Feature Learning for Nonlinear Dimensionality Reduction toward Maximal Extraction of Hidden Patterns

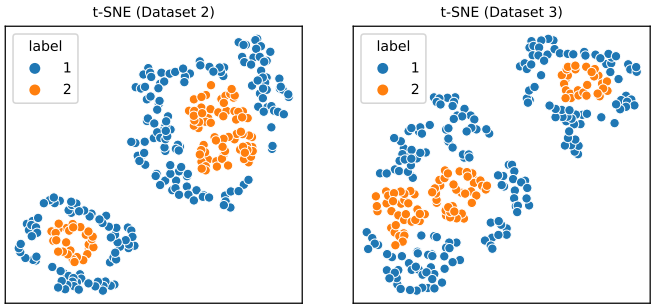
Supplementary Explanations and Experiments

Takanori Fujiwara, Yun-Hsin Kuo, Anders Ynnerman, and Kwan-Liu Ma

A Supplementary Experiments for Sect. 3 Motivating Examples

A.1 t-SNE Results

Fig. A.1 shows t-SNE results of the datasets in Sect. 3, demonstrating that the issues described in Sect. 3 also happen even when using t-SNE.

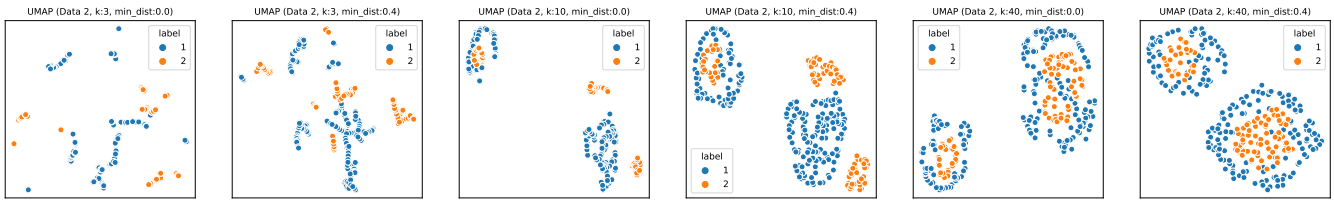


(a) The dataset shown in Fig. 1-d. (b) The dataset shown in Fig. 1-g.

Figure A.1: t-SNE results of the datasets in Sect. 3.

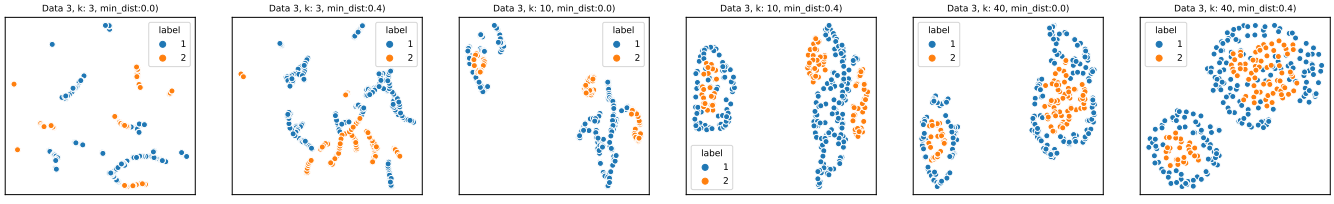
A.2 UMAP Results with Different Hyperparameters

We applied UMAP to the datasets in Sect. 3 with different hyperparameters to demonstrate that the described issues cannot be solved by the hyperparameter adjustment. As UMAP's most important hyperparameters are `n_neigbors` and `min_dist` (refer to https://umap-learn.readthedocs.io/en/latest/parameters.html), we produced UMAP with different `n_neigbors` (from 3 to 40) and `min_dist` (from 0 to 0.8). Fig. A.2 and Fig. A.3 list a subset of the UMAP results (a full set of the results is in our online repository, https://takanori-fujiwara.github.io/s/fealm/).



(a) `n_neigbors=3`, `min_dist=0` (b) `n_neigbors=3`, `min_dist=0.4` (c) `n_neigbors=10`, `min_dist=0.0` (d) `n_neigbors=10`, `min_dist=0.4` (e) `n_neigbors=40`, `min_dist=0.0` (f) `n_neigbors=40`, `min_dist=0.4`

Figure A.2: UMAP results of the dataset shown in Fig. 1-d with different hyperparameters.



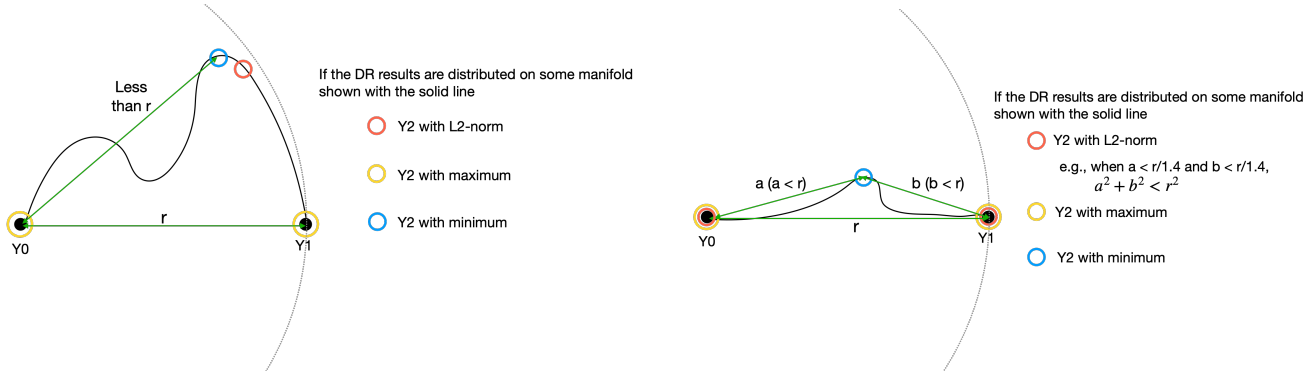
(a) `n_neigbors=3`, `min_dist=0` (b) `n_neigbors=3`, `min_dist=0.4` (c) `n_neigbors=10`, `min_dist=0.0` (d) `n_neigbors=10`, `min_dist=0.4` (e) `n_neigbors=40`, `min_dist=0.0` (f) `n_neigbors=40`, `min_dist=0.4`

Figure A.3: UMAP results of the dataset shown in Fig. 1-g with different hyperparameters.

B Supplementary Explanations for Sect. 4 FEALM Framework

B.1 Graphical Explanation for the Reasoning of Using a Minimum as Φ.

Fig. B.1 shows two concrete examples that demonstrate the effectiveness of taking a minimum as Φ. We can assume DR results $(\mathbf{Y}_0, \mathbf{Y}_1, \cdots)$ are distributed on a certain multidimensional manifold. Here, we explain with 2D curved shapes to provide simple examples. Consider the case where some optimization using Eq. 1 successfully found the best DR result, $\mathbf{Y}_1$, which is maximally different from the original DR result, $\mathbf{Y}_0$. We denote the dissimilarity between $\mathbf{Y}_0$ and $\mathbf{Y}_1$ by r. Then, because $\mathbf{Y}_1$ has the largest dissimilarity from $\mathbf{Y}_0$ (and vice versa when using a symmetric measure), $\mathbf{Y}_2$ that has the largest dissimilarities from $\mathcal{Y}_i = \{\mathbf{Y}_0, \mathbf{Y}_1\}$ should have the dissimilarity smaller than or equal to r from $\mathbf{Y}_0$. As a result, when the manifold has a shape shown in Fig. B.1-a and taking a maximum is set as Φ, the optimization selects either $\mathbf{Y}_0$ or $\mathbf{Y}_1$ as $\mathbf{Y}_2$, and keeps selecting either of them as $\mathbf{Y}_3$ and so on. Both computing the $L2$-norm and minimum can provide similar results for this case. However, for manifolds similar to Fig. B.1-b, where $\mathbf{Y}_0$ and $\mathbf{Y}_1$'s dissimilarities are much larger than those from other potential DR results, even the $L2$-norm can cause a similar issue to the maximum. As illustrated in Fig. B.1-b, taking a minimum can find an appropriate $\mathbf{Y}_2$.



(a) A case where taking a maximum is not suitable.

(b) A case where both taking a maximum and the $L2$-norm (or the Frobenius norm for higher dimensions) are not suitable.

Figure B.1: Graphical explanations of the reasoning why we should use taking a minimum as Φ.

C Supplementary Explanations and Experiments for Sect. 5 Exemplifying Method

C.1 Pseudocode of FEALM-UMAP

Algorithm 1 FEALM-UMAP's optimization.

Inputs:
$\mathbf{X}$, k, r // original data, # of neighbors, # of repeats of the optimization
`constraints_on_P`, p // description of constraints added to a projection matrix, # of parameters optimized in NMM
`n_evaluations`, `n_recommendations` // # of evaluations in NMM, # of recommendations utilizing non-best solutions

Initialize functions and settings
Prepare a k-NN graph construction function, f_{Gr}, with k
Prepare a manifold, `manifold`, that satisfies `constraints_on_P`
Construct an initial graph with $G_0 = f_{\mathrm{Gr}}(\mathbf{X})$
Set $\mathcal{G} = [G_0]$, $\mathcal{P} = [\,]$
Perform optimization with the enhanced NMM
for $i = 1, 2, \cdots, r$ **do**
  // Random search to select (p + 1) initial solutions
  From `manifold`, obtain randomly initialized (10p + 1) `solutions`
  Evaluate `solutions` with Eq. 2 using NSD **in parallel**
  Keep the best (p + 1) `solutions` and set the best solution as $\mathbf{P}_i$.
  // The adaptive NMM using (p + 1) initial solutions
  for $j = 1, 2, \cdots,$ `n_evaluations` **do**
    Update `solutions` along `manifold` by following the adaptive NMM procedure.
    Evaluate `solutions` with Eq. 2 using NSD (**in parallel** when the NMM performs the shrinkage).
    Update $\mathbf{P}_i$ if some of `solutions` is better than the current $\mathbf{P}_i$
    If reaching the convergence, **break**.
  Append G_i to $\mathcal{G}$, where $G_i = f_{\mathrm{Gr}}(\mathbf{X}\mathbf{P}_i)$
  Append $\mathbf{P}_i$ to $\mathcal{P}$
Perform spectral-clustering-based recommendations (optional)
Produce UMAP results, `embeddings`, corresponding to $\mathcal{P}$
Apply spectral clustering on `embeddings` while using NSD for the dissimilarity calculation
Extract `recommended_indices`—projection matrix indices corresponding to `embeddings` that are closest to the cluster center
Return results
return $\mathcal{P}$, `recommended_indices`

C.2 Performance Comparison of Various Graph Dissimilarity Measures

We tested the performance of all the graph dissimilarity measures implemented in netrd [27]. We used the same experimental settings with the "performance of f_{DR}, f_{Gr}, d_{Gr}, and the optimization" in. Sect. 6, and recorded the completion time of 1000 executions. While Fig. C.1 shows ten representative measures (including our implementation of NetLSD and DeltaCon), the completion times for the other measures are available in our online repository (https://takanori-fujiwara.github.io/s/fealm/).

As shown in Fig. C.1, `NetLSD (netrd)`, `NetLSD (ours)`, `DeltaCon (netrd)`, `DeltaCon (ours)`, and `Frobenius` recorded the top-5 fastest completion time. While netrd's DeltaCon shows faster completion times than netrd's NetLSD, this is caused by netrd's NetLSD does not fully utilize the matrix computations (i.e., relying on iterative loops). When both NetLSD and DeltaCon have the implementations fully relying on the matrix computations (i.e., `NetLSD (ours)` and `DeltaCon (ours)`), NetLSD shows comparable or even faster completion times. Also, when $n = 800$, `NetLSD (ours)` is approximately 20 times faster than `NetLSD (netrd)`. As mentioned in Sect. 5, computational efficiency is critical when designing the optimization that requires a large amount of executions of functions/measures. In addition to the computational efficiency, when designing the neighbor-shape dissimilarity measure, NSD, we visually compared the DR results generated through the NMM-based optimization. From the visual comparison, we observed NetLSD produced graphs with more various different shapes than other measures, such as `Frobenius`, `DeltaCon`, and `Communicability JSD`.

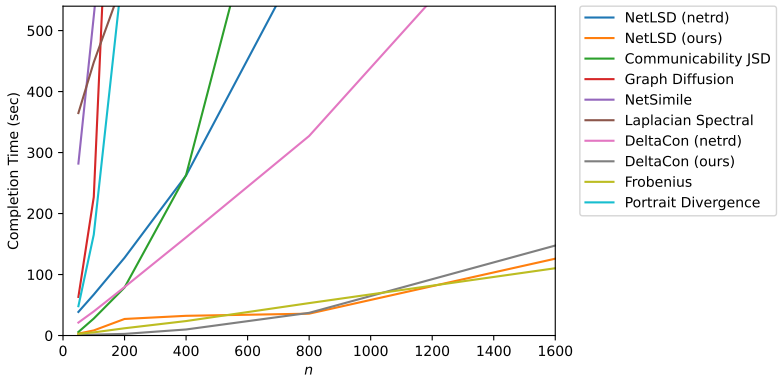


Figure C.1: Various graph dissimilarity measures' completion times of 1000 executions.

C.3 Transition of the Objective Value by the Increase of the Number of Generated DR Results

As FEALM-UMAP uses taking the minimum as the reduce function, Φ, the objective value of Eq. 2 tends to decrease as the number of generated DR results, r, increases. By observing the objective value, we can roughly guess how many DR results should be generated by FEALM-UMAP to comprehend significantly different patterns hidden in the data. For example, Fig. C.2 shows the case where FEALM-UMAP is applied to the dataset corresponding to Fig. 1-d. While there is fluctuation of the objective value (due to the random initialization of the NMM), we can generally see a decreasing trend. In this case, we can expect roughly 10–20 DR results are sufficient for this dataset.

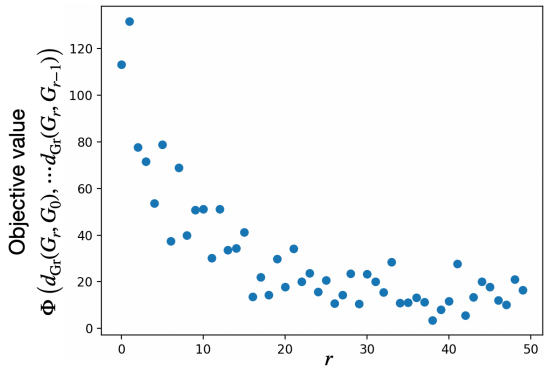


Figure C.2: The transition of the objective value as the increase of the number of generated DR results when applying FEALM-UMAP to the dataset shown in Fig. 1-d.

D Supplementary Video for Sect. 7 Visual Interface

Refer to the online supplementary materials (https://takanori-fujiwara.github.io/s/fealm/).

E Supplementary Materials for Sect. 8 Case Studies

E.1 Detailed Settings and Used Datasets

For the datasets, refer to the online supplementary material page (https://takanori-fujiwara.github.io/s/fealm/). For the detailed settings, refer to the source code in our online repository linked from the supplementary material page.

E.2 Comparison with the Method by Lehmann and Theisel

Since Lehmann and Theisel's work [24] does not provide the source code of their method, we implemented the method based on their paper (the source code is available in our online repository). Then, we produced 20 different projections for each of the Wine and the 2020 Cooperative Election Study (CES) datasets. From the results shown below (Fig. E.1 and Fig. E.2), we cannot find the patterns that FEALM-UMAP identified. We expect that this is because Lehmann and Theisel's method only generates 2D linear projection results, focuses only on the (extended) Procrustes distance, and takes the Frobenius norm as a reduction function.

One noteworthy finding from these results is that although Lehmann and Theisel's method is designed to comprehend all important patterns within $m/2$ (m: the number of attributes) projections (see [24]), this property might not be held for some cases. For example, the Wine dataset has 13 attributes ($m/2 = 6.5$); thus, 7 projections should be sufficient to cover all significantly different linear projections. However, we can see that Embedding 18 reveals a clear pattern that cannot be seen in Embedding 1–17. In Embedding 18, we can see clear clusters: one consisting of `Labels` 1 and 2 and one with `Label` 3.

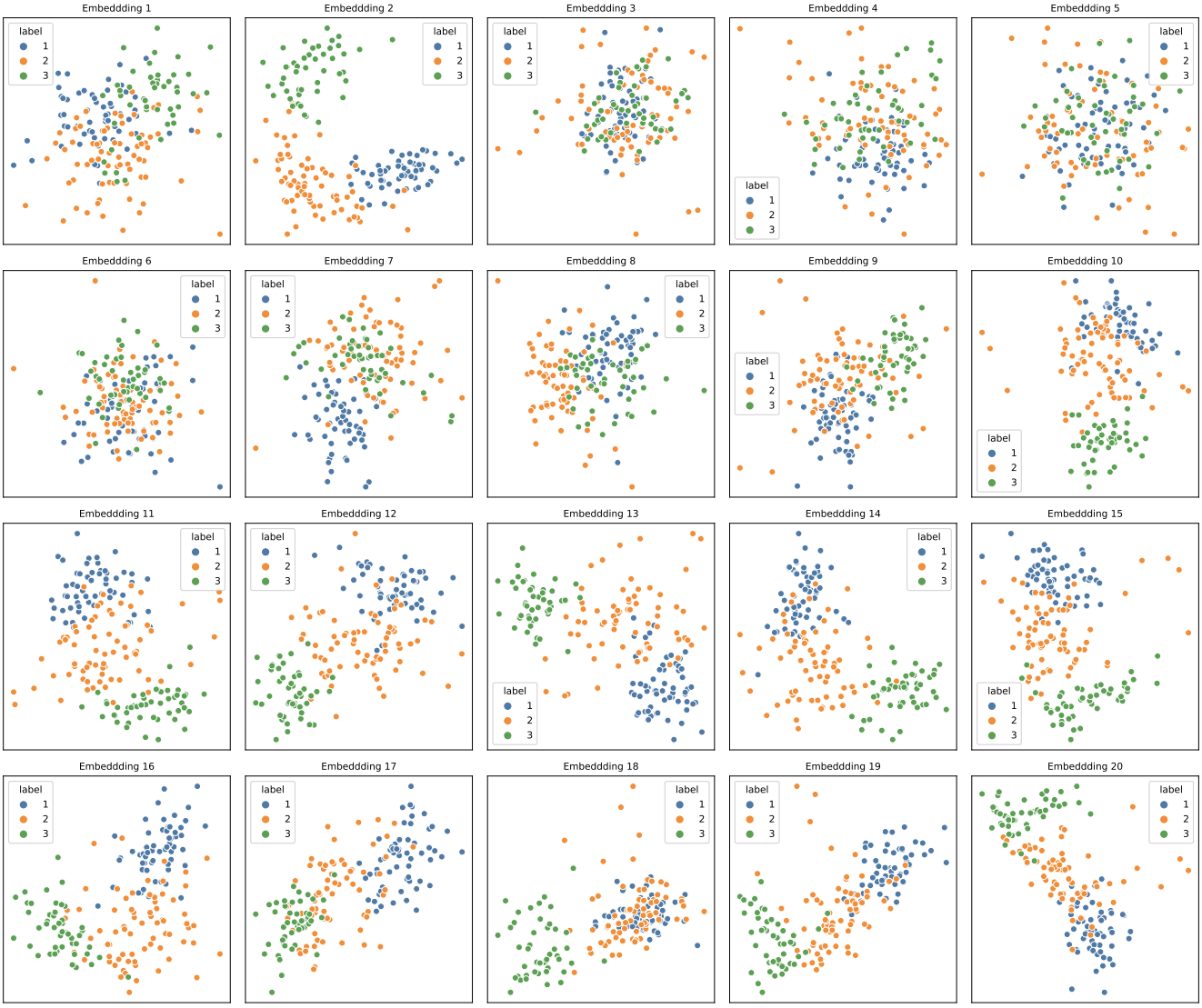


Figure E.1: 2D projections of the Wine dataset, using the method by Lehmann and Theisel [24] and listed by the produced order.

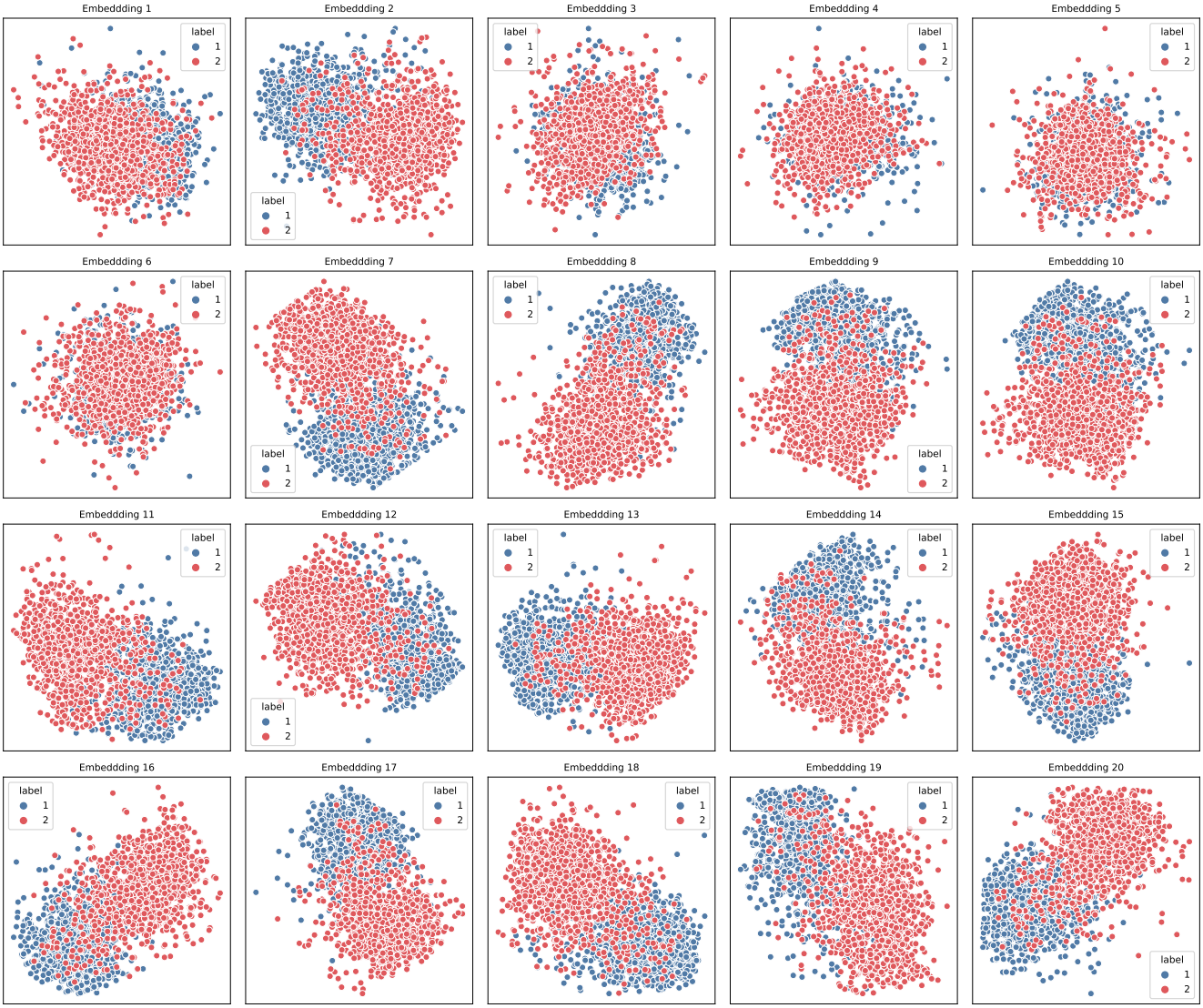
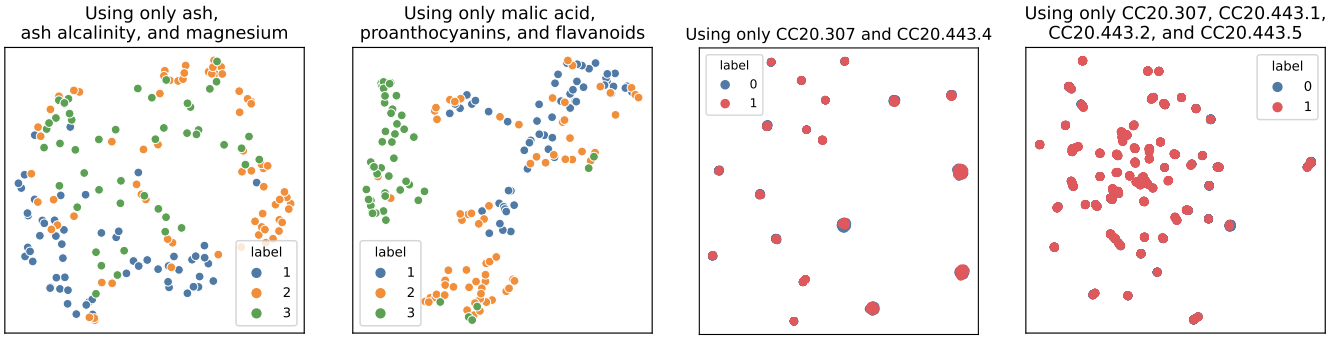


Figure E.2: 2D projections of the CES dataset, using the method by Lehmann and Theisel [24] and listed by the produced order.

E.3 Comparison with Feature Selection

Here we test whether or not the patterns uncovered in the case studies can be found by simply selecting the attributes that had large weights in the corresponding projection matrix. As shown in Fig. E.3, simply selecting a subset of attributes cannot reveal similar patterns to those found by FEALM-UMAP. These results demonstrate the importance of utilizing multiple attributes while assigning different weights.



(a) Wine corresponding to Fig. 7-a2. (b) Wine corresponding to Fig. 7-b2. (c) CES corresponding to Fig. 8-b. (d) CES corresponding to Fig. 8-c.

Figure E.3: The UMAP results with simple attribute selection. Labels 0 (Dem) and 1 (Rep) are heavily overlapped in (c) and (d).

E.4 Additional Case Study: Grouping of Handwritten Digits

In this case study, we analyze the MNIST handwritten digits dataset [32] as a case handling a large number of attributes. This dataset contains 70,000 handwritten digits (i.e., instances) stored in 28×28 pixels (i.e, 784 attributes). While applying UMAP to this dataset often reveals clusters corresponding to different digits (e.g., a cluster of `digit 4`), we seek patterns different from those clusters. As an analysis example, we compare `digits 4`, `7`, and `9`, all of which have a similar vertical or diagonal stroke. We further sample 200 instances for each digit. A UMAP result of this data is shown in Fig. E.4-a, where `digits 4`, `7`, and `9` are moderately separated into 3 clusters.

As there are 784 attributes, it is difficult for the NMM-based solver to find the optimal projections. Thus, we first reduce the number of attributes by applying PCA. Although we can apply ordinal PCA, it produces principal components (PCs) based on eigenvalue decomposition, leading to the situation where the first few PCs contain more information of the data (specifically, variance information). Consequently, as shown in Fig. E.4-b, more bottom PCs tend to capture less meaningful structures (e.g., PCs 6 and 9), which is also difficult to interpret for us. We, instead, apply the manifold-optimization-based PCA [12], which equally treats the importance of PCs (i.e., each of which captures a more similar amount of variance information). From 784 attributes, we generate 9 PCs shown in Fig. E.4-c, which explains 35% of the variance of the original data as well as produces a UMAP result (see Fig. E.4-d) with a similar cluster formation to the one using 784 attributes.

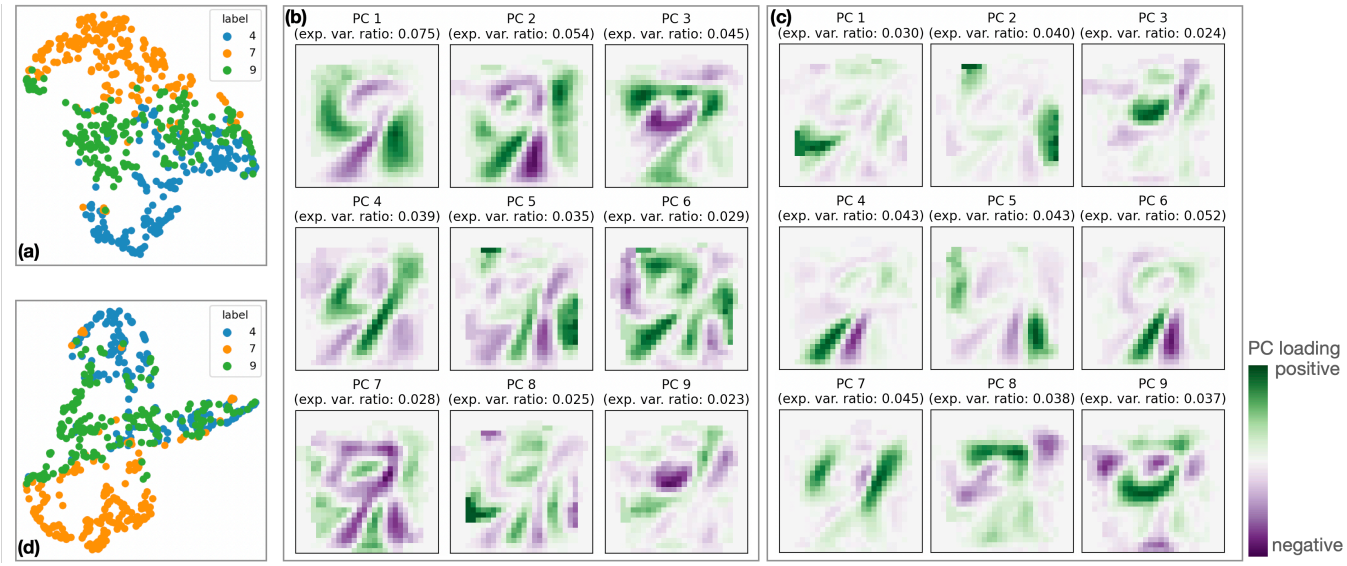


Figure E.4: UMAP results (a, d) and PCs (b, c): (a) The UMAP results using all 784 attributes as they are, (b) 9 PCs obtained with ordinary PCA (total explained variance ratio: 0.351), (c) 9 PCs obtained with the manifold-optimization-based PCA (total explained variance ratio: 0.351), and (d) the UMAP result using 9 PCs shown in (c).

We then apply FEALM-UMAP to this processed data consisting of 600 instances and 9 PCs/attributes. Fig. E.5–E.7 show a subset of UMAP results produced with the optimized projections, $\mathcal{P}$. For example, in Fig. E.5-a, we can see a better separation between `digit 7` and others than the one in Fig. E.4-d. From the auxiliary information in Fig. E.5-b, we can see that FEALM-UMAP assigns large weights to PC 3, PC 8, and PC 9. Also, based on the attribute contribution information, `digit 7` tends to have more negative PC 3, positive PC 8, and negative PC 9. By collectively looking at the information of the PCs visualized with a heatmap using a purple-green divergent colormap, we can expect that this UMAP result successfully separates `digit 7` by emphasizing the structures that `digit 7` typically holds or does not holds. For example, when drawing 7, while we often use the green horizontal stroke in PC 8, we usually do not use the green areas in PC 3 and PC 9.

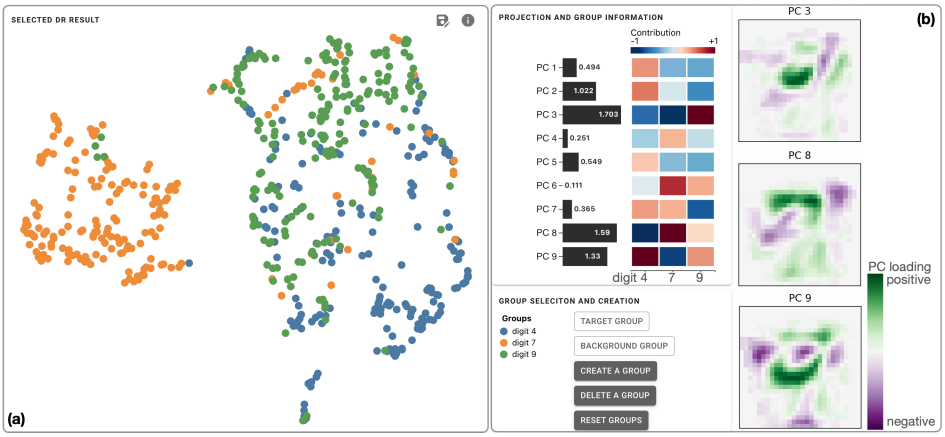


Figure E.5: FEALM-UMAP result 1: (a) the UMAP result where we can see a clear separation between `digit 7` and the others; and (b) the corresponding auxiliary information.

In Fig. E.6-a1, a2, we can see two distinct clusters mainly consisting of `digit 4` and `digit 7`. To investigate these clusters, we manually create new groups, `Groups A`–`C`, and update the attributes' contributions, accordingly. From the information visualized in Fig. E.6-b, we see that PC 6 and PC 8 are the main attributes contributing to `Groups A` and `B`'s characteristics, which are consistent with the large weights assigned to these two PCs by FEALM-UMAP. Based on the information of PCs, we can say that PC 6 captures the difference between the vertical (purple) and diagonal (green) strokes. On the other hand, PC 8 emphasizes the horizontal stroke (green) typically seen in `digit 7` but not in `digit 4`. We further update the sizes of points based on these PCs, as shown in Fig. E.6-a1, a2. From these observations, we can expect that FEALM-UMAP distinguishes digits with the diagonal stroke from those with the vertical stroke (i.e., the left side of Fig. E.6-a1 corresponds to the diagonal stroke) and also clearly separates `Groups A`–`B` from `Group C` by referring to the horizontal stroke. To further verify this finding, we visualize subsets of `digit 4` belonging to `Groups A` and `Group C` in Fig. E.6-c1, c2, respectively. We can see that Fig. E.6-c1 contains `digit 4` with similar diagonal strokes, while Fig. E.6-c2 shows various shapes of `digit 4`.

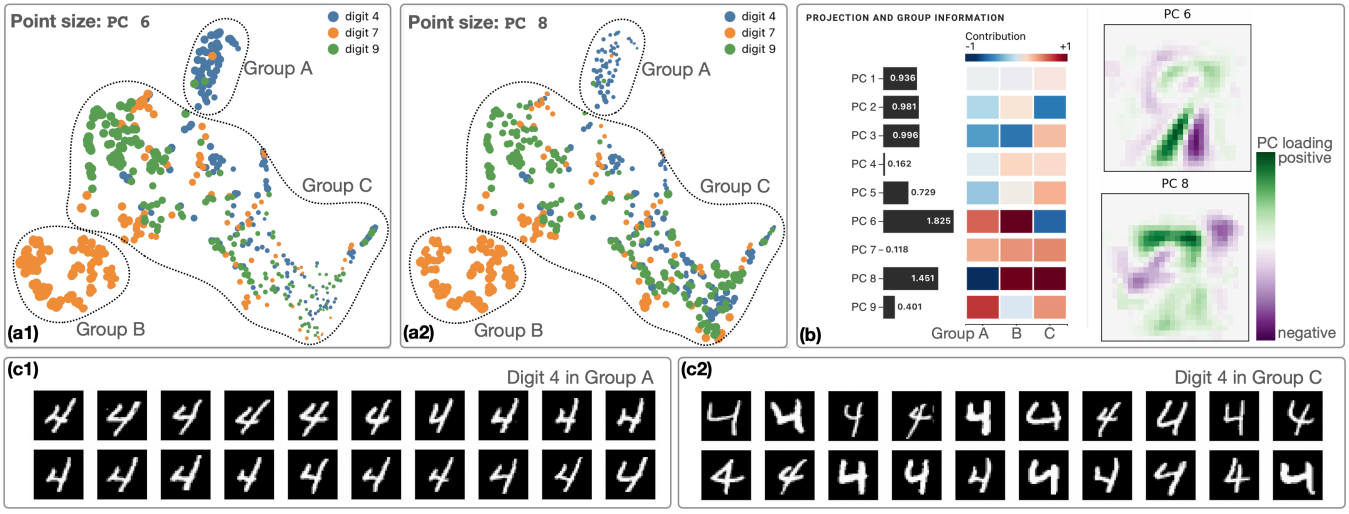


Figure E.6: FEALM-UMAP result 2: (a1, a2) the UMAP results where the sizes of points represent values of PC 6 and PC 8, respectively; (b) the corresponding auxiliary information; and (c1, c2) `digit 4`'s gray-scale images that belong to `Groups A` and `C`, respectively.

Similarly, we review another result shown in Fig. E.7-a. Since we see several clusters, we create `Groups D`–`H` and visualize the auxiliary information, as shown in Fig. E.7-b. For this result, PC 2, PC 5, PC 6, and PC 8 have both large weights and significant contributions. For example, we can expect that `Group D`, which consists of `digits 4`, 7, and 9, tends to contain the green vertical strokes shown in PC 2 and PC 5. In fact, as shown in Fig. E.7-c, the instances belonging to `Group D` have a clear vertical stroke.

This case study demonstrates how we can effectively use FEALM-UMAP even when analyzing a large number of attributes (e.g., over 700 attributes).

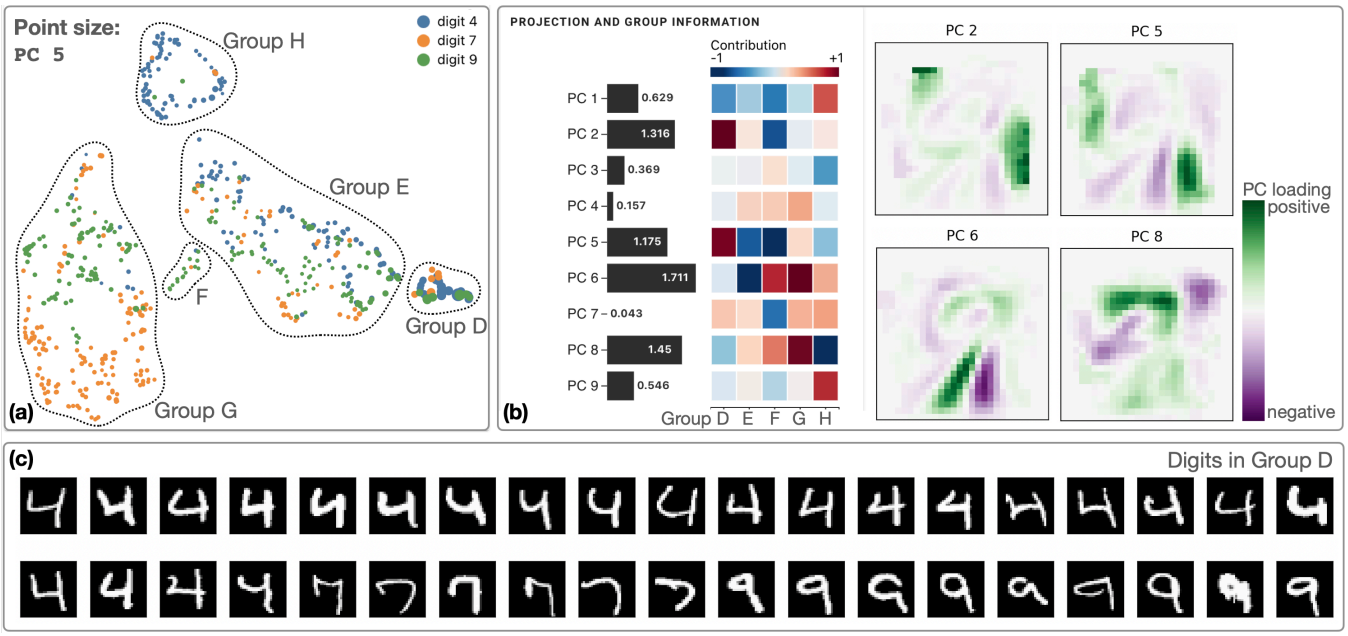


Figure E.7: FEALM-UMAP result 3: (a) the UMAP result where the sizes of points represent values of PC 5; (b) the corresponding auxiliary information; and (c) the gray-scale images of all instances in `Group D`.