Inaugural-Dissertation

zur

Erlangung der Doktorwürde

der

Gesamtfakultät für Mathematik, Ingenieur- und Naturwissenschaften

der

Ruprecht-Karls-Universität
Heidelberg

vorgelegt von

Sebastian Damrich, MASt, M.Sc.
aus Bad Soden am Taunus, Deutschland

Tag der mündlichen Prüfung:

DISCOVERING STRUCTURE WITHOUT LABELS

Advisors: Prof. Dr. Fred A. Hamprecht
Prof. Dr. Christoph Schnörr

ABSTRACT

The scarcity of labels combined with an abundance of data makes unsupervised learning more attractive than ever. Without annotations, inductive biases must guide the identification of the most salient structure in the data. This thesis contributes to two aspects of unsupervised learning: clustering and dimensionality reduction.

The thesis falls into two parts. In the first part, we introduce Mod Shift, a clustering method for point data that uses a distance-based notion of attraction and repulsion to determine the number of clusters and the assignment of points to clusters. It iteratively moves points towards crisp clusters like Mean Shift but also has close ties to the Multicut problem via its loss function. As a result, it connects signed graph partitioning to clustering in Euclidean space.

The second part treats dimensionality reduction and, in particular, the prominent neighbor embedding methods UMAP and *t*-SNE. We analyze the details of UMAP's implementation and find its actual loss function. It differs drastically from the one usually stated. This discrepancy allows us to explain some typical artifacts in UMAP plots, such as the dataset size-dependent tendency to produce overly crisp substructures. Contrary to existing belief, we find that UMAP's high-dimensional similarities are not critical to its success.

Based on UMAP's actual loss, we describe its precise connection to the other state-of-the-art visualization method, *t*-SNE. The key insight is a new, exact relation between the contrastive loss functions negative sampling, employed by UMAP, and noise-contrastive estimation, which has been used to approximate *t*-SNE. As a result, we explain that UMAP embeddings appear more compact than *t*-SNE plots due to increased attraction between neighbors. Varying the attraction strength further, we obtain a spectrum of neighbor embedding methods, encompassing both UMAP- and *t*-SNE-like versions as special cases. Moving from more attraction to more repulsion shifts the focus of the embedding from continuous, global to more discrete and local structure of the data. Finally, we emphasize the link between contrastive neighbor embeddings and self-supervised contrastive learning. We show that different flavors of contrastive losses can work for both of them with few noise samples.

ZUSAMMENFASSUNG

Die Menge an verfügbaren Daten steigt unaufhörlich, aber annotierte Daten bleiben weiterhin selten. Das macht Methoden attraktiv, welche die Struktur von Daten ohne Trainingsbeispiele lernen können. Diese Arbeit leistet Beiträge zu zwei solchen Bereichen: Clusteranalyse und Dimensionsreduktion.

Die Arbeit besteht aus zwei Teilen. Im ersten Teil führen wir die Methode Mod Shift ein. Sie gruppiert Punkte im euklidischen Raum basierend auf deren Distanz. Nahe Punkte ziehen sich an und entfernte Punkte stoßen sich ab, bis sich klare Gruppen bilden. Die Balance beider Kräfte bestimmt die Anzahl der gefundenen Cluster. Wie Mean Shift bewegt auch Mod Shift die Punkte schrittweise hin zu kompakten Gruppen. Seine Zielfunktion ist jedoch vom Multicut Problem inspiriert, sodass wir eine Verbindung zwischen Graphpartitionierung und Punktgruppierung herstellen.

Der zweite Teil behandelt Dimensionsreduktion mit den populären Nachbareinbettungsmethoden UMAP und *t*-SNE. Wir zeigen auf, dass sich die Zielfunktion, die von UMAPs Implementierung tatsächlich optimiert wird, deutlich von der bisher akzeptierten unterscheidet. Diese Diskrepanz erklärt einige Artefakte in UMAP-Einbettungen, wie die Tendenz besonders für große Datensätze Einbettungen mit linienartigen Bereichen zu produzieren. Im Gegesatz zum bisherigen Verständnis von UMAP ist dessen Ähnlichkeitsmaß zwischen Punkten im hochdimensionalen Raum nicht maßgebend für die Einbettungsqualität.

Basierend auf UMAPs tatsächlicher Zielfunktion beschreiben wir die genaue Verbindung zur Visualisierungsmethode *t*-SNE. Der Kernpunkt ist unser neues Verständnis wie sich die beiden kontrastiven Lernmethoden negative sampling, verwendet von UMAP, und noise-contrastive estimation, mit der *t*-SNE approximiert werden kann, zu einander verhalten. Auf Basis dieser Verbindung finden wir heraus, dass UMAP mehr Anziehung auf benachbarte Datenpunkte ausübt als *t*-SNE und deswegen kompaktere Einbettungen produziert. Andere Attraktionsniveaus sind ebenso möglich und führen zu einem Kontinuum von Nachbareinbettungsmethoden, das UMAP- und *t*-SNE-artige Verfahren als Spezialfälle umfasst. Höhere Anziehung stellt die kontinuierlichen und globalen Aspekte eines Datensatzen besser heraus, während stärkere Repulsion die lokale Struktur von diskreten Clustern genauer darstellt. Abschließend weisen wir auf die Verbindung von kontrastiven Nachbareinbettungsmethoden und kontrastivem selbstüberwachtem Lernen hin und zeigen, dass beide mit den selben Zielfunktionen und nur wenigen Kontrastbeispielen optimiert werden können.

ACKNOWLEDGMENTS

This thesis would not have come about without a number of people. Major thanks are due to my supervisor, Fred Hamprecht. He introduced me to the fascinating realm of machine learning and created a productive and friendly work environment. I appreciate his broad range of interests and expertise that allowed me to work on a variety of projects and receive valuable feedback. Our discussions were always helpful and thought-provoking. I am grateful for his mentorship, much freedom in choosing research topics, and the opportunity to collaborate with many other students.

Barbara Werner was a reliable counselor and proactive supporter in all bureaucratic matters. Without her smooth handling of administrative issues, I would have had less time for research. Owen Vincent kept our IT running flawlessly, even beyond his time in Heidelberg, and was a great lunch partner.

I am glad and grateful for having been a part of the Image Analysis and Learning group over the past four years. We had countless stimulating discussions about science and everything else, and lots of cake. In particular, I thank my office mate Nasim Rahaman for guiding my plunge into machine learning, conversations about its current trends, and coding advice. Similarly, while sharing an office with Kalyan Ram Ayyalasomayajula, I received many valuable pointers on software development. I much enjoyed the collaborations with Enrique Fita Sanmartín on graph algorithms as well as the projects with Quentin Garrido, Florin Walter, Alexander Jäger, Daria Damm, and Philipp Nazari. Alberto Bailoni, Roman Remme, and Lorenzo Cerrone were always available for bouncing ideas back and forth, for coding support, a relaxed chat, and much other help. Lorenzo deserves extra thanks for breaks with delicious espresso and for proofreading parts of this thesis. I am grateful to Roman for his help with the Mod Shift project. Furthermore, I thank Ocima Kamboj for insightful discussions on single-cell topics, Manuel Haußmann for all-too-relatable Ph.D. comics and for tirelessly explaining variational inference, and Peter Lippmann for fun brain-storming sessions.

Philipp Berens, Dmitry Kobak, and Fred Hamprecht gave me the fantastic opportunity to spend a research stay in the Data Science for Vision Research group in Tübingen. The collaboration with Dmitry and Jan-Niklas Böhm distinctly shaped the fourth chapter of this thesis. Dmitry's detailed feedback and Nik's code bases proved invaluable.

Many thanks go to Christoph Schnörr for enabling me to pursue an interdisciplinary Ph.D. in computer science and mathematics. The STRUCTURES cluster hosted several fascinating events at the intersection of machine learning and mathematics and fostered interaction with other groups at the university. I am thankful for having profited from this stimulating environment.

Special thanks go to my parents, Sabine and Bernhard Damrich. I am grateful for their unconditional, continuous support during my Ph.D. and throughout my life. They paved the way to where I am today and sustained me along it. I also thank my sister, Christine Damrich, for always being there for me.

Finally, huge thanks go to Nadja-Mira Yolcu for her resourceful advice and careful proofreading of parts of this thesis. But most of all for her love, and unwavering belief in me; for keeping me afloat, and enriching my life so very much.

CONTENTS

1 INTRODUCTION

While data is becoming increasingly abundant, it rarely comes with labels, especially not in the sciences. As a result, annotated data remains a scarce resource in many domains. Unsupervised learning aims to find structure in data without annotations or labels.

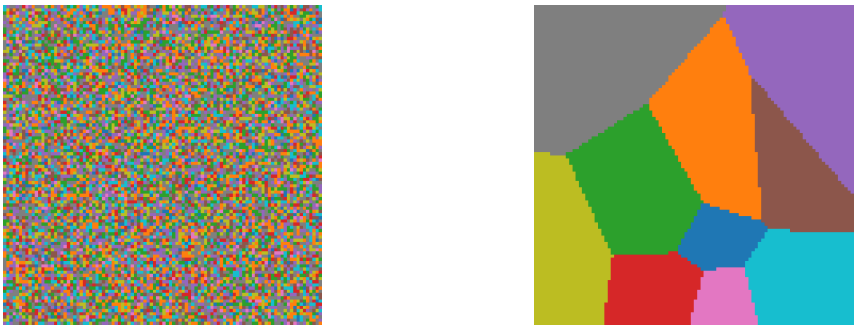
Annotating scientific data is, at best, a tedious, dull, and costly task, which often can only be completed by a domain expert. At worst, it is infeasible by the very nature of the addressed question: In an exploratory context, the researcher wants to understand the new dataset for the first time and hence does not have any labels yet. In these situations, unsupervised learning shines.

However, there is no free lunch, and in the absence of a clear learning signal derived from annotated samples, some form of inductive bias or a weak form of supervision is still required, e.g., the number of clusters or neighbors. Not all assumptions built into an unsupervised method will align with the properties of all datasets, and the right method needs to be employed for a specific dataset and task of interest. We will discuss examples of this in Chapters 2 and 4.

Therefore, learning with little supervision is difficult. However, humans are an excellent example for its feasibility. The understanding and mimicking of human cognition inspired many advances in machine learning. The current working horse of machine learning, the artificial neural network, was inspired by the natural neural network in our brains [110, 136]. Several prominent variants build on more specific aspects of human cognition. For example, the convolutional neural network imitates parts of our visual apparatus [55], the transformer architecture [162] capitalizes on the idea of attention [103] and recent efforts [13, 60] towards modeling higher-level cognitive capabilities, such as reasoning, draw inspiration from Kahneman's notion of 'system 1' and 'system 2' [73]. In this spirit, it is remarkable that humans learn to a large extent with very little supervision. While we are generally aware of supervised learning, for instance, in school, much of our learning comes from simply observing and interacting with our environment. The extent of unsupervised learning is particularly evident in the development of infants. At six months, infants already appear to have developed some object permanence, seem to be able to count small sets, show signs of a first intuitive understanding of gravity, and behave as if they can visually distinguish coarse object classes [128]. However, it is only at that age that they show the first indication of language comprehension [17]. So, at least much of our very early learning takes place with only little supervision, making unsupervised learning a promising research direction [92].

Two common forms of unsupervised learning and exploratory data analysis are clustering and dimensionality reduction. Both are instances of representation learning. A data point is either represented by its cluster or by a lower-dimensional point. This work makes contributions to clustering in Chapter 2 and to dimensionality reduction in Chapters 3 and 4.

It is often difficult to measure the quality of an unsupervised method's output. One option for vetting a method is applying it to data for which labels do exist but



(a) Random partition

(b) Partition with spatially connected elements

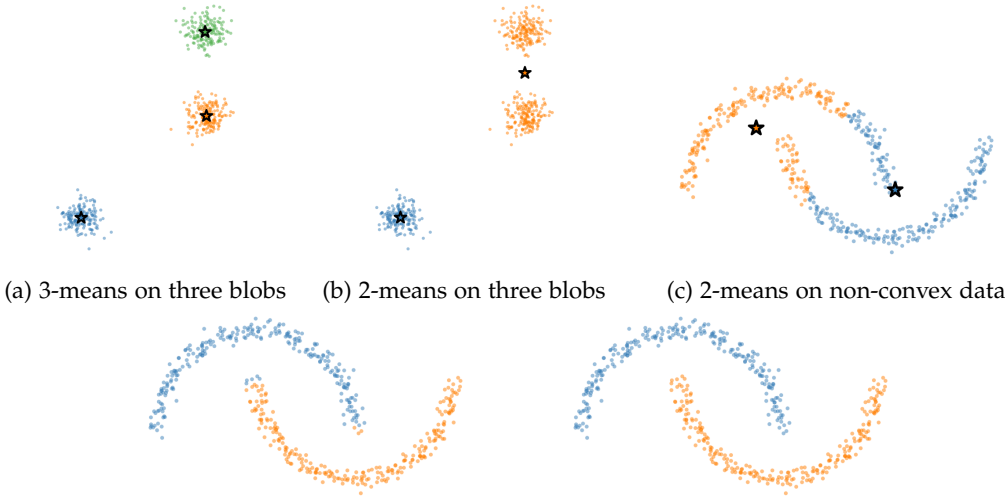
Figure 1.1: Two partitions of the pixels in an image. The elements have the same size distribution in both partitions. In 1.1a, the pixels are randomly assigned to partition elements, while they form Voronoï cells in 1.1b, so that they are convex and, in particular, spatially connected. Some degree of such a spatial prior is usually desirable when looking for partitions, thus excluding a vast part of the set of all possible partitions.

are not used in the unsupervised stage. For instance, we use available cluster information to validate our clustering method Mod Shift in Sec. 2.7 and existing labels to better understand our dimensionality reduction results in Chapters 3 and 4. There are also more intrinsic measures for clustering, such as the silhouette score [138], and for dimensionality reduction, e.g., kNN-recall [83, 94], but these typically only assess a particular aspect of the learned representation. In an exploratory context, the main aim is to understand the dataset better and develop research hypotheses. Then, these need to be independently verified, e.g., with targeted experiments [83]. So the ultimate raison d'être for many unsupervised learning methods is their ability to suggest structure relevant to the human investigators. Unfortunately, this application-dependence is very hard to quantify. A chemist cares about clusters of atoms; an astrophysicist might look out for clusters of galaxies. Biologists are interested both in discrete cell types and how stem cells continuously differentiate into them. In Chapter 4 we discuss a trade-off between visualizing continuous and discrete structure.

1.1 CLUSTERING

Clustering is the task of finding meaningful groups in a dataset. It is one of the oldest machine learning problems [104, 109], one of the central tools in exploratory data analysis [106] and often part of more involved machine learning pipelines [22, 86, 129].

Coarsely summarizing a dataset by its salient grouping is such a straightforward idea that one can easily overlook the hidden combinatorial complexity. The number of possible partitions, or groupings or clusterings, of a dataset with n elements equals the n-th Bell number $B(n)$ [12], which grows exponentially $B(2n) > n^n$, see Proof of Prop. 10 in [142]. There are more ways of clustering a set with 50 elements than there are atoms in the observable universe, as estimated by Eddington [168]. Hence, brute-force search for a clustering is not an option. Fortunately, many



(a) 3-means on three blobs (b) 2-means on three blobs (c) 2-means on non-convex data

(d) Mean Shift on non-convex data (e) DBSCAN on non-convex data

Figure 1.2: 1.2a–1.2c show that the assumptions of k-means clustering need to fit the dataset to produce good results. The stars are the learned cluster centers. If clusters are spherical, k-means can identify them correctly only if k is chosen properly (1.2a, b). Geared towards spherical clusters, it struggles on non-convex clusters (1.2c). On such datasets, density-based methods such as Mean Shift (1.2d) or DBSCAN (1.2e) can perform better for a suitable choice of hyperparameters.

partitions are undesirable. For example, one is usually interested in clusters that are in some sense spatially connected and not fragmented, as in Fig. 1.1.

A good clustering algorithm builds on inductive biases and the dataset's structure to only search among a limited set of partitions or directly construct a single one, which captures the most salient aspects of the dataset. The assumptions on the dataset structure need to match up with the inductive biases of the objective function and the optimization method. Consider for instance the classical k-means clustering, in which a dataset $x_1, \ldots, x_n \in \mathbb{R}^d$ of n different points shall be clustered into k clusters by minimizing the sum of squared distances to the closest of a set of k cluster centroids $\mu_1, \ldots, \mu_k \in \mathbb{R}^d$

$$\operatorname*{argmin}_{\mu_1,\ldots,\mu_k} \sum_{i=1}^{n} \|x_i - \mu_{c(i)}\|^2, \text{ where } c(i) = \operatorname*{argmin}_{j=1,\ldots,k} \|x_i - \mu_j\|. \tag{1.1}$$

This intuitive objective function makes at least two assumptions. First and most obviously, the method will find exactly k clusters no matter the data, as there are only k centroids, and it is always beneficial to place at least one point into each cluster. This can be seen as a strong inductive bias or even a weak form of supervision. Second, since a point is always assigned to its nearest centroid, the points forming a cluster must lie in their centroid's Voronoï region. In particular, they will be roughly spherically shaped. Non-convex clusters are thus impossible for k-means clustering. These assumptions work well for certain datasets, but fail for others Fig. 1.2a-c.

Another popular inductive bias is identifying clusters as high-density data regions. This allows for non-convex clusters and can determine the number of clusters internally so that it does not have to be specified by the user. There will be as many clusters as there are high-density regions. Examples of density-based clustering

algorithms are (H)DBSCAN [50, 111] and Mean Shift [31, 34]. In (H)DBSCAN points form a cluster if they lie in the same high-density region, where a point is a high-density point if sufficiently many points lie nearby. Mean Shift constructs a kernel density estimate (KDE) and moves points via gradient ascent to the modes of this KDE. Both can handle non-convex clusters well, see Fig. 1.2d-e but still require some weak supervision to determine their notion of high density. For Mean Shift, this is the choice of kernel and bandwidth used to construct the kernel density estimate of the data, and for HDBSCAN, it is the number of neighbors that need to lie in close vicinity.

So far, we have discussed clustering point clouds where the similarity cues came from the points' spatial configuration. Graphs, however, directly encode relational information in the edges. Vertices connected by an edge are often deemed similar, and the lack of an edge connection signifies dissimilarity. Weighing the graph edges gives a finer notion of similarity. Lacking a spatial notion of density, one might try to cluster or partition the graph into densely connected node sets. This is the aim of modularity clustering [119]. More precisely, modularity clustering identifies dense regions as those more densely connected than they would be in a random graph with the same node degrees. The loss function is

$$\underset{P \text{ a node partition}}{\operatorname{argmax}} \sum_{i<j} \left(w_{ij} - \frac{d_i d_j}{2w_{tot}} \right) \cdot \mathbb{1}(i \sim_P j), \tag{1.2}$$

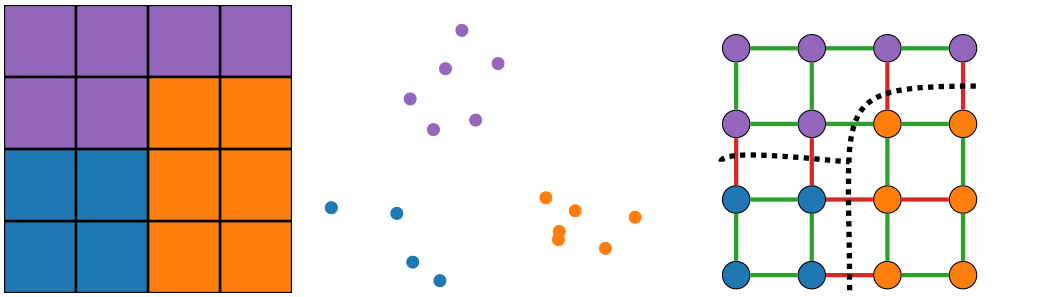
where w_{ij} are the node weights, $d_i = \sum_j w_{ij}$ is the node degree of i, $w_{tot} = \frac{1}{2}\sum_i d_i$ the total weight of the graph, P a node partition and $\mathbb{1}(i \sim_P j)$ is the indicator function for whether nodes i and j are in the same partition element of P. While not apparent from this density maximization formulation of the loss, Davis and Sethuraman [43] recently showed that the comparison to the random graph model induces a hidden inductive bias towards equisized clusters.

When the graph edges w_{ij} are allowed to be signed, that is, have an either positive or negative sign, we are in an even more general setting for graph partitioning. Positive edges still indicate similarity. However, dissimilarity is not just indirectly represented as a missing edge but explicitly as an edge of negative weight. A very natural strategy is to seek the partition that maximizes the sum of edge weights within partition elements and to minimize the sum of edge weights between different partition elements. Both aims are equivalent, and we obtain the Multicut problem [32], here phrased in the intra-cluster formulation

$$\underset{P \text{ a node partition}}{\operatorname{argmax}} \sum_{i<j} w_{ij} \cdot \mathbb{1}(i \sim_P j). \tag{1.3}$$

Setting the Multicut weights to $\tilde{w}_{ij} := w_{ij} - \frac{d_i d_j}{2w_{tot}}$ shows that modularity clustering is a special case of the Multicut problem.

Clustering is ubiquitous in science and engineering. For instance, k-means clustering is employed for vector quantization to compress data [99]. In some cases, both the point cloud and the graph-partitioning approaches are possible. Consider a digital image of a crowd of people. Segmenting the different persons in the image amounts to clustering the pixels belonging to one person in the same cluster. This could be achieved as point clustering if we have feature vectors for each pixel, e.g., extracted by a learned pixel embedding network [86, 129]. Alternatively, we could



(a) Digital image (b) Pixel features (c) Image graph with signed edges

Figure 1.3: Two ways of tackling the segmentation of a digital image (1.3a): By extracting features for each pixel and clustering the resulting points in feature space (1.3b) or by predicting similarity values between neighboring pixels and partitioning the resulting grid graph (1.3c).

consider the image as a grid graph on the pixels and give positive edge weights between pixels that are likely part of the same person and negative edge weights if the edge crosses the contour of a person. Again, these weights could be predicted by some learned boundary-detecting network such as in [169]. Now, we could treat the segmentation problem as graph partitioning via Multicut. Both segmentation strategies are illustrated in Fig. 1.3.

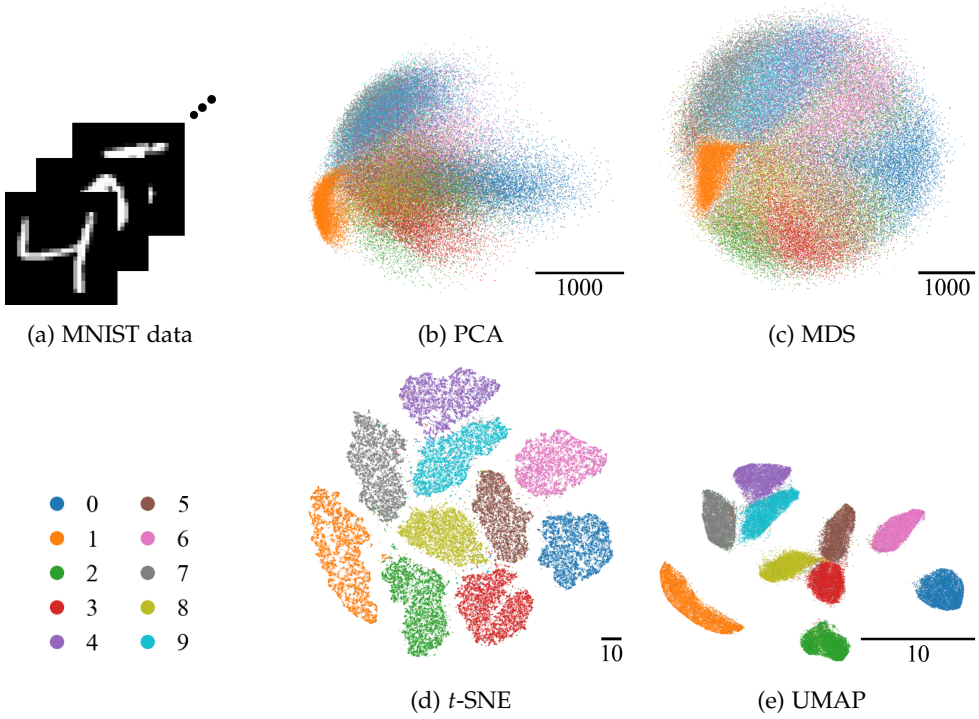
Chapter 2 considers clustering in more detail. We will introduce a new clustering method, Mod Shift, which, like Mean Shift, iteratively shifts points towards more concentrated clusters. It does not predominantly rely on the data density but allows the user to inject prior knowledge on the range of attraction and repulsion. Moreover, the point clustering method Mod Shift is intimately related to Multicut's signed graph partition objective. Mod Shift thus bridges the realms of point clustering and graph partitioning.

1.2 DIMENSIONALITY REDUCTION

To capture as much information as possible, researchers often measure as many features of their samples as possible, leading to high-dimensional datasets. Despite the potentially richer structure of such datasets, high-dimension also has drawbacks. Some of the dimensions might mostly contain noise that hinders downstream analysis. Others might contain redundant information that could be compressed, speeding-up subsequent steps without severe loss of structure. Moreover, humans are visual beings [37] and are therefore very skilled at seeing patterns and structure. However, we can see at most three dimensions. So to use our powerful built-in pattern-recognition capacities, the data needs to be presented in at most three dimensions.

As a result, reducing the dimensionality of a dataset while retaining its relevant characteristics is an important task. The question is, of course, what is the salient structure of a dataset and how to maintain it in lower dimension? What inductive biases should guide the method?

Principal Component Analysis (PCA) [130] and Multidimensional Scaling (MDS) [158], two classical methods for dimensionality reduction, follow intuitive strategies. PCA simply tries to find a projection of the data to a linear subspace of



(a) MNIST data (b) PCA (c) MDS

(d) *t*-SNE (e) UMAP

Figure 1.4: Different visualizations of the MNIST dataset consisting of 70 000 images with 28×28 pixels showing handwritten digits, which are treated as 786 dimensional vectors. Classical methods such as PCA (1.4b) or MDS (1.4c) achieve poor class separation, while *t*-SNE (1.4d) and UMAP (1.4e) isolate the classes near perfectly. Labels are only used for plotting, not for computing the visualizations.

smaller dimension that retains the highest amount of variance. Constant features are of little interest, so focusing on the largest variation seems plausible. Moreover, if the interesting variation is on a larger scale than the noise, PCA can denoise and also find correlated features. Its major limitation is that it only fits a linear subspace to the data. In contrast, MDS can find non-linear structure in a dataset by aiming for a low-dimensional layout that minimizes the squared deviation between all corresponding high- and low-dimensional pairwise distances. PCA and MDS can be considered "global" methods in that they focus on the global structure of the dataset, that is, the relation of distant parts of the data. Distant parts in the dataset likely align with the direction of largest variance, and large distances dominate the quadratic MDS loss. However, large distances can be misleading. For example, the intrinsic dimension of the dataset might be smaller than that of the ambient high-dimensional feature space. In that case, the dataset could curve so that data points, which are intrinsically far apart, might be fairly close in ambient space.

This leads to one of the currently best-accepted assumptions for dimensionality reduction, the manifold hypothesis. The data is assumed to lie on or at least near a manifold of lower dimension than feature space. The manifold hypothesis further motivates the search for a lower-dimensional representation. We might simply try to find the intrinsic manifold on which the data resides. A manifold looks only locally like Euclidean space. Globally, it may curve or meet itself. Therefore, the strategy for inferring the manifold is to focus on the local connections in the data and build up the global shape of the manifold from them [11, 139, 156, 167].

The local structure is often captured as a variant of the k-nearest neighbor graph of the data [1, 4, 11, 66, 112, 139, 156, 160, 161, 167]. The family of neighbor embeddings (NE) aims to find a low-dimensional layout that keeps neighbors nearby while placing non-neighbors not too close to each other. The two most prominent NE methods are t-SNE [160, 161] and UMAP [112]. Their two or three-dimensional visualizations of many datasets show the relevant structure of the dataset often so clearly that today they have become the de-facto standard for non-linear dimensionality reduction, having been cited more than 2000 times each in 2021 alone according to Google Scholar [58, 59]. For an example illustrating the power of t-SNE and UMAP on MNIST compared to PCA and MDS, consider Fig. 1.4.

Chapters 3 and 4 look under the hood of UMAP. We first derive its true loss function, which enables us to explain UMAP's tendency for overly crisp visualizations. Based on this correct understanding of UMAP, we derive its relation to t-SNE in Chapter 4. Up to inconsequential design choices, UMAP is a sampling-based approximation to t-SNE with higher attraction. Varying the amount of attraction leads to a whole spectrum of embedding methods inter- and extrapolating t-SNE and UMAP.

1.2.1 *Limitations of dimensionality reduction*

While it is often desirable and, to some extent, possible to reduce the dimensionality of a dataset without losing its main features, there are limits to what can be achieved. Many point configurations in high-dimensional Euclidean space cannot be reproduced in lower dimension. Any Euclidean configuration of n points can be realized in $\mathbb{R}^{n-1}$, but not necessarily in $\mathbb{R}^d$ for $d < n-1$ [145]. For instance, in less than $n-1$ dimensions, there is insufficient space to place n points equidistantly.

Lemma 1.1. *It is possible to place n points equidistantly in $\mathbb{R}^d$ if and only if $d \geq n-1$.*

Proof. This statement follows from Thm.1 in [145]. We give a more elementary proof here for intuition.

For the "if" part, consider the standard basis vectors in $\mathbb{R}^n$. They all have distance $\sqrt{2}$ from each other and span an affine subspace of dimension $n-1$.

For the "only if" part, we show by induction that an additional equidistant point requires an additional affine dimension. The full proof is in Appendix A, here we only sketch the argument and give the reason why there is not enough space for another equidistant point p in the affine hull of $n-1$ equidistant points $x_1, \ldots, x_{n-1}$. By symmetry arguments, p would have to be the midpoint $\frac{1}{n-1}\sum_{k=1}^{n-1} x_k$, but then it does not have the same distance to the x_i's that the x_i's have among each other. Indeed, we have

$$\|p - x_1\| = \left\| \frac{1}{n-1} \sum_{k=2}^{n-1} (x_k - x_1) \right\| \leq \frac{1}{n-1} \sum_{k=2}^{n-1} \|x_k - x_1\| = \frac{n-2}{n-1} d < d,$$

where d is the distance between x_1 and any of the $x_2, \ldots, x_{n-1}$. □

Exact reproduction of the pairwise distances is a strong requirement. Natural relaxations allow for small distortion of the distances or focus only on preserving relations between subsets of points. For instance, the Johnston-Lindenstrauß

Lemma [71] implies that for n points in Euclidean space, there is a layout in dimension $O(\log(n)/\varepsilon^2)$ that only distorts the Euclidean distances by a factor of at most $1 \pm \varepsilon$ for $\varepsilon \in (0, 1)$. Unfortunately, this still does not permit relatively faithful dimensionality reduction to two or three dimensions for large datasets. For instance, for the MNIST dataset with $n = 70\,000$, the sufficient dimension in [41] would be $267 \gg 3$, even for the highest ε. Conversely, Chari *et al.* [24] investigate how close a layout in $\mathbb{R}^2$ can be to equidistance. They find that the ratio of maximal distance to minimal distance grows with $\sqrt{n}$.

These limits should caution against over-interpreting very low-dimensional visualizations as produced by t-SNE and UMAP. We will discuss typical artifacts of UMAP plots in Chapter 3 and describe a whole spectrum of neighbor embeddings that highlight different aspects of a dataset in Chapter 4. However, this does not render such visualizations useless. The manifold hypothesis implies that typical data does not vary in all dimensions of the high-dimensional space, thus restricting the possible distances. Moreover, t-SNE and UMAP have impressively demonstrated that they do show structure whose existence can be corroborated by independent validation experiments [9, 83, 90, 144]. Hence, their embeddings can serve at the very least to generate hypotheses about the structure of the explored datasets.

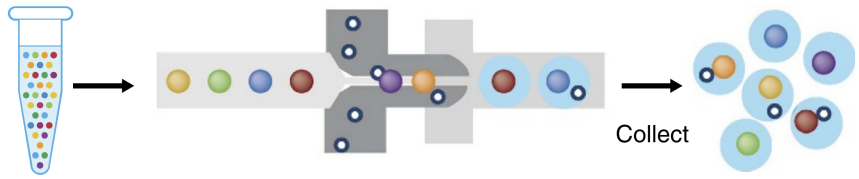
1.2.2 *Single-cell RNA sequencing*

One type of dataset used in Chapters 3 and 4 are single-cell RNA sequencing datasets, which we introduce briefly here. The cells of one organism carry the same genes yet fulfill vastly different tasks. This is possible because only certain genes are "active" or "expressed" in a given cell. The proteins encoded by these expressed genes ultimately determine the behavior and thus the function of a cell. In order to understand how an organism functions at the level of cells, it is therefore instructive to analyze the genes expressed in the cells. Until recently, this was only possible on a large scale with bulk sequencing, so the assignment of individual gene expression profiles to individual genes was lost [98]. Single-cell RNA sequencing (scRNA-seq) allows capturing the gene expression pattern of individual cells [143]. The resulting datasets often contain tens to hundreds of thousands of cells and, for each of them, the measurement of thousands of genes [19, 126], making this data modality very high-dimensional. After some initial preprocessing and a first linear dimensionality reduction, a common step in analyzing scRNA-seq data is to study t-SNE or UMAP plots [9, 83, 106]. Due to t-SNE's and UMAP's popularity in this field of bioinformatics, we chose to illustrate many of our results in Chapters 3 and 4 on this data modality. Fig. 1.5 illustrates some steps in a typical scRNA-seq pipeline.

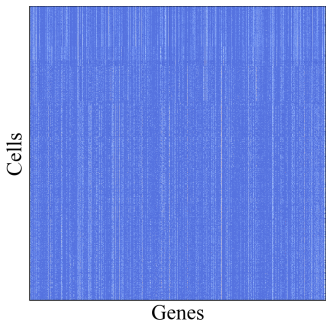
1.3 FORCE-DIRECTED LAYOUTS

Force-directed layouts [8, 49, 53, 69, 77, 122] are a popular approach for aesthetically visualizing graphs in two or three dimensions. While not the focus of this work, they are intimately connected to our work on clustering and dimensionality reduction.

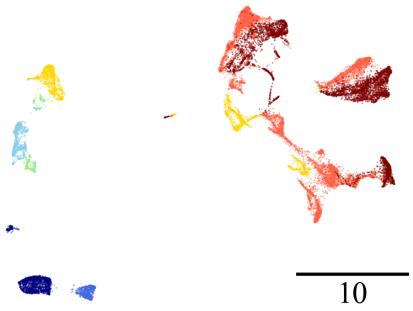
Typically, edges in an (unsigned) graph indicate the similarity of the incident nodes. Therefore, a graph layout aims at placing connected nodes nearby and non-



(a) Droplet-based scRNA-seq



(b) Count matrix



(c) UMAP plot

Figure 1.5: A droplet-based scRNA-seq pipeline. 1.5a shows the preparation of droplets for single-cell RNA sequencing. A barcoded bead (colored) and a cell are encapsulated into a droplet. Inside, the barcode tags the expressed RNA of the cell. This tagging allows RNA counts sequenced in bulk to be assigned back to individual cells (1.5a). The resulting count matrix (1.5b) can contain tens of thousands of cells and genes. After some preprocessing (not shown), *t*-SNE or UMAP plots (1.5c) can reveal meaningful structure. The data in 1.5b, c is taken from a developing human brain organoid [74]. Fig. 1.5a is adapted from [179] which is licensed under CC BY 4.0.

connected nodes further apart. Force-directed layouts achieve this by an attractive force between connected edges, which tries to pull their node embeddings together, and a repulsive force between all pairs of points, which spreads out the embedding. These forces move the node embeddings from an initial layout until attraction and repulsion balance and the final layout is found.

For a visually pleasing layout, different nodes should not superpose, and the entire layout should not diverge [122]. Therefore, on short ranges, the repulsive force typically dominates, while on long ranges, the attractive force is stronger. One way to implement these desiderata is as follows. The attractive force of node i exerted on node j is

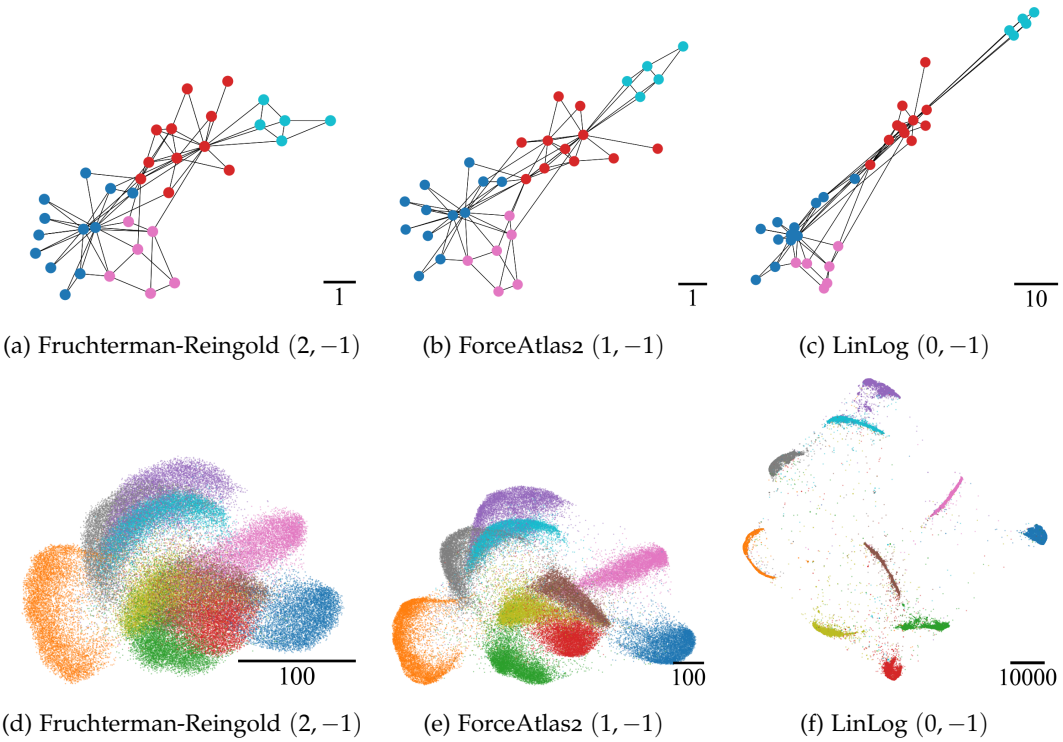
$$F_{\text{attr},ij} = w_{ij}\|e_i - e_j\|^a \frac{e_i - e_j}{\|e_i - e_j\|} \tag{1.4}$$

and the repulsive force is

$$F_{\text{rep},ij} = -d_i d_j \|e_i - e_j\|^r \frac{e_i - e_j}{\|e_i - e_j\|}, \tag{1.5}$$

where w_{ij} is the weight of the edge ij (zero if the edge does not exist), $d_i = \sum_j w_{ij}$ the degree of node i, and $a > r$ two real numbers. The condition $a > r$ leads to long-range attraction and short range repulsion [122].

The inspiration for force-directed layouts comes from the physical "spring-electric" analogy. Eades [49] envisaged the attractive forces akin to springs that connect pairs of embeddings of nodes that are incident to the same edge. These springs pull



(a) Fruchterman-Reingold $(2, -1)$ (b) ForceAtlas2 $(1, -1)$ (c) LinLog $(0, -1)$

(d) Fruchterman-Reingold $(2, -1)$ (e) ForceAtlas2 $(1, -1)$ (f) LinLog $(0, -1)$

Figure 1.6: Force-directed layouts of Zachary's karate club network [177] (top) and the kNN graph of the MNIST dataset (bottom) for different combinations of attraction and repulsion coefficients (a, r). Reducing the attractive force from the Fruchterman-Reingold layout to that of ForceAtlas2 and finally to the LinLog model highlights cluster structure more but also contracts clusters visually. The karate club network is colored by its modularity clustering. As predicted by Noack [122], the LinLog model aligns well with the modularity clustering. This figure is created using the code of Böhm *et al.* [15]. More (a, r) combinations for the MNIST dataset can be found in [14]. The MNIST dataset is colored by digit class, and we omitted the kNN graph for visual clarity.

exactly the linked node pairs together. To avoid collapse, all points repel each other similar to charges of the same sign.

By Hooke's law [147, p. 369] the force of a spring is linear in its displacement, and Coulomb's law [147, p. 564] states that the force between charged particles decays with their inverse squared distance. So in the strict sense of this physical analogy, we would have $a = 1$ and $r = -2$. In practice, other choices for (a, r) became popular. Fruchterman and Reingold [53] use $(a, r) = (2, -1)$ in their popular graph layout algorithm, ForceAtlas2 [69] utilizes $(1, -1)$ and the LinLog method [122] employs $(0, -1)$, see Fig. 1.6.

In Secs. 2.4.1, 4.3 and 4.3.2 we will see that our clustering method Mod Shift as well as the dimensionality reduction methods UMAP and t-SNE can be viewed as force-directed algorithms. All three construct a graph from the point data. UMAP and t-SNE both use a kNN graph but further depart from the "spring-electric" analogy as their attractive and repulsive forces are not proportional to any power of the distance in embedding space. For the clustering method Mod Shift, superposition of points in the same cluster is actually desired. Therefore, in Mod Shift, we have attraction for points that start out close and repulsion for points that

were initially already distant. In essence, the main algorithms treated in this work have force-directed character.

Noack [122] observed a connection between the loss functions of modularity clustering [119] and force-directed layouts, particularly for their LinLog setting $(a, r) = (0, -1)$, and illustrate how their LinLog layouts align with modularity clusters of the same graph, compare Fig.1.6c, f. We find a similar connection between a modified version of Parametric UMAP and modularity clustering in Sec. 3.7.1. These conceptual links show that, despite their seemingly different nature with discrete and continuous outputs, clustering and dimensionality reduction often have close ties. Dmitry Kobak summarized this nicely during an ICLR 2022 workshop panel [82] as "clustering is embedding into dimension zero."

1.4 THESIS OVERVIEW

This thesis contributes to two aspects of unsupervised learning: clustering and dimensionality reduction. Correspondingly, the thesis falls into two parts: Chapter 2 deals with clustering and Chapters 3 and 4 treat dimensionality reduction.

In Chapter 2, we introduce Mod Shift, a new clustering method for point data that uses a distance-based notion of attraction and repulsion to determine the number of clusters and the assignment of points to clusters. It iteratively moves points towards crisp clusters like Mean Shift but also has close ties to the Multicut problem via its loss function. As a result, it connects signed graph partitioning to clustering in Euclidean space. After exploring Mod Shift's relation to the Multicut problem and to Mean Shift theoretically, we exhibit its practical value for pixel embedding-based instance segmentation. Chapter 2 is based on [40].

Chapter 3 establishes a new understanding of the prominent dimensionality reduction and visualization method UMAP. We analyze the details of UMAP's implementation and find its actual loss function. It differs drastically from the one usually stated and uses much less repulsion. This discrepancy allows us to explain some typical artifacts in UMAP plots, such as the dataset size-dependent tendency to produce overly crisp substructures. Contrary to existing belief, we find that UMAP's high-dimensional similarities are not critical to its success as the diminished reduction effectively binarizes the high-dimensional similarities. Chapter 3 is based on [39].

Equipped with this corrected view of UMAP, we describe its precise connection to the other state-of-the-art visualization method, t-SNE, in Chapter 4. The key insight is a new, exact relation between the contrastive loss functions negative sampling, employed by UMAP, and noise-contrastive estimation, which has been used to approximate t-SNE. As a result, we explain that UMAP embeddings appear more compact than t-SNE plots due to increased attraction between neighbors. Varying the attraction strength further, we obtain a spectrum of neighbor embedding methods, encompassing both UMAP and t-SNE as special cases. Moving from more attraction to more repulsion shifts the focus of the embedding from continuous, global to more discrete and local structure of the data. Finally, we emphasize the link between contrastive neighbor embeddings and self-supervised contrastive learning. Chapter 4 is based on [38].

Finally, Chapter 5 concludes the thesis and gives an outlook on future work.

Part I

CLUSTERING

2

MOD SHIFT: A PRINCIPLED ALTERNATIVE TO MEAN SHIFT CLUSTERING USING LONG-RANGE REPULSION

Motivated by the idea of making the Multicut problem differentiable, we derive a novel point clustering algorithm, Mod Shift, and relate it to Mean Shift, thus establishing a new connection between graph partitioning and clustering in Euclidean space. Mod Shift sets itself apart from Mean Shift in that it uses both attractive and repulsive forces to iteratively shift points in Euclidean space. The number of clusters emerges as a result of the optimization process and does not have to be specified by the user. Experiments show that Mod Shift performs comparably to Mean Shift in a challenging neural segmentation task. This chapter is based on [40].

2.1 INTRODUCTION

Proposal-free bottom-up approaches have recently become popular for the challenging task of instance segmentation [30, 44, 51, 86, 95, 107, 117, 118, 129]. These train neural networks to predict a feature or associative embedding vector in $\mathbb{R}^k$ for each pixel following the main idea of metric learning: that pixels belonging to the same instance should have nearby embeddings, while pixels of different instances should have embeddings that are far apart. However, limited by the network architecture and training process, these embeddings are usually not crisp. Embeddings of pixels that belong to the same instance are not identical but have some spread, and the pixel embeddings associated with different instances sometimes overlap without having a clear separating margin. Thus, a subsequent clustering step, which determines the number of clusters and their members, is necessary to produce an actual instance segmentation of the pixels.

In this chapter, we introduce "Mod Shift", a principled general purpose clustering algorithm inspired by the concept of metric learning, that proximity in Euclidean space relates to similarity of the data points while distance is associated with dissimilarity. Indeed, for the clusters to become crisp and well-separated, Mod Shift moves nearby points closer together while distant points get pushed further apart. This is realized via a differentiable point shifting procedure, similar to Mean Shift [31, 34, 54], or force-directed graph layouts [8, 77].

While Mod Shift is a clustering method for points in Euclidean space, its objective function resembles a continuous formulation of the Multicut problem [32, 75]. Any instance of Mod Shift translates to a Multicut problem by discarding all positional information beyond computing the weights. The optimal solution of the Multicut problem is also the best integral Mod Shift solution. In this way, Mod Shift has close ties to graph partitioning despite operating entirely in Euclidean space. As for Multicut, balancing attraction and repulsion between points during the optimization process controls the number of clusters. Therefore, it does not need to be specified in advance.

Empirically, Mod Shift is on par with Mean Shift but additionally has a theoretical relation to Multicut. If the number of clusters is *known*, energy statistics provides a different connection between clustering points in $\mathbb{R}^k$ and graph partitioning [52].

In summary, our contributions are

1. proposing a principled differentiable clustering algorithm (Sec. 2.4.1),
2. relating it conceptually to two well-established clustering methods, Multicut and Mean Shift, thus exploring a new bridge between graph-based and Euclidean clustering (Sections 2.4, 2.5, 2.6),
3. an open-source implementation at `https://github.com/ModShift/ModShift`,
4. illustrating Mod Shift's behavior on toy data and evaluating its performance for clustering pixel embeddings on real datasets (Sec. 2.7).

2.2 RELATED WORK

MULTICUT: Inspired by the Multicut (correlation clustering) problem for graph partitioning [32, 75], we derive a differentiable objective function for clustering points in Euclidean space, see Sec. 2.4. The Multicut problem has been used for graph-based computer vision in many ways, in particular for tracking [154], image labeling [75, 88] and image partitioning both by semantics and by instances [2, 75, 76, 79, 81]. Most optimization strategies however are not differentiable but based on ILPs [75], on dual approaches [176] or on move-making strategies [10]. An exception are Song *et al.* [151] who cast the Multicut problem as a conditional random field over node labels. Instead, we achieve differentiability by encoding the points' partitioning by their spatial configuration in Euclidean space. Thus, in contrast to [151], the cycle inequalities hold automatically for Mod Shift, but the edge variables are not guaranteed to be integral, see Sec. 2.5.

PIXEL EMBEDDINGS FOR INSTANCE SEGMENTATION: One of the first works learning dense pixel embeddings with a neural network for subsequent vision tasks was [118]. Since then, it has become popular to use pixel embeddings for instance segmentation [30, 44, 51, 86, 95, 107, 117, 129]. The idea is always similar: A neural network produces pixel embeddings that are subsequently clustered. The network's loss function penalizes embeddings of pixels in different instances that are too close and embeddings of pixels belonging to the same instance that are far apart. The embedded pixels must be clustered in a second step to obtain the instances without knowing the number of clusters in advance. In most cases this is done non-differentiably, e.g., with HDBSCAN [129], with k-nearest neighbors classifiers using a few labeled pixels [30], or with a distance threshold around (learned) seeds [44, 51, 117]. An exception are Kong and Fowlkes [86] who use the differentiable Mean Shift clustering. Our approach offers a principled alternative to Mean Shift in such a pipeline.

MEAN SHIFT: The Mean Shift algorithm is one of the few clustering algorithms for Euclidean data that does not need a prespecified number of clusters and is differentiable as it shifts data points to a kernel-weighted mean of neighboring

points. It was introduced as a general purpose clustering approach in [31, 54] but became most popular in the computer vision community for image filtering, segmentation [34] and tracking [35]. While our Mod Shift is also designed as a point shifting and hence differentiable clustering algorithm, it is primarily distance- and not density-focused. Also, Mod Shift crucially uses both attraction and repulsion, see Sec. 2.6.

FORCE-DIRECTED LAYOUTS: Many methods for drawing graphs [8, 77] assume a physical model in which the nodes of the graph exert attractive and repulsive forces on each other. By balancing these forces, a layout is found. A short-range repulsive force usually prevents identical node locations. Mod Shift can also be viewed as a force-directed approach. The main difference is that we are interested in crisp, well-separated clusters and not in a visually pleasing layout. Thus, Mod Shift uses short-range attraction and long-range repulsion. Force-directed layouts are also related to the combinatorial modularity clustering problem [120, 122].

DEEP CLUSTERING NETWORKS: Similar in spirit to our work but usually not used for instance segmentation are deep clustering networks [70, 135, 173]. Here, a self-supervised neural network clusters a dataset by learning from the most pronounced similarity relations in the data. This is akin to our idea in that Mod Shift pulls points closer together if they start close and pushes distant points even further apart. What is different in our setting and crucial for instance segmentation is that we do not need a prespecified number of clusters. Other than Kong and Fowlkes [86], the only such deep clustering approach is [149], which uses the robust continuous clustering loss of [148]. Since this has a (robust) quadratic attraction term, clusters do not become crisp.

2.3 BACKGROUND

2.3.1 *Notation*

Let $x = \{x_1, \dots, x_n\}$ be n points in $\mathbb{R}^k$. When these are the initial points to be clustered, e.g., the pixel embeddings that an embedding network produced, we write them as $x^0 = \{x_1^0, \dots, x_n^0\}$. During the clustering process, they are iteratively shifted to positions $x^t = \{x_1^t, \dots, x_n^t\}, t \in \mathbb{N}_0$ so that clusters become more pronounced. Most quantities of interest will be defined in terms of the distances $d_{ij}^t := \|x_i^t - x_j^t\|$, $1 \leq i < j \leq n$, where $\|\cdot\|$ is the Euclidean norm.

2.3.2 *Summary Mean Shift*

Mean Shift [31, 34, 54] refers to a group of clustering algorithms for points in Euclidean space that do not need a prespecified number of clusters. Given a kernel function K, often a flat kernel or an RBF kernel, points x_i^t are shifted iteratively to a kernel-weighted mean of their neighbors

$$x_i^{t+1} = \frac{\sum_{j=1}^n K(\|x_i^t - y_j\|) y_j}{\sum_{j=1}^n K(\|x_i^t - y_j\|)}, \tag{2.1}$$

where either $y_j = x_j^0$ for all j (fixed version) or $y_j = x_j^t$ for all j (adaptive version). These correspond to gradient ascent with implicitly defined step size on a (changing) kernel density estimate of the y_j's with a kernel related to K, see [31]. It can be interpreted as minimizing Renyi's (cross-) entropy [134].

2.3.3 *Summary Multicut*

The NP-hard Multicut (correlation clustering) problem [32] is the problem of finding a node partition of an undirected graph with signed edge weights such that the cumulative weight of the edges between different partition elements is minimal. On a complete graph with nodes $\{1, \dots, n\}$ and edge weights $\tilde{w}_{ij}$ the problem can be formulated as

$$\underset{P \text{ a partition of } \{1,\dots,n\}}{\operatorname{argmin}} \sum_{i<j} y_{ij} \cdot \tilde{w}_{ij} \text{ such that } y_{ij} = 1 - \mathbb{1}(\exists p \in P : i, j \in p), \tag{2.2}$$

where $\mathbb{1}$ is the indicator function. The binary edge indicator variables y_{ij} encode whether two points, i and j, belong to the same ($y_{ij} = 0$) or to different clusters ($y_{ij} = 1$), i.e., whether the edge ij is cut. The vector y is also called Multicut vector. The optimization can be performed directly over the $y_{ij} \in \{0, 1\}$ if one ensures that these correspond to a valid partition. This is the case if the cycle inequalities hold, see Lem. 2.2 in [32]. For a complete graph, they reduce to the triangle inequalities, see Thm. 3.2 in [32], such that we can equivalently rewrite the problem (2.2) as:

$$\underset{y_{ij} \in \{0,1\}}{\operatorname{argmin}} \sum_{i<j} y_{ij} \cdot \tilde{w}_{ij}, \text{ such that } \forall i, j, k : y_{ik} \leq y_{ij} + y_{jk}. \tag{2.3}$$

2.4 FROM MULTICUT TO MOD SHIFT

2.4.1 *Introducing Mod Shift*

In this section, we introduce our point-clustering algorithm Mod Shift by making the objective function of the graph-clustering problem Multicut differentiable. A key difficulty of the combinatorial Multicut problem (2.3) stems from the integrality constraints on y_{ij}, which make the problem non-differentiable and NP-hard. In contrast, Mod Shift moves points in continuous Euclidean space to solve the differentiable optimization problem

$$\underset{x=(x_1,\dots,x_n)^\top \in \mathbb{R}^{n \times k}}{\operatorname{argmin}} \sum_{i<j} \rho(\|x_i - x_j\|) \cdot w(\|x_i^0 - x_j^0\|). \tag{2.4}$$

While structurally parallel to problem (2.3), in Mod Shift's objective function the binary edge indicator variables y_{ij} are replaced by a differentiable separatedness function ρ. Moreover, this separatedness ρ and the weights w, both defined below, depend on point configurations x and x^0 in Euclidean space. This allows Mod Shift to optimize its objective (2.4), or energy

$$E(x, x^0) := \sum_{i<j} \rho(\|x_i - x_j\|) \cdot w(\|x_i^0 - x_j^0\|) \tag{2.5}$$

of the configuration x given initial positions x^0, iteratively using gradient descent starting at $x = x^0$ and then shifting x^t to x^{t+1} at step t, so that they form crisp

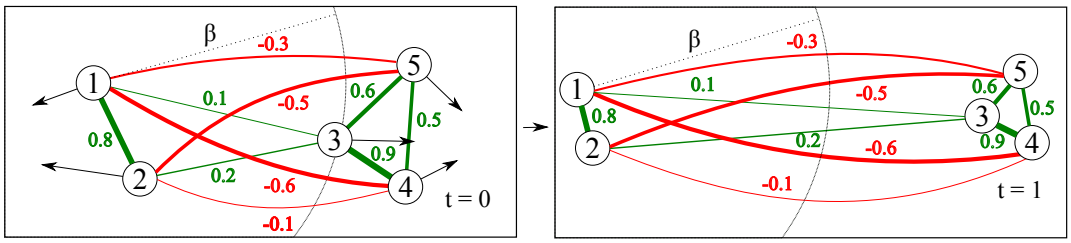


Figure 2.1: Schematic illustration of the first Mod Shift iteration on five points. The edge labels, colors, and widths show the weights between all pairs of points (green attractive, red repulsive). For instance, all points within (outside) a radius of β around point 1 are attracted to (repelled from) it. The black arrows indicate the shift from step $t = 0$ to $t = 1$. The Mod Shift step makes clusters more apparent by moving points 1 and 2 as well as 3, 4, and 5 closer together while separating these two groups from each other. Point 3 is more attracted by points 4 and 5 than by 1 and 2 and consequently moves to the right. As the weights depend only on the *initial* point positions, they do not change from $t = 0$ to $t = 1$. Therefore, at $t = 1$ point 3 is still attracted to point 1, although their distance is now greater than β.

and well-separated clusters. The gradient $-\partial E(x^t, x^0)/\partial x^t$ at time t can be viewed as the forces according to which the points are shifted, exhibiting Mod Shift's force-directed character:

$$-\frac{\partial E(x^t, x^0)}{\partial x_i^t} = \sum_{j \neq i} \rho'(d_{ij}^t) \cdot w(d_{ij}^0) \cdot \frac{x_j^t - x_i^t}{\|x_j^t - x_i^t\|}. \tag{2.6}$$

Point x_i^t moves towards x_j^t if the two points attract each other ($w(\|x_i^0 - x_j^0\|) > 0$) and away from x_j^t if they repel each other ($w(\|x_i^0 - x_j^0\|) < 0$) as illustrated in Fig. 2.1. The Mod Shift algorithm is summarized in Alg. 1. There, the standard gradient descent in line 7 could be replaced by a more powerful optimizer such as Adam [80].

In the following subsections, we show how Mod Shift capitalizes on the correspondence of proximity in $\mathbb{R}^k$ to similarity and distance to dissimilarity in two ways: through the weights w between points, which drive the clustering process by modulating attraction and repulsion between all pairs of points; and through the cluster affiliation $1 - \rho$, which indicates to what extent two points cluster together.

We dub our method "Mod Shift" due to its point-shifting nature and the modulation of attraction and repulsion.

2.4.2 *Choice of the weights w*

In contrast to Multicut, where the weights are given as input, in Mod Shift we infer them from the initial point configuration in the metric space $\mathbb{R}^k$. This has a useful regularizing effect [107]. If the data points are, for instance, pixel embeddings produced by a neural network, we trust the network to place them at positions so that their distances meaningfully indicate which pixels should belong to the same and which to different instances. Thus, if two points are initially close together (i.e., two pixels are embedded nearby), we assume they should belong to the same cluster, which we model with a positive (attractive) weight between them.

Algorithm 1: Mod Shift

input : points $x_1^0, \dots, x_n^0$,
number of shifts m,
learning rate α,
weight function w,
derivative of separatedness ρ'

output: shifted points $x_1^{\text{shifted}}, \dots, x_n^{\text{shifted}}$

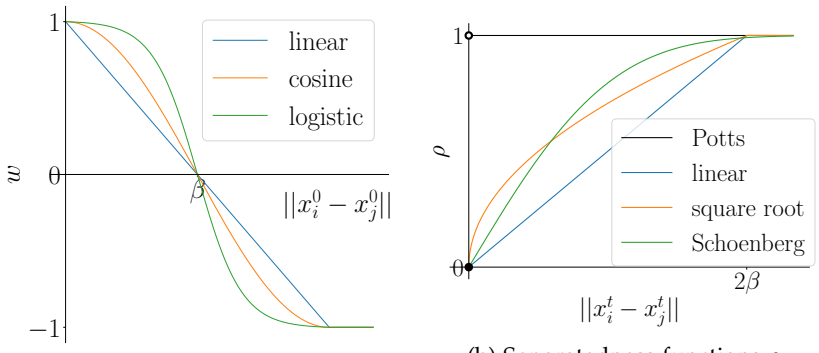
for $t = 0$ **to** $m - 1$ **do**
  for $i = 1$ **to** n **do**
    $\text{grad}_i \leftarrow 0$
    for $j = 1, \dots, i-1, i+1, \dots, n$ **do**
      $\text{grad}_{ij} \leftarrow \rho'(||x_i^t - x_j^t||) \cdot w(||x_i^0 - x_j^0||) \cdot \frac{x_i^t - x_j^t}{\|x_i^t - x_j^t\|}$
      $\text{grad}_i \leftarrow \text{grad}_i + \text{grad}_{ij}$
    $x_i^{t+1} = x_i^t - \alpha \cdot g_i$
return $x_1^m, \dots, x_n^m$

Conversely, points that start far apart should likely be separated, modeled by a negative weight (repulsion) between them. So, in our model, the initial distances suggest which points should (not) be clustered together. A non-increasing weight function $w(d_{ij}^0) \in \mathbb{R}$ determines these weights as a function of the initial distances d_{ij}^0. It should be positive at 0 and negative for large distances. Its zero-crossing $\beta := w^{-1}(0)$ defines the scale of the clustering: Pairs of points that start out closer than β attract each other, while points with an initial distance larger than β repel each other. This scale parameter, which is always present in clustering methods for Euclidean points, e.g., in the form of a bandwidth or the number of nearest neighbors, can be chosen based on prior domain knowledge. In particular, if an embedding network was trained with a contrastive loss [44], one could choose β based on the margin below which the loss penalizes pixel embeddings of different instances. Note that such a repulsion, which complements local attraction, is not modeled by density-based clustering approaches such as Mean Shift [31] or HDBSCAN [111]. A simple choice is $w(d) = \max(1 - \frac{d}{\beta}, -1)$. It linearly decreases the weight from 1 to -1 over the interval $[0, 2\beta]$ and saturates beyond 2β. This function and other choices for w are depicted in Fig. 2.2a.

2.4.3 *Choice of the cluster affiliation* $1 - \rho$

While we use the initial point positions to derive weights encoding which points should (not) be clustered together, inferring a hard clustering is non-trivial. Therefore, Mod Shift moves the points in $\mathbb{R}^k$ according to their weights towards a crisp, well-separated configuration so that cluster affiliation becomes clear.

In order to capture the situation prior to such an obvious clustering differentiably, we define a differentiable cluster affiliation for all pairs of points based on their distance. Two points belong more strongly to the same cluster if they are closer and are more clearly in different clusters if they are further apart. We formalize this with a non-decreasing, differentiable "separatedness" function $\rho(d_{ij}^t) \in [0, 1]$



(a) Weight functions w

(b) Separatedness functions ρ

Figure 2.2: 2.2a: Choices for Mod Shift's weight function w. Note that the weight only depends on the initial distance between points and is never updated in fixed Mod Shift. 2.2b: Choices for Mod Shift's separatedness function ρ of the distance of two points x_i^t and x_j^t after t iterations: the Potts function and concave approximations given by a rectified linear, rectified square root and Schoenberg transformation [146]. Formulae are given in Appendix B.1.

with $\rho(0) = 0$ and $[0,1) \subset \mathrm{im}(\rho)$. The cluster affiliation is given by $1 - \rho$. In words, $\rho(d_{ij}^t) = 0$ means that i and j belong to the same cluster while $\rho(d_{ij}^t) = 1$ represents their association with different clusters. Intermediate values interpolate these clear cluster affiliations. The separatedness function ρ transforms a spatial point configuration into a soft partition. In the following, we discuss sensible choices for ρ.

Remarkably, the well-known Lem. 2.1 (see Lem. 9.0.2 in [45] or Prop. 2.3 [36]) shows that the triangle inequalities are always satisfied in Mod Shift if we choose a concave function ρ. We prove it for the convenience of the reader.

Lemma 2.1. *Let (X,d) be a pseudo-metric space, that is a metric space in which different elements may have zero distance, and $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ a non-decreasing function with $f(0) = 0$, which is subadditive or concave. Then $(X, f \circ d)$ also is a pseudo-metric space.*

Proof. Symmetry of $f \circ d$ is clear as well as $f(d(x,x)) = 0$ for all $x \in X$. We check the triangle inequality for subadditive f first. Let $x, y, z \in X$. Then

$$d(x,z) \leq d(x,y) + d(y,z) \tag{2.7}$$

and the facts that f does not decrease and is subadditive imply

$$\begin{aligned} f(d(x,z)) &\leq f(d(x,y) + d(y,z)) \\ &\leq f(d(x,y)) + f(d(y,z)), \end{aligned} \tag{2.8}$$

which is the triangle inequality for $(X, f \circ d)$. It remains to show that when $f(0) = 0$, concavity implies subadditivity. In this setting we have for any $c \in \mathbb{R}_{\geq 0}$ and $t \in [0,1]$:

$$f(tc) = f(tc + (1-t) \cdot 0) \geq tf(c) + (1-t) \cdot f(0) = tf(c). \tag{2.9}$$

Let $a, b \in \mathbb{R}_{\geq 0}$ not both be 0 and $t = \frac{a}{a+b}$. Then we get

$$f(a) = f\big(t \cdot (a+b)\big) \geq tf(a+b) \tag{2.10}$$

$$f(b) = f\big((1-t) \cdot (a+b)\big) \geq (1-t)f(a+b), \tag{2.11}$$

whose sum is just subadditivity. □

In Prop. 2.3 we show how to encode partitions in Euclidean space with a concave separatedness function ρ. But first, we need a technical lemma. Here and in the rest of this section, we always write maximum and minimum even if ρ reaches the extreme value only asymptotically.

Lemma 2.2. *Let $\rho : \mathbb{R}_{\geq 0} \to [0,1]$ be a concave and non-decreasing function with $\rho(0) = 0$, at which point it is continuous. Then ρ is continuous on $\mathbb{R}_{\geq 0}$ and strictly increasing up to some argument μ (potentially $\mu = \infty$) at which it reaches its maximum value.*

Proof. The continuity on $\mathbb{R}_{>0}$ is a typical analysis textbook result, see, for instance, Thm. 1.3.3. in [121]. If $\rho(x) = \rho(x')$ for $x' > x$, then for all $x' > y > x$ we have

$$\rho(y) \geq \lambda\rho(x') + (1-\lambda)\rho(x) = \rho(x) \tag{2.12}$$

for $\lambda = \frac{y-x}{x'-x}$. But at the same time $\rho(y) \leq \rho(x') = \rho(x)$ as ρ is non-decreasing. Similarly, writing x' as the mixture of x and any $y > x'$ yields that $\rho(y) = \rho(x)$ for every $y > x'$. Thus, from x onwards, ρ is constant. □

In the following Prop. 2.3, we prove that a concave separatedness function ρ provides a criterion for unambiguously inferring a clustering from a spatial configuration.

Proposition 2.3. *Let $\rho : \mathbb{R}_{\geq 0} \to [0,1]$ be a continuous, non-decreasing and concave function with $\rho(0) = 0$ and $[0,1) \subset \mathrm{im}(\rho)$. Let further $x_1, \ldots, x_n$ be in $\mathbb{R}^k$. Suppose there is some $0 \leq \varepsilon < \frac{1}{4}$ such that for all i, j the separatedness $\rho_{ij} := \rho(\|x_i - x_j\|)$ is either not above ε or not below $1-\varepsilon$. Then there is a partition of $\{1, \ldots, n\}$ so that whenever i, j are in the same partition element we have $\rho_{ij} \leq \varepsilon$ and $\rho_{ij} \geq 1-\varepsilon$ if they are in different partition elements. The thresholds ε and $1-\varepsilon$ translate to thresholds on the Euclidean distances $\|x_i - x_j\|$.*

Proof. Consider the sets $S_i := \{j | \rho_{ij} \leq \varepsilon\}$. We show that they are pairwise identical or disjoint. By Lem. 2.1, ρ preserves the triangle inequality. Let there be some $k \in S_i \cap S_j$. Then, $\rho_{ij} \leq \rho_{ik} + \rho_{kj} \leq 2\varepsilon$. Thus, for $l \in S_i$,

$$\rho_{jl} \leq \rho_{ji} + \rho_{il} \leq 3\varepsilon < 1 - \varepsilon. \tag{2.13}$$

By assumption this means $\rho_{jl} \leq \varepsilon$ and hence $l \in S_j$. So $S_i \subset S_j$ and equality holds by symmetry. Clearly, the S_i's cover $\{1, \ldots, n\}$ and hence form a partition with the desired properties by construction.

By Lem. 2.2 ρ is strictly increasing on some $[0, \mu) \subset [0, \infty)$ on which ρ ranges over $[0,1)$. Choose the unique preimages of ε and $1-\varepsilon$ in $(0, \mu)$ as distance thresholds d_1 and d_2 if $\varepsilon > 0$. For $\varepsilon = 0$, choose the thresholds 0 and $\min(x | \rho(x) = 1)$. Then all pairs of points have a distance either below d_1 or above d_2. In the former case, they belong to the same partition element, and in the latter, they belong to different partition elements. □

In other words, if the separatedness values ρ_{ij} are sufficiently close to 0 and 1, there are no "bridges" between clusters. Trivial clustering algorithms such as thresholded Single Linkage clustering or even just thresholding around arbitrary points can safely obtain a crisp clustering.

We have established that a metric representation of points is suitable for encoding partitions. In particular, when $\rho(\|x_i^t - x_j^t\|)$ converges to 0 or 1 for each

pair i, j as $t \to \infty$, the vector $\left(\rho(\|x_i^t - x_j^t\|)\right)_{i,j}$ is close to a binary Multicut vector and encodes a valid partition of the data. Defining the feasible set of Mod Shift $F_\rho := \{\rho(\|x_i - x_j\|) | x_1, \dots, x_n \in \mathbb{R}^k\}$, we conclude that its integral points are precisely the Multicut vectors.

Corollary 2.4. *Let $\rho : \mathbb{R}_{\geq 0} \to [0, 1]$ be a surjective, continuous, non-decreasing and concave function with $\rho(0) = 0$. Then the integral points of Mod Shift's feasible set F_ρ are exactly the Multicut vectors. Hence, the optimal Multicut solution to the signed graph partitioning problem with Mod Shift's weights $w(d_{ij}^0)$ as edge weights is the same as the optimal integral solution of Mod Shift.*

Proof. The integral points of F_ρ are such that for every pair of points x_i and x_j we have $\rho_{ij} := \rho(\|x_i - x_j\|) = 0$ or $\rho_{ij} = 1$. Applying Prop. 2.3 with $\varepsilon = 0$ to such an integral point yields the partition of data points whose Multicut vector is $(\rho_{ij})_{ij}$. Conversely, a Multicut vector encodes a partition. Place the points $x_1, \dots, x_n$ in $\mathbb{R}^k$ such that points in the same partition element have the same position and place points of different partition elements at any distance greater than the smallest value at which ρ equals 1. This is possible even for $k = 1$. Then ρ_{ij} is either zero or one, hence integral, and equals the Multicut vector.

The statement on optimal solutions follows directly from the above and the similarity of the objective functions of Mod Shift and Multicut. □

As the points x_i^t lie in Euclidean space, a metric space, their distances always respect the triangle inequality. By choosing ρ to be concave, the triangle inequalities also hold for the separatedness $\rho(d_{ij}^t)$. In the Multicut formulation (2.3) the triangle inequalities over edge indicators need to be enforced independently. By modeling the separatedness based on point configurations in a metric space, they are automatically satisfied in Mod Shift, and we do not have to impose them explicitly. In other words, our separatedness based on points in a metric space is a relaxation of the binary Multicut vector that is by design regularized towards encoding a partition. Since we use Euclidean space as metric space, we restrict even more than just by the triangle inequalities, which we discuss in the following Sec. 2.5.

Prop. 2.3 leaves some freedom for choosing ρ. As points in Euclidean space are identical if and only if they have zero distance, one might be inclined to choose $\rho(d) = \mathbb{1}(d = 0)$, the Potts function. However, the Potts function violates the continuity assumption and does not provide useful gradients, making gradient descent impossible. Therefore, we approximate the Potts function with a function ρ with non-zero gradients.

Often, the clustering objective for nearby points in Euclidean space is essentially quadratic [26, 68, 148]. While this yields numeric stability for pairs of points whose distance approaches zero, it also means that small non-zero distances have little incentive to become crisp and the clusters remain spread out. But to make the extraction of a hard clustering trivial, we want cluster members to collapse to a single point. Therefore, we choose a ρ with a strictly positive derivative at zero so that zero distance is strongly preferred over a small non-zero distance. In fact, the concavity requirement of Lem. 2.1 and Prop. 2.3 makes a strictly positive derivative of ρ at 0 a necessity. The resulting numerics at distance zero, where Mod Shift's objective function (2.33) lacks a derivative, are not a problem for powerful gradient descent optimizers such as Adam or suitable learning rate decays. We discuss

convergence guarantees for Adam in more detail in Appendix B.5. One realization of our specifications is $\rho(d) = \min(\frac{d}{2\beta}, 1)$, depicted together with others in Fig. 2.2b.

2.5 MOD SHIFT'S FEASIBLE SET AND THE MULTICUT POLYTOPE

So far, we only used the fact that the space $\mathbb{R}^k$ in which the points x_i lie is a metric space. However, distances in Euclidean space are restricted by more than the triangle inequality. This section explores how Mod Shift's feasible set F_ρ relates to (relaxations of) the convex hull of all Multicut vectors, the Multicut polytope MC.

The following lemma analyzes spaces as depicted in Fig. 2.3 and will be handy later.

Lemma 2.5. *The set of four points $\Delta = \{0, 1, 2, 3\}$ with symmetric map $d : \Delta^2 \to \mathbb{R}_{\geq 0}$ given by $d(i, i) = 0$, $d(j, j') = 1$ and $d(0, j) = \alpha$ for $i \in \Delta$ and $j, j' \in \Delta \backslash \{0\}$ is a metric space if and only if $\alpha \geq 0.5$ and can be realized as a point configuration in Euclidean space if and only if $\alpha \geq \frac{1}{\sqrt{3}}$.*

Proof. Symmetry holds by construction, and for $\alpha > 0$, we also have definiteness. To show that (Δ, d) is a metric space, all that remains is the triangle inequality. The only triangle inequalities that could be violated are of the form

$$1 = d(1, 2) \leq d(1, 0) + d(0, 2) = 2\alpha. \tag{2.14}$$

Hence, all triangle inequalities hold precisely for $\alpha \geq 0.5$.

Suppose the three points $1, 2, 3$ form an equilateral triangle in Euclidean space. As they all have the same distance α from 0, the only possible positions for 0 are on a line through the center of the triangle given by $1, 2, 3$ and perpendicular to the plane it spans. Hence, the metric space (Δ, d) can be realized in Euclidean space exactly if α is at least $\frac{1}{\sqrt{3}}$, which is the distance from 0 to the other points if it lies at the center of the triangle given by $1, 2, 3$. □

2.5.1 *Mod Shift and Multicut relaxation by triangle inequalities*

Here, we relate Mod Shift's feasible set F_ρ with the known Multicut polytope relaxation C given as the feasible set of the following relaxed Multicut problem (2.3), see [175]:

$$\operatorname*{argmin}_{y_{ij} \in [0,1]} \sum_{i<j} y_{ij} \cdot \tilde{w}_{ij}, \text{ such that } \forall i, j, k : y_{ik} \leq y_{ij} + y_{jk}. \tag{2.15}$$

In other words, C is the intersection of the unit hypercube in $\mathbb{R}^{\binom{n}{2}}$ with the set of distances that respect the triangle inequalities (the metric cone [45]). By Lem. 2.1, Mod Shift respects the triangle inequalities for concave ρ, so that F_ρ is contained in C. The next proposition shows that this inclusion is strict for suitable ρ.

Proposition 2.6. *Let $\rho : \mathbb{R}_{\geq 0} \to [0, 1]$ be surjective, non-decreasing, concave with $\rho(0) = 0$ and continuous at this point. If $n \geq 4$ there exist vertices y of $C \subset \mathbb{R}^{\binom{n}{2}}$ which are not in F_ρ.*

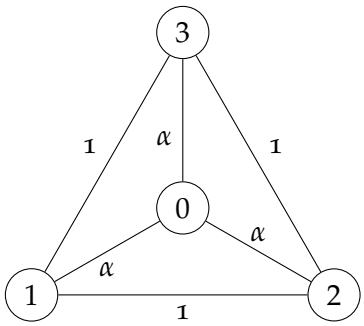


Figure 2.3: Example of the regularization of point distances in a metric space of four elements. The outer distances are 1, and the ones to node 0 are α. The triangle inequality implies that if two points in a metric space have a distance d, then a third point must have a distance of at least $\frac{d}{2}$ to at least one of them. This way, chaining effects are somewhat limited in metric spaces. So, if the points of the figure are in a metric space, α must be at least 0.5. If these points live in Euclidean space, the regularization is even stronger. In fact, $\alpha = 0.5$ is not possible in this case as point 0 would have to lie on the midpoints of the lines $\overline{12}$, $\overline{23}$ and $\overline{31}$ simultaneously. The smallest possible value of α in a Euclidean space is $\frac{1}{\sqrt{3}}$, in which case the figure is an equilateral triangle with 0 at its midpoint, see Lem. 2.5.

Proof. We start with the case $n = 4$. It suffices to show that the distances y in the configuration in Fig. 2.3 with $\alpha = 0.5$ constitute a vertex of C that is not in F_ρ. Clearly, y lies in C as all triangle inequalities are satisfied, and its values are in $[0, 1]$. Since y lies in $\mathbb{R}^{\binom{4}{2}} = \mathbb{R}^6$, we need to find 6 linearly independent constraints valid for C that are active at y, to show that y is a vertex of C. It is easy to check that the three triangle inequalities induced by the central vertex and two outer vertices together with the three outer edges of maximal value 1 form such a set. Thus, y is a vertex of C.

Next, we show that it cannot lie in F_ρ. By Lems. 2.1 and 2.2, we know that ρ continuously and strictly increases from 0 to 1 and remains constant from there on. Suppose, for a contradiction, that there were four points $x_1, \dots, x_4 \in \mathbb{R}^k$ with $\rho_{ij} := \rho(\|x_i - x_j\|) = y_{ij}$ for all $1 \leq i < j \leq 4$. Denote the distance between points x_i and x_j by d_{ij}. Since we assume $\rho_{ij} = y_{ij}$, we get $\rho_{12} = \rho_{14} + \rho_{24}$. Without loss of generality, we can assume $d_{14} \leq d_{24}$. By concavity and monotonicity of ρ and because of $\rho(0) = 0$, we get the first three of the inequalities in

$$\frac{\rho_{14}}{d_{14}} \geq \frac{\rho_{24}}{d_{24}} \geq \frac{\rho_{12}}{d_{12}} \geq \frac{\rho_{12} - \rho_{24}}{d_{12} - d_{24}} \geq \frac{\rho_{14}}{d_{14}}. \tag{2.16}$$

The last inequality stems from our previous computation and the triangle inequality $d_{12} \leq d_{14} + d_{24}$. But this induces

$$\frac{\rho_{14}}{d_{14}} = \frac{\rho_{24}}{d_{24}} = \frac{\rho_{12}}{d_{12}} \tag{2.17}$$

which implies, together with the concavity and monotonicity of ρ, that ρ is linear between 0 and d_{12}. But at d_{12} we know that ρ reaches its maximal value $\rho(d_{12}) = y_{12} = 1$. Thus, $\rho(x) = \min(\frac{x}{d_{12}}, 1)$. Now, by symmetry and easy computation, we get that $d_{ij} = y_{ij} \cdot d_{12}$ for all $1 \leq i < j \leq 4$. So the point configuration $x_1, \dots, x_4$ is just a scaled version of Fig. 2.3 with $\alpha = 0.5$. By Lem. 2.5 such a

point configuration cannot be realized in Euclidean space so that we arrive at a contradiction. Thus, y does not lie in F_ρ.

Finally, we describe how to construct such vertices y of C not in F_ρ for $n > 4$. The key is to observe that if y is mapped to the above configuration on four points by a projection $\mathbb{R}^{\binom{n}{2}} \to \mathbb{R}^6$ induced by a map $\{1,\dots,n\} \to \{1,2,3,4\}$, then y cannot lie in F_ρ by the arguments for $n = 4$ above. A simple way to construct such a y is by setting y_{ij} as above for $1 \leq i < j \leq 4$ and $y_{ij} = 1$ for $j \geq 5$ and all i. This y clearly lies in C and projects as desired. It is a vertex of C because we add sufficiently many variables at their maximal value: For $m \geq 4$ the equations $y_{1m} = 1, \dots, y_{m-1,m} = 1$ add $m-1$ active constraints to the set of active linearly independent constraints. So we have

$$6 + \sum_{m=5}^{n} (m-1) = \binom{n}{2} \tag{2.18}$$

linearly independent active constraints. □

To summarize, we have $F_\rho \subsetneq C$.

2.5.2 *Odd-Wheel Inequalities*

After the triangle inequalities, the odd-wheel inequalities are the next important set of inequalities valid for Multicut vectors. Here, show how Mod Shift's separatedness variables ρ_{ij} fail to respect them and how this can lead to unwanted minima of Mod Shift in Prop. 2.7.

Given an even number of nodes $v_0, \dots, v_m$ in a positively edge-weighted graph $G = (V, E, \rho)$ in which all edges mentioned below exist, the odd-wheel inequalities [46] are inequalities of the form

$$\sum_{i=1}^{m} \rho_{i,i+1} - \sum_{j=1}^{m} \rho_{0,j} \leq \frac{m-1}{2}, \tag{2.19}$$

where the m-th summand of the first sum is $\rho_{1,m}$. Intuitively, this means that the weight of the wheel's circumference $v_1, \dots, v_m$ with center v_0 must be sufficiently larger than the sum of the spokes' weights $\rho_{0,j}$. These inequalities must hold for any ρ_{ij} that are to encode a partition of the nodes (where $\rho_{ij} = 0$ (1) stands for i and j in the same (different) element(s) of the partition). In the presence of the triangle inequalities, the critical case for weights in $\{0,1\}$ is when all $\rho_{i,i+1}$ are 1. Then the odd-wheel inequalities ensure that the center node is, on average, sufficiently separated from consecutively different partition elements. However, they do not hold for all distances between Euclidean points, and thus Mod Shift's approximation to the Multicut problem does not guarantee them:

Proposition 2.7. *Mod Shift with $\rho(d) = \min(\frac{d}{2\beta}, 1)$ and $w(d) = \max(1 - \frac{d}{\beta}, -1)$ has an equilateral triangle of side length 2β with fourth point at its center, see Fig. 2.3, as a local minimum and convergences to this minimum for certain start configurations. It becomes a global minimum for suitably chosen weights. However, the resulting ρ-values violate an odd-wheel inequality and do not encode a crisp, well-separated clustering.*

Proof. Let the points of the equilateral triangle be denoted x_1, x_2, x_3 and its center x_0. We call this configuration C^*. We first show how it violates an odd-wheel inequality.

Applying ρ to the point configuration, we get a metric space consisting of three points with mutual distance 1 and a fourth point that has distance $\frac{1}{\sqrt{3}}$ to each of them. We compute that this configuration violates the odd-wheel inequality for the wheel consisting of the former three points with the fourth point as the wheel center:

$$3 - \frac{3}{\sqrt{3}} = (3 - \sqrt{3}) \not\leq 1 = \frac{3-1}{2}. \tag{2.20}$$

We now argue that C^* is a global solution for Mod Shift with ρ as in the setting of the proposition and for weights $w_{ij} = M \ll 0$ for $i, j \neq 0$ and $w_{0j} = 1$ for $j = 1, 2, 3$. For small enough M, any solution of Mod Shift's objective (2.4) has $\rho_{ij} = 1$ for $i, j \neq 0$, as the derivative of Mod Shift's objective by ρ_{ij} is at most $M + 3 \ll 0$ and it is clearly feasible to have all $\rho_{ij} = 1$ for $i, j \neq 0$ at the same time. Hence, in an optimal configuration, we have $d_{ij} \geq 2\beta$ for $i, j \neq 0$.

If x_1, x_2, x_3 are colinear, then an optimal position for x_0 would be on top of any of the other x_j, yielding an objective value of $3M + 2$. We obtain the same value if x_1, x_2, x_3 form a proper triangle and x_4 coincides with one of its corners. But configuration C^* yields objective value $3M + \sqrt{3}$, which is smaller. Hence, in the optimal configuration x_1, x_2, x_3 form a proper triangle Δ and the optimal position of x_4 is at the geometric median of Δ, which is in the interior of Δ, see [166]. Since ρ is non-decreasing, scaling the whole configuration down such that Δ's shortest side length is 2β does not worsen the objective function and strictly improves it if some d_{0j} is less than 2β. Otherwise $d_{0j} \geq 2\beta$ for $j = 1, 2, 3$ and the objective value is $3M + 3$ and hence superoptimal. Therefore, we know that in any optimal configuration one side of Δ has length 2β. If two sides have larger lengths, one can move the node incident to both directly towards x_0, keeping all side lengths of Δ at least 2β but decreasing the objective as soon as the distance of the moving point to x_0 gets smaller than 2β. Assume only one side length is larger than 2β, say the one incident to x_2 and x_3. Moving x_2 slightly along the circle of radius 2β around x_1 towards x_3 keeps d_{12} and d_{01} fixed but decreases d_{02} (which must be less than 2β in this situation) and hence lowers the objective value.

Thus, any locally but not globally optimal configuration has an objective value larger than C^*. We are done if C^* is locally optimal. Clearly, x_0's position as the geometric median is optimal given Δ. Any nonzero subgradient of (2.4) at $x_j, j \geq 1$ points in a direction that increases some side of Δ incident to x_j. Changing C^* according to any such subgradient yields a configuration deemed worse by the above arguments as it would increase at least one side of Δ above 2β. So C^* is the only global minimum of Mod Shift for weights as defined above.

It remains to show that C^* is the convergence point for some initial point configurations with weights given by the weight function w from the setting of the proposition. Suppose the initial configuration of four points is a scaled-down version of C^* by a factor $\alpha \in (\frac{3+\sqrt{3}}{8}, \frac{\sqrt{3}}{2})$. Then one can choose some $\varepsilon > 0$ such that

$$\alpha > \max\left(\frac{\sqrt{3}}{2}(1-\varepsilon), \frac{1}{2}\left(1 + \frac{\varepsilon}{\sqrt{3}}\right)\right) \tag{2.21}$$

which implies

$$w := w(\|x_i^0 - x_j^0\|) < -\frac{\varepsilon}{\sqrt{3}} \tag{2.22}$$

$$0 < w_0 := w(\|x_0^0 - x_i^0\|) < \varepsilon. \tag{2.23}$$

As all point distances are below 2β, (2.4) is differentiable. By symmetry, we have $\frac{\partial E(x,x^0)}{\partial x_0} = 0$ and x_0 is at its optimal position given x_1, x_2, x_3 as $w_0 > 0$. For the points x_j with $j > 0$, we have

$$\frac{\partial E(x,x^0)}{\partial x_j} = \frac{\sqrt{3}w + w_0}{2\beta} \frac{x_j - x_0}{\|x_j - x_0\|}. \tag{2.24}$$

The first fraction is negative by the estimates on w and w_0. Hence, gradient descent moves x_j directly away from x_0. If the step size of the gradient descent is small enough, each update will slightly scale up the point configuration keeping x_0 in its place. Once the side lengths of the triangle get to 2β, a local optimum is reached. □

Since we know that the odd-wheels inequalities are valid for the Multicut vectors and thus for the Multicut polytope, but not necessarily for Mod Shift's feasible set $F_\rho := \{\rho(\|x_i - x_j\|) | x_1, \dots, x_n \in \mathbb{R}^k\}$, we deduce $F_\rho \not\subset MC$. However, the $0/1$-valued points in both MC and F_ρ are exactly the Multicut vectors if ρ is as in Cor. 2.4.

2.5.3 *Mod Shift's feasible set F_ρ is not convex*

We finish this section by showing that the ρ's concavity and the non-convexity of the cone of Euclidean distances, see Section 6.3 in [42], make F_ρ non-convex such that it is also not an outer relaxation of MC.

Proposition 2.8. *If $n \geq 4$ and ρ is as in Prop. 2.3 and surjective, then F_ρ is not convex. Furthermore, it does not contain MC.*

Proof. It suffices to prove the claim for $n = 4$. For a higher number of points, consider 4 points as described below and add arbitrary other points. We will show that a certain convex combination of Multicut vectors on four points lies outside of F_ρ. Since the Multicut vectors all lie in F_ρ this shows non-convexity. As MC is convex and contains the Multicut vectors, it also contains any convex combination of them, showing that F_ρ does not contain MC. Consider four points $0, 1, 2, 3$ and the three cuts induced by separating $1, 2$ or 3 from the other three points. Let y be their mean. Then we have $y_{12} = y_{13} = y_{23} = \frac{2}{3}$ and $y_{01} = y_{02} = y_{03} = \frac{1}{3}$. Suppose for a contradiction that there were four points $x_0, \dots, x_3$ in Euclidean space, such that $\rho(\|x_i - x_j\|) = y_{ij}$ for all $0 \leq i < j \leq 3$. As ρ strictly increases from 0 to 1 (Lem. 2.2) there are unique preimages d and d' of $\frac{1}{3}$ and $\frac{2}{3}$ under ρ. Thus, x_1, x_2, x_3 form an equilateral triangle of side length d' and x_4 is of distance d to each of them. By monotonicity and concavity of ρ as well as $\rho(0) = 0$, we know that

$$\frac{1}{3} = \rho(d) \geq \frac{\rho(d')}{d'} \cdot d = \frac{2d}{3d'}, \tag{2.25}$$

or equivalently that $d' \geq 2d$. But by the triangle inequality, we also have $d' \leq 2d$. Hence, $d' = 2d$ and the points would form a scaled version of the configuration

depicted in Fig. 2.3 with $\alpha = 0.5$. By Lem. 2.5 this is not realizable in Euclidean space. Thus, we arrive at a contradiction and conclude that y does not belong to F_ρ. □

2.5.4 *Discussion of modeling cluster affiliations with ρ*

We summarize our insights on the advantages and disadvantages of modeling cluster affiliations with a separatedness function ρ and points in Euclidean space. While using ρ instead of Multicut's binary edge indicator variables y_{ij} makes Mod Shift's objective differentiable and dispenses with the need for the triangle inequalities (Lem. 2.1, Prop. 2.3), there is no guarantee that $\rho(d_{ij}^t)$ converges to 0 or 1. In fact, the second most prominent class of inequalities valid for Multicut vectors, the odd-wheel inequalities [46], are not automatically satisfied in the Mod Shift setting (Prop. 2.7). This can lead to minima of the Mod Shift objective at which $\rho(d_{ij}^t)$ is not integral and hence does not correspond to a clustering. Moreover, the set of feasible ρ-values F_ρ, which contains all Multicut vectors (2.4), is in general non-convex due to the non-convexity of ρ and of the space of Euclidean distances, see (Prop. 2.8 and Sec. 6.3 in [42]) and does not contain the convex hull of all Multicut vectors.

The non-convexity of F_ρ might challenge finding a global minimum via gradient descent. Despite these theoretical issues, we empirically find in Sec. 2.7 that Mod Shift does produce well-pronounced clusters and observe in Appendix B.4.10 that Mod Shift approximates the Multicut energy well.

2.6 MEAN SHIFT AND MOD SHIFT

Mean Shift is the best-known point-shifting clustering algorithm, particularly popular in image processing. In this section, we compare it to Mod Shift and see how Mod Shift avoids some problems of Mean Shift. For instance, Mod Shift can converge to multiple clusters even if the data density has a single mode, see Prop. 2.15, while Mean Shift necessarily yields a single cluster and hence may merge data points that are arbitrarily far apart.

2.6.1 *Update rule*

Mean Shift and Mod Shift are most similar in the fixed setting. Since the sum of all weights is constant in Mod Shift, we can replace ρ with $1 - \rho$ and minimization with maximization in problem (2.4). Thus, Mod Shift can be seen as gradient ascent on a weighted sum of kernels $1 - \rho(\|\cdot - \cdot\|)$. However, very different from Mean Shift, some weights can be negative. This makes the update rule of Mean Shift given by a fix-point iteration for the equation $\frac{\partial \mathrm{KDE}(x^t, x^0)}{\partial x_i^t} = 0$, where KDE is the kernel density estimate of the data, unsuitable for Mod Shift. Such an update rule for Mod Shift would result in

$$x_i^{t+1} := \frac{\sum_{j \neq i} \frac{\rho'(d_{ij}^t)}{d_{ij}^t} \cdot w_{ij} \cdot x_j^t}{\sum_{j \neq i} \frac{\rho'(d_{ij}^t)}{d_{ij}^t} \cdot w_{ij}}. \tag{2.26}$$

As the weights w_{ij} are allowed to be negative in Mod Shift, the denominator can be (close to) zero for points on which both attraction and repulsion act while the numerator is not. This yields arbitrarily large update steps. Therefore, we did not explore this optimization strategy further but relied on modern gradient descent optimizers for Mod Shift

2.6.2 *Adaptive Mod Shift*

Adaptive Mean Shift sends points to kernel-weighted means of the neighboring points of the current time step instead of to kernel-weighted means of nearby initial point locations as in fixed Mean Shift. It is possible to define corresponding versions of Mod Shift, too. The version of Sec. 2.4.1 can be viewed as "fixed" since it derives the weights from the initial point positions and thus keeps them constant during optimization. Due to its relation to the Multicut problem and because of the importance it gives to the process that generated the initial point positions (e.g., a neural network), we prefer the fixed version and refer to it when speaking of Mod Shift unless specified otherwise.

In the adaptive version of Mod Shift, the weights depend on the current point positions instead of the initial point positions as in fixed Mod Shift. However, naively doing so, i.e., considering the problem

$$\operatorname*{argmin}_{x \in \mathbb{R}^{n \times k}} \sum_{i<j} \rho(\|x_i - x_j\|) \cdot w(\|x_i - x_j\|) \tag{2.27}$$

is not ideal. Firstly, the distance threshold below which two points attract and above which they repel each other does not correspond directly to w's zero-crossing β anymore, making it harder to interpret.

Lemma 2.9. *Let $\rho : [0, \infty) \to [0, 1]$ be increasing and both differentiable and strictly increasing on $[0, 2\beta]$ for some $\beta > 0$ and let $w : [0, \infty) \to [-1, 1]$ be positive below β and negative above β as well as strictly decreasing and differentiable on $[0, 2\beta]$ with negative derivative at β. Then the naive adaptive Mod Shift version (2.27) repels points with distance in $[\gamma, 2\beta]$ for some $\gamma < \beta$. If ρ and w are additionally concave below β, then we can choose γ such that the naive adaptive Mod Shift version also attracts points with distance below γ.*

Proof. We compute the derivative of the objective function in Eq. (2.27) with respect to the distance $d := \|x_i - x_j\|$ of some pair of points x_i and x_j

$$(\rho \cdot w)'(d) = \rho'(d)w(d) + \rho(d)w'(d). \tag{2.28}$$

The first summand is positive below β and negative in $(\beta, 2\beta]$. The second summand is non-positive everywhere and strictly negative at β. Thus, their sum is negative on $[\gamma, 2\beta]$ for some $\gamma < \beta$. Points with a distance between γ and 2β repel each other.

For the second part, we observe that by ρ's concavity the term $\rho'(d)w(d)$ is strictly decreasing on $[0, \beta]$ and by the concavity of w the negative term $\rho(d)w'(d)$ is decreasing on $[0, \beta]$. Hence, $(\rho \cdot w)'$ is strictly decreasing on $[0, \beta]$ and thus has at most one root. Choosing γ to be this root or, if none exists, to be 0 makes $(\rho \cdot w)'$ positive on $[0, \gamma)$, so that points with distance below γ attract each other. □

In addition to a less interpretable β parameter, the naive adaptive Mod Shift version has an undesirable unique global minimum.

Lemma 2.10. *Let $\rho : [0,\infty) \to [0,1]$ be increasing and $w : [0,\infty) \to [1,-1]$ be decreasing with negative infimum. Then the unique global minimum (or infimum) value of the naive adaptive Mod Shift objective function in Eq. (2.27) is attained when all points are far apart from each other.*

Proof. We write the proof as if neither ρ nor w change beyond some value 2β, but if either of them does not become constant, the same proof applies with "minimum" replaced by "infimum" or "maximum" by "supremum". The term $\rho(d) \cdot w(d)$ is lower bounded by $\max_\rho \cdot \min_w$, where $\max_\rho$ is the maximum of ρ and $\min_w$ is w's minimum, which is negative. Hence, the objective function in Eq. (2.27) becomes globally minimal when for all pairs of points ρ is maximal and w minimal. That is, if all pairs of points are far apart from each other according to the monotonicity of ρ and w. □

Note that the conditions in Lem. 2.9 and 2.10 are met for all functions ρ and w in Fig. 2.2.

Like naive adaptive Mod Shift, both fixed and adaptive Mean Shift's objectives also have single global optima. But here, all points are collapsed to a single cluster. However, fixed Mean Shift's update rule does not necessarily converge to this trivial solution. We will see below that a better version of adaptive Mod Shift will have meaningful global optima.

The naive Mod Shift version is clearly not a good choice. The problem is that it only considers the interaction between clusters, which always benefits from decreasing the weight function. A better notion of "adaptive" Mod Shift avoids this focus on repulsion and corresponds to a different but equivalent formulation of the fixed version. We put $\tilde{\rho} := 2\rho - 1$ and consider the problem

$$\operatorname*{argmin}_{x \in \mathbb{R}^{n\times k}} \sum_{i<j} \tilde{\rho}(\|x_i - x_j\|) \cdot w(\|x_i^0 - x_j^0\|). \tag{2.29}$$

Intuitively speaking, we do not only try to minimize attraction and maximize repulsion between clusters, as in (2.4), but also to maximize attraction and minimize repulsion within a cluster. If the weights depend on the initial distances, this is equivalent to our fixed version of Mod Shift.

Lemma 2.11. *The optimization problem (2.29) is equivalent to fixed Mod Shift (2.4).*

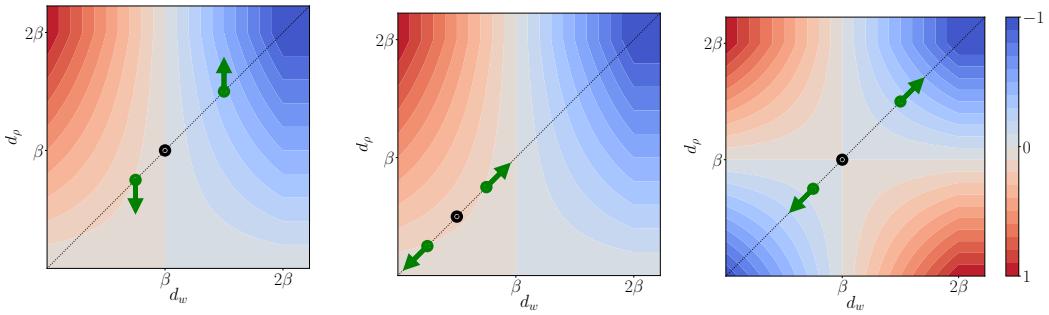
Proof. We have

$$\sum_{i<j} \tilde{\rho}(\|x_i - x_j\|) \cdot w(\|x_i^0 - x_j^0\|) = 2\sum_{i<j} \rho(\|x_i - x_j\|) \cdot w(\|x_i^0 - x_j^0\|) - \sum_{i<j} w(\|x_i^0 - x_j^0\|). \tag{2.30}$$

The last term is a constant, only depending on the initial point positions. The first term on the right-hand side is just twice the objective function of fixed Mod Shift (2.4) □

In the Multicut literature, this more symmetric viewpoint and its equivalence to only considering intra- or only inter-cluster interaction are well-known. Motivated by the equivalent form of fixed Mod Shift in Eq. (2.29), we define the following optimization problem for "adaptive" Mod Shift

$$\operatorname*{argmin}_{x \in \mathbb{R}^{n\times k}} \sum_{i<j} \tilde{\rho}(\|x_i - x_j\|) \cdot w(\|x_i - x_j\|). \tag{2.31}$$



(a) fixed, at time t: $d_\rho = \|x_i^t - x_j^t\|$, $d_w = \|x_i^0 - x_j^0\|$

(b) naive-adaptive, at time t: $d_\rho = \|x_i^t - x_j^t\| = d_w$

(c) adaptive, at time t: $d_\rho = \|x_i^t - x_j^t\| = d_w$

Figure 2.4: Contour plots of Mod Shift's objective function for a single pair of points. ρ and w are the simple rectified linear functions from 2.4.2 and 2.4.3. The distances d_w and d_ρ on which the weight w and the separatedness ρ depend are coupled differently for fixed, naive-adaptive, and adaptive Mod Shift. To illustrate these differences, we plot them on different axis and describe how they can change as the different Mod Shift versions update. Initially, at $t = 0$, the distance of the pair of points lies on the diagonal as current and initial point positions coincide. Two initial situations with update directions are depicted as green dots with arrows in each subfigure. In fixed Mod Shift only the distance for the separatedness ρ gets updated as only the separatedness depends on the current point positions. Hence, updates move along vertical lines in the left plot. In adaptive Mod Shift the distance for both ρ and w changes jointly as they both depend on the current point positions. Therefore, the updates follow the diagonal. The two points attract each other in any situation to the left of the black dot. In situations to the right of the black dot, the points repel each other. Note that in (b) this threshold differs from $d_w = \beta$. Figure best viewed in color.

This version of adaptive Mod Shift does not suffer from the issues in Lems. 2.9 and 2.10.

Lemma 2.12. *Let $\rho : [0,\infty) \to [0,1]$ differentiable with positive derivative on $[0,2\beta)$ for some $\beta > 0$ at which $\rho(\beta) = 0.5$ and let $w : [0,\infty) \to [-1,1]$ be positive below β and negative above β as well as differentiable on $[0,2\beta)$ with negative derivative. Then points with a distance below β attract each other and points with a distance in $(\beta,2\beta)$ repel each other under adaptive Mod Shift (2.31).*

Proof. As in the proof of Lem. 2.9, we analyze the derivative of the term $\tilde{\rho} \cdot w$ by the distance d of some pair of points

$$(\tilde{\rho} \cdot w)'(d) = 2\rho'(d)w(d) + (2\rho(d) - 1)w'(d). \tag{2.32}$$

By the assumptions, both summands are positive on $[0,\beta)$ and negative on $(\beta,2\beta)$. □

So with the additional rescaling of ρ, such that it equals 0.5 at β, adaptive Mod Shift 2.31 retains the interpretability of the parameter β: Points closer than β attract, while points further apart than β repel each other.

We depict the regimes of attraction and repulsion of fixed, naive adaptive and adaptive Mod Shift in Fig. 2.4.

Having defined the proper notion of adaptive Mod Shift, we move on to its analysis. The adaptive version of Mean Shift will eventually collapse all points when using a kernel with infinite support, such as the RBF kernel, see Thm. 3 of [31]. This is clearly undesirable, and one advantage of Mod Shift, both fixed and adaptive, is that it can produce stable multi-cluster solutions by balancing attractive and repulsive interaction. We have discussed the minima of fixed Mod Shift in Sections 2.4 and 2.5. Now, we focus on the minima of adaptive Mod Shift.

We show in Lem. 2.13 that the partitions of the dataset are in one-to-one correspondence with the global minima of adaptive Mod Shift. Whenever we speak of minima, we also include infima in the case that ρ reaches 1 only asymptotically.

Lemma 2.13. *Let $\rho : [0,\infty) \to [0,1]$ be an increasing function with $[0,1) \subset \mathrm{im}(\rho)$ and let $w : [0,\infty) \to [-1,1]$ be decreasing and surjective with $w^{-1}(0) = 0$ and $w^{-1}(-1) = [2\beta,\infty)$ for some $\beta > 0$. Then the global minima of adaptive Mod Shift (2.31) correspond to the partitions of data points and vice versa.*

Proof. We have $\overline{\mathrm{im}(\tilde{\rho})} = [-1,1]$ and $\mathrm{im}(w) = [-1,1]$. Therefore, the term $\tilde{\rho}_{ij}^t w_{ij}^t$ gets minimal for $\tilde{\rho}_{ij}^t = 1$ and $w_{ij}^t = -1$ or for $\tilde{\rho}_{ij}^t = -1$ and $w_{ij}^t = 1$. The first case corresponds to a distance between x_i^t and x_j^t of at least 2β; the second case corresponds to $x_i^t = x_j^t$. Hence, global minima of adaptive Mod Shift correspond to separatedness values $\rho(d_{ij}^t) \in \{0,1\}$, i.e., to a valid partition, see Prop. 2.3.

Conversely, realizing a partition of the data points by separating different clusters sufficiently far and placing points of the same cluster on the same location induces $\rho(d_{ij}^t) \in \{0,1\}$ and thus a global minimum of adaptive Mod Shift. □

We have exhibited that partitions correspond to global minima of adaptive Mod Shift, but there might be additional local minima that do not translate to pronounced clusters.

2.6.3 *Mean Shift suffers more from long-range merges than Mod Shift*

Mean Shift's focus on the data density makes identifying multiple clusters difficult if the density does not indicate them. In particular, points can move arbitrarily far towards a region of high density, see Lems. 2.14 and 2.16. In contrast, Mod Shift's long-range repulsion provides a bias against such distant merges, see Props. 2.15 and 2.17.

We know that both fixed Mean Shift and Mod Shift separate all data points from each other for sufficiently small scale parameters. However, to illustrate that Mean Shift suffers more from arbitrarily far movement of points than Mod Shift, we thus consider a dataset given as a density function f rather than discretely. This way, choosing a scale parameter smaller than the smallest distance between data points becomes impossible. Moreover, this limit case permits an analytical treatment.

We appreciate that both Mod Shift and Mean Shift are designed for finite datasets, which might be *sampled* from a continuous density. Nevertheless, we will treat the data as (and not sampled from) a continuous density in the following. This can be interpreted as an infinite sample and can provide us with some intuition on the different behavior of Mean Shift and Mod Shift on large finite datasets. We confirm that the intuition gained from the infinite limit case does carry over to

the finite regime by empirically validating the formal results in this section. For instance, this continuous setting has been used in Section 2 of [23] for the convergence speed analysis of Mean Shift. Furthermore, clustering continuous densities is a well-established research field in physics, for instance, for modeling phase separation, see for instance [56] and references therein.

We now describe what we mean by Mod Shift if the data is a continuous density function f. Given continuous ρ and w as before, the objective of Mod Shift becomes finding a measurable map $\vartheta : \operatorname{supp}(f) \to \mathbb{R}^k$ that minimizes the energy

$$E(\vartheta, f) = \int_x \int_y \rho(\|\vartheta(x) - \vartheta(y)\|) \cdot w(\|x - y\|) \cdot f(x) f(y) dx dy. \tag{2.33}$$

The interpretation is that data density starting out at x moves to $\vartheta(x)$. We typically consider functions ϑ with finite image, i.e., finite partitions of the dataset. Mod Shift's optimization strategy still is gradient-descent: It defines a sequence of maps $\vartheta_t : \operatorname{supp}(f) \to \mathbb{R}^k$ mapping the density $f(x)$ to its location $x^t := \vartheta_t(x)$ at time t. Given the configuration at time $t-1$, the next map will be given by

$$\vartheta_t(x) = x^{t-1} - \text{learning rate} \cdot \frac{\partial E(\vartheta_{t-1}, f)}{\partial x^{t-1}}, \tag{2.34}$$

where ϑ_0 is the identity map. In such an update step, we say that the data density $f(x)$ is shifted from $\vartheta_{t-1}(x)$ to $\vartheta_t(x)$. We only consider fixed Mod Shift in Sec 2.6.3.

In the following, we will illustrate cases in which fixed and adaptive Mean Shift both merge arbitrarily far points, but Mod Shift does not. The critical property of Mod Shift that prevents very long merges is its distance-based repulsion. Distant points repel each other in Mod Shift avoiding merges between such points. In contrast, there is no repulsion in Mean Shift. Thus, a sufficiently high data density can attract points arbitrarily far away.

Single Mode Datasets

Let the dataset be given by a differentiable probability density function $f : \mathbb{R} \to \mathbb{R}_{\geq 0}$ with a unique extremum, which is an isolated maximum.

Lemma 2.14. *Fixed and adaptive Mean Shift with an RBF kernel have a single optimum for f for any bandwidth. This optimum corresponds to a single cluster.*

Proof. Thm. 3 of [31] shows the statement for the adaptive version. The fixed Mean Shift version performs gradient ascent with implicitly defined step-size on the density $h = f * g$ given as the convolution of f with an RBF kernel g. By scale-space theory, e.g., Thm. 5 in [100], h has a single extremum, an isolated maximum, like f. Hence, the entire dataset converges to the single mode. □

In particular, data density can merge with arbitrarily distant parts of data density in Mean Shift, limited only by the support of f. Note that this is independent of the bandwidth of the RBF kernel g. The argument even holds for zero bandwidth, where $h = f$.

In contrast, it is not optimal for Mod Shift to collapse the entire dataset.

Proposition 2.15.

1. *Let w be a weight function with a scale parameter β such that $w(d) = \phi(d/\beta)$ for a continuous, decreasing function ϕ independent of β, which is zero at β and negative above β. Then for sufficiently small β collapsing the entire density to a single point is suboptimal for Mod Shift.*

2. *Let w and ρ be continuous and point-symmetric on $[0, 2\beta]$ around $(\beta, 0)$ and $(\beta, 0.5)$, respectively. As usual, assume that ρ is increasing and $\rho(0) = 0$ and that w decreases. For small enough β, the initial update of Mod Shift moves data density away from the mode.*

Proof. 1. Consider some $x_0 \in \mathbb{R}$ such that neither $F(x_0)$ nor $1 - F(x_0)$ is 0, where F is the cumulative probability density function of f. First, we show that for sufficiently small β splitting the dataset at x_0 yields a negative and thus better objective value for Mod Shift than converging it to a single point, which would result in an objective value of 0. Suppose

$$\vartheta(x) = \begin{cases} a, \text{ if } x \leq x_0 \\ b, \text{ if } x > x_0 \end{cases} \tag{2.35}$$

for some $a < b$ and suppose further that $\rho(b - a) = 1$. We omit the potential limit $\rho \to 1$ for simplicity of exposition. Now, we show that for sufficiently small β, this ϑ, which splits the dataset into two sets at x_0, yields a lower objective value than collapsing the entire dataset into a single point. We prove this by observing that as we decrease β, the attraction between points below and above x_0 tends to zero, while the repulsion between these sets is positive, see Fig. 2.5 for an illustration. Let $1 > \varepsilon > 0$ be small enough that we can choose $\delta > 1$ such that $\phi(\delta) < -\varepsilon$. Then for any β we have $w(d) < -\varepsilon$ for $d > \beta\delta$. For x and x' on the same side of x_0, we have

$$\rho(\|\vartheta(x) - \vartheta(x')\|) = \rho(0) = 0, \tag{2.36}$$

while for x and x' on different sides of x_0, we have

$$\rho(\|\vartheta(x) - \vartheta(x')\|) = \rho(b - a) = 1. \tag{2.37}$$

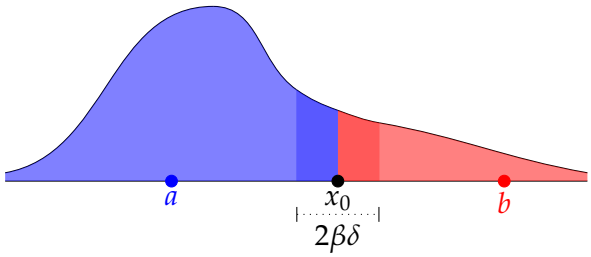


Figure 2.5: Illustration of why for sufficiently small β it is suboptimal for Mod Shift to collapse even a density with a single mode. Indeed, it is better in terms of Mod Shift's objective to split at an arbitrary point x_0. We consider the situation in which the blue (red) shaded area has converged to point a (b) such that a and b are sufficiently far away that $\rho(b-a) \approx 1$. The density in the lightly shaded areas has a distance of at least $\beta\delta$ to the area of the other color. The number δ is chosen such that this induces some small repulsion $\varepsilon > 0$ independent of β. Attraction can only exist between the darker shaded parts. But for $\beta \to 0$, this area vanishes so that eventually repulsion dominates and the split at x_0 becomes favorable to collapsing the entire density. For more details confer Prop. 2.15.

Putting $\psi(x, x') := w(\|x - x'\|) f(x') f(x) dx' dx$ the objective value of this configuration is hence

$$
\begin{aligned}
&\int_{-\infty}^{x_0} \int_{x_0}^{\infty} \psi(x, x') \qquad (2.38)\\
&= \int_{-\infty}^{(x_0-\beta\delta)} \int_{x_0}^{\infty} \psi(x, x') + \int_{(x_0-\beta\delta)}^{x_0} \int_{(x_0+\beta\delta)}^{\infty} \psi(x, x') + \int_{(x_0-\beta\delta)}^{x_0} \int_{x_0}^{(x_0+\beta\delta)} \psi(x, x') \\
&\overset{(i)}{<} -\varepsilon\big(F(x_0-\beta\delta)(1-F(x_0)) + (F(x_0)-F(x_0-\beta\delta))(1-F(x_0+\beta\delta))\big) \\
&\quad + \int_{(x_0-\beta\delta)}^{x_0} \int_{x_0}^{(x_0+\beta\delta)} \psi(x, x') \\
&\overset{(ii)}{<} -\varepsilon\big(F(x_0-\beta\delta)(1-F(x_0)) + (F(x_0)-F(x_0-\beta\delta))(1-F(x_0+\beta\delta))\big) \\
&\quad + w(0)(F(x_0)-F(x_0-\beta\delta)) \cdot (F(x_0+\beta\delta)-F(x_0)) \\
&\xrightarrow[\beta\to 0]{(iii)} -\varepsilon F(x_0)(1-F(x_0)) < 0.
\end{aligned}
$$

At (i) we used the fact that $w(d) < -\varepsilon$ for $d > \beta\delta$. At (ii) we used that w is non-increasing. The limit process in (iii) follows from the fact that F is continuous. The calculation shows that whenever we split the data at some point x_0 in the interior of the support of f and choose the scale parameter sufficiently small, this yields a better objective value for Mod Shift than collapsing everything to a single cluster.

2. Consider some point x_0 of positive density but with negative derivative, which is at least 2β to the right of the mode. Such a point exists for sufficiently small β. We show that the density at x_0 gets moved towards an area of lower density in the initial Mod Shift step. Here, we assume that w and ρ are point-symmetric about

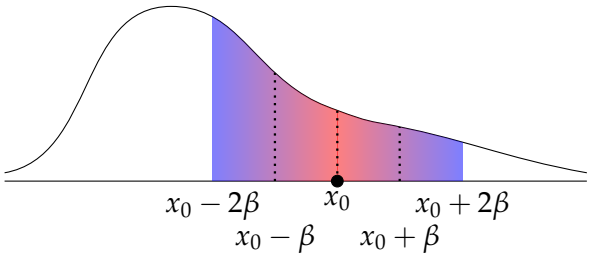


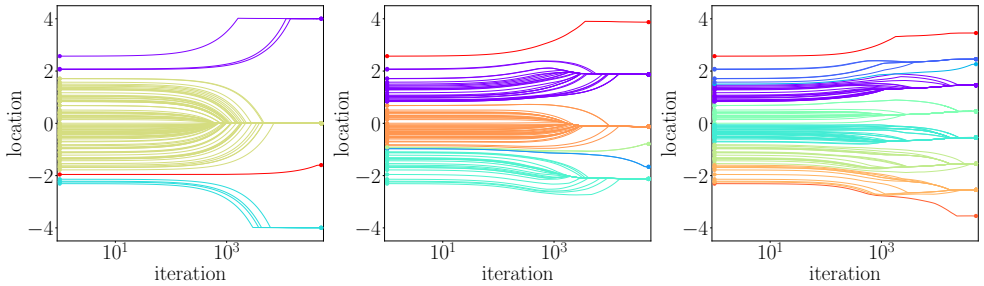
Figure 2.6: Illustration for why a point over whose 2β-ball the data density is monotonous initially gets shifted down the density by Mod Shift for point-symmetric ρ and w, both clipped at 2β. The color encodes the signed magnitude of the force exerted on x_0 by the other points. As ρ is clipped at 2β, only density within a ball of radius 2β sends a gradient to x_0. The density between $x_0 - 2\beta$ and $x_0 - \beta$ exerts repulsion on x_0. Due to the downward slope of the density, this outweighs the attraction of the density between $x_0 - \beta$ and x_0. As a result, the net force from density on the left of x_0 points to the right. Similarly, attraction outweighs repulsion for density on the right, and the resulting force on x_0 also points towards the right. In total, the initial update will move x_0 towards a region of lower density. For more detail confer Prop. 2.15.

their values at β on $[0, 2\beta]$. As ρ maps to $[0, 1]$, this implies that ρ is constant at value 1 beyond 2β. The gradient of $E(\vartheta_0, f)$ at x_0^0 is

$$\begin{aligned}
\frac{\partial E(\vartheta_0, f)}{\partial x_0^0} &= \int_{x<x_0} \rho'(\|x - x_0\|) w(\|x - x_0\|) f(x) f(x_0) \frac{x_0 - x}{\|x_0 - x\|} dx \\
&\quad + \int_{x>x_0} \rho'(\|x - x_0\|) w(\|x - x_0\|) f(x) f(x_0) \frac{x_0 - x}{\|x_0 - x\|} dx \\
&\overset{(a)}{=} f(x_0) \int_0^\beta \rho'(\beta - t) w(\beta - t) \cdot \big(f(x_0 - \beta + t) - f(x_0 - \beta - t)\big) dt \\
&\quad - f(x_0) \int_0^\beta \rho'(\beta - t) w(\beta - t) \cdot \big(f(x_0 + \beta - t) - f(x_0 + \beta + t)\big) dt \\
&\overset{(b)}{<} 0.
\end{aligned}$$

In (a), we reparametrized the integral by separately ranging over attractive and repulsive forces of the same magnitude above and below x_0. Note that we used the point symmetry of $\rho' w$ and that $\rho'(\|x - x_0\|) = 0$ for points further apart from x_0 than 2β. We used that f is decreasing on $[x_0 - 2\beta, x_0 + 2\beta]$ in b). So there is more density below x_0 that repels it with a certain magnitude than density that attracts it with the same magnitude and vice versa for points above x_0. See Fig. 2.6 for an illustration. Mod Shift updates each point by gradient descent on $E(\vartheta_0, f)$ and hence increases x_0 in the first step. So it moves x_0 away from the mode and down the slope of f. By symmetry, a point at which f increases gets shifted to the left. □

Note that the point symmetry assumption of Prop. 2.15 2. together with the concavity that we often assume for ρ implies that ρ is the rectified linear function $\rho(d) = \min(d/\beta, 1)$.



(a) $\beta = 2$ (b) $\beta = 1$ (c) $\beta = \frac{1}{2}$

Figure 2.7: Mod Shift on 2048 samples of a standard normal distribution with rectified linear ρ and w. The trajectories of 100 subsampled points are depicted over 50000 iterations. The points and their trajectories are colored according to the clustering. Although the data distribution has a single mode, Mod Shift produces clusters mainly dependent on the distance between data points. Similar to Prop. 2.15 2., points that start out at least 2β away from the mode at location 0 are moved away from the mode in the first iterations.

Fig. 2.7 backs the above Prop. 2.15 empirically for the rectified linear versions of ρ and w. Let us summarize its takeaways. The main convergence points are separated by 2β, as for larger distance ρ sends no gradient. A very low learning rate is chosen to illustrate Mod Shift's behavior in detail, which is why some points on borders of clusters did not converge yet. As Prop. 2.15 predicts, points x for which $[x - 2\beta, x + 2\beta]$ does not contain the mode at 0, initially move away from the mode and usually end up in different clusters than the mode. In 2.7a and 2.7b, there is a cluster centred at the mode, while in 2.7c the distribution seems to be split at the mode. This shows that distance has a stronger influence on the clustering than point density.

Uniform Data

Similar to the case of a dataset with a single, isolated mode above, we now compare Mean Shift's and Mod Shift's behavior on a uniform density dataset.

Again, both fixed and adaptive Gaussian Mean Shift collapse the whole dataset irrespective of the selected bandwidth and thus can lead to arbitrarily far merges.

Lemma 2.16. *For the dataset $[0, 1]$ with uniform density both fixed and adaptive Mean Shift with an RBF kernel of any bandwidth $\sigma > 0$ have a single minimum, in which case the entire dataset is contracted.*

Proof. Since the RBF kernel has infinite support, the result for adaptive Mean Shift is just a continuous version of Thm. 3 of [31]. For fixed Mean Shift, which performs gradient ascent on the "shadow" kernel density estimate, it suffices to show that this kernel density estimate has a single critical point, which is a maximum. The kernel density estimate is

$$\mathrm{KDE}(x) = \int_0^1 H(\|x - y\|)dy. \tag{2.39}$$

By Thm. 2 of [31] we know that $H = \frac{1}{\pi\sqrt{2\sigma}} \exp(-\frac{z^2}{2\sigma^2})$ is a Gaussian as well. Thus, we have

$$\begin{aligned} \mathrm{KDE}(x) &= \int_0^1 \frac{1}{\pi\sqrt{2\sigma}} \exp\left(-\frac{(x-y)^2}{2\sigma^2}\right) dy \\ &= \frac{1}{\pi\sqrt{2\sigma}} \cdot \left(\int_{-\infty}^{x} \exp\left(-\frac{z^2}{2\sigma^2}\right) dz - \int_{-\infty}^{x} \exp\left(-\frac{(z-1)^2}{2\sigma^2}\right) dz \right). \end{aligned} \tag{2.40}$$

Taking derivatives, we obtain

$$\begin{aligned} \mathrm{KDE}'(x) &= \frac{1}{\pi\sqrt{2\sigma}} \cdot \left(\exp\left(-\frac{x^2}{2\sigma^2}\right) - \exp\left(-\frac{x^2}{2\sigma^2}\right) \exp\left(-\frac{x}{\sigma^2}\right) \exp\left(\frac{1}{2\sigma^2}\right) \right) \\ &= -\frac{1}{\pi\sqrt{2\sigma}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \cdot \left(1 - \exp\left(-\frac{x}{\sigma^2}\right) \exp\left(\frac{1}{2\sigma^2}\right) \right), \end{aligned}$$

which is only zero for $x = 0.5$. This only critical point is a maximum as $\mathrm{KDE}(x)$ integrates to one and hence must decay to zero for large $|x|$. □

The following proposition strengthens the claim that Mod Shift tends to separate points according to their distance by showing that the optimal partition of a uniform line is to break it up into equisized intervals.

Proposition 2.17. *Consider the dataset $[0,1]$ with uniform density. Let w be point symmetric about $(\beta, 0)$ on $[0, 2\beta]$, strictly decreasing on this interval, everywhere pointwise non-decreasing in β and constant on $[2\beta, \infty)$. Consider the measurable maps $\vartheta : [0,1] \to \mathbb{R}$ with countable image consisting of isolated points at least d apart, where $\rho(d) = 1$[1]. Moreover, assume that the preimage of ϑ induces a partition P of $[0,1]$ consisting of Lebesgue measurable sets with a boundary of zero measure.*[2] *For such a ϑ Mod Shift's energy at ϑ is*

$$E(P) := E(\vartheta, u) = \int_{0 \le x < y \le 1} \rho_P(x,y) w(\|x-y\|) d(x,y), \tag{2.41}$$

with $\rho_P(x,y) = \mathbb{1}(\nexists p \in P : x, y \in p)$.

1. *For all $\beta \neq 0.5$, there is an up to null sets unique optimal partition among the considered partitions. It is a finite equidistant subdivision of $[0,1]$ into intervals.*
2. *For each optimal partition P in 1., its number of intervals $|P|$ satisfies $\frac{1}{2|P|} \le \beta \le \frac{1}{|P|}$ for any $\beta \neq 0.5$.*

Proof. We recall that for fixed β, the sum of all weights is constant, see Sec. 2.6.2. Hence, we can and will freely switch between minimizing inter-partition-element interaction as in (2.41) and maximizing intra-partition-element interaction. Moreover, only non-null sets contribute to the objective, so without loss of generality, we only consider partitions all but one of whose elements have strictly positive measure and do not contain isolated points. Note that we can assume the optimal partition to contain elements of positive measure as we assume it to be countable.

1 We omit the potential limit $\rho \to 1$ for ease of exposition

2 This is a real restriction as the "fat" Cantor set is a closed Lebesgue measurable set with empty interior; hence it is its own boundary. Nevertheless, it has positive measure.

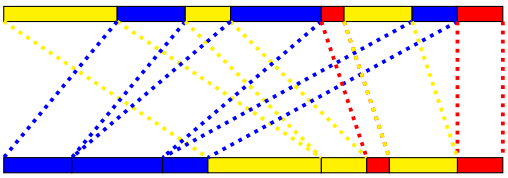


Figure 2.8: Schematic illustration of the map ψ_p in the proof of Prop. 2.17 for p being the blue partition element. Assuming the blue blocks are open, wherever two blue lines meet, ψ_p misses a point, and wherever a red and a yellow line meet on the lower line, ψ_p maps two points to the same image. The distance between several elements of p has decreased, and the distances between members of the other partition elements have not increased. Hence, the energy of the upper partition is higher than that of the lower partition. Consequently, an optimal partition consists of connected intervals.

1. Let λ be the Lebesgue measure. Suppose we have a partition element $p \in P$ that is separated by some non-null set, i.e., there is a set S with $\lambda(S) > 0$ and $\lambda(S \cap p) = 0$ such that $\lambda([0, \inf S] \cap p) > 0$ and $\lambda([\sup S, 1] \cap p) > 0$. We show that we can replace P by a partition $\bar{P}$ of smaller objective value by "contracting" p, as depicted in Fig. 2.8. In order to do so, we have to exclude some measure-theoretic pitfalls. Namely, we assume that $\lambda(\partial p) = 0$. Then we know that the interior of p differs from p only by a null set, and we can assume p to be open without loss of generality. It is thus a countable union of disjoint open intervals $p = \sqcup_{i \in \mathbb{N}}(k_i, l_i), k_i < l_i$ for all i (possibly intersected with $[0, 1]$).

Consider the function

$$\psi_p : [0,1] \to [0,1], \tag{2.42}$$
$$x \mapsto \begin{cases} \psi_{p,1}(x) := \lambda([0, x) \cap p), & \text{if } x \in p \\ \psi_{p,2}(x) := x + \lambda((x, 1] \cap p), & \text{if } x \notin p, \end{cases}$$

which we will use to contract p. We show that this function is almost everywhere bijective in the sense that $\cup\{\psi_p^{-1}(x) | x \in [0,1], |\psi_p^{-1}(x)| > 1\}$ and $[0,1] \setminus \mathrm{im}(\psi_p)$ are both null sets. In fact, we will see that injectivity can only fail on the boundary of p, and we can only miss a countable and thus null set, which again corresponds to (parts) of the boundary of p, see Fig. 2.8.

We start with the almost injectivity. Clearly, ψ_p maps elements of p below or equal to $\lambda(p)$ and elements of its complement p^c above or equal to $\lambda(p)$. Let $x \leq y$ be both in p with $\psi_p(x) = \psi_p(y)$, i.e., $\lambda((x, y) \cap p) = 0$. But p is open and contains x and y, so $x = y$. Only a single point of p, namely 1 if it is contained in p, could be mapped to $\lambda(p)$. So the only non-injectivity we have to consider comes from

$$x + \lambda((x, 1] \cap p) = y + \lambda((y, 1] \cap p) \tag{2.43}$$

for some $x \leq y$ in p^c. But this means that

$$y - x = \lambda((x, y) \cap p). \tag{2.44}$$

Hence, x must be a closure point of the k_i's and y a closure point of the l_i's. So, both x, y are in the boundary of p, which was assumed to be of measure zero.

Before we turn to the almost surjectivity, we remark that the function

$$\psi_{p,1} : [0,1] \to [0, \lambda(p)],\ x \mapsto \lambda([0,x) \cap p) \tag{2.45}$$

is a continuous, non-decreasing function which takes the values $\lambda([0,k_i) \cap p), i \in \mathbb{N}$ on p^c. Continuity is clear since changing some value x by at most ε can change the value of $\psi_{p,1}(x)$ by at most ε. Similarly,

$$\psi_{p,2} : [0,1] \to [\lambda(p), 1],\ x \mapsto x + \lambda((x,1] \cap p) \tag{2.46}$$

is also continuous, non-decreasing and takes the values $\lambda(p) + \lambda([0,k_i) \cap p^c), i \in \mathbb{N}$ on p.

For almost surjectivity, this implies that

$$[0, \lambda(p)) \setminus \operatorname{im}(\psi_p) = \{\lambda([0,k_i) \cap p) | i \in \mathbb{N}\}, \tag{2.47}$$

which is of measure zero. The closed set p^c contains its infimum and supremum. So,

$$\psi_p(\min p^c) = \psi_{p,2}(\min p^c) = \lambda(p) \text{ and} \tag{2.48}$$
$$\psi_p(\max p^c) = \psi_{p,2}(\max p^c) = 1. \tag{2.49}$$

By the continuity of $\psi_{p,2}$ and the intermediate value theorem, the whole interval $[\lambda(p), 1]$ clearly is in the range of $\psi_{p,2}$. Finally, note that if $\psi_{p,2}$ takes some value y at $x \in (k_i, l_i) \subset p$, then we also have

$$\psi_p(k_i) = \psi_{p,2}(k_i) = y, \tag{2.50}$$

which shows that $[\lambda(p), 1] \subset \operatorname{im}(\psi_p)$.

We show that ψ_p is measurable as a function from the Lebesque space $[0,1]$ to itself and preserves the measure. First, we show measurability. We can write

$$\psi_p = \psi_{p,1} \cdot \mathbb{1}_p + \psi_{p,2} \cdot \mathbb{1}_{p^c}. \tag{2.51}$$

The functions $\psi_{p,1}$ and $\psi_{p,2}$ are continuous and thus measurable. Since p is measurable, so is p^c and thus are their indicator functions $\mathbb{1}_p$ and $\mathbb{1}_{p^c}$. Hence, ψ_p is measurable as the sum of products of measurable functions.

Next, we show that ψ_p is measure-preserving. For $(a,b) \subset [0,1]$ we have

$$\lambda(\psi_p(a,b)) = \lambda(\psi_p((a,b) \cap p)) + \lambda(\psi_p((a,b) \cap p^c)), \tag{2.52}$$

which we can separately compute as follows

$$\begin{aligned}
\lambda(\psi_p((a,b) \cap p)) &= \sum_i \lambda\big(\psi_{p,1}((a,b) \cap (k_i, l_i))\big) \\
&= \sum_i \lambda\big(\psi_{p,1}((\max(a,k_i), \min(b,l_i)))\big) \\
&= \sum_i \lambda\Big(\big(\lambda([0, \max(a,k_i)) \cap p), \\
&\qquad\qquad \lambda([0,\max(a,k_i)) \cap p) + \min(b,l_i) - \max(a,k_i)\big)\Big) \\
&= \sum_i \lambda\big((a,b) \cap (k_i,l_i)\big) \\
&= \lambda((a,b) \cap p).
\end{aligned} \tag{2.53}$$

Since $\psi_{p,2}$ maps $[0,1]$ continuously and non-decreasingly onto $[\lambda(p),1]$, i.e., by the discussion before Eq. (2.50), we have that

$$\begin{aligned}\lambda(\psi_p((a,b)\cap p^c)) =&\lambda\big(\psi_{p,2}((a,b))\big)\\ =&\lambda\big((\psi_{p,2}(a),\psi_{p,2}(b))\big)\\ =&\lambda\big((a+\lambda((a,1]\cap p),b+\lambda((b,1]\cap p))\big)\\ =&\lambda((a,b))-\lambda((a,b)\cap p).\end{aligned} \tag{2.54}$$

Combining Eqs. (2.53) and (2.54), we get that ψ_p preserves the measure of sets. By the almost bijectivity of ψ_p, the set

$$\bar{P}=\{\psi_p(p')|p'\in P\} \tag{2.55}$$

is a partition of $[0,1]$ (up to null sets) with

$$\rho_P(\psi_p^{-1}(x),\psi_p^{-1}(y))=\rho_{\bar{P}}(x,y) \tag{2.56}$$

for almost all x,y. We write ψ_p^{-1} in abuse of notation, as ψ_p is not actually invertible. Since it is almost invertible, this only creates problems on null sets, which we can ignore. We will show that the objective value of Mod Shift on $\bar{P}$ is strictly lower than that of P. By the construction of ψ_p, the distance of any two points $x<y$ in the same partition element of P does not increase by applying ψ_p. Indeed,

$$\begin{aligned}y-x&=\psi_p(y)-\psi_p(x)+\lambda((x,y)\cap p^c)\\ &\geq\psi_p(y)-\psi_p(x)\text{ if }x,y\in p\text{ and}\end{aligned} \tag{2.57}$$

$$\begin{aligned}y-x&=\psi_p(y)-\psi_p(x)+\lambda((x,y)\cap p)\\ &\geq\psi_p(y)-\psi_p(x)\text{ if }x,y\in p^c.\end{aligned} \tag{2.58}$$

This implies that for almost all x,y from the same partition element in $\bar{P}$ that

$$w(\|x-y\|)\geq w(\|\psi_p^{-1}(x)-\psi_p^{-1}(y)\|). \tag{2.59}$$

We obtain for the intra-partition-element interaction

$$\begin{aligned}&\int(1-\rho_{\bar{P}}(x,y))w(\|x-y\|)d\lambda(x,y)\\ &\overset{(\star)}{>}\int(1-\rho_P(\psi_p^{-1}(x),\psi_p^{-1}(y)))\\ &\qquad\cdot w(\|\psi_p^{-1}(x)-\psi_p^{-1}(y)\|)d\lambda(x,y)\\ &=\int(1-\rho_P(x,y))w(\|x-y\|)d(\lambda\circ\psi_p^{-1})(x,y).\end{aligned} \tag{2.60}$$

The inequality in $(\star)$ stems from the fact that for some non-zero measure set of pairs (x,y) the inequality (2.57) and thus (2.59) is strict by existence of the set S, which separates two positive measure parts of p. More explicitly, we will show that removing the separating set S moves some non-null subsets of p closer together so that the resulting distance is less than 2β. Then we know that the weight function has decreased on this subset as it strictly decreases below 2β. More technically, we

can choose $\inf S \leq x < y \leq \sup S$ in $[0,1]$ such that $y - x < 2\beta$ and $\lambda(S \cap [x,y]) > 0$. Let $\varepsilon := \frac{2\beta - y + x}{2}$. Since there is some measure of p on both sides of S, we have

$$\begin{aligned} x_0 &:= \inf\{x' \leq x | \lambda((x',x) \cap p) = 0\} > 0 \\ y_0 &:= \sup\{y' \geq y | \lambda((y,y') \cap p) = 0\} < 1. \end{aligned} \tag{2.61}$$

Then $\lambda((x_0 - \varepsilon, x_0) \cap p) > 0$ and $\lambda((y_0, y_0 + \varepsilon) \cap p) > 0$ as well as

$$\lambda((x_0, y_0) \cap p) = \lambda((x,y) \cap p) < 2\beta. \tag{2.62}$$

Now, for all $x' \in (x_0 - \varepsilon, x_0)$ and $y' \in (y_0, y_0 + \varepsilon)$ we have

$$\begin{aligned} \|x' - y'\| &= \|\psi_p(x') - \psi_p(y')\| + \lambda((x',y') \cap p^c) \\ &\geq \|\psi_p(x') - \psi_p(y')\| + \lambda((x_0,y_0) \cap p^c) \\ &> \|\psi_p(x') - \psi_p(y')\| \end{aligned} \tag{2.63}$$

and

$$\begin{aligned} \|\psi_p(x') - \psi_p(y')\| &= \|x' - y'\| - \lambda((x',y') \cap p^c) \\ &\leq \|x' - y'\| - \lambda((x_0,y_0) \cap p^c) \\ &< 2\varepsilon + y_0 - x_0 - \lambda((x_0,y_0) \cap p^c) \\ &= 2\varepsilon + \lambda((x_0,y_0) \cap p) \\ &= 2\varepsilon + \lambda((x,y) \cap p) \\ &< 2\varepsilon + y - x < 2\beta, \end{aligned} \tag{2.64}$$

so that we obtain for all these x' and y' that

$$w(\|x' - y'\|) < w(\|\psi_p(x') - \psi_p(y')\|) \tag{2.65}$$

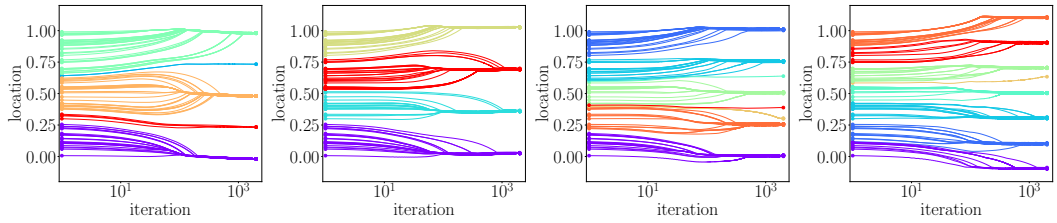
by the monotonicity assumption on w. Since ψ_p is almost a bijection and measure-preserving, we can take preimages under ψ_p in (2.65) and get $(\star)$. As mentioned above, inequality (2.60) translates into a lower Mod Shift energy for $\bar{P}$ than for P.

So in an optimal partition, no partition element can be disconnected by any set of positive measure that leaves some positive measure on either side. For open partition elements, this is the same as saying that any set of positive measure cannot disconnect it. Hence without loss of generality, we can assume that in an optimal partition, all partition elements are connected, i.e., intervals. Thus, we can identify an optimal partition with its set of cuts, the points at which two different partition elements meet and 0 and 1, up to null sets. From now on, we will omit the non-uniqueness of an optimal partition up to null sets.

Next, we show that an optimal partition must be finite. If not, the cuts have a limit point in $[0,1]$. In particular, there are two cuts c_1, c_3 of distance below β between which there is another cut c_2. However, all the points between c_1 and c_3 attract each other and are separated from all other points. Hence, removing c_2 connects points that attract each other, contradicting the optimality of the partition.

If $\beta = 0.5$, we have by w's symmetry that any bipartition yields inter-partition energy 0. In fact, for a cut at c the energy is given by

$$\int_0^c \int_c^1 w(y-x)dydx = \int_0^{\frac{c}{2}} \int_c^1 w(y+z-c) + w(y-z)dydz. \tag{2.66}$$



(a) $\beta = \frac{1}{4}$ (b) $\beta = \frac{1}{6}$ (c) $\beta = \frac{1}{8}$ (d) $\beta = \frac{1}{10}$

Figure 2.9: Mod Shift on 2048 samples of a uniform distribution with rectified linear ρ and w. Trajectories of 100 subsampled points are depicted with a logarithmic time scale. The points and trajectories are colored according to the clustering. As predicted in Prop. 2.17, Mod Shift divides the uniform line into equisized intervals and with decreasing β the number of clusters increases. Note that Mod Shift is not a hierarchical clustering algorithm in β: Borders between different clusters at some level of β do not persist for all lower values of β.

the point symmetry reads $w(y) = -w(-y+1)$ for $\beta = 0.5$, so that the terms in the inner integral cancel out

$$\int_c^1 w(y+z-c)dy = \int_z^{1+z-c} w(y)dy = \int_z^{1+z-c} -w(-y+1)dy = -\int_{c-z}^{1-z} w(y)dy$$
$$= -\int_c^1 w(y-z)dy. \tag{2.67}$$

Since w (strictly) increases pointwise with β (for distances less than 2β), we have that $E(P)$ increases strictly with β. Hence, for $\beta < 0.5$, any bipartition is strictly better than the trivial one-element partition of objective value 0. Moreover, if a bipartition is optimal, then the cut must lie at $[2\beta, 1-2\beta] \cup \{0.5\}$, as otherwise, density next to the cut in the larger segment has net attraction to the smaller segment but no attraction to its own segment, so increasing the smaller segment is advantageous.

Suppose the interval $[2\beta, 1-2\beta]$ contains points other than 0.5. Then $\beta < 0.25$. Suppose there is a cut at some $c \in [2\beta, 0.5)$. Then the segment $[c, 1]$ is longer than 2β, and by the same argument as above, we find that an additional cut in $[c, 1]$ yields a better objective value than just cutting at c. Hence, if a single cut is optimal and $\beta \neq 0.5$, then it bisects the interval in the middle.

Now, suppose we have an optimal partition with k elements. Fixing three consecutive cuts c_1, c_2, c_3, we know that the cut c_2 optimally bipartitions $[c_1, c_3]$. Again, this is just a down-scaled version of the original problem. The position of c_2 is optimal given c_1 and c_3 since all interaction across c_1 and c_3 is fixed so that we have an optimal subproblem property. As a result, we can deduce that either $c_3 - c_2 = c_2 - c_1$, i.e., c_2 is directly in the middle of c_1 and c_3, or $2\beta = c_3 - c_1$. Iterating this argument for all consecutive triples of cuts, we get that consecutive cuts have the same distance, or the distance of a cut to the one after the next is 2β. Therefore, an optimal partition can only consist of intervals of at most two different sizes, and if there are two differently sized intervals, their lengths add up to 2β.

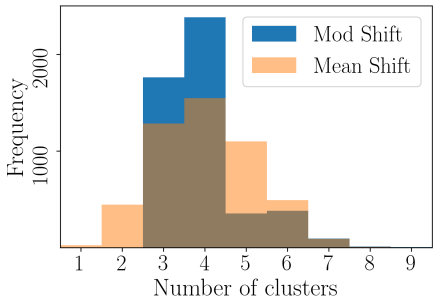


Figure 2.10: Mod Shift is more stable than Mean Shift with respect to small changes in the density. Frequencies of the number of clusters obtained by Mod Shift and Mean Shift from 5000 samples of 100 points from a uniform distribution.

Now consider some partition consisting of at least three intervals whose intervals have two different lengths, which add up to 2β. There must be three consecutive cuts c_1, c_2, c_3 such that $c_3 - c_1 = 2\beta$ and $c_3 - c_2 \neq c_2 - c_1$. As any single cut is equally good for $\beta = 0.5$, we can deduce for the down-scaled situation $[c_1, c_3]$ that we can move c_2 without changing the energy of the partition. In particular, without changing the energy, we can get to a partition with intervals of more than two distinct lengths. However, such a partition cannot be optimal. Consequently, the only optimal partition with at least three elements must consist of equisized intervals. Only for $\beta = 0.5$ any bipartition can be optimal.

2. For any $\beta \neq 0.5$, an optimal partition is of the form in 1. Hence, it is optimal to subdivide an interval of length $2/|P|$ but not one of length $1/|P|$. Therefore, we get $\beta \leq 0.5 \cdot 2/|P|$ and $\beta \geq 0.5 \cdot 1/|P|$ by the arguments in the first part of the proof.

□

We ran Mod Shift on a toy dataset consisting of points drawn from a uniform distribution, see Fig. 2.9. This experiment confirms Prop. 2.17's predictions that the uniform distribution is torn up into equisized intervals of sizes between β and 2β. We strongly believe that the optimal number of subdivisions in Prop. 2.17 monotonically decreases with β, but were unable to prove this theoretically. The experiments in Fig. 2.9 support this hypothesis. Fig. 2.9 reassures us that the convergence points of most points are either identical or separated by 2β as we would expect since ρ has zero gradient for distances above 2β but positive gradient for distances below 2β. The learning rate was chosen deliberately small to visualize the exact behavior of Mod Shift, which is why some points on borders of clusters did not converge yet. Typically, points of distance β to an end of the line have the largest initial shift. This is because they experience roughly zero net force towards the center of the line but attraction towards the points at the very ends of the line. Once they have moved away from the line's center, points closer to the center have less negative than positive interaction with points at the end of the line and hence start moving towards the end.

2.6.4 *Mod Shift and Mean Shift on noisy uniform data*

Mod Shift's focus on distances makes it more robust to fluctuations of the data distribution than the density-based Mean Shift. Such fluctuations might result

in spurious modes of a KDE, which can change fixed Mean Shift's result. We found that the number of clusters that Mod Shift produces on data sampled from a uniform distribution varies less than that produced by Mean Shift on the same data, see Fig. 2.10.

We sampled 100 points from a uniform distribution over $[0, 1]$ and ran Mod Shift with $\beta = 0.2$ as well as Mean Shift with a flat kernel of bandwidth 0.2 and counted how many clusters were found. This experiment was repeated 5000 times and the frequencies for the number of clusters are depicted in Fig. 2.10. We clearly see that Mod Shift is more stable with respect to the noisy samples and predicts 3.95 clusters on average with a variance of 0.98. While Mean Shift predicts about the same number of clusters on average (4.03), it does so with a 50% higher variance of 1.47. The variations of the sample density cause more variable results for Mean Shift than for Mod Shift.

Additionally, we observe that Mod Shift mostly finds 3 or 4 clusters. This is in accordance with Prop. 2.17, which would predict a number of clusters between $\frac{1}{2\beta} = 2.5$ and $\frac{1}{\beta} = 5$ if the data was truly uniform and not a finite sample.

In these experiments, we used the Adam optimizer for Mod Shift with a learning rate of 0.005 and learning rate decay by 0.1 after 4250 iterations. The hard clustering was obtained by Single Linkage at a threshold of 0.05.

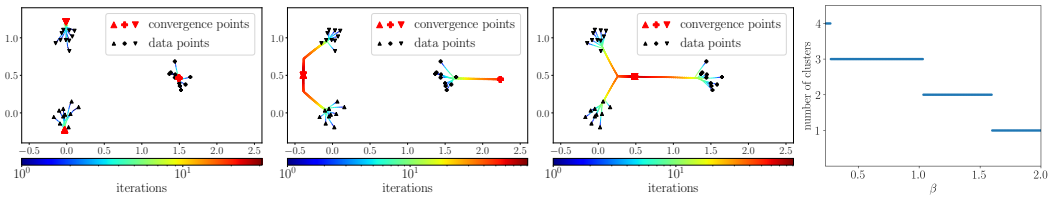
2.6.5 *Mod Shift versus Mean Shift*

In this section, we contrasted the properties of Mean Shift and Mod Shift. The most relevant difference is Mod Shift's usage of both attraction and repulsion above and below its scale parameter β. This makes β highly interpretable and allows Mod Shift to find multiple clusters in situations where Mean Shift cannot. While fixed Mean Shift can produce multiple stable clusters determined by the number of modes of the KDE, these are a joint consequence of Mean Shift's scale parameter and the data density. Therefore, Mean Shift is more density-driven, while Mod Shift is more directly distance-driven. Both inductive biases have advantages and disadvantages. While density is a very plausible cue for clusters, Mean Shift allows points to move arbitrarily far towards a region of higher density. If prior knowledge on the spatial cluster size or the range of attraction and repulsion is available, e.g., if the dataset was generated via metric learning, Mod Shift's strong distance-prior is more appropriate.

2.7 EXPERIMENTS

2.7.1 *Toy dataset*

We start by exploring Mod Shift's behavior on a simple toy dataset sampled from three normal distributions, see Fig. 2.11. The point shifting nature of Mod Shift can be seen in Figs. 2.11a–c for different values of β yielding different numbers of clusters. Note how the convergence points are always separated by at least 2β, beyond which ρ becomes constant. Fig. 2.11d summarizes how the number of resulting clusters changes with β. As β increases, more distant points cease to repel and start to attract each other. We see that if β is above the mean distance between



(a) $\beta = 0.7$ (b) $\beta = 1.3$ (c) $\beta = 2.5$ (d) Number of clusters by β

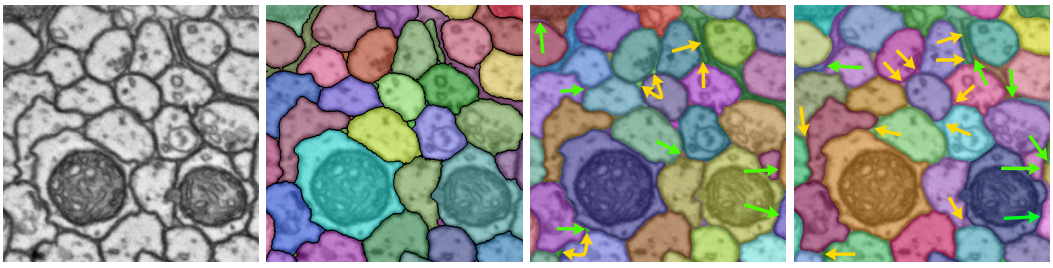
Figure 2.11: Mod Shift on a toy dataset illustrating the effect of β. **2.11a–c** show the trajectories of the points to three (two, one) cluster(s) under Mod Shift for $\beta = 0.7$ $(1.3, 2.5)$. **2.11d:** Number of clusters for $\beta \in [0.23, 2.0]$. As β increases, points that are further apart start to attract each other so that the number of convergence points and thus clusters decreases. For sufficiently small β (not depicted), all 30 points form their own cluster. Figure best viewed in color.

two point clouds (≈ 1 for the two left point clouds, ≈ 1.5 for any of them to the right one), they get merged, as the total interaction between them is attractive. For smaller β, the total interaction between the point clouds becomes repulsive, and the point clouds get separated. Confer also Appendix B.3.

2.7.2 *Mod Shift on pixel embeddings*

In this section we demonstrate that Mod Shift is competitive with Mean Shift [86] and HDBSCAN [129] in clustering pixel embeddings for instance segmentation. We use pixel embeddings generated by embedding networks from subvolumes of the CREMI A, B, and C datasets [18] as well as from the ISBI 2012 challenge dataset [3]. We let Mod Shift and Mean Shift run for 50 iterations and then obtain the hard cluster assignment by thresholded Single Linkage clustering, i.e., we transitively merge pairs of points whose distance is below some threshold. We perform a grid search to find the parameters for each clustering method that minimize the CREMI score [18] on each dataset. Further details and additional results can be found in Appendix B.4. Mod Shift achieves similar scores as Mean Shift on these datasets. HDBSCAN performs better than both on the subvolumes of CREMI B and C but is non-differentiable and has less intuitive hyperparameters. Results for HDBSCAN, fixed Mod Shift and Mean Shift with a flat kernel and with a threshold equal to the scale parameter on the data from CREMI A can be found in Tab. 2.1, which shows Mod Shift's effectiveness on real-world data. The best performing parameter setting of Mod Shift and Mean Shift yield comparable results on all four datasets, see Tabs. B.5–B.8 in the Appendix.

In Fig. 2.12, we show the respective segmentations for Mod Shift and Mean Shift for a slice of the CREMI A subvolume. In contrast to Mean Shift, Mod Shift manages to identify more of the tiny ground truth segments, which we attribute to Mod Shift's use of repulsion and an ensuing reduction in false merges compared to Mean Shift, see Lem. 2.14 and Prop. 2.15. Nevertheless, the elongated cells (violet and green in the upper part of Fig. 2.12b) are correctly identified as single segments by Mod Shift.



(a) Raw data (b) Ground truth (c) Mod Shift segmentation (d) Mean Shift segmentation

Figure 2.12: Comparison of Mod Shift and Mean Shift for instance segmentation in a crop of the CREMI A dataset [18]. **Left to right:** Raw data, ground truth, Mod Shift's, and Mean Shift's segmentations. Black lines in 2.12b indicate segment boundaries. Yellow (green) arrows in 2.12c and 2.12d highlight false merges (splits). In contrast to Mean Shift, Mod Shift manages to separate more of the small cells and does not split up the elongated cells. Figure best viewed in color.

Table 2.1: Results on data from CREMI A by metrics of [18], lower is better. Mod Shift is on par with Mean Shift. The non-differentiable HDBSCAN tends to perform better on data from subsets B and C of CREMI.

Method	CREMI score	ARAND	VOI split	VOI merge
HDBSCAN	0.0703	0.0231	0.1687	0.0459
Mean Shift	0.0629	**0.0215**	**0.1408**	0.0428
Mod Shift	**0.0628**	0.0217	0.1446	**0.0367**

We also compared Mod Shift to Mutex Watershed [169, 170] and Multicut on a downsampled version of the data. Here, Mod Shift yields better CREMI scores than both and is much faster, see Appendix B.4.10 for more details.

2.8 CONCLUSION

We have introduced a new algorithm to cluster points in Euclidean space. It has close ties to the graph-based Multicut problem. Based on a scale parameter β, it internally determines the number of clusters by using both attraction and repulsion. It shifts points during the clustering process similar to Mean Shift and is fully differentiable. We have analyzed the algorithm thoroughly and related it to existing clustering approaches, thereby exhibiting new exciting connections between graph-based and Euclidean clustering. Finally, we have illustrated Mod Shift's practical potential on toy and real datasets.

In future work, we plan to use Mod Shift with signed weights that are not inferred from the point positions but, e.g., obtained as the output of a neural network such as in [169]. In this way, Mod Shift would allow the practitioner to combine feature and relational information from different sources.

Part II

DIMENSIONALITY REDUCTION

3

ON UMAP'S TRUE LOSS FUNCTION

UMAP has supplanted t-SNE as state-of-the-art for visualizing high-dimensional datasets in many disciplines, but the reason for its success is not well understood. In this work, we investigate UMAP's sampling-based optimization scheme in detail. We derive UMAP's true loss function in closed form and find that it differs from the published one in a dataset size-dependent way. Consequently, we show that UMAP does not aim to reproduce its theoretically motivated high-dimensional UMAP similarities. Instead, it tries to reproduce similarities that only encode the k nearest neighbor graph, thereby challenging the previous understanding of UMAP's effectiveness. Alternatively, we consider the implicit balancing of attraction and repulsion due to the negative sampling to be key to UMAP's success. We corroborate our theoretical findings on toy and single-cell RNA sequencing data. This chapter is based on [39].

3.1 INTRODUCTION

Today's most prominent methods for non-parametric, non-linear dimension reduction are t-Distributed Stochastic Neighbor Embedding (t-SNE) [160, 161] and Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) [112]. The heart of UMAP is claimed to be its sophisticated method for extracting high-dimensional similarities. However, the reason for UMAP's excellent visualizations is not immediately obvious from this approach. In particular, UMAP's eponymous uniformity assumption, see Section 3.3, is arguably difficult to defend for the wide range of datasets on which UMAP performs well. Therefore, it is not well understood what aspect of UMAP is responsible for its great visualizations.

Both t-SNE and UMAP have to overcome the computational obstacle of considering the quadratic number of interactions between all pairs of points. The breakthrough for t-SNE came with a Barnes-Hut approximation [160]. Instead, UMAP employs a sampling-based approach to avoid a quadratic number of repulsive interactions. Other than [15] little attention has been paid to this sampling-based optimization scheme. In this work, we fill this gap and analyze UMAP's optimization method in detail. In particular, we derive the effective, closed form loss function, which UMAP's optimization scheme actually minimizes. While UMAP's use of negative sampling was intended to avoid quadratic complexity, we find, surprisingly, that the resulting effective loss function differs significantly from UMAP's purported loss function. The weight of the loss function's repulsive term is drastically reduced.

As a consequence, UMAP is not actually geared towards reproducing the clever high-dimensional similarities. In fact, we show that most information beyond the symmetric kNN graph connectivity is essentially ignored as UMAP approximates a binarized version of the high-dimensional similarities. These theoretical findings underpin some empirical observations in [15] and demonstrate that the gist of

UMAP is not its high-dimensional similarities. This resolves the disconnect between UMAP's uniformity assumption and its success on datasets of varying density.

From a user's perspective, it is essential to gain an intuition for deciding which visualization features can be attributed to the data and which ones are more likely artifacts of the visualization method. With our analysis, we can explain UMAP's tendency to produce crisp, over-contracted substructures, which increases with the dataset size, as a side effect of its optimization.

Without the motivation of reproducing sophisticated high-dimensional similarities in embedding space, it seems unclear why UMAP performs well. We propose an alternative explanation for UMAP's success: The sampling-based optimization scheme balances the attractive and repulsive loss terms despite the sparse high-dimensional attraction. Consequently, UMAP can effectively leverage the connectivity information of the symmetric kNN graph via gradient descent.

3.2 RELATED WORK

For most of the past decade t-SNE [160, 161] was considered state-of-the-art for nonlinear dimension reduction. In the last years, UMAP [112] at least ties with t-SNE. In both methods, points are embedded to reproduce high-dimensional similarities. Unlike the original t-SNE, the high-dimensional similarities are sparse for UMAP and do not need to be normalized over the entire dataset. Additionally, t-SNE adapts the local scale of high-dimensional similarities by achieving a predefined perplexity, while UMAP uses its uniformity assumption. The low-dimensional similarity functions also differ. Recently, Böhm *et al.* [15] placed both UMAP and t-SNE on a spectrum of dimension reduction methods that mainly differ in the amount of repulsion employed. They argue that UMAP uses less repulsion than t-SNE. A parametric version of UMAP was proposed in [141].

UMAP's success, particularly in the biological community [9, 126], sparked interest in understanding UMAP more deeply. The original paper [112] motivates the choice of the high-dimensional similarities using concepts from algebraic topology and thus justifies UMAP's transition from local similarities $\mu_{i\to j}$ to global similarities μ_{ij}. The authors find that while the algorithm focuses on reproducing the local similarity pattern similar to t-SNE, it achieves better global results. In contrast, Kobak and Linderman [84] attribute the better global properties of UMAP visualizations to the more informative initialization and show that t-SNE manages to capture more global structure if initialized similarly. Wang *et al.* [165] also analyze the local and global properties of t-SNE, UMAP, and TriMAP [1].

Narayan *et al.* [116] observe that UMAP's uniformity assumption leads to visualizations in which denser regions are more spread out while sparser regions get overly contracted. They propose an additional loss term that aims to reproduce the local density around each point and thus spreads out sparser regions. We provide an additional explanation for overly contracted visualizations: UMAP does not reproduce the high-dimensional similarities but exaggerates the attractive forces over the repulsive ones, which can result in overly crisp visualizations, see Figs. 3.2b, 3.2c and 3.3a.

Our work aligns with Böhm *et al.* [15]. The authors conjecture that the sampling-based optimization procedure of UMAP prevents the minimization of the supposed

loss function, thus not reproducing the high-dimensional similarities in embedding space. They substantiate this hypothesis by qualitatively estimating the relative size of attractive and repulsive forces. In addition, they implement a Barnes-Hut approximation to the loss function (3.6) and find that it yields a diverged embedding. We analyze UMAP's sampling procedure in depth, compute UMAP's true loss function in closed form and contrast it against the supposed loss in Sec. 3.6. Based on this analytic effective loss function, we can further explain Böhm *et al.* [15]'s empirical finding that the specific high-dimensional similarities provide little gain over the binary weights of a symmetric kNN (skNN) graph,[1] see Sec. 3.8. Finally, our theoretical framework leads us to a new tentative explanation for UMAP's success in Sec. 3.11.

3.3 BACKGROUND ON UMAP

UMAP assumes that the data lies on a low-dimensional Riemannian manifold (R, g) in ambient high-dimensional space. Moreover, the data is assumed to be uniformly distributed with respect to the metric tensor g, or put simply, with respect to the distance along the manifold. In turn, the metric tensor g is assumed to be locally constant. The key idea of UMAP [112] is to compute pairwise similarities in high-dimensional space, which inform the optimization of the low-dimensional embedding. Let $x_1, ..., x_n \in \mathbb{R}^D$ be high-dimensional, mutually distinct data points for which low-dimensional embeddings $e_1, ..., e_n \in \mathbb{R}^d$ shall be found, where $d \ll D$, often $d = 2$ or 3.

First, UMAP extracts high-dimensional similarities between the data points. For this, it computes the k nearest neighbor graph. Let $i_1, ..., i_k$ denote the indices of x_i's k nearest neighbors in increasing order of distance to x_i. Then, using its uniformity assumption, UMAP fits a local notion of similarity for each data point i by selecting a scale σ_i such that the total similarity of each point to its k nearest neighbors is normalized, i.e., find σ_i such that

$$\sum_{\kappa=1}^{k} \exp\Big(- \big(d(x_i, x_{i_\kappa}) - d(x_i, x_{i_1})\big) / \sigma_i \Big) = \log_2(k). \tag{3.1}$$

This defines the directed high-dimensional similarities

$$\mu_{i\to j} = \begin{cases} \exp\Big(- \big(d(x_i, x_j) - d(x_i, x_{i_1})\big) / \sigma_i \Big) \text{ for } j \in \{i_1, \dots, i_k\} \\ 0 \text{ else.} \end{cases} \tag{3.2}$$

Finally, these are symmetrized to obtain undirected high-dimensional similarities or input similarities between items i and j

$$\mu_{ij} = \mu_{i\to j} + \mu_{j\to i} - \mu_{i\to j}\mu_{j\to i} \in [0, 1]. \tag{3.3}$$

While each node has exactly k non-zero directed similarities $\mu_{i\to j}$ to other nodes, which sum to $\log_2(k)$, this does not hold exactly after symmetrization. Nevertheless, typically the μ_{ij} are highly sparse, each node has positive similarity to about k

1 The symmetric k nearest neighbor graph contains an edge ij if i is among j's k nearest neighbors or vice versa.

other nodes and the degree of each node $d_i = \sum_{j=1}^{n} \mu_{ij}$ is approximately constant and close to $\log_2(k)$, see Fig. 3.1a and 3.1b in the next Sec. 3.4. For convenience of notation, we set $\mu_{ii} = 0$ and define the total similarity as $\mu_{\text{tot}} = \frac{1}{2}\sum_{i=1}^{n} d_i$.

Distance in embedding space is transformed to low-dimensional similarity by a smooth approximation to the high-dimensional similarity function, $\phi(d; a, b) = (1 + ad^{2b})^{-1}$, for all pairs of points. The shape parameters a, b are essentially hyperparameters of UMAP. We will overload notation and write

$$v_{ij} = \phi(||e_i - e_j||) = \phi(e_i, e_j) \tag{3.4}$$

for the low-dimensional or embedding similarities and usually suppress their dependence on a and b.

Supposedly, UMAP approximately optimizes the following objective function with respect to the embeddings $e_1, \ldots, e_n$:

$$\mathcal{L}(\{e_i\}|\{\mu_{ij}\}) = -2 \sum_{1 \le i < j \le n} \big(\mu_{ij} \log(v_{ij}) \quad + (1 - \mu_{ij}) \log(1 - v_{ij})\big) \tag{3.5}$$

$$= -2 \sum_{1 \le i < j \le n} \big(\mu_{ij} \underbrace{\log(\phi(e_i, e_j))}_{-\mathcal{L}^a_{ij}} + (1 - \mu_{ij}) \underbrace{\log(1 - \phi(e_i, e_j))}_{-\mathcal{L}^r_{ij}}\big). \tag{3.6}$$

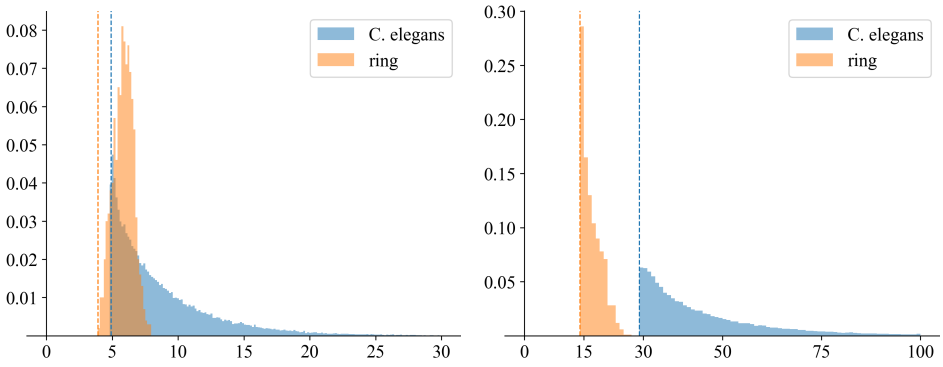
While the high-dimensional similarities μ_{ij} are symmetric, UMAP's implementation does consider their direction during optimization. For this reason, our losses in equations (3.5) and (3.6) differ by a factor of 2 from the one given in [112]. Viewed through the lens of a force-directed model, the derivative of the first term in each summand of the loss function, $-\partial\mathcal{L}^a_{ij}/\partial e_i$, captures the attraction of e_i to e_j due to the high-dimensional similarity μ_{ij} and the derivative of the second term, $-\partial\mathcal{L}^r_{ij}/\partial e_i$, represents the repulsion that e_j exerts on e_i due to a lack of similarity in high dimension, $1 - \mu_{ij}$. Alternatively, the loss can be seen as the sum of binary cross entropy losses for each pairwise similarity. Thus, it is minimal if the low-dimensional similarities v_{ij} exactly match their high-dimensional counterparts μ_{ij}, i.e., if UMAP manages to reproduce the input similarities in embedding space.

UMAP uses a sampling-based stochastic gradient descent to optimize the low-dimensional embedding, typically starting from a Laplacian Eigenmap layout [11, 84]. Our main contribution is to show that the sampling-based optimization leads to a different objective function so that UMAP does not reproduce the high-dimensional similarities in low-dimensional space, see Secs. 3.5 to 3.8.

3.4 UMAP'S DEGREE DISTRIBUTION

Before diving deeper into the analysis of UMAP, we gain some insights into the statistics of UMAP's weighted skNN graph, both theoretically and empirically. In this chapter, we will mainly use two datasets: a toy dataset consisting of 1000 points sampled uniformly from a ring and a biological dataset[2] which contains gene expression data of 86 024 cells of C. elegans [116, 126]. For the C. elegans dataset, we start with a 100 dimensional PCA of the data, use the cosine metric in high-dimensional space and consider $k = 30$ neighbors. For more details confer Appendices C.3.1 and C.3.2.

2 obtained from http://cb.csail.mit.edu/cb/densvis/datasets/. We informed the authors of our use of the dataset, which they license under CC BY-NC 2.0.



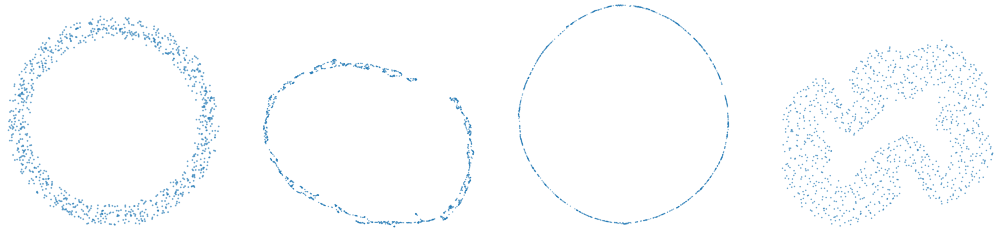
(a) UMAP degrees (b) skNN degrees

Figure 3.1: **3.1a:** Histogram over the UMAP degree distributions for the toy ring and the C. elegans datasets. Both distributions are fairly peaked close to their lower bounds $\log_2(k)$, highlighted as dashed lines. **3.1b:** Histogram over the degree distribution in the skNN graph for the toy ring and the C. elegans datasets. Both distributions are fairly peaked close to their lower bounds $k-1$, highlighted as dashed lines. Since UMAP's implementation considers a point its first nearest neighbor, but the μ_{ii} are set to zero, the degree is one lower than the intended number of nearest neighbors k.

Prior to symmetrization, the degree of each node $\vec{d}_i = \sum_{j=1}^{n} \mu_{i\to j}$ equals $\log_2(k)$ due to UMAP's uniformity assumption. For UMAP's default value of $k = 15$ this is ≈ 3.9. For the C. elegans dataset we used $k = 30$ in which case $\log_2(30) \approx 4.9$. Symmetrizing changes the degree in a dataset-dependent way. For $a, b \in [0, 1]$, we have $\max(a, b) \leq a + b - ab$. So, the symmetric degrees $d_i = \sum_{j=1}^{n} \mu_{ij}$ are lower bounded by $\log_2(k)$. Empirically, we find that the degree distribution is fairly peaked close to this lower bound, see Fig. 3.1a. In the skNN graph, each node has a degree of at least k. Empirically, the degree distribution is fairly peaked at this lower bound, see Fig. 3.1b.

3.5 UMAP DOES NOT REPRODUCE HIGH-DIMENSIONAL SIMILARITIES

UMAP produces scientifically useful visualizations in many domains and is fairly robust to its hyperparameters. Since any visualization of intrinsically high-dimensional data must be somehow unfaithful, it is not straightforward to check the visualization quality other than by its downstream use. Some quantitative metrics such as the correlation between high- and low-dimensional distances [9, 84] exist. We follow a different route to show some unexpected properties of UMAP. Consider the toy example of applying UMAP to data that is already low-dimensional so that no reduction in dimension is required: $D = d = 2$. Ideally, the data would be preserved in this situation. One might also expect that UMAP achieves its aim of reproducing the input similarities perfectly. Surprisingly, UMAP meets neither of these expectations. In Fig. 3.2, we depict 2D UMAP visualizations of a 2D uniform ring dataset. To ease UMAP's task, we initialized the embedding with the original data, hoping that UMAP's optimization would just deem this layout to be optimal. We also used a longer run time to ensure convergence of UMAP's optimization procedure. Otherwise, we used the default hyperparameters. The results with default initialization and run-time are qualitatively similar and shown in Fig. C.7. UMAP



(a) Original data (b) UMAP (c) UMAP on larger dataset (d) UMAP from dense similarities

Figure 3.2: UMAP does not preserve the data even when no dimension reduction is required. **3.2a:** Original data consisting of 1000 points sampled uniformly from a ring in 2D. All optimizations ran for 10000 epochs and were initialized at the input data. **3.2b:** Result of UMAP. The circular shape is visible, but the ring width is nearly completely contracted. **3.2c:** Result of UMAP on a dataset consisting of twenty rings as in 3.2a spaced so they do not interact in the input similarity computation. Only the embedding of one ring is shown. There should not be a significant difference from Fig. 3.2b. However, the ring width is completely contracted. **3.2d:** Result of UMAP for dense input similarities computed from the original data in Fig. 3.2a with the similarity function ϕ from Sec. 3.3. No change from the initialization would be optimal in this setting. Instead, the output has spurious curves and larger width. More figures with the default number of epochs and initialization are depicted in Fig. C.7.

manages to capture the ring shape of the data but changes its appearance significantly. It contracts the width of the ring nearly to a line, see Fig. 3.2b, a phenomenon also observed on real-world datasets, see Fig. 3.3a. Whether this exaggeration of the ring shape is beneficial depends on the use case. This finding goes beyond Narayan *et al.* [116]'s observation that over-contraction happens in regions of low density since our toy dataset is sampled uniformly from a circular ring.

Moreover, we embedded the same ring with UMAP as part of a larger dataset of twenty such rings, spaced so that the other rings do not influence the input similarities for each ring. According to Section 3.3, we expect an embedding qualitatively similar to that of the individual ring. However, we see in Fig. 3.2c that UMAP produces an even crisper embedding. In another experiment, we varied the number of samples from a single ring and observed again that the tendency for over-contraction increases with the dataset size. For more information on UMAP's dependence on the dataset size, confer Sec. 3.9. The dataset size dependence is particularly noteworthy, as computing UMAP embeddings of subsets or subsamples of a dataset is common practice.

As described in Sec. 3.3, UMAP employs different methods in input and embedding space to transform distances into similarities. In particular, in input space, similarities are zero for all but the closest neighbors, while in embedding space, they are computed with the heavy-tailed function ϕ. To test whether this prevents the reproduction of the input data, we also computed the dense similarities on the original data with ϕ. We used these as input similarities for the embedding in Fig. 3.2d. Since we also initialize with the original dataset, a global optimum of the objective function (3.6) for this choice of input similarity, one would expect no change by UMAP's optimization scheme. However, in this setting, UMAP produces spurious curves and increases the width of the ring.

Table 3.1: UMAP loss value for various combinations of input and embedding similarities, μ_{ij}, ν_{ij}, of the toy example in Fig. 3.2. The loss for the UMAP embedding in Fig. 3.2b (middle column) is always higher than for another two-dimensional layout (**bold**). Hence, UMAP does not minimize its purported loss. Results are averaged over seven runs, and one standard deviation is given, see also Appendix C.1.1.

	Embedding similarities ν_{ij}		
Input similarities μ_{ij}	$\mathbb{1}(i == j)$ (diverged layout)	$\phi(\{e_1, \dots, e_n\})$ (UMAP result)	$\phi(\{x_1, \dots, x_n\})$ (input layout)
$\mu(\{x_1, \dots, x_n\})$	$\mathbf{62959 \pm 82}$	70235 ± 1301	136329 ± 721
$\phi(\{x_1, \dots, x_n\})$	902757 ± 2788	331666 ± 1308	$\mathbf{224584 \pm 8104}$

Böhm *et al.* [15] implemented a Barnes-Hut approximation of UMAP's objective function (3.6), which produced a diverged embedding. Inspired by this finding, we compute the loss values according to equation (3.5) for various input and embedding similarities μ_{ij} and ν_{ij} in our toy example, see Tab. 3.1.

Consider the row with the usual input similarities ($\mu_{ij} = \mu(\{x_1, \dots, x_n\})$). In a completely diverged embedding, all self similarities are one and all others zero ($\nu_{ij} = \mathbb{1}(i == j)$). We find that the loss for such an embedding is lower than for the optimized UMAP embedding. This aligns with Böhm *et al.* [15]'s Barnes-Hut experiment and shows that UMAP does not optimize its supposed objective function (3.6) as a diverged embedding is approximately feasible in two dimensions. This discrepancy is not just due to input and embedding similarities being computed differently: The second row of Tab. 3.1 contains loss values for the setting in which we use the dense similarities as input similarities as in Fig. 3.2d. We initialize the embedding at the optimal loss value ($\nu_{ij} = \phi(\{x_1, \dots, x_n\}) = \mu_{ij}$), but UMAP's optimization moves away from this layout and towards an embedding with higher loss ($\nu_{ij} = \phi(\{e_1, \dots, e_n\})$) although we always compute similarity in the same manner. Clearly, UMAP's optimization yields unexpected results.

3.6 UMAP'S SAMPLING AND EFFECTIVE LOSS FUNCTION

UMAP uses a sampling-based approach to optimize its loss function, to reduce complexity. A simplified version of the sampling procedure can be found in Alg. 2. The main contribution of this chapter is the derivation of the loss function that UMAP's implementation actually optimizes.

Proposition 3.1. *UMAP's implemented optimization procedure updates an embedding e_i in expectation by*

$$2 \sum_{j=1}^{n} \left(\mu_{ij} \cdot \frac{\partial \mathcal{L}_{ij}^{a}}{\partial e_i} + \frac{d_i m}{2n} \cdot \frac{\partial \mathcal{L}_{ij}^{r}}{\partial e_i} \right). \tag{3.7}$$

Proof. We start by collecting all updates that Alg. 2 executes. An edge ij is sampled according to its high-dimensional similarity, and the embeddings e_i and e_j of the incident nodes are pulled towards each other. For each such sampled edge ij, the algorithm next samples m negative samples s uniformly from all nodes, and

Algorithm 2: UMAP's optimization

input : input similarities μ_{ij},
initial embeddings $e_1, \ldots, e_n$,
number of epochs T,
learning rate α,
number of negative samples m

output: final embeddings $e_1, \ldots, e_n$

1 **for** $t = 0$ **to** T **do**
2 **for** $ij \in \{1, \ldots, n\}^2$ **do**
3 $r \sim \text{Uniform}(0, 1)$
4 **if** $r < \mu_{ij}$ **then**
5 $e_i = e_i - \alpha \cdot \frac{\partial \mathcal{L}^a_{ij}}{\partial e_i}$
6 $e_j = e_j - \alpha \cdot \frac{\partial \mathcal{L}^a_{ij}}{\partial e_j}$
7 **for** $l = 1$ **to** m **do**
8 $s \sim \text{Uniform}(\{1, \ldots, n\})$
9 $e_i = e_i - \alpha \cdot \frac{\partial \mathcal{L}^r_{is}}{\partial e_i}$
 `// Next line is omitted in UMAP's implementation, but included for our analysis`
10 /* $e_s = e_s - \alpha \cdot \frac{\partial \mathcal{L}^r_{is}}{\partial e_s}$ */

the embedding of i is repelled from that of each negative sample. Note that the embedding of i does not repel those of the negative samples, see Alg. 2 line 10. So there are three types of gradient applied to an embedding e_i during an epoch:

1. e_i is pulled towards e_j when edge ij is sampled (Alg.2 line 5)
2. e_i is pulled towards e_j when edge ji is sampled (Alg.2 line 6)
3. e_i is pushed away from negative sample embeddings when some edge ij is sampled (Alg.2 line 9).

The full update of embedding e_i during epoch t is, according to UMAP's implementation, given by

$$g_i^t = \sum_{j=1}^{n} \left(X_{ij}^t \cdot \frac{\partial \mathcal{L}^a_{ij}}{\partial e_i} + X_{ji}^t \cdot \frac{\partial \mathcal{L}^a_{ji}}{\partial e_i} + X_{ij}^t \cdot \sum_{s=1}^{n} Y_{ij,s}^t \cdot \frac{\partial \mathcal{L}^r_{is}}{\partial e_i} \right), \tag{3.8}$$

where X_{ab}^t is the binary random variable indicating whether edge ab was sampled in epoch t and $Y_{ab,s}^t$ is the random variable for the number of times s was sampled as a negative sample for edge ab in epoch t if ab was sampled in epoch t and zero otherwise. By construction, $\mathbb{E}(X_{ab}^t) = \mu_{ab}$ and $\mathbb{E}(Y_{ab,s}^t | X_{ab}^t = 1) = m/n$. Taking the expectation over the random events in an epoch, we obtain the expected update of UMAP's optimization procedure

$$\begin{aligned}
\mathbb{E}\left(g_i^t\right) &= \mathbb{E}\left(\sum_{j=1}^{n}\left(X_{ij}^t \cdot \frac{\partial \mathcal{L}_{ij}^a}{\partial e_i} + X_{ji}^t \cdot \frac{\partial \mathcal{L}_{ji}^a}{\partial e_i} + X_{ij}^t \cdot \sum_{s=1}^{n} Y_{ij,s}^t \cdot \frac{\partial \mathcal{L}_{is}^r}{\partial e_i}\right)\right) \\
&= \sum_{j=1}^{n}\left(\mathbb{E}(X_{ij}^t) \cdot \frac{\partial \mathcal{L}_{ij}^a}{\partial e_i} + \mathbb{E}(X_{ji}^t) \cdot \frac{\partial \mathcal{L}_{ji}^a}{\partial e_i} + \sum_{s=1}^{n} \mathbb{E}(X_{ij}^t Y_{ij,s}^t) \cdot \frac{\partial \mathcal{L}_{is}^r}{\partial e_i}\right) \\
&= \sum_{j=1}^{n}\left(\mu_{ij} \cdot \frac{\partial \mathcal{L}_{ij}^a}{\partial e_i} + \mu_{ji} \cdot \frac{\partial \mathcal{L}_{ji}^a}{\partial e_i}\right) + \sum_{s=1}^{n}\sum_{j=1}^{n} \frac{\mu_{ij} m}{n} \cdot \frac{\partial \mathcal{L}_{is}^r}{\partial e_i} \\
&= 2\sum_{j=1}^{n}\left(\mu_{ij} \cdot \frac{\partial \mathcal{L}_{ij}^a}{\partial e_i} + \frac{d_i m}{2n} \cdot \frac{\partial \mathcal{L}_{ij}^r}{\partial e_i}\right). \qquad (3.9)
\end{aligned}$$

From line 2 to 3, we computed $\mathbb{E}(X_{ij}^t Y_{ij,s}^t) = \mathbb{E}_{X_{ij}^t}\left(X_{ij}^t \cdot \mathbb{E}(Y_{ij,s}^t | X_{ij}^t)\right) = \frac{\mu_{ij} m}{n}$ and from line 3 to 4 we used the symmetry of μ_{ij} and $\mathcal{L}_{ij}^a$ and collected the high-dimensional similarities $\sum_j \mu_{ij}$ into the degree d_i. □

Comparing the above closed formula for the expectation of the UMAP updates of the low-dimensional embeddings to the gradient of UMAP's loss function (3.6)

$$\frac{\partial \mathcal{L}}{\partial e_i} = 2\sum_{j=1}^{n}\left(\mu_{ij} \cdot \frac{\partial \mathcal{L}_{ij}^a}{\partial e_i} + (1-\mu_{ij}) \cdot \frac{\partial \mathcal{L}_{ij}^r}{\partial e_i}\right), \qquad (3.10)$$

we find that the sampling procedure yields the correct weight for the attractive term in expectation, as designed. But, as noticed by Böhm *et al.* [15], the negative sampling changes the weight for the repulsive term significantly. Our closed formula helps to make their qualitative arguments precise: Instead of $1-\mu_{ij}$, we have a term $\frac{d_i m}{2n}$, which depends on the number of negative samples m per sampled edge. Contrary to the intention of McInnes *et al.* [112], the repulsive weights are not uniform but vary with the degree of each point d_i, which, however, is typically close to $\log_2(k)$, see Sec. 3.4.

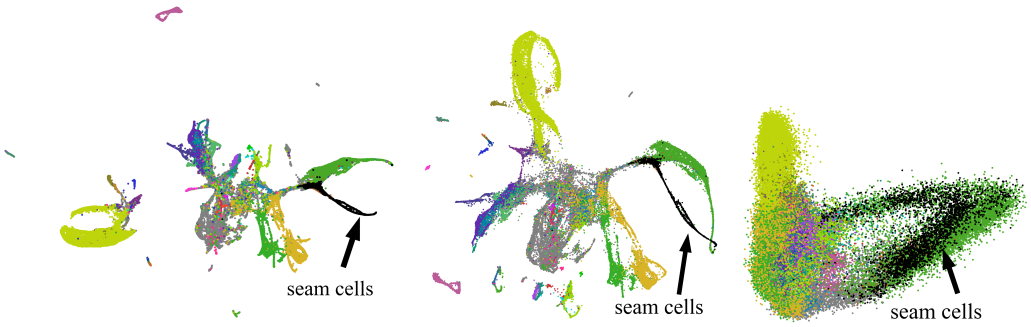
More practically, since the non-zero high-dimensional similarities are sparse, $1-\mu_{ij}$ is equal to 1 for most ij. In contrast, the expected repulsive weight is small for large datasets as the numerator $d_i m$ is of the order of $\log_2(k)m$ independent of the dataset size n but the denominator scales with n.

Another effect of the negative sampling is that, in general, the expected update (3.9) does not correspond to any loss function.

Lemma 3.2. *If the degrees d_i of UMAP's similarity graph are non-constant, there is no loss function whose gradients equal the update steps of UMAP in Prop. 3.1.*

Proof. The update in Eq. (3.9) is continuously differentiable unless two embedding points coincide. Therefore, if it had an antiderivative, that would be twice continuously differentiable at configurations where all embeddings are pairwise distinct and thus needs to have a symmetric Hessian at these points. However, we have

$$\begin{aligned}
\frac{\partial \mathbb{E}\left(g_i^t\right)}{\partial e_j} &= 2\mu_{ij} \cdot \frac{\partial^2 \mathcal{L}_{ij}^a}{\partial e_j \partial e_i} + \frac{d_i m}{2n} \cdot \frac{\partial \mathcal{L}_{ij}^r}{\partial e_j \partial e_i} \\
\frac{\partial \mathbb{E}\left(g_j^t\right)}{\partial e_i} &= 2\mu_{ij} \cdot \frac{\partial^2 \mathcal{L}_{ij}^a}{\partial e_i \partial e_j} + \frac{d_j m}{2n} \cdot \frac{\partial \mathcal{L}_{ij}^r}{\partial e_i \partial e_j}. \qquad (3.11)
\end{aligned}$$



(a) UMAP (b) Inverted weights UMAP (c) PCA

Figure 3.3: UMAP on C. elegans data from [116, 126]. **3.3a:** UMAP visualization. Several parts of the embedding appear locally one-dimensional, for instance, the seam cells.[3] **3.3b:** Same as 3.3a but with inverted positive high-dimensional similarities. The result is qualitatively similar, if not better. **3.3c:** Two dimensional PCA of the dataset. Highlighted seam cells clearly have two-dimensional variance in the PCA plot but are overly contracted to nearly a line in the UMAP plots 3.3a and 3.3b. The full legend with all cell types and further information can be found in Fig. C.10. We report quantitative metrics in Appendix C.2. Figure best viewed in color.

Since $\mathcal{L}^a_{ij}$ and $\mathcal{L}^r_{ij}$ are themselves twice continuously differentiable, their second-order partial derivatives are symmetric. But this makes the two expressions in equation (3.11) unequal unless d_i equals d_j as neither $\mathcal{L}^a_{ij}$ nor $\mathcal{L}^r_{ij}$ have vanishing second order partial derivatives anywhere. □

The problem is that the negative samples are not updated themselves, see commented line 10 of Alg. 2. We remedy this by additionally pushing the embedding of a negative sample i away from the embedding node e_j, whenever i was sampled as a negative sample to some edge jk, see Alg.2 line 10, and obtain a closed form loss function for UMAP's implementation.

Theorem 3.3. *If negative samples are pushed away from positive samples, as in Alg. 2, line 10, UMAP's optimization procedure optimizes the expected loss function*

$$\tilde{\mathcal{L}} = \mathbb{E}(\tilde{\mathcal{L}}^t) = 2 \sum_{1 \le i < j \le n} \left(\mu_{ij} \cdot \mathcal{L}^a_{ij} + \frac{(d_i + d_j)m}{2n} \cdot \mathcal{L}^r_{ij} \right). \tag{3.12}$$

Proof. Together with Alg. 2, line 10 the update of embedding e_i in epoch t becomes

$$\tilde{g}^t_i = \sum_{j=1}^{n} \left(X^t_{ij} \cdot \frac{\partial \mathcal{L}^a_{ij}}{\partial e_i} + X^t_{ji} \cdot \frac{\partial \mathcal{L}^a_{ji}}{\partial e_i} + X^t_{ij} \cdot \sum_{s=1}^{n} Y^t_{ij,s} \cdot \frac{\partial \mathcal{L}^r_{is}}{\partial e_i} + \sum_{k=1}^{n} X^t_{jk} Y^t_{jk,i} \cdot \frac{\partial \mathcal{L}^r_{ji}}{\partial e_i} \right), \tag{3.13}$$

corresponding to a loss in epoch t of

$$\tilde{\mathcal{L}}^t = \sum_{1 \le i,j \le n} \left(X^t_{ij} \cdot \mathcal{L}^a_{ij} + \sum_{s=1}^{n} X^t_{ij} Y^t_{ij,s} \cdot \mathcal{L}^r_{is} \right). \tag{3.14}$$

3 Near the tip of the seam cells, the points seem to lie on a one-dimensional manifold. Closer to the middle of the seam cells, the spread appears wider. However, there are actually two dense lines of points.

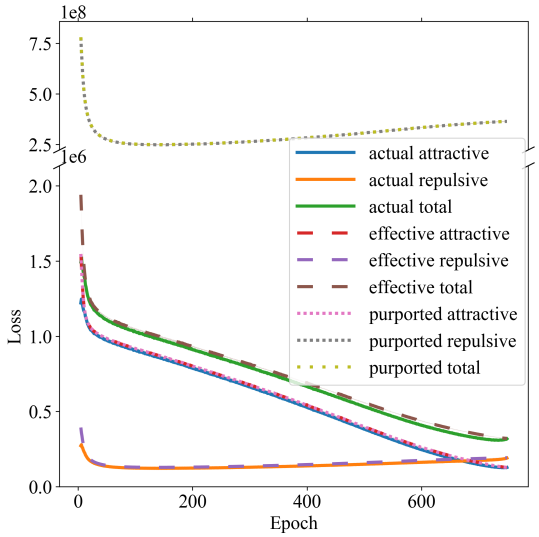


Figure 3.4: Loss curves for the optimization leading to Fig. 3.3a. Our effective loss (3.12) closely matches the actual loss (3.14), while the purported UMAP loss (3.6) is two orders of magnitude higher. The total overlays the repulsive purported loss. An average over seven runs is plotted with shaded areas of one standard deviation, see also Appendix C.1.1. Figure best viewed in color.

Similar to the proof of Prop. 3.1, we use the symmetry of μ_{ij}, $\mathcal{L}^a_{ij}$ and $\mathcal{L}^r_{ij}$ in i and j to compute the effective loss

$$\tilde{\mathcal{L}} = \mathbb{E}(\tilde{\mathcal{L}}^t) = 2 \sum_{1 \le i < j \le n} \left(\mu_{ij} \cdot \mathcal{L}^a_{ij} + \frac{(d_i + d_j)m}{2n} \cdot \mathcal{L}^r_{ij} \right). \tag{3.15}$$

□

In fact, the above remedy of also pushing the negative samples does not affect the behavior of UMAP qualitatively, see for instance Figs. C.8 and C.11.[4] In this light, we can treat $\tilde{\mathcal{L}}$ as the effective objective function that is optimized via SGD by UMAP's optimization procedure. It differs from UMAP's loss function (3.6) by having a drastically reduced repulsive weight of $\frac{(d_i+d_j)m}{2n}$ instead of $1 - \mu_{ij}$.

We illustrate our analysis on gene expression measurements of 86 024 cells of C. elegans [116, 126]. We start out with a 100 dimensional PCA of the data and use the cosine metric in high-dimensional space, consider $k = 30$ neighbors and optimize for 750 epochs, similar to [116]. The resulting visualization is depicted in Fig. 3.3a. On this dataset the average value of $1 - \mu_{ij}$ is 0.9999 but the maximal effective repulsive weight $\max_{ij} \frac{(d_i+d_j)m}{2n}$ is 0.0043, showing the dramatic reduction of repulsion due to negative sampling. After each optimization epoch, we log our effective loss $\tilde{\mathcal{L}}$ (3.12), the actual loss $\tilde{\mathcal{L}}^t$ (3.14) of each epoch computed based on the sampled (negative) pairs as well the purported UMAP loss $\mathcal{L}$ (3.6).[5] We always consider the embeddings at the end of each epoch. Note that UMAP's implementation updates each embedding e_i not just once at the end of the epoch but as soon as i is incident to a sampled edge or sampled as a negative sample. This difference does not change the actual loss much, see Appendix C.1. The loss

4 In fact, the parametric version of UMAP [141] does include the update of negative samples.
5 Our code is publicly available at https://github.com/hci-unihd/UMAPs-true-loss.

curves are plotted in Fig. 3.4. We can see that our predicted loss matches its actual counterpart nearly perfectly. While both, $\tilde{\mathcal{L}}$ and $\tilde{\mathcal{L}}^t$, agree with the attractive part of the supposed UMAP loss, its repulsive part and thus the total loss are two orders of magnitude larger. Furthermore, driven by the repulsive part, the total intended UMAP loss increases during much of the optimization process, while the actual and effective losses decrease, exemplifying that UMAP really optimizes our effective loss $\tilde{\mathcal{L}}$ (3.12) instead of its purported loss $\mathcal{L}$ (3.6). Additional loss curves for UMAP on other scRNA-seq datasets and CIFAR-10 [87] can be found in Appendix C.3.

3.7 PARAMETRIC UMAP'S SAMPLING AND EFFECTIVE LOSS FUNCTION

Recently, a parametric version of UMAP was proposed [141]. Instead of directly optimizing the embeddings, a parametric function, a neural network, is trained to map the input points to the embedding space. As usual, a mini-batch of data points is fed through the neural network at each training iteration. The loss is computed for this mini-batch, and then the neural network parameters are updated via stochastic gradient descent. To avoid the quadratic complexity of the repulsive term, a sampling strategy is employed, sketched in Alg. 3. There are three differences to the optimization scheme of Non-Parametric UMAP: First, since automatic differentiation is used, not only the head of a negative sample edge is repelled from the tail, but both repel each other. Second, the same number of edges are sampled in each epoch. Third, since only the embeddings of the current mini-batch are available, negative samples are produced not from the full dataset but only from within the non-uniformly assembled batch. This leads to a different repulsive weight for Parametric UMAP as described in

Theorem 3.4. *The effective loss function of Parametric UMAP with neural network f_θ, batch size b, and total similarity $\mu_{\text{tot}} = \frac{1}{2}\sum_{i=1}^{n} d_i$ is*

$$-\frac{1}{2(m+1)\mu_{tot}} \sum_{i,j=1}^{n} \Bigg[\mu_{ij} \cdot \log\Big(\phi\big(f_\theta(x_i), f_\theta(x_j)\big)\Big) + m\left(\frac{(b-1)}{b}\frac{d_i d_j}{2\mu_{tot}} + \frac{\mu_{ij}}{b}\right) \cdot \log\Big(1 - \phi\big(f_\theta(x_i), f_\theta(x_j)\big)\Big)\Bigg].$$

Proof. Let P_{ij} be the random variable for the number of times that edge ij is sampled into the batch B of some iteration t. Let further N_{ij} be the random variable holding the number of negative sample pairs ij in that iteration. Then the loss at iteration t is given by

$$\mathcal{L}^t = -\frac{1}{(m+1)b} \sum_{i,j=1}^{n} \Bigg[P_{ij} \cdot \log\Big(\phi\big(f_\theta(x_i), f_\theta(x_j)\big)\Big) \tag{3.16}$$

$$+ N_{ij} \cdot \log\Big(1 - \phi\big(f_\theta(x_i), f_\theta(x_j)\big)\Big)\Bigg] \tag{3.17}$$

To compute the expectation of this loss, we need to find the expectations of the P_{ij}'s and N_{ij}'s. The edges in batch B are sampled independently with replacement from the categorical distribution over all pairs rs with probability proportional to

Algorithm 3: Parametric UMAP's sampling-based optimization

input : high-dimensional similarities μ_{ij},
negative sample rate m,
number of epochs T,
learning rate α,
embedding network f_θ,
batch size b

output: final embeddings e_i

for $\tau = 0$ **to** T **do**
  Assemble batch
  $B_h, B_t = [\,], [\,]$ `// Initialize mini-batches for heads and tails`
  for $\beta = 1$ **to** b **do** `// Sample edge by input similarity into batch`
    $ij \sim \text{Categorical}(\{1,\dots,n\}^2, \{\frac{\mu_{rs}}{2\mu_{\text{tot}}}\}_{r,s=1,\dots,n})$
    B_h.append$(f_\theta(x_i))$
    B_t.append$(f_\theta(x_j))$
  Compute loss
  $l = 0$
  for $\beta = 1$ **to** b **do** `// Add attractive loss for sampled edges`
    $l = l + \mathcal{L}^a(B_h[\beta], B_t[\beta])$
  $\pi \sim \text{Uniform}(\text{permutations of } \{1,\dots,m \cdot b\})$
  for $\beta = 1$ **to** mb **do** `// Add repulsive loss for negative samples`
    $l = l + \mathcal{L}^r(mB_h[\beta], mB_t[\pi(\beta)])$ `// mB repeats B m times`
  $l = \frac{l}{(m+1)b}$
  Update parameters
  $\theta = \theta - \alpha \cdot \nabla_\theta l$
return $f_\theta(x_1), \dots, f_\theta(x_n)$

the high-dimensional similarities. Thus, P_{ij} follows the multinomial distribution $\text{Mult}(b, \{\frac{\mu_{rs}}{2\mu_{\text{tot}}}\}_{r,s=1,\dots,n})$ and $\mathbb{E}(P_{ij}) = \frac{b\mu_{ij}}{2\mu_{\text{tot}}}$.

Each entry of the heads B_h and tails B_t in B is repeated m times to get the negative sample pairs. We introduce the random variables H_r and T_r for $r = 1,\dots,n$, representing the number of occurrences of node r among the repeated heads and tails. N_{ij} counts how often the sampled permutation, π, of the repeated tails assigns a tail j to a head i. This can be viewed as selecting a tail from mB_t (tails repeated m times) for each of the H_i heads i without replacement. There are T_j tails that could lead to a negative sample pair ij. Therefore, N_{ij} follows a hypergeometric distribution $\text{Hyp}(mb, H_i, T_j)$. So, $\mathbb{E}_\pi(N_{ij}) = \frac{H_i T_j}{mb}$. We have

$$H_i = m \cdot \sum_{s=1}^{n} P_{is} \text{ and } T_j = m \cdot \sum_{r=1}^{n} P_{rj}. \tag{3.18}$$

Since the multinomially distributed P_{rs}'s have covariance $\mathrm{Cov}(P_{rs}, P_{r's'}) = -b\frac{\mu_{rs}\mu_{r's'}}{4\mu_{\mathrm{tot}}^2}$ for $(r,s) \neq (r',s')$ and variance $\mathrm{Var}(P_{rs}) = b\frac{\mu_{rs}}{2\mu_{\mathrm{tot}}}\left(1 - \frac{\mu_{rs}}{2\mu_{\mathrm{tot}}}\right)$, we get

$$\mathbb{E}_B(P_{rs}P_{r's'}) = \mathrm{Cov}(P_{rs}, P_{r's'}) + \mathbb{E}_B(P_{rs})\mathbb{E}_B(P_{r's'}) = b(b-1)\frac{\mu_{rs}\mu_{r's'}}{4\mu_{\mathrm{tot}}^2} \tag{3.19}$$

$$\mathbb{E}_B(P_{rs}^2) = \mathrm{Var}(P_{rs}) + \mathbb{E}_B(P_{rs})^2 = b(b-1)\frac{\mu_{rs}^2}{4\mu_{\mathrm{tot}}^2} + b\frac{\mu_{rs}}{2\mu_{\mathrm{tot}}}. \tag{3.20}$$

Thus, we compute the expectation of $\mathbb{E}_\pi(N_{ij})$ with respect to the batch assembly as

$$\begin{aligned}
\mathbb{E}_B(\mathbb{E}_\pi(N_{ij})) &= \frac{1}{mb}\mathbb{E}_B(H_iT_j) \\
&= \frac{1}{mb}\mathbb{E}_B\left(m\sum_{s=1}^{n} P_{is} \cdot m\sum_{r=1}^{n} P_{rj}\right) \\
&= \frac{m}{b}\sum_{r,s=1}^{n} \mathbb{E}_B(P_{is}P_{rj}) \\
&= \frac{m}{b}\left[\sum_{r\neq i \text{ or } s\neq j}^{n} \mathbb{E}_B(P_{is}P_{rj}) + \mathbb{E}_B(P_{ij}^2)\right] \\
&= \frac{m}{b}\left[\sum_{r,s=1}^{n} b(b-1)\frac{\mu_{is}\mu_{rj}}{4\mu_{\mathrm{tot}}^2} + b\frac{\mu_{ij}}{2\mu_{\mathrm{tot}}}\right] \\
&= m\left((b-1)\frac{d_id_j}{4\mu_{\mathrm{tot}}^2} + \frac{\mu_{ij}}{2\mu_{\mathrm{tot}}}\right).
\end{aligned} \tag{3.21}$$

Finally, as the random process of the batch assembly is independent of the choice of the permutation, we can split the total expectation up and get the expected loss

$$\begin{aligned}
&\mathbb{E}_{(B,\pi)}(\mathcal{L}^t) \\
&= \mathbb{E}_B\mathbb{E}_\pi\Bigg(-\frac{1}{(m+1)b}\sum_{i,j=1}^{n}\Big(P_{ij}\cdot\log\Big(\phi\big(f_\theta(x_i), f_\theta(x_j)\big)\Big) \\
&\qquad\qquad + N_{ij}\cdot\log\Big(1-\phi\big(f_\theta(x_i), f_\theta(x_j)\big)\Big)\Big)\Bigg) \\
&= -\frac{1}{(m+1)b}\sum_{i,j=1}^{n}\Big(\mathbb{E}_B\mathbb{E}_\pi(P_{ij})\cdot\log\Big(\phi\big(f_\theta(x_i), f_\theta(x_j)\big)\Big) \\
&\qquad\qquad + \mathbb{E}_B\mathbb{E}_\pi(N_{ij})\cdot\log\Big(1-\phi\big(f_\theta(x_i), f_\theta(x_j)\big)\Big)\Big) \\
&= -\frac{1}{(m+1)b}\sum_{i,j=1}^{n}\Big(\frac{b\mu_{ij}}{2\mu_{\mathrm{tot}}}\cdot\log\Big(\phi\big(f_\theta(x_i), f_\theta(x_j)\big)\Big) \\
&\qquad\qquad + m\Big((b-1)\frac{d_id_j}{4\mu_{\mathrm{tot}}^2} + \frac{\mu_{ij}}{2\mu_{\mathrm{tot}}}\Big)\cdot\log\Big(1-\phi\big(f_\theta(x_i), f_\theta(x_j)\big)\Big)\Big) \\
&= -\frac{1}{2(m+1)\mu_{\mathrm{tot}}}\sum_{i,j=1}^{n}\Big(\mu_{ij}\cdot\log\Big(\phi\big(f_\theta(x_i), f_\theta(x_j)\big)\Big) \\
&\qquad\qquad + m\Big(\frac{(b-1)}{b}\frac{d_id_j}{2\mu_{\mathrm{tot}}} + \frac{\mu_{ij}}{b}\Big)\cdot\log\Big(1-\phi\big(f_\theta(x_i), f_\theta(x_j)\big)\Big)\Big).
\end{aligned} \tag{3.22}$$

□

While the exact loss function for Parametric UMAP differs slightly from that of Non-Parametric UMAP in the precise value of the repulsive weight, our qualitative analysis still applies to the parametric case. We argued above that Non-Parametric UMAP's repulsive weight is $\frac{(d_i+d_j)m}{2n} \approx \frac{\log_2(k)m}{n}$, which is very small since $n \gg m, k$. The repulsive weight for Parametric UMAP is similarly small since, for large b, we have

$$m\left(\frac{(b-1)}{b}\frac{d_i d_j}{2\mu_{\text{tot}}} + \frac{\mu_{ij}}{b}\right) \approx m\frac{d_i d_j}{2\mu_{\text{tot}}} \tag{3.23}$$

$$= m\frac{d_i d_j}{\sum_{l=1}^{n} d_l} \tag{3.24}$$

$$\approx m\frac{\log_2(k)^2}{n\log_2(k)} \tag{3.25}$$

$$= \frac{\log_2(k)m}{n}. \tag{3.26}$$

3.7.1 *Relation of Parametric UMAP to modularity clustering*

The objective function of Parametric UMAP is loosely related to modularity clustering [43, 119]. The modularity of a clustering of a weighted graph is given by

$$Q = \frac{1}{2\tilde{\mu}_{tot}}\sum_{i,j=1}^{n}\left(\tilde{w}_{ij} - \frac{\tilde{d}_i\tilde{d}_j}{2\tilde{\mu}_{tot}}\right)\cdot\delta(c_i, c_j), \tag{3.27}$$

where in accordance to our notation $\tilde{w}_{ij}$ is the weight of edge ij, $\tilde{d}_i = \sum_{j=1}^{n}\tilde{w}_{ij}$ is the degree of node i, $\tilde{\mu}_{tot} = \frac{1}{2}\sum_{i,j=1}^{n}\tilde{w}_{ij}$ is the total weight of the graph, c_i is the cluster of node i, and finally δ is the Kronecker delta function. Modularity measures how much larger the edge weight within the clusters, $\sum_{i,j=1}^{n}\tilde{w}_{ij}\cdot\delta(c_i, c_j)$, is than the amount expected under the configuration model, $\sum_{i,j=1}^{n}\tilde{d}_i\tilde{d}_j/(2\tilde{\mu}_{tot})\cdot\delta(c_i, c_j)$. The *NP*-hard objective of modularity clustering is to find a clustering with maximal modularity.

Parametric UMAP's negative sampling is uniform from a batch that is itself sampled according to μ_{ij}. Thm. 3.4 shows that this yields a weighing of attractive and repulsive terms akin to modularity clustering. To relate modularity clustering further to the objective of Parametric UMAP, we observe that while $\delta(c_i, c_j)$ measures whether nodes i and j are in the same cluster, $\log(v_{ij})$ is maximal if the similarity of e_i and e_j in embedding space is high. Thus, both quantities encode whether in the output (a clustering or an embedding) i and j are deemed similar. In neighbor embeddings [66], it is typically deemed particularly bad to embed neighboring points very far apart and to embed non-neighboring points very close by. The divergence of $\log(v_{ij})$ and $\log(1-v_{ij})$ at $v_{ij}=0$ and $v_{ij}=1$, respectively, captures this intuition. However, to relate Parametric UMAP to modularity clustering, we need a single (soft) measure of affiliation to the same cluster. Also, in modularity clustering, one might place two graph nodes into the same cluster, even if they are not directly connected, but only indirectly via a chain of neighboring nodes. So, for clustering, the divergence behavior of $\log(1-v_{ij})$ is not desired. Therefore, we replace Parametric UMAP's $\log(1-v_{ij})$ by $-\log(v_{ij})$ yielding the objective function

$$\frac{1}{2(m+1)\mu_{\text{tot}}} \sum_{i,j=1}^{n} \left(\mu_{ij} \cdot \log(\nu_{ij}) + m \left(\frac{(b-1)}{b} \frac{d_i d_j}{2\mu_{\text{tot}}} + \frac{\mu_{ij}}{b} \right) \cdot (-\log(\nu_{ij})) \right) \tag{3.28}$$

$$= \frac{1}{2(m+1)\mu_{\text{tot}}} \sum_{i,j=1}^{n} \left(\left(\mu_{ij} - m \left(\frac{(b-1)}{b} \frac{d_i d_j}{2\mu_{\text{tot}}} + \frac{\mu_{ij}}{b} \right) \right) \cdot \log(\nu_{ij}) \right) \tag{3.29}$$

$$\approx \frac{1}{2(m+1)\mu_{\text{tot}}} \sum_{i,j=1}^{n} \left(\left(\mu_{ij} - m \frac{d_i d_j}{2\mu_{\text{tot}}} \right) \cdot \log(\nu_{ij}) \right), \tag{3.30}$$

where the approximation holds for large b. We see that minimizing it loosely corresponds to maximizing a quantity similar to modularity, especially for $m = 1$. We leave a more in-depth exploration of this connection for future work.

3.8 TRUE TARGET SIMILARITIES

Since the effective objective function $\tilde{\mathcal{L}}$ (3.12) that UMAP optimizes is different from $\mathcal{L}$ (3.6), we cannot hope that UMAP truly tries to find a low-dimensional embedding whose similarities reproduce the high-dimensional similarities. Nevertheless, using the effective loss $\tilde{\mathcal{L}}$, we can compute the true target similarities ν^*_{ij}, which UMAP tries to achieve in embedding space.

Proposition 3.5. *UMAP's effective loss* $\tilde{\mathcal{L}}$ (3.12) *becomes minimal for* $\nu^*_{ij} = \frac{\mu_{ij}}{\mu_{ij} + \frac{(d_i+d_j)m}{2n}}$.

Proof. The effective loss $\tilde{\mathcal{L}}$ is a sum of non-normalized binary cross-entropy loss functions

$$-\left(\mu_{ij} \cdot \log(\nu_{ij}) + \frac{(d_i + d_j)m}{2n} \cdot \log(1 - \nu_{ij}) \right), \tag{3.31}$$

which are individually minimal for

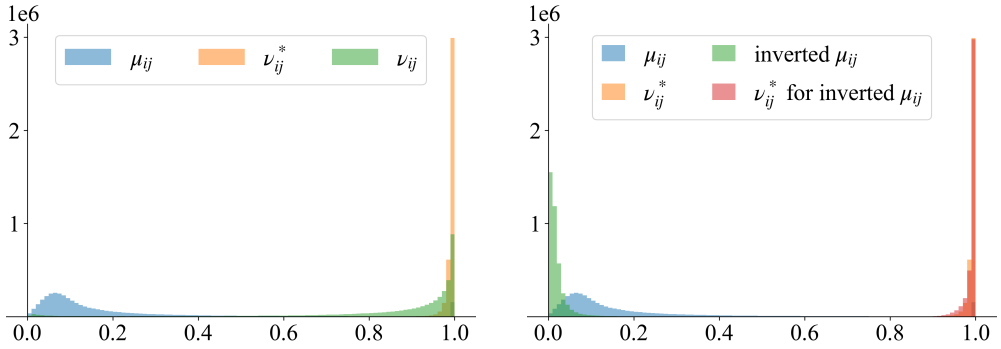
$$\nu^*_{ij} = \frac{\mu_{ij}}{\mu_{ij} + \frac{(d_i+d_j)m}{2n}}. \tag{3.32}$$

When all its summands are individually minimized, the total loss $\tilde{\mathcal{L}}$ is also minimal. □

The target similarities ν^*_{ij} are not realizable as $\nu_{ij} = \phi(\|e_i - e_j\|)$ for any point configuration $e_1, \ldots, e_n$ in $\mathbb{R}^d$, simply because the Cauchy kernel is strictly positive. Nevertheless, they show the true aim of UMAP's optimization strategy. In the typical case in which $\frac{(d_i+d_j)m}{2n} \approx 0$ the target similarities can be approximated as

$$\nu^*_{ij} = \frac{\mu_{ij}}{\mu_{ij} + \frac{(d_i+d_j)m}{2n}} \begin{cases} = 0 \text{ if } \mu_{ij} = 0 \\ \approx 1 \text{ if } \mu_{ij} > 0. \end{cases} \tag{3.33}$$

In other words, even the smallest high-dimensional similarity appears maximal compared to the massively reduced repulsion weight. Thus, the negative sampling essentially binarizes the input similarities. UMAP's high-dimensional similarities are non-zero exactly on the skNN graph edges of the high-dimensional data. Therefore, the binarization explains why Böhm *et al.* [15] find that using the binary



(a) Similarities for $\mu_{ij} > 0$

(b) Original and inverted similarities for $\mu_{ij} > 0$

Figure 3.5: Histograms of high-dimensional (μ_{ij}), target (ν^*_{ij}) and low-dimensional (ν_{ij}) similarities on the C. elegans dataset [116, 126] for pairs with positive high-dimensional similarity. **3.5a:** The similarities of UMAP's low-dimensional embedding reproduce the target similarities instead of the high-dimensional ones. The target similarities are heavily skewed towards one. **3.5b:** Comparison of positive input and target similarities for the original and inverted input similarities. While the histograms of the input similarities differ noticeably, their target similarities do not, and neither do the embedding similarities (not shown). The binarization essentially ignores all information beyond the skNN graph.

weights of the skNN graph does not deteriorate UMAP's performance much.[6] The binarization even helps UMAP to overcome disrupted high-dimensional similarities, as long as only the edges of the skNN graph have non-zero weight. In Fig. 3.3b, we invert the original positive high-dimensional weights on the C. elegans dataset. That means that the k-th nearest neighbor will have a higher weight than the nearest neighbor. The resulting visualization even improves on the original by keeping the layout more compact. This underpins Böhm *et al.* [15]'s claim that the elaborate theory used to compute the high-dimensional similarities is not the reason for UMAP's practical success. In fact, we show that UMAP's optimization scheme even actively ignores most information beyond the skNN graph. In Fig. 3.5, we show histograms of the various notions of similarity for the C. elegans dataset. We see in Fig. 3.5b how the binarization equalizes the positive target similarities for the original and the inverted high-dimensional similarities.

UMAP's tendency for over-contraction is already strong for small datasets. Consider UMAP's default setting of $k = 15$ and $m = 5$. Then for datasets with $n = 500$ points each input similarity $\mu_{ij} > 0.2$ is mapped to a target similarity $\nu^*_{ij} > 0.83$, see Sec. 3.9.4.

The binary cross entropy terms in the effective loss $\tilde{\mathcal{L}}$ (3.12) are not normalized. This leads to a different weighing of the binary cross-entropy terms for each pair ij

$$\tilde{\mathcal{L}} = 2 \sum_{1 \le i < j \le n} \left(\mu_{ij} \cdot \mathcal{L}^a_{ij} + \frac{(d_i + d_j)m}{2n} \cdot \mathcal{L}^r_{ij} \right) \tag{3.34}$$

$$= -2 \sum_{1 \le i < j \le n} \left[\left(\mu_{ij} + \frac{(d_i + d_j)m}{2n} \right) \cdot \left(\nu^*_{ij} \log(\nu_{ij}) + (1 - \nu^*_{ij}) \log(1 - \nu_{ij}) \right) \right]. \tag{3.35}$$

6 Böhm *et al.* [15] used a scaled version of the skNN graph, but the scaling factor cancels for the target weights.

As $\frac{(d_i+d_j)m}{2n}$ is very small for large datasets, the term $\mu_{ij} + \frac{(d_i+d_j)m}{2n}$ is dominated by μ_{ij}. Hence, the reduced repulsion not only binarizes the high-dimensional similarities but also puts higher weight on the positive than the zero target similarities. Therefore, we can expect that the positive target similarities are better approximated by the embedding similarities than the zero ones. Indeed, Fig. 3.5a shows that the low-dimensional similarities match the positive target similarities very well, as expected from the weighted BCE reading of the effective loss function (3.35).

3.8.1 *Explaining artifacts in UMAP visualizations*

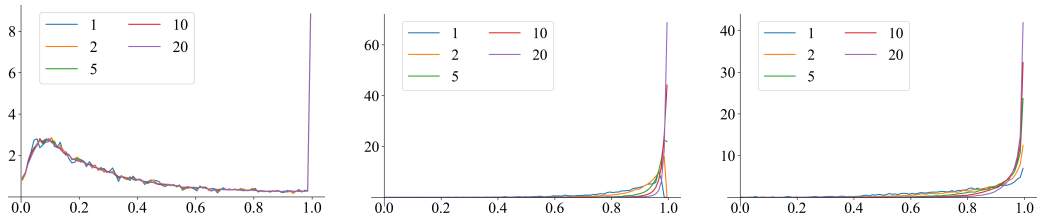
We conclude this section by explaining the observed artifacts of UMAP's visualization in Figs. 3.2 and 3.3 in the light of the above analysis. The regular UMAP optimization contracts the ring in Fig. 3.2b even when initialized at the original layout because the reduced repulsion yields nearly binary target similarities. All pairs that are part of the skNN graph not only want to be sufficiently close that their high-dimensional similarity is reproduced but so close that their similarity is nearly one. The fact that the effective loss weighs the terms with target similarity near one much more than those with target similarity near zero reinforces this trend. As a result, the ring gets contracted. The same argument applies to the over-contracted parts of the UMAP visualization of the C. elegans dataset in Fig. 3.3.

When we increase the dataset size by adding more rings, as in Fig. 3.2c, negative sample pairs can come from different rings, reducing the frequency that a pair of points from the same ring is sampled. This reduces the repulsion further. Thus, there is stronger binarization, and the embedding looks crisper. If we sample a single ring more densely, over-contraction is also increased. However, the visual appearance is more nuanced as over-contraction can appear in the form of densely clustered and line-like substructures within the ring width, see Sec. 3.9.3.

Our framework can also explain the opposite behavior of UMAP when the dense similarities are used as input similarities in Fig. 3.2d. In this setting, the average degree of a node is about 100. With a negative sample rate $m = 5$ and a dataset size $n = 1000$ this yields repulsive weights $\frac{(d_i+d_j)m}{2n} \approx 0.5$. Thus, we increase the repulsion on pairs with high input similarity but decrease it on pairs with low input similarity. Now, the target similarities are not a binarization of the input similarities but instead skewed towards 0.5. Hence, we can expect embedding points to increase their distance to nearest neighbors but distant points to move closer towards each other. This is what we observe in Fig. 3.2d, where the width of the ring has increased and the ring curves to bring distant points closer together.

3.9 UMAP'S DEPENDENCE ON THE DATASET SIZE

UMAP's true loss function (3.12) depends on the dataset, in particular on its size n. As discussed in Sec. 3.8, the reduction of repulsion, the binarization, and the over-contraction will all be stronger for larger datasets. This section discusses this observation in more detail and explores the effect empirically.



(a) Input similarities μ_{ij} (b) Target similarities ν^*_{ij} (c) Embedding similarities ν_{ij}

Figure 3.6: Distributions of input (μ_{ij}), target (ν^*_{ij}), and embedding similarities (ν_{ij}) for pairs with positive input similarity from datasets with a varying number of rings. All distributions are normalized to one. The legends indicate the number of rings in the dataset. **3.6a:** The high-dimensional similarities do not change qualitatively between the datasets. Their distribution only gets smoother as the dataset gets larger. **3.6b:** The larger the dataset, the stronger the binarization, and thus the more the distributions of target similarities are skewed to one. **3.6c:** The larger the dataset, the stronger the over-contraction, and thus the more the distributions of embedding similarities are skewed to one.

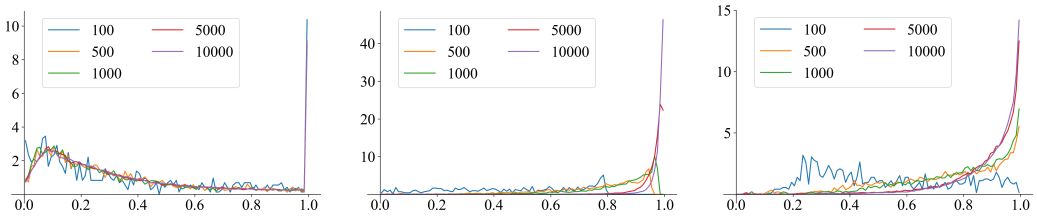
3.9.1 *Multiple rings experiments*

We applied UMAP to datasets containing multiple 2D rings of 1000 points each. The rings were spaced so that no point in one ring had any of its k nearest neighbors in another ring. As the input similarities μ_{ij} only depend on the distances to the k nearest neighbors, this ensures that the input similarities for points from one ring are the same as if the dataset consisted only of that ring. The input similarities between points in different rings are all zero.

We varied the number of rings from 1 to 20. We formed the datasets by iteratively adding rings, so the datasets with fewer rings are subsets of those with more rings. To ensure convergence, we ran the UMAP optimization for 10000 epochs as for the single ring experiments in Fig. 3.2. Moreover, we initialized the optimization at the original layout. Other hyperparameters were kept at their default values. The results can be found in Fig. 3.8.

With this setup, one might expect that the embedding of the first ring would look similar for all datasets. However, we observe that the embedding of the rings gets crisper the more rings the dataset contains. The repulsion weight in our effective loss function (3.12) depends on the degrees of the points, the negative sample rate, and the dataset size n. Since the input similarities are the same as in individual rings by construction, the degree of each point is independent of the presence of other rings. However, the dataset size, of course, increases with the number of rings so that the repulsion gets smaller for the larger datasets. Viewed from the point of the target similarities (3.32), the binarization gets stronger. This explains why we see more over-contraction. In Fig. 3.6, we show histograms for input, target, and embedding similarities (as graphs for visual clarity), which confirm that the input similarities do not change fundamentally, but the target and thus the embedding similarities get more skewed towards one as the number of rings increases.

We can also give a more concrete explanation for the reduced repulsion. The edges for attraction are sampled equally frequently, independent of the number of rings. For each such sampled attractive edge, m negative samples are generated. However, these may come from any of the rings in the dataset. So if there are



(a) Input similarities μ_{ij} (b) Target similarities v^*_{ij} (c) Embedding similarities v_{ij}

Figure 3.7: Distributions of input (μ_{ij}), target (v^*_{ij}), and embedding similarities (v_{ij}) for pairs of points with positive input similarity from single ring datasets with a varying number of points. All distributions are normalized to one. The legends indicate the number of sample points of the ring. **3.6a:** The high-dimensional similarities do not change qualitatively between the datasets. Their distribution only gets smoother as the dataset gets larger. **3.6b:** The larger the dataset, the stronger the binarization, and thus the more the distributions of target similarities are skewed to one. In particular, for smaller datasets, the target similarities are bound away from one by equation (3.32). **3.6c:** The larger the dataset, the stronger the over-contraction, and thus the more the distributions of embedding similarities are skewed to one.

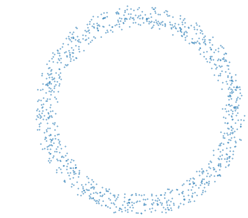
more rings, it is more likely that a negative sample will come from a different ring. Consequently, it gets less likely that it comes from the same ring. So repulsion within the ring is decreased while the attraction within the ring remains the same, explaining the more contracted embeddings.

3.9.2 *Varying the sample size of a single ring*

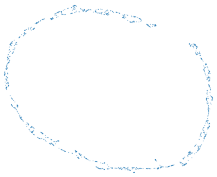
In this set of experiments, we computed UMAP embeddings of datasets consisting of a varying number of points sampled from a 2D ring. We sampled between 100 and 10000 points. Like for multiple rings above, our analysis predicts that the larger the dataset, the stronger the reduction in repulsion, the binarization, and the over-contraction. Indeed, this is what we see in Fig. 3.7, where we show the distributions of input, target, and embedding similarities for pairs of points with positive input similarity.

Because of UMAP's uniformity assumption, the size of the input similarities depends on the relative distances of a point to its k nearest neighbors. Since the points are always sampled uniformly from the ring, we can expect that the distribution of input similarities does not change qualitatively as we increase the sample size, confirmed in Fig. 3.7a. For the target similarities, a larger number of points implies that they should be more skewed towards one, as shown in Fig. 3.7b. Note that for the smallest datasets, speaking of binarization is not quite justified. E.g., the dataset with 100 points has a typical repulsion strength of $\frac{(d_i+d_j)m}{2n} \approx 0.2$, so that the largest possible target similarity is $1/1.2 \approx 0.83$. Finally, the embedding similarities also get skewed more towards one as the dataset gets larger, indicating increasing over-contraction.

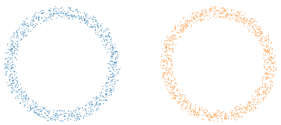
As usual, we optimized the UMAP embeddings for 10000 epochs, initialized at the original positions, and kept all other hyperparameters at their defaults. The resulting embeddings can be found in Fig. 3.10. While our theory predicts increasing over-contraction as the dataset gets larger, corroborated by the similarity



(a) Dataset with 1 ring

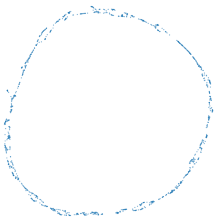


(b) UMAP embedding



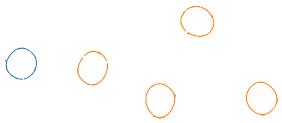
(c) Dataset with 2 rings

(d) UMAP embedding

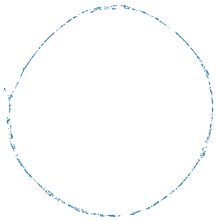


(e) Embedding of blue ring in 3.8d

(f) Dataset with 5 rings

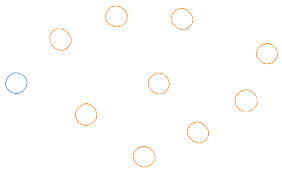


(g) UMAP embedding

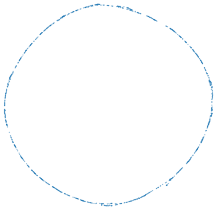


(h) Embedding of blue ring in 3.8g

(i) Dataset with 10 rings

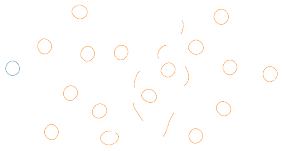


(j) UMAP embedding

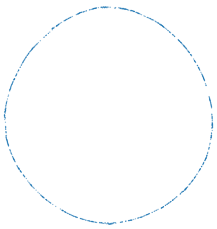


(k) Embedding of blue ring in 3.8j

(l) Dataset with 20 rings



(m) UMAP embedding



(n) Embedding of blue ring in 3.8m

Figure 3.8: UMAP's embedding of datasets with multiple rings. **Left column:** Datasets of multiple 2D rings à 1000 points per ring. Datasets with fewer points are subsets of those with more points. The blue ring is the same in all datasets. The larger datasets are plotted smaller for shortage of space. **Middle column:** UMAP embedding of the dataset on the left. The larger the dataset, the more contracted the ring width. Embeddings of larger datasets are plotted smaller for shortage of space. **Right column:** Embedding of the blue ring at the same size. The larger the dataset, the crisper the embedding becomes, although the blue ring is the same in all datasets. Figure best viewed digitally.

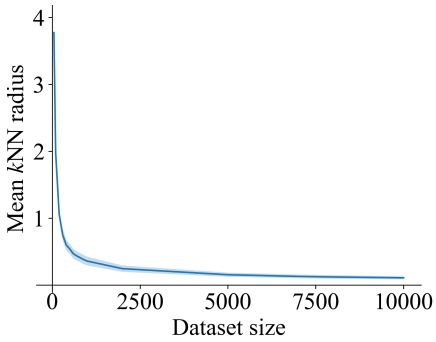
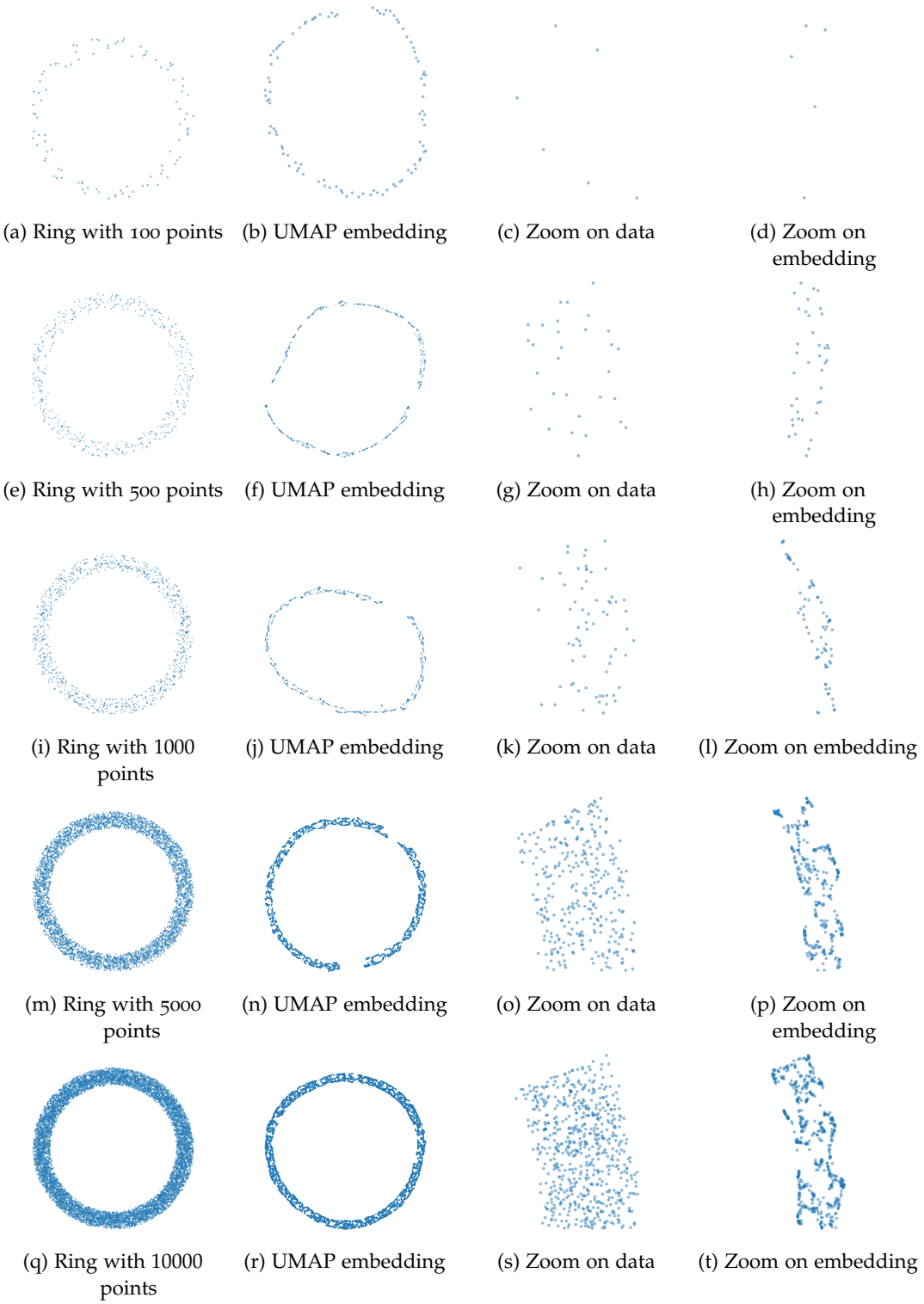


Figure 3.9: Average distance to the k-th nearest neighbor in ring datasets with varying density. The average is over all points in each dataset. One standard deviation is given. The larger the number of samples, the smaller the distance to the k-th nearest neighbor.

distributions in Fig. 3.7, the embedding of the dataset with 500 points appears to have the smallest ring width, while the embeddings of the dataset with 100 or of those with more than 500 points seem to have larger ring widths. In particular, for the dataset of 10000 points, the apparent ring width is nearly as large as in the input data. Nevertheless, this is not at odds with our predictions: An over-contracted ring width is only one way over-contraction can manifest. In the zoomed-in segments of the ring embeddings in Fig. 3.10, we see that the structure of points within the embedding is far from uniform: Points cluster or form lines, much like in Fig. C.9, where we apply UMAP to uniform data. The predicted and measured over-contraction is present but on a smaller scale than the ring width, so it is not immediately visible from the global embedding. Note that these are spurious artifacts of UMAP's optimization procedure as the original data is uniform.

We conjecture that the reason for whether one observes over-contraction on a global scale has to do with the relative size of the local neighborhoods and the width of the ring. If the local neighborhood is large relative to the width of the ring, it is more likely to be contracted to a line. If it is small, the over-contraction happens within the width of the ring.

Another way of looking at this phenomenon is that for fixed k but increasing sample size of the toy ring, the distance to the k-th nearest neighbor and thus the radius of the local neighborhood decrease relative to the fixed width of the whole ring. We computed the mean distance to the k-th nearest neighbor over all data points and plotted the results in Fig. 3.9. At 500 data points, when the width of the embedding has nearly reached 0, the mean k-th nearest neighbor radius is about 0.54. Hence, some points' local neighborhoods span the ring's width. At 1000 points, when the embedding ring width increases again, the average k-th nearest neighbor radius is only 0.35, smaller than the original ring width of 1. We note for completeness that for some toy rings with few points, the ring structure was torn up to a line or even a set of clusters (not shown). For the dataset with only 100 points, the mean distance to the k-th nearest neighbor is about 2, but the small number of data points reduces the repulsion strength not enough for the embedding to become visibly over-contracted.



(a) Ring with 100 points (b) UMAP embedding (c) Zoom on data (d) Zoom on embedding

(e) Ring with 500 points (f) UMAP embedding (g) Zoom on data (h) Zoom on embedding

(i) Ring with 1000 points (j) UMAP embedding (k) Zoom on data (l) Zoom on embedding

(m) Ring with 5000 points (n) UMAP embedding (o) Zoom on data (p) Zoom on embedding

(q) Ring with 10000 points (r) UMAP embedding (s) Zoom on data (t) Zoom on embedding

Figure 3.10: UMAP embedding of rings with a varying number of points. **First column:** Datasets of rings in 2D. **Second column:** UMAP embedding of the dataset on the left. The embedding of the dataset with 500 points, 3.10f, seems most crisp, although it is not the largest dataset. **Third column:** Zoom of the original data within the angular segment $[0, \pi/8]$. **Fourth column:** Zoom on the embedding of the points that are in the third column. We see that for the larger datasets, there is over-contraction within the width of the ring in the form of clustered and line-like substructures. Figure best viewed digitally.

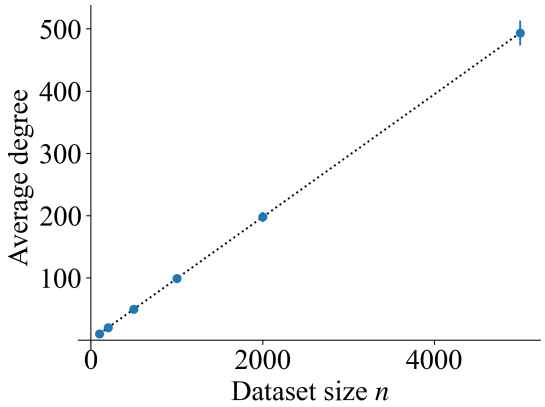


Figure 3.11: Average degree of a point in the 2D ring datasets of varying size of Fig. 3.12. The average is over all points in the dataset, and the error bar is one standard deviation. We used the dense input similarities. The dotted line is the best linear fit, which perfectly explains the data.

Since applying UMAP on an independent subset or a down- or up-sampled version of a dataset are both standard practices, we expect the resulting artifacts to appear often in practice.

3.9.3 *Varying the sample size for dense input similarities*

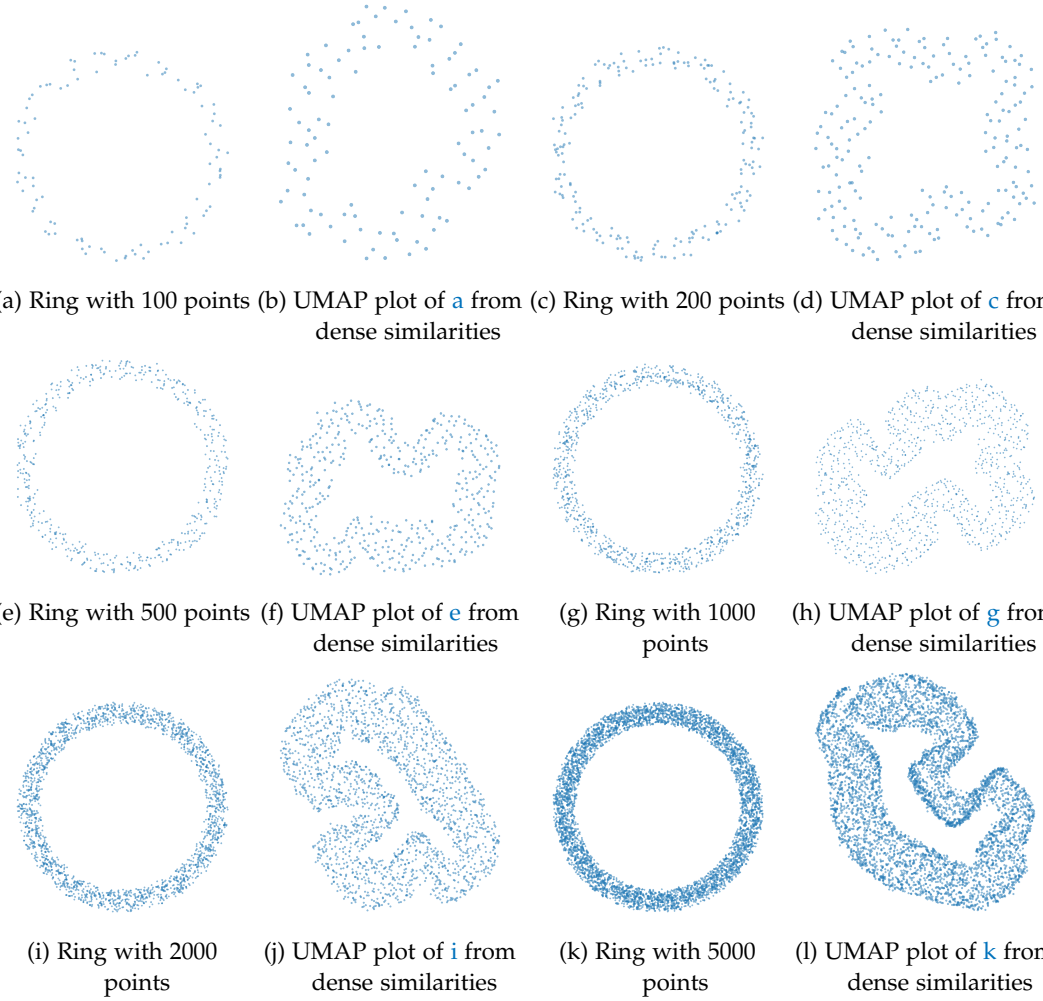
We also computed UMAP embeddings based on the dense input similarities $\mu_{ij} = \phi(||x_i - x_j||)$ for rings with varying number of samples. Here, the analysis differs from the typical, s*k*NN-based input similarities. As the dataset size increases, each point is similar to more points. Therefore, the degree d_i for a point i scales with the dataset size, and the effective repulsive pre-factor $\frac{(d_i+d_j)m}{2n}$ does not decrease with the dataset size but remains approximately constant. We show the linear scaling of the average degree with the dataset size in Fig. 3.11. Hence, we do not expect any qualitative difference in this setting as we vary the number of points in the ring. Indeed, experiments in which we changed the number of points of the ring from 100 to 5000 confirm this prediction, see Fig. 3.12.

3.9.4 *Discussion of the influence of m, k and n*

The effective repulsion weight $\frac{(d_i+d_j)m}{2n}$ and thus the transformation from input to target similarities depends on the degrees of nodes i and j, the negative sample rate m and the dataset size n. As discussed in Sec. 3.4, the degree $d_i = \sum_{j=1}^{n} \mu_{ij}$ is typically close to $\log_2(k)$. Plugging this into equation (3.32) shows that the target similarities of UMAP's implementation are roughly

$$v_{ij}^* \approx \frac{\mu_{ij}}{\mu_{ij} + \log_2(k)m/n}. \tag{3.36}$$

This shows how the relation between target and high-dimensional similarities depends on k, m, and n. Note that there are no parameter settings for m and k that



(a) Ring with 100 points (b) UMAP plot of a from dense similarities (c) Ring with 200 points (d) UMAP plot of c from dense similarities

(e) Ring with 500 points (f) UMAP plot of e from dense similarities (g) Ring with 1000 points (h) UMAP plot of g from dense similarities

(i) Ring with 2000 points (j) UMAP plot of i from dense similarities (k) Ring with 5000 points (l) UMAP plot of k from dense similarities

Figure 3.12: Datasets of 2D rings with a varying number of points and their UMAP's embeddings from dense input similarities. Qualitatively the UMAP embedding does not change with the size of the dataset.

allow the (near) exact reconstruction of the whole value range of input similarities μ_{ij}'s for a fixed n as this would require

$$\log_2(k)m/n = 1 - \mu_{ij} \forall i, j. \tag{3.37}$$

For instance, we can see in Fig. 3.5 that most non-zero high-dimensional similarities are either 1 or close to 0.05 for the C. elegans dataset with $k = 30$ and $n = 86024$. To preserve the μ_{ij}'s that equal 1, we would need

$$m = (1 - 1) \cdot 86024 / \log_2(30) = 0, \tag{3.38}$$

while to preserve the μ_{ij}'s near 0.05, we would need

$$m = (1 - 0.05) \cdot 86024 / \log_2(30) \approx 16500 \tag{3.39}$$

negative samples, which would be prohibitively slow computationally. Changing k would additionally change the algorithm's notion of locality, which has implications on both speed and appearance.

As another example, consider UMAP's default setting of $k = 15$ and $m = 5$. Then already for $n = 500$ each input similarity $\mu_{ij} > 0.2$ would be mapped to a target similarity

$$v_{ij}^* \gtrapprox \frac{0.2}{0.2 + \frac{5\log_2(15)}{500}} \approx 0.83. \tag{3.40}$$

So even for relatively small datasets, the over-contraction can be strong, even if the target similarities are not necessarily binary.

It is tempting to adapt m and k to the dataset size so that $\log_2(k)m/n$ remains constant, and thus UMAP becomes less dependent on the dataset size. However, as mentioned above, increasing m with n makes UMAP's run time quadratic in n, and increasing $\log_2(k)$ with n makes UMAP's scale exponentially with n and would also change the notion of locality. The most severe problem that we see with keeping the repulsion pre-factor $\frac{(d_i+d_j)m}{2n}$ constant is that this means that the total amount of repulsion acting on a single embedding point scales with the dataset size while the total amount of attraction remains constant. This way, we risk arriving at diverged embeddings for sufficiently large datasets, as observed by Böhm *et al.* [15] and discussed in Section 3.11.1. While the inverse relation of the repulsion pre-factor with the dataset size can lead to over-contraction on the skNN graph edges, it also counteracts the quadratic number of non-skNN graph edges.

3.10 NEGATIVE SAMPLING IN LARGEVIS

Our analysis also applies to the visualization method LargeVis [153], which uses a very similar optimization scheme to UMAP, but with a slightly different negative sampling distribution. Its intended loss function is given in [153] as

$$-\sum_{ij\in \text{s}k\text{NN graph}} \mu_{ij}\log(v_{ij}) - \sum_{ij\notin \text{s}k\text{NN graph}} \gamma\log(1-v_{ij}), \tag{3.41}$$

with a constant γ, set to 7 per default in [153]. Different from UMAP, μ_{ij} and v_{ij} are the high- and low-dimensional similarities from t-SNE [160]; in particular the μ_{ij} are only non-zero on the edges of the skNN graph. For optimization, Tang *et al.* [153] employ negative sampling and arrive at the following, different objective function

$$\mathcal{L}^{\text{LargeVis}} = -\sum_{ij\in \text{s}k\text{NN graph}} \mu_{ij}\left(\log(v_{ij}) + \sum_{s=1}^{m} \mathbb{E}_{j_s\sim P(a)}\left(\gamma\log(1-v_{ij_s})\right)\right), \tag{3.42}$$

with negative sample distribution $P(a) \propto d_a^{0.75}$. As for UMAP, we can compute a closed form loss function:

Theorem 3.6. *The loss function of LargeVis [153] is*

$$\mathcal{L}^{\text{LargeVis}} = -\sum_{1\le i<j\le n}\left(\mu_{ij}\log(v_{ij}) + \frac{m\gamma(d_i d_j)^{0.75}(d_i^{0.25}+d_j^{0.25})}{2\sum_{l=1}^{n} d_l^{0.75}}\log(1-v_{ij})\right). \tag{3.43}$$

Proof. We can compute the expectation and rearrange, similar to UMAP's loss.

$$
\begin{aligned}
\mathcal{L}^{\text{LargeVis}} &= -\sum_{ij \in s k\text{NN graph}} \mu_{ij} \left(\log(\nu_{ij}) + \sum_{s=1}^{m} \mathbb{E}_{j_s \sim P(a)} \big(\gamma \log(1 - \nu_{ij_s}) \big) \right) \\
&= -\frac{1}{2} \sum_{1 \le i,j \le n} \mu_{ij} \left(\log(\nu_{ij}) + \sum_{s=1}^{m} \mathbb{E}_{j_s \sim P(a)} \big(\gamma \log(1 - \nu_{ij_s}) \big) \right) \\
&= -\frac{1}{2} \sum_{1 \le i,j \le n} \mu_{ij} \left(\log(\nu_{ij}) + m\gamma \sum_{\alpha=1}^{n} \left(\frac{d_\alpha^{0.75}}{\sum_{l=1}^{n} d_l^{0.75}} \log(1 - \nu_{i\alpha}) \right) \right) \\
&= -\frac{1}{2} \sum_{1 \le i,j \le n} \mu_{ij} \log(\nu_{ij}) - \frac{m\gamma}{2} \sum_{1 \le i,j \le n} \left(\mu_{ij} \sum_{\alpha=1}^{n} \left(\frac{d_\alpha^{0.75}}{\sum_{l=1}^{n} d_l^{0.75}} \log(1 - \nu_{i\alpha}) \right) \right) \\
&= -\frac{1}{2} \sum_{1 \le i,j \le n} \mu_{ij} \log(\nu_{ij}) - \frac{m\gamma}{2} \sum_{1 \le i,\alpha \le n}^{n} \left(\frac{d_\alpha^{0.75}}{\sum_{l=1}^{n} d_l^{0.75}} \log(1 - \nu_{i\alpha}) \sum_{j=1}^{n} \mu_{ij} \right) \\
&= -\frac{1}{2} \sum_{1 \le i,j \le n} \mu_{ij} \log(\nu_{ij}) - \frac{m\gamma}{2} \sum_{1 \le i,\alpha \le n}^{n} \left(\frac{d_i d_\alpha^{0.75}}{\sum_{l=1}^{n} d_l^{0.75}} \log(1 - \nu_{i\alpha}) \right) \\
&= -\frac{1}{2} \sum_{1 \le i,j \le n} \mu_{ij} \log(\nu_{ij}) - \frac{1}{2} \sum_{1 \le i,\alpha \le n}^{n} \left(\frac{m\gamma (d_i d_\alpha)^{0.75} (d_i^{0.25} + d_\alpha^{0.25})}{2 \sum_{l=1}^{n} d_l^{0.75}} \log(1 - \nu_{i\alpha}) \right) \\
&= -\frac{1}{2} \sum_{1 \le i,j \le n} \mu_{ij} \log(\nu_{ij}) - \frac{1}{2} \sum_{1 \le i,j \le n}^{n} \left(\frac{m\gamma (d_i d_j)^{0.75} (d_i^{0.25} + d_j^{0.25})}{2 \sum_{l=1}^{n} d_l^{0.75}} \log(1 - \nu_{ij}) \right) \\
&= -\sum_{1 \le i < j \le n} \left(\mu_{ij} \log(\nu_{ij}) + \frac{m\gamma (d_i d_j)^{0.75} (d_i^{0.25} + d_j^{0.25})}{2 \sum_{l=1}^{n} d_l^{0.75}} \log(1 - \nu_{ij}) \right). \qquad (3.44)
\end{aligned}
$$

First, we used the fact that μ_{ij} is zero outside of the skNN graph, in particular that $\mu_{ii} = 0$. Second, we computed the expectation

$$
\mathbb{E}_{j_s \sim P(a)} \big(\log(1 - \nu_{ij_s}) \big) = \sum_{\alpha=1}^{n} \left(\frac{d_\alpha^{0.75}}{\sum_{l=1}^{n} d_l^{0.75}} \log(1 - \nu_{i\alpha}) \right). \qquad (3.45)
$$

Here, we use the convention for the undefined term $\log(1 - \nu_{ii}) = \log(0)$ described in the beginning of Appendix C.1. Third and fourth, we rearranged the summations. Fifth, we computed the degree d_i, and sixth, we used the symmetry $\nu_{i\alpha} = \nu_{\alpha i}$. Seventh, we relabelled α to j, and finally, we rearranged the summation one last time using both $\mu_{ii} = 0$ and the convention for $\log(0)$. □

LargeVis does not motivate its loss function as a sum of binary cross-entropy losses for each edge but instead as attraction on edges of the skNN graph plus constant repulsion on all non-skNN edges. Nevertheless, the negative sampling-based optimization turns this into a sum of non-normalized binary cross-entropy losses as for UMAP. While we have not empirically computed values of the repulsion pre-factor, we firmly believe that it is tiny since the denominator contains a sum over the entire dataset. In this case, what we called "target similarities" would look binary for LargeVis, too. This puts LargeVis in even closer proximity to UMAP as the different choice of input similarity μ_{ij} or negative sampling distribution in both methods matters little and helps to explain why LargeVis and UMAP embeddings often look very similar.

3.11 DISCUSSION

3.11.1 *Balancing attraction and repulsion*

By deriving UMAP's proper loss function and target similarities, we are able to explain several peculiar properties of UMAP visualizations. According to our analysis, UMAP does not aim to reproduce the high-dimensional UMAP similarities in low dimension but rather the binary weights of the skNN graph of the input data. This raises the question of just what part of UMAP's optimization leads to its excellent visualization results. Apparently, the exact formula for the repulsive weights is not crucial as it differs for Non-Parametric UMAP and Parametric UMAP, while both produce similarly high-quality embeddings. A first tentative step toward an explanation might be the different weighing of the BCE terms in the effective loss function (3.35). Focusing more on the similar rather than the dissimilar pairs might help to overcome the imbalance between an essentially linear number of attractive and a quadratic number of repulsive pairs. Inflated attraction was also found beneficial for t-SNE, in the form of early exaggeration [102].

Put another way, the decreased repulsive weights result in comparable total attractive and repulsive weights, which might facilitate the SGD-based optimization. Indeed, the total attractive weight in UMAP's effective loss function is $2\mu_{\text{tot}} = \sum_{i,j=1}^{n} \mu_{ij}$ and the total repulsive weight amounts to $2m\mu_{\text{tot}} = \sum_{i,j=1}^{n} \frac{(d_i+d_j)m}{2n}$ for Non-Parametric UMAP. For Parametric UMAP the total attractive and repulsive weights are $1/(m+1)$ and $m/(m+1)$, respectively. For the default value of $m = 5$, the total attractive and repulsive weights are roughly of the same order of magnitude. Moreover, we observe in Fig. 3.4 that the resulting attractive and repulsive losses are also of comparable size. Using UMAP's purported loss function, however, would yield dominating repulsion.

3.11.2 *Improved UMAP*

The reason for the binarization of the input similarities is the reduced repulsion on the pairs of points that are linked in the skNN graph. As discussed above, it might not be desirable to increase the repulsion on all pairs of points to $1 - \mu_{ij}$. However, it might be useful to change the optimization procedure of UMAP to decrease the repulsion per edge only on the non-skNN graph edges and not on the skNN graph edges. In addition to the positive and negative sampling of UMAP, we suggest to explicitly sample the skNN graph edges for repulsion with probability $1 - \mu_{ij} - (d_i + d_j)m/(2n)$. This would correct the repulsion weight for edges in the skNN graph. The loss function would read

$$-2\Bigg[\sum_{ij\in k\text{NN graph}} \Big(\mu_{ij}\log(v_{ij}) + (1-\mu_{ij})\log(1-v_{ij})\Big) \tag{3.46}$$

$$+\sum_{ij\notin k\text{NN graph}} \left(\frac{(d_i+d_j)m}{2n}\log(1-v_{ij})\right)\Bigg]. \tag{3.47}$$

Now, the target similarities are precisely the high-dimensional similarities μ_{ij} for all pairs i, j. Hence, in an optimal embedding, all μ_{ij}'s are exactly reproduced and

not just a binary version. We hope this will improve the embedding quality and decrease over-contraction. Crucially, the amount of repulsion per data point is kept constant and is not increasing with n. For the quadratic number of non-skNN graph edges, the individual repulsion pre-factors still decrease with $1/n$. The total loss function is a sum of normalized cross-entropy terms for the edges in the skNN graph and of non-normalized and drastically down-weighted cross-entropy terms for the edges not in the skNN graph. Since not all $O(n^2)$ negative pairs are considered explicitly and the additional sampling of the skNN graph edges for repulsion only scales with $O(n)$, the complexity of computing the embedding would still scale linearly as $O(kmn \cdot \mathrm{num}_{\mathrm{epochs}})$ and not quadratically in n.

A more in-depth investigation of why precisely balanced attraction and repulsion is beneficial for a useful embedding and implementation of the improved sampling scheme are left for future work.

3.12 CONCLUSION

In this work, we investigated UMAP's optimization procedure in depth. In particular, we computed UMAP's effective loss function analytically and found that it differs slightly between the non-parametric and parametric versions of UMAP and significantly from UMAP's alleged loss function. The optimal solution of the effective loss function is typically a binarized version of the high-dimensional similarities. This shows why the sophisticated form of the high-dimensional UMAP similarities does not add much benefit over the skNN graph. Instead, we conjecture that the resulting balance between attraction and repulsion is the main reason for UMAP's excellent visualization capability. Our analysis can explain some artifacts of UMAP visualizations, particularly its tendency to produce over-contracted embeddings, which gets stronger for larger datasets.

4

CONTRASTIVE LEARNING UNIFIES *t*-SNE AND UMAP

Neighbor embedding methods *t*-SNE and UMAP are the de facto standard for visualizing high-dimensional datasets. They appear to use very different loss functions with different motivations, and their exact relationship used to be unclear. Here, we clarify the relationship between *t*-SNE and UMAP: We show that UMAP effectively applies negative sampling to the *t*-SNE loss function. We explain the difference between negative sampling and noise-contrastive estimation (NCE), which has been used to optimize *t*-SNE and is called "NCVis" in this setting. We prove that, unlike NCE, negative sampling learns a scaled data distribution. When applied in the neighbor embedding setting, it yields more compact embeddings with increased attraction, explaining differences in appearance between UMAP and *t*-SNE. Further, we generalize the notion of negative sampling and obtain a spectrum of embeddings, encompassing visualizations similar to *t*-SNE, NCVis, and UMAP. Finally, we explore the connection between representation learning in the SimCLR setting and neighbor embeddings, and show that (i) *t*-SNE can be optimized with the InfoNCE loss and it also works in a parametric setting; (ii) various contrastive losses with only a few noise samples can yield competitive performance in the SimCLR setup. This chapter is based on [38].

4.1 INTRODUCTION

Low-dimensional visualization of high-dimensional data is a ubiquitous step in exploratory data analysis, and the toolbox of visualization methods has been rapidly growing in the last years [1, 112, 116, 152, 153, 165]. Since all of these methods necessarily distort the true data layout [24, 71], it is beneficial to have different tools at one's disposal. However, only with a theoretical understanding of the aims of and relationships between different methods, practitioners can make informed decisions about which visualization to use for which purpose and how to interpret the results.

The state-of-the-art for non-parametric, non-linear dimensionality reduction employs the neighbor embedding framework [66]. The two most popular examples of neighbor embedding methods are *t*-SNE [160, 161] and UMAP [112]. Prior work [15, 39, 84] has peeled away layers of complexity of *t*-SNE and UMAP to separate key ingredients from less essential design choices. The main conceptual difference between *t*-SNE and UMAP lies in their loss functions, which are motivated very differently. Recent work [15, 39] found that UMAP's application of negative sampling [113], avoiding the quadratic number of repulsive interactions between embedding points, drastically alters its loss function and decreases the repulsion compared to *t*-SNE. Nevertheless, a conceptual connection between their loss functions is still missing.

Our work fills this gap. We describe a new, precise connection between negative sampling and noise-contrastive estimation [61, 62], which has recently been applied

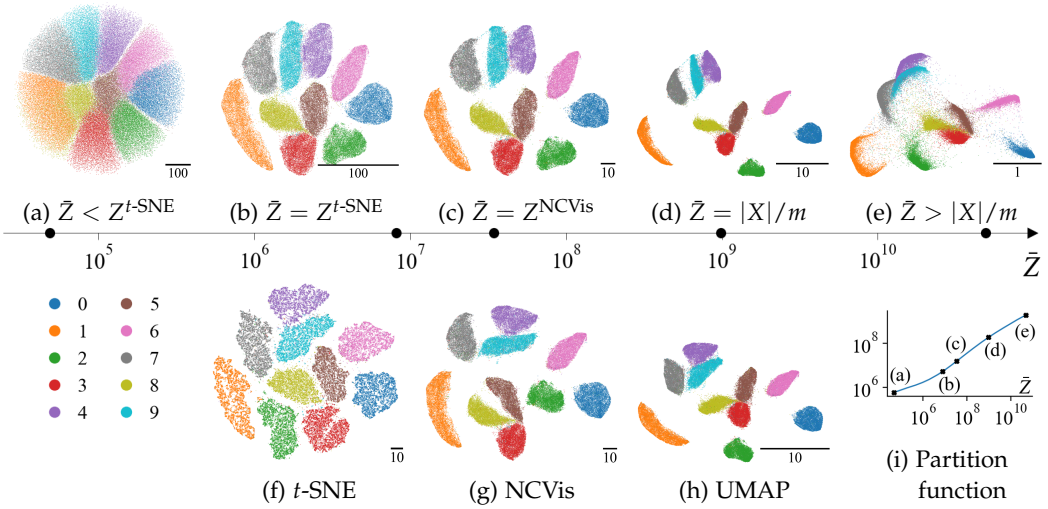


Figure 4.1: **(a – e)** Neg-t-SNE embeddings of the MNIST dataset for various values of the fixed normalization constant $\bar{Z}$. As $\bar{Z}$ increases, the scale of the embedding decreases, and clusters become more compact and separated before eventually starting to merge. The Neg-t-SNE spectrum produces embeddings very similar to those of **(f)** t-SNE, **(g)** NCVis, and **(h)** UMAP, when $\bar{Z}$ equals the partition function of t-SNE, the learned normalization parameter Z of NCVis, or $|X|/m = \binom{n}{2}/m$ used by UMAP, as predicted in Sec. 4.4–4.6. **(i)** The partition function $\sum_{ij}(1+d_{ij}^2)^{-1}$ tries to match $\bar{Z}$ and grows with it. Here, we initialize all Neg-t-SNE runs using $\bar{Z} = |X|/m$; without this 'early exaggeration', low values of $\bar{Z}$ yield fragmented clusters (Fig. 4.2).

to t-SNE and was dubbed "NCVis" in this setting [4] (Sec. 4.4), resulting in a very efficient implementation. Our analysis gives rise to a spectrum of 'contrastive' neighbor embedding methods (Fig. 4.1) akin to that in [15], interpolating between UMAP and NCVis/t-SNE (Sec. 4.5). We demonstrate that UMAP can be seen as negative sampling applied to the t-SNE problem (Sec. 4.6) and explain why UMAP embeddings differ from t-SNE ones. Finally, we develop a unified PyTorch framework for contrastive parametric and non-parametric neighbor embedding methods,[1] including a new method based on the InfoNCE loss [124], popular in self-supervised learning [7, 22, 27, 64, 91, 124, 133, 157, 172], and show the relevance of all these losses in the 'contrastive learning' [63] setting of SimCLR [27] (Sec. 4.8).

4.2 RELATED WORK

One of the most popular methods for data visualization is t-SNE [101, 160, 161]. Recently developed NCVis [4] employs noise-contrastive estimation [61, 62] to approximate t-SNE in a sampling-based way. UMAP [112], motivated by sophisticated mathematical theory, has matched t-SNE's popularity at least in computational biology [9], and also uses a sampling-based optimization, namely negative sampling [113]. UMAP sparked the development of related methods, e.g., TriMAP [1] optimizes its embedding by sampling triplets, while PacMAP [165] recommends using mid-range pairs of points.

1 Our code is available at https://github.com/hci-unihd/cl-tsne-umap.

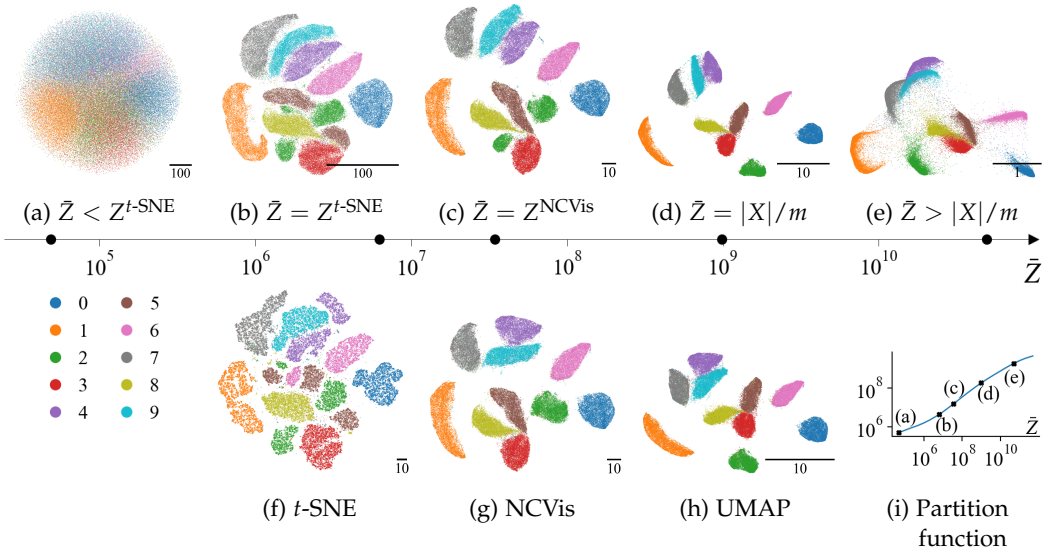


Figure 4.2: **(a – e)** Neg-t-SNE embeddings of the MNIST dataset for various values of the fixed normalization constant $\bar{Z}$. As $\bar{Z}$ increases, the scale of the embedding decreases, and clusters become more compact and separated before eventually starting to merge. The Neg-t-SNE spectrum produces embeddings very similar to those of **(f)** t-SNE, **(g)** NCVis, and **(h)** UMAP, when $\bar{Z}$ equals the partition function of t-SNE, the learned normalization parameter Z of NCVis, or $|X|/m = \binom{n}{2}/m$ used by UMAP, as predicted in Sec. 4.4–4.6. **(i)** The partition function $\sum_{ij}(1+d_{ij}^2)^{-1}$ tries to match $\bar{Z}$ and grows with it. In contrast to Fig. 4.1, we do not use early exaggeration here, but initialize the Neg-t-SNE and t-SNE with PCA rescaled so that the first dimension has standard deviation 1 and 0.0001, respectively. This makes the embeddings with small $\bar{Z}$ values show cluster fragmentation, similar to the t-SNE embedding in **(f)** without early exaggeration. For very low $\bar{Z}$, the Neg-t-SNE embedding in **(a)** shows very little structure.

Given their success, t-SNE and UMAP have been scrutinized to identify which aspects are essential to their performance. On the one hand, prior work [84, 102] found initialization to be important for both methods and strongly influencing the resulting global structure. On the other hand, the exact choice of the low-dimensional similarity kernel [15] or weights of the k-nearest-neighbor graph [39] are largely inconsequential. Both algorithms have similar relevant hyper-parameters, e.g., the heavy-tailedness of the similarity kernel [85, 174].

The main difference between t-SNE and UMAP is in their loss functions, which have been studied in [15, 39, 165], but never conceptually connected. We achieve this by deepening the link between negative sampling (NEG) and noise-contrastive estimation (NCE).

NEG was introduced as an ad hoc replacement for NCE in the context of learning word embeddings [113]. The relationship between NEG and NCE has been discussed before [48, 91, 97, 108, 140], but here we go further and provide the precise meaning of NEG: We show that, unlike NCE, NEG learns a model proportional but not equal to the actual data distribution. This allows us to explain the qualitative difference between NEG and NCE, which we demonstrate using neighbor embeddings.

Both t-SNE and UMAP have parametric versions [141, 159] with very different implementations. Here, we present a unified PyTorch framework for non-parametric and parametric contrastive neighbor embedding approaches. It encompasses UMAP, NEG, as well as t-SNE approximations both with NCE (like NCVis) and the InfoNCE loss [72, 124, 150], which has not been applied to neighbor embeddings. Moreover, we show that all of the mentioned loss functions (NCE/InfoNCE/NEG) can work similarly well in the SimCLR setting [27].

4.3 BACKGROUND

4.3.1 *Noise-contrastive estimation (NCE)*

The goal of parametric density estimation is to fit a parametric model q_θ to iid samples $s_1, \dots, s_N$ from an unknown data distribution p over a space X. For maximum likelihood estimation (MLE), the parameters θ are chosen to maximize the log-likelihood of the observed samples

$$\theta^* = \operatorname*{argmax}_\theta \sum_{i=1}^N \log\left(q_\theta(s_i)\right). \tag{4.1}$$

This approach crucially requires q_θ to be a normalized model. It is otherwise trivial to increase the likelihood arbitrarily by scaling q_θ.

Gutmann and Hyvärinen [61, 62] introduced NCE to circumvent the expensive computation of the partition function $Z(\theta) = \sum_{x\in X} q_\theta(x)$. NCE turns the unsupervised problem of parametric density estimation into a supervised problem in which the data samples need to be identified in a set S containing N data samples $s_1, \dots, s_N$ and m times as many noise samples $t_1, \dots, t_{mN}$. The noise samples are drawn from a noise distribution ξ, which can be (but does not have to be) the uniform distribution. In other words, we are interested in the posterior probability $\mathbb{P}(y|x)$ of element $x \in S$ coming from the data ($y = \text{data}$) rather than from the noise distribution ($y = \text{noise}$).

The probability of sampling x from noise, $\mathbb{P}(x|\text{noise})$, is just the noise distribution ξ, and similarly $\mathbb{P}(x|\text{data})$ is the data distribution p. As the latter is unknown, it is replaced by the model q_θ. Since S contains m times as many noise samples as data samples, the prior class probabilities are $\mathbb{P}(\text{data}) = 1/(m+1)$ and $\mathbb{P}(\text{noise}) = m/(m+1)$. Thus, the unconditional probability of an element of S is $\mathbb{P}(x) = (q_\theta(x) + m\xi(x))/(m+1)$. The posterior probability for classifying some given element x of S as data rather than noise is thus

$$\mathbb{P}(\text{data}|x) = \frac{\mathbb{P}(x|\text{data})\mathbb{P}(\text{data})}{\mathbb{P}(x)} = \frac{q_\theta(x)}{q_\theta(x) + m\xi(x)}. \tag{4.2}$$

NCE optimizes the parameters θ by maximizing the log-likelihood of the posterior class distributions or, equivalently, by minimizing the negative log-likelihoods. This is the same as a sum over binary cross-entropy losses:

$$\theta^* = \operatorname*{argmin}_\theta \left[-\sum_{i=1}^N \log\left(\frac{q_\theta(s_i)}{q_\theta(s_i) + m\xi(s_i)}\right) - \sum_{i=1}^{mN} \log\left(1 - \frac{q_\theta(t_i)}{q_\theta(t_i) + m\xi(t_i)}\right)\right]. \tag{4.3}$$

In expectation, we have the loss function

$$\mathcal{L}^{\text{NCE}}(\theta) = -\mathbb{E}_{s\sim p} \log\left(\frac{q_\theta(s)}{q_\theta(s) + m\xi(s)}\right) - m\mathbb{E}_{t\sim\xi} \log\left(1 - \frac{q_\theta(t)}{q_\theta(t) + m\xi(t)}\right). \tag{4.4}$$

Since

$$\frac{q_\theta(x)}{q_\theta(x)+m\xi(x)} = \frac{1}{1+\left(\frac{q_\theta(x)}{m\xi(x)}\right)^{-1}}, \tag{4.5}$$

NCE's loss function can also be seen as binary logistic regression loss function with $\log\left(\frac{q_\theta(x)}{m\xi(x)}\right)$ as the input to the logistic function:

$$\mathcal{L}^{\mathrm{NCE}}(\theta) = -\mathbb{E}_{s\sim p}\log\left(\sigma\left(\log\left(\frac{q_\theta(s)}{m\xi(s)}\right)\right)\right) - m\mathbb{E}_{t\sim\xi}\log\left(1-\sigma\left(\log\left(\frac{q_\theta(t)}{m\xi(t)}\right)\right)\right), \tag{4.6}$$

where $\sigma(x) = 1/\left(1+\exp(-x)\right)$ is the logistic function.

The key advantage of NCE is that the model does not need to be explicitly normalized by the partition function but nevertheless learns to equal the data distribution p and hence be normalized:

Theorem 4.1 ([61, 62]). *Suppose there exists some θ^* such that $q_{\theta^*} = p$. Then θ^* is a minimum of NCE's expected loss function $\mathcal{L}^{\mathrm{NCE}}(\theta)$ (4.4) and any other minimum $\tilde{\theta}$ also satisfies $q_{\tilde{\theta}} = p$. In addition, if $\xi(x)$ is non-zero wherever $p(x)$ is non-zero, then these are the only extrema of $\mathcal{L}^{\mathrm{NCE}}(\theta)$.*

In NCE, the model typically includes an optimizable normalization parameter Z which we emphasize by writing $q_{\theta,Z} = q_\theta/Z$. But importantly, Thm. 4.1 applies to any model q_θ that is able to match the data distribution p, even if it does not contain a learnable normalization parameter.

In the setting of learning language models, Jozefowicz *et al.* [72] proposed a different version of NCE, called InfoNCE. Instead of classifying each sample independently as noise or data as above, the aim here is to predict the position of a data sample in an $(m+1)$-tuple $T = (x_0, \ldots, x_m)$ containing m noise samples and one data sample. In other words, InfoNCE replaces the binary classification of NCE with an $(m+1)$-class classification.

Let Y be the random variable that holds the index of the data sample. A priori, we have $\mathbb{P}(Y=k) = 1/(m+1)$ for all $k = 0, \ldots, m$. Moreover, conditioned on sample k coming from the data distribution, all other samples must come from the noise distribution, i.e., we have $\mathbb{P}(x_i|Y=k) = \xi(x_i)$ for $i \neq k$. As the data distribution is unknown, we model it with $\mathbb{P}(x_k|Y=k) = q_\theta(x_k)$ as above. This yields the likelihood of tuple T given the data index $Y=k$

$$\mathbb{P}(T|Y=k) = q_\theta(x_k)\prod_{i\neq k}\xi(x_i) = \frac{q_\theta(x_k)}{\xi(x_k)}\prod_{i=0}^{m}\xi(x_i). \tag{4.7}$$

Marginalizing over Y, we obtain

$$\mathbb{P}(T) = \frac{1}{m+1}\prod_{i=0}^{m}\xi(x_i)\sum_{k=0}^{m}\frac{q_\theta(x_k)}{\xi(x_k)}. \tag{4.8}$$

Finally, we can compute the posterior via Bayes' rule as

$$\mathbb{P}(Y=k|T) = \frac{\mathbb{P}(T|Y=k)\mathbb{P}(Y=k)}{\mathbb{P}(T)} = \frac{\frac{q_\theta(x_k)}{\xi(x_k)}}{\sum_{i=0}^{m}\frac{q_\theta(x_i)}{\xi(x_i)}} = \frac{q_\theta(x_k)}{\sum_{i=0}^{m}q_\theta(x_i)}, \tag{4.9}$$

where the last equality holds for the uniform noise distribution. The InfoNCE loss is the cross-entropy loss with respect to the true position of the data sample, i.e., in expectation and for uniform ξ, it reads:

$$\mathcal{L}^{\text{InfoNCE}}(\theta) = -\mathop{\mathbb{E}}_{\substack{x\sim p\\ x_1,\dots,x_m\sim\xi}} \log\left(\frac{q_\theta(x)}{q_\theta(x)+\sum_{i=1}^m q_\theta(x_i)}\right). \tag{4.10}$$

Similar to how the NCE loss can be seen as the binary logistic regression loss function, the InfoNCE loss can be viewed as the multinomial logistic regression loss function with the terms $\log\left(\frac{q_\theta(x_i)}{\xi(x_i)}\right)$ entering the softmax function.

Ma and Collins [108] showed that an analogue of Thm. 4.1 applies to InfoNCE.

4.3.2 *Neighbor embeddings*

Neighbor embeddings (NE) [66] are a group of dimensionality reduction methods, including UMAP [112], NCVis [4], and t-SNE [161], that aim to find a low-dimensional embedding $e_1,\dots,e_n\in\mathbb{R}^d$ of high-dimensional input points $x_1,\dots,x_n\in\mathbb{R}^D$, with $D\gg d$ and usually $d=2$ for the purpose of visualization. NE methods define a notion of similarity over pairs of input data points which encodes the neighborhood structure and informs the low-dimensional embedding.

The exact high-dimensional similarity distribution differs between the NE algorithms, but recent work found empirically [15] and theoretically [39] that t-SNE and UMAP results stay practically the same when using the binary symmetric k-nearest-neighbor graph (skNN) instead of t-SNE's Gaussian or UMAP's Laplacian similarities. An edge ij is in skNN if x_i is among the k nearest neighbors of x_j or vice versa. The high-dimensional similarity function is then given by $p(ij)=\mathbb{1}(ij\in \text{s}k\text{NN})/|\text{s}k\text{NN}|$, where $|\text{s}k\text{NN}|$ denotes the number of edges in the skNN graph and $\mathbb{1}$ is the indicator function. NCVis uses the same similarities.

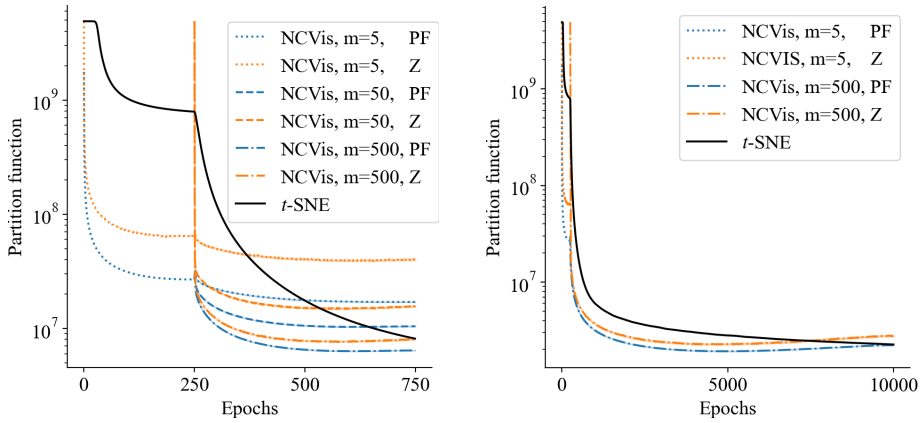
There are other differences in the choice of low-dimensional similarity between t-SNE and UMAP, but [15] shows that they are negligible. Therefore, here we use the Cauchy kernel $\phi(d_{ij})=1/(d_{ij}^2+1)$ for all NE methods to transform distances $d_{ij}=\|e_i-e_j\|$ in the embedding space into low-dimensional similarities. We abuse the notation slightly by also writing $\phi(ij)=\phi(d_{ij})$.

All NE methods discussed in this work can be cast in the framework of parametric density estimation. Here, p is the data distribution to be approximated with a model q_θ, meaning that the space X on which both p and q_θ live is the set of all pairs ij with $1\le i<j\le n$. The embedding positions $e_1,\dots,e_n$ become the learnable parameters θ of the model q_θ. For t-SNE, NCVis, and UMAP, q_θ is proportional to $\phi(\|e_i-e_j\|)$, but the proportionality and the loss functions are different.

t-SNE uses MLE and therefore requires a normalized model $q_\theta(ij)=\phi(ij)/Z(\theta)$, where $Z(\theta)=\sum_{k\ne l}\phi(kl)$ is the partition function. The loss function

$$\mathcal{L}^{t\text{-SNE}}(\theta) = -\mathbb{E}_{ij\sim p}\log\big(q_\theta(ij)\big) = -\sum_{i\ne j}\Big(p(ij)\log\big(\phi(ij)\big)\Big) + \log\left(\sum_{k\ne l}\phi(kl)\right) \tag{4.11}$$

is the expected negative log-likelihood of the embedding positions θ, making t-SNE an instance of MLE. Usually, t-SNE's loss function is introduced as the Kullback-



(a) (b)

Figure 4.3: NCVis learns to have the same partition function (PF) as t-SNE on the MNIST dataset. The higher the number m of noise samples or the longer the optimization, the better the match. Both methods use early exaggeration, which for NCVis means to start with $m = 5$ noise samples for the first 250 epochs. The learned normalization parameter Z converges to but does not precisely equal NCVis' partition function $\sum_{ij} q_\theta(ij)$. Nevertheless, it is of the same order of magnitude. Again, the match is better for more noise samples. Since we reinitialize the learnable Z for NCVis after the early exaggeration phase, there are short jumps in the partition function and Z at the beginning of the non-exaggerated phase.

Leibler divergence between p and q_θ, which is equivalent as the entropy of p does not depend on θ.

NCVis uses NCE and optimizes the expected loss function

$$\mathcal{L}^{\text{NCVis}}(\theta, Z) = -\mathbb{E}_{ij\sim p} \log\left(\frac{q_{\theta,Z}(ij)}{q_{\theta,Z}(ij) + m\xi(ij)}\right) - m\mathbb{E}_{ij\sim\xi} \log\left(1 - \frac{q_{\theta,Z}(ij)}{q_{\theta,Z}(ij) + m\xi(ij)}\right), \tag{4.12}$$

where $q_{\theta,Z}(ij) = \phi(ij)/Z$ with learnable Z and ξ is approximately uniform, see Sec. 3.4.

According to Thm. 4.1, NCVis has the same optimum as t-SNE and can hence be seen as a sampling-based approximation of t-SNE. Indeed, we find that Z in NCVis and the partition function $Z(\theta)$ in t-SNE converge approximately to the same value, see Fig. 4.3.

UMAP's expected loss function is derived in Chapter 3. For better comparison with t-SNE and NCVis, we cast it in a slightly different form here, see Sec. 4.3.3:

$$\mathcal{L}^{\text{UMAP}}(\theta) = -\mathbb{E}_{ij\sim p} \log\left(q_\theta(ij)\right) - m\mathbb{E}_{ij\sim\xi} \log\left(1 - q_\theta(ij)\right), \tag{4.13}$$

with $q_\theta(ij) = \phi(ij)$ and ξ is approximately uniform, see Sec. 3.4. This is the effective loss function actually implemented in the UMAP algorithm, but note that it has only about m/n of the repulsion compared to the loss stated in the original UMAP paper [112], as shown in Chapter 3.

In practice, the expectations in UMAP's and NCVis' loss functions are evaluated via sampling, like in Eq. (4.3). This leads to a fast, $\mathcal{O}(n)$, stochastic gradient descent optimization scheme. Both loss functions are composed of an attractive term pulling similar data points (edges of the skNN graph) closer together and a repulsive term pushing random pairs of points further apart. Similarly, t-SNE's loss yields

attraction along the graph edges while repulsion arises through the normalization term.

4.3.3 *UMAP's loss function*

In the original UMAP paper, McInnes *et al.* [112] define weights $\mu(ij) \in [0,1]$ on the skNN graph and state that these shall be reproduced by the low-dimensional similarities $\phi(ij)$ using a sum of binary cross-entropy loss functions, one for each edge ij

$$-\sum_{ij}\Big[\mu(ij)\log\big(\phi(ij)\big)+\big(1-\mu(ij)\big)\log\big(1-\phi(ij)\big)\Big]. \tag{4.14}$$

Indeed, this loss has its minimum at $\mu(ij)=\phi(ij)$ for all ij. However, it is, of course, impossible to achieve zero loss for any real-world data using the Cauchy kernel in two dimensions. Experiments show that this loss function in practice leads to excess repulsion and consequently to very poor embeddings [15]. The actual UMAP implementation has much less repulsion due to the sampling of repulsive edges, see below.

As the weights $\mu(ij)$ are only supported on the sparse skNN graph, most of the $1-\mu(ij)$ terms are equal to one. To simplify the loss function, the UMAP paper replaces all $1-\mu(ij)$ terms by 1, leading to the loss function

$$-\sum_{ij}\Big[\mu(ij)\log\big(\phi(ij)\big)+\log\big(1-\phi(ij)\big)\Big]. \tag{4.15}$$

In the implementation, UMAP samples the repulsive edges, which drastically changes the loss [15] to the effective loss derived in Chapter 3

$$-\sum_{ij}\Big[\mu(ij)\log\big(\phi(ij)\big)+\frac{m(d_i+d_j)}{2n}\log\big(1-\phi(ij)\big)\Big], \tag{4.16}$$

where $d_i=\sum_j\mu(ij)$ denotes the degree of node i and the number of negative samples m is a hyperparameter. By default, $m=5$. Since $d_i\approx\log(k)$, the effective loss only has about $m\log(k)/n$ of the repulsion in the originally stated loss function. As a result, this loss function does not reproduce the $\mu(ij)$ in the embedding space.

We rewrite this effective loss function further to fit into the framework of this chapter. The attractive prefactors $\mu(ij)$ sum to $\sum_{ij}\mu(ij)$, while the repulsive prefactors add up to m times this factor. Dividing the entire loss function by this term does not change its properties. But then, we can write the prefactors as probability distributions $p(ij)=\mu(ij)/\sum_{ij}\mu(ij)$ and $\xi(ij)=\big(p(i)+p(j)\big)/(2n)$ using $p(i)=\sum_j p(ij)$. With this, we can write the effective loss function as

$$-\sum_{ij}p(ij)\log\big(\phi(ij)\big)-m\sum_{ij}\xi(ij)\log\big(1-\phi(ij)\big), \tag{4.17}$$

or in the expectation form as

$$-\mathbb{E}_{ij\sim p}\log\big(\phi(ij)\big)-m\mathbb{E}_{ij\sim\xi}\log\big(1-\phi(ij)\big), \tag{4.18}$$

like we do in Eq. (4.13).

4.3.4 *Noise distributions*

Here, we discuss the various noise distributions used by UMAP, NCVis, and our framework. The central claim is that all these noise distributions are sufficiently close to uniform, even though their exact shape depends on the implementation details.

Since our actual implementation and the reference implementations of UMAP and NCVis consider edges ij and ji separately, we will do so from now on. Hence, there is now a total of $E := 2|s\!k\text{NN}|$ edges. We always assume that $p(ij) = p(ji)$ and adding up the probabilities for both directions yields one: $\sum_{i,j=1}^{n} p(ij) = 1$. For a given data distribution over pairs of points ij, we define $p(i) = \sum_{j=1}^{n} p(ij)$ so that $\sum_i p(i) = 1$. As discussed in Sec. 3.4, the $p(i)$ values are approximately constant when $p(ij)$ is uniform on the skNN graph or proportional to the UMAP similarities.

UMAP's noise distribution is derived in Sec. 3.6 and reads in the notation of this chapter

$$\xi(ij) = \frac{p(i) + p(j)}{2n}\,. \tag{4.19}$$

Note that UMAP uses a weighted version of the skNN graph. Still, ξ is close to uniform, see Sec. 3.4.

The noise distribution of NCVis is also close to being uniform and equals [4]

$$\xi(ij) = \frac{p(i)}{n}\,. \tag{4.20}$$

This is a slightly different noise distribution than in UMAP, and in particular, it is asymmetric. However, we argue that in practice, it is equivalent UMAP's noise distribution. The noise distribution is used in two ways in NCVis: for sampling noise samples and in the posterior class probabilities

$$\mathbb{P}(\text{data}|ij) = \frac{q_{\theta,Z}(ij)}{q_{\theta,Z}(ij) + m\xi(ij)}\,. \tag{4.21}$$

Both in the reference NCVis implementation and ours, for the second role, the noise distribution is explicitly approximated by the uniform one, and we use the posterior probabilities

$$\mathbb{P}(\text{data}|ij) = \frac{q_{\theta,Z}(ij)}{q_{\theta,Z}(ij) + m\frac{1}{2|s\!k\text{NN}|}}\,. \tag{4.22}$$

Together with the symmetry of Euclidean distance, this implies that the repulsion on the embedding vectors e_i and e_j from noise samples ij and ji is the same. As a result, the expectation

$$\mathbb{E}_{ij\sim\xi} \log\left(1 - \frac{q_{\theta,Z}(ij)}{q_{\theta,Z}(ij) + m\frac{m}{2|s\!k\text{NN}|}}\right) \tag{4.23}$$

is the same for $\xi(ij) = p(i)/n$ and for UMAP's noise distribution

$$\xi(ij) = \frac{p(i) + p(j)}{2n}\,. \tag{4.24}$$

Algorithm 4: Batched contrastive neighbor embedding algorithm

input : list of directed skNN graph edges $E = [i_1 j_1, \dots, i_{|E|} j_{|E|}]$
parameters θ
// θ = embeddings (non-param.)/ network weights (param.)
number of epochs T
learning rate η
number of noise samples m
Cauchy kernel q
// q acts on embeddings (non-param.)/ network output (param.)
batch size b
loss mode *mode*
normalization constant $\bar{Z}$
// default $\bar{Z} = |E|/m$, required only for *mode* = Neg-t-SNE

output: final embeddings $e_1, \dots, e_n$

if *mode* $=$ *NCVis* **then**
  $Z = 1$
for $t = 0$ **to** T **do**
  // Learning rate annealing
  $\eta_t = \eta \cdot (1 - \frac{t}{T})$
  $\alpha = 0$
  while $\alpha < |E|$ **do**
    $\mathcal{L} = 0$
    for $\beta = 1, \dots, b$ **do**
      // Sample noise edge tails but omit head of considered edge
      $j_1^-, \dots, j_m^- \sim \text{Uniform}(\{i_{\alpha+1}, j_{\alpha+1}, \dots, \widehat{i_{\alpha+\beta}}, \dots, j_{\alpha+b}\})$
      // Aggregate loss based on mode
      if *mode* $=$ *Neg-t-SNE* **then**
        $\mathcal{L} = \mathcal{L} - \log\left(\frac{q_\theta(i_{\alpha+\beta} j_{\alpha+\beta})}{q_\theta(i_{\alpha+\beta} j_{\alpha+\beta}) + \bar{Z} m/|E|}\right) - \sum_{\mu=1}^m \log\left(\frac{q_\theta(i_{\alpha+\beta} j_\mu^-)}{q_\theta(i_{\alpha+\beta} j_\mu^-) + \bar{Z} m/|E|}\right)$
      else if *mode* $=$ *NCVis* **then**
        $\mathcal{L} = \mathcal{L} - \log\left(\frac{q_\theta(i_{\alpha+\beta} j_{\alpha+\beta})/Z}{q_\theta(i_{\alpha+\beta} j_{\alpha+\beta})/Z + m}\right) - \sum_{\mu=1}^m \log\left(\frac{q_\theta(i_{\alpha+\beta} j_\mu^-)/Z}{q_\theta(i_{\alpha+\beta} j_\mu^-)/Z + m}\right)$
      else if *mode* $=$ *InfoNC-t-SNE* **then**
        $\mathcal{L} = \mathcal{L} - \log\left(\frac{q_\theta(i_{\alpha+\beta} j_{\alpha+\beta})}{q_\theta(i_{\alpha+\beta} j_{\alpha+\beta}) + \sum_{\mu=1}^m q_\theta(i_{\alpha+\beta} j_\mu^-)}\right)$
      else if *mode* $=$ *UMAP* **then**
        $\mathcal{L} = \mathcal{L} - \log\left(q_\theta(i_{\alpha+\beta} j_{\alpha+\beta})\right) - \sum_{\mu=1}^m \log\left(q_\theta(i_{\alpha+\beta} j_\mu^-)\right)$
    // Update parameters with SGD (non-param.) or Adam (param.)
    $\theta = \theta - \eta_t \cdot \nabla_\theta \mathcal{L}$
    if *mode* $=$ *NCVis* **then**
      $Z = Z - \eta_t \nabla_Z \mathcal{L}$
    $\alpha = \alpha + b$
  Shuffle E
return θ

Batched Neighbor Embeddings

In our framework, the noise distribution is influenced by the batched training procedure (Alg. 4) because the negative samples can come only from the current training batch.

Lemma 4.2. *The noise distribution induced by the batched neighbor embedding framework in Alg. 4 is given by*

$$\xi(ij) = \begin{cases} \frac{1}{2b-1}\left(\frac{E-b}{E-1}p(ij) + 2\left(b - \frac{E-b}{E-1}\right)p(i)p(j)\right) & \text{if } i \neq j \\ \frac{1}{2b-1}\left(-\frac{b-1}{E-1}p(i) + 2\left(b - \frac{E-b}{E-1}\right)p(i)^2\right) & \text{if } i = j, \end{cases} \tag{4.25}$$

where E is twice the number of edge in the skNN graph and b is the batch size

Proof. In every epoch, we first shuffle the set of directed edges of the skNN graph and then chunk it into batches. To emphasize, the batches consist of *directed edges* and not of the original data points. For each edge in a batch, we take its head and sample m indices from the heads and tails of all edges in the batch (excluding the already selected head) and use them as tails to form negative sample pairs.

To obtain a negative sample pair ij, the batch must contain some directed edge ik, providing the head of the negative sample, and some pair lj or jl, providing the tail. We want to derive the expected number of times a directed edge ij is considered as a negative sample in a batch. For simplicity, let us assume that the number of batches divides the number of directed edges E. As the set of skNN edges is shuffled every epoch, the expected number of pairs ij as negative samples is the same for all batches.

Let us consider a batch B of size b. We denote by Y_{rs} the random variable that holds the number of times edge rs appears in B. We also introduce random variables $Y_{\neg rs} = \sum_{t \neq r} Y_{ts}$ and $Y_{r\neg s} = \sum_{t \neq s} Y_{rt}$. Let $p(r\neg s) = p(\neg sr) := p(r) - p(rs)$. For each occurrence of an i as the head of an edge in B, we sample m tails to create negative samples uniformly from all heads and tails in B with replacement, but we prevent sampling the identical head i as negative sample tail. If, however, the same node i is part of the other edges in the batch, then it may be sampled and would create a futile negative sample ii. There are m chances for creating a negative sample edge ij for every head i and any occurrence of j in the batch. The number of heads i in the batch is $Y_{ij} + Y_{i\neg j}$ and the number of occurrences of j is $Y_{ij} + Y_{ji} + Y_{\neg ij} + Y_{j\neg i}$. Since we sample the tail of a negative sample pair uniformly with replacement, any occurrence of j has a probability of $1/(2b-1)$ to be selected. Hence, the expected number N_{ij} of times that the ordered pair ij with $i \neq j$ is considered as a negative sample in batch B is

$$N_{ij} = m(Y_{ij} + Y_{i\neg j})\frac{Y_{ij} + Y_{ji} + Y_{\neg ij} + Y_{j\neg i}}{2b-1}. \tag{4.26}$$

Since a head i may not choose itself to form a negative sample, the expected number of times that ii appears as a negative sample in the batch is

$$N_{ii} = m\sum_j Y_{ij}\frac{Y_{ij} - 1 + Y_{i\neg j} + Y_{ji} + Y_{\neg ji}}{2b-1}. \tag{4.27}$$

Since the batches are sampled without replacement, the random variables Y_{rs} are distributed according to a multivariate hypergeometric distribution, meaning that

$$\mathbb{E}(Y_{rs}) = bp(rs) \tag{4.28}$$
$$\mathbb{E}(Y_{\neg rs}) = bp(\neg rs) \tag{4.29}$$
$$\mathrm{Var}(Y_{rs}) = b\frac{E-b}{E-1}p(rs)(1-p(rs)) \tag{4.30}$$
$$\mathrm{Cov}(Y_{rs}, Y_{\neg uv}) = -b\frac{E-b}{E-1}p(rs)p(\neg uv). \tag{4.31}$$

We use these expressions and analogous ones together with the symmetries $p(rs) = p(sr)$ to compute (leaving out intermediate algebra steps) the expectation of N_{ij} over the shuffles:

$$\mathbb{E}(N_{ij}) = \frac{mb}{2b-1}\left(\frac{E-b}{E-1}p(ij) + 2\left(b - \frac{E-b}{E-1}\right)p(i)p(j)\right). \tag{4.32}$$

Since we sample m negative samples for each positive sample and since each batch contains b positive samples, we need to divide $\mathbb{E}(N_{ij})$ by mb to obtain $\xi(ij)$:

$$\xi(ij) = \frac{1}{2b-1}\left(\frac{E-b}{E-1}p(ij) + 2\left(b - \frac{E-b}{E-1}\right)p(i)p(j)\right). \tag{4.33}$$

Similarly,

$$\mathbb{E}(N_{ii}) = \frac{mb}{2b-1}\left(-\frac{b-1}{E-1}p(i) + 2\left(b - \frac{E-b}{E-1}\right)p(i)^2\right) \tag{4.34}$$

and hence the noise distribution value for the pair ii is

$$\xi(ii) = \frac{1}{2b-1}\left(-\frac{b-1}{E-1}p(i) + 2\left(b - \frac{E-b}{E-1}\right)p(i)^2\right). \tag{4.35}$$

□

We see that the noise distribution depends on the batch size b. This is not surprising: For example, if the batch size is equal to one, the ordered pair ij can only be sampled as a negative sample in the single batch that consists of that pair. Indeed, for $b = 1$, our formula yields

$$\xi(ij) = p(ij), \tag{4.36}$$

meaning that the data and the noise distributions coincide. Conversely, if $b = E$ and there is only one batch, we obtain

$$\xi(ij) = \frac{2E}{2E-1}p(i)p(j). \tag{4.37}$$

Hence, the noise distribution is close to uniform. The noise distribution is between these two extremes for batch sizes between 1 and E. For MNIST, $E \approx 1.5 \cdot 10^6$, and in our experiments we use $b = 1024$. This means that the prefactor of the share of the data distribution is about 0.0005 while that of the near-uniform distribution $p(i)p(j)$ is about 0.9995, so the resulting noise distribution is close to uniform. Note that Thm. 4.1 and Cor. 4.3 only require the noise distribution to be nonzero where the data distribution is nonzero, which is the case for any batch size.

4.4 FROM NOISE-CONTRASTIVE ESTIMATION TO NEGATIVE SAMPLING

In this section, we work out the precise relationship between NCE and NEG, going beyond prior work [48, 57, 97, 140]. NEG differs from NCE by its loss function and by the lack of the learnable normalization parameter Z. In our setting, NEG's loss function amounts to[2]

$$\mathcal{L}^{\text{NEG}}(\theta) = -\mathbb{E}_{x\sim p} \log\left(\frac{q_\theta(x)}{q_\theta(x)+1}\right) - m\mathbb{E}_{x\sim\xi} \log\left(1-\frac{q_\theta(x)}{q_\theta(x)+1}\right). \tag{4.38}$$

In order to relate it to NCE's loss function, we first generalize the latter in a way allowing it to learn a model that is not equal but proportional to the true data distribution.

Corollary 4.3. *Let $\bar{Z}, m \in \mathbb{R}_+$. Assume that $\xi(x)$ is non-zero wherever $p(x)$ is non-zero and that there exist θ^* such that $q_{\theta^*} = \bar{Z}p$. Then the generalized NCE loss function*

$$\mathcal{L}^{\text{NCE}}_{\bar{Z}}(\theta) = -\mathbb{E}_{x\sim p} \log\left(\frac{q_\theta(x)}{q_\theta(x)+\bar{Z}m\xi(x)}\right) - m\mathbb{E}_{x\sim\xi} \log\left(1-\frac{q_\theta(x)}{q_\theta(x)+\bar{Z}m\xi(x)}\right) \tag{4.39}$$

has its only extrema where $q_\theta = \bar{Z}p$.

Proof. The result follows from Thm. 4.1 applied to the model distribution $\tilde{q}_\theta := q_\theta/\bar{Z}$. □

It has been pointed out before [48, 140] that for a uniform noise distribution $\xi(x) = 1/|X|$ and as many noise samples as the size of X ($m = |X|$), the loss functions of NCE and NEG coincide, since $m\xi(x) = 1$. However, the main point of NCE and NEG is to use far fewer noise samples to attain a speed-up over MLE. Our Cor. 4.3 for the first time explains NEG's behavior in this more realistic setting ($m \ll |X|$). If the noise distribution is uniform, the generalized NCE loss function with $\bar{Z} = |X|/m$ equals the NEG loss function since $(|X|/m)m\xi(x) = 1$. By Cor. 4.3, any minimum θ^* of the NEG loss function yields $q_{\theta^*} = (|X|/m)p$, assuming that there are parameters that make this equation hold. In other words, NEG aims to find a model q_θ that is proportional to the data distribution with the proportionality factor $|X|/m$, which is typically huge. This is different from NCE, which aims to learn a model equal to the data distribution. Therefore, the optimal parameters for NEG and NCE are typically different.

Choosing $m \ll |X|$ not only offers a computational speed-up but is necessary when optimizing NEG for a neural network with SGD, as we do in Sec. 4.8. Only one single mini-batch is passed through the neural network during each iteration and is thus available for computing the loss. Hence, all noise samples must come from the current mini-batch, and their number m is upper-bounded by the mini-batch size b. Mini-batches are typically much smaller than n and, hence, $|X|$. Thus, this standard training procedure requires $m \ll |X|$ highlighting the relevance of Cor. 4.3.

While NCE uses a model q_θ/Z with learnable Z, we can interpret NEG as using a model $q_\theta/\bar{Z}$ with fixed and very large normalization constant $\bar{Z} = |X|/m$. As a

2 We focus on the loss function, ignoring design choices of [113] specific for learning word embeddings.

result, q_θ in NEG needs to attain much larger values to match the large $\bar{Z}$. This can be illustrated in the setting of neighbor embeddings. Applying NEG to the neighbor embedding framework yields an algorithm that we call "Neg-t-SNE". Recall that in this setting, the parameters $\theta = \{e_1, \dots e_n\}$ are the embedding positions and $|X| = \binom{n}{2}$ is the number of pairs of points. Böhm *et al.* [15] found empirically that t-SNE's partition function $Z(\theta)$ is typically between $50n$ and $100n$, while in Neg-t-SNE, $\bar{Z} = \mathcal{O}(n^2)$ is much larger for modern big datasets. To attain the larger values of $\phi(ij)$ required by NEG, points that are connected in the skNN graph have to move much closer together in the embedding than in t-SNE. Indeed, using our PyTorch implementation of Neg-t-SNE on the MNIST dataset, we confirm that Neg-t-SNE (Fig. 4.1d) produces more compact clusters than t-SNE (Fig. 4.1f). See Appendix D.2 and Alg. 4 for implementation details.

We emphasize that the conclusions of this section only hold because NEG does not contain a learnable normalization parameter Z. If it did, then such a learnable Z would be able to absorb the term $\bar{Z}$ in loss functions (4.38) and (4.39) while leaving the parameters θ unchanged.

4.5 NEGATIVE SAMPLING SPECTRUM

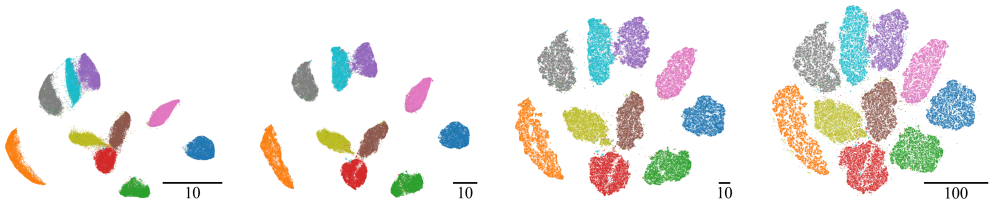
Varying the fixed normalization constant $\bar{Z}$ in Eq. (4.39) has important practical effects that lead to a whole spectrum of embeddings in the NE setting. The original NEG loss function (4.38) corresponds to Eq. (4.39) with $\bar{Z} = |X|/m$. We still refer to the more general case of using an arbitrary $\bar{Z}$ in Eq. (4.39) as "negative sampling", and "Neg-t-SNE" in the context of neighbor embeddings.

Figs. 4.1a–e show a spectrum of Neg-t-SNE visualizations of the MNIST dataset for varying $\bar{Z}$. Per Cor. 4.3, higher values of $\bar{Z}$ induce higher values for q_θ, meaning that points move closer together. Indeed, the embedding scale decreases for higher $\bar{Z}$. Moreover, clusters become increasingly compact and then even start to merge. For lower values of $\bar{Z}$, the embedding scale is larger, clusters are more spread out, eventually losing almost any separation and starting to overlap for very small $\bar{Z}$.

Cor. 4.3 implies that the partition function $\sum_x q_\theta(x)$ should grow with $\bar{Z}$, and indeed this is what we observe (Fig. 4.1i). The match between the sum of Cauchy kernels $\sum_{ij} \phi(ij)$ and $\bar{Z}$ is not perfect, but that is expected. Indeed, the Cauchy kernel is bounded by 1 from above, so values $\bar{Z} > \binom{n}{2}$ are not matchable. Similarly, very small values of $\bar{Z}$ are difficult to match because of the heavy tail of the Cauchy kernel.

By adjusting the $\bar{Z}$ value, one can obtain Neg-t-SNE embeddings very similar to NCVis and t-SNE. If the NCVis loss function (4.12) has its minimum at some θ^* and Z^{NCVis}, then the Neg-t-SNE loss function (4.39) with $\bar{Z} = Z^{\text{NCVis}}$ is minimal at the same θ^*. We confirm this experimentally: Setting $\bar{Z} = Z^{\text{NCVis}}$ (for MNIST, $Z^{\text{NCVis}} = 3.4 \cdot 10^7$, Fig. 4.3), yields a Neg-t-SNE embedding (Fig. 4.1c) closely resembling the NCVis embedding (Fig. 4.1g). Similarly, setting $\bar{Z}$ to the partition function $Z(\theta^{t\text{-SNE}})$, obtained by running t-SNE (for MNIST, $Z(\theta^{t\text{-SNE}}) = 8.1 \cdot 10^6$, Fig. 4.3), yields a Neg-t-SNE embedding closely resembling the t-SNE embedding (compare Figs. 4.1b and 4.1f).

The Neg-t-SNE spectrum is strongly related to the attraction-repulsion spectrum from Böhm *et al.* [15]. They introduced a prefactor ("exaggeration") to the repulsive



(a) $m = 5$ (b) $m = 50$ (c) $m = 500$ (d) $m = 2b - 2 = 2046$

Figure 4.4: Neg-t-SNE embeddings of the MNIST dataset for varying number of noise samples m and using batch size $b = 1024$. While for NCVis and InfoNC-t-SNE more noise samples improve the approximation to t-SNE, see Figs. 4.9 and 4.10, changing m in Neg-t-SNE moves the result along the attraction-repulsion spectrum (Fig. 4.1) with more repulsion for larger m. However, the computational complexity of Neg-t-SNE scales with m, so moving along the spectrum via changing $\bar{Z}$ is much more efficient. For the first 250 epochs, m is set to 5, to achieve an effect similar to early exaggeration (Appendix D.2).

term in the t-SNE's loss, which exaggerates the attraction over the repulsion, and obtained embeddings similar to our spectrum when varying this parameter. We can explain this as follows. The repulsive term in the NCE loss (4.4) has a prefactor m and our spectrum arises from the loss (4.39) by varying $\bar{Z}$ in the term $\bar{Z}m$. Equivalently, our spectrum can be obtained by varying the m value (number of noise samples per one skNN edge) while holding $\bar{Z}$ fixed (Fig. 4.4). In other words, our spectrum arises from varying the repulsion strength in the contrastive loss setting, while Böhm *et al.* [15] obtained the analogous spectrum by varying the repulsion strength in the t-SNE setting.

4.6 UMAP'S CONCEPTUAL RELATION TO t-SNE

Our comparison of NEG and NCE in Sec. 4.4 allows us for the first time to conceptually relate UMAP and t-SNE. The UMAP algorithm was introduced in [112] by elaborate theory that motivated a specific choice of weights on the skNN graph and the binary cross-entropy loss. Following LargeVis [153], McInnes *et al.* [112] implemented a sampling-based scheme to overcome the quadratic complexity of all repulsive forces in the purported binary cross-entropy loss.

Chapter 3 and [15] pointed to the drastic effect of this optimization scheme and argued that it alters the effective loss. The UMAP paper [112] referred to their optimization as 'negative sampling', however UMAP's loss function (4.13) does not look like the NEG loss (4.38), and hence it has been unclear if UMAP actually uses NEG in the sense of [113].

Where NEG's loss function (4.38) has fractions $q_\theta(ij)/(q_\theta(ij) + 1)$, UMAP's loss (4.13) only has $q_\theta(ij)$. Nevertheless, we can rearrange the terms to make UMAP appear as a proper instance of NEG:

Lemma 4.4. *UMAP's loss function* (4.13) *is NEG* (4.38) *with the parametric model* $\tilde{q}_\theta(ij) = 1/d_{ij}^2$.

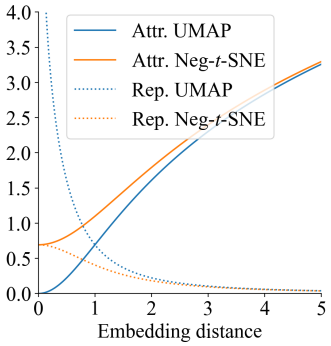


Figure 4.5: Attractive and repulsive loss terms of UMAP and Neg-t-SNE. The main difference is that UMAP's repulsive loss diverges at zero challenging its numerical optimization. The attractive terms are $\log(1+d_{ij}^2)$ and $\log(2+d_{ij}^2)$ for UMAP and Neg-t-SNE, respectively, and the repulsive ones are $\log\left((1+d_{ij}^2)/d_{ij}^2\right)$ and $\log\left((2+d_{ij}^2)/(1+d_{ij}^2)\right)$, respectively.

Proof. The statement follows from the simple computation

$$q_\theta(ij) = \phi(ij) = \frac{1}{1+d_{ij}^2} = \frac{1/d_{ij}^2}{1/d_{ij}^2+1} = \frac{\tilde{q}_\theta(ij)}{\tilde{q}_\theta(ij)+1}. \tag{4.40}$$

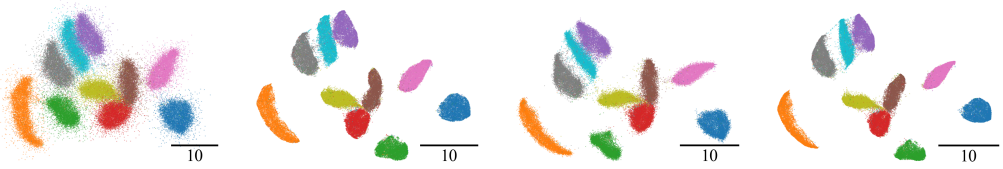
□

Lem. 4.4 tells us that UMAP uses NEG but not with a parametric model given by the Cauchy kernel. Instead it uses a parametric model $\tilde{q}_\theta$, which equals the squared inverse distance between embedding points.

For large embedding distances, d_{ij} both models behave similarly, but for nearby points, they strongly differ: The inverse-square kernel $1/d_{ij}^2$ diverges when $d_{ij} \to 0$, whereas the Cauchy kernel $1/(1+d_{ij}^2)$ does not. Despite this qualitative difference, we find empirically that UMAP embeddings look very similar to Neg-t-SNE embeddings at $\bar{Z} = |X|/m$, see Figs. 4.1d and 4.1h for the MNIST example.

To explain this observation, it is instructive to compare the loss terms of Neg-t-SNE and UMAP: The attractive term amounts to $-\log(1/(d_{ij}^2+1)) = \log(1+d_{ij}^2)$ for UMAP and $-\log[1/(d_{ij}^2+1)/(1/(d_{ij}^2+1)+1)] = \log(2+d_{ij}^2)$ for Neg-t-SNE, while the repulsive term equals $\log((1+d_{ij}^2)/d_{ij}^2)$ and $\log((2+d_{ij}^2)/(1+d_{ij}^2))$, respectively. While the attractive terms are very similar, the repulsive term for UMAP diverges at zero but that of Neg-t-SNE does not (Fig. 4.5). This divergence introduces numerical instability into the optimization process of UMAP. In fact, UMAP employs several optimization tricks in order to overcome these instabilities. One of the tricks is annealing the learning rate down to zero so that the learning rate becomes very small in the last optimization epochs.

We find that UMAP strongly depends on this annealing (Figs. 4.6a, b). Without it, clusters appear fuzzy as noise pairs can experience powerful repulsion and get catapulted out of their cluster (Fig. 4.6a). While Neg-t-SNE also benefits from the annealing scheme (Fig. 4.6d), it produces a very similar embedding even without any annealing (Fig. 4.6c). Thus, UMAP's effective choice of the $1/d_{ij}^2$ kernel makes it less numerically stable and more dependent on learning rate annealing than



(a) UMAP no annealing (b) UMAP with annealing (c) Neg-t-SNE no annealing (d) Neg-t-SNE with annealing

Figure 4.6: Embeddings of the MNIST dataset with UMAP and Neg-t-SNE with and without learning rate annealing. UMAP does not work well without annealing because it implicitly uses the diverging $1/d_{ij}^2$ kernel in NEG, while Neg-t-SNE uses the more numerically stable Cauchy kernel.

Neg-t-SNE.[3] The following Sec. 4.7 shows that UMAP's other optimization tricks have little effect.

We conclude that at its heart, UMAP is NEG applied to the t-SNE framework. UMAP's sampling-based optimization is much more than a mere optimization trick; it enables us to connect it theoretically to t-SNE. When UMAP's loss function is seen as an instance of NEG, UMAP does not use the Cauchy kernel but rather the inverse-square kernel. However, this does not make a big difference due to the learning rate decay. As discussed in Sec. 4.4, the fixed normalization constant $\bar{Z}$ in Neg-t-SNE/UMAP is much larger than the learnt Z in NCVis or the partition function in t-SNE. This explains why UMAP pulls embedding points closer together than both NCVis and t-SNE and is the reason for the typically more compact clusters in UMAP embeddings [15].

4.7 FURTHER OPTIMIZATION TRICKS IN UMAP'S ORIGINAL IMPLEMENTATION

UMAP's repulsive term

$$-\log(1-\phi(ij)) = \log\left(\frac{1+d_{ij}^2}{d_{ij}^2}\right) \tag{4.41}$$

can lead to numerical problems if the two points of the negative sample pair are very close. In addition to the learning rate decay, discussed in Sec. 4.6, UMAP's implementation uses additional tricks to prevent unstable or even crashing training.

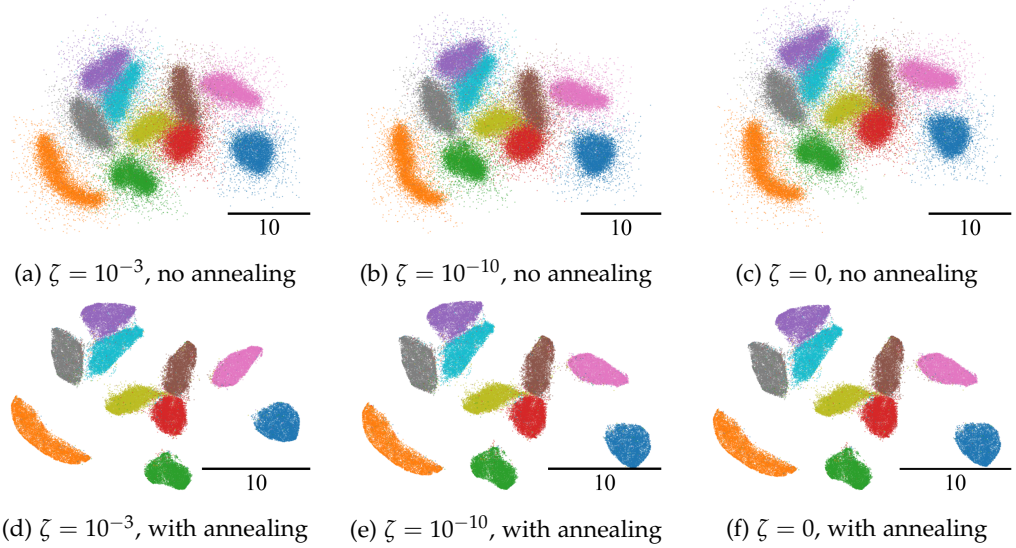
In non-parametric UMAP's reference implementation, the gradient on embedding position e_i exerted by a single sampled repulsive pair ij is actually

$$2\frac{1}{d_{ij}^2+\zeta}\frac{1}{1+d_{ij}^2}(e_j-e_i) \tag{4.42}$$

with $\zeta = 0.001$ instead of $\zeta = 0$. This corresponds to the full loss function

$$-\mathbb{E}_{ij\sim p}\log(\phi(ij)) - m\left(1+\frac{\zeta}{1-\zeta}\right)\mathbb{E}_{ij\sim\xi}\log\left(1+\frac{\zeta}{1-\zeta}-\phi(ij)\right). \tag{4.43}$$

3 Recently, a pull request to UMAP's GitHub repository changed the effective kernel of parametric UMAP to the Cauchy kernel, in order to overcome numerical instabilities via an ad hoc fix, see Sec. 4.7.



(a) $\zeta = 10^{-3}$, no annealing (b) $\zeta = 10^{-10}$, no annealing (c) $\zeta = 0$, no annealing

(d) $\zeta = 10^{-3}$, with annealing (e) $\zeta = 10^{-10}$, with annealing (f) $\zeta = 0$, with annealing

Figure 4.7: UMAP embeddings of the MNIST dataset, ablating numerical optimization tricks of UMAP's reference implementation. The learning rate annealing is crucial (bottom row) but safeguarding against divisions by zero in UMAP's repulsive term (4.42) by adding ζ to the denominator has little effect. These experiments are run using the reference implementation, modified to change the ζ value and/or to switch off the learning rate annealing.

However, we find that ζ does not influence the appearance of a UMAP embedding much. Fig. 4.7 shows MNIST embeddings obtained using the original UMAP implementation modified to use different values of ζ. Neither a much smaller positive value such as $\zeta = 10^{-10}$ nor setting $\zeta = 0$ substantially change the embedding (even though some runs with $\zeta = 0$ do crash). The learning rate annealing plays a much bigger role in how the embedding looks (Fig. 4.7, bottom row).

The reference implementation of parametric UMAP uses automatic differentiation instead of implementing the gradients manually. To avoid terms such as $\log(0)$ in the repulsive loss, it clips the argument of the logarithm from below at the value $\varepsilon = 10^{-4}$, effectively using the loss function

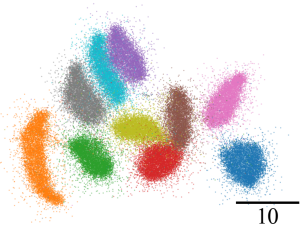
$$-\mathbb{E}_{ij\sim p}\log\left(\max\left\{\varepsilon, \frac{1}{1+d_{ij}^2}\right\}\right) - m\mathbb{E}_{ij\sim\xi}\log\left(\max\left\{\varepsilon, 1-\frac{1}{1+d_{ij}^2}\right\}\right). \tag{4.44}$$

We employ a similar clipping in our code whenever we apply the logarithm function. Again, we find that the exact value of ε is not essential for our UMAP reimplementation, while using the learning rate annealing is (Fig. 4.8, top two rows). In the extreme case of setting $\varepsilon = 0$, our UMAP runs crashed. We believe that the reason is that we allow negative sample pairs to be of the form ii, which do not send any gradient but lead to a zero argument to the logarithm. The reference implementation of UMAP excludes such negative sample pairs ii.

Our Neg-t-SNE approach does not have any of these problems, as the repulsive term is

$$-\log\left(1-\frac{\frac{1}{1+d_{ij}^2}}{\frac{1}{1+d_{ij}^2}+1}\right) = \log\left(\frac{2+d_{ij}^2}{1+d_{ij}^2}\right) \le \log(2) \tag{4.45}$$

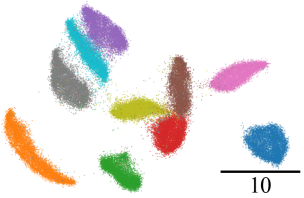
(a) UMAP, $\varepsilon = 10^{-4}$, no ann.



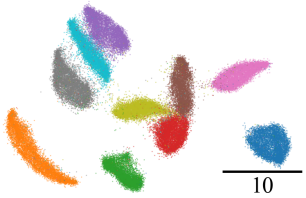
(b) UMAP, $\varepsilon = 10^{-10}$, no ann.

(c) UMAP, $\varepsilon = 10^{-4}$, with ann.

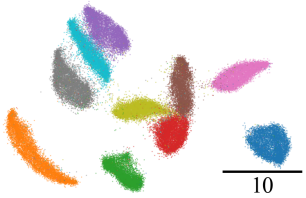
(d) UMAP, $\varepsilon = 10^{-10}$, with ann.



(e) Neg-t-SNE, $\varepsilon = 10^{-4}$, no ann.



(f) Neg-t-SNE, $\varepsilon = 10^{-10}$, no ann.



(g) Neg-t-SNE, $\varepsilon = 0$, no ann.

(h) Neg-t-SNE, $\varepsilon = 10^{-4}$, with ann.

(i) Neg-t-SNE, $\varepsilon = 10^{-10}$, with ann.

(j) Neg-t-SNE, $\varepsilon = 0$, with ann.

Figure 4.8: UMAP and Neg-t-SNE embeddings of the MNIST dataset using different values ε at which we clip arguments to logarithm functions. These experiments are performed using our implementation. Varying ε does not strongly influence the appearance of the embedding. But setting $\varepsilon = 0$ leads to crashing UMAP runs. Annealing the learning rate is important for UMAP, yet not for Neg-t-SNE.

and does not diverge for $d_{ij} \to 0$. For this reason, Neg-t-SNE is not very sensitive to the value at which we clip arguments to the logarithm and works even with $\varepsilon = 0$, both with and without learning rate annealing (Fig. 4.8, bottom two rows).

The attractive terms in the loss functions do not pose numerical problems in practice due to the heavy tail of the Cauchy kernel. To keep different experiments and losses comparable, we use clipping in both the attractive and the repulsive loss terms with $\varepsilon = 10^{-10}$ for all neighbor embedding plots computed with our framework unless otherwise stated.

A recent pull request[4] to the parametric part of UMAP's reference implementation proposed another way to ameliorate the numerical instabilities. The clipping of the logarithm's arguments was replaced with a sigmoid of the logarithm of the Cauchy kernel so that the attractive and repulsive terms become

$$-\log\Big(\max\big(\varepsilon, \phi(ij)\big)\Big) \to -\log\Big(\sigma\Big(\log\big(\phi(ij)\big)\Big)\Big) \tag{4.46}$$

$$-\log\Big(\max\big(\varepsilon, 1-\phi(ij)\big)\Big) \to -\log\Big(1-\sigma\Big(\log\big(\phi(ij)\big)\Big)\Big), \tag{4.47}$$

where $\sigma(x) = 1/\big(1+\exp(-x)\big)$ is the sigmoid function. This change can seem drastic as, e.g., for $\phi(ij) = 1$ we have $\max\big\{\varepsilon, \phi(ij)\big\} = 1$, but $\sigma\Big(\log\big(\phi(ij)\big)\Big) = 1/2$. But unraveling the definitions shows that this turns the loss function precisely into our Neg-t-SNE loss function since

$$\sigma\Big(\log\big(\phi(ij)\big)\Big) = \frac{1}{1+\exp\Big(-\log\big(\phi(ij)\big)\Big)} = \frac{1}{1+\phi(ij)^{-1}} = \frac{\phi(ij)}{\phi(ij)+1}. \tag{4.48}$$

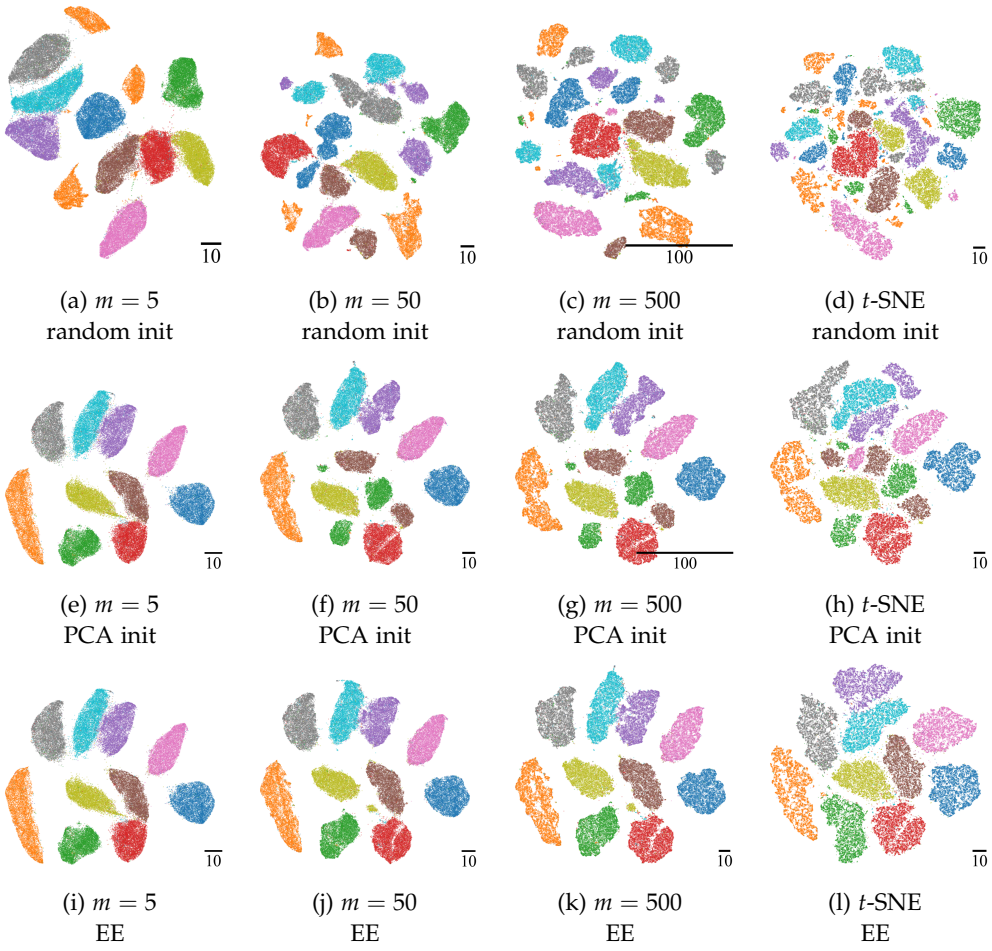
So, to overcome the numerical problems incurred by UMAP's implicit choice of $1/d_{ij}^2$ as similarity kernel, the pull request suggested a fix that turns out to be equivalent to negative sampling using the Cauchy kernel. We encourage this change to UMAP as it makes its loss function equivalent to our Neg-t-SNE and thus also conceptually more related to t-SNE. We also suggest implementing it in the non-parametric case.

4.8 CONTRASTIVE NEIGHBOR EMBEDDINGS AND SELF-SUPERVISED LEARNING

Contrastive self-supervised representation learning [7, 22, 27, 64, 91, 124, 133, 157, 172] and 'contrastive neighbor embeddings' (a term we suggest for NCVis, Neg-t-SNE, UMAP, etc.) are conceptually very similar. The key difference is that the latter use a fixed kNN graph to find similar objects, while the former relies on data transformations or data origin to generate pairs of similar objects on the fly. Other differences include the representation dimension ($\sim$ 128 vs. 2), the use of a neural network for parametric mapping, or the flavor of contrastive loss (InfoNCE [27, 124] vs. NCE / NEG). However, these other differences are not crucial, as demonstrated in this section using our unified PyTorch framework 4.

As an example, we demonstrate that t-SNE can also be optimized using the InfoNCE loss, resulting in InfoNC-t-SNE (Appendix D.2 and Alg. 4). The result of InfoNC-t-SNE on MNIST (Fig. 4.11c) is similar to the result of NCVis (Fig. 4.11b). For

4 https://github.com/lmcinnes/umap/pull/856



(a) $m = 5$ random init (b) $m = 50$ random init (c) $m = 500$ random init (d) t-SNE random init

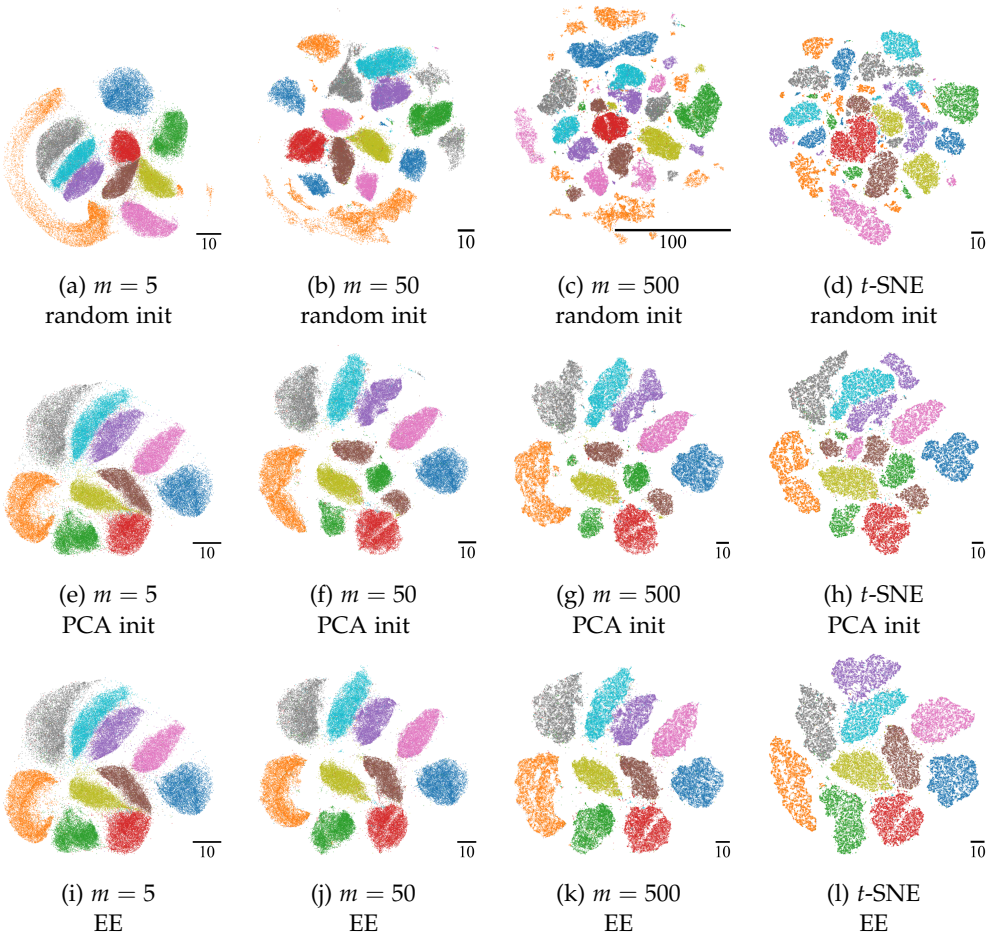
(e) $m = 5$ PCA init (f) $m = 50$ PCA init (g) $m = 500$ PCA init (h) t-SNE PCA init

(i) $m = 5$ EE (j) $m = 50$ EE (k) $m = 500$ EE (l) t-SNE EE

Figure 4.9: NCVis (our implementation) on the MNIST dataset for varying number of noise samples m and different starting conditions. A higher number of noise samples m improves the approximation quality to t-SNE (last column). The first row is initialized with isotropic Gaussian noise and the second and the third rows with PCA (both normalized to have a standard deviation of one or 0.0001 in the first dimension for NCVis or t-SNE, respectively). In the third row, the first 250 epochs use $m = 5$, and the latter use the given m value for NCVis. This is similar to t-SNE's early exaggeration that we use in panel l. NCVis seems less dependent on early exaggeration than t-SNE, especially for low m values.

the default number of noise samples, $m = 5$, both algorithms produce embeddings that are visibly different from t-SNE proper (Fig. 4.1f). Recent work employing the InfoNCE loss for self-supervised learning [27] generally reports improved performance for more noise samples, in agreement with the theoretical results [61, 62, 108]. We observe that both NCVis and InfoNC-t-SNE visualizations using $m = 500$ approximate t-SNE much better (Figs. 4.9k and 4.10k). Like t-SNE, but unlike the $m = 5$ setting, $m = 500$ required early exaggeration to prevent cluster fragmentation, see Figs. 4.9 and 4.10.

Next, we use our PyTorch framework to obtain parametric versions of all contrastive NE algorithms discussed here (NCVis, InfoNC-t-SNE, Neg-t-SNE, UMAP). We use a fully connected neural network with four linear layers and ReLU activations as a parametric $\mathbb{R}^D \to \mathbb{R}^d$ mapping and optimize its parameters using



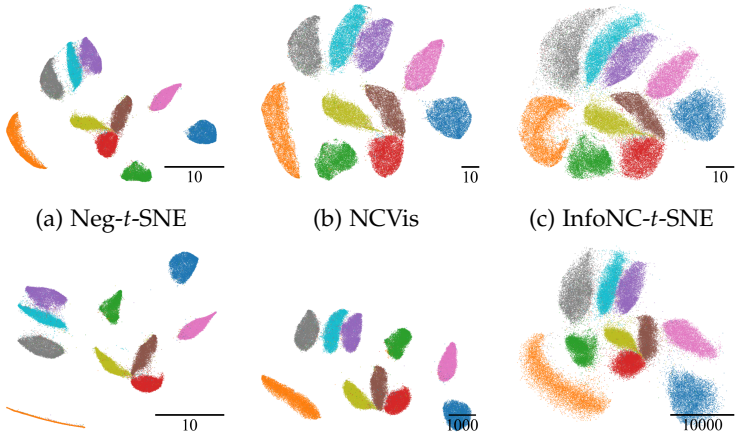
(a) $m = 5$ random init (b) $m = 50$ random init (c) $m = 500$ random init (d) t-SNE random init

(e) $m = 5$ PCA init (f) $m = 50$ PCA init (g) $m = 500$ PCA init (h) t-SNE PCA init

(i) $m = 5$ EE (j) $m = 50$ EE (k) $m = 500$ EE (l) t-SNE EE

Figure 4.10: InfoNC-t-SNE on the MNIST dataset for varying number of noise samples m and different starting conditions. A higher number of noise samples m improves the approximation quality to t-SNE (last column). The first row is initialized with isotropic Gaussian noise and the second and the third rows with PCA (both normalized to have a standard deviation of one or 0.0001 in the first dimension for InfoNC-t-SNE or t-SNE, respectively). In the third row, the first 250 epochs use $m = 5$, and the latter use the given m value for InfoNC-t-SNE. This is similar to t-SNE's early exaggeration that we use in panel l. InfoNC-t-SNE seems to be less dependent on early exaggeration than t-SNE, especially for low m values.

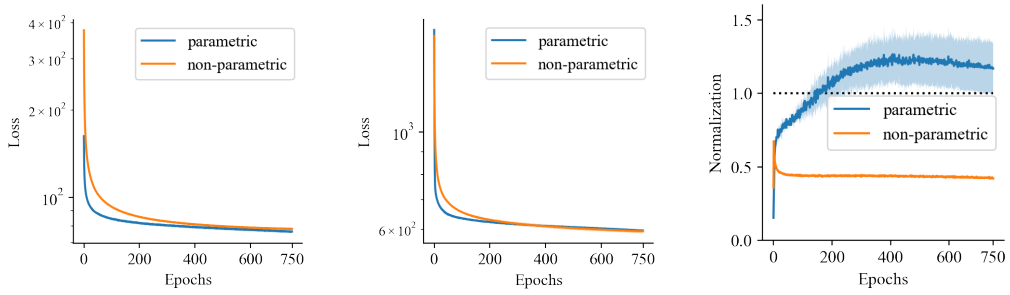
Adam [80] (Appendix D.2). We use batch size $b = 1024$ and sample all m negative samples from within the batch; the data set is shuffled each epoch before batching. Using all four loss functions, we are able to get parametric embeddings of MNIST that are qualitatively similar to their non-parametric versions (Fig. 4.11). The parametric versions of NCE and InfoNCE produce much larger embeddings than their non-parametric counterparts, however the final loss values are very similar (Figs. 4.12a, b). For NCE, the larger scale of the parametric embedding is compensated by a smaller learned normalization parameter Z, so that both parametric and non-parametric versions are approximately normalized (Fig. 4.12c). Our parametric UMAP implementation is very similar to the implementation of Sainburg *et al.* [141]. But our parametric, approximate t-SNE implementations are very different from the parametric t-SNE of van der Maaten [159], which constructed separate kNN



(a) Neg-t-SNE (b) NCVis (c) InfoNC-t-SNE

(d) Parametric Neg-t-SNE (e) Parametric NCVis (f) Parametric InfoNC-t-SNE

Figure 4.11: NE plots of the MNIST dataset are qualitatively similar in the non-parametric (top row) and parametric (bottom row) settings. All panels use our PyTorch framework with $m = 5$ and batch size $b = 1024$.



(a) InfoNC-t-SNE loss (b) NCVis loss (c) NCVis normalization

Figure 4.12: **(a, b)** Loss curves for the parametric and non-parametric InfoNC-t-SNE and NCVis optimizations leading to Figs. 4.11b, c, e, and f. While the embedding scale differs drastically between the non-parametric and the parametric run, the loss values are close. **(c)** Normalization of the model $\sum_{ij} \phi(ij)/Z$ for the parametric and non-parametric NCVis optimizations. The difference in the embedding scale is compensated by three orders of magnitude higher Z so that both versions learn approximately normalized models. These experiments use our NCVis reimplementation.

Table 4.1: Top-1 accuracies on CIFAR-10 representations learned with various contrastive learning losses for different number of noise samples m (in parentheses) and with batch size $b = 1024$. Classification accuracy is computed on a test set, using the ResNet18 output $H \in \mathbb{R}^{512}$. We report the mean and the standard deviation across three random seeds. In the first row, the classifier is trained using data augmentations [27].

	InfoNCE ($2b-2$)	InfoNCE (16)	NCE (16)	NEG (16)
Linear classifier (augm.)	$88.7 \pm 0.6\%$	$91.1 \pm 0.4\%$	$89.3 \pm 0.1\%$	$90.4 \pm 0.6\%$
Linear classifier	$88.7 \pm 0.6\%$	$89.9 \pm 0.7\%$	$86.3 \pm 0.5\%$	$87.9 \pm 0.8\%$
kNN classifier	$89.0 \pm 0.5\%$	$89.4 \pm 0.6\%$	$85.1 \pm 0.3\%$	$86.1 \pm 1.1\%$

graphs within each batch and optimized the vanilla t-SNE loss function, whereas we use the full kNN graph and rely on NCE/InfoNCE losses.

Finally, as a proof-of-concept, we demonstrate that the different contrastive loss functions can all work for image-based self-supervised learning, and specifically, NCE and NEG can work similarly well to InfoNCE in SimCLR [27], using only $m = 16$ noise samples. We use a SimCLR setup to train representations of the CIFAR-10 dataset [87] using a ResNet18 [65] backbone and a fully-connected projector head to 128 embedding dimensions. We use the same data augmentations to create pairs of similar images as in [27], also constrained the output to the sphere S^{128}, and use the exponential of the cosine similarity (Appendix D.2). NCE and NEG produce embeddings of similar quality as InfoNCE, measured by the classification accuracy on the ResNet output (Tab. 4.1). In contrast to existent common practice in the literature [6, 27, 64], we achieve competitive results with only a few, non-curated [137, 171] noise samples. Recent work found conflicting evidence when varying m for a fixed batch size [5, 115, 123]. Future research is needed to perform more systematic benchmarks and to study the effects of the contrastive loss function on the SimCLR performance. Here, we present these results only as a proof-of-principle.

4.9 DISCUSSION AND CONCLUSION

In this work, we studied the relationship between two popular unsupervised learning methods, noise-contrastive estimation (NCE) and negative sampling (NEG). We focused on their application to neighbor embeddings (NE) because this is an active and vital application area, but also because NEs allow visualizing the NCE/NEG outcome directly, forming an intuitive understanding of how different algorithm choices affect the result. Our study makes three conceptual advances.

First, we showed that NEG replaces NCE's learnable normalization parameter Z by a large constant $\bar{Z}$, forcing NEG to learn a scaled data distribution. In the NE setting, this leads to the method Neg-t-SNE, which differs from NCVis [4] (which could be nicknamed "NC-t-SNE") by a simple switch from the learnable to a fixed normalization constant. We argued that this can be a useful hyperparameter to adjust because it moves the embedding along the attraction-repulsion spectrum

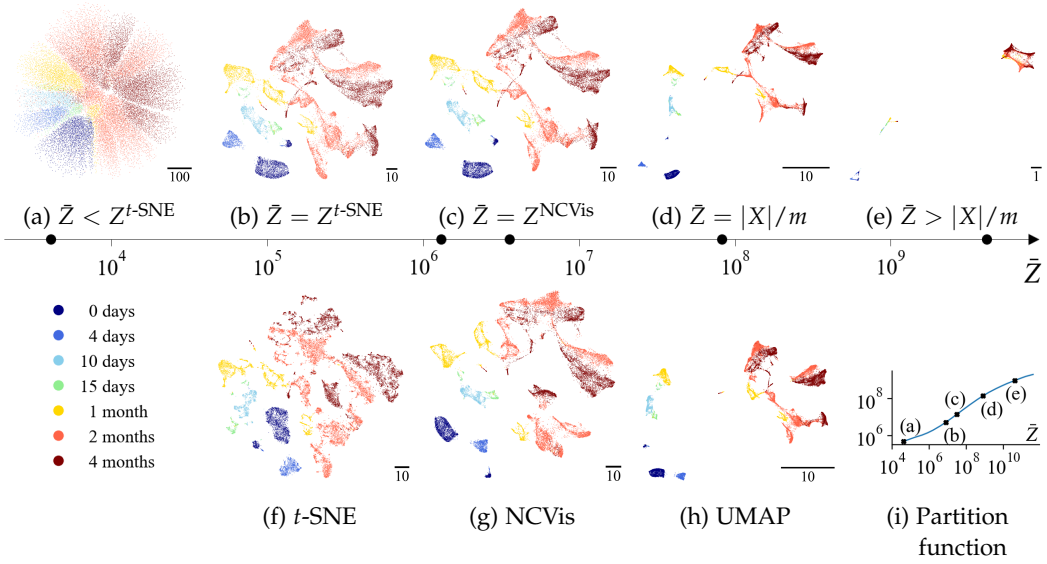


Figure 4.13: **(a – e)** Neg-t-SNE spectrum on the developmental single-cell RNA sequencing dataset from [74] for various parameters $\bar{Z}$. As $\bar{Z}$ increases, the scale of the embedding decreases, and the continuous structure (corresponding to the developmental stage) becomes more apparent, making higher $\bar{Z}$ more suitable for visualizing continuous datasets [15]. The spectrum produces embeddings very similar to those of **(f)** t-SNE and **(g)** NCVis when $\bar{Z}$ equals the partition function of t-SNE or the learned normalization parameter of NCVis. The UMAP embedding in **(h)** closely resembles the Neg-t-SNE embedding at $\bar{Z} = |X|/m = \binom{n}{2}/m$. **(i)** The partition function $\sum_{ij}(1+d_{ij}^2)^{-1}$ of the Neg-t-SNE embeddings increases with $\bar{Z}$. Similar to early exaggeration in t-SNE we start all Neg-t-SNE runs using $\bar{Z} = |X|/m$ and only switch to the desired $\bar{Z}$ for the last two thirds of the optimization.

similar to [15], and hence can either emphasize more discrete or more continuous data patterns (see Fig. 4.13, D.2 and D.3 for examples using single-cell transcriptomic datasets). In addition, when $\bar{Z}$ equals the partition function of a t-SNE embedding, the resulting Neg-t-SNE embedding resembles that of t-SNE. Exploration of the spectrum does not require specialized knowledge. For UMAP $\bar{Z}^{\text{UMAP}} = \binom{n}{2}/m$ and Böhm *et al.* [15] find that t-SNE's partition function $Z^{t\text{-SNE}}$ typically lies in $[50n, 100n]$. Both quantities only depend on the dataset size and the number of noise samples. We envisage an API that allows the user to move along the spectrum with a slider parameter s such that $s = 0, 1$ corresponds to $\bar{Z} = Z^{t\text{-SNE}}, \bar{Z}^{\text{UMAP}}$, respectively, without any need for specifying $\bar{Z}$ directly.

An advantage of our Neg-t-SNE spectrum is the linear time complexity $\mathcal{O}(nmdk)$ of the embedding optimization (m, d, and k are usually small constants), while for t-SNE elaborate approximation schemes are necessary to achieve $\mathcal{O}(nk2^d)$ complexity. These are only implemented for $d = 2$ [101] and would scale exponentially with d. That said, we do not optimize our code for speed, see Appendix D.2, and opt for a simple PyTorch framework that can be easily inspected and adapted by the machine learning community.

An important caveat is that Thm. 4.1 and Cor. 4.3 both assume that the model is rich enough to fit the data distribution perfectly. This is a very strong and unrealistic assumption. In the context of NEs, the data distribution is zero for most pairs of

points, which is impossible to match using the Cauchy kernel. Nevertheless, our theoretical analysis is confirmed experimentally: For higher $\bar{Z}$ the embedding decreases in scale and exhibits more attraction, moving along the attraction-repulsion spectrum, and we find t-SNE-like embeddings at the $\bar{Z}$ value that is expected according to our theory.

Second, we demonstrated that UMAP, which uses NEG, is essentially Neg-t-SNE and differs from it only in UMAP's implicit use of a less numerically stable similarity function. This isolates the key aspect of UMAP's success: Instead of UMAP's appraised [33, 125] high-dimensional similarities, the refined Cauchy kernel, or its stated cross-entropy loss function, it is the application of NEG that lets UMAP perform well and makes clusters in UMAP plots more compact and connections between them more continuous than in t-SNE, in agreement with [15]. To the best of our knowledge, this is the first time UMAP's and t-SNE's loss functions, motivated very differently, have been conceptually connected. Note that here we refer to UMAP's *effective* loss function, which has $\sim n/m$ less repulsion than the one stated in the original paper [39]. Since NCVis' and UMAP's refined Cauchy kernel and the specific high-dimensional similarities of UMAP and t-SNE do not influence the embedding significantly [15, 39], our use of the standard Cauchy kernel and the binary skNN graph throughout the chapter is only a mild limitation.

Third, we argued that contrastive NEs are closely related to the contrastive self-supervised learning methods such as SimCLR [27] which can be seen as parametric InfoNC-t-SNE for learning representations in S^{128} based on the unobservable similarity graph implicitly constructed via data augmentations. We feel that this connection has been underappreciated, with the literature on NEs and self-supervised contrastive learning mostly staying disconnected. To bridge the gap between these two worlds, we presented InfoNC-t-SNE, as well as parametric versions of all considered NE methods, useful for adding out-of-sample data to an existing visualization. Moreover, we demonstrated the feasibility of NCE, InfoNCE, and NEG with few noise samples in the SimCLR setup. Finally, we developed a concise PyTorch framework optimizing *all* of the above, which we hope will facilitate future dialogue between the two research communities.

Overall, our insights into NEs should enable practitioners to explore their datasets more reliably. UMAP and t-SNE plots often help to generate hypotheses in the early stages of a research project. Our conceptual relation between UMAP and t-SNE is therefore of practical impact – as is the fact that both are just instances on a spectrum of embeddings highlighting more continuous or more discrete structures. We also believe that connections between existing approaches and fields are crucial for guiding future research into representation learning, dimensionality reduction, and visualization.

5

CONCLUSION

Most scientific data is collected without labels. For instance, scRNA-seq datasets, which capture the gene expressions of cell populations, reach up to a million data points, each with tens of thousands of measured features. Equipping each data point by hand with a meaningful biological label, such as the cell types, is very expensive if at all possible. Therefore, unsupervised learning techniques are needed to investigate such unlabeled datasets. Many learn a more concise representation of the data, for instance, representing data points by clusters or lower-dimensional vectors. In exploratory data analysis, where labels are unavailable by definition, clustering and dimensionality reduction to two dimensions allow practitioners literally to take a first look at their data. In this thesis, we made contributions both to clustering and dimensionality reduction.

In the first part of the thesis, we focused on clustering. Chapter 2 introduced a new point clustering method, Mod Shift, which uses a threshold on the initial distances to determine which pairs of points attract and, crucially, which repel each other. This interpretable hyperparameter ultimately determines the number of clusters. Mod Shift's inductive bias on distance rather than density sets it apart from established point clustering methods such as Mean Shift [31, 34] and (H)DBSCAN [50, 111]. Through its loss function, Mod Shift resembles a differential version of the signed graph-partitioning problem Multicut. We theoretically explored the resulting approximation of the Multicut polytope via point configurations in Euclidean space and contrasted Mod Shift's and Mean Shift's behavior. On the practical side, we showed Mod Shift's usefulness on toy data and pixel embedding data for a difficult neural segmentation task.

An interesting path for future work is to further flesh out the exciting link between point clustering and graph-partitioning. Our idea is to keep the tie to Multicut by modeling separatedness based on a point configuration in Euclidean space. However, we envisage that a practitioner could inject relational information, e.g., obtained from a boundary predicting neural network, as in [169, 170], as interaction weights instead of inferring them from the initial point configuration. On the practical side, it would be interesting to speed up Mod Shift on image data by limiting the complete interaction between pixels to an (augmented) image grid, as in [169, 170].

In the second part of the thesis, we looked under the hood of the current state-of-the-art for non-linear dimension reduction *t*-SNE and, in particular, UMAP. The implementation of UMAP does not optimize its alleged loss function. In Chapter 3 we identified this problem and derived UMAP's true loss function. As the true loss function effectively binarizes the similarities that UMAP extracts in high dimension, we showed that, while much acclaimed [33, 125], these are not the key aspects of UMAP. Moreover, this binarization explains a common artifact in UMAP plots, overly-crisp substructures, and their prevalence in large datasets. Given UMAP's

popularity, particularly in bioinformatics [9], our insights into its workings are highly relevant.

In the last years, practitioners struggled to choose between UMAP and its predecessor *t*-SNE for non-linear visualization of their datasets. Many arguments in favor of one method or the other were ultimately obsolete, such as the exact weight of similarities in high-dimensional space [15, 39] or easily compensable, like the initialization [84]. Miscrediting UMAP's success to its sophisticated mathematical underpinning [33, 125] obscured the remaining difference.

Based on UMAP's true loss, however, we resolved this controversy and unified the two state-of-the-art visualization methods *t*-SNE and UMAP in Chapter 4. UMAP relates to *t*-SNE via the method NCVis, which approximates *t*-SNE with noise-contrastive estimation. UMAP relies on another contrastive method, negative sampling. We worked out the precise relation between noise-contrastive estimation and negative sampling and thus connected UMAP via NCVis to *t*-SNE. This connection between contrastive losses might be of independent interest, e.g., for the NLP community when learning word embeddings or language models. Our analysis also showed that UMAP uses a numerically unstable similarity kernel in low dimension and requires aggressive learning rate annealing to arrive at a crisp layout. Our principled approach suggested the remedy, too, which would make UMAP equivalent to the method we called "Neg-*t*-SNE".

We found that negative sampling learns a scaled version of the data distribution and thus employs more attraction than *t*-SNE explaining UMAP's more compact embeddings. Other ratios of attraction to repulsion are also possible and give rise to a spectrum of neighbor embedding methods, encompassing approaches similar to UMAP and *t*-SNE as special cases.

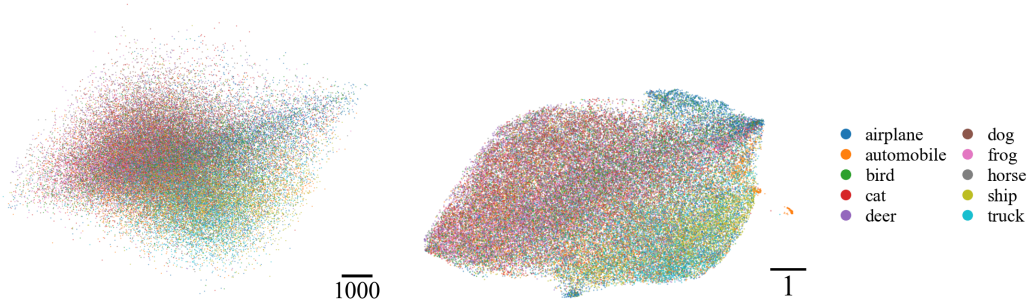
Embeddings along the spectrum focus on different aspects of the data. High attraction embeddings better exhibit continuous structures, e.g., in developmental scRNA-seq datasets. But this comes at the expense of local resolution. When the substructure of the clusters in a dataset is of interest, more repulsion can help. Every visualization necessarily distorts the data [24, 42, 71]. Therefore, we suggested inspecting various points along the attraction-repulsion spectrum to understand a new dataset more accurately.

Finally, we emphasized the ties between contrastive neighbor embeddings and contrastive self-supervised learning in the SimCLR [27] setting and show that both can enrich each other.

We see three lines of future work on dimensionality reduction. First, new metrics that align better with the vague goal of "generating useful research hypotheses" are needed. Chari *et al.* [24] criticize UMAP and *t*-SNE embeddings for not optimizing existing metrics while the merit of their embeddings is undoubtedly immense yet hard to quantify. Such metrics might help to explain the success of *t*-SNE and UMAP as well as guide the development of new visualization methods.

Second, since both methods ultimately rely on neighborhood relations, they have trouble capturing the global aspects of a dataset. More attraction or larger neighborhoods help but do not address the fundamental problem. A fruitful strategy might be combining neighbor embedding methods with classical, global methods such as MDS, as recently done in [89].

Third, the quality of a neighbor embedding hinges on the notion of similarity extracted from the high-dimensional dataset. It is impressive and, so far, not entirely



(a) PCA (b) UMAP

Figure 5.1: UMAP needs meaningful similarities for an insightful embedding. The UMAP plot (5.1b) of the CIFAR-10 dataset [87] shows nearly as little cluster separation as the PCA (5.1a). The kNN graph for UMAP was computed on a 50-dimensional PCA of the raw pixel values. The UMAP embedding does show some interesting structures, such as the separate cluster of automobiles on the right, which consists mostly of near-duplicate images of the same car. When using features from a pre-trained ResNet, similarities become more meaninful and UMAP can achieve descent class separation, see Fig. C.6.

clear why the kNN graph of the raw data goes such a long way. Nevertheless, it is insufficient for a meaningful neighbor embedding of CIFAR-10, see Fig. 5.1. Self-supervised learning employs data augmentations as the source of similarity. Incorporating existing data-agnostic [163, 178] or new data-specific augmentations might extend the practicality of neighbor embedding methods.

In this thesis, we have made contributions chiefly by conceptually relating different aspects of clustering (Chapter 2) and by understanding (Chapter 3) as well as unifying (Chapter 4) popular dimensionality reduction methods. We believe that such theoretical groundwork is crucial for the development of unsupervised learning and hope that it sparks interesting new ideas for discovering structure with little supervision.

APPENDIX

A

SUPPLEMENTARY TO THE INTRODUCTION

We provide the full proof of Lem 1.1 here.

Lemma A.1. *It is possible to place n points equidistantly in $\mathbb{R}^d$ if and only if $d \geq n-1$.*

Proof. This statement follows from Thm.1 in [145], but for intuition, we give a more elementary proof here.

For the "if" part, consider the standard basis vectors in $\mathbb{R}^n$. They all have distance $\sqrt{2}$ from each other but span an affine subspace of dimension $n-1$.

For the "only if" part, we show by induction that n equidistant points in Euclidean space span an affine subspace of dimension $n-1$. This is clear for $n=2$. Let $n>2$ and $x_1,\dots,x_n$ equidistant points in some $\mathbb{R}^d$. The equidistance from x_n to any pair x_i, x_j can be expressed as x_n lying on the hyperplane H_{ij} that orthogonally bisects the line from x_i to x_j. That is,

$$\langle x_i - x_j, x_n - p_{ij}\rangle = 0 \tag{A.1}$$

for any $p_{ij} \in H_{ij}$. By assumption, $x_k \in H_{ij}$ for all $k \neq i,j$. Since we also have $\frac{x_i+x_j}{2} \in H_{ij}$ and H_{ij} is convex, we can choose $p = p_{ij} = \frac{1}{n-1}\sum_{k=1}^{n-1} x_k$ independent of i and j. In other words, the vector $x_n - p$ is orthogonal to all $x_i - x_j$ for $i,j<n$ or zero. We consider $x_n \neq p$ first. By assumption, the equidistant set $\{x_1,\dots,x_{n-1}\}$ spans an affine subspace of dimension $n-2$. So $x_2-x_1,\dots x_{n-1}-x_1$ are linearly independent. As $x_n - p$ is orthogonal to all these vectors, adding it retains linear independence. The difference

$$(x_n - p) - (x_n - x_1) = \frac{1}{n-1}\sum_{k=2}^{n-1}(x_k - x_1) \tag{A.2}$$

is in the span of $x_2-x_1,\dots x_{n-1}-x_1$, so that $x_2-x_1,\dots x_n-x_1$ are also linearly independent. We conclude that the affine dimension of the span of $x_1,\dots,x_n$ is $n-1$ unless $x_n = p$. But the latter case violates equidistance as the midpoint p is not sufficiently far away

$$\|p - x_1\| = \left\|\frac{1}{n-1}\sum_{k=2}^{n-1}(x_k - x_1)\right\| \leq \frac{1}{n-1}\sum_{k=2}^{n-1}\|x_k - x_1\| = \frac{n-2}{n-1}d < d,$$

where d is the distance x_1 has to all other points. □

B

MOD SHIFT

B.1 CHOICES OF w AND ρ

Guided by Sec. 2.4.2 and the proofs of Lem. 2.13 and Props. 2.15 and 2.17, we only consider choices of w that are point symmetric about $(\beta, 0)$, range from -1 to 1, are constant beyond 2β and can be written as $w(d) = \phi(\frac{d}{\beta})$ for some continuous, decreasing function ϕ independent of β, thus illustrating β's role as a scale parameter. Some possibilities are depicted in Fig. 2.2a, which we repeat for the reader's convenience here as Fig. B.1. Their formulae are:

$$w_{\text{lin}}(d) = \max\left(1 - \frac{d}{\beta}, -1\right), \tag{B.1}$$

$$w_{\cos}(d) = \begin{cases} \cos(\frac{d\pi}{2\beta}), & \text{for } d/\beta \in [0,2] \\ -1, & \text{else,} \end{cases} \tag{B.2}$$

$$w_{\log}(d) = \max\left(\frac{\frac{1}{1+\exp(\lambda(d/\beta-1))} - 0.5}{c}, -1\right), \tag{B.3}$$

$$c = \frac{1}{1+\exp(-\lambda)} - 0.5, \lambda > 0. \tag{B.4}$$

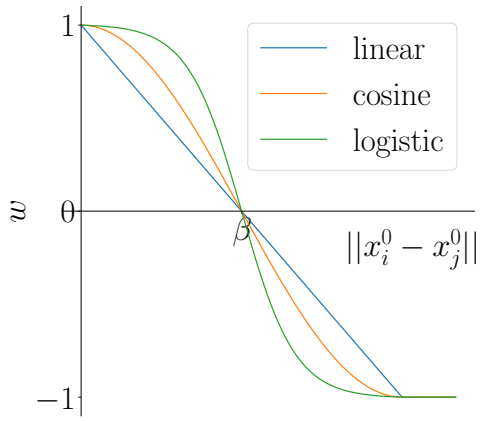


Figure B.1: Different choices of the weight function w given as a rectified linear, cosine, and logistic function. The weight only depends on the initial distance between points and is never updated.

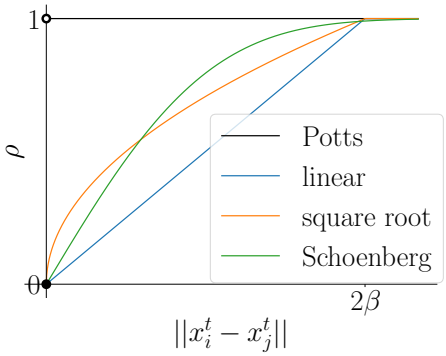


Figure B.2: Separatedness of two points x_i^t and x_j^t after t iterations based on their distance $\|x_i^t - x_j^t\|$ for different choices of ρ: Potts function and concave approximations given by a rectified linear, rectified square root and Schoenberg transformation [146].

Similarly, we gave some examples for possible choices of ρ respecting the considerations in Sec. 2.4.3 in Fig. 2.2b, which we also repeat as Fig. B.2. Their formulae are

$$\rho_{\text{Potts}}(d) = \mathbb{1}(d = 0), \tag{B.5}$$

$$\rho_{\text{lin}}(d) = \min\left(\frac{d}{2\beta}, 1\right), \tag{B.6}$$

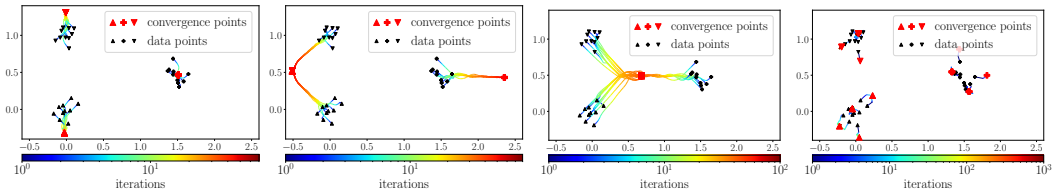
$$\rho_{\text{sqrt}}(d) = \min\left(\sqrt{\frac{d}{2\beta}}, 1\right), \tag{B.7}$$

$$\rho_{\text{Schoenberg}}(d) = \sqrt{1 - \exp\left(-\frac{d^2}{\beta^2}\right)}. \tag{B.8}$$

B.2 IMPLEMENTATION

For an exact execution of a single Mod Shift update, all distances $\|x_i^t - x_j^t\|$ have to be computed, resulting in a quadratic time complexity per iteration as for Mean Shift. Due to the small required number of iterations this results in a complexity of $O(n^2)$, which is tractable with today's GPUs. We use the Python package PyKeOps [25] to avoid a quadratic memory-footprint, while retaining access to PyTorch's [127] GPU-based (stochastic) gradient descent framework with powerful optimizers such as Adam [80] to solve Mod Shift's optimization problem. Our code alongside reproducibility instructions and datasets is publicly available.[1] We ran our experiments on a server with a "Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz" CPU with 56 kernels and a "Nvidia GeForce GTX 1080 Ti" and used the rectified

1 https://github.com/ModShift/ModShift

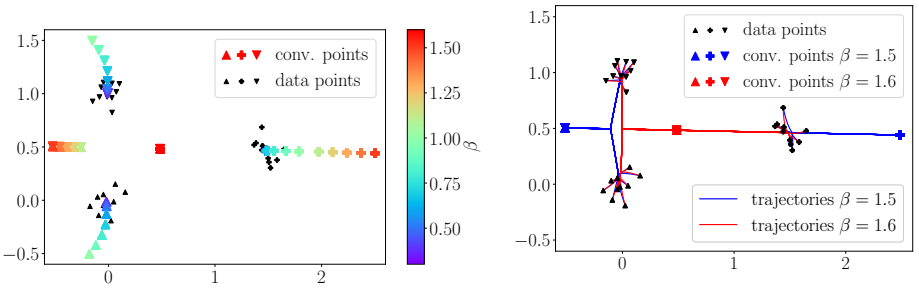


(a) Trajectories for $\beta = 0.7$

(b) Trajectories for $\beta = 1.3$

(c) Trajectories for $\beta = 2.5$

(d) Trajectories for $\beta = 0.15$ with SGD



(e) Convergence points for different values of β.

(f) Trajectories for β just before and after two clusters merge.

Figure B.3: Mod Shift on a toy dataset illustrating the effect of β. B.3a–c show the points' trajectories under Mod Shift for $\beta = 0.7, 1.3$ and 2.5. with the Adam optimizer. B.3d shows the points' trajectories for Mod Shift with SGD and a small enough β that the point clouds are split up. B.3e shows the convergence points for several values of β with SGD optimizer. As β increases, further points attract each other so that the number of convergence points decreases from 3 to 2 at about $\beta = 1$ and from 2 to 1 for β at about 1.6. These values correspond to the mean distances between the point clouds. B.3f shows the trajectories under SGD optimization for β-values close to the jump at about 1.6.

linear functions from Sec. 2.4 for ρ and w. Further hyperparameters are discussed in Appendices B.3 and B.4.

B.3 DETAILS ON THE TOY EXPERIMENT

The toy dataset consists of three sets of 10 points sampled from two-dimensional normal distributions with diagonal covariance of value 0.01 and means $(0,0)$, $(0,1)$ and $(1.5, 0.5)$, respectively. In order to get an exact analysis of Mod Shift's behavior on this toy dataset, we used deliberately low learning rates and vanilla gradient descent. With higher learning rates, Mod Shift converges much faster, but we would get coarser trajectories. Since we want to obtain crisp clusters, we used a learning rate decay towards the end of the optimization (by a factor of 10 in the last 15% of iterations) to deal with the positive gradient of ρ at zero. For the results in Figs. 2.11a–c, we used a learning rate of 0.01 and ran Mod Shift for 30/90/90 iterations, respectively. In order to obtain the number of clusters and convergence points in Figs. 2.11d and B.3e, we used a learning rate of 0.01 and ran Mod Shift for 5000 iterations. To plot the exact trajectories of the β values near cluster merges in Fig. B.3f, we decreased the learning rate to 0.001 and ran for 20000 iterations. We also show trajectories for this dataset with the Adam optimizer using a learning rate of 0.02 without decay and 70/80/100 iterations in Figs. B.3a–c.

In Fig. B.3e, the convergence points move further apart as β increases, before they collapse. This is due to the fact that the function ρ becomes constant at 2β. So while the attraction increases as β increases, the margin by which different clusters should be separated also increases. Once β is large enough that two point clouds have net attraction, they converge to a single cluster, i.e., their convergence points collapse. Such discontinuous behavior cannot be mitigated when one wants to cluster points with a continuously varying scale parameter into well-separated crisp clusters.

We plot the trajectories for two β values slightly above and below such a cluster collapse in Fig. B.3f. The points' trajectories start out similarly before they either diverge from or converge to those of other point clouds. This is because the interaction between points of the same cloud is similar for similar values of β, but the total interaction between different clouds changes sign.

B.4 DETAILS ON THE PIXEL EMBEDDING EXPERIMENTS

B.4.1 *CREMI data*

The data consists of 16-dimensional pixel/voxel embeddings for a $384 \times 384 \times 16$ voxel subvolume of the CREMI A, B, and C datasets [18][2] all predicted by a variant of a 3D U-Net. The network is based on [96] with $(28, 36, 48, 64, 80, 120, 180)$ channels at each resolution before a final convolutional layer reduces the channels to 16. ELU activation functions are used for all but this final convolution. For the loss, offsets are randomly generated, and a logistic similarity is computed from the embedding distances of the thus linked voxel pairs. This similarity is compared to the ground truth similarity using a Dice-loss which compares the separated pairs. While there is thus not a built-in margin by which embeddings of different instances should be separated, we observe that similar scale parameters work best across algorithms and subvolumes. To ease the reproduction of our clustering experiments, we provide the embeddings online. The network was trained jointly on subvolumes of the same size from CREMI A, B, and C. The subvolume we use in our clustering experiments was not used for training the network but was part of the validation set.

B.4.2 *ISBI data*

As in the CREMI experiment, the data consists of 64-dimensional pixel/voxel embeddings of a $512 \times 512 \times 5$ voxel subvolume of the ISBI 2012 Neuro-Segmentation Challenge [3, 20, 21] training dataset.[3] Here, a variant of a PoseNet [29], i.e., two stacked U-Nets with summation instead of concatenation, was used. The network was trained on 2D slices. The data we used here was part of the validation set. The

2 The CREMI dataset uses the CC-BY license. We informed Jan Funke about our use of the dataset.

3 We informed the organizers of the ISBI challenge [3] of our use of the dataset. Its license reads "You are free to use this data set for the purpose of generating or testing non-commercial image segmentation software. If any scientific publications derive from the usage of this data set, you must cite TrakEM2 and the following publication:
Cardona A, Saalfeld S, Preibisch S, Schmid B, Cheng A, Pulokas J, Tomancak P, Hartenstein V. 2010. An Integrated Micro- and Macroarchitectural Analysis of the Drosophila Brain by Computer-Assisted Serial Section Electron Microscopy. PLoS Biol 8(10): e1000502. doi:10.1371/journal.pbio.1000502."

loss was the segmentwise contrastive loss [44] with a pull margin of 0 and a push margin of 1 and $L1$ loss. In order to speed up the clustering process, we reduced the dimensionality of the embeddings from 64 to 8 with PCA applied to each slice individually. This retains most information as the first 8 principal components explain more than 99.8% of the variance in each slice.

B.4.3 *Metrics*

In our experiments, we considered each of the 16 slices of size 384×384 (5 slices of size 512×512) separately, and we used the metrics from [18]. So each method clusters the points of these slices independently, and we compare the results per slice to the ground truth. To aggregate over slices, we take the arithmetic mean of the Variation of Information and Adapted RAND scores of individual slices. The CREMI score is the geometric mean between the averages of the Variation of Information and Adapted RAND scores per slice. We call pixels that neighbor a pixel of a different label in the 4-connected pixel grid boundary pixels. As in [18], the boundary pixels of the ground truth segmentation are excluded when computing the metrics.

B.4.4 *Clustering algorithms*

We compared fixed and adaptive Mod Shift to the two main seed-less clustering methods used for pixel-based instance segmentation: Mean Shift [86] and HDBSCAN [129]. Of these two, only Mean Shift is differentiable and thus directly comparable to Mod Shift. We considered fixed and adaptive Mean Shift with flat and RBF kernels. Single Linkage clustering is used to obtain a hard clustering after the shifting process. A restriction of thresholded Single Linkage to spatial neighbors was used in [107] for pixel-based instance segmentation.

B.4.5 *Implementations*

We used the Single Linkage implementation of [111] for both HDBSCAN and Single Linkage clustering. We implemented Mean Shift and Mod Shift in PyTorch [127] optionally using PyKeOps [25] and we provide these implementations in our GitHub repository.

B.4.6 *Hyperparameters*

We ran both Mean Shift and Mod Shift for 50 iterations. For Mod Shift we used the Adam optimizer [80] with a learning rate of 0.001 without any learning rate decay and the rectified linear functions for ρ and w described in the main text.

B.4.7 *Parameter fine-tuning*

For Mean Shift and Mod Shift we performed a grid search for the respective scale parameters (bandwidth, β) over $[0.01, 0.3]$ in steps of 0.01 for CREMI and

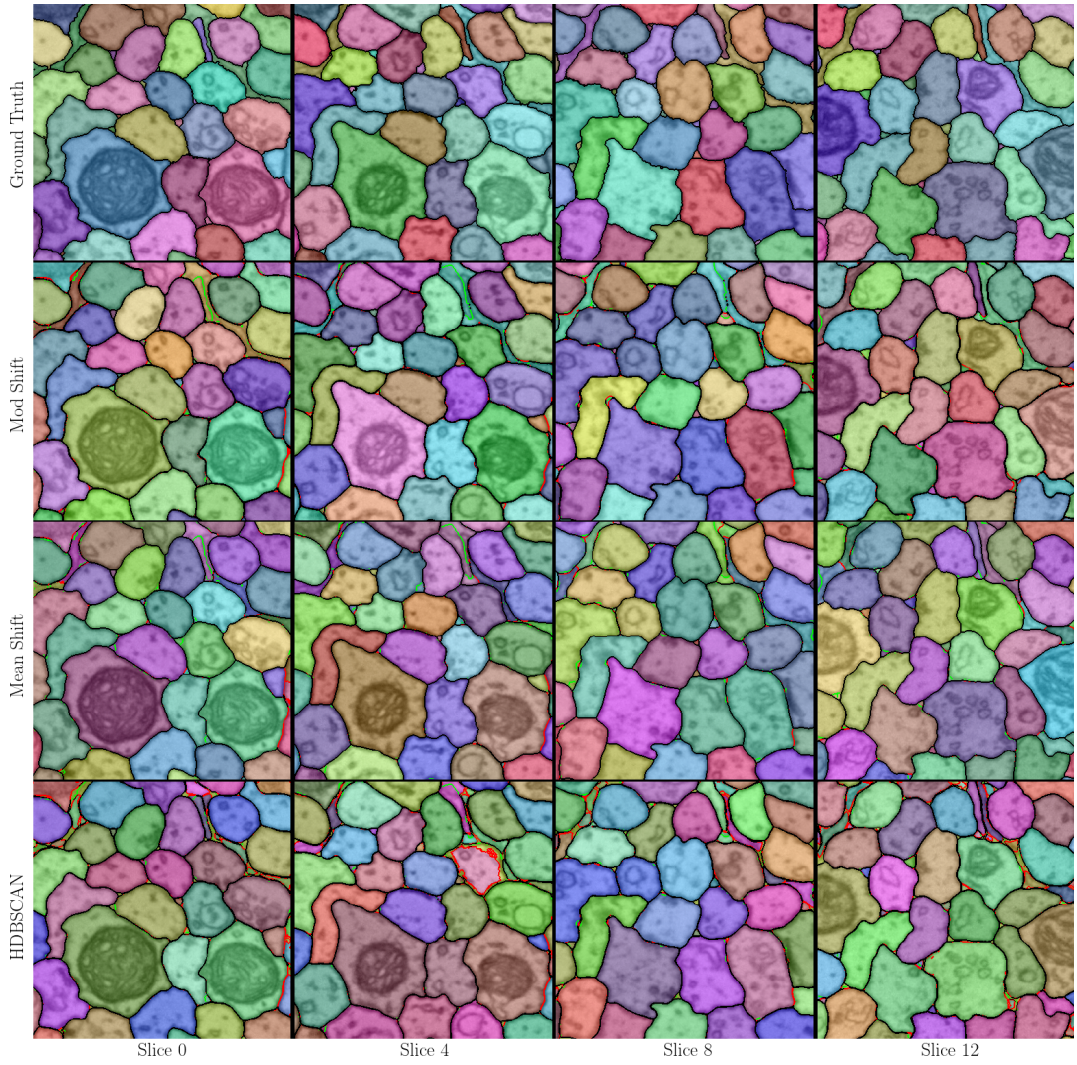


Figure B.4: Segmentations of slices 0, 4, 8 and 12 of the crop of CREMI A. **First row:** Ground truth segmentation with boundary pixel in black. **Rows 2–4:** Segmentations of Mod Shift, Mean Shift and HDBSCAN corresponding to the parameters in lines 3, 7 and 9 of Tab. B.5. Black (green, red) boundaries are correct (false negative, false positive) with respect to the ground truth. A segmentation boundary pixel is considered correct if it is at most one pixel away from a ground truth boundary pixel. Otherwise, it is a false positive. If there is no segmentation boundary pixel at most one pixel away from a ground truth boundary pixel, it is considered a missed boundary. Mod Shift does best on the long elongated segments in the upper left and right corners. Both Mean Shift and HDBSCAN falsely split these up. Mean Shift fails to separate more of the small segments, an indication of Mean Shift's tendency to produce merge errors, see Sec. 2.6.3. HDBSCAN and Mod Shift, with its explicit repulsive forces, do better on these small segments.

over $[0.1, 2.0]$ in steps of 0.1 for ISBI. We varied the threshold for the Single Linkage step that produced the hard clusters for Mean Shift and Mod Shift over $\{0.2, 0.1, 0.04, 0.02, 0.01, 0.001\}$ (for ISBI additionally $\{2., 1, 0.5\}$) and also report the setting where the threshold equals the respective scale parameter. For HDBSCAN we performed a grid search over the two main parameters `min_cluster_size` in $\{2, 5, 10, 20, 50, 100, 150, 200, 250, 300, 350, 400, 450,\ 500, 550, 600\}$ and `min_samples` in $\{1, 2, 5, 10, 20, 50, 100\}$ as well as equal to the parameter `min_cluster_size`. We tried substantially lower values for `min_samples` than the default value of `min_samples=min_cluster_size` in order to reduce the number of pixels that are classified as "noise". After HDBSCAN, the noise points were assigned the label of their closest non-noise neighbor in embedding space. We used the exact minimum spanning tree computation of HDBSCAN. Thus all of our methods are exact. The resulting scores for the optimal parameters in terms of CREMI score can be found in Tabs. B.5–B.8. We always reported the best score of each Mean Shift and Mod Shift clustering method for scale parameter = threshold and an additional setting if varying the threshold improves the result.

B.4.8 *Figures*

Fig. 2.12 shows the results of slice 0 of the crop of CREMI A for Mod Shift and Mean Shift with parameters as in lines 7 and 9 of Tab. B.5. Fig. B.4 adds slices 4, 8 and 12 and HDBSCAN with parameters as in line 3 of Tab. B.5.

B.4.9 *Discussion of results*

The main takeaway message from our experiments is that the best performing Mod Shift setting is competitive with the best performing Mean Shift setting for pixel embedding-based instance segmentation and that both of them tend to be outperformed by non-differentiable HDBSCAN. In most cases, the fixed versions of Mod Shift and Mean Shift yield better results than the respective adaptive versions. In half of the cases, the RBF kernel is better, and in half of the cases worse than the flat kernel for Mean Shift. Moreover, we found that the best performing fixed versions of Mean Shift using a flat kernel and Mod Shift used similar values for their scale parameters, indicating an ideal "range of attraction". As the RBF kernel has infinite support, it is plausible that its best bandwidths were usually lower than for Mean Shift with a flat kernel or Mod Shift. Decreasing the threshold below β often improves the scores of Mod Shift significantly and is crucial for making it competitive, while for Mean Shift, increasing the threshold above the bandwidth is better on the CREMI data and significantly better on the ISBI data. On the majority of experiments decreasing `min_sample_size` below `min_cluster_size` yields the best results, as expected.

Moreover, we found that for all Mod Shift parameter settings described in Tabs. B.5–B.8 98% of all pairs of Mod Shifted points meet the condition of Prop. 2.3 at $\varepsilon = 0.25$.

Table B.1: Comparison of multicut energies [10^3] on downsampled subvolumes of CREMI A–C and of ISBI. Lower is better. Deviation to the lowest energy for each dataset is given in brackets.

	CREMI A	CREMI B	CREMI C	ISBI
GT	−40611	−36950	−38025	−128501
	(127)	(1466)	(537)	(550)
Initial points	−40609	−38158	−38252	−128800
	(129)	(258)	(310)	(251)
Shifted points	−40735	−38403	−38553	−128912
	(3)	(13)	(9)	(139)
Mod Shift	−40737	−37901	−38513	−128913
	(1)	(515)	(49)	(138)
Mutex WS	-40737 ± 0	-38331 ± 9	-38550 ± 1	-129034 ± 0
	(1)	(85)	(12)	(17)
Multicut (KL)	$\mathbf{-40738 \pm 0}$	$\mathbf{-38416 \pm 0}$	$\mathbf{-38562 \pm 0}$	$\mathbf{-129051 \pm 0}$
	(0)	(0)	(0)	(0)

B.4.10 *Mod Shift as a Multicut surrogate*

In this section, we empirically demonstrate that Mod Shift is a good Multicut surrogate on the CREMI and ISBI data. Usually, instance segmentation with Multicut assumes an image grid graph, possibly augmented with additional long-range edges [67]. However, the Multicut energy on this sparse graph is not comparable to Mod Shift's energy on the complete pixel graph. To properly compare the quality of Mod Shift as a Multicut surrogate, we computed the Multicut segmentation on the complete pixel graph as well. In this setting exact Multicut solvers are infeasible, and we opted for a GAEC [78] warm-started approximate Kernighan-Lin solver (KL) as in [169]. Even with this approximate solver, we had to subsample the images by a factor of 4 to achieve acceptable runtimes for the approximate Multicut solver. Wolf *et al.* [169, 170] proposed the Mutex Watershed (Mutex WS) algorithm as a fast alternative to Multicut for signed graph partitioning. We also compare Mod Shift to the Mutex WS algorithm in the above setting.

For each of the best Mod Shift parameter settings in Tabs. B.5–B.8, we computed the graph weights according to Mod Shift's weight function. Given these weights, we computed the approximately optimal Multicut via KL, the optimal Mutex Watershed, and the Mod Shift segmentation. Note that we added Gaussian noise of variance 0.01 on the weights before computing the Multicut or the Mutex Watershed to break ties among the numerous weights that equal -1. Since Mod Shift can shift pixels in different images in parallel in a GPU by treating each image as a batch element, we also computed Mutex WS and KL in parallel over the images of each subvolume.

Tab. B.1 shows the Multicut energies for various methods. In addition to the Multicut energies of the KL, Mutex WS, and Mod Shift segmentations, we report the Mutlicut energy of the ground truth segmentation and the Mod Shift energy (2.5) for the initial point configuration and that of the point configuration after Mod Shift's

Table B.2: Run time [s] comparison Mod Shift, Mutex WS and Multicut on downsampled subvolumes of CREMI A–C and of ISBI. Lower is better.

	CREMI A	CREMI B	CREMI C	ISBI
Multicut (KL)	951 ± 57	1180 ± 55	856 ± 31	4773 ± 304
Mutex WS	611 ± 43	833 ± 23	796 ± 28	1190 ± 66
Mod Shift	$\mathbf{6.9 \pm 0.1}$	$\mathbf{7.5 \pm 0.1}$	$\mathbf{8.1 \pm 0.0}$	$\mathbf{3.9 \pm 0.1}$

Table B.3: CREMI score comparison Mod Shift, Mutex WS and Multicut on downsampled subvolumes of CREMI A–C and of ISBI. Lower is better.

	CREMI A	CREMI B	CREMI C	ISBI
Multicut (KL)	0.0454 ± 0.000	0.6399 ± 0.002	0.1985 ± 0.000	0.1366 ± 0.000
Mutex WS	0.0453 ± 0.000	0.5592 ± 0.005	0.1783 ± 0.003	0.1250 ± 0.003
Mod Shift	**0.0440**	**0.4177**	**0.1702**	**0.0775**

point shifting procedure. We see that the KL segmentation always has the lowest energy. It is the only one that directly optimizes the Multicut energy. The energy after the shifting of Mod Shift is always close to the optimal Multicut energy. This shows that Mod Shift's point shifting procedure approximates the Multicut objective well. The energy of the Mutex WS segmentation is typically similar to that of the points after Mod Shift. Both are much lower than the energies of the input point configuration or the ground truth segmentation. In particular, we see that the point shifting of Mod Shift indeed reduced the energy drastically. On half of the datasets, the energy of Mod Shift's segmentation, i.e., after the point shifting and the single linkage clustering, is similar to that of the point configuration after shifting. On the other subvolumes, it is much higher. Note that the single linkage threshold was chosen to optimize the CREMI score, not the Multicut energy. The ground truth segmentation always has high Multicut energy, which shows that both objectives are not perfectly aligned for this task. For KL and Mutex WS, we give averages over five runs with one standard deviation. The other energies do not depend on randomness. Note that in the complete pixel graph, most pairs of pixels repel each other and have a weight of -1, resulting in very negative energies.

We show the run times of KL, Mutex WS, and Mod Shift in Tab. B.2 averaged over five runs and with an uncertainty of one standard deviation. Mod Shift is much faster than both the KL-based Multicut or Mutex WS on these complete pixel graphs.

At the same time, Tab. B.3 shows that Mod Shift yields competitive CREMI scores in this setting. Note that these scores are on the downsampled images and thus differ from the ones reported in Tabs. B.5–B.8.

B.4.11 *Stability and run times*

The implementations we use for Mod Shift, Mean Shift, and HDBSCAN do not depend on any random seeds. Hence, results across different runs do not vary.

Table B.4: Run times [s] of representative experiments averaged over 5 runs and with one standard deviation. We always used the best setting for flat, non-adaptive Mean Shift, non-adaptive Mod Shift and HDBSCAN in Tabs. B.5–B.8.

	CREMI A	CREMI B	CREMI C	ISBI
HDBSCAN	102 ± 1	96 ± 1	99 ± 1	92 ± 58
Mean Shift	392 ± 4	480 ± 62	454 ± 1	224 ± 1
Mod Shift	631 ± 33	609 ± 2	726 ± 1	310 ± 1

Therefore, we have omitted giving error bars for the outcomes of these results. As discussed in Sec. B.4.10, the Multicut and Mutex Watershed implementations do depend on random seeds, and we give means over five runs and uncertainties of one standard deviation in Tab. B.3 for these methods.

We have measured the run times of representative experiments. We always ran five times and reported the mean run time and one standard deviation. The results on the downsampled images are given in Tab. B.2. The results on the original data are in Tab. B.4. Note that we always excluded the one-time cost of computing the PyKeOps kernels for Mod Shift and Mean Shift. Moreover, we computed the hard cluster assignment and HDBSCAN sequentially over the different images of a subvolume. This could be parallelized.

Let us estimate the total run time of reproducing all experiments in this chapter. We run Mutex Watershed and Kernighan-Lin five times for each of the four downsampled subvolumes. The Mod Shift times are negligible here. On each of the full-size CREMI subvolumes, we compute Mean Shift and Mod Shift for 30 scale parameters. The time for computing the results of the seven thresholds does not matter significantly, so we omit this factor. On the ISBI subvolume, we run Mean Shift and Mod Shift for 20 scale parameters each. Moreover, we compute Mean Shift with a flat and a Gaussian kernel and both Mean and Mod Shift in the adaptive and fixed versions. For HDBSCAN, we search over eight choices of `min_sample_size`. The search over `min_cluster_size` does not matter much. Using the run times in Tab. B.2 and the representative run times in Tab. B.4 we thus compute a total run time of 106 hours if all instance segmentation experiments were to be run one after the other. The experiments on the toy data were very fast, so we do not include them in this estimate.

B.5 CONVERGENCE OF MOD SHIFT WITH ADAM

We propose to optimize Mod Shift's non-convex objective function with modern deep learning optimizers, such as Adam [80], motivated by their outstanding success at optimizing the highly non-convex objective functions of deep neural networks [16]. For classical clustering methods, proofs of convergence are essential. Gradient descent, however, is not necessarily guaranteed to converge, especially without appropriate step-size annealing, see, for instance, the Perceptron Cycling Theorem [114]. Nevertheless, we did not experience convergence issues when optimizing Mod Shift's objective function using Adam, even with a fixed step size. In the following, we describe tweaks of the Mod Shift setup in which sufficient

conditions for convergence are known. As our experiments did not make them necessary, we have not pursued them experimentally.

While the general convergence of Adam-type optimizers in non-convex settings is an open problem, [28] provides assumptions and a sufficient condition for convergence to a stationary point of the objective function. The assumptions require the objective function to be differentiable with Lipschitz gradient, lower bounded, and to have bounded true and noisy gradients. In addition, the noisy gradients need to be unbiased and of independent noise.

For Mod Shift, the assumption on the noisy gradient is satisfied as we usually perform full-batch gradient updates and thus only work with the proper gradients. Mod Shift's objective function E in (2.4) satisfies the remaining assumptions for certain choices of ρ. Differentiability at generic point configurations, boundedness, and Lipschitz-continuity of the gradient can be easily achieved by choosing a ρ with these properties, as evident from the formula for Mod Shift's gradient in (2.6). Kinks such as in the rectified linear ρ that we used in our experiments could be removed by approximation with a smooth function. All that is missing is the differentiability of E at point configurations in which points coincide. This can be pragmatically achieved with a ρ of vanishing gradient at zero. However, as a consequence, ρ would not fulfill the theoretical desiderata discussed in Sec. 2.4.3 as it would neither be concave nor of positive slope at zero. Nevertheless, in this case, Thm. 3.1 in [28] provides a sufficient condition for the convergence of Adam-type optimizers, and Sec. 3.3 in [28] describes annealing learning-rate schemes that guarantee convergence of such a tweaked Mod Shift using certain variants of Adam.

B.6 DETAILED RESULTS

Table B.5: Results on crop of CREMI A by metrics of [18], lower is better. Mod Shift outperforms Mean Shift, HDBSCAN is worse than both.

Method	parameters	CREMI score	ARAND	VOI split	VOI merge
HDBSCAN	`min_samples` $= 50$, `min_cluster_size` $= 50$	0.0704	0.0231	0.1688	0.0459
Gaussian Mean Shift	bandwidth $= 0.03$, threshold $= 0.03$	0.0657	0.0219	0.1625	0.0344
Gaussian adaptive Mean Shift	bandwidth $= 0.03$, threshold $= 0.03$	0.0668	0.0226	0.1630	**0.0343**
Gaussian adaptive Mean Shift	bandwidth $= 0.03$, threshold $= 0.1$	0.0666	0.0225	0.1622	0.0344
flat Mean Shift	bandwidth $= 0.11$, threshold $= 0.11$	0.0629	0.0215	0.1408	0.0428
flat adaptive Mean Shift	bandwidth $= 0.12$, threshold $= 0.12$	0.0610	0.0212	0.1295	0.0462
Mod Shift	$\beta = 0.09 =$ threshold	0.0628	0.0217	0.1446	0.0367
Mod Shift	$\beta = 0.11$, threshold $= 0.1$	**0.0601**	**0.0211**	**0.1280**	0.0430
adaptive Mod Shift	$\beta = 0.12 =$ threshold	0.0648	0.0222	0.1465	0.0426

Table B.6: Results on crop of CREMI B by metrics of [18], lower is better. Varying the threshold is crucial to obtain descent Mod Shift results. Here, HDBSCAN beats both Mean Shift and Mod Shift.

Method	parameters	CREMI score	ARAND	VOI split	VOI merge
HDBSCAN	`min_samples` $= 10$, `min_cluster_size` $= 300$	**0.2930**	**0.1189**	0.4371	0.2852
Gaussian Mean Shift	bandwidth $= 0.05$, threshold $= 0.05$	0.3401	0.1445	**0.4362**	0.3643
Gaussian Mean Shift	bandwidth $= 0.04$, threshold $= 0.1$	0.3190	0.1354	0.4511	0.3006
Gaussian adaptive Mean Shift	bandwidth $= 0.04$, threshold $= 0.04$	0.3523	0.1532	0.5248	0.2850
Gaussian adaptive Mean Shift	bandwidth $= 0.04$, threshold $= 0.1$	0.3522	0.1532	0.5237	0.2857
flat Mean Shift	bandwidth $= 0.09$, threshold $= 0.09$	0.3380	0.1458	0.4954	0.2883
flat Mean Shift	bandwidth $= 0.09$, threshold $= 0.1$	0.3378	0.1457	0.4923	0.2908
flat adaptive Mean Shift	bandwidth $= 0.09$, threshold $= 0.09$	0.3523	0.1516	0.5665	**0.2524**
flat adaptive Mean Shift	bandwidth $= 0.1$, threshold ≤ 0.02	0.3509	0.1518	0.5370	0.2740
Mod Shift	$\beta = 0.02 =$ threshold	0.4097	0.1600	0.6885	0.3604
Mod Shift	$\beta = 0.1$, threshold $= 0.02$	0.3564	0.1443	0.5293	0.3511
adaptive Mod Shift	$\beta = 0.09 =$ threshold	0.6858	0.3339	0.9880	0.4205
adaptive Mod Shift	$\beta = 0.17$, threshold $= 0.04$	0.3572	0.1371	0.5910	0.3398

Table B.7: Results on crop of CREMI C by metrics of [18], lower is better. Mod Shift outperforms Mean Shift if the threshold is decreased. HDBSCAN is better than both.

Method	parameters	CREMI score	ARAND	VOI split	VOI merge
HDBSCAN	`min_samples` $= 20$, `min_cluster_size` $= 150$	**0.1472**	**0.0459**	**0.3551**	0.1169
Gaussian Mean Shift	bandwidth $= 0.04$, threshold $= 0.04$	0.1742	0.0569	0.4154	0.1177
Gaussian Mean Shift	bandwidth $= 0.03$, threshold $= 0.1$	0.1628	0.0530	0.4005	0.0990
Gaussian adaptive Mean Shift	bandwidth $= 0.03$, threshold $= 0.03$	0.1959	0.0638	0.5048	0.0968
flat Mean Shift	bandwidth $= 0.08$, threshold $= 0.08$	0.1800	0.0586	0.4425	0.1100
flat Mean Shift	bandwidth $= 0.06$, threshold $= 0.1$	0.1764	0.0580	0.4383	**0.0986**
flat adaptive Mean Shift	bandwidth $= 0.09$, threshold $= 0.09$	0.1932	0.0632	0.4619	0.1288
flat adaptive Mean Shift	bandwidth $= 0.09$, threshold $= 0.1$	0.1923	0.0629	0.4587	0.1288
Mod Shift	$\beta = 0.03 =$ threshold	0.1969	0.0616	0.5078	0.1217
Mod Shift	$\beta = 0.1$, threshold $= 0.04$	0.1599	0.0497	0.3823	0.1317
adaptive Mod Shift	$\beta = 0.11 =$ threshold	0.2432	0.0820	0.5146	0.2068
adaptive Mod Shift	$\beta = 0.25$, threshold $= 0.01$	0.2058	0.0559	0.6371	0.1208

Table B.8: Results on crop of ISBI by metrics of [18], lower is better. Reducing the threshold for Mod Shift is is crucial to make it competitive.

Method	parameters	CREMI score	ARAND	VOI split	VOI merge
HDBSCAN	`min_samples` $= 20$, `min_cluster_size` $= 350$	0.0762	**0.0260**	0.1097	0.1130
Gaussian Mean Shift	bandwidth $= 0.7$, threshold $= 0.7$	0.0848	0.0376	0.1173	0.0739
Gaussian Mean Shift	bandwidth $= 0.6$, threshold $= 2$	0.0815	0.0356	0.1149	0.0717
Gaussian adaptive Mean Shift	bandwidth $= 0.6$, threshold $= 0.6$	0.0870	0.0382	0.1267	0.0714
Gaussian adaptive Mean Shift	bandwidth $= 0.6$, threshold $= 2$	0.0868	0.0382	0.1260	0.0714
flat Mean Shift	bandwidth $= 1.5$, threshold $= 1.5$	0.0852	0.0378	0.1214	0.0708
flat Mean Shift	bandwidth $= 0.5$, threshold $= 1.0$	**0.0759**	0.0279	0.1645	**0.0422**
flat adaptive Mean Shift	bandwidth $= 1.5$, threshold $= 1.5$	0.0897	0.0390	0.1380	0.0682
flat adaptive Mean Shift	bandwidth $= 0.2$, threshold $= 0.5$	0.0789	0.0285	0.1631	0.0549
Mod Shift	$\beta = 0.5 =$ threshold	0.0952	0.0444	**0.0936**	0.1102
Mod Shift	$\beta = 0.7$, threshold $= 0.2$	0.0877	0.0316	0.1828	0.0609
adaptive Mod Shift	$\beta = 0.4 =$ threshold	0.0781	0.0318	0.1078	0.0841

C

ON UMAP'S TRUE LOSS FUNCTION

C.1 IMPLEMENTATION DETAILS

To deal with the quadratic complexity when computing the dense low-dimensional similarities v_{ij}, we used the Python package PyKeOps [25] that parallelizes the computations on the GPU.

The repulsive loss $\mathcal{L}^r_{ij}$ is undefined for $i = j$. However, the implemented gradient update treats $\frac{\partial \mathcal{L}^r_{ii}}{\partial e_i}$ as zero. As $\mu_{ii} = 0$, we can thus safely replace $2\sum_{j=1}^{n}$ by $2\sum_{j=1, i\neq j}^{n}$ in formulae for expected updates. Moreover, we may treat $\mathcal{L}^r_{ii}$ as a constant so that its inclusion in the loss does not matter for the optimization. Together with the symmetry of μ_{ij}, $\mathcal{L}^a_{ij}$ and $\mathcal{L}^r_{ij}$, we can thus also replace $\sum_{i,j=1}^{n}$ with $2\sum_{1\leq i<j\leq n}$ in formulae for loss functions. When logging losses, we include the $\mathcal{L}^r_{ii}$ terms.
To guard us against numerical instabilities from $\log(x)$ for x close to zero, we always use $\log(\min(x + 0.0001, 1))$.

When computing the various loss terms for UMAP, we use the embeddings after each full epoch. The embeddings in UMAP are updated as soon as an incident edge is sampled. Thus, an embedding might be updated several times during an epoch, and gradient computations always use the current embedding, which might differ slightly from the embedding after the whole epoch. Logging the loss given the embeddings at the time of each individual update yields a slightly lower attractive loss term, see Fig. C.1.

Our description of UMAP's implementation is based on the original paper [112] and version 0.5.0 of the umap-learn package.[1]

Our code is available at `https://github.com/hci-unihd/UMAPs-true-loss`.

C.1.1 *Stability*

Whenever we report loss values, we compute the average over seven runs and give an uncertainty of one standard deviation. Sources of randomness are in the approximate kNN computation via nearest neighbor descent [47], the Gaussian noise added to the Laplacian Eigenmap initialization, the negative sampling, and the sampling of the toy data. Note that the sampling of attractive pairs is implemented deterministically and includes an edge ij every $\max_{ab} \mu_{ab}/\mu_{ij}$-th epoch. We find that the deviation in the loss values is very small across different runs. In fact, as the standard deviation is barely visible in Fig. 3.4, we include the same figure but with shaded areas corresponding to ten standard deviations in Fig. C.2. Nevertheless, the visual effect of different random seeds can be noticeable, as depicted in Fig. C.3.

1 https://github.com/lmcinnes/umap

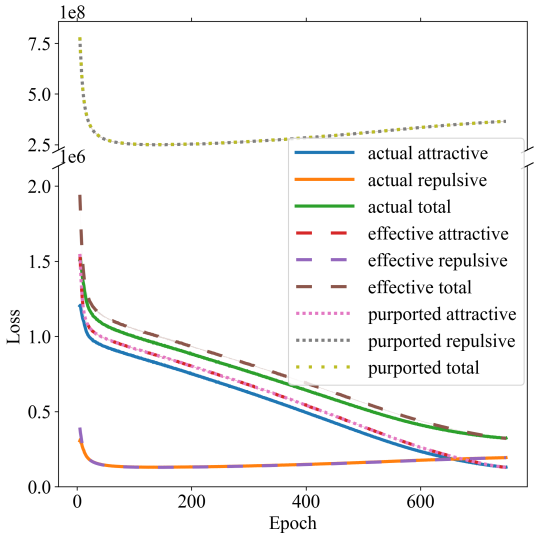


Figure C.1: Same as Fig. 3.4, but actual losses are computed with the embeddings at the time of update, not with the embeddings after the full epoch as all other losses.

C.1.2 *Compute*

We ran all our experiments on a machine with 20 "Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz" CPUs and six "Nvidia GeForce GTX 1080 Ti" GPUs. We only ever used a single GPU and solely for computing the effective and purported losses $\tilde{\mathcal{L}}$ (eq. (3.12)) and $\mathcal{L}$ (eq. (3.6)). Tab. C.1 shows the run times for the main experiments averaged over seven runs. Uncertainties indicate one standard deviation. Logging the losses during optimization quintuples UMAP's run time on the C. elegans dataset. This is due to the quadratic complexity of evaluating the effective and purported loss functions. Nevertheless, with our GPU implementation, this longer run time is still easily manageable for the reasonably large real-world C. elegans dataset. The toy ring experiments with a dense and thus much larger input graph take about 25 times longer than with the standard, sparse input similarities.

We estimate the total compute by adding the run times of this chapter's experiments. The number of comparable experiments needed to reproduce the chapter is given in Tab. C.1. The total run time amounts to about 63 hours.

C.2 QUANTITATIVE METRICS

While it is natural to wonder about metrics that quantify how accurate and thus "good" a low-dimensional visualization is, there is no consensus as to which metrics align best with the subjective, human impression of a "descent" visualization. Becht *et al.* [9] employ the Pearson correlation between all pairwise distances in high- and low-dimensional space for a subsample of the dataset (for efficiency). Kobak and Berens [83] use the Spearman correlation instead and consider it a "global" measure for the faithfulness of the embedding. As "local" measure Kobak and Berens [83] compute the average share of k nearest neighbors of a point in high dimension that are also k nearest neighbors in the low-dimensional embedding. They use

2 Only two runs were measured due to the long run time.

Table C.1: Run times of key experiments averaged over seven runs with one standard deviation and number of runs of similar experiments needed to reproduce the chapter

Experiment	Run rime [s]	Runs
C. elegans without loss logging (Fig. C.10)	520 ± 14	15
C. elegans with loss logging (Fig. 3.3)	2427 ± 7	21
PBMC (Fig. C.4)	2087 ± 7	7
Lung Cancer (Fig. C.5)	1491 ± 4	7
CIFAR-10 (Fig. C.6)	302 ± 0.3	7
Toy ring (Fig. 3.2b)	48 ± 0.01	12
Multiple rings (Fig. 3.8)	1726 ± 17	7
Toy ring of variable size (Fig. 3.10)	1647 ± 13	7
Toy ring with dense μ_{ij}'s (Fig. 3.2d)	1040 ± 2	10
Toy ring with dense μ_{ij}'s and variable size (Fig. 3.12)[2]	52349 ± 224	2

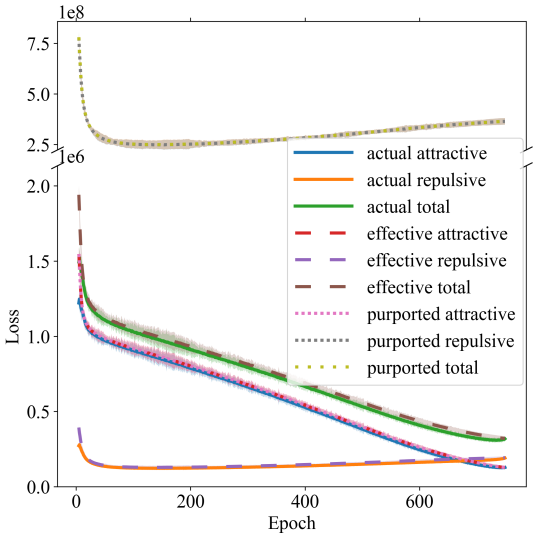
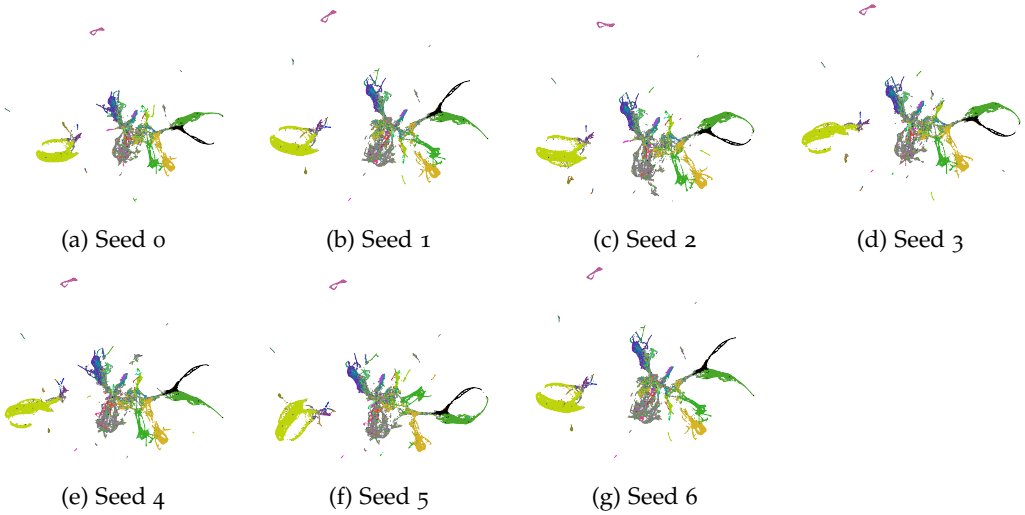


Figure C.2: Same as Fig. 3.4 but here the shaded region corresponds to ten standard deviations.



(a) Seed 0 (b) Seed 1 (c) Seed 2 (d) Seed 3

(e) Seed 4 (f) Seed 5 (g) Seed 6

Figure C.3: Visualizations of the C. elegans dataset with UMAP for seven different random seeds. While the losses vary very little, see Figs. 3.4 and C.2, the visualizations show significant differences, such as closed versus open loops and placement of subgroups. All plots were subjectively flipped and rotated by multiples of $\pi/2$ to ease a visual comparison.

$k = 10$. We have computed these measures for the two-dimensional visualizations of the C.elegans dataset with PCA, UMAP, and the UMAP version with inverted weights, see Fig. 3.3. We used subsamples of size 10000 from the 86024 cells in the C. elegans dataset for the correlation measures. For the kNN accuracy we used $k = 10$ as in [83] as well as $k = 30$, because on the C. elegans dataset we used 30 neighbors in UMAP's approximate kNN graph computation. In addition, we report Kullback-Leibler divergences between the input and embedding similarities as this is the objective function in t-SNE [160, 161]. Unlike in t-SNE, in UMAP, neither the input nor the embedding similarities are normalized over all pairs of points. We report two versions of the KL divergence. In both, we normalize the input similarities. In the first, we normalize the embedding similarities only with respect to the pairs of points with positive input similarity. In the second, we normalize with respect to all pairs of points. Finally, we compute the loss values of UMAP's purported and true loss function. Both in the KL divergences and the two losses, we take the original UMAP input similarities as reference. We report the mean and standard deviation over seven runs in Tab. C.2.

As previously observed by Kobak and Berens [83], PCA is good at preserving the global structure, as indicated by the high correlation coefficients. However, we also see a very high Pearson correlation for the inverted UMAP setting. Conversely, PCA is much worse than both UMAP methods at preserving the local neighborhood structure. This is unsurprising as UMAP tries to reproduce local similarities in low-dimensional space, which are positive only on the kNN edges. Our binarization analysis shows that UMAP essentially just uses the binary high-dimensional kNN structure to guide its low-dimensional embedding. The inverted UMAP setting still does much better than PCA on the kNN accuracies, but not quite as well as the original UMAP, presumably, because the binarization does not entirely efface the effect of inverting the positive input similarities.

Table C.2: Quantitative measures for visualizations of the C.elegans dataset with PCA, UMAP with inverted positive input weights, and original UMAP. Arrows indicate whether higher or lower is better. Best method in **bold**.

Metric	PCA	UMAP inv	UMAP
Pearson's r ↑	0.577 ± 0.003	$\mathbf{0.628 \pm 0.005}$	0.562 ± 0.008
Spearman's r ↑	$\mathbf{0.611 \pm 0.003}$	0.607 ± 0.006	0.551 ± 0.011
kNN accuracy ($k = 10$) ↑	0.006 ± 0.000	0.079 ± 0.001	$\mathbf{0.156 \pm 0.001}$
kNN accuracy ($k = 30$) ↑	0.013 ± 0.000	0.185 ± 0.001	$\mathbf{0.256 \pm 0.001}$
KL divergence pos ↓	0.693 ± 0.000	0.577 ± 0.002	$\mathbf{0.549 \pm 0.002}$
KL divergence ↓	6.46 ± 0.00	5.13 ± 0.01	$\mathbf{4.92 \pm 0.00}$
purported UMAP loss [10^8] ↓	5.37 ± 0.00	4.32 ± 0.03	$\mathbf{3.64 \pm 0.02}$
true UMAP loss [10^5] ↓	12.4 ± 0.00	3.85 ± 0.02	$\mathbf{3.21 \pm 0.02}$

The two KL divergences and the actual UMAP loss paint a similar picture. The inverted UMAP version is nearly as good as the original, while PCA is much worse. The ranking of the purported UMAP loss is the same, but the difference between PCA and the two UMAP versions is less extreme.

The results of the kNN accuracies, the KL divergences, and the true UMAP loss are in accordance with the visual quality of the embeddings and might suggest that our effective UMAP loss aligns well with what one might call a "good" embedding.

C.3 DATASETS

C.3.1 *Toy datasets*

The support for the uniform distribution of the toy rings has a width of one and a radius (measured to the middle of the ring's width) of four. When placing multiple rings in a row, we spaced them so that neighboring rings have a distance of 12 between their centers. The uniform dataset is sampled uniformly from a unit square.

Unless otherwise stated, we kept all UMAP hyperparameters at their defaults, but optimized for 10 000 epochs to ensure convergence. We also initialized the optimization of the embeddings with the 2D datasets themselves.

C.3.2 *C. elegans dataset*

The C. elegans dataset contains gene expression data of 86 024 cells of the flatworm C. elegans [116, 126].[3]

We start with a 100 dimensional PCA of the data, use the cosine metric in high-dimensional space and consider $k = 30$ neighbors. We optimize the embedding for 750 epochs. We perturb the input similarities in Figs. 3.3b, C.10, and C.11 as follows: In the permuted case, we permute the weights of the skNN graph edges. This way

3 obtained from http://cb.csail.mit.edu/cb/densvis/datasets/. We informed the authors of our use of the dataset, which they license under CC BY-NC 2.0.

the statistics of the input similarities are preserved. All zero weights remain zero. In the uniformly random case, we sample new weights for the skNN graph edges uniformly between zero and one, but keep the weights for all edges not in the skNN graph at zero. Before optimization, UMAP filters its graph by setting all weights below $\max_{ij}(\mu_{ij})/\text{n\_epochs}$ to zero. In the inverted setting, we filter the original UMAP graph in this way manually. Then, we select the smallest positive edge weight $\mu_{\min}$ of this manually filtered graph. Finally, we invert the positive weights of the filtered graph to $\mu_{\min}/\mu_{ij}$. The resulting weighted graph is then passed to UMAP, which filters it again internally.

C.3.3 *PBMC dataset*

In this section, we analyze the UMAP embedding of the scRNA-seq PMBC dataset of [179]. The dataset consists of gene expression measurements for 68 551 peripheral blood mononuclear cells. We used the 50-dimensional pre-processed version of the dataset from [116].[4] We used the same hyperparameters as for the C. elegans dataset. In other words, we used the cosine distance in input space, worked with $k = 30$ nearest neighbors, and optimized for 750 epochs. The results can be found in Fig. C.4. We see over-contraction in the embedding, that only our effective loss matches the measured sampling-based loss and that the embedding similarities approximate the nearly binary target similarities.

C.3.4 *Lung cancer dataset*

In this section, we analyze the UMAP embedding of the scRNA-seq dataset of [180] containing gene expression measurements for 48 969 lung cancer and immune cells. We used the 306-dimensional pre-processed version of the dataset from [116].[5] We used the same hyperparameters as for the C. elegans dataset. In other words, we used the cosine distance in input space, worked with $k = 30$ nearest neighbors, and optimized for 750 epochs. The results can be found in Fig. C.5. We see over-contraction in the embedding, that only our effective loss matches the measured sampling-based one and that the embedding similarities approximate the nearly binary target similarities.

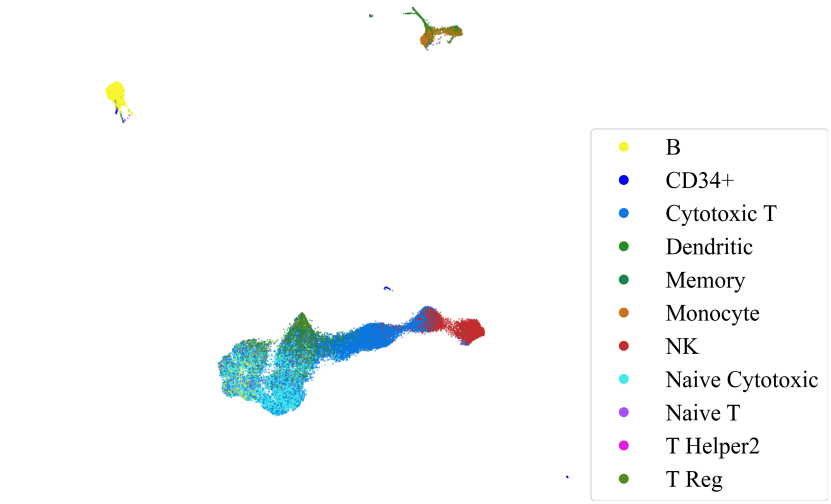
C.3.5 *Resnet50 features of CIFAR-10*

In this section, we corroborate our results on image data. More precisely, we use the CNN backbone of a Resnet50 [65] pre-trained on ImageNet as a feature extractor. With it we obtain 2048-dimensional image features for the CIFAR-10 dataset [87]. These high-dimensional features are then embedded to two dimensions via UMAP with default hyperparameters. The resulting embedding and the loss curves are depicted in Fig. C.6, which supports our theoretical predictions.
We used the pre-trained Resnet50 from torchvision 0.8.2 and transformed the

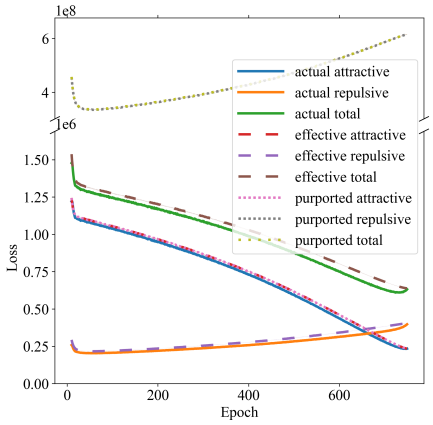
4 obtained from http://cb.csail.mit.edu/cb/densvis/datasets/. We informed the authors of our use of the dataset, which they license under CC BY-NC 2.0.

5 obtained from http://cb.csail.mit.edu/cb/densvis/datasets/. We informed the authors of our use of the dataset, which they license under CC BY-NC 2.0.

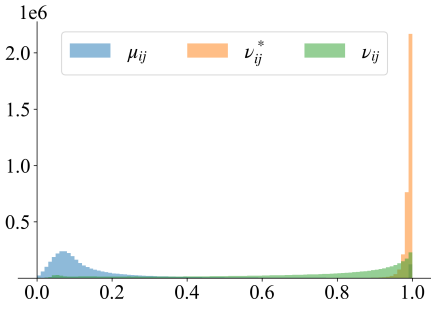
CIFAR-10 images as expected by the network. While ImageNet labels were used for pretraining the Resnet50, no CIFAR-10 labels were used for the feature extraction or the UMAP dimension reduction. For this reason, we used the training and test set of CIFAR-10 jointly.



(a) 2D UMAP embedding

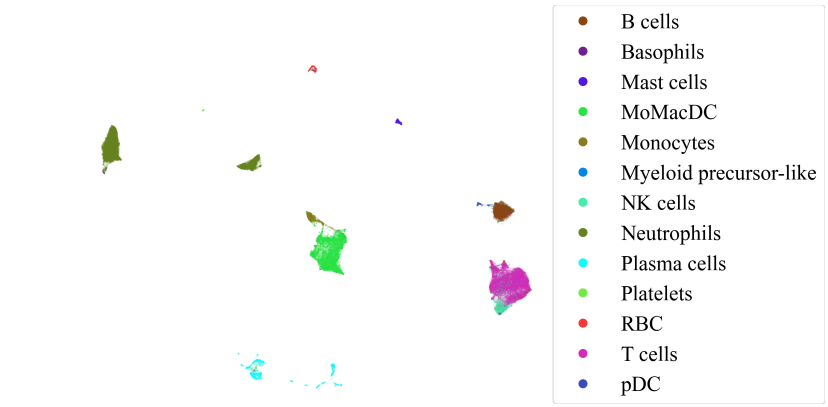


(b) loss curves

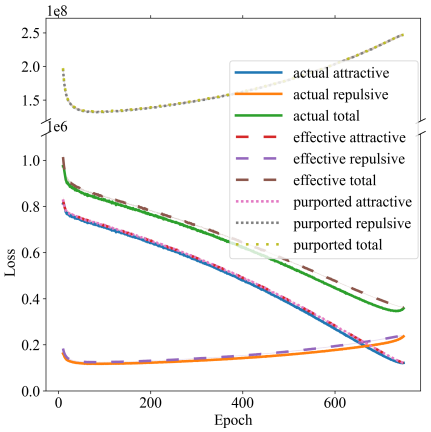


(c) Similarities for $\mu_{ij} > 0$

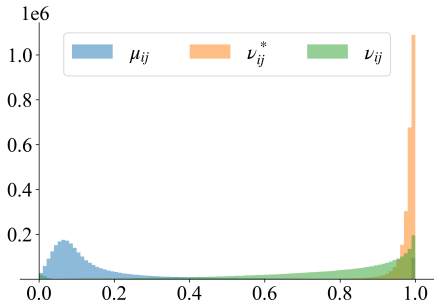
Figure C.4: UMAP on the PBMC dataset [179]. **C.4a:** The 2D UMAP embedding exhibits over-contraction for instance in the Dendritic and the B cells. **C.4b:** Loss curves for the optimization leading to Fig. C.4a. Similar to Fig. 3.4 our effective loss (3.12) closely matches the actual loss (3.14), while the purported UMAP loss (3.6) is two orders of magnitude higher. The total overlays the repulsive purported loss. An average over seven runs is plotted with shaded areas of one standard deviation. **C.4c:** Histogram of high-dimensional (μ_{ij}), target (ν^*_{ij}) and low-dimensional (ν_{ij}) similarities of the PBMC dataset for pairs with positive high-dimensional similarity. The similarities of UMAP's low-dimensional embedding reproduce the target similarities instead of the high-dimensional ones. The target similarities are heavily skewed towards one. Figure best viewed in color.



(a) 2D UMAP embedding

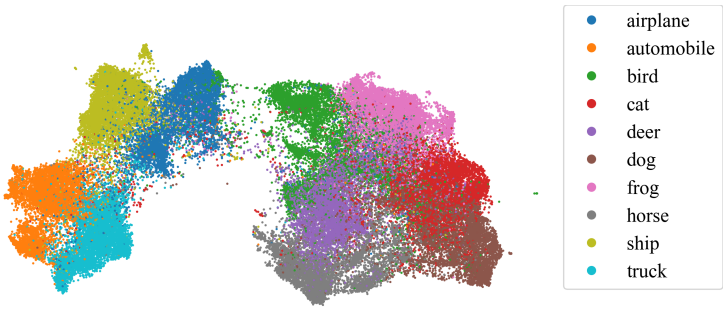


(b) loss curves

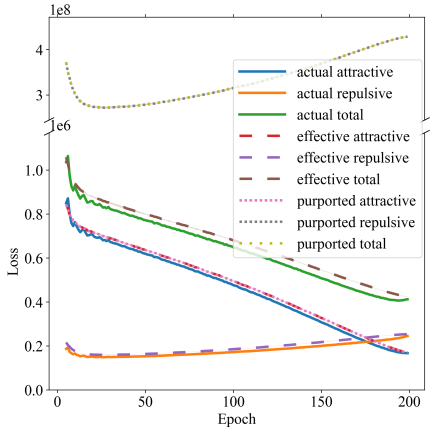


(c) Similarities for $\mu_{ij} > 0$

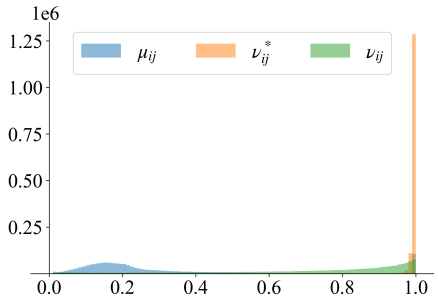
Figure C.5: UMAP on the lung cancer dataset [180]. **C.5a:** The 2D UMAP embedding exhibits over-contraction, for instance, in the plasma cells and the neutrophil cells near the MoMacDC cells. **C.5b:** Loss curves for the optimization leading to Fig. C.5a. Similar to Fig. 3.4 our effective loss (3.12) closely matches the actual loss (3.14), while the purported UMAP loss (3.6) is two orders of magnitude higher. The total overlays the repulsive purported loss. An average over seven runs is plotted with shaded areas of one standard deviation. **C.5c:** Histogram of high-dimensional (μ_{ij}), target (ν^*_{ij}) and low-dimensional (ν_{ij}) similarities of the lung cancer dataset for pairs with positive high-dimensional similarity. The similarities of UMAP's low-dimensional embedding reproduce the target similarities instead of the high-dimensional ones. The target similarities are heavily skewed towards one. Figure best viewed in color.



(a) 2D UMAP embedding



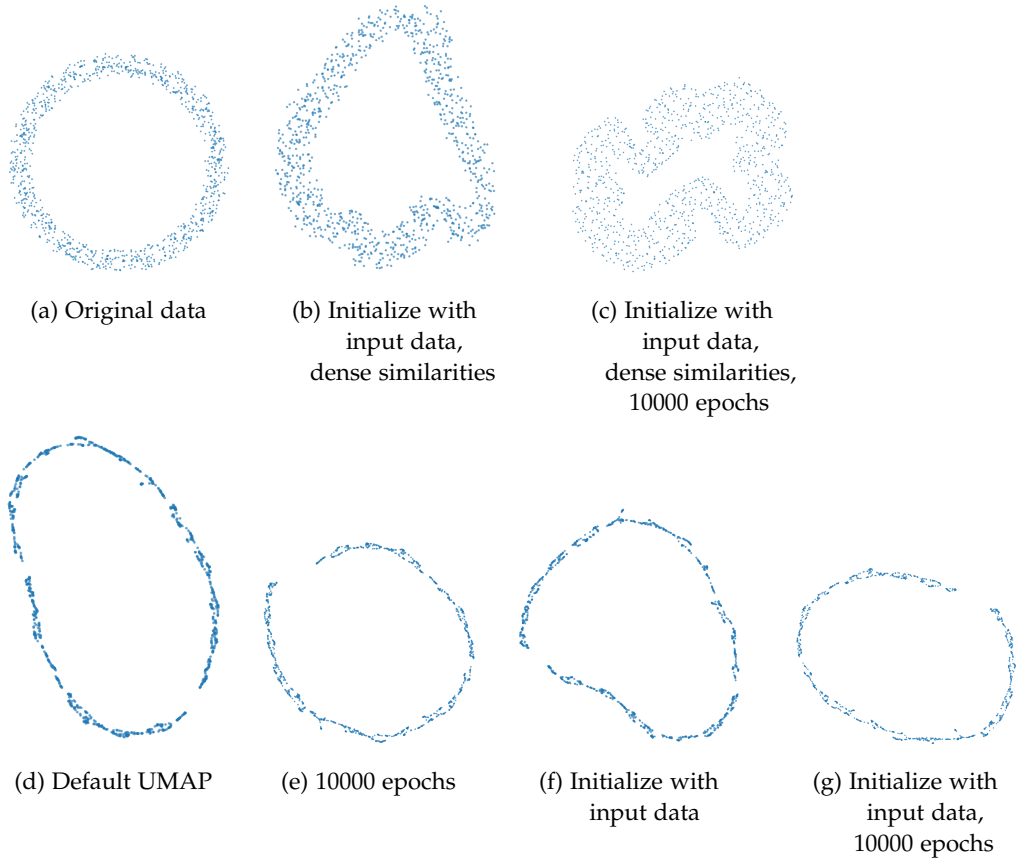
(b) loss curves



(c) Similarities for $\mu_{ij} > 0$

Figure C.6: UMAP on Resnet50 features of CIFAR-10. **C.6a:** The 2D UMAP embedding exhibits decent class separation, although the labels were only used to color the 2D plot. The pre-trained Resnet50 extracted semantically meaningful features that UMAP was able to embed well. We see a clear separation between the vehicle and animal classes, bordered by the bird and airplane classes. **C.6b:** Loss curves for the optimization leading to Fig. C.6a. Similar to Fig. 3.4 our effective loss (3.12) closely matches the actual loss (3.14), while the purported UMAP loss (3.6) is two orders of magnitude higher. The total overlays the repulsive purported loss. An average over seven runs is plotted with shaded areas of one standard deviation. **C.6c:** Histogram of high-dimensional (μ_{ij}), target (ν^*_{ij}) and low-dimensional (ν_{ij}) similarities of the CIFAR-10 dataset for pairs with positive high-dimensional similarity. The similarities of UMAP's low-dimensional embedding reproduce the target similarities instead of the high-dimensional ones. The target similarities are heavily skewed towards one. Figure best viewed in color.

C.4 ADDITIONAL FIGURES



(a) Original data

(b) Initialize with input data, dense similarities

(c) Initialize with input data, dense similarities, 10000 epochs

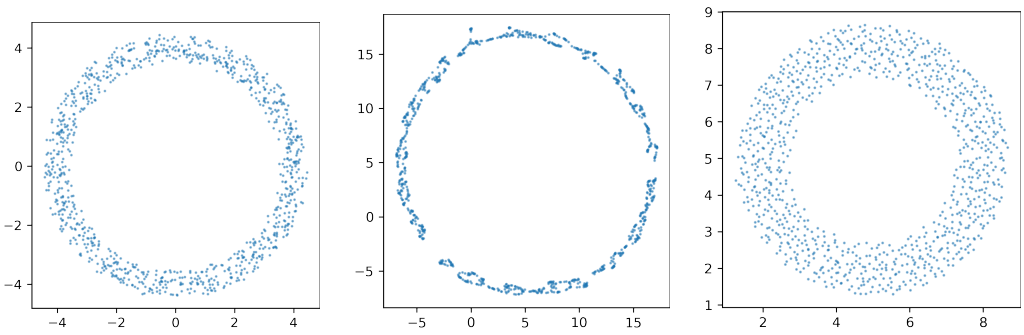
(d) Default UMAP

(e) 10000 epochs

(f) Initialize with input data

(g) Initialize with input data, 10000 epochs

Figure C.7: UMAP does not preserve the data even when embedding to the input dimension. Extension of Fig. 3.2. **C.7a:** Original data: 1000 uniform samples from a ring in 2D. **C.7b:** Result of UMAP when initialized with the original data and using dense input space similarities computed from the original data with ϕ. **C.7c:** Same as C.7b but optimized for 10000 epochs. **C.7d:** UMAP visualization with default hyperparameters. **C.7e:** Same as C.7d but optimized for 10000 epochs. **C.7f:** UMAP visualization initialized with the original data. **C.7g:** Same as C.7f but optimized for 10000 epochs.

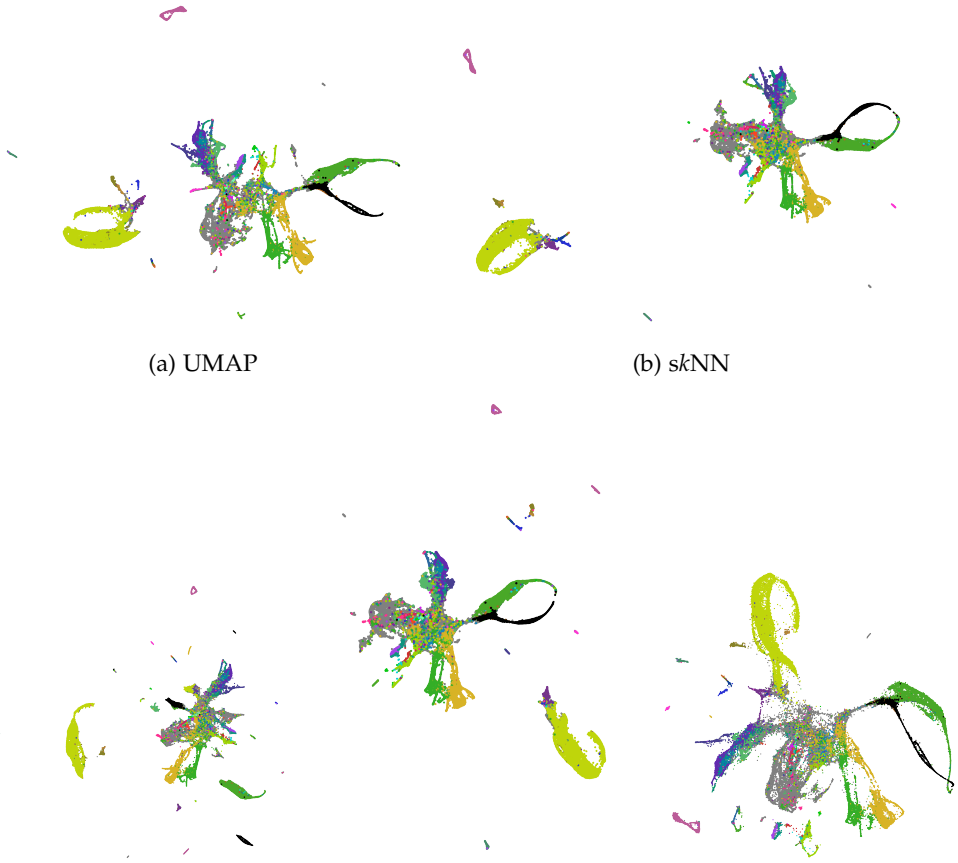


(a) Original data (b) UMAP (c) UMAP from dense similarities

Figure C.8: Similar to Fig. 3.2 but here, the tail of a negative sample is repelled from its head. **C.8a:** Original data. **C.8b:** The UMAP result looks similarly over-contracted but slightly rounder than 3.2b. **C.8c:** When initialized with the dense input similarities, the UMAP embedding has a wider than expected ring width similar to 3.2d but without the spurious curves. Instead, the radius of the ring is smaller than in the original. Both the larger ring width and the smaller radius match the analysis in Section 3.8.1.

(a) Original data (b) UMAP (c) UMAP from dense similarities

Figure C.9: UMAP does not preserve the data even when no dimension reduction is required. **C.9a:** Original data consisting of 1000 uniform samples from a unit square in 2D. **C.9b:** Result of UMAP after 10000 epochs, initialized with the original data. The embedding is much more clustered than the original data and, in many locations, nearly one-dimensional. **C.9c:** Result of UMAP after 10000 epochs for dense input space similarities computed from the original data with ϕ, initialized with the original embedding. Reproducing the input would be optimal for the purported UMAP loss in this setting. Instead, the output is circular with a slightly higher density in the middle. It appears even more regular than the original data.



(a) UMAP (b) s*k*NN

(c) Permuted (d) Uniformly random (e) Inverted

Figure C.10: The precise value of the positive μ_{ij}'s matters little: UMAP produces qualitatively similar results even for severely perturbed μ_{ij}. The panels depict UMAP visualizations of the C.elegans dataset but with disturbed positive high-dimensional similarities. **C.10a:** Usual UMAP μ_{ij}'s. **C.10b:** Positive μ_{ij} all set to one, so that the weights encode the s*k*NN graph as done in [15]. **C.10c:** Positive μ_{ij} randomly permuted. **C.10d:** Positive μ_{ij} overwritten by uniform random samples from $[0, 1]$. **C.10e:** Positive μ_{ij} filtered as in UMAP's optimization procedure (set all weights to zero below $\max \mu_{ij}/\text{n\_epochs}$) and inverted at the minimal positive value $\mu_{ij} = \min_{ab} \mu_{ab}/\mu_{ij}$. Amazingly, the visualizations still show the main structures identified by the unimpaired UMAP. While C.10c tears up the seam cells, C.10e even places the outliers conveniently compactly around the main structure. The level of visual similarity for different perturbations seems particularly high when compared to Fig. C.3 which shows the global placement of subgroups and whether loops are open or closed (e.g., seam cells and hypodermis cells) depend even on the random seed. We used random seed 0 in this figure. All C. elegans UMAP embeddings were subjectively flipped and rotated by multiples of $\pi/2$ to ease a visual comparison.

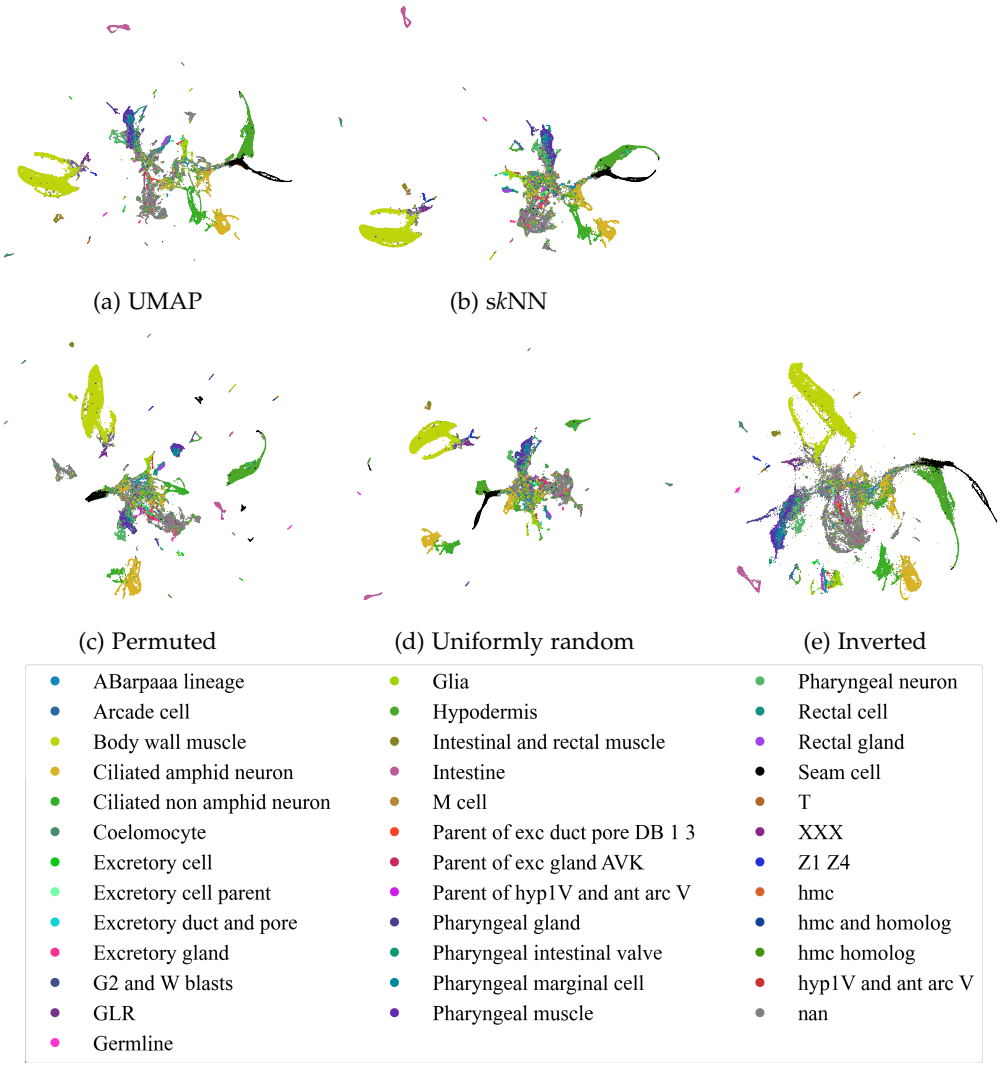


Figure C.11: Same as Fig. C.10 but here, the tail of a negative sample is repelled from its head. There is little qualitative difference between Fig. C.10 and this figure overall.

D

CONTRASTIVE LEARNING UNIFIES t-SNE AND UMAP

D.1 DATASETS

We use the well-known MNIST [93] dataset for most of our experiments. We downloaded it via the torchvision API from `http://yann.lecun.com/exdb/mnist/`. Unfortunately, this website does not give a license. However, `https://keras.io/api/datasets/mnist/` and `http://www.pymvpa.org/datadb/mnist.html` name Yann LeCun and Corinna Cortes as copyright holders and claim MNIST to be licensed under CC BY-SA 3.0, which permits use and adaptation. The MNIST dataset consists of 70 000 grayscale images, 28×28 pixels each, that show handwritten digits.

The Kuzushiji-49 dataset [155] was downloaded from `https://github.com/rois-codh/kmnist` where it is licensed under CC-BY-4.0. It contains 270 912 grayscale images, 28×28 pixels each, that show 49 different cursive Japanese characters.

The SimCLR experiments are performed on the CIFAR-10 [87] dataset, another standard machine learning resource. We downloaded it via `scikit-learn` [131]'s `sklearn.datasets.fetch_openml` API from `https://www.openml.org/search?type=data&sort=runs&id=40927&status=active`. Unfortunately, we were not able to find a license for this dataset. CIFAR-10 consists of 60 000 images, 32×32 RGB pixels each, depicting objects from five animal and five vehicle classes.

The transcriptomic dataset of Kanton *et al.* [74] was downloaded from `https://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-7552/`, which permits free use. We only use the 20 272 cells in the human brain organoid cell line '409b2'. The transcriptomic dataset [164] was downloaded in its scanpy version from `https://kleintools.hms.harvard.edu/paper_websites/wagner_zebrafish_timecourse2018/mainpage.html`. The full dataset at `https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE112294` is free to download and reproduce. The dataset contains gene expressions for 63 530 cells from a developing zebrafish embryo. The downloaded UMIs of both datasets are preprocessed as in [15, 83]. After selecting the 1000 most variable genes, we normalize the library sizes to the median library size in the dataset, log-transform the normalized values with $\log_2(x+1)$, and finally reduce the dimensionality to 50 via PCA.

The transcriptomic dataset of the C. elegans flatworm [116, 126] was obtained from `http://cb.csail.mit.edu/cb/densvis/datasets/` with the consent of the authors who license it under CC BY-NC 2.0. It is already preprocessed to 100 principal components.

D.2 IMPLEMENTATION

All contrastive embeddings are computed with our PyTorch [127] implementation of Neg-t-SNE, NCVis, UMAP, and InfoNC-t-SNE. Exceptions are Fig. 4.1, 4.2, and 4.13 as well as the analogous Figs. D.1 – D.4. There for panel h we use the reference implementation of NCVis [4] (with a fixed number of noise samples m, and not the

default schedule), and for panel g we use UMAP 0.5. The t-SNE plots are created with the openTSNE [132] (version 0.6.1) package. Similarly, we use the reference UMAP implementation in Fig. 4.7 and openTSNE in Figs. 4.9 and 4.10.

We extended these implementations of NCVis, UMAP, and t-SNE to make them accept custom embedding initializations and unweighted skNN graphs and to log various quantities of interest. We always use the standard Cauchy kernel for better comparability.

All PCAs are computed with sklearn [131]. We use PyKeOps [25] to compute the exact skNN graph and to handle the quadratic complexity of computing the partition functions on a GPU. The same PCA initialization and skNN graph with $k = 15$ are used for all embeddings. The skNN graph for MNIST is computed on a 50-dimensional PCA of the dataset.

When computing logarithms during the optimization of neighbor embeddings, we clip the arguments to the range $[10^{-10}, 1]$, save for Fig. 4.8, where we ablate this lower bound. The lower bound is smaller than in the reference implementation of parametric UMAP, where it is set to 10^{-4}.

Our defaults are a batch size of 1024, linear learning rate annealing from 1 (non-parametric) or 0.001 (parametric) to 0 (save for Figs. 4.6 – 4.8), 750 epochs (save for Figs. 4.3 and D.5 as well as Tab. D.3) and $m = 5$ noise samples (save for Figs. 4.3, 4.4, 4.9, 4.10, and D.6).

We initialize all embeddings with a scaled version of PCA (save for in Figs. 4.9, and 4.10). For t-SNE embeddings, we rescale the initialization so that the first dimension has a standard deviation of 0.0001 (as is the default in openTSNE), for all other embeddings to a standard deviation of 1.

We employ some version of 'early exaggeration' [161] for the first 250 epochs in most non-parametric plots. For t-SNE it is the default early exaggeration of openTSNE. When varying $\bar{Z}$ in non-parametric Neg-t-SNE, early exaggeration means using $\bar{Z} = |X|/m$ for the first 250 epochs (save for Fig. 4.2). When varying the number of noise samples in Figs. 4.4, 4.9 and 4.10, we still use $m = 5$ for the first 250 epochs. In Figs. 4.6, 4.7, 4.8, and 4.11 as well as in all reference NCVis or UMAP plots, we did not use early exaggeration as neither small $\bar{Z}$ nor high m made it necessary. When we use some form of early exaggeration and learning rate annealing, the annealing to zero takes place over the first 250 epochs, is then reset, and annealed again to zero for the remaining, typically 500, epochs.

Non-parametric runs are optimized with SGD without momentum, and parametric runs with the Adam optimizer [80]. Parametric runs use the same feed-forward neural net architecture as the reference parametric UMAP implementation. That is, four layers with dimensions $\mathit{input\ dimension} - 100 - 100 - 100 - 2$ with ReLU activations in all but the last one. We use the vectorized, 786-dimensional version of MNIST as input to the parametric neighbor embedding methods (and not the 50-dimensional PCA; but the skNN graph is computed in the PCA space for consistency with non-parametric embeddings).

Like the original NCVis implementation, we use the fractions $q_{\theta,Z}(x)/(q_{\theta,Z}(x) + m)$ instead of $q_{\theta,Z}(x)/\big(q_{\theta,Z}(x) + m\tilde{\xi}(x)\big)$. This is a mild approximation as the noise distribution is close to uniform. But it means that the model learns a scaled data distribution (cf. Cor. 4.3), so we need to multiply the learned normalization parameter Z by $n(n-1)$ when comparing to t-SNE or checking normalization of the NCVis model. Similarly, we also approximate the true noise distribution by

the uniform distribution for the fractions $q_\theta(x)/(q_\theta(x) + \bar{Z}m/|X|)$ – instead of $q_\theta(x)/\big(q_\theta(x) + \bar{Z}m\xi(x)\big)$ – in our Neg-t-SNE implementation.

We mentioned in Sec. 4.5 and show in Fig. 4.4 that one can move along the attraction-repulsion spectrum also by changing the number of noise samples m, instead of the fixed normalization constant $\bar{Z}$. UMAP's reference implementation has a scalar prefactor γ for the repulsive forces. Theoretically, adjusting γ should also move along the attraction-repulsion spectrum, but setting it higher than 1 led to convergence problems in [15], Fig. A11. We do not have such issues when varying our $\bar{Z}$.

For panels i in Figs. 4.1, 4.2, 4.13, and D.1 – D.4 the Neg-t-SNE spectra are computed for $\bar{Z}$ equal to $Z(\theta^{t\text{-SNE}})$, Z^{NCVis}, and $\frac{n(n-1)}{m} \cdot x$, where $x \in \{5 \cdot 10^{-5}, 1 \cdot 10^{-4}, 2 \cdot 10^{-4}, 5 \cdot 10^{-4}, \dots, 1 \cdot 10^{2}, 2 \cdot 10^{2}, 5 \cdot 10^{2}\}$.

For the SimCLR experiments, we train the model for 1000 epochs, of which we use 10 epochs for warmup. The learning rate during warmup is linearly interpolated from 0 to the initial learning rate. After the warmup epochs, we anneal the learning rate with a cosine schedule (without restarts) to 0 [105]. We optimize the model parameters with SGD and momentum 0.9. We use the same data augmentations as in [27]. In addition, we use a ResNet18 [65] as the backbone and a projection head consisting of two linear layers ($512 - 1024 - 128$) with a ReLU activation in-between. The loss is applied to the L_2-normalized output of the projection head, but like Chen *et al.* [27] we use the ResNet output as the representation for the linear evaluation. As the similarity function, we use the exponential of the normalized scalar product (cosine similarity) and always keep the temperature at 0.5, as suggested in Chen *et al.* [27]. When considering the entire batch as negative samples, we omit both the head and tail of the considered positive sample, while we only omit the head when sampling a smaller number of negative samples.

The ResNet is trained on the combined CIFAR-10 train and test sets. When training the classifier, we freeze the ResNet and only use the train set. The reported metrics are computed on the test set. We chose sklearn's KNearestNeighbors classifier with cosine metric and $k = 15$ neighbors and sklearn's SGDClassifier. For the linear classifier with augmentations, we follow Chen *et al.* [27] and randomly flip the images as well as randomly resize the images. The linear classifier is a simple linear layer that maps from 512 to the 10 classes and is trained via a cross-entropy loss. The classifier is trained for 100 epochs and an initial learning rate of 30 that is annealed with a cosine schedule to 0.

Our code is publicly available at `https://github.com/hci-unihd/cl-tsne-umap`.

D.2.1 *Stability*

Whenever we report a metric or show a graph, we ran the experiments for three different random seeds and report the mean $\pm$ the standard deviation. When the standard deviation is very small, we omit it from the main text and report it here. t-SNE does not depend on a random seed save for the usually approximate skNN graph computation. As we compute the skNN exactly with PyKeOps [25], t-SNE is deterministic in our framework. The normalization parameter Z^{NCVis} learned by NCVis and the values of the partition function for t-SNE can be found in Tab. D.1.

Table D.1: Learned normalization parameter for NCVis and partition function of t-SNE in our experiments. Mean and standard deviation is computed over three random seeds. In our setup, t-SNE is deterministic.

	Z^{NCVis} $[10^6]$	$Z(\theta^{t\text{-SNE}})$ $[10^6]$
MNIST Fig. 4.1	34.3 ± 0.1	8.13
MNIST without EE Fig. 4.2	34.3 ± 0.1	6.25
Human brain organoid Fig. 4.13	3.57 ± 0.03	1.30
MNIST imbalanced Fig. D.1	6.15 ± 0.06	3.12
Zebrafish Fig. D.2	30.8 ± 0.1	7.98
C. elegans Fig. D.3	36.9 ± 0.7	11.7
Kuzushiji-49 Fig. D.4	395 ± 3	89.6

Table D.2: Run time overview for the most compute-heavy experiments

	Runs	Time per run [min] (mean±std. dev.)
Neg-t-SNE for Figs. 4.1, 4.2, 4.13, D.2, D.3	360	39 ± 4
NCVis (our implementation) for Fig. 4.3b	3	786 ± 2
Neg-t-SNE for Fig. D.4	24	121 ± 44
Neg-t-SNE for Fig. D.5	3	3592 ± 44
SimCLR runs for Fig. D.6 and Tab. 4.1	21	694 ± 28

Panels i in Figs. 4.1, 4.2, 4.13, and D.1 – D.4 show the standard deviation as shaded area. Again, the standard deviations are very small and barely visible. The ratio of standard deviation to mean is never larger than 0.006 in these panels. Similarly, the standard deviation in Figs. 4.3 and 4.12, shown as shaded area, is mostly smaller than the line width.

D.2.2 *Compute*

We ran our neighbor embedding experiments on a machine with 56 Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz, 502 GB RAM and 10 NVIDIA TITAN Xp GPUs. The SimCLR experiments were conducted on a Slurm cluster node with 8 cores of an Intel(R) Xeon(R) Gold 5220 CPU @ 2.20GHz and an Nvidia V100 GPU with a RAM limit of 54.3 GB. Each experiment uses one GPU at most.

Our total compute is dominated by the neighbor embedding runs for Figs. 4.1, 4.2, 4.13, D.2 – D.5, and 4.3b and by the SimCLR experiments. Tab. D.2 lists the number of runs and the average run time. We thus estimate the total compute time to be about 750 hours. Our implementation uses pure PyTorch and relies on GPUs. However, for the non-parametric experiments, much of the computation consists of sampling negative neighbors and computationally light updates to the embedding positions. Therefore, the CPU-based numba or C++ reference implementations of

UMAP and NCVis are much faster. However, our implementation is arguably easier to inspect and adapt by the machine learning community and seamlessly integrates non-parametric and parametric settings as well as all four contrastive loss functions. Thus, it is more suited as a research tool than for a large throughput application.

As the change from NCVis to Neg-t-SNE is as simple as fixing the learnable normalization parameter to a constant, we have also adapted the original NCVis code to compute Neg-t-SNE. We have not used this for any of the experiments in the chapter.

D.3 ADDITIONAL FIGURES

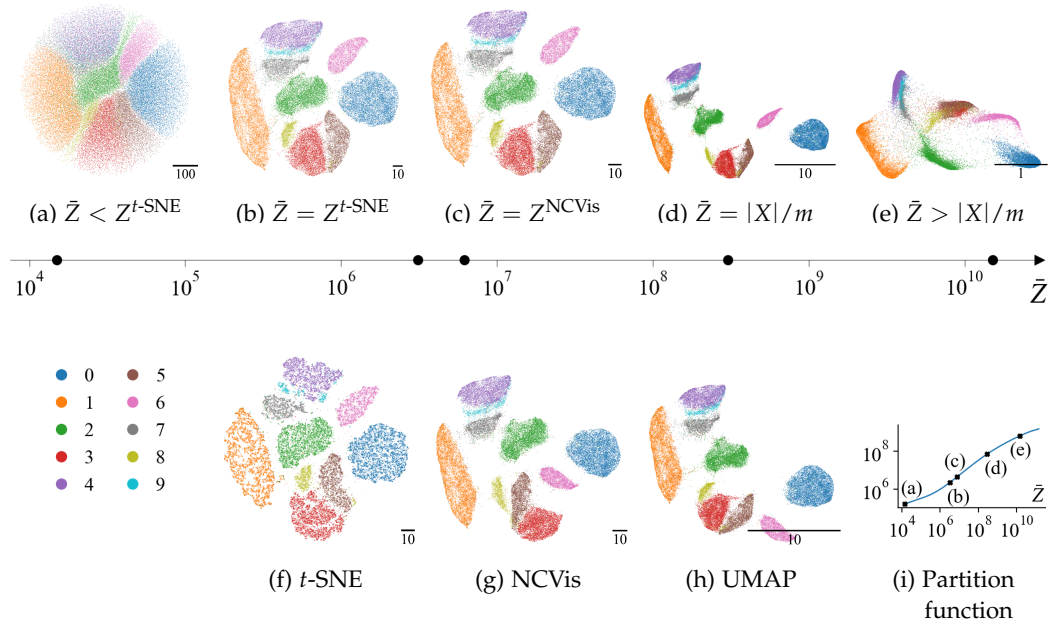


Figure D.1: **(a – e)** Neg-t-SNE embeddings of an imbalanced version of the MNIST dataset for various values of the fixed normalization constant $\bar{Z}$. As $\bar{Z}$ increases, the scale of the embedding decreases, and clusters become more compact and separated before eventually starting to merge. The Neg-t-SNE spectrum produces embeddings very similar to those of **(f)** t-SNE, **(g)** NCVis, and **(h)** UMAP, when $\bar{Z}$ equals the partition function of t-SNE, the learned normalization parameter Z of NCVis, or $|X|/m = \binom{n}{2}/m$ used by UMAP, as predicted in Sec. 4.4–4.6. **(i)** The partition function $\sum_{ij}(1 + d_{ij}^2)^{-1}$ tries to match $\bar{Z}$ and grows with it. Similar to early exaggeration in t-SNE we start all Neg-t-SNE runs using $\bar{Z} = |X|/m$ and only switch to the desired $\bar{Z}$ for the last two thirds of the optimization. The dataset is created by randomly removing $10 \cdot c\%$ of the class of digit c so that the class sizes linearly decrease from digit 0 to digit 9.

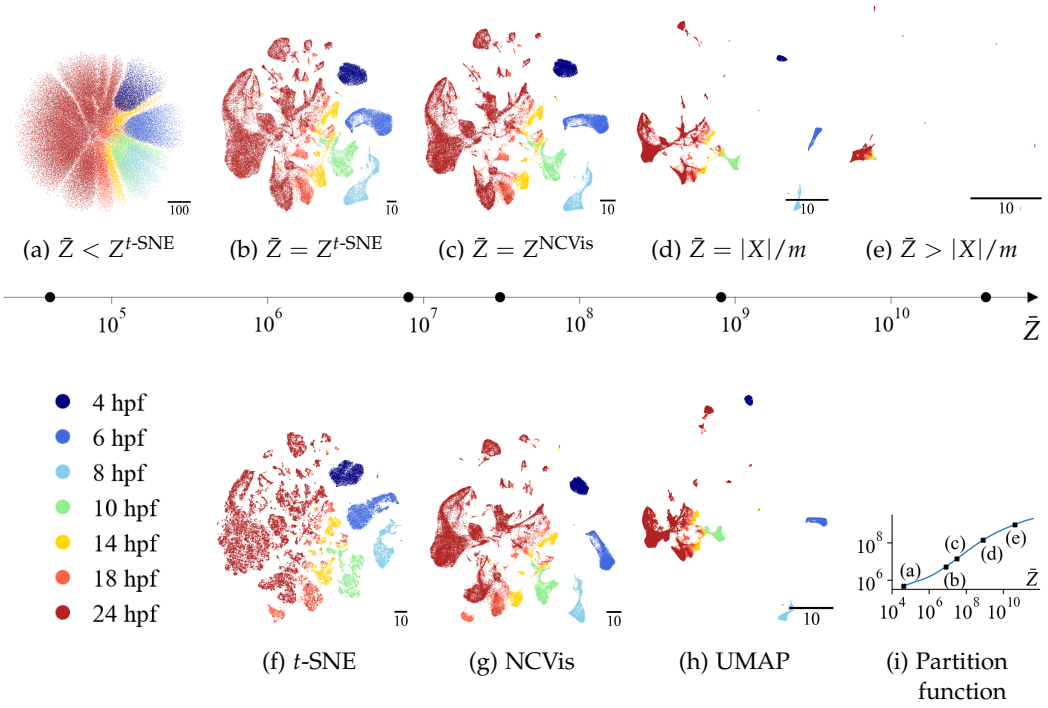


Figure D.2: **(a – e)** Neg-t-SNE spectrum on the single-cell RNA sequencing dataset of a developing zebrafish embryo [164] for various parameters $\bar{Z}$. As $\bar{Z}$ increases, the scale of the embedding decreases, and the continuous structure (corresponding to the developmental stage) becomes more apparent, making higher $\bar{Z}$ more suitable for visualizing continuous datasets [15]. The spectrum produces embeddings very similar to those of **(f)** t-SNE and **(g)** NCVis when $\bar{Z}$ equals the partition function of t-SNE or the learned normalization parameter of NCVis. The UMAP embedding in **(h)** closely resembles the Neg-t-SNE embedding at $\bar{Z} = |X|/m = \binom{n}{2}/m$. **(i)** The partition function $\sum_{ij}(1+d_{ij}^2)^{-1}$ of the Neg-t-SNE embeddings increases with $\bar{Z}$. Similar to early exaggeration in t-SNE we start all Neg-t-SNE runs using $\bar{Z} = |X|/m$ and only switch to the desired $\bar{Z}$ for the last two thirds of the optimization. The dataset contains 63 530 cells and is colored by the hours post fertilization (hpf). There are ten times fewer cells collected after 8 hours than after 24.

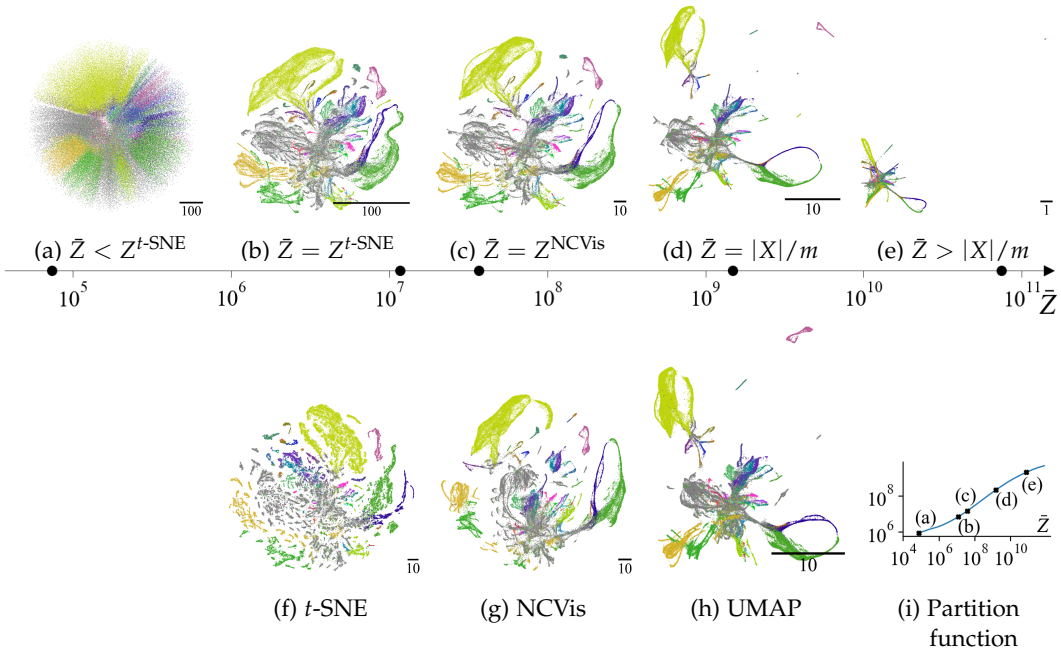


Figure D.3: **(a – e)** Neg-t-SNE spectrum of the single-cell RNA sequencing dataset of the C. elegans flatworm [116, 126] for various values of the fixed normalization constant $\bar{Z}$. As $\bar{Z}$ increases, the scale of the embedding decreases, and more continuous structure becomes apparent. The Neg-t-SNE spectrum produces embeddings very similar to those of **(f)** t-SNE, **(g)** NCVis, and **(h)** UMAP, when $\bar{Z}$ equals the partition function of t-SNE, the learned normalization parameter Z of NCVis, or $|X|/m = \binom{n}{2}/m$ used by UMAP, as predicted in Sec. 4.4–4.6. **(i)** The partition function $\sum_{ij}(1+d_{ij}^2)^{-1}$ tries to match $\bar{Z}$ and grows with it. Similar to early exaggeration in t-SNE we start all Neg-t-SNE runs using $\bar{Z} = |X|/m$ and only switch to the desired $\bar{Z}$ for the last two thirds of the optimization. The dataset contains information on 86 024 cells of 37 types indicated by the colors. It is imbalanced with only 25 cells of the least abundant type but 31 375 cells of unknown type (grey).

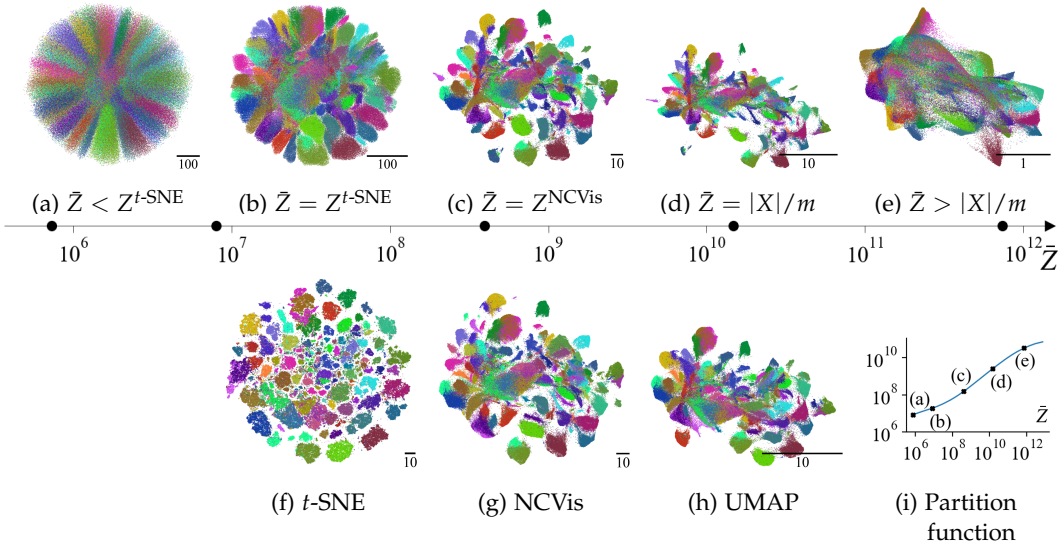


Figure D.4: **(a – e)** Neg-t-SNE embeddings of the Kuzushiji-49 dataset [155] for various values of the fixed normalization constant $\bar{Z}$. As $\bar{Z}$ increases, the scale of the embedding decreases, and clusters become more compact and separated before eventually starting to merge. The Neg-t-SNE spectrum produces embeddings similar to those of **(f)** t-SNE, **(g)** NCVis, and **(h)** UMAP, when $\bar{Z}$ equals the partition function of t-SNE, the learned normalization parameter Z of NCVis, or $|X|/m = \binom{n}{2}/m$ used by UMAP, as predicted in Sec. 4.4–4.6. **(i)** The partition function $\sum_{ij}(1 + d_{ij}^2)^{-1}$ tries to match $\bar{Z}$ and grows with it. Similar to early exaggeration in t-SNE we start all Neg-t-SNE runs using $\bar{Z} = |X|/m$ and only switch to the desired $\bar{Z}$ for the last two thirds of the optimization. The dataset contains 270 912 images of 49 different Japanese characters. The classes are imbalanced with 456 to 7 000 samples per class. We see that a higher level of repulsion than UMAP's $\bar{Z} = |X|/m$ helps to visualize the dataset's discrete structure. The sampling-based Neg-t-SNE embedding at $\bar{Z} = Z^{t\text{-SNE}}$ has less structure than the t-SNE embedding. Fig.D.5 and Tab. D.3 show that the Neg-t-SNE result improves for longer optimization.

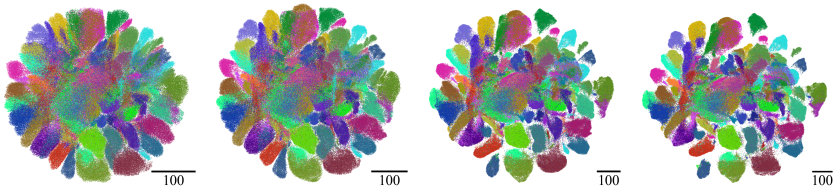


Figure D.5: Neg-t-SNE embeddings of the Kuzushiji-49 dataset [155] for $\bar{Z} = Z^{t\text{-SNE}}$ show more structure when optimized longer. Similar to early exaggeration in t-SNE we start all Neg-t-SNE runs using $\bar{Z} = |X|/m$ for 250 epochs and only switch to the desired $\bar{Z}$ for the remaining number of epochs indicated in the subcaptions.

Table D.3: Longer run times improve the Neg-t-SNE optimization on the Kuzushiji-49 dataset [155] for $\bar{Z} = Z^{t\text{-SNE}}$. The KL divergence is computed with respect to the normalized model $q_\theta / (\sum_{ij} q_\theta(ij))$.

Epochs	500	1000	5000	10000
Partition function $[10^6]$	18.96 ± 0.02	13.27 ± 0.01	6.93 ± 0.01	5.75 ± 0.01
Neg-t-SNE loss	1021 ± 2	832 ± 2	630 ± 1	599 ± 1
KL divergence	5.52 ± 0.01	5.18 ± 0.01	4.76 ± 0.01	4.65 ± 0.00

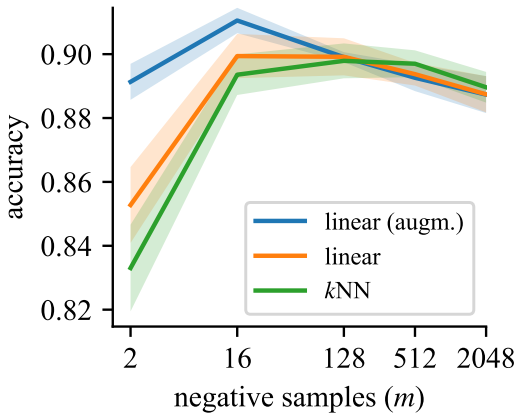


Figure D.6: SimCLR results on the CIFAR-10 dataset with InfoNCE loss and a varying number of negative samples m, extending the results of Tab. 4.1. The solid line and shaded area indicate the mean and standard deviation over three random seeds, respectively. For very few negative samples, the accuracies deteriorate. The linear classifier and the kNN classifier plateau for $m \geq 16$, while for the linear classifier trained with data augmentations, the peak performance is at $m = 16$.

PUBLICATIONS

This thesis is based on the following three publications/preprints:

[Chapter 2] Damrich, S., Remme, R., and Hamprecht, F. A., "Mod Shift: A Principled Alternative to Mean Shift Clustering Using Long-Range Repulsion," Preprint.

[Chapter 3] Damrich, S. and Hamprecht, F. A., "On UMAP's True Loss Function," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 5798–5809.

[Chapter 4] Damrich, S., Böhm, J. N., Hamprecht, F. A., and Kobak, D., "Contrastive learning unifies t-SNE and UMAP," *arXiv preprint arXiv: 2206.01816*, 2022.

The author further contributed to the following publications:

[a] Fita Sanmartín, E., Damrich, S., and Hamprecht, F. A., "Probabilistic Watershed: Sampling all spanning forests for seeded segmentation and semi-supervised learning," in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 2780–2791.

[b] ——, "Directed Probabilistic Watershed," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 20 076–20 088.

[c] ——, "The Algebraic Path Problem for Graph Metrics," in *Proceedings of the International Conference on Machine Learning*, PMLR, 2022, pp. 19 178–19 204.

[d] Garrido, Q., Damrich, S., Jäger, A., Cerletti, D., Claassen, M., Najman, L., and Hamprecht, F. A., "Visualizing hierarchies in scRNA-seq data using a density tree-biased autoencoder," *Bioinformatics*, vol. 38, no. Supplement 1, pp. i316–i324, 2022.

[e] Walter, F. C., Damrich, S., and Hamprecht, F. A., "Multistar: Instance Segmentation of Overlapping Objects with Star-Convex Polygons," in *IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, 2021, pp. 295–298.

[1] Amid, E. and Warmuth, M. K., "TriMap: Large-scale Dimensionality Reduction Using Triplets," *arXiv preprint arxiv: 1910.00204*, 2019.

[2] Andres, B., Kroeger, T., Briggman, K. L., Denk, W., Korogod, N., Knott, G., Koethe, U., and Hamprecht, F. A., "Globally Optimal Closed-surface Segmentation for Connectomics," in *Proceedings of the European Conference on Computer Vision*, Springer, 2012, pp. 778–791.

[3] Arganda-Carreras, I., Turaga, S. C., Berger, D. R., Cireşan, D., Giusti, A., Gambardella, L. M., Schmidhuber, J., Laptev, D., Dwivedi, S., Buhmann, J. M., *et al.*, "Crowdsourcing the creation of image segmentation algorithms for connectomics," *Frontiers in Neuroanatomy*, vol. 9, pp. 1–13, 2015, 142.

[4] Artemenkov, A. and Panov, M., "NCVis: Noise Contrastive Approach for Scalable Visualization," in *Proceedings of The Web Conference 2020*, 2020, pp. 2941–2947.

[5] Ash, J., Goel, S., Krishnamurthy, A., and Misra, D., "Investigating the Role of Negatives in Contrastive Representation Learning," in *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, 2022, pp. 7187–7209.

[6] Bachman, P., Hjelm, R. D., and Buchwalter, W., "Learning Representations by Maximizing Mutual Information across Views," in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 15 535–15 545.

[7] Baevski, A., Zhou, Y., Mohamed, A., and Auli, M., "Wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 12 449–12 460.

[8] Battista, G. D., Eades, P., Tamassia, R., and Tollis, I. G., *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR, 1998.

[9] Becht, E., McInnes, L., Healy, J., Dutertre, C.-A., Kwok, I. W., Ng, L. G., Ginhoux, F., and Newell, E. W., "Dimensionality reduction for visualizing single-cell data using UMAP," *Nature Biotechnology*, vol. 37, no. 1, pp. 38–44, 2019.

[10] Beier, T., Andres, B., Köthe, U., and Hamprecht, F. A., "An Efficient Fusion Move Algorithm for the Minimum Cost Lifted Multicut Problem," in *Proceedings of the European Conference on Computer Vision*, Springer, 2016, pp. 715–730.

[11] Belkin, M. and Niyogi, P., "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering," in *Advances in Neural Information Processing Systems*, vol. 14, 2002, pp. 585–591.

[12] Bell, E. T., "Exponential Numbers," *The American Mathematical Monthly*, vol. 41, no. 7, pp. 411–419, 1934.

[13] Bengio, Y., "From system 1 deep learning to system 2 deep learning," in *Advances in Neural Information Processing Systems*, Posner Lecture, 2019.

[14] Böhm, J. N., "Dimensionality reduction with neighborhood embeddings," M.S. thesis, University of Tübingen, 2020.

[15] Böhm, J. N., Berens, P., and Kobak, D., "Attraction-Repulsion Spectrum in Neighbor Embeddings," *Journal of Machine Learning Research*, vol. 23, no. 95, pp. 1–32, 2022.

[16] Bottou, L., Curtis, F. E., and Nocedal, J., "Optimization Methods for Large-Scale Machine Learning," *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.

[17] Bremner, J. G. and Wachs, T. D., *The Wiley-Blackwell Handbook of Infant Development, Volume 1: Basic Research*. John Wiley & Sons, 2011, vol. 1.

[18] CREMI, *Miccai challenge on circuit reconstruction from electron microscopy images*. https://cremi.org, 2016.

[19] Cao, J. *et al.*, "The single-cell transcriptional landscape of mammalian organogenesis," *Nature*, vol. 566, no. 7745, pp. 496–502, 2019.

[20] Cardona, A., Saalfeld, S., Preibisch, S., Schmid, B., Cheng, A., Pulokas, J., Tomancak, P., and Hartenstein, V., "An Integrated Micro-and Macroarchitectural Analysis of the Drosophila Brain by Computer-Assisted Serial Section Electron Microscopy," *PLOS Biology*, vol. 8, no. 10, pp. 1–17, 2010.

[21] Cardona, A., Saalfeld, S., Schindelin, J., Arganda-Carreras, I., Preibisch, S., Longair, M., Tomancak, P., Hartenstein, V., and Douglas, R. J., "TrakEM2 Software for Neural Circuit Reconstruction," *PLOS ONE*, vol. 7, no. 6, pp. 1–8, 2012.

[22] Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A., "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 9912–9924.

[23] Carreira-Perpiñán, M. Á., "Fast Nonparametric Clustering with Gaussian Blurring Mean-Shift," in *Proceedings of the International Conference on Machine Learning*, 2006, pp. 153–160.

[24] Chari, T., Banerjee, J., and Pachter, L., "The Specious Art of Single-Cell Genomics," *bioRxiv*, 2021.

[25] Charlier, B., Feydy, J., Glaunès, J. A., Collin, F.-D., and Durif, G., "Kernel Operations on the GPU, with Autodiff, without Memory Overflows," *Journal of Machine Learning Research*, vol. 22, no. 74, pp. 1–6, 2021.

[26] Chen, D., Lv, J., and Zhang, Y., "Unsupervised Multi-Manifold Clustering by Learning Deep Representation," in *Workshops at The Thirty-first AAAI Conference on Artificial Intelligence*, 2017, pp. 385–391.

[27] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G., "A Simple Framework for Contrastive Learning of Visual Representations," in *Proceedings of the International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, 2020, pp. 1597–1607.

[28] Chen, X., Liu, S., Sun, R., and Hong, M., "On the Convergence of a Class of Adam-Type Algorithms for Non-Convex Optimization," in *Proceedings of the International Conference on Learning Representations*, 2019, pp. 1–30.

[29] Chen, Y., Shen, C., Wei, X.-S., Liu, L., and Yang, J., "Adversarial PoseNet: A Structure-Aware Convolutional Network for Human Pose Estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1221–1230.

[30] Chen, Y., Pont-Tuset, J., Montes, A., and Van Gool, L., "Blazingly Fast Video Object Segmentation with Pixel-Wise Metric Learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1189–1198.

[31] Cheng, Y., "Mean Shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, 1995.

[32] Chopra, S. and Rao, M. R., "The partition problem," *Mathematical Programming*, vol. 59, no. 1-3, pp. 87–115, 1993.

[33] Coenen, A. and Pearce, A., *A deeper dive into UMAP theory*, https://pair-code.github.io/understanding-umap/supplement.html, Accessed: 2022-05-11, 2022.

[34] Comaniciu, D. and Meer, P., "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[35] Comaniciu, D., Ramesh, V., and Meer, P., "Real-Time Tracking of Non-Rigid Objects using Mean Shift," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, vol. 2, 2000, pp. 142–149.

[36] Corazza, P., "Introduction to Metric-Preserving Functions," *The American Mathematical Monthly*, vol. 106, no. 4, pp. 309–323, 1999.

[37] Cronin, T. W., Johnsen, S., Marshall, N. J., and Warrant, E. J., *Visual Ecology*. Princeton University Press, 2014.

[38] Damrich, S., Böhm, J. N., Hamprecht, F. A., and Kobak, D., "Contrastive learning unifies t-sne and umap," *arXiv preprint arxiv: 2206.01816*, 2022.

[39] Damrich, S. and Hamprecht, F. A., "On UMAP's True Loss Function," vol. 34, 2021, pp. 5798–5809.

[40] Damrich, S., Remme, R., and Hamprecht, F. A., "Mod shift: A principled alternative to mean shift clustering using long range repulsion," Preprint.

[41] Dasgupta, S. and Gupta, A., "An elementary proof of a theorem of johnson and lindenstrauss," *Random Structures & Algorithms*, vol. 22, no. 1, pp. 60–65, 2003.

[42] Dattorro, J., *Convex optimization & Euclidean distance geometry*. Lulu. com, 2010.

[43] Davis, E. and Sethuraman, S., "Consistency of modularity clustering on random geometric graphs," *Annals of Applied Probability*, vol. 28, no. 4, pp. 2003–2062, 2018.

[44] De Brabandere, B., Neven, D., and Van Gool, L., "Semantic Instance Segmentation with a Discriminative Loss Function," *arXiv preprint arxiv: 1708.02551*, 2017.

[45] Deza, M. M. and Laurent, M., *Geometry of Cuts and Metrics*, ser. Algorithms and Combinatorics. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, vol. 15.

[46] Deza, M., Grötschel, M., and Laurent, M., "Clique-Web Facets for Multicut Polytopes," *Mathematics of Operations Research*, vol. 17, no. 4, pp. 981–1000, 1992.

[47] Dong, W., Moses, C., and Li, K., "Efficient K-Nearest Neighbor Graph Construction for Generic Similarity Measures," in *Proceedings of the 20th International Conference on World Wide Web*, 2011, pp. 577–586.

[48] Dyer, C., "Notes on Noise Contrastive Estimation and Negative Sampling," *arXiv preprint arxiv: 1410.8251*, 2014.

[49] Eades, P., "A heuristic for graph drawing," *Congressus numerantium*, vol. 42, pp. 149–160, 1984.

[50] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X., "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, vol. 96, 1996, pp. 226–231.

[51] Fathi, A., Wojna, Z., Rathod, V., Wang, P., Song, H. O., Guadarrama, S., and Murphy, K. P., "Semantic Instance Segmentation via Deep Metric Learning," *arxiv: 1703.10277 [cs]*, 2017, arxiv: 1703.10277.

[52] França, G., Rizzo, M., and Vogelstein, J. T., "Kernel k-Groups via Hartigan's Method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[53] Fruchterman, T. M. and Reingold, E. M., "Graph Drawing by Force-directed Placement," *Software: Practice and experience*, vol. 21, no. 11, pp. 1129–1164, 1991.

[54] Fukunaga, K. and Hostetler, L., "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on Information Theory*, vol. 21, no. 1, pp. 32–40, 1975.

[55] Fukushima, K. and Miyake, S., "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition," in *Competition and Cooperation in Neural Nets*, Springer, 1982, pp. 267–285.

[56] Garcke, H., Preusser, T., Rumpf, M., Telea, A. C., Weikard, U., and Van Wijk, J. J., "A Phase Field Model for Continuous Clustering on Vector Fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 3, pp. 230–241, 2001.

[57] Goldberg, Y. and Levy, O., "Word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method," *arXiv preprint arxiv: 1402.3722*, 2014.

[58] Google Scholar, *Number of citations of "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction"*, https://scholar.google.com/scholar?hl=de&as_sdt=2005&sciodt=0%2C5&cites=13168625861022180360&scipsc=&as_ylo=2021&as_yhi=2021, Accessed: 2022-08-17.

[59] ——, *Number of citations of "visualizing data using t-SNE."* https://scholar.google.com/scholar?hl=de&as_sdt=2005&sciodt=0%2C5&cites=487516874590466294&scipsc=&as_ylo=2021&as_yhi=2021, Accessed: 2022-08-17.

[60] Goyal, A. and Bengio, Y., "Inductive Biases for Deep Learning of Higher-Level Cognition," *arXiv preprint arxiv: 2011.15091*, 2020.

[61] Gutmann, M. U. and Hyvärinen, A., "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 297–304.

[62] ——, "Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics.," *Journal of Machine Learning Research*, vol. 13, no. 2, 2012.

[63] Hadsell, R., Chopra, S., and LeCun, Y., "Dimensionality Reduction by Learning an Invariant Mapping," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 1735–1742, 2006.

[64] He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R., "Momentum Contrast for Unsupervised Visual Representation Learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9729–9738.

[65] He, K., Zhang, X., Ren, S., and Sun, J., "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[66] Hinton, G. E. and Roweis, S., "Stochastic Neighbor Embedding," in *Advances in Neural Information Processing Systems*, vol. 15, 2002, pp. 857–864.

[67] Horňáková, A., Lange, J.-H., and Andres, B., "Analysis and Optimization of Graph Decompositions by Lifted Multicuts," in *Proceedings of the International Conference on Machine Learning*, JMLR. org, vol. 70, 2017, pp. 1539–1548.

[68] Huang, P., Huang, Y., Wang, W., and Wang, L., "Deep Embedding Network for Clustering," in *2014 22nd International Conference on Pattern Recognition*, IEEE, 2014, pp. 1532–1537.

[69] Jacomy, M., Venturini, T., Heymann, S., and Bastian, M., "ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software," *PLOS ONE*, vol. 9, no. 6, pp. 1–12, 2014.

[70] Ji, X., Henriques, J. F., and Vedaldi, A., "Invariant Information Clustering for Unsupervised Image Classification and Segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9865–9874.

[71] Johnson, W. B. and Lindenstrauss, J., "Extensions of Lipschitz mappings into a Hilbert space," *Contemporary Mathematics*, vol. 26, pp. 189–206, 1984.

[72] Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y., "Exploring the Limits of Language Modeling," *arXiv preprint arxiv: 1602.02410*, 2016.

[73] Kahneman, D., *Thinking, fast and slow*. Macmillan, 2011.

[74] Kanton, S., Boyle, M. J., He, Z., Santel, M., Weigert, A., Sanchís-Calleja, F., Guijarro, P., Sidow, L., Fleck, J. S., Han, D., *et al.*, "Organoid single-cell genomic atlas uncovers human-specific features of brain development," *Nature*, vol. 574, no. 7778, pp. 418–422, 2019.

[75] Kappes, J. H., Speth, M., Andres, B., Reinelt, G., and Schnörr, C., "Globally Optimal Image Partitioning by Multicuts," in *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, Springer, 2011, pp. 31–44.

[76] Kappes, J. H., Speth, M., Reinelt, G., and Schnörr, C., "Higher-order segmentation via multicuts," *Computer Vision and Image Understanding*, vol. 143, pp. 104–119, 2016.

[77] Kaufmann, M. and Wagner, D., *Drawing Graphs: Methods and Models*, ser. Lecture Notes in Computer Science. Springer, 2003, vol. 2025.

[78] Keuper, M., Levinkov, E., Bonneel, N., Lavoué, G., Brox, T., and Andres, B., "Efficient Decomposition of Image and Mesh Graphs by Lifted Multicuts," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1751–1759.

[79] Kim, S., Yoo, C. D., Nowozin, S., and Kohli, P., "Image Segmentation Using Higher-Order Correlation Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 9, pp. 1761–1774, 2014.

[80] Kingma, D. P. and Ba, J., "Adam: A Method for Stochastic Optimization," in *Proceedings of the International Conference on Learning Representations*, 2015, pp. 1–15.

[81] Kirillov, A., Levinkov, E., Andres, B., Savchynskyy, B., and Rother, C., "InstanceCut: From Edges to Instances with MultiCut," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5008–5017.

[82] Kobak, D, *What are 2D neighbour embeddings of scRNA-seq data actually useful for?* Panel Discussion A: Data-Driven Manifold Learning at the ICLR Workshop on Geometrical and Topological Representation Learning, Apr. 2022.

[83] Kobak, D. and Berens, P., "The art of using t-SNE for single-cell transcriptomics," *Nature Communications*, vol. 10, no. 1, pp. 1–14, 2019.

[84] Kobak, D. and Linderman, G. C., "Initialization is critical for preserving global data structure in both t-SNE and UMAP," *Nature Biotechnology*, pp. 1–2, 2021.

[85] Kobak, D., Linderman, G., Steinerberger, S., Kluger, Y., and Berens, P., "Heavy-Tailed Kernels Reveal a Finer Cluster Structure in t-SNE Visualisations," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2019, pp. 124–139.

[86] Kong, S. and Fowlkes, C., "Recurrent Pixel Embedding for Instance Grouping," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9018–9028.

[87] Krizhevsky, A., "Learning multiple layers of features from tiny images," M.S. thesis, University of Toronto, 2009.

[88] Kroeger, T., Kappes, J. H., Beier, T., Koethe, U., and Hamprecht, F. A., "Asymmetric Cuts: Joint Image Labeling and Partitioning," in *German Conference on Pattern Recognition*, Springer, 2014, pp. 199–211.

[89] Lambert, P., Verleysen, M., and Lee, J. A., "SQuadMDS: A lean Stochastic Quartet MDS improving global structure preservation in neighbor embedding like t-SNE and UMAP," *Neurocomputing*, pp. 17–27, 2022.

[90] Lause, J., Berens, P., and Kobak, D., "Analytic Pearson residuals for normalization of single-cell RNA-seq UMI data," *Genome Biology*, vol. 22, no. 1, pp. 1–20, 2021.

[91] Le-Khac, P. H., Healy, G., and Smeaton, A. F., "Contrastive Representation Learning: A Framework and Review," *IEEE Access*, vol. 8, pp. 193 907–193 934, 2020.

[92] LeCun, Y, *The future is self-supervised*, https://iclr.cc/virtual_2020/speaker_7.html, Reflections from the Turing Award Winners at ICLR 2020, talk, Apr. 2020.

[93] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[94] Lee, J. A. and Verleysen, M., "Quality assessment of dimensionality reduction: Rank-based criteria," *Neurocomputing*, vol. 72, no. 7-9, pp. 1431–1443, 2009.

[95] Lee, K., Lu, R., Luther, K., and Seung, H. S., "Learning Dense Voxel Embeddings for 3D Neuron Reconstruction," *arXiv preprint arxiv: 1909.09872*, 2019.

[96] Lee, K., Zung, J., Li, P., Jain, V., and Seung, H. S., "Superhuman Accuracy on the SNEMI3D Connectomics Challenge," *arXiv preprint arxiv: 1706.00120*, 2017.

[97] Levy, O. and Goldberg, Y., "Neural Word Embedding as Implicit Matrix Factorization," in *Advances in Neural Information Processing Systems*, vol. 27, 2014, pp. 2177–2185.

[98] Li, X. and Wang, C.-Y., "From bulk, single-cell to spatial RNA sequencing," *International Journal of Oral Science*, vol. 13, no. 1, pp. 1–6, 2021.

[99] Linde, Y., Buzo, A., and Gray, R., "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84–95, 1980.

[100] Lindeberg, T., "Scale-Space for Discrete Signals," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 3, pp. 234–254, 1990.

[101] Linderman, G. C., Rachh, M., Hoskins, J. G., Steinerberger, S., and Kluger, Y., "Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data," *Nature Methods*, vol. 16, no. 3, pp. 243–245, 2019.

[102] Linderman, G. C. and Steinerberger, S., "Clustering with t-SNE, Provably," *SIAM Journal on Mathematics of Data Science*, vol. 1, no. 2, pp. 313–332, 2019.

[103] Lindsay, G. W., "Attention in Psychology, Neuroscience, and Machine Learning," *Frontiers in Computational Neuroscience*, vol. 14, pp. 1–21, 2020.

[104] Lloyd, S., "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[105] Loshchilov, I. and Hutter, F., "SGDR: Stochastic Gradient Descent with Warm Restarts," *Proceedings of the International Conference on Learning Representations*, pp. 1–16, 2017.

[106] Luecken, M. D. and Theis, F. J., "Current best practices in single-cell RNA-seq analysis: A tutorial," *Molecular Systems Biology*, vol. 15, no. 6, pp. 1–23, 2019.

[107] Luther, K. and Seung, H. S., "Learning Metric Graphs for Neuron Segmentation in Electron Microscopy Images," *arXiv preprint arxiv: 1902.00100*, 2019.

[108] Ma, Z. and Collins, M., "Noise Contrastive Estimation and Negative Sampling for Conditional Models: Consistency and Statistical Efficiency," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3698–3707.

[109] Max, J., "Quantizing for Minimum Distortion," *IRE Transactions on Information Theory*, vol. 6, no. 1, pp. 7–12, 1960.

[110] McCulloch, W. S. and Pitts, W., "A Logical Calculus of the Ideas Immanent in Nervous Activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[111] McInnes, L., Healy, J., and Astels, S., "Hdbscan: Hierarchical density based clustering.," *The Journal of Open Source Software*, vol. 2, no. 11, pp. 1–2, 2017.

[112] McInnes, L., Healy, J., and Melville, J., "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," *arXiv preprint arxiv: 1802.03426*, 2018.

[113] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J., "Distributed Representations of Words and Phrases and their Compositionality," in *Advances in Neural Information Processing Systems*, vol. 26, 2013, pp. 3111–3119.

[114] Minsky, M. and Papert, S. A., *Perceptrons: An introduction to computational geometry*. MIT press, 2017.

[115] Mitrovic, J., McWilliams, B., and Rey, M., "Less can be more in contrastive learning," in *Proceedings on "I Can't Believe It's Not Better!" at NeurIPS Workshops*, 2020, pp. 70–75.

[116] Narayan, A., Berger, B., and Cho, H., "Assessing single-cell transcriptomic variability through density-preserving data visualization," *Nature Biotechnology*, pp. 1–10, 2021.

[117] Neven, D., Brabandere, B. D., Proesmans, M., and Gool, L. V., "Instance Segmentation by Jointly Optimizing Spatial Embeddings and Clustering Bandwidth," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8837–8845.

[118] Newell, A., Huang, Z., and Deng, J., "Associative Embedding: End-to-End Learning for Joint Detection and Grouping," in *Advances in Neural Information Processing Systems*, 2017, pp. 2277–2287.

[119] Newman, M. E., "Analysis of weighted networks," *Physical Review E*, vol. 70, no. 5, p. 056 131, 2004.

[120] Newman, M. E. and Girvan, M., "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, pp. 1–15, 2004.

[121] Niculescu, C. and Persson, L.-E., *Convex Functions and their Applications*. Springer, 2006.

[122] Noack, A., "Modularity clustering is force-directed layout," *Physical Review E*, vol. 79, no. 2, pp. 1–9, 2009.

[123] Nozawa, K. and Sato, I., "Understanding Negative Samples in Instance Discriminative Self-supervised Representation Learning," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 5784–5797.

[124] Oord, A. Van den, Li, Y., and Vinyals, O., "Representation Learning with Contrastive Predictive Coding," *arXiv e-prints*, arXiv–1807, 2018.

[125] Oskolkov, N., *How Exactly UMAP Works*, https://towardsdatascience.com/how-exactly-umap-works-13e3040e1668, Accessed: 2022-05-11, 2022.

[126] Packer, J. S. *et al.*, "A lineage-resolved molecular atlas of C. elegans embryogenesis at single-cell resolution," *Science*, vol. 365, no. 6459, pp. 1265–1274, 2019.

[127] Paszke, A. *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 8024–8035.

[128] Pauen, S. and Rauh, H., "Frühe Kindheit," *Enzyklopädie der Psychologie*, pp. 67–126, 2008.

[129] Payer, C., Štern, D., Neff, T., Bischof, H., and Urschler, M., "Instance segmentation and tracking with cosine embeddings and recurrent hourglass networks," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2018, pp. 3–11.

[130] Pearson, K., "On Lines and Planes of Closest Fit to Systems of Points in Space," *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, vol. 2, no. 11, pp. 559–572, 1901.

[131] Pedregosa, F. *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[132] Poličar, P. G., Stražar, M., and Zupan, B., "OpenTSNE: A modular Python library for t-SNE dimensionality reduction and embedding," *bioRxiv*, 2019.

[133] Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., Wang, K., and Tang, J., "GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1150–1160.

[134] Rao, S., Medeiros Martins, A. de, and Príncipe, J. C., "Mean shift: An information theoretic perspective," *Pattern Recognition Letters*, vol. 30, pp. 222–230, 2009.

[135] Rebuffi, S.-A., Ehrhardt, S., Han, K., Vedaldi, A., and Zisserman, A., "LSD-C: Linearly Separable Deep Clusters," *arXiv preprint arxiv: 2006.10039*, 2020.

[136] Rosenblatt, F., "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[137] Roth, K., Milbich, T., and Ommer, B., "PADS: Policy-Adapted Sampling for Visual Similarity Learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6568–6577.

[138] Rousseeuw, P. J., "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Msathematics*, vol. 20, pp. 53–65, 1987.

[139] Roweis, S. T. and Saul, L. K., "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[140] Ruder, S., *On word embeddings - Part 2: Approximating the Softmax*, http://ruder.io/word-embeddings-softmax, Accessed: 2022-05-17, 2016.

[141] Sainburg, T., McInnes, L., and Gentner, T. Q., "Parametric UMAP Embeddings for Representation and Semisupervised Learning," *Neural Computation*, vol. 33, no. 11, pp. 2881–2907, 2021.

[142] Salamon, A. Z., "Streaming bounds from difference ramification.," in *Electronic Colloquium on Computational Complexity*, vol. 19, 2012, pp. 1–14.

[143] Sandberg, R., "Entering the era of single-cell transcriptomics in biology and medicine," *Nature methods*, vol. 11, no. 1, pp. 22–24, 2014.

[144] Scala, F., Kobak, D., Bernabucci, M., Bernaerts, Y., Cadwell, C. R., Castro, J. R., Hartmanis, L., Jiang, X., Laturnus, S., Miranda, E., *et al.*, "Phenotypic variation of transcriptomic cell types in mouse motor cortex," *Nature*, vol. 598, no. 7879, pp. 144–150, 2021.

[145] Schoenberg, I. J., "Remarks to Maurice Frechet's Article "Sur La Definition Axiomatique D'Une Classe D'Espace Distances Vectoriellement Applicable Sur L'Espace De Hilbert"," *Annals of Mathematics*, pp. 724–732, 1935.

[146] ——, "Metric spaces and completely monotone functions," *Annals of Mathematics*, pp. 811–841, 1938.

[147] Serway, R. A. and Faughn, J. S., *Holt Physics*. Austin, TX: Holt, Rinehart, and Winston, 2006.

[148] Shah, S. A. and Koltun, V., "Robust continuous clustering," *Proceedings of the National Academy of Sciences*, vol. 114, no. 37, pp. 9814–9819, 2017.

[149] ——, "Deep Continuous Clustering," *arXiv preprint arxiv: 1803.01449*, 2018.

[150] Sohn, K., "Improved Deep Metric Learning with Multi-class N-pair Loss Objective," in *Advances in Neural Information Processing Systems*, vol. 29, 2016, pp. 1857–1865.

[151] Song, J., Andres, B., Black, M., Hilliges, O., and Tang, S., "End-to-end Learning for Graph Decomposition," *arXiv preprint arxiv: 1812.09737*, 2018.

[152] Szubert, B., Cole, J. E., Monaco, C., and Drozdov, I., "Structure-preserving visualisation of high dimensional single-cell datasets," *Scientific reports*, vol. 9, no. 1, pp. 1–10, 2019.

[153] Tang, J., Liu, J., Zhang, M., and Mei, Q., "Visualizing large-scale and high-dimensional data," in *Proceedings of the 25th International Conference on World Wide Web*, 2016, pp. 287–297.

[154] Tang, S., Andriluka, M., Andres, B., and Schiele, B., "Multiple People Tracking by Lifted Multicut and Person Re-Identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3539–3548.

[155] Tarin, C., Mikel, B, Asanobu, K, Alex, L, Kazuaki, Y, and David, H, "Deep Learning for Classical Japanese Literature," in *Proceedings of 2018 Workshop on Machine Learning for Creativity and Design (Thirty-second Conference on Neural Information Processing Systems)*, vol. 3, 2018, pp. 1–8.

[156] Tenenbaum, J. B., De Silva, V., and Langford, J. C., "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[157] Tian, Y., Krishnan, D., and Isola, P., "Contrastive Multiview Coding," in *Proceedings of the European Conference on Computer Vision*, Springer, 2020, pp. 776–794.

[158] Torgerson, W. S., "Multidimensional scaling: I. theory and method," *Psychometrika*, vol. 17, no. 4, pp. 401–419, 1952.

[159] van der Maaten, L., "Learning a Parametric Embedding by Preserving Local Structure," in *Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, 2009, pp. 384–391.

[160] ——, "Accelerating t-SNE using Tree-Based Algorithms," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3221–3245, 2014.

[161] van der Maaten, L. and Hinton, G., "Visualizing data using t-SNE.," *Journal of Machine Learning Research*, vol. 9, no. 11, pp. 2579–2605, 2008.

[162] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I., "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 6000–6010.

[163] Verma, V., Luong, T., Kawaguchi, K., Pham, H., and Le, Q., "Towards Domain-Agnostic Contrastive Learning," in *Proceedings of the International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, 2021, pp. 10 530–10 541.

[164] Wagner, D. E., Weinreb, C., Collins, Z. M., Briggs, J. A., Megason, S. G., and Klein, A. M., "Single-cell mapping of gene expression landscapes and lineage in the zebrafish embryo," *Science*, vol. 360, no. 6392, pp. 981–987, 2018.

[165] Wang, Y., Huang, H., Rudin, C., and Shaposhnik, Y., "Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMAP, and PaCMAP for Data Visualization," *Journal of Machine Learning Research*, vol. 22, pp. 1–73, 2021.

[166] Weber, A., "Über den Standort der Industrien, Teil I: Reine Theorie des Standorts," *JCB Mohr, Tübingen,(English ed. by CJ Friedrichs, Univ. Chicago Press, 1929)*, 1909.

[167] Weinberger, K. and Saul, L., "Unsupervised Learning of Image Manifolds by Semidefinite Programming," in *Proceedings of the International Journal of Computer Vision*, vol. 70, 2006, pp. 77–90.

[168] Whittaker, E., "Eddington's Theory of the Constants of Nature," *The Mathematical Gazette*, vol. 29, no. 286, pp. 137–144, 1945.

[169] Wolf, S., Bailoni, A., Pape, C., Rahaman, N., Kreshuk, A., Köthe, U., and Hamprecht, F. A., "The Mutex Watershed and its Objective: Efficient, Parameter-Free Graph Partitioning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3724–3738, 2021.

[170] Wolf, S., Pape, C., Bailoni, A., Rahaman, N., Kreshuk, A., Kothe, U., and Hamprecht, F. A., "The Mutex Watershed: Efficient, Parameter-Free Image Partitioning," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 546–562.

[171] Wu, C.-Y., Manmatha, R, Smola, A. J., and Krahenbuhl, P., "Sampling Matters in Deep Embedding Learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2840–2848.

[172] Wu, Z., Xiong, Y., Yu, S. X., and Lin, D., "Unsupervised Feature Learning via Non-Parametric Instance Discrimination," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3733–3742.

[173] Xie, J., Girshick, R., and Farhadi, A., "Unsupervised Deep Embedding for Clustering Analysis," in *Proceedings of the International Conference on Machine Learning*, 2016, pp. 478–487.

[174] Yang, Z., King, I., Xu, Z., and Oja, E., "Heavy-Tailed Symmetric Stochastic Neighbor Embedding," in *Advances in Neural Information Processing Systems*, 2009, pp. 2169–2177.

[175] Yarkony, J., "Analyzing PlanarCC: Demonstrating the Equivalence of PlanarCC and the Multi-Cut LP Relaxation," in *NIPS Workshop on Discrete Optimization*, vol. 4, 2014, pp. 1–6.

[176] Yarkony, J., Ihler, A., and Fowlkes, C. C., "Fast Planar Correlation Clustering for Image Segmentation," in *Proceedings of the European Conference on Computer Vision*, Springer, 2012, pp. 568–581.

[177] Zachary, W. W., "An Information Flow Model for Conflict and Fission in Small Groups," *Journal of Anthropological Research*, vol. 33, no. 4, pp. 452–473, 1977.

[178] Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D., "Mixup: Beyond Empirical Risk Minimization," in *Proceedings of the International Conference on Learning Representations*, 2018, pp. 1–13.

[179] Zheng, G. X., Terry, J. M., Belgrader, P., Ryvkin, P., Bent, Z. W., Wilson, R., Ziraldo, S. B., Wheeler, T. D., McDermott, G. P., Zhu, J., *et al.*, "Massively parallel digital transcriptional profiling of single cells," *Nature Communications*, vol. 8, no. 1, pp. 1–12, 2017.

[180] Zilionis, R., Engblom, C., Pfirschke, C., Savova, V., Zemmour, D., Saatcioglu, H. D., Krishnan, I., Maroni, G., Meyerovitz, C. V., Kerwin, C. M., *et al.*, "Single-cell transcriptomics of human and mouse lung cancers reveals conserved myeloid populations across individuals and species," *Immunity*, vol. 50, no. 5, pp. 1317–1334, 2019.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*. `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Final Version as of August 21, 2022 (classicthesis v4.6).