A Self-Organizing UMAP for Clustering

Josh Taylor(✉) and Stella Offner

The University of Texas at Austin, Austin, TX, USA
joshtaylor@utexas.edu

Abstract. We introduce SOUMAP (Self-Organizing Uniform Manifold Approximation and Projection) as an alternative to regular Self-Organizing Map (SOM) learning intended to improve clustering of the resulting output space. Improvement is achieved by abandoning the SOM's rigid lattice structure in favor of a more expressive output topology afforded by Uniform Manifold Approximation and Projection (UMAP), which is incrementally learned in conjunction with SOUMAP prototypes. As in regular SOM learning, the Hebbian connection formed between input and output spaces results in a topologically trustworthy low-dimensional embedding amenable to clustering. Through controlled experiments we show that SOUMAP's more expressive "lattice" improves the quality of clusterings obtained from it.

Keywords: Self-Organizing Maps · Dimensionality Reduction · Clustering

1 Introduction

Dimensionality reduction (DR) techniques serve various purposes in data analysis and machine learning. As feature selectors, they are often used as a preprocessor to either reduce the computational burden or improve performance of subsequent learning stages. When used to project high-dimensional data to two or three dimensions, DR techniques allow visualization of high-d structures; such visualizations are often used to inform or validate data clusterings, and as scaffolding upon which summary statistics of high-d manifolds can be visualized in an organized manner. For success in these latter roles, it is crucial that the outputs of DR algorithms are fiducial representations of high-d structure.

Modern DR techniques such as t-Stochastic Neighbor Embedding (t-SNE, [8]) and Uniform Manifold Approximation and Projection (UMAP, [10]) have become commonplace tools for data summarization across research disciplines, with heavy use in the life and physical sciences. Despite recent work shedding light on how t-SNE and UMAP produce such expressive embeddings [1,2] confusion persists about about how faithfully either preserve underlying data structure. This confusion arises partly from potential user (mis-)parameterization, but also from optimization of their cost functions [2,3].

Authors acknowledge partial support from NSF AAG 2107942 and seed funding from NSF's Institute for Foundations in Machine Learning.

T. Villmann et al. (Eds.): WSOM+2024, LNNS 1087, pp. 63–73, 2024.
https://doi.org/10.1007/978-3-031-67159-3_8

On the other end of the (dis-)trust spectrum is Kohonen's Self-Organizing Map (SOM), whose non-linear projections are constrained to a low-dimensional (usually 2- or 3-d) lattice. While this constraint typically results in a less expressive embedding of high-d manifolds, SOM embeddings are assumed to be (approximately) topology-preserving due to both their robustness to (mis-) parameterization and their explicit and simple Hebbian learning link between low- and high-d spaces [6]. This link establishes a feedback mechanism between learned representations in input and output spaces, bolstering our trust in both at termination of learning. Further, associated visualizations [12] readily convey where a SOM is *not* topology preserving, which provides insurance against faulty inferences from its learning.

This work is motivated by a desire to combine the visual expressiveness of t-SNE or UMAP DR with the trustworthiness and inferential confidence offered by the SOM. We achieve this by replacing the rigid SOM lattice with a more nimble low-d topology obtained via UMAP during iterative self-organizing learning. Thus, our contributions can be thought of as either *adding* a competitive learning phase to UMAP, or *removing* the dependence of the SOM on its lattice; both are correct. As in Kohonen's mapping, and in contrast to similarly motivated work [5], SOUMAP's learning rule inextricably marries its prototype formation to the topology of its (prevailing) output space which bolsters confidence that the latter accurately represents the former. In the following sections we outline the methodology for a Self-Organizing UMAP (SOUMAP) (Sect. 3) and highlight experiments showing its utility (Sect. 5).

We stress that SOUMAP is a flexible framework marrying competitive Hebbian learning with alternative output space topologies, which could arise from any DR technique. Here, we have chosen UMAP due to its speed but could just as easily have prescribed a Self-Organizing {t-SNE, Autoencoder, ISOMAP, ...} following the same logic presented here.

2 Background

2.1 The Self-Organizing Map

A SOM is an unsupervised neural network that mimics how the cortex of the human brain summarizes and organizes information. SOMs consists of a rigid (usually 2-d) lattice $\mathscr{L} = \{l_i\}_{i=1}^{M}$ of M artificial neurons, each of which is connected to its input space by a $d-$dimensional weight vector known as a *prototype*. In this work we denote prototypes by $W = \{w_i \in \mathbb{R}^d\}_{i=1}^{M}$. The SOM learns a sample $X = \{x_s \in \mathbb{R}^d\}_{s=1}^{N}$ of size N, assumed to arise from manifold $\mathscr{M}$ with density $f(x)$, by simultaneously (1) positioning its $M << N$ prototypes in $\mathbb{R}^d$ to estimate f and (2) organizing the prototypes' associated neurons on $\mathscr{L}$ to represent the structure of $\mathscr{M}$. Thus, the SOM performs vector quantization and nonlinear DR. SOM-based clusterings utilize both of these learned results to partition data.

The topology-preserving SOM mapping $\mathscr{M} \rightarrow \mathscr{L}$ is formed via an iterative two-step learning rule relying on competition (among the prototypes to represent

each datum) and cooperation (among lattice neurons as prototypes are updated). The latter is important to this project so we reiterate: the SOM lattice $\mathscr{L}$ influences its prototype formation during learning. This influence makes the SOM robust to spurious network initialization and bestows the topology-preserving properties upon the learned mapping.

To be more explicit we define $BMU1(s) = \arg\min_i \delta(x_s, w_i)$ as the best matching unit (i.e., the index of the prototype of minimum distance) of a particular observations x_s. We have let $\delta(\cdot,\cdot)$ denote an arbitrary distance, although it is typically prescribed as Euclidean. At the start of learning, prototypes w_i and their associated lattice positions l_i are initialized (either randomly, or via some weakly-organizined manner such as a Principal Components Analysis of the underlying data, [6]). At each iteration t, batch SOM learning (which we use in this work) updates the prototypes as a weighted average of all input data,

$$w_i(t) = \frac{\sum_{s=1}^{N} \eta(i, BMU1(s), t) \star x_s}{\sum_{s=1}^{N} \eta(i, BMU1(s), t)}, \tag{1}$$

where the weighting is determined by the neighborhood function η, defined on $\mathscr{L}$. η is typically a Gaussian smoothed by a time-annealed $\sigma(t)$:

$$\eta(i, j, t) = \exp\left(-\delta(l_i, l_j)^2 / 2\sigma(t)^2\right), \tag{2}$$

where, here, $\delta(l_i, l_j)$ is usually a geodesic distance on $\mathscr{L}$. We denote geodesics by $\Delta(\cdot,\cdot)$ in what follows. Annealing $\sigma(t)$ from some initially large value to a terminal small value $\sigma(T) \approx 1$ ensures organization of $\mathscr{L}$ with respect to W.

2.2 UMAP

UMAP [10] learns an embedding $Y = \{y_i \in \mathbb{R}^{d'}\}_{i=1}^{N}$ of arbitrary high-d points $P = \{p_i \in \mathbb{R}^{d}\}_{i=1}^{N}$, where $d' << d$ is user-specified at the onset of learning. Y arises via stochastic gradient descent of the UMAP cost function

$$C_U = -\sum_{ij} \left[\underbrace{\mu_{ij} \log(\phi_{ij})}_{\text{attractive}} + \underbrace{(1-\mu_{ij}) \log(1-\phi_{ij})}_{\text{repulsive}} \right]. \tag{3}$$

The μ_{ij} terms are similarities $\in [0,1]$ between high-d points p_i and p_j:

$$\mu_{ij} = (\mu_{j|i} + \mu_{i|j}) - \mu_{j|i}\mu_{i|j}, \quad \mu_{j|i} = e^{-\beta(\delta(p_i,p_j)-\rho_i)}, \quad \rho_i = \min_{j' \neq i} \delta(p_i, p_{j'})$$

where $\delta(\cdot,\cdot)$ is usually Euclidean. ρ_i is the nearest neighbor distance from p_i, so that $\mu_{j|i}$ is a smoothed similarity between p_i and all other points such that $\max \mu_{j|i} = 1$. β is a smoothing parameter found via binary search such that $\sum_j \mu_{j|i} = \log_2(k_U)$. $\mathbf{k_U}$ is the most important user-supplied parameter to UMAP controlling the number of nearest-neighbors UMAP attempts to preserve in its embedding. μ_{ij} merely symmetrizes the $\mu_{j|i}$.

Likewise, the ϕ_{ij} terms of (3) are similarities of the associated points y_i and y_j in the low-d embedding, given by $\phi_{ij} = (1 + a \star \delta(y_i, y_j)^{2b})^{-1}$. ϕ is a generalized Cauchy probability density, where tail thickness is controlled by a and b. Minimization of UMAP's cost proceeds by stochastic gradient descent for a specified number of epochs T_U.

3 SOUMAP Methodology

SOUMAP is a hybrid, iterative paradigm linking competitive and cooperative SOM learning (Sect. 2.1) with a UMAP embedding (Sect. 2.2) of the prevailing SOM prototypes at each iteration.

SOUMAP initializes a regular SOM lattice and, after T_ϵ iterations of SOM learning performs a small number (T_U) of UMAP iterations on the prevailing values of W, replacing the neuron locations l_i with the UMAP result y_i. Prototype updates proceed as normal, except the distance δ in the neighborhood activation function η (2) is now computed with respect to the topology of Y. For this purpose, we compute the Delaunay graph $\mathscr{D}_Y$ [4] of embedded points Y, and measure the geodesic $\Delta(\cdot, \cdot)$ along $\mathscr{D}_Y$, instead of along $\mathscr{L}$. From here on, every SOUMAP iteration t involves a prototype update, and every T_ϵ iterations another UMAP update is performed to modify $Y(t)$, using $Y(t - T_\epsilon)$ as UMAP's initial coordinates.

By allowing the neurons to abandon their initial lattice positions and slowly drift toward something deemed more optimal (by UMAP), SOUMAP has the capacity to produce a terminal configuration that is more expressive than $\mathscr{L}$, and more topology-preserving than UMAP would produce on its own, if the terminal SOM prototypes were embedded in a subsequent (separate) stage post-learning. We present the overall SOUMAP algorithm pseudocode in Algorithm 1 and address several of its more exotic steps (compared to regular SOM learning) in the remainder of this section.

During early SOM learning, larger values of $\sigma(t)$ tend to push many SOM prototypes W to very similar values (major density peaks) for a short period of time; this behavior is rectified as $\sigma(t)$ is annealed, which allows the prototypes W to represent more nuanced regions of the underlying data density. This is perfectly normal SOM behavior, where the lattice helps the prototypes unwind to their ultimate refined positions.

However, UMAP is particularly sensitive to the high degree of correlation among these infant prototypes and responds by embedding many prototypes to nearly identical Y at early stages. Because we use $Y(t - T_\epsilon)$ as UMAP's initialization for the UMAP update at iteration t, this imparts lasting effects on Y. Refraining from any UMAP updates until the SOM enters its convergence phase would mitigate this, but would also prevent some of the clustering benefits of SOUMAP discussed in Sect. 5. Instead, we have adopted a strategy where, during early learning, UMAP updates are performed on the prevailing prototypes at iteration t *in conjunction* with a sub-sample $S(t)$ of X of size $\gamma(t)N$ which decreases with t. That is, UMAP is applied to $W \cup S(t)$, and the prototype

Algorithm 1. SOUMAP

Inputs: X (data), $\mathscr{L}_x, \mathscr{L}y$ (lattice width and height), T_ϵ (UMAP update frequency), T (max. iterations), σ_0 (starting width of SOM neighborhood), T_U (number of UMAP epochs performed during incremental UMAP updates to Y)

Initialization: of SOM lattice $\mathscr{L}$, geodesic distances Δ along $\mathscr{L}$, and prototypes W

Learning:
$t \leftarrow 1$
while $t \leq T$ **do**
  $\sigma(t) \leftarrow$ anneal neighborhood radius
  $W \leftarrow$ SOM prototype update, using η with Δ and $\sigma(t)$
  if $t \mod T_\epsilon = 0$ **then**
    $\gamma(t) \leftarrow$ anneal sub-sampling rate, sample $S(t)$ from X
    $Y \leftarrow$ UMAP projection (T_U epochs) of $W \cup S(t)$
    $\mathscr{D}_Y \leftarrow$ Update Delaunay graph of Y
    $\Delta \leftarrow$ Update geodesic distances along $\mathscr{D}_Y$
  $t \leftarrow t + 1$

embeddings are extracted from this (joint) UMAP output. $\gamma(t) = \sigma^2(t)/|\mathscr{L}|$ is the sub-sampling rate at iteration t, where $|\mathscr{L}|$ is the cardinality of $\mathscr{L}$. Over time, $|S(t)|$ decreases, ensuring the embedded prototypes Y reflect only their high-d counterparts W at termination. We believe this sub-sampling strategy predominately necessary when UMAP is used as the DR technique to produce Y; UMAP's sensitivity appears to stem from the oddities of its optimization routine [3], as we do not observe similar behaviour when, e.g., t-SNE is used instead.

4 Experiment Descriptions

4.1 Datasets, Parameterization, and Design

To assess SOUMAP performance for both of its tasks (prototype learning and DR) we have conducted a series of experiments on samples of size $N = 20,000$ from the following datasets: **worms64**[1] is a synthetic dataset of dimension $d = 64$ containing 25 clusters of "worm-like" shapes; **MNIST**, **Fashion-MNIST**[2], and **Kazushiji-MNIST**[3] are commonly used databases of 28×28 pixel ($d = 784$) grayscale images of handwritten digits 0-9, 10 different articles of clothing, and 10 different Japanese Hiragana characters, respectively.

For each dataset above, we perform regular (lattice-based) batch SOM learning ($M \approx 2000$ prototypes for worms64, $M \approx 3600$ for MNIST and its relatives)and compare it to SOUMAP learning for the same number of total epochs

[1] https://cs.uef.fi/sipu/datasets/.
[2] https://github.com/zalandoresearch/fashion-mnist.
[3] https://github.com/rois-codh/kmnist.

$T = 50$. For SOUMAP we allow UMAP updates of $T_U = 100$ epochs for every $T_\epsilon = 5$ of its iterations. The initial SOM geodesic radius σ_0 was set to 25% of the hexagonal lattice width and annealed exponentially to $\sigma(T = 50) = 1$. Thus, both the SOM and SOUMAP receive 50 batch updates to their prototypes; SOUMAP receives a total of 1,000 ($\frac{T=50}{T_\epsilon=5} \times (T_U = 100)$) epochs of UMAP learning of the evolving prototypes. For now, we have kept all UMAP parameters (in particular, number of neighbors $k_U = 15$) at their defaults. The increased computational time for UMAP is negligble for these data; worst cases (MNIST and its relatives, where $M \approx 3600$ 784-d prototypes) require less than 10 additional seconds for all $1,000$ UMAP training epochs. To investigate a simpler analog to SOUMAP we also compute prototypes via K-means (using the same M discussed above for SOM & SOUMAP), and embed them with default UMAP. We denote this two-stage approach, where prototype learning and embedding are sequentially separate processes, by KM+U. All three prototype generation methods and their associated embeddings are compared in Sect. 5.

4.2 Quality Measures

We assess the performance of SOM and SOUMAP learning by measuring the topology preservation and clustering amenability of each method.

Topology preservation measures (TPMs) assess how well high-d neighborhoods are preserved when represented in a low-d embedding. Numerous TPMs exist, varying mostly in their definition of "neighborhood." We have selected two for consideration: the k-Ary score, and the Weighted Average Folding Length. The k-Ary score [7] measures the proportion of a high-d k-nearest neighborhood around each prototype w_i that is preserved after embedding w_i as y_i in low-d, averaged over all $i \in \{1, \ldots, M\}$ prototypes. As this K-ary measure ([7, equation 15]) yields a performance curve over $k \in \{1, \ldots, M-1\}$, we report the area under such curve (AUC), normalized by its theoretical maximum (M-2), for comparison across datasets. k-Ary is in the range $[0, 1]$, with 1 indicating perfect preservation of high-d neighborhoods of all sizes k.

The Weighted Average Folding Length (WAFL) [13] is a more local measure of TP *with respect to* the data manifold. WAFL is based on the CADJ graph with prototypes as vertices [12]. CADJ is a weighted version of the Induced Delaunay Graph [9], which estimates topological adjacencies within the manifold $\mathcal{M}$ from which X was sampled. Such adjacencies may be different, e.g., when two prototypes are nearest Euclidean neighbors, but no data X exists between them. Thus, CADJ is tuned to highlight seams in the learned manifold caused by its disconnection at cluster boundaries. CADJ edge weights are given by

$$CADJ_{ij} = \sum_s I(BMU1(s) = i \wedge BMU2(s) = j) \tag{4}$$

where $I(\cdot)$ is the indicator function, and $BMU1(s)$ and $BMU2(s)$ return the indices of the first and second best matching units (nearest, and second-nearest prototypes) to datum x_s, respectively. The WAFL, defined as

$$WAFL = \frac{1}{N} \sum_{ij} CADJ_{ij} \star \Delta_{ij}, \tag{5}$$

reports the weighted average geodesic neighborhood radius along the Delaunay graph $\mathcal{D}_Y$ of embedded prototypes required to include all high-d CADJ neighbors. Thus, it measures the degree to which each CADJ edge is preserved as a nearest geodesic neighbor in embedded space. An ideal value $WAFL = 1$ indicates all CADJ neighbors are preserved, while any $WAFL > 1$ grades the mis-preservation accordingly.

Clustering assessments of both the learned SOUMAP prototypes, and their projection Y, can help determine the suitability for SOUMAP as a clustering tool. To this end, we compute an average of clustering quality metrics (the Adjusted Rand Index – ARI, and Adjusted Mutual Information – AMI) using the known data labels as ground truth; as each is more applicable in certain conditions [11] we have averaged them into a single measure, $cl = (AMI + ARI)/2$. To cluster learned W or Y we employ hierarchical agglomerative clustering with single (HACS), complete (HACC), average (HACA), and Ward (HACW) linkages, compute cl for each, and accept the maximum resulting cl value as the reported metric for each dataset. We cluster both W and Y in this manner, producing quality measures for both of SOM/SOUMAP's prototypes and resulting embeddings.

5 Results

Visual results from our experiments with each dataset are presented in Fig. 1, where the final SOM embedding is shown at left, and the final SOUMAP embedding shown at right. In each panel, neurons are colored by their corresponding class label, which was assigned by plurality vote of the class labels of data in each neuron's receptive field (RF), which is defined as the set of $\{x_s\}$ mapped to each neuron i. Each neuron's plotted size is scaled to the size of its receptive field; dead neurons (those representing no data) have been removed. Additionally, the thin gray lines connecting neurons depict the edges of the CADJ graph (4). This visualization technique, which is an un-weighted analog of the CONNvis visualization [12], helps identify topology violations; maps with better topology preservation have overall shorter gray edges with minimal edge crossings.

Visual inspection of Fig. 1 reveals strong organization for both the SOM and SOUMAP. However, the final SOUMAP embeddings (right column) contain more tightly packed clusters and, hence, more separation between clusters, contributing to a more expressive representation of overall cluster structure. This is the hallmark characteristic of UMAP which motivated SOUMAP in the first place. SOUMAP's early abandonment of the lattice has visually appealing benefits, particularly for the more complicated data considered in these experiments. For example, the SOM has signaled, to the best of its ability, a boundary between the blue and orange clusters of FMNIST (representing images of boots and handbags, respectively); SOUMAP's delineation of this boundary clearly

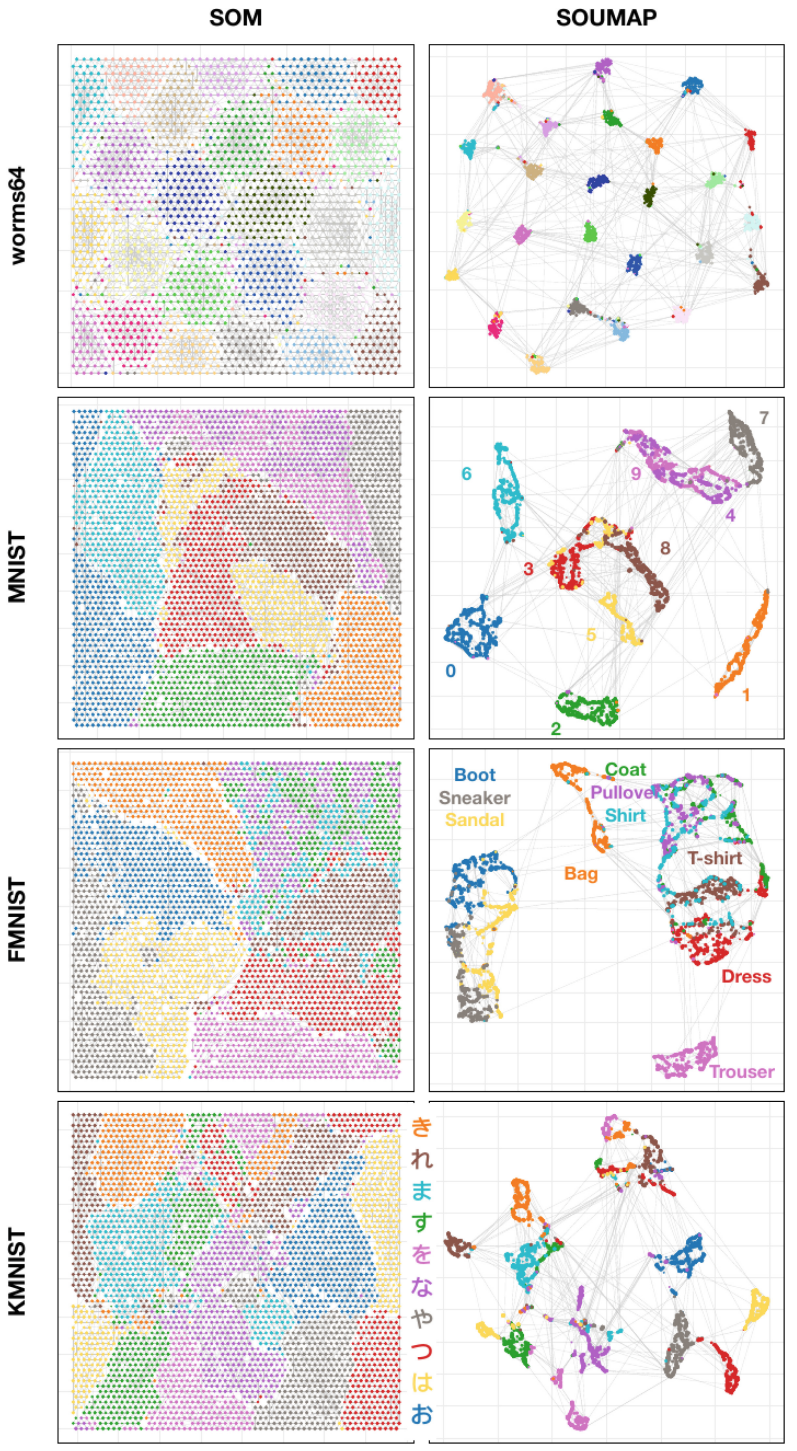

Fig. 1. Learned prototype embeddings of our experimental data from Sect. 4.

Table 1. Quantitative Performance Measures, as defined in Sect. 4.2. Directional arrows indicate better performance according to each measure.

			worms64	MNIST	FMNIST	KMNIST
Topology	k-Ary ↑	SOM	0.56	0.60	0.65	0.61
		SOUMAP	0.54	0.54	0.58	0.55
		KM+U	0.21	0.42	0.44	0.41
	WAFL ↓	SOM	1.37	1.44	1.24	1.34
		SOUMAP	1.28	1.23	1.16	1.17
		KM+U	5.11	3.02	2.83	2.88
Clustering	W ↑	SOM	0.70	0.54	0.46	0.44
		SOUMAP	0.73	0.62	0.51	0.45
		KM+U	0.00	0.17	0.18	0.13
	Y ↑	SOM	0.52	0.41	0.42	0.34
		SOUMAP	0.73	0.75	0.52	0.51
		KM+U	0.48	0.77	0.49	0.48

separates bags not only from boots but also from the mixed coat/pullover/shirt super class. More pronounced separation is visible for SOUMAP's KMNIST embedding; the SOM only weakly signals separation between the orange/brown, green/pink, and gray/red character classes, but SOUMAP shows marked delineation for these characters.

From Table 1 we observe that, of the two TPMs considered here, k-Ary and WAFL report divergent trends: the SOM has a consistently higher (better) k-Ary score, but also a consistently higher (worse) WAFL score, than SOUMAP. Our UMAP parameterization is the cause of this behavior, as UMAP was only instructed to attempt preservation of the $k_U = 15$ nearest high-d neighbors; it is doing so, but at the expense of higher-order neighborhoods. This is a known trade-off between global and local views of data in DR techniques. However, as the CADJ graph has proven a useful tool for SOM clustering, we believe fiducial representation of CADJ neighborhoods more important than representation of neighborhoods of all sizes, which is what k-Ary is designed to measure.

Turning to the clustering quality metric of Table 1, we see SOUMAP's strength is producing embeddings Y that are more amenable to clustering than the regular SOM. This agrees with our visual assessments above which found SOUMAP more expressive in general. Interestingly, there is also some small improvement when clustering SOUMAP's prototypes W directly; we assume this is a result of SOUMAP's output topology becoming tightly organized early in the learning process. If such embedded structures are "real" (meaning they faithfully represent analogous structure in high-d), it should be expected that the learning reinforcement provided by the neighborhood function η would help refine the high-d prototypes and bolster prototype clusterings.

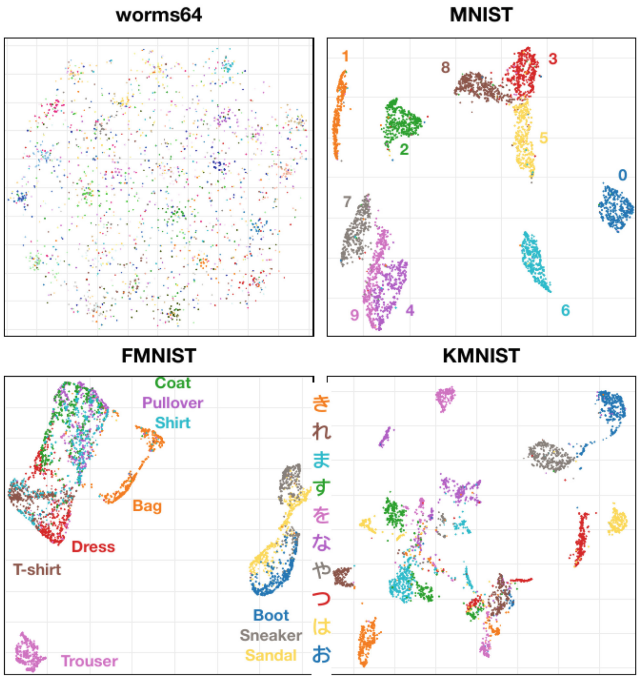

Fig. 2. K-means + UMAP Embeddings of the data from Sect. 4.

The hybrid K-means + UMAP baseline strategy produces seemingly coherent embeddings of our data (Fig. 2) but underperforms both the SOM and SOUMAP according to the quantitative measures of Table 1. While the embeddings resulting from KM+U possess clusterability similar to SOUMAP, the K-means prototypes are not nearly as amenable to clean clustering. Additionally, the low-d mapping of the two-stage KM+U procedure suffers from large degradation in topology preservation. Combined, these results indicate there is much to be gained from integrating the prototype learning and DR tasks, vs. performing them sequentially.

6 Conclusions and Future Work

We have presented SOUMAP, a hybrid prototype learning scheme to harness all of the benefits of SOM-based vector quantization (sample size and noise reduction) in conjunction with the expressive low-d visualizations of data provided by UMAP. Our experiments show that SOUMAP is capable of both preserving the high quality of SOM-based VQ while simultaneously providing visually appealing embeddings that are more amenable to cluster inference.

With the prototype learning rule established, the next stage in SOUMAP development is a deeper analysis into the parameterization of its UMAP update

stage. Ultimately we would prefer SOUMAP to be completely self-parameterized and, with so many parameters at play, our UMAP stage would likely benefit from parameter tuning. [13] outlines an automated procedure for specifying t-SNE's perplexity parameter based on an analysis of the CADJ graph (4); as UMAP shares a similar parameter (number of neighbors, denoted k_U here), we believe a related analysis may partly contribute to a self-governed SOUMAP updating process. Because UMAP is more complex than t-SNE (i.e., has more parameters, and a wildly different optimization procedure), expanding upon [13] in a UMAP-specific context is a necessary next step.

References

1. Böhm, J.N., Berens, P., Kobak, D.: Attraction-repulsion spectrum in neighbor embeddings. J. Mach. Learn. Res. **23**(95), 1–32 (2022)
2. Damrich, S., Böhm, N., Hamprecht, F.A., Kobak, D.: From t-SNE to UMAP with contrastive learning. In: The Eleventh International Conference on Learning Representations (2022)
3. Damrich, S., Hamprecht, F.A.: On UMAP's true loss function. Adv. Neural. Inf. Process. Syst. **34**, 5798–5809 (2021)
4. Fortune, S.: Voronoi diagrams and delaunay triangulations. In: Computing in Euclidean Geometry, pp. 225–265 (1995)
5. Fritzke, B.: Growing cell structures-a self-organizing network for unsupervised and supervised learning. Neural Netw. **7**(9), 1441–1460 (1994)
6. Kohonen, T.: Self-Organizing Maps, vol. 30. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-56927-2
7. Lee, J.A., Peluffo-Ordóñez, D.H., Verleysen, M.: Multi-scale similarities in stochastic neighbour embedding: reducing dimensionality while preserving both local and global structure. Neurocomputing **169**, 246–261 (2015)
8. Van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**(11) (2008)
9. Martinetz, T., Schulten, K.: Topology representing networks. Neural Netw. **7**(3), 507–522 (1994)
10. McInnes, L., Healy, J., Melville, J.: UMAP: uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426 (2018)
11. Romano, S., Vinh, N.X., Bailey, J., Verspoor, K.: Adjusting for chance clustering comparison measures. J. Mach. Learn. Res. **17**(1), 4635–4666 (2016)
12. Tasdemir, K., Merényi, E.: Exploiting data topology in visualization and clustering of self-organizing maps. IEEE Trans. Neural Networks **20**(4), 549–562 (2009)
13. Taylor, J., Merényi, E.: Automating t-SNE parameterization with prototype-based learning of manifold connectivity. Neurocomputing **507**, 441–452 (2022)