**1. Algorithm Name**

[SubspaceKMeans](https://github.com/tetutaro/subspacekmeans)

**2. Reference**

Dominik M, Wei Y, Claudia P, Christian B. (2017). "Towards an Optimal Subspace for K-Means ". [KDD '17](http://www.kdd.org/kdd2017/), Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining: Pages 365-373, Ludwig-Maximilians-University Munchen, Munich, Germany

**3. Motivation for the algorithm (or which problems it tries to solve?)**

The interpretation of what K-means algorithm finds becomes increasingly difficult with growing number of dimensions. Even in lower dimensional spaces it is sometimes hard to tell what structure the algorithm has found.

SubKmeans, is a technique, which extends k-means. It finds a clustering partition and simultaneously a transformation which highlights the structure found in the dataset.

**4. Short Description:**

With in each iteration step sub-kmeans rotates the data based on the current cluster partition. Than, it splits the new feature space in to two orthogonal subspaces first subspace represents the clusters and we call it the cluster subspace. The second space is the complementing space and it is assumed to be unimodal of no cluster structure containing only noisy features. The dimensionality of these two subspaces is optimized in the same time. So sub-kmeans rotates and partitions the data such that the clustered space is a permanent cluster structure and contains only relevant features.

**5. Pseudo-Code**

Remarks:

1. **Don’t copy the code/equations as figures – write your own equations using MS Equation Editor**
2. **Provide two pseudo-code: one for build and one for classify**
3. **Use proper indentation + lines number**
4. **Use the following notations:**
5.  - represent the training set which contains *m* instances
6. - represents the set of classes.
7. *I* – represent an inducer (learning algorithm/base learner)
8. *Mt* – a classifier (classification model) that was trained.

**MultiBoost- Building the ensemble**

Input: *S* – a labeled training set

*I* – a based indcuer

*T* – number of iterations

*J* – vector of integers specifying the iteration at which each subcommittee

should terminate.

1. S’ 🡨S with instance weights assigned to be 1.

2. k🡨1

3. FOR t=1 to T

4. IF *Jk*=t THEN

5. reset S’ to random weights drawn from continuous Poisson distribution.

6. standardize S’ to sum to m.

7. k++

8. END IF

9. *Mt=I(S’)*

10. 

11. IF  THEN

12. reset S’ to random weights drawn from continuous Poisson distribution.

13. standardize S’ to sum to m.

14. k++

15. GOTO 9

16. ELSE IF  THEN

17. 

18. reset S’ to random weights drawn from continuous Poisson distribution.

19. standardize S’ to sum to m.

20. k++

21. ELSE

22. 

23. FOR each 

24. IF THEN

25. 

26. ELSE

27. 

28. END IF

29. IF THEN

30. 

31. END IF

32. END FOR

33. END IF

33. END FOR

**MultiBoost- Classify an instance**

Input: *x* – an instance needed to be labeled

1. Return 

Figure 1: The MultiBoost Pseudo Code

**5+6. Pseudo-Code and Algorithm Explanation:**

|  |  |
| --- | --- |
| **Symbol** | **Interpretation** |
|  | Dimensionality of original space |
|  | Dimensionality of the clustered space |
|  | Number of Clusters |
|  | Set of all objects |
|  | Set of objects assigned to cluster i |
|  | A data point or object |
|  | Dataset mean in the original space |
|  | Mean of cluster i in the original space |
|  | Scatter matrix of the dataset in the original space |
|  | Scatter matrix of cluster i in the original space |
|  | . We perform an eigenvalue decomposition of Σ and use the eigenvectors—sorted ascending to their corresponding eigenvalue—as columns in V . |
|  | Projection onto the first m attributes |
|  | Projection onto the last d −m attributes |
|  | (orthonormal) matrix of a rigid transformation |
|  | l × l identity matrix |
|  | l × r zero matrix |

|  |  |  |
| --- | --- | --- |
| 1 | |  |
| Explanation | The input of the subkmeans algorithm is: 1) the dataset D  2) the number of clusters K | |
| 2 | |  |
| Explanation | | The output of subkmeans is:  1) k clusters  2) the rotation matrix (which is the eigenvector matrix)  3) the dimensionality of the clustered space |
| 3 | |  |
| Explanation | | Initializing a random orthonormal matrix. That means: . |
| 4 | |  |
| Explanation | | Initializing the number of dimensions to be used in a cluster after dimension reduction process. This value will be changed and converged. |
| 5 | |  |
| Explanation | | Calculating the mean vector of all data set D. |
| 6 | |  |
| Explanation | | Calculating the Scatter matrix which is used to make [estimates](https://en.wikipedia.org/wiki/Estimation_of_covariance_matrices) of the [covariance matrix](https://en.wikipedia.org/wiki/Covariance_matrix). The covariance matrix is used in the eigendecomposition process. |
| 7 | |  |
| Explanation | | For each K, which will become a cluster we choose a random point that based upon her the cluster will start building itself. |
| 8 | | **While (**number of dimensions decreases): |
| Explanation | | Start while loop that stops when the number of dimensions is stops decreasing after dimension reduction of the clusters. |
|  | | //Assignment step |
| 8 | |  |
| Explanation | | Assign to every cluster an empty group. |
| 9 | |  |
| 10 | |  |
| 11 | |  |
| Explanation | | Assign every instance to a cluster. Do it by finding the cluster that is the closest to the instance based on the transformed dimensions. Using calculations of eigendecomposition projection differences of X and the cluster mean point using the first m eigenvalues. |
|  | | //Update step |
| 12 | |  |
| Explanation | | **For** I in 1:k (for every cluster) |
| 13 | |  |
| Explanation | | Evaluate the mean of the I’th cluster |
| 14 | |  |
| Explanation | | Evaluate the scatter matrix/covariance matrix of the I’th cluster |
|  | | //Eigendecomposition: |
|  | | // Ɛ: list if eigenvalues in ascending order |
|  | | // V: corresponding eigenvector |
| 15 | |  |
| Explanation | | Eigen decomposition – calculate eigenvectors and eigenvalues |
| 16 | |  |
| Explanation | | Setting m to the number of negative eigenvalues of Σ |
|  | | **End for** |
|  | | **End while** |
|  | |  |

**7. Illustration**

* Provide a figure(s) that can explain the essence of how the algorithm works.
* In addition, you need to add a short running example of how the algorithm works on small a data sample (e.g. demonstrate how the algorithm converges in the first iterations) – see the following example:

In this section we illustrate the first four iterations of MultiBoosting algorithm using the following settings T=8 and J= {4,7}. The decisionStump algorithm is chosen to be the base inducer. A simplified version of the **labor** dataset (Table 1) is used as the training set. Initially all instances have the same weight. The first decisionStump classifier that was induced is presented in Figure 2 with error level . Thus, according to line 22 the weight of the first classifier is set to  and the instances are reweighted according to procedure indicated in lines 23-32. The new instances weights are presented in the second column in Table 2. Note that the weights of the correctly classified instances decreased to 0.59375 while the weights of the incorrectly classified instances increased to 3.166667.

The second decisionStump is presented in Figure 2 with error level  and . The updated weights are presented in the third column in Table 2.

The third decisionStump is presented in Figure 2. Because the condition in line 4 is met, the new weights are drawn from continuous Poisson distribution. The new weights are presented in the fourth column of Table 2 and are used to induce the fourth classifier presented in Figure 2. Note that the attribute selected in the fourth classifier is equal to the attribute used in the third classifier. Still the two classifiers are not identical. Each classifier uses different instances’ weights. Thus, each classifier provides a different class distribution.

|  |  |  |
| --- | --- | --- |
| Iteration 1: Decision Stump; | | |
|  | Class Distribution | |
|  | bad | good |
| wage-increase-first-year <= 2.65 | 0.8667 | 0.1333 |
| wage-increase-first-year > 2.65 | 0.1707 | 0.8293 |
| wage-increase-first-year is missing | 0 | 1 |

|  |  |  |
| --- | --- | --- |
| Iteration 2: Decision Stump; | | |
|  | Class Distribution | |
|  | bad | good |
| longterm-disability-assistance = yes | 0.4141 | 0.5859 |
| longterm-disability-assistance != yes | 1 | 0 |
| longterm-disability-assistance is missing | 0.21 | 0.79 |

|  |  |  |
| --- | --- | --- |
| Iteration 3: Decision Stump; | | |
|  | Class Distribution | |
|  | Bad | Good |
| statutory-holidays <= 10.5 | 0.9521 | 0.0479 |
| statutory-holidays > 10.5 | 0.434 | 0.566 |
| statutory-holidays is missing | 0 | 1 |

|  |  |  |
| --- | --- | --- |
| Iteration 4: Decision Stump; | | |
|  | Class Distribution | |
|  | Bad | Good |
| statutory-holidays <= 10.5 | 0.9021 | 0.0979 |
| statutory-holidays > 10.5 | 0.2214 | 0.7786 |
| statutory-holidays is missing | 0 | 1 |

Figure 2: The first four DecisionStumps obtained by the MultiBoosting algorithm

Table 1: Initial dataset

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| duration | wage-increase-first-year | statutory-holidays | vacation | longterm-disability-assistance | contribution-to-dental-plan | bereavement-assistance | contribution-to-health-plan | class |
| 1 | 5 | 11 | average | ? | ? | yes | ? | good |
| 2 | 4.5 | 11 | below\_average | ? | full | ? | full | good |
| ? | ? | 11 | generous | yes | half | yes | half | good |
| 3 | 3.7 | ? | ? | ? | ? | yes | ? | good |
| 3 | 4.5 | 12 | average | ? | half | yes | half | good |
| 2 | 2 | 12 | average | ? | ? | ? | ? | good |
| 3 | 4 | 12 | generous | yes | none | yes | half | good |
| 3 | 6.9 | 12 | below\_average | ? | ? | ? | ? | good |
| 2 | 3 | 11 | below\_average | yes | half | yes | ? | good |
| 1 | 5.7 | 11 | generous | yes | full | ? | ? | good |
| 3 | 3.5 | 13 | generous | ? | ? | yes | full | good |
| 2 | 6.4 | 15 | ? | ? | full | ? | ? | good |
| 2 | 3.5 | 10 | below\_average | no | half | ? | half | bad |
| 3 | 3.5 | 13 | generous | ? | full | yes | full | good |
| 1 | 3 | 11 | generous | ? | ? | ? | ? | good |
| 2 | 4.5 | 11 | average | ? | full | yes | ? | good |
| 1 | 2.8 | 12 | below\_average | ? | ? | ? | ? | good |
| 1 | 2.1 | 9 | below\_average | yes | half | ? | none | bad |
| 1 | 2 | 11 | average | no | none | no | none | bad |
| 2 | 4 | 15 | generous | ? | ? | ? | ? | good |
| 2 | 4.3 | 12 | generous | ? | full | ? | full | good |
| 2 | 2.5 | 11 | below\_average | ? | ? | ? | ? | bad |
| 3 | 3.5 | ? | ? | ? | ? | ? | ? | good |
| 2 | 4.5 | 10 | generous | ? | half | ? | full | good |
| 1 | 6 | 9 | generous | ? | ? | ? | ? | good |
| 3 | 2 | 10 | below\_average | ? | half | yes | full | bad |
| 2 | 4.5 | 10 | below\_average | yes | none | ? | half | good |
| 2 | 3 | 12 | generous | ? | ? | yes | full | good |
| 2 | 5 | 11 | below\_average | yes | full | yes | full | good |
| 3 | 2 | 10 | average | ? | ? | yes | full | bad |
| 3 | 4.5 | 11 | average | ? | half | ? | ? | good |
| 3 | 3 | 10 | below\_average | yes | half | yes | full | bad |
| 2 | 2.5 | 10 | average | ? | ? | ? | ? | bad |
| 2 | 4 | 10 | below\_average | no | none | ? | none | bad |
| 3 | 2 | 10 | below\_average | no | half | yes | full | bad |
| 2 | 2 | 11 | average | yes | none | yes | full | bad |
| 1 | 2 | 11 | generous | no | none | no | none | bad |
| 1 | 2.8 | 9 | below\_average | yes | half | ? | none | bad |
| 3 | 2 | 10 | average | ? | ? | yes | none | bad |
| 2 | 4.5 | 12 | average | yes | full | yes | half | good |
| 1 | 4 | 11 | average | no | none | no | none | bad |
| 2 | 2 | 12 | generous | yes | none | yes | full | bad |
| 2 | 2.5 | 12 | average | ? | ? | yes | ? | bad |
| 2 | 2.5 | 11 | below\_average | ? | ? | yes | ? | bad |
| 2 | 4 | 10 | below\_average | no | none | ? | none | bad |
| 2 | 4.5 | 10 | below\_average | no | half | ? | half | bad |
| 2 | 4.5 | 11 | average | ? | full | yes | full | good |
| 2 | 4.6 | ? | ? | yes | half | ? | half | good |
| 2 | 5 | 11 | below\_average | yes | ? | ? | full | good |
| 2 | 5.7 | 11 | average | yes | full | yes | full | good |
| 2 | 7 | 11 | ? | yes | full | ? | ? | good |
| 3 | 2 | ? | ? | yes | half | yes | ? | good |
| 3 | 3.5 | 13 | generous | ? | ? | yes | full | good |
| 3 | 4 | 11 | average | yes | full | ? | full | good |
| 3 | 5 | 11 | generous | yes | ? | ? | full | good |
| 3 | 5 | 12 | average | ? | half | yes | half | good |
| 3 | 6 | 9 | generous | yes | full | yes | full | good |

Table 2: Weights of instances in multiboosting.

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration 1 | Iteration 2 | Iteration 3 | Iteration 4 |
| 1 | 0.59375 | 0.378319 | 0.282369 |
| 1 | 0.59375 | 0.378319 | 0.802696 |
| 1 | 0.59375 | 0.378319 | 1.415129 |
| 1 | 0.59375 | 0.378319 | 0.990933 |
| 1 | 0.59375 | 0.378319 | 0.029601 |
| 1 | 3.166667 | 2.017699 | 4.589053 |
| 1 | 0.59375 | 0.378319 | 0.033378 |
| 1 | 0.59375 | 0.378319 | 0.055931 |
| 1 | 0.59375 | 0.378319 | 0.048936 |
| 1 | 0.59375 | 0.378319 | 0.058601 |
| 1 | 0.59375 | 0.378319 | 0.83149 |
| 1 | 0.59375 | 0.378319 | 0.951744 |
| 1 | 3.166667 | 2.017699 | 1.102148 |
| 1 | 0.59375 | 0.378319 | 0.612591 |
| 1 | 0.59375 | 0.378319 | 1.939929 |
| 1 | 0.59375 | 0.378319 | 0.234793 |
| 1 | 0.59375 | 0.378319 | 0.374361 |
| 1 | 0.59375 | 1.379032 | 1.668619 |
| 1 | 0.59375 | 0.378319 | 0.875561 |
| 1 | 0.59375 | 0.378319 | 1.771881 |
| 1 | 0.59375 | 0.378319 | 0.327757 |
| 1 | 0.59375 | 1.379032 | 0.195141 |
| 1 | 0.59375 | 0.378319 | 4.766106 |
| 1 | 0.59375 | 0.378319 | 0.583465 |
| 1 | 0.59375 | 0.378319 | 0.266366 |
| 1 | 0.59375 | 1.379032 | 1.757438 |
| 1 | 0.59375 | 0.378319 | 0.65771 |
| 1 | 0.59375 | 0.378319 | 0.547333 |
| 1 | 0.59375 | 0.378319 | 0.495068 |
| 1 | 0.59375 | 1.379032 | 1.427354 |
| 1 | 0.59375 | 0.378319 | 0.425636 |
| 1 | 3.166667 | 7.354839 | 1.52083 |
| 1 | 0.59375 | 1.379032 | 4.086921 |
| 1 | 3.166667 | 2.017699 | 1.644271 |
| 1 | 0.59375 | 0.378319 | 1.553856 |
| 1 | 0.59375 | 1.379032 | 0.554234 |
| 1 | 0.59375 | 0.378319 | 0.023963 |
| 1 | 3.166667 | 7.354839 | 1.264916 |
| 1 | 0.59375 | 1.379032 | 0.837525 |
| 1 | 0.59375 | 0.378319 | 1.373259 |
| 1 | 3.166667 | 2.017699 | 0.755798 |
| 1 | 0.59375 | 1.379032 | 1.311098 |
| 1 | 0.59375 | 1.379032 | 0.105108 |
| 1 | 0.59375 | 1.379032 | 2.936797 |
| 1 | 3.166667 | 2.017699 | 0.47154 |
| 1 | 3.166667 | 2.017699 | 0.380823 |
| 1 | 0.59375 | 0.378319 | 1.910357 |
| 1 | 0.59375 | 0.378319 | 0.384537 |
| 1 | 0.59375 | 0.378319 | 0.014317 |
| 1 | 0.59375 | 0.378319 | 1.419375 |
| 1 | 0.59375 | 0.378319 | 0.884058 |
| 1 | 3.166667 | 2.017699 | 0.692734 |
| 1 | 0.59375 | 0.378319 | 0.988519 |
| 1 | 0.59375 | 0.378319 | 0.732741 |
| 1 | 0.59375 | 0.378319 | 0.616774 |
| 1 | 0.59375 | 0.378319 | 0.001008 |
| 1 | 0.59375 | 0.378319 | 0.415522 |

**8. Strengths**

1. *“The non-optimized version is easy to implement and only uses standard features provided by all data computing frameworks.” – It is very simple to implement since it uses no additional parameters in order to receive the non-optimized clustering result.*
2. *“it is fast, though this could most likely further increased – The “time consumer” is the calculation of the covariance matrix, which is an o(n^2 ). Means, even when a very large dataset, the time of processing is quite fast.*

**9. Drawbacks**

1. *“It should be noted that this value may not be chosen too big or too small, as it balances the probability that the initial clustered space contains parts of the cluster structure we want to find and the expressiveness of the similarity measure. For simplicity, we use in our implementation d/2 as it depends on d. Yet, it is also possible to let the user choose an initial value. The optimal value form is subsequently found during the optimization.” –* Since they choose m without reference to the eigenvalues before starting the iterations, the convergence process might be very long. Instead, they should have chosen the m value using the eigenvalues.
2. *“Last but not least, the initial cluster centers could for example be picked randomly from the dataset or set by k-means++”. –* Again, it is inefficient to choose center points randomly. There are better ways to choose the initial centroids such in k-means++.
3. *“Future efforts may be directed towards the approximation of the transformation matrix V. We are very confident that this can be achieved quite easily by using a randomized singular value decomposition, resulting in further performance improvements for very high-dimensional datasets.” – The convergence process can get a lot faster when the transformation matrix V is “decomposed”, especially when we have high dimensional dataset.*

**10. Experimental Results**

A. Measures (Accuracy and Runtime):

1. Accuracy (Percent Correct)
2. Area under Roc (AUC)
3. Execution Time
4. …

B. Base Classifiers – Think about all the baselines that are relevant for your project

1. DecisionStump

2. J4.8

3. Logistic

4. BayesNet

C. Ensemble size (or any other important parameters of the algorithm)

1. 5

2. 10

3. 15

4. 20

D. Baselines - Think about all the baselines that are relevant for your project, for example If we suggest a new ensemble algorithm We should compare it to other ensembles such as Random Forest and Adaboost, as well to single strong learner like ANN.

E. Datasets

E. Compare to AdaBoost performance using t-test

1. 5 runs of 10 fold cross validation.

2. 16 Tables – each include the average, standard deviation, and an indication for significance.

**Results for Ensemble Size 5**

Accuracy (Percent Correct)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dataset | Adaboost | | MultiBoost | | Statistical Significance |
| Accuracy | Standard Deviation | Accuracy | Standard Deviation |
| audio.symbolic | 47 | 2.47 | 47 | 2.47 |  |
| breast-cancer | 70.63 | 8.65 | 71.12 | 8.09 |  |
| wisconsin-breast-cancer | 94.85 | 2.35 | 92.42 | 3.77 | \* |
| horse-colic | 82.76 | 5.94 | 81.51 | 5.73 |  |
| contact-lenses | 73 | 25.4 | 66 | 32.64 |  |
| credit-rating | 85.54 | 3.76 | 85.51 | 3.78 |  |
| german\_credit | 70.68 | 3.6 | 70 | 0 |  |
| pima\_diabetes | 73.8 | 5.21 | 72.24 | 5.33 |  |
| Glass | 44.9 | 3.04 | 44.9 | 3.04 |  |
| cleveland-14-heart-diseas | 82.7 | 6.55 | 74.98 | 7.39 | \* |
| hungarian-14-heart-diseas | 81.99 | 6.99 | 80.91 | 7.52 |  |
| heart-statlog | 82.07 | 6.95 | 74.59 | 8.56 | \* |
| hepatitis | 79.51 | 9.35 | 78.2 | 6.65 |  |
| hypothyroid | 92.9 | 0.86 | 95.39 | 0.63 | v |
| ionosphere | 88.83 | 5.53 | 82.8 | 5.37 | \* |
| iris | 93.73 | 5.46 | 66.67 | 0 | \* |
| 'ISUKSHONE.csv-weka.filte | 73.63 | 0.49 | 73.6 | 0.54 |  |
| kr-vs-kp.csv-weka.filters | 86.8 | 1.98 | 66.05 | 1.73 | \* |
| labor-neg-data | 84.67 | 15.24 | 77.6 | 14.33 |  |
| LED.symbolic | 20.27 | 3.07 | 20.27 | 3.07 |  |
| letter.symbolic | 7.01 | 0.21 | 7.01 | 0.21 |  |
| lung-cancer.csv-weka.filt | 50.33 | 17.41 | 46 | 23.22 |  |
| lymphography | 75.1 | 10.2 | 74.3 | 11.08 |  |
| monks1.symbolic | 73.44 | 8.01 | 73.44 | 8.01 |  |
| monks2.symbolic | 57.39 | 8.76 | 59.4 | 7.08 |  |
| monks3.symbolic | 92.6 | 8.38 | 72.32 | 12.69 | \* |
| mushroom.symbolic | 96.09 | 0.67 | 88.68 | 1.11 | \* |
| nurse.symbolic | 66.25 | 0.04 | 66.25 | 0.04 |  |
| optg.symbolic | 19.32 | 0.41 | 19.32 | 0.41 |  |
| sick | 97.17 | 0.84 | 96.55 | 0.97 | \* |
| sonar.symbolic | 65.99 | 8.54 | 63.46 | 7.6 |  |
| soybean | 27.97 | 2.16 | 27.97 | 2.16 |  |
| splice | 79.97 | 1.68 | 62.38 | 1.58 | \* |
| TTT | 74.43 | 3.9 | 69.94 | 4.28 | \* |
| vehicle | 39.76 | 1.62 | 39.76 | 1.62 |  |
| vote.symbolic | 95.45 | 3.84 | 95.86 | 3.88 |  |
| vowel | 17.43 | 0.81 | 17.43 | 0.81 |  |
| waveform | 63.37 | 3.41 | 57.08 | 1.48 | \* |
| wine.symbolic | 58.8 | 6.54 | 58.01 | 7.38 |  |
| zoo.symbolic | 60.4 | 2.44 | 60.4 | 2.44 |  |

AUC

**….**

Execution Time

**….**

**Results for Ensemble Size 10**

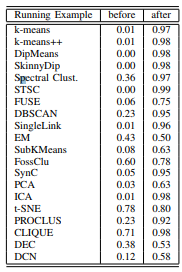
**….**

**11. Conclusions**

What are the conclusions from the results?

**12. Citations**

The paper was cited in 4 other papers:

1. Jang, H. J., Kim, B., Kim, J., & Jung, S. Y. (2018). An Efficient Grid-Based K Prototypes Algorithm for Sustainable Decision-Making on Spatial Objects. Sustainability, 10(8), 2614.‏ - They used sub k-means to reduce complexity of the basic k-means algorithm
2. Schelling, B., & Plant, C. (2018, November). DipTransformation: Enhancing the Structure of a Dataset and thereby improving Clustering. In 2018 IEEE International Conference on Data Mining (ICDM) (pp. 407-416). IEEE.‏ - They compared different techniques that intend reducing dimensionality with the goal of finding a subspace compatible with k-means. The results showed Sub-k-means is not the optimal algorithm:   
   
3. Wang, J., Hsieh, C. J., & Shi, D. (2018, May). NLRR++: Scalable Subspace Clustering via Non-Convex Block Coordinate Descent. In Proceedings of the 2018 SIAM International Conference on Data Mining (pp. 28-36). Society for Industrial and Applied Mathematics.‏ - They mentioned the sub-kmeans clustering method as a past decade method with some other subspace clustering methods.
4. Mautz, D., Ye, W., Plant, C., & Böhm, C. (2018, July). Discovering Non-Redundant K-means Clusterings in Optimal Subspaces. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 1973-1982). ACM.‏ - “ Slightly different to the above-described methods is the recently proposed algorithm SubKmeans. It aims to combine the kmeans algorithm with a simultaneous dimensionality reduction step to find one arbitrarily oriented subspace. It can be seen as a special instance of Nr-Kmeans, in which we only assume a single clustered space with a single clustering and a complementing noise space.”