of pairwise distances implicitly centers the data. This immediately yeilds the following characterization of PCA.

Theorem 2 (Principal Component Analysis (PCA))**.** *Let* $\{x_i\}_{i=1}^N \subset \mathbb{R}^n$ *have mean* $\mu = \frac{1}{N}\sum_{i=1}^N x_i$ *and* centered *covariance matrix* $C = XLX^\top = (X - \mu\vec{1}^\top)(X - \mu\vec{1}^\top)^\top = \sum_{i=1}^N (x_i - \mu)(x_i - \mu)^\top$ *with orthogonal eigendecomposition* $C = W\Lambda W^\top$ *with* $\Lambda_{11} \geq \cdots \geq \Lambda_{nn}$*. The centered point set* $\{y_i\}_{i=1}^N \subset \mathbb{R}^m$*, that minimizes* $F_{\mathrm{PCA}}(X, Y)$ *is given by the PCA projection,* $Y = [I_{m\times m} \quad 0_{m\times(n-m)}]W^\top(X - \mu\vec{1}^\top)$.

Proof. Let $\tilde{X} = XL = X - \mu\vec{1}^\top$ be the centered data and let $E_{ij} = ||x_i - x_j||^2 = ||\tilde{x}_i - \tilde{x}_j||^2$ and $E^Y_{ij} = ||y_i - y_j||^2$ be the matrices of pairwise distances. Thus, we can write $F_{\mathrm{PCA}}(X, Y) = ||L(E - E^Y)L||_F^2$ and since $\tilde{X}$ is centered, Corollary 1 says that Y minimizes,

$$
\begin{aligned}
F_{\mathrm{MDS}}(\tilde{X}, Y) = ||\tilde{X}^\top\tilde{X} - Y^\top Y||_F^2 &= \frac{1}{4}||LEL - LE^Y L||_F^2 = \frac{1}{4}||L(E - E^Y)L||_F^2 \\
&= \frac{1}{4}\sum_{k,l=1}^N \left(\sum_{i,j=1}^N L_{ki}(E_{ij} - E^Y_{ij})L_{jl}\right)^2 \\
&= \frac{1}{4}\sum_{k,l=1}^N \sum_{i,j,r,s=1}^N L_{ki}L_{kr}L_{jl}L_{sl}(E_{ij} - E^Y_{ij})(E_{rs} - E^Y_{rs}) \\
&= \frac{1}{4}\sum_{i,j,r,s=1}^N L_{ir}L_{js}(E_{ij} - E^Y_{ij})(E_{rs} - E^Y_{rs}) \\
&= \frac{1}{4}||L(E - E^Y)||_F^2
\end{aligned}
$$

and thus Y minimizes F_{PCA}. □

We can further simplify the expression for F_{PCA} as,

$$
\begin{aligned}
F_{\text{PCA}}(X,Y) &= ||L(E-E^Y)||_F^2 \\
&= \sum_{i,j,r,s=1}^{N} L_{ir}L_{js}(E_{ij}-E_{ij}^Y)(E_{rs}-E_{rs}^Y) \\
&= \sum_{i,j,r,s=1}^{N} (\delta_{ir}-1/N)(\delta_{js}-1/N)(E_{ij}-E_{ij}^Y)(E_{rs}-E_{rs}^Y) \\
&= \sum_{i,j,r,s=1}^{N} (\delta_{ir}\delta_{js}-\delta_{ir}/N-\delta_{js}/N+1/N^2)(E_{ij}-E_{ij}^Y)(E_{rs}-E_{rs}^Y) \\
&= ||E-E^Y||_F^2 - \frac{2}{N}\sum_{i,j,s=1}^{N}(E_{ij}-E_{ij}^Y)(E_{is}-E_{is}^Y) + \frac{1}{N^2}\left(\sum_{i,j=1}^{N} E_{ij}-E_{ij}^Y\right)^2.
\end{aligned}
$$

Now note that since the data is centered we have $\sum_{i=1}^N E_{ij} = \sum_{i=1}^N ||x_i||^2 + ||x_j||^2 - 2\langle x_i, x_j\rangle = N||x_j||^2 + \sum_{i=1}^N ||x_i||^2$ so setting $e_i = ||x_i||^2 - ||y_i||^2$ we have, $\sum_{i=1}^N E_{ij} - E_{ij}^Y = Ne_j + \sum_{i=1}^N e_i$. This allows us to write,

$$
\begin{aligned}
F_{\text{PCA}}(X,Y) &= ||E-E^Y||_F^2 - \frac{2}{N}\sum_{i=1}^{N}\left(Ne_i+\sum_{j=1}^{N}e_j\right)^2 + \frac{1}{N^2}\left(2N\sum_{i=1}^{N}e_i\right)^2 \\
&= ||E-E^Y||_F^2 - \frac{2}{N}\sum_{i=1}^{N}\left(N^2e_i^2 + Ne_i\sum_{j=1}^{N}e_j + \sum_{j,s=1}^{N}e_je_s\right) + 4\sum_{i,j=1}^{N}e_ie_j \\
&= ||E-E^Y||_F^2 - 2N\sum_{i=1}^{N}e_i^2 \\
&= \sum_{i,j=1}^{N}(||x_i-x_j||^2 - ||y_i-y_j||^2)^2 - 2(||x_i||^2-||y_i||^2)^2.
\end{aligned}
$$

4.3 Identifying PCA in terms of the framework

We can now proceed by identifying the constituent components of PCA within our framework. Recall that we defined our problem as finding dataset Y that will minimize the following expression

$$F(X,Y) = \tilde{F}(D^X, D^Y) = \sum_{i,j}^{N} \hat{F}\left(S^X\left(v(D_i^X), v(D_j^X)\right), S^Y\left(w(D_i^Y), w(D_j^Y)\right)\right)$$

Thus, we can fully represent PCA through its constituent components:

- $\hat{F}$ is the Euclidean distance, making $\tilde{F}$ the Frobenius norm
- $v(D_i^X) = L(D_i^X)^{\mathrm{T}}$ and $w(D_i^Y) = L(D_i^Y)^{\mathrm{T}}$
- $S^X(a,b) = a$ and $S^Y(a,b) = a$. These are arbitrarily chosen to be the first element since the kernels are already symmetric.

Chapter 5: Applying the Framework to tSNE

We begin by recognizing that tSNE does not cleanly fit into the framework. This can be clearly seen through the inconsistent normalization between the high- and low-dimensional spaces. Namely, tSNE normalizes the high-dimensional probabilities across rows of the matrix, whereas the low-dimensional kernels are normalized across the entire matrix. Thus, when they are minimizing the KL, the authors are actually comparing N high-dimensional distributions to a single low-dimensional one.

We, however, defined our framework with

$$F(X,Y) = \tilde{F}(D^X, D^Y) = \sum_{i,j}^{N} \hat{F}\left(S^X\left(v(D_i^X), v(D_j^X)\right), S^Y\left(w(D_i^Y), w(D_j^Y)\right)\right).$$

This requires that our kernels depend only on rows of the similarity matrices, disallowing a matrix-wise normalization factor. The reader may ask why the framework was defined in such a way so as to not generalize for tSNE's normalization. We note, however, that tSNE is a strict outlier in its treatment of the probability distributions. We have not found another algorithm that uses a similar asymmetric normalization pattern. Thus, rather than generalizing our framework to such an extent that it bears no information, we instead choose to identify tSNE as the outlier.

We proceed by a significantly more in-depth proof by contradiction. To do this, we will assume that tSNE does fit into the more specific framework, discuss what this would mean, and then show why it is not the case.

5.1 Formulating t-SNE in terms of Bernoulli random variables

The naive method for reparameterizing t-SNE in terms of Bernoulli random variables is to remove the scaling terms in t-SNE's kernel functions. Specifically, the high-dimensional t-SNE kernel is the Gaussian distribution

$$p_{j|i} = \frac{\exp(-||y_i - y_j||^2/2\sigma_i^2)}{\sum\limits_{\substack{k=1\ldots n \\ k\neq i}} \exp(-||y_i - y_k||^2/2\sigma_i^2)}, \text{ and}$$

$$p_{ij} = p_{ji} = \frac{p_{j|i} + p_{i|j}}{2},$$

where σ_i is chosen to provide a user-defined perplexity. To simplify the notation, we denote the row-sum denominator in the high-dimensional space by Z_i^X for row i. The normalization term in the denominator establishes that all of these $p_{j|i}$'s make up a probability distribution across all of point i's incoming edges. Removing this normalization term gives us Bernoulli random variables

$$E_{j|i}^X = \exp(-||y_i - y_j||^2/2\sigma_i^2),$$

where the likelihood of the edge not existing is simply $1 - E_{j|i}^X$.

We can do the same in the low-dimensional student-T distribution. Recall that t-SNE's low-dimensional kernel is given by

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum\limits_{k\neq l}(1 + ||y_k - y_l||^2)^{-1}} = \frac{w(D(y_i, y_j))}{Z^Y},$$

where $w(D^Y(y_i, y_j))$ is the low-dimensional kernel of the distance as defined in 3.1, which we refer to as w_{ij} going forward. However, notice that the normalization term occurring

here is dividing by $\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}$. This is now with respect to all pairs of non-identical points rather than just all of point i's incoming edges. As such, it is not sufficient to remove the normalization terms in t-SNE's high- and low-dimensional kernels blindly, as that would introduce extraneous scaling into the optimization. Going forward, we define this matrix-sum denominator in the low dimensional space by Z^Y.

To maintain the appropriate scaling between the high- and low-dimensional distributions, we will remove the scaling along the rows of both the high- and low-dimensional adjacency matrices. Due to the symmetry of the low-dimensional representation, however, we arbitrarily choose to remove the low-dimensional scaling across columns rather than rows. That way we will still have row-wise summations in the low-dimensional space and can maintain consistent notation. Therefore, our t-SNE Bernoulli random variables are defined in the high-dimensional space with the kernel

$$E^X_{j|i} = \exp(-||y_i - y_j||^2/2\sigma_i^2); \text{ with } p_{j|i} = \frac{E^X_{j|i}}{\sum\limits_{\substack{k=1\ldots n \\ k \neq i}} \exp(-||y_i - y_k||^2/2\sigma_i^2)} = \frac{E^X_{j|i}}{Z^X_i}. \tag{5.1}$$

We then refer to the high dimensional symmetrized kernel E^X_{ij} as the symmetrization function S applied to the two conditional kernels, so

$$E^X_{ij} = S(E^X_{j|i}, E^X_{i|j}) \tag{5.2}$$

Similarly, the low-dimensional ones are defined with the row-normalized kernel

$$E^Y_{j|i} = \frac{w_{ij}}{\sum\limits_{\substack{k=1\ldots n \\ k \neq i}} w_{ik}} = \frac{w_{ij}}{Z^Y_i}; \text{ with } q_{ij} = \frac{E^Y_{ij} \cdot Z^Y_i}{\sum_k Z^Y_k} \tag{5.3}$$

where we define the row-sum denominator in the low-dimensional space by Z^Y_i for row i.

Applying the general symmetrization function S, we have

$$E_{ij}^Y = S(E_{j|i}^Y, E_{i|j}^Y) \tag{5.4}$$

5.2 Establishing similarities in optimization criteria

We now aim to show that t-SNE's original optimization criterion is similar to optimizing the above Bernoulli random variable interpretation.

First, let us understand t-SNE's optimization formulation. They minimize the KL divergence between the high- and low-dimensional distributions. This amounts to minimizing the sum

$$\mathrm{KL}(P||Q) = \sum_{i \neq j} \left[p_{ij} \log \frac{p_{ij}}{q_{ij}} \right].$$

Since we are optimizing with respect to the low-dimensional points Q, this amounts to minimizing $-\sum_{i \neq j} [p_{ij} \log(q_{ij})]$.

We now aim to interpret the q_{ij} variables in terms of their Bernoulli random variables $E_{ij}^Y = \frac{(1 + ||y_i - y_j||^2)^{-1}}{Z_i^Y}$. First, recall that $q_{ij} = \frac{E_{ij}^Y \cdot Z_i^Y}{\sum_k Z_k^Y}$. Plugging this in gives us

$$\mathrm{KL}(P||Q) = -\sum_{i \neq j} \left[p_{ij} \log \left(\frac{E_{ij}^Y \cdot Z_i^Y}{\sum_l Z_l^Y} \right) \right].$$

We separate the log to obtain

$$\mathrm{KL}(P||Q) = -\sum_{i \neq j} \left[p_{ij} \left(\log(E_{ij}^Y) + \log(Z_i^Y) - \log(\sum_l Z_l^Y) \right) \right].$$

Looking at the elements, we see

1. $log(E_{ij}^Y)$ – the edge weight for the Laplacian as we defined it in the low-dimensional space

2. $log(Z_i^Y)$ – the row-wise normalization term of the low-dimensional kernel

3. $log(\sum_l Z_l^Y)$ – a sum over the low-dimensional adjacency matrix

Distributing appropriately gives us three minimization criteria of:

List of tSNE Forces 1.

1.
$$-\sum_{i \neq j} \left[p_{ij} \log(E_{ij}^Y) \right]$$

This seeks to maximize the likelihood of the edge existing in the low-dimensional space in accordance with how likely it is to exist in the high-dimensional space.

2.
$$-\sum_{i \neq j} \left[p_{ij} \log(Z_i^Y) \right]$$

This tries to maximize the normalization term for each low-dimensional edge.

3.
$$\sum_{i \neq j} \left[p_{ij} \log(\sum_l Z_l^Y) \right]$$

This is a weighted L1 penalty on the entire low-dimensional adjacency matrix.

Each of these is scaled by the high-dimensional probability that the two points are nearest neighbors.

We now look at the optimization problem from the Bernoulli interpretation and hope to arrive to a similar set of forces. Recall from equation 3.2 that the cross-entropy over Bernoulli random variables v and w is $H(v, w) = -v(x)log(w(x)) - (1 - v(x))log(1 - w(x))$.

The sum of cross entropies across all the edges, then, come out to

$$H(X,Y) = -\sum_{i \neq j} \left[-E_{ij}^X log(E_{ij}^Y) - (1 - E_{ij}^X) log(1 - E_{ij}^Y) \right]$$

The first term in the sum is exactly the first force that we just derived in (1), up to different constants. This makes sense, as we are optimizing the edge weights according to their Bernoulli kernels that don't have any scaling. To show equivalence it is therefore sufficient to show that the forces from $-(1 - E_{ij}^X) log(1 - E_{ij}^Y)$ are equivalent to the last two forces in (1). Looking at the gradients of this remaining term, we get

$$\nabla_{y_i} \sim -(1 - E_{ij}^X) \nabla_i log(1 - E_{ij}^Y) = -(1 - E_{ij}^X) \nabla_i log(1 - E_{ij}^Y),$$

where E_{ij}^X is the constant term $E_{ij}^X = S^Y(E_{j|i}^X, E_{i|j}^X)$ in the Bernoulli optimization criterion.

Handling the gradient of the log gives us the derivative in terms of E_{ij}^Y as

$$\nabla_{y_i} \sim \frac{1 - E_{ij}^X}{1 - E_{ij}^Y} \nabla_i E_{ij}^Y.$$

To look at the forces, let us decompose the E_{ij}^Y into its constituent w_{ij} functions. Recall that

$$E_{ij}^Y = S^Y(E_{j|i}^Y, E_{i|j}^Y) = S^Y(\frac{w_{ij}}{Z_i^Y}, \frac{w_{ij}}{Z_j^Y}) = S^Y \left(\frac{w_{ij}}{\sum\limits_{\substack{k=1...n \\ k \neq i}} w_{ik}}, \frac{w_{ij}}{\sum\limits_{\substack{k=1...n \\ k \neq j}} w_{jk}} \right)$$

Plugging this in gives us the gradient for point y as

$$\nabla_{y_i} \sim \frac{1 - E_{ij}^X}{1 - E_{ij}^Y} \nabla_i S^Y \left(\frac{w_{ij}}{Z_i^Y}, \frac{w_{ij}}{Z_j^Y} \right)$$

Because we are currently using the Bernoulli edge likelihood interpretation, we can plug in the symmetrization function $S(a,b) = a + b - ab$.

$$\nabla_{y_i} \sim \frac{1 - E_{ij}^X}{1 - E_{ij}^Y} \nabla_i \left(\frac{w_{ij}}{Z_i^Y} + \frac{w_{ij}}{Z_j^Y} - \frac{w_{ij}}{Z_i^Y} \cdot \frac{w_{ij}}{Z_j^Y} \right)$$

As a next step, we take a w_{ij} out from the numerators and apply the product rule to obtain

$$\nabla_{y_i} \sim \frac{1 - E_{ij}^X}{1 - E_{ij}^Y} \left[(\nabla_i(w_{ij})) \left(\frac{1}{Z_i^Y} + \frac{1}{Z_j^Y} - \frac{w_{ij}}{Z_i^Y \cdot Z_j^Y} \right) + w_{ij} \nabla_i \left(\frac{1}{Z_i^Y} + \frac{1}{Z_j^Y} - \frac{w_{ij}}{Z_i^Y \cdot Z_j^Y} \right) \right] \tag{5.5}$$

Looking at the second gradient term $w_{ij} \nabla_i \left(\frac{1}{Z_i^Y} + \frac{1}{Z_j^Y} - \frac{w_{ij}}{Z_i^Y \cdot Z_j^Y} \right)$, we can distribute the gradient and apply the quotient and product rules to obtain

$$w_{ij} \left(\frac{\nabla_i \left(Z_i^Y \right)}{(Z_i^Y)^2} + \frac{\nabla_i \left(Z_j^Y \right)}{(Z_j^Y)^2} - \frac{\nabla_i(w_{ij}) \cdot \left(Z_i^Y \cdot Z_j^Y \right) + w_{ij} \cdot \left[\nabla_i \left(Z_i^Y \right) \cdot Z_j^Y + Z_i^Y \cdot \nabla_i \left(Z_j^Y \right) \right]}{\left(Z_i^Y \cdot Z_j^Y \right)^2} \right)$$

We plug this back into 5.5 and rearrange by the like terms $\nabla_i(w_{ij})$ and w_{ij} to get:

$$\begin{aligned} \nabla_{y_i} \sim & \frac{1 - E_{ij}^X}{1 - E_{ij}^Y} \left[(\nabla_i(w_{ij})) \left(\frac{1}{Z_i^Y} + \frac{1}{Z_j^Y} - \frac{w_{ij}}{Z_i^Y \cdot Z_j^Y} - \frac{w_{ij} Z_i^Y Z_j^Y}{\left(Z_i^Y Z_j^Y \right)^2} \right) \right] \\ & + \frac{1 - E_{ij}^X}{1 - E_{ij}^Y} \left[w_{ij} \left(\frac{\nabla_i \left(Z_i^Y \right)}{(Z_i^Y)^2} + \frac{\nabla_i \left(Z_j^Y \right)}{(Z_j^Y)^2} - \frac{w_{ij} \cdot \left[\nabla_i \left(Z_i^Y \right) \cdot Z_j^Y + Z_i^Y \cdot \nabla_i \left(Z_j^Y \right) \right]}{\left(Z_i^Y \cdot Z_j^Y \right)^2} \right) \right] \end{aligned}$$

We now notice that $\nabla_i Z_j^Y$ is 0 for every element of the sum except for at $k = i$, so

$\nabla_i Z_j^Y = \nabla_i w_{ji} = \nabla_i w_{ij}$. We plug this in,

$$\nabla_{y_i} \sim \frac{1-E_{ij}^X}{1-E_{ij}^Y}\left[(\nabla_i(w_{ij}))\left(\frac{1}{Z_i^Y}+\frac{1}{Z_j^Y}-\frac{2w_{ij}Z_i^Y Z_j^Y}{\left(Z_i^Y\cdot Z_j^Y\right)^2}\right)\right]$$

$$+\frac{1-E_{ij}^X}{1-E_{ij}^Y}\left[w_{ij}\left(\frac{\nabla_i\left(Z_i^Y\right)}{(Z_i^Y)^2}+\frac{\nabla_i w_{ij}}{(Z_j^Y)^2}-\frac{w_{ij}\cdot\nabla_i\left(Z_i^Y\right)\cdot Z_j^Y}{\left(Z_i^Y\cdot Z_j^Y\right)^2}-\frac{w_{ij}Z_i^Y\cdot\nabla_i w_{ij}}{\left(Z_i^Y\cdot Z_j^Y\right)^2}\right)\right]$$

and distribute accordingly:

$$\nabla_{y_i} \sim \frac{1-E_{ij}^X}{1-E_{ij}^Y}\left[(\nabla_i(w_{ij}))\left(\frac{1}{Z_i^Y}+\frac{1}{Z_j^Y}+\frac{w_{ij}}{(Z_j^Y)^2}-\frac{2w_{ij}Z_i^Y Z_j^Y-w_{ij}Z_i^Y}{\left(Z_i^Y\cdot Z_j^Y\right)^2}\right)\right]$$

$$+\frac{1-E_{ij}^X}{1-E_{ij}^Y}\left[w_{ij}\nabla_i\left(Z_i^Y\right)\left(\frac{1}{(Z_i^Y)^2}-\frac{w_{ij}Z_j^Y}{\left(Z_i^Y\cdot Z_j^Y\right)^2}\right)\right].$$

We lastly write out $\nabla_i Z_i^Y$ in terms of the constituent w_{ik} components to obtain:

$$\nabla_{y_i} \sim \frac{1-E_{ij}^X}{1-E_{ij}^Y}\left[(\nabla_i(w_{ij}))\left(\frac{1}{Z_i^Y}+\frac{1}{Z_j^Y}+\frac{w_{ij}}{(Z_j^Y)^2}-\frac{2w_{ij}Z_i^Y Z_j^Y-w_{ij}Z_i^Y}{\left(Z_i^Y\cdot Z_j^Y\right)^2}\right)\right]$$

$$+\frac{1-E_{ij}^X}{1-E_{ij}^Y}\left[w_{ij}\cdot\nabla_i\left[\sum_{\substack{k=1\ldots n\\ k\neq j}} w_{ik}\right]\left(\frac{1}{(Z_i^Y)^2}-\frac{w_{ij}Z_j^Y}{\left(Z_i^Y\cdot Z_j^Y\right)^2}\right)\right];$$

and pull the gradient terms out to the front so that we may examine their scalings.

$$\nabla_{y_i} \sim \nabla_i(w_{ij}) \left[\frac{1 - E_{ij}^X}{1 - E_{ij}^Y} \left(\frac{1}{Z_i^Y} + \frac{1}{Z_j^Y} + \frac{w_{ij}}{(Z_j^Y)^2} - \frac{2w_{ij} Z_i^Y Z_j^Y - w_{ij} Z_i^Y}{\left(Z_i^Y \cdot Z_j^Y \right)^2} \right) \right]$$

$$+ \nabla_i Z_i^Y \left[\frac{1 - E_{ij}^X}{1 - E_{ij}^Y} w_{ij} \cdot \left(\frac{1}{(Z_i^Y)^2} - \frac{w_{ij} Z_j^Y}{\left(Z_i^Y \cdot Z_j^Y \right)^2} \right) \right].$$

Again, recall that we are trying to show that this gradient would be equivalent to the one obtained from the second two terms of the list of tSNE forces in (1).

There we have

$$\nabla_{y_i} \sim p_{ij} \nabla_i \left[\log(\sum_{l=1 \ldots n} Z_l^Y) - \log(Z_i^Y) \right].$$

Recall that $Z_i^Y = \sum_{\substack{k=1 \ldots n \\ k \neq i}} w_{ik}$. This gives us a double summation in the first log, so we write $\sum_{k \neq l}$ to mean the sum over all non-diagonal elements for legibility purposes. Plugging this in and applying the chain rule to the logarithms gives us

$$\nabla_{y_i} \sim p_{ij} \left[\frac{\nabla_i \left[\sum_{k \neq l} w_{kl} \right]}{\sum_{k \neq l} w_{kl}} - \frac{\nabla_i \left[\sum_{\substack{k=1 \ldots n \\ k \neq i}} w_{ik} \right]}{\sum_{\substack{k=1 \ldots n \\ k \neq i}} w_{ik}} \right].$$

Note, however, that the gradient with respect to y_i of $\sum_{k \neq l} w_{kl}$ is 0 when $k \neq i$ and $l \neq i$. Thus we can remove all of those elements from the sums. This gives us $2 \sum_{\substack{k=1 \ldots n \\ k \neq i}} w_{ik}$ on the left, as there is one row with $k = i$ and one column with $l = i$. We distribute the

numerators out to get:

$$\nabla_{y_i} \sim p_{ij} \nabla_i Z_i^Y \left[\frac{2}{\sum_{k \neq l} w_{kl}} - \frac{1}{\sum\limits_{\substack{k=1..n \\ k \neq i}} w_{ik}} \right].$$

We can see that these are clearly not equivalent, as tSNE's original gradient is with respect to Z_i^Y, whereas the Bernoulli interpretation gives us gradients with respect to both Z_i^Y and w_{ij}.

Chapter 6: Incorporating UMAP

The framework took large inspiration from UMAP's approach to dimensionality reduction. As such, UMAP fits very nicely into the configuration of kernel functions. Specifically, we use UMAP's definitions of the kernel functions

$$v_{j|i} = exp[(-d(x_i, x_j) - \rho_i)/\sigma_i]; v_{ij} = v_{j|i} + v_{i|j} - v_{j|i}v_{i|j}$$

$$w_{ij} = (1 + a||y_i - y_j||_2^{2b})^{-1}$$

, where v_{ij} is the high-dimensional likelihood of a nearest-neighbor relationship and w_{ij} is the low-dimensional counterpart. Recall that ρ_i is the distance from x_i to its nearest neighbor, and that σ_i is the result of a binary search to obtain a user-defined perplexity value.

Then $S^X(a, b) = a + b - ab$ and $S^Y(a, b)$ can be any linear interpolation of the two, as the low-dimensional kernel is already symmetric. We lastly define $\hat{F}$ as the KL divergence between the Bernoulli distributions E_{ij}^X and E_{ij}^Y.

This concludes the definition of UMAP in terms of the framework.

Chapter 7: Conclusion and Future Work

We have attempted to show that there is a cohesive framework that can unite seemingly separate dimensionality reduction algorithms. By relying on the fact that every such algorithm defines functionals on the distances in high- and low-dimensional space, which can be abstracted away. Despite our attempts to generalize across all of the identified algorithms, we recognize that tSNE performs an abnormal set of steps that are too different to be captured in any useful framework. Nonetheless, we identified UMAP and PCA in terms of our framework and gave in-depth analysis of what it means for tSNE to not fit into it.

There are multiple next steps that we have identified, which we devote several small sections of this chapter to.

7.1 Computational Analysis

Thus far, the work we have done has been purely theoretical. It remains to show that defining the framework computationally indeed successfully recreates the investigated algorithms. There are further discrepancies in the optimization schemas between the algorithms that would need to be addressed. Namely, PCA is optimized through a simple set of linear operations while tSNE and UMAP perform gradient descent. Furthermore, tSNE's and UMAP's gradient descent approaches differ in how they choose to sparsify the search space. In [10], the authors mention that they speed up tSNE's gradient descent through tree-based distance calculations to identify nearest neighbors to perform the search on. Once these nearest neighbors have been identified, tSNE sums their respective contributions across the entire dataset and applies a single step of gradient descent. While UMAP does a similar operation with respect to the nearest neighbors, it additionally performs sampling over those nearby points to further sparsify the attractive and repulsive forces.

If we were to unite these under a single computational framework, we would be interested in understanding the tradeoffs when applying gradient descent to PCA's optimization criterion. The authors in [11] showed that PCA, given some assumptions on the initialization, is guaranteed to converge with stochastic gradient descent. We hope to verify this computationally in the context of our framework.

7.2 Incorporating Additional Algorithms

We have additionally only discussed three dimensionality reduction algorithms. As hinted at in the introduction, there are hundreds, if not thousands, of unique approaches. While it is infeasible to apply the framework to all of them, we do hope to incorporate the most famous dimensionality reduction methods such as Isomap, Laplacian eigenmaps, and LargeVis [12] to name a few.

We can leverage the work done in [6] for the graph-based Isomap and Laplacian eigenmaps. There, the authors defined optimization criteria through graph Laplacians. However, it would remain to show theoretical guarantees on convergence of gradient descent for the above approaches.

LargeVis, on the other hand, draws inspiration from tSNE. However, it modifies the low-dimensional kernel to utilize the Bernoulli interpretation that we have already been using. Therefore, it is a perfect fit as a substitute for tSNE within our framework, as its kernel functions should nominally be very similar to those for UMAP. Furthermore, LargeVis already uses gradient descent for its optimization.

7.3 Ablation Studies and Searching the Space of Algorithms

Once the above algorithms have been incorporated into the framework, it would be interesting to perform ablation studies in which parameters are modified along the spectrum between algorithms. It is interesting to consider approaches that choose one algorithm's high-dimensional kernel and another's low-dimensional kernel.

We also point out that we have, in some sense, defined an invertible mapping between the spaces of kernels and the spaces of algorithms. As such, it should be possible to identify a search criterion over the space of kernels themselves, which would correspond to a search over dimensionality reduction algorithms. Given some metric defining the quality of a dimensionality reduction, we posit that it would be possible to identify new kernel functions that are particularly effective for specific datasets. This would have implications for clustering analysis as well, as dimensionality reduction algorithms are often intimately linked to clustering approaches.

Bibliography

[1] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[2] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[3] M. Budninskiy, G. Yin, L. Feng, Y. Tong, and M. Desbrun, "Parallel transport unfolding: a connection-based manifold learning approach," *arXiv preprint arXiv:1806.09039*, 2018.

[4] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

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

[6] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 1, pp. 40–51, 2006.

[7] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *International conference on database theory*. Springer, 2001, pp. 420–434.

[8] J. I. Marden, *Analyzing and modeling rank data*. CRC Press, 1996.

[9] J. B. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, no. 1, pp. 1–27, 1964.

[10] L. Van Der Maaten, "Accelerating t-sne using tree-based algorithms," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3221–3245, 2014.

[11] O. Shamir, "Convergence of stochastic gradient descent for pca," in *International Conference on Machine Learning*. PMLR, 2016, pp. 257–265.

[12] J. Tang, J. Liu, M. Zhang, and Q. Mei, "Visualizing large-scale and high-dimensional data," in *Proceedings of the 25th international conference on world wide web*, 2016, pp. 287–297.

Curriculum Vitae

Experience

MACHINE LEARNING ENGINEER AT EXPEDITION TECHNOLOGY — Dulles, VA — July 2017 - Present

1. Created digital signal classifier for a several terabyte dataset with an emphasis on identifying out-of-distribution examples
2. Incorporated complex-valued network layers to extract signal information appropriately
3. Utilized advanced clustering techniques to explore anomaly detection and open-set recognition techniques
4. Developed state of the art 3D object-detection system on large point clouds with AWS parallelization
5. Protoyped GAN for placing objects of interest into LIDAR scenes that blend in with the background environment

Education

MASTER OF SCIENCE — AUGUST 2019 - MAY 2021 — GEORGE MASON UNIVERSITY

Department: Mathematics

Related Coursework: Dimensionality Reduction; Linear Analysis; Fourier Analysis; Ordinary Differential Equations; Numerical Linear Algebra; Algebra; Numerical Analysis; Topology (Spring 2021); Two semesters of master's thesis;

BACHELOR OF ARTS — AUGUST 2013 - MAY 2017 — UNIVERSITY OF VIRGINIA

Departments: Double major in mathematics and computer science

Related Coursework: Calculus I, II, III; Advanced Calculus; Real Analysis; Survey of Algebra; Discrete Mathematics; Linear Algebra; Intro to Statistics; Mathematical Statistics; Software Development Methods; Human Computer Interaction; Algorithms; Program and Data Representation; Artificial Intelligence; Machine Learning; Database Systems; Computer Architecture;

Teaching:

Taught a course on neural networks that focused on understanding current literature and applications Was an undergraduate teaching assistant for the Program and Data Representations and Algorithms courses

GED — THOMAS JEFFERSON HIGH SCHOOL FOR SCIENCE AND TECHNOLOGY — Graduated in 2013

Publications

Open-set Recognition Through Unsupervised and Class-Distance Learning - Andrew Draganov, Carter Brown, Enrico Mattei, Cass Dalton, and Jaspreet Ranjit. 2020. In Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning (WiseML '20).

Feature Learning for Enhanced Security in the Internet of Things - Enrico Mattei, Cass Dalton, Andrew Draganov, Brent Marin, Michael Tinston, Greg Harrison, Bob Smarrelli, and Marc Harlacher. 2019. In IEEE Global Conference on Signal and Information Processing.