Supplement - 'Automatic Parameter Selection for Non-Redundant Clustering'

Collin Leiber[*] Dominik Mautz[*] Claudia Plant[†] Christian Böhm[*]

1 Symbols

We use the following symbols in our paper as well as in the supplement:

Table 1: Description of the used symbols.

Symbol	Interpretation
$N \in \mathbb{N}$	Number of objects
$d \in \mathbb{N}$	Dimensionality of the feature space
$J \in \mathbb{N}$	Number of subspaces
$V \in \mathbb{R}^{d \times d}$	Orthogonal (rotation) matrix
$\delta \in \mathbb{R}$	The precision of the encoding
$I_{m_j} \in \mathbb{N}^{m_j \times m_j}$	$m_j \times m_j$ identity matrix
$X \subseteq \mathbb{R}^d$	Set of all objects
$k_j \in \mathbb{N}$	Number of clusters in subspace j
$m_j \in \mathbb{N}$	Dimensionality of subspace j
$P_j \in \mathbb{N}^{d \times m_j}$	Projection matrix of subspace j
$p_j \in \mathbb{N}$	Number of distribution-specific parameters in subspace j
$X_j \subseteq \mathbb{R}^{m_j}$	X projected to subspace j
$O_j \subseteq X_j$	Set of outliers in subspace j
$\mu_{j,i} \in \mathbb{R}^{m_j}$	Center of cluster i in subspace j
$\Sigma_{j,i} \in \mathbb{R}^{m_j \times m_j}$	Covariance matrix of cluster i in subspace j
$C_{j,i} \subseteq X_j$	Objects of cluster i in subspace j
$J_p \in \mathbb{N}$	Number of predicted subspaces
$Jgt \in \mathbb{N}$	Number of true subspaces
$R^p \in \mathbb{N}^{N \times J_p}$	Prediction labels matrix
$R^{gt} \in \mathbb{N}^{N \times J_{gt}}$	Ground truth labels matrix

2 Encoding the Constant Values

At this point, we would like to give a brief intuition on how the constant components of the encoding strategy we present in the paper could be handled.

The number of objects N and dimensionality d can again be encoded using the natural prior for integers. Therefore, we need $L^0(N) + L^0(d)$ bits to encode these values.

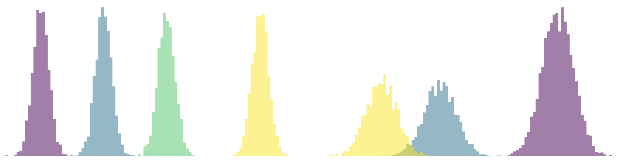
In general real values r can be encoded by separately encoding the integer part $\lfloor r \rfloor$ and the decimal places [5]. Here, the precision δ is required for the decimal places. Hence, the following applies:

$$L(r) = L^0(\lfloor r \rfloor) - \log_2(\delta).$$

This can be used to encode the $d + 1$ values needed to define the hypercube. It can also be used

[*]LMU Munich. {leiber, mautz, boehm}@dbs.ifi.lmu.de

[†]Faculty of Computer Science, ds:UniVie, University of Vienna, Vienna, Austria. claudia.plant@univie.ac.at



(a) Subspace with 4 clusters. (b) Subspace with 3 clusters.

Figure 1: Histograms of two 1-dimensional subspaces.

to encode V. Since all the column vectors of V are orthonormal, they all have a length of 1. Therefore, all values of a column are less than or equal to 1 and we can consequently ignore $L^0(\lfloor r \rfloor)$. Moreover, since the orientation is indifferent, the last entry can be calculated using the first $d-1$ entries. Thereby each column loses one degree of freedom. Furthermore, the orthogonal property means that each following column loses an additional degree of freedom. Therefore, we can encode the first column using $-\log_2(\delta)(d-1)$ bits, the second with $-\log_2(\delta)(d-2)$ bits, and so forth. All in all, the code length of V is $-\log_2(\delta)\frac{d(d-1)}{2}$.

From these encodings, it is very easy to see that the values are actually constants that are independent of a particular clustering result.

3 Search Space Restrictions

In this section, we would like to justify our restrictions on the search space with an example.

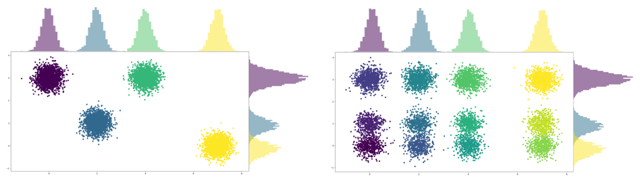
In the paper, we say that we restrict the number of clusters in case of a *cluster space* split as follows:

$$\max(k_{\text{split}_1}, k_{\text{split}_2}) \leq k_{\text{original}} \leq k_{\text{split}_1} \cdot k_{\text{split}_2}.$$

Also, we restrict the number of clusters for a *cluster space* merge with the inverted rule.

$$\max(k_{\text{original}_1}, k_{\text{original}_2}) \leq k_{\text{merge}} \leq k_{\text{original}_1} \cdot k_{\text{original}_2}$$

To better understand these rules, assume that we have the 1-dimensional subspaces shown in Figure 1 with 4 and 3 clusters, respectively. If we want to merge these subspaces, we will always get at least 4 clusters, since 4 clusters are already contained in the first subspace. Moreover, there are a maximum of 12 cluster combinations that can occur. Both extreme situations are illustrated in Figure 2.



(a) Combination with the minimum number of 4 clusters. (b) Combination with the maximum number of 12 clusters.

Figure 2: Two possible combinations of the two subspaces from Figure 1.

With a *cluster space* split, essentially the same applies, but in reverse. Here, no subspace may be created that already has more than the original number of clusters. Furthermore, $k_{\text{split}_1} \cdot k_{\text{split}_2}$ must be greater than or equal to the original number of clusters. If one of these two rules is not met, subspaces would be created that do not fit the structure of the original subspace.

These rules can also be applied to higher dimensional subspaces.

4 Pseudo-code

In order to determine the number of subspaces and clusters within subspaces for non-redundant clustering, the following steps are executed:

- *Noise Space* Split
- *Cluster Space* Split
- *Cluster Space* Merge

Additionally, we regularly combine model parameters to perform a full-space execution. To better understand how all these steps are linked, Algorithm 1 can be analyzed.

5 Implementation Details of AutoNR

We want to give additional information regarding the implementation of *AutoNR*.

Unfortunately, *Nr-Kmeans* introduced another parameter that has to be set by the user. The algorithm optimizes V and P_j through eigenvalue decompositions. Here, the eigenvectors represent the direction, and the signs of the eigenvalues E determine to which subspace the dimensions are assigned. Dimensions not matching the structure of any *cluster space* are assigned to the *noise space*. Consequently, the *noise space* will capture all dimensions corresponding to eigenvalues ≥ 0. Yet, the supplementary of [6] states that eigenvectors with a negative eigenvalue close to zero should also be assigned to the *noise space*. An example value is given in the respective publication. However, the optimal threshold changes depending on the input dataset. We want to avoid such hard thresholds in our approach. Therefore, we utilize the described encoding strategy to determine which dimensions should be contained in the cluster and which in the *noise space*.

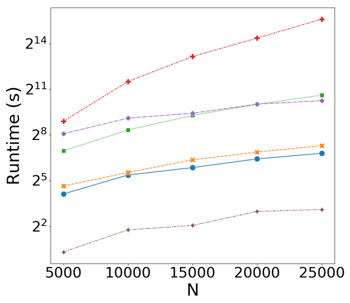
The rotation matrix V can be updated independently of the new subspace dimensionalities. Therefore, V can be calculated a priori and used in the process to define the new m_{cluster} and m_{noise}. The parameters present in the current iteration of *Nr-Kmeans* can be used to calculate the temporary MDL costs of the model. Since the cluster assignments, cluster centers, and scatter matrices stay constant during this operation, only those costs that depend on the subspace dimensionalities m_j and the projections P_j need to be considered. We start with a *cluster space* that only obtains the dimension corresponding to the lowest eigenvalue and a *noise space* containing the other $|E| - 1$ dimensions. The approach is repeated with a rising number of *cluster space* dimensions until the MDL costs exceed the result from the previous iteration or the dimensionality of the *cluster space* equals the number of negative eigenvalues. This means that an initial threshold is no longer necessary

We further utilize the initialization procedure of k-means++ [1] to seed the cluster centers.

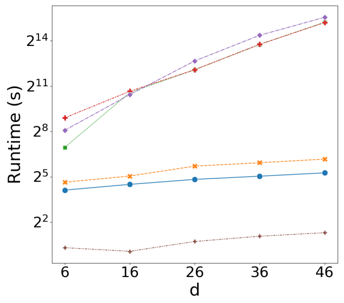
6 Evaluation Setup

Datasets: *syn3* is a synthetic dataset with three subspaces containing 4, 3, and 2 clusters. Each cluster was created using a Gaussian distribution. For *syn3o*, we randomly added 150 uniformly distributed outliers in each subspace. We additionally created the *NRLetters* dataset. It consists of 10000 9×7 RGB images of the letters 'A', 'B', 'C', 'X', 'Y', and 'Z' in the colors pink, cyan, and yellow. Moreover, in each image, a corner pixel is highlighted in the color of the letter. This results in three possible clusterings. An extract of this dataset can be seen in the paper in Figure 1. *Wine* is a real-world dataset from the UCI[1] repository with three clusters. The UCI dataset *Optdigits* consists of 5620 8×8 images, each representing a digit. The *Fruits* [4] dataset was created using 105 images of apples, bananas, and grapes in red, green, and yellow. The images have been preprocessed, resulting in six attributes. The *Amsterdam Library of Object Image*[2] dataset (*ALOI*) contains images of 1000 objects recorded from different angles. For our analysis, we use a common subset of this data consisting of 288 images illustrating the objects 'box' and 'ball'

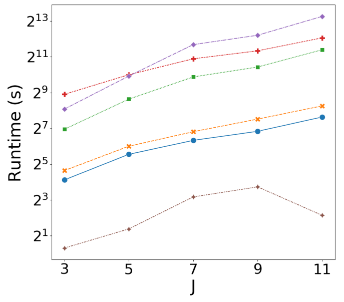
[1]https://archive.ics.uci.edu/ml/index.php
[2]http://aloi.science.uva.nl/



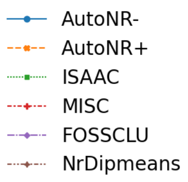
(a) Increasing N ($d = 6, J = 3$).



(b) Increasing d ($N = 5000, J = 3$).



(c) Increasing J ($N = 5000, d = 2J$).



(d) Legend.

Figure 3: Scalability of *AutoNR* without (-) and with (+) outlier detection compared to its competitors. All tests are repeated ten times and the mean is stated.

in the colors green and red. *Dancing Stick Figures* [3] (*DSF*) is a dataset containing 900 20×20 images. It comprises two subspaces describing three upper- and three lower-body motions. *CMUface* is again taken from the UCI repository and is composed of 640 30×32 gray-scaled images showing 20 persons in four different poses (up, straight, left, right). Among those images, 16 show glitches resulting in 624 useful objects. The *WebKB*[3] dataset contains 1041 Html documents from four universities. These web pages belong to one of four categories. We preprocessed the data using stemming and removed stop words and words with a document frequency $< 1\%$. Afterward, we removed words with a variance < 0.25, resulting in 323 features.

Comparison Methods: We compare the results of *AutoNR* without (*AutoNR-*) and with (*AutoNR+*) outlier detection against the parameter-free algorithms *ISAAC* [9] and *MISC* [8] as well as *NrDipmeans* [7]. Furthermore, we extend the subspace clustering approach *FOSSCLU* [2] to iteratively identify new subspaces by removing the subspaces found in previous iterations. For *NrDipmeans* and *FOSSCLU* we have to state the desired number of subspaces. In case of *FOSSCLU* we need to define limits for m_j and k_j. We set those to $1 \leq m_j \leq 3$ and $2 \leq k_j \leq 10$. We wanted to set the upper bound of k_j to 20 for *CMUface*, so *FOSSCLU* would be able to determine all parameters correctly. Unfortunately, this leads to memory issues. Where required, *AutoNR* runs 15 executions of *Nr-Kmeans*. The significance for *NrDipmeans* is set to 0.01.

Experiments are conducted using the Scala implementations of *Nr-Kmeans* and *NrDipmeans* and the Matlab implementations of *ISAAC* and *MISC* as referenced in [6], [7], [9] and [8] respectively. Regarding *FOSSCLU*, we extend the Java version referenced in [2] as described above. *AutoNR* is implemented in Python.

[3]http://www.cs.cmu.edu/ webkb/

7 Runtime Analysis

We conduct runtime experiments on datasets with a rising number of objects N, dimensions d, and subspaces J. The created *cluster spaces* are always two-dimensional and contain three Gaussian clusters each.

All experiments are performed on a computer with an Intel Core i7-8700 3.2 GHz processor and 32GB RAM. The runtime results again correspond to the average of ten consecutive executions. The outcomes are illustrated in Figure 3.

The charts show that our approach is well applicable to high-dimensional datasets. The runtime increases only slightly with additional *noise space* dimensions (3(b)). *ISAAC* and *MISC* have to conduct an ISA which does not scale well to high-dimensional datasets. *FOSSCLU* has to perform Givens rotations multiple times, which is an expensive operation. On the other hand, our framework performs most steps in lower-dimensional subspaces where the overall dimensionality has no significant influence. If additional *cluster spaces* accompany a higher dimensionality, the runtimes of all algorithms behave similarly (3(c)). For large datasets, the differences in runtime are also much less prominent (3(a)). Only *MISC* needs significantly more time because a kernel graph regularized semi-nonnegative matrix factorization has to be performed.

Due to the additional operations required to calculate the outlier distance threshold for each subspace in each iteration, the execution of *AutoNR* with outlier detection expectably takes more time than without. Furthermore, the cluster centers and covariance matrices are updated after each outlier detection procedure.

NrDipmeans is the fastest in all experiments. However, it must be noted that *NrDipmeans* knows the correct number of subspaces and therefore does not need to run tests to identify J. Furthermore, in the case of $J = 11$, it settles with the initial two clusters in

Algorithm 1: Parameter search algorithm.

```
1  Function main(dataset $X$)
2      $V$ = initialize randomly
3      $R_{\text{best}}$ = Initial result (single noise space)
4      // Sort by MDL costs, add noise space last
5      $sortedSpaces$ = sortSubspaces($R_{\text{best}}$)
6      for $j \in sortedSpaces$ do
7          $X_j = \{xVP_j | x \in X\}$
8          if $j$ is cluster space then
9              // Split space into two cluster spaces
10             $j_{s_1}, j_{s_2}$ = clusterSpaceSplit($X_j$, $k_j$)
11         else if $j$ is noise space then
12             // Split space into cluster and noise space
13             $j_{s_1}, j_{s_2}$ = noiseSpaceSplit($X_j$)
14         // Check MDL costs
15         if cost($j_{s_1}$) + cost($j_{s_2}$) < cost($j$) then
16             // Join parameters for full-space execution
17             $V_{\text{tmp}}, P_{\text{tmp}}, \mu_{\text{tmp}}$ = joinParams($R_{\text{best}}, V, j_{s_1}, j_{s_2}$)
18             $R_{\text{tmp}}$, $V_{\text{tmp}}$ = fullSpace($X, V_{\text{tmp}}, P_{\text{tmp}}, \mu_{\text{tmp}}$)
19             // Check full-space MDL costs
20             if cost($R_{\text{tmp}}$) < cost($R_{\text{best}}$) then
21                 $R_{\text{best}} = R_{\text{tmp}}$; $V = V_{\text{tmp}}$
22                 go to line 5
23     $R_{\text{best}}, V$ = merging($R_{best}, V$)
24     if merging was successful then
25         go to line 5
26     return $R_{best}$
27 Function merging($R_{best}, V$)
28     for each pair ($j_1$, $j_2$) of cluster spaces do
29         $X_{j_1,j_2} = \{xVP_{j_1,j_2} | x \in X\}$
30         $j_m$ = clusterSpaceMerge($X_{j_1,j_2}$, $k_{j_1}$, $k_{j_2}$)
31         // Check MDL costs
32         if cost($j_m$) < cost($j_1$) + cost($j_2$) then
33             // Join parameters for the full-space execution
34             $V_{\text{tmp}}, P_{\text{tmp}}, \mu_{\text{tmp}}$ = joinParams($R_{\text{best}}, V, j_m$)
35             $R_{\text{tmp}}, V_{\text{tmp}}$ = fullSpace($X, V_{\text{tmp}}, P_{\text{tmp}}, \mu_{\text{tmp}}$)
36             // Check full-space MDL costs
37             if cost($R_{\text{tmp}}$) < cost($R_{\text{best}}$) then
38                 $R_{\text{best}} = R_{\text{tmp}}$; $V = V_{\text{tmp}}$
39     if better result found then
40         go to line 28
41     return $R_{best}, V$
```

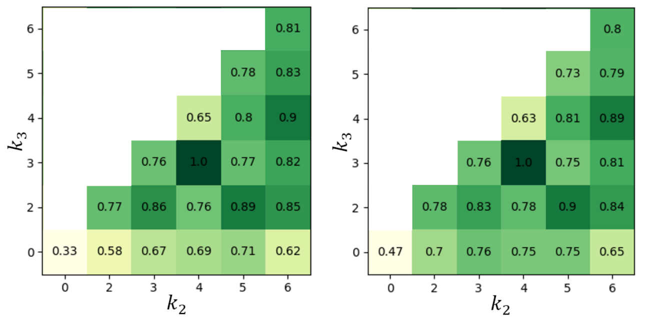


Figure 4: Results of various parametrizations of *Nr-Kmeans* on *NRLetters*. k_1 is set to 6. The left image shows the NMI, and the right the F1 results.

each subspace and does not invest time in finding better structures. Therefore, it seems to have problems running with a high J. *AutoNR*, on the other hand, almost always correctly identifies all clusters in all subspaces.

Our procedure could be further accelerated by, for example, parallelizing the multiple executions of *Nr-Kmeans* with identical parameters.

8 Comparison to *Nr-Kmeans*

We perform additional experiments using the original *Nr-Kmeans* algorithm, to show that the good experimental results are based on our proposal and not merely on the integration of *Nr-Kmeans*. The new results are shown in Table 2. As in the paper, we repeated each experiment ten times and added the average score ± the standard deviation to the table.

AutoNR returns superior results in most experiments, even though *Nr-Kmeans* already knows the correct number of subspaces and clusters for each subspace. Only for the non-redundant dataset *ALOI* does the original *Nr-Kmeans* perform better regarding the F1 score. This case, however, has already been mentioned in the paper. The biggest advantage of our application is the fact that it discovers structures one by one while preserving the ability to adjust already found subspaces. This gives great flexibility, so that possible errors can be compensated in a following iteration. Another advantage is the definition of the *noise space* using MDL (see supplement Section 5), as it can be seen with the datasets *CMUface*, *WebKB*, *NRLetters*, *Wine* and *Optdigits*.

One could argue that the multiple repetitions of *Nr-Kmeans* included in each run of *AutoNR* strongly favor our algorithm. However, we would like to counter this by stating that *Nr-Kmeans* by itself is often unable to achieve a perfect result just once (e.g., for *syn3*). In

Table 2: Results of *AutoNR-* and *AutoNR+* compared to our *Nr-Kmeans* version, where the dimensionality of the *noise space* is determined through MDL (see supplement Section 5), and the original *Nr-Kmeans* implementation on various datasets. The left side shows the NMI results in %, the F1 results in % are shown on the right. All experiments were run ten times, and the average result ± the standard deviation is stated. The best algorithm for each subspace is marked in bold.

		NMI (%)				F1 (%)			
Dataset	**Subspace**	*AutoNR-*	*AutoNR+*	*Nr-Kmeans* (MDL-based *noise space*)	*Nr-Kmeans* (Original)	*AutoNR-*	*AutoNR+*	*Nr-Kmeans* (MDL-based *noise space*)	*Nr-Kmeans* (Original)
syn3	1st (k_j=4)	**100 ± 0**	**100 ± 0**	73 ± 20	59 ± 16	**100 ± 0**	**100 ± 0**	72 ± 18	61 ± 11
(N=5000, d=11)	2nd (k_j=3)	**100 ± 0**	**100 ± 0**	81 ± 13	75 ± 17	**100 ± 0**	**100 ± 0**	82 ± 12	78 ± 16
	3rd (k_j=2)	**100 ± 0**	**100 ± 0**	72 ± 22	75 ± 17	**100 ± 0**	**100 ± 0**	79 ± 16	81 ± 14
syn3o	1st (k_j=4)	86 ± 10	**97 ± 0**	62 ± 17	54 ± 13	83 ± 18	**99 ± 0**	64 ± 14	57 ± 10
(N=5150, d=11)	2nd (k_j=3)	90 ± 10	**96 ± 0**	67 ± 15	69 ± 11	91 ± 17	**99 ± 0**	73 ± 14	75 ± 9
	3rd (k_j=2)	77 ± 20	**94 ± 0**	56 ± 10	68 ± 13	80 ± 27	**99 ± 0**	63 ± 10	77 ± 13
Fruits	Species (k_j=3)	**85 ± 9**	83 ± 7	70 ± 14	74 ± 11	**89 ± 9**	87 ± 7	78 ± 11	78 ± 11
(N=105, d=6)	Color (k_j=3)	17 ± 2	**18 ± 1**	15 ± 2	16 ± 2	**47 ± 3**	44 ± 5	42 ± 3	44 ± 3
ALOI	Shape (k_j=2)	62 ± 4	**64 ± 3**	47 ± 26	54 ± 30	65 ± 2	65 ± 1	73 ± 13	**77 ± 15**
(N=288, d=611)	Color (k_j=2)	62 ± 4	**64 ± 3**	34 ± 0	31 ± 10	65 ± 2	65 ± 1	**66 ± 0**	**66 ± 0**
DSF	Body-up (k_j=3)	**100 ± 0**	**100 ± 0**	70 ± 20	81 ± 26	**100 ± 0**	**100 ± 0**	77 ± 16	85 ± 20
(N=900, d=400)	Body-low (k_j=3)	**100 ± 0**	**100 ± 0**	63 ± 22	56 ± 30	**100 ± 0**	**100 ± 0**	70 ± 17	68 ± 22
CMUface	Identity (k_j=20)	68 ± 4	64 ± 4	**78 ± 6**	75 ± 7	38 ± 4	34 ± 3	**57 ± 9**	52 ± 9
(N=624, d=960)	Pose (k_j=4)	**35 ± 3**	33 ± 1	28 ± 8	26 ± 6	**45 ± 4**	42 ± 4	41 ± 10	37 ± 10
WebKB	Category (k_j=4)	32 ± 2	**34 ± 3**	32 ± 3	30 ± 2	50 ± 5	**58 ± 7**	48 ± 3	48 ± 2
(N=1041, d=323)	University (k_j=4)	56 ± 4	**57 ± 3**	47 ± 8	45 ± 7	51 ± 2	52 ± 3	**54 ± 7**	52 ± 3
NRLetters	Letter (k_j=6)	**100 ± 0**	**100 ± 0**	85 ± 9	83 ± 9	**100 ± 0**	**100 ± 0**	78 ± 13	72 ± 12
(N=10000, d=189)	Color (k_j=3)	**100 ± 0**	**100 ± 0**	52 ± 29	39 ± 25	**100 ± 0**	**100 ± 0**	61 ± 22	52 ± 18
	Corner (k_j=4)	**100 ± 0**	**100 ± 0**	57 ± 26	48 ± 25	**100 ± 0**	**100 ± 0**	61 ± 23	51 ± 19
Wine (N=178, d=13)	Type (k_j=3)	76 ± 5	85 ± 3	**87 ± 2**	79 ± 15	81 ± 6	90 ± 4	**92 ± 2**	87 ± 10
Optdigits (N=5620, d=64)	Digit (k_j=10)	73 ± 1	**74 ± 1**	72 ± 2	70 ± 2	54 ± 4	58 ± 4	**66 ± 3**	64 ± 3

contrast, *AutoNR* often assigns the points to the correct clusters every time. This shows that the iterative identification of subspaces can be beneficial, with the effect becoming stronger the more subspaces there are.

9 Importance of Correct Parametrization

To better assess the importance of a correct parametrization of non-redundant clustering approaches, we perform another experiment. Suppose that we know that *NRLetters* comprises six different letters. We know nothing about the other clustering possibilities. Therefore, we try different parameters for *Nr-Kmeans* using a brute-force search. Here, we assume that no subspace contains more than six clusters. The NMI and F1 results can be seen in Figure 4. To arrive at a single number that indicates the quality of a non-redundant clustering result, we compute the average result over all three subspaces.

$$\text{score}(R^{gt}, R^p) = \frac{1}{J_{gt}} \sum_{1 \le j \le J_{gt}} score_j(R^{gt}, R^p),$$

where $score_j(R^{gt}, R^p)$ is the evaluation method as described in the paper and J_{gt} is the number of label sets in the ground truth. Each run is repeated ten times and the best result is added to the heatmap.

We see that the quality of the results deteriorates away from the optimum ($k_2 = 4, k_3 = 3$) even though we know the correct number of clusters in the first subspace. This shows the value of our framework, which achieved a perfect result in all ten iterations and that, without prior knowledge.

References

[1] D. Arthur and S. Vassilvitskii, *k-means++: The advantages of careful seeding*, in 18th ACM-SIAM SODA, Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[2] S. Goebl, X. He, C. Plant, and C. Böhm, *Finding the optimal subspace for clustering*, in 14th IEEE ICDM, IEEE, 2014, pp. 130–139.

[3] S. Günnemann, I. Färber, M. Rüdiger, and T. Seidl, *Smvc: semi-supervised multi-view clustering in subspace projections*, in 20th ACM SIGKDD, 2014, pp. 253–262.

[4] J. Hu, Q. Qian, J. Pei, R. Jin, and S. Zhu, *Finding multiple stable clusterings*, Knowledge and Information Systems, 51 (2017), pp. 991–1021.

[5] T. C. Lee, *An introduction to coding theory and the two-part minimum description length principle*, International statistical review, 69 (2001), pp. 169–183.

[6] D. Mautz, W. Ye, C. Plant, and C. Böhm, *Discovering non-redundant k-means clusterings in optimal subspaces*, in 24th ACM SIGKDD, 2018, pp. 1973–1982.

[7] D. Mautz, W. Ye, C. Plant, and C. Böhm, *Non-redundant subspace clusterings with nr-kmeans and nr-dipmeans*, ACM Transactions on Knowledge Discovery from Data (TKDD), 14 (2020), pp. 1–24.

[8] X. Wang, J. Wang, C. Domeniconi, G. Yu, G. Xiao, and M. Guo, *Multiple independent subspace clusterings*, in AAAI, vol. 33, 2019, pp. 5353–5360.

[9] W. Ye, S. Maurus, N. Hubig, and C. Plant, *Generalized independent subspace clustering*, in 16th IEEE ICDM, IEEE, 2016, pp. 569–578.