Ostfalia
Hochschule für angewandte
Wissenschaften

Fakultät Informatik

Abgabe Gesprächs- und Verhandlungsführung

Carina Illgen, Paloma ...
Matrikel-Nr. 70486617, ...

Abgabe im Studiengang Informatik
Wintersemester 2025/26
für das Seminar Gesprächs- und Verhandlungsführung

Ostfalia Hochschule für angewandte Wissenschaften

Prüferin: Mona Saleh

Salzgitter · Suderburg · **Wolfenbüttel** · Wolfsburg

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere, dass ich alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe. Dies gilt explizit auch für die Verwendung von text- oder code-generierenden KI-Werkzeugen. Die eingereichte Arbeit ist weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen. Ich habe zur Kenntnis genommen, dass die Arbeit einer elektronischen Plagiatsprüfung unterzogen werden kann.

Ort, Datum	Unterschrift

Contents

Abkürzungsverzeichnis

- OSC...Orthographic Star Coordinates
- SC...Star Coordinates
- CO...Composition Operators
- LSS...Least Square Solution
- DSC...Distance Consistency
- CD...Centroid Density
- CDC...Centroid Distance Change
- VML...Visual Machine Learning

1. Introduction

Your Introduction goes here...

2. Related Work

Your related work goes here...

2.1. Visual Analytics

Figure 1 shows ...

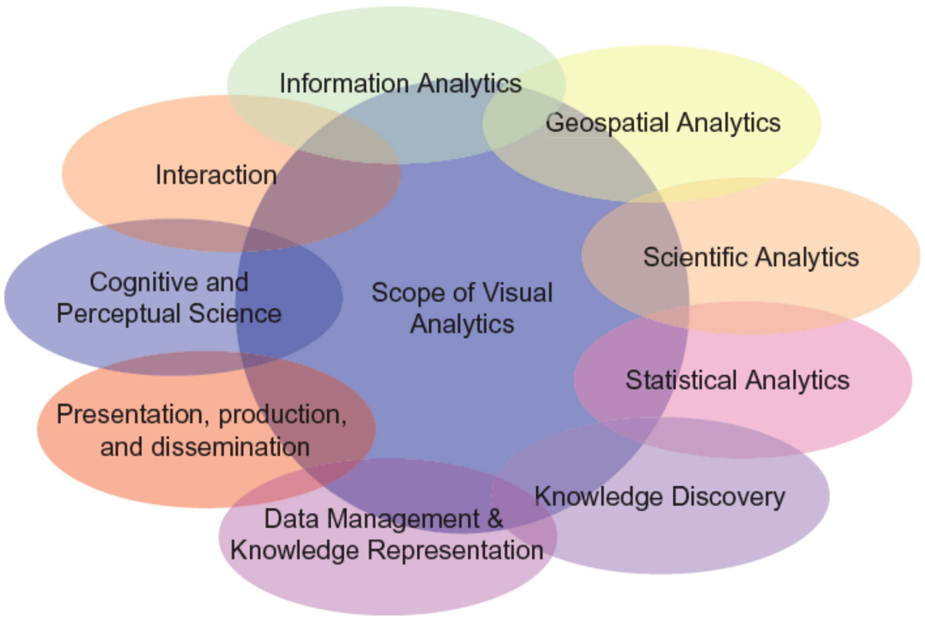
Information Analytics
Geospatial Analytics
Interaction
Scientific Analytics
Cognitive and Perceptual Science
Scope of Visual Analytics
Statistical Analytics
Presentation, production, and dissemination
Knowledge Discovery
Data Management & Knowledge Representation

Figure 1.: The Scope of Visual Analytics [15]

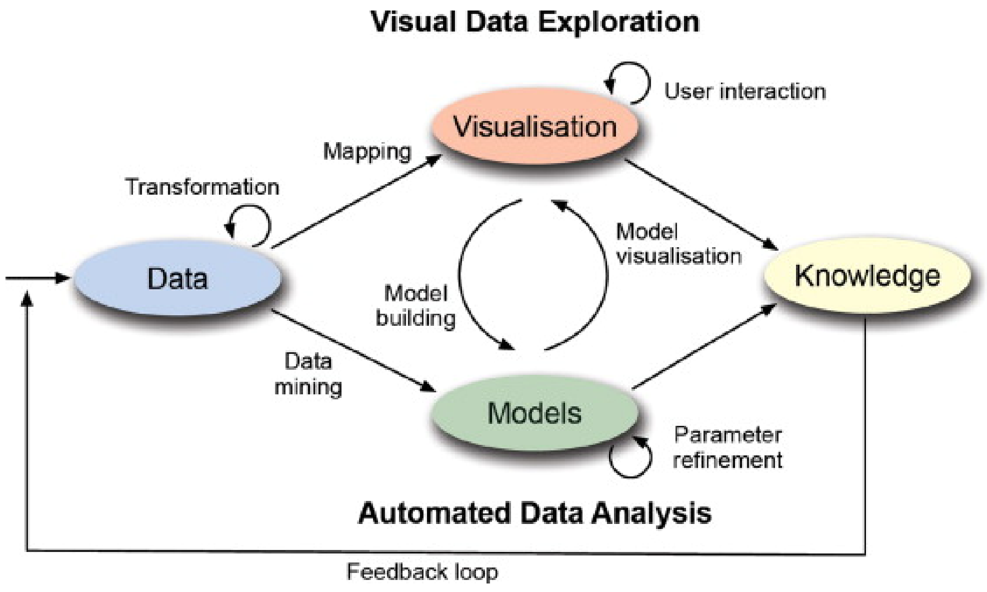

Figure 2.: The Visual Analytics Process [16]

2.2. Non-linear Projections

[22]
[6]
Isomap [23]
SNE [10]
Laplacian Eigenmaps [2]
[24]
t-SNE [24]
UMAP [18]
[12]
[13]
[11]
[5]
[14]
[21]
[7]
[17]
[9]

[3]
[15]
[15]
[16]

2.3. Heuristics

P.M. Todd defines heuristics as:

> ’[...]approximate strategies or ‘rules of thumb’ for decision making and problem solving that do not guarantee a correct solution but that typically yield a reasonable solution or bring one closer to hand. As such, they stand in contrast to algorithms that will produce a correct solution given complete and correct inputs.[...]’ [19]

Table 1 shows a comparison between heuristics and complete search:

	heuristics	complete search
computation	fast	slow
solution	error prone	exact
mathematically provable	in most cases no	yes
based on	intuition, exploration, guesses	finite set of instructions

Table 1.: Comparison between heuristics and algorithms

According to Ankerst et al. [1]

1. pattern recognition capabilities of human brain can increase the effectiveness
2. deeper understanding of the results and thus more trust into the system
3. domain knowledge by the user can lead to better results and avoid overfitting

Rauber et al. [20] Heidari et al. [8] Chu et al. [4]

3. Background

$1_n = (1\ 1\ ...\ 1)^T$ is defined the one column vector, I_n is an $n \times n$ identity matrix. $\|a\|_2$ is the euclidean norm of a column vector a calculated as:

$$\|a\|_2 = \sqrt{\sum_i^n a_i^2} \tag{3.1}$$

It is defined as:

$$D = (d_1\ d_2 ... d_m)\ with\ d = (d_1\ d_2 ... d_n)^T \tag{3.2}$$

$$M = (m_1\ m_2 ... m_q) \tag{3.3}$$

$$Q = (q_1\ q_2 ... q_k)\ with\ M \cap Q = \emptyset \tag{3.4}$$

Their corresponding projections are P_m respectively P_q.

With D' being all data records that are neither sticky nor shift-able, our dataset D is composed of the following components:

$$\underbrace{\overbrace{D}^{Data}}_{n\times m} = \underbrace{D'}_{n\times(m-q-k)} \cup \underbrace{\overbrace{M}^{Shift\text{-}able\ \ Data}}_{n\times q} \cup \underbrace{\overbrace{Q}^{Sticky\ \ Data}}_{n\times k} \tag{3.5}$$

Thus, M_ε equals M. are defined by two polynomials ${f_1}^\delta$ and ${f_2}^\delta$ of degree 2 (see 3.6).

$${f_1}^\delta = 3 \cdot \delta - 2\delta^2,\ {f_2}^\delta = 1 - 4 \cdot \delta + 4\delta^2 \tag{3.6}$$

... as shown in Formula (3.7).

$$\overline{A} = \left(\frac{1}{1 + \|m\|_2^2}\right) \Delta p \cdot m^T + A \tag{3.7}$$

... shift-able sets is calculated by Formula (3.8) and Formula (3.9).

$$\overline{A_\varepsilon} = \left[\Delta p \cdot (M_\varepsilon \cdot 1_q)^T \cdot H_\varepsilon^{-1}\right] + A \tag{3.8}$$

with

$$H_\varepsilon = (M_\varepsilon M_\varepsilon^T + I_n) \tag{3.9}$$

Formula (3.10) with Formula (3.12) shows the Composition Operator without memory and with control ε and blending δ.

$$\overline{A}^- = \left[f_1^\delta \cdot \Delta p \cdot (M_\varepsilon \cdot 1_q)^T \cdot H_{\varepsilon\delta}^{-1}\right] + A \tag{3.10}$$

Formula (3.11) with Formula (3.12) shows the Composition Operator with memory, control ε and blending δ.

$$\overline{A}^+ = \left[f_1^\delta \cdot \Delta p \cdot (M_\varepsilon \cdot 1_q)^T + f_2^\delta \cdot P_q Q^T) + A(I_N + f_1^\delta \cdot M_\varepsilon M_\varepsilon^T\right] H_{\varepsilon\delta}^{-1} \tag{3.11}$$

with

$$H_{\varepsilon\delta} = (f_1^\delta \cdot M_\varepsilon M_\varepsilon^T + f_2^\delta \cdot QQ^T + I_n) \tag{3.12}$$

This shift vector Δp is influenced by three different components.

1. distance d_{ij} between a chosen centroid C_i and its nearest centroid C_j
2. distance d_{iC} between a chosen centroid C_i and the central centroid C_C
3. randomly generated noise factor, following a normal distribution between -1 and 1

The shift vector Δp is calculated by the following Formula (3.13).

$$\Delta p = \underbrace{C_i - C_j}_{d_{ij}} + \frac{100}{100+n} \cdot \underbrace{C_i - C_C}_{d_{iC}} + \frac{100}{100+n} \cdot noise \tag{3.13}$$

as shown in Formula (3.14).

$$\Delta p = \underbrace{\frac{\|d_{iC}\|_2^2}{(\|d_{ij}\|_2^2 + \|d_{iC}\|_2^2)} \cdot d_{ij}}_{v_{ij}} + \frac{100}{100+n} \cdot \underbrace{\frac{\|d_{ij}\|_2^2}{(\|d_{ij}\|_2^2 + \|d_{iC}\|_2^2)} \cdot d_{iC}}_{v_{ij}} + \frac{100}{100+n} \cdot noise \tag{3.14}$$

$$DSC = \frac{|\{p_{q,i} : \; \|p_{q,i} - C_q\| \leq \|p_{q,i} - C_k\| \;\; \forall \; k \in |\{M\}|\}|}{N} \tag{3.15}$$

$$CD = \sum_q \sum_i \|C_q - p_q, i\| \tag{3.16}$$

4. Approach

Your approach goes here ...

5. Implementation

Your implementation goes here...

5.1. Description of the Front End

5.2. Description of the Back End

```
Heuristic 1: Minimum Selection Shift - MSS
1  function heuristic_order_selection_shift(df_p, iterator, num_classes):
       Data: df_p is the dataframe in projection space, iterator is the current
             iteration step, num_classes is the amount of different classes in the
             dataset
       Result: dp shift vector, selected_class to be shifted, calculated DSC_value,
               calculated CD_value, calculated total_dist
       /* extract star coords and class of data to a new dataframe              */
2      df_star = df_p[['X','Y','class']]
       /* calculate class centroids and save them in a new dataframe            */
3      df_centroids = df_star.groupby('class', sort=True).mean().reset_index()
       /* calculate coordinates of the central centroid                         */
4      central_centroid = df_centroids[['X', 'Y']].mean()
       /* call a function to calculate all distances between centroids          */
5      df_distances = calc_centroid_distances(df_centroids)
       /* calculate the distance from each point to its associated
          centroid                                                              */
6      df_centroid_distances = calc_dist_p_to_assoc_centroid(df_centroids,
         df_star)
       /* calculate CD, DSC and total_dist for result                           */
7      CD_value = df_centroid_distances['distance'].sum()
8      DSC_value = calc_dsc(df_centroids, df_star)
9      total_dist = df_distances['distance'].sum()
       /* find the minimum distance between two class centroids                 */
10     min_dist_idx = df_distances['distance'].idxmin()
11     min_class1 = df_distances.loc[min_dist_idx, 'class1']
12     min_class2 = df_distances.loc[min_dist_idx, 'class2']
       /* select the class that is to be shifted                                */
13     selected_class, other_class = select_shifting_class(df_distances, min_class1,
         min_class2)
       /* calculate the new shifting vector dp                                  */
14     dp = calc_dp(df_centroids, selected_class, other_class, central_centroid,
         num_iter)
```

Figure 3.: Pseudocode for MSS

shown in Figure 4.

```
function calc_centroid_distances(df_centroids):
    Data: dataframe df_centroids with all positions of the centroids in projection space
    Result: dataframe df_distances with all distances between class centroids
    /* create dataframe for results */
    df_distances = pd.DataFrame(columns=['class1', 'class2', 'distance'], dtype=float)
    /* index for writing to df_distances */
    idx = 0
    /* calculate euclidean distance for every combination of centroids */
    for index in list(combinations(df_centroids.index, 2)) do
        p1 = [df_centroids.loc[index[0], 'X'], df_centroids.loc[index[0], 'Y']]
        p2 = [df_centroids.loc[index[1], 'X'], df_centroids.loc[index[1], 'Y']]
        df_distances.loc[idx, 'class1'] = index[0]
        df_distances.loc[idx, 'class2'] = index[1]
        df_distances.loc[idx, 'distance'] = math.dist(p1, p2)
        /* increase the index before going through the next iteration */
        idx = idx + 1
    end
```

Figure 4.: Function calc_centroid_distances

```
function
  $handle\_penalty\_counter(cnt, DSC\_o, DSC\_n, td\_o, td\_n, CD\_o, CD\_n)$:
    Data: counter $cnt$ for the current penalty $\tau$, old and new values for $DSC$, $TD$
          and $CD$ to compare towards termination criteria
    Result: new value $cnt$ for $\tau$
    if $DSC\_n \leq DSC\_o$ /* DSC decreased */
     then
        $cnt = cnt + 1$
    end
    if $td\_n \leq td\_o$ /* or td decreased */
     then
        $cnt = cnt + 1$
    end
    if $CD\_n \geq CD\_o$ /* or CD decreased */
     then
        $cnt = cnt + 1$
    end
    if $DSC\_n > DSC\_o$ /* DSC increased */
     then
        $cnt = 0$
    end
    /* DSC did not change and CD decreased or td increased */
    if $(DSC\_n == DSC\_o)$ & $((CD\_n < CD\_o) \mid (td\_n > td\_o))$ then
        $cnt = 0$
    end
    if $(CD\_n < CD\_o)$ & $(td\_n > td\_o)$/* CD decreased and td increased
        */
     then
        $cnt = 0$
    end
```

Figure 5.: Function handle_penalty_counter

6. Evaluation

Your Evaluation goes here ...

6.1. Datasets Used for Evaluation

seq_name	mcg	gvh	lip	chg	aac	alm1	alm2	class
other	number	number	number	number	number	number	number	classifier

Table 2.: Allocation of data types for *ecoli.header*

Before delving into each individual dataset, Table 3 provides a brief overview of their key data.

dataset	instances	attributes	numeric attributes	classes
iris	150	4	4	3
ecoli	336	8	7	7
wdbc	569	32	30	2
wine	179	13	13	3
yeast	1484	8	7	9
statlog	2000	36	36	6(7)

Table 3.: Key data on used datasets

6.2. Design of Evaluation

6.2.1. Metrics of Evaluation

Convergence and Reliability

6.2.2. Initial Configuration for Evaluation

dataset	DSC_{start}	CD_{start}	$d_{c,total,start}$
ecoli	63.88%	11636	3073
iris	89.93%	3585	340
statlog	21.06%	65743	112
wdbc	86.27%	22820	79
wine	72.32%	7703	201
yeast	27.44%	44547	2613

Table 4.: Key values of the initial projection for each dataset

6.2.3. Comparison

Max. Number of Iterations	Penalty Threshold	Orthographic
1000	100	Yes

Table 5.: Standard configuration for heuristic comparison

6.2.4. Behaviour with Different Parameters

Modification of Termination Criteria

Max. Number of Iterations	Penalty Threshold	Orthographic
1000	25 | 50	Yes

Table 6.: Configuration for heuristic comparison with reduced

Max. Number of Iterations	Penalty Threshold	Orthographic
1000	150 | 200	Yes

Table 7.: Configuration for heuristic comparison with increased

Max. Number of Iterations	Penalty Threshold	Orthographic
500 | 1000 | 2000	None	Yes

Table 8.: Configuration for heuristic comparison without PT and different iterations

6.3. Results of Evaluation

Qualitative Comparison

Quantitative Comparison

dataset	ΔDSC	ΔCD	CDC	iterations
ecoli	1.19%	99.87%	100.42%	12
iris	0%	102.49%	105.29%	75
statlog	7.7%	99.67%	198.21%	64
wdbc	8.45%	138.83%	83.54%	852
wine	20.34%	102.17%	175.62%	423
yeast	-0.06%	99.19%	100.84%	33

Table 9.: Impact on key values in comparison to initial projection for each dataset by RSS

dataset	ΔDSC	ΔCD	CDC	iterations
ecoli	0%	99.86%	100.16%	8
iris	0.67%	102.92%	107.06%	79
statlog	1.7%	99.93%	117.86%	13
wdbc	8.45%	139.15%	82.28%	685
wine	20.9%	102.04%	177.61%	456
yeast	0%	99.63%	100.11%	18

Table 10.: Impact on key values in comparison to initial projection for each dataset by OSS

dataset	ΔDSC	ΔCD	CDC	iterations
ecoli	3.88%	100.79%	102.05%	214
iris	3.36%	115.24%	130.59%	551
statlog	20.21%	104.97%	370.54%	396
wdbc	10.03%	121.54%	124.05%	858
wine	20.9%	104.25%	191.04%	499
yeast	0.88%	96.54%	105.82%	68

Table 11.: Impact on key values in comparison to initial projection for each dataset by MSS

dataset	ΔDSC	ΔCD	iterations
ecoli	-0.49%	99.21%	3
iris	0%	100%	3
statlog	1.45%	113.94%	20
wdbc	5.1%	129.91%	53
wine	27.68%	136.55%	255
yeast	-2.63%	101.41%	15

Table 12.: Impact on key values in comparison to initial projection for each dataset by PSS

6.3.1. Results for Behaviour with Different Parameters

parameter	ΔDSC	ΔCD	CDC	iterations
Standard	3.88%	100.79%	102.05%	214
PT=25	3.88%	101.01%	102.12%	210
PT=50	3.88%	100.99%	102.12%	216
PT=150	3.88%	100.88%	102.02%	235
PT=200	3.88%	100.88%	102.05%	254
500 Iterations	4.48%	100.32%	101.5%	500
1000 Iterations	5.08%	99.26%	99.97%	1000
2000 Iterations	5.08%	97.69%	96.29%	2000
PT Calculation	4.18%	100.71%	102.18%	163
Non-Orthographic	2.99%	71.55%	140.81%	48

Table 13.: Impact on key values by parameter change for ecoli dataset by MSS

Analysis of Results for Different Parameters

6.3.2. Results for Convergence and Reliability

6.3.3. Comparison with Other Results

7. Discussion

Your discussion goes here ...

8. Future work

Your future work goes here ...

Bibliography

[1] Ankerst, Mihael ; Ester, Martin ; Kriegel, Hans-Peter: Towards an effective cooperation of the user and the computer for classification. In: Ramakrishnan, Raghu (Hrsg.) ; Stolfo, Sal (Hrsg.) ; Bayardo, Roberto (Hrsg.) ; Parsa, Ismail (Hrsg.): *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '00*. New York, New York, USA : ACM Press, 2000, S. 179–188. – ISBN 1581132336

[2] Belkin, Mikhail ; Niyogi, Partha: Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. In: *Neural Computation* 15 (2003), Nr. 6, S. 1373–1396. – ISSN 0899-7667

[3] Blum, Christian ; Roli, Andrea: Hybrid Metaheuristics: An Introduction. In: Kacprzyk, Janusz (Hrsg.) ; Blum, Christian (Hrsg.) ; Aguilera, Maria José B. (Hrsg.) ; Roli, Andrea (Hrsg.) ; Sampels, Michael (Hrsg.): *Hybrid Metaheuristics* Bd. 114. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008, S. 1–30. – ISBN 978-3-540-78294-0

[4] Chu, Jun-Uk ; Moon, Inhyuk ; Mun, Mu-Seong: A real-time EMG pattern recognition system based on linear-nonlinear feature projection for a multifunction myoelectric hand. In: *IEEE transactions on bio-medical engineering* 53 (2006), Nr. 11, S. 2232–2239

[5] Daniels, Karen ; Grinstein, Georges ; Russell, Adam ; Glidden, Mason: Properties of normalized radial visualizations. In: *Information visualization* 11 (2012), Nr. 4, S. 273–300. – ISSN 1473-8716

[6] Demartines, P. ; Herault, J.: Curvilinear component analysis: a self-organizing neural network for nonlinear mapping of data sets. In: *IEEE transactions on neural networks* 8 (1997), Nr. 1, S. 148–154. – ISSN 1045-9227

[7] Dua, Dheeru ; Graff, Casey: *UCI Machine Learning Repository*. 2017. – URL `http://archive.ics.uci.edu/ml`

[8] Heidari, Morteza ; Lakshmivarahan, Sivaramakrishnan ; Mirniaharikandehei, Seyedehnafiseh ; Danala, Gopichandh ; Maryada, Sai Kiran R. ; Liu, Hong ; Zheng, Bin: Applying a Random Projection Algorithm to Optimize Machine Learning Model for Breast Lesion Classification. In: *IEEE transactions on bio-medical engineering* 68 (2021), Nr. 9, S. 2764–2775

[9] Hertwig, Ralph ; Pachur, Thorsten: Heuristics, History of. In: *International Encyclopedia of the Social & Behavioral Sciences*. Elsevier, 2015, S. 829–835. – ISBN 9780080970875

[10] Hinton, Geoffrey E. ; Roweis, Sam: Stochastic Neighbor Embedding. In: S. Becker (Hrsg.) ; S. Thrun (Hrsg.) ; K. Obermayer (Hrsg.): *Advances in Neural Information Processing Systems* Bd. 15, MIT Press, 2002. – URL `https://proceedings.neurips.cc/paper/2002/file/6150ccc6069bea6b5716254057a194ef-Paper.pdf`

[11] Hoffman, P. ; Grinstein, G. ; Marx, K. ; Grosse, I. ; Stanley, E.: DNA visual and analytic data mining. In: *Proceedings. Visualization '97 (Cat. No. 97CB36155)*, IEEE, 19-24 Oct. 1997, S. 437–441. – ISBN 0-8186-8262-0

[12] Inselberg, Alfred: The plane with parallel coordinates. In: *The Visual Computer* 1 (1985), Nr. 2, S. 69–91. – ISSN 0178-2789

[13] Inselberg, Alfred: *Parallel Coordinates*. New York, NY : Springer New York, 2009. – ISBN 978-0-387-21507-5

[14] Kandogan, Eser: Star Coordinates: A Multi-dimensional Visualization Technique with Uniform Treatment of Dimensions. In: *In Proceedings of the IEEE Information Visualization Symposium, Late Breaking Hot Topics*, 2000, S. 9–12

[15] Keim, D. A. ; Mansmann, F. ; Schneidewind, J. ; Ziegler, H.: Challenges in Visual Data Analysis. In: *Tenth International Conference on Information Visualisation (IV'06)*, IEEE, 05-07 July 2006, S. 9–16. – ISBN 0-7695-2602-0

[16] Keim, Daniel (Hrsg.): *Mastering the information age: Solving problems with visual analytics*. Goslar : Eurographics Association, 2010. – ISBN 978-3-905673-77-7

[17] Lehmann, D. J. ; Theisel, H.: Orthographic Star Coordinates. In: *IEEETransactionsonVisualizationandComputerGraphics(ProceedingsIEEEInformationVisualization)* (2013)

[18] McInnes, Leland ; Healy, John ; Melville, James: *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. – URL `http://arxiv.org/pdf/1802.03426v3`

[19] P.M., Todd: Heuristics for Decision and Choice. In: Neil, J. S. (Hrsg.) ; Paul, B. B. (Hrsg.): *International Encyclopedia of the Social & Behavioral Sciences*. Oxford :

Pergamon, 2001, S. 6676–6679. – URL https://www.sciencedirect.com/science/article/pii/B008043076700629X. – ISBN 978-0-08-043076-8

[20] Rauber, Paulo E. ; Falcão, Alexandre X. ; Telea, Alexandru C.: Projections as visual aids for classification system design. In: *Information visualization* 17 (2018), Nr. 4, S. 282–305. – ISSN 1473-8716

[21] Rubio-Sánchez, Manuel ; Raya, Laura ; Díaz, Francisco ; Sanchez, Alberto: A comparative study between RadViz and Star Coordinates. In: *IEEE transactions on visualization and computer graphics* 22 (2016), Nr. 1, S. 619–628

[22] Sammon, J. W.: A nonlinear mapping for data structure analysis. In: *IEEE Transactions on Computers* 18 (1969), Nr. 5, S. 401–409

[23] Tenenbaum, J. B. ; Silva, V. de ; Langford, J. C.: A global geometric framework for nonlinear dimensionality reduction. In: *Science (New York, N.Y.)* 290 (2000), Nr. 5500, S. 2319–2323. – ISSN 0036-8075

[24] van der Maaten, Laurens ; Hinton, Geoffrey: Viualizing data using t-SNE. In: *Journal of Machine Learning Research* 9 (2008), S. 2579–2605

Appendix

Appendix A:

Pseudocode for Heuristics

Heuristic 2: Random Selection Shift RSS

```
function heuristic_random_selection_shift(df_p, iterator, num_classes):
    Data: df_p is the dataframe in projection space, iterator is the current
          iteration step, num_classes is the amount of different classes in the
          dataset
    Result: dp shift vector, selected_class to be shifted, calculated DSC_value,
            calculated CD_value, calculated total_dist
    /* extract star coords and class of data to a new dataframe     */
    df_star = df_p[['X','Y','class']]
    /* calculate class centroids and save them in a new dataframe   */
    df_centroids = df_star.groupby('class', sort=True).mean().reset_index()
    /* calculate coordinates of the central centroid                */
    central_centroid = df_centroids[['X', 'Y']].mean()
    /* call a function to calculate all distances between centroids */
    df_distances = calc_centroid_distances(df_centroids)
    /* calculate the distance from each point to its associated
       centroid                                                     */
    df_centroid_distances = calc_dist_p_to_assoc_centroid(df_centroids,
      df_star)
    /* calculate CD, DSC and total_dist for result                  */
    CD_value = df_centroid_distances['distance'].sum()
    DSC_value = calc_dsc(df_centroids, df_star)
    total_dist = df_distances['distance'].sum()
    /* randomly choose a class to be shifted                        */
    selected_class = np.random.randint(num_classes)
    /* select all distances for selected class                      */
    selected_class_distances = df_distances.(df_distances[selected_class])
    /* other_class is the nearest class to selected_class           */
    min_selected_class_distance = selected_class_distances['distance'].idxmin()
    /* calculate the new shifting vector dp                         */
    dp = calc_dp(df_centroids, selected_class, other_class, central_centroid,
      num_iter)
```

Figure 6:: Pseudocode for RSS

Heuristic 3: Order Selection Shift - OSS

```
1  function heuristic_order_selection_shift(df_p, iterator, num_classes):
       Data: df_p is the dataframe in projection space, iterator is the current
             iteration step, num_classes is the amount of different classes in the
             dataset
       Result: dp shift vector, selected_class to be shifted, calculated DSC_value,
               calculated CD_value, calculated total_dist
       /* extract star coords and class of data to a new dataframe     */
2      df_star = df_p[['X','Y','class']]
       /* calculate class centroids and save them in a new dataframe   */
3      df_centroids = df_star.groupby('class', sort=True).mean().reset_index()
       /* calculate coordinates of the central centroid                */
4      central_centroid = df_centroids[['X', 'Y']].mean()
       /* call a function to calculate all distances between centroids */
5      df_distances = calc_centroid_distances(df_centroids)
       /* calculate the distance from each point to its associated
          centroid                                                     */
6      df_centroid_distances = calc_dist_p_to_assoc_centroid(df_centroids,
         df_star)
       /* calculate CD, DSC and total_dist for result                  */
7      CD_value = df_centroid_distances['distance'].sum()
8      DSC_value = calc_dsc(df_centroids, df_star)
9      total_dist = df_distances['distance'].sum()
       /* choose a class to be shifted by order                        */
10     selected_class = num_iter % num_classes)
       /* select all distances for selected class                      */
11     selected_class_distances = df_distances.(df_distances[selected_class])
       /* other_class is the nearest class to selected_class           */
12     min_selected_class_distance = selected_class_distances['distance'].idxmin()
       /* calculate the new shifting vector dp                         */
13     dp = calc_dp(df_centroids, selected_class, other_class, central_centroid,
         num_iter)
```

Figure 7:: Pseudocode for OSS

Heuristic 4: Point Selection Shift - PSS

```
1  function heuristic_order_selection_shift(df_p, iterator, num_classes):
       Data: df_p is the dataframe in projection space, iterator is the current
             iteration step, num_classes is the amount of different classes in the
             dataset
       Result: dp shift vector, selected_class to be shifted, calculated DSC_value,
               calculated CD_value, calculated total_dist
       /* extract star coords and class of data to a new dataframe     */
2      df_star = df_p[['X','Y','class']]
       /* calculate class centroids and save them in a new dataframe   */
3      df_centroids = df_star.groupby('class', sort=True).mean().reset_index()
       /* calculate coordinates of the central centroid                */
4      central_centroid = df_centroids[['X', 'Y']].mean()
       /* call a function to calculate all distances between centroids */
5      df_distances = calc_centroid_distances(df_centroids)
       /* calculate the distance from each point to its associated
          centroid                                                     */
6      df_centroid_distances = calc_dist_p_to_assoc_centroid(df_centroids,
         df_star)
       /* calculate CD, DSC and total_dist for result                  */
7      CD_value = df_centroid_distances['distance'].sum()
8      DSC_value = calc_dsc(df_centroids, df_star)
9      total_dist = df_distances['distance'].sum()
       /* find the maxmimum distance                                   */
10     max_dist_idx = df_centroid_distances['distance'].idxmax()
       /* select the point that is to be shifted                       */
11     centroid_id = df_centroid_distances.loc[max_dist_idx, 'class']
12     centroid_coords = [df_centroids.loc[centroid_id, 'X'],
         df_centroids.loc[centroid_id, 'Y']]
13     point_coord = [df_centroid_distances.loc[max_dist_idx, 'X'],
         df_centroid_distances.loc[max_dist_idx, 'Y']]
       /* create noise                                                 */
14     noise = np.random.normal(0, 1, 1) * 100 / (1000 + num_iter)
       /* calculate the new shifting vector dp                         */
15     dp = [centroid_coords[0] - point_coord[0] + noise, centroid_coords[1] -
         point_coord[1] + noise]
```

Figure 8:: Pseudocode for PSS

Appendix B:

Key values for all heuristics for each dataset

dataset	DSC_{start}	DSC_{end}	CD_{start}	CD_{end}	$d_{c,total,start}$	$d_{c,total,end}$
ecoli	63.88%	65.07%	11636	11651	3073	3086
iris	89.93%	89.93%	3585	3498	340	358
statlog	21.06%	28.76%	65743	65963	112	222
wdbc	86.27%	94.72%	22820	16437	79	66
wine	72.32%	92.66%	7703	7539	201	353
yeast	27.44%	27.38%	44547	44911	2613	2635

Table 14:: Key values for RSS for each dataset

dataset	DSC_{start}	DSC_{end}	CD_{start}	CD_{end}	$d_{c,total,start}$	$d_{c,total,end}$
ecoli	63.88%	63.88%	11636	11652	3073	3078
iris	89.93%	90.6%	3585	3483	340	364
statlog	21.06%	22.76%	65743	65784	112	132
wdbc	86.27%	94.72%	22820	16399	79	65
wine	72.32%	93.22%	7703	7549	201	357
yeast	27.44%	27.44%	44547	44713	2613	2616

Table 15:: Key values for OSS for each dataset

dataset	DSC_{start}	DSC_{end}	CD_{start}	CD_{end}	$d_{c,total,start}$	$d_{c,total,end}$
ecoli	63.88%	67.76%	11636	11545	3073	3136
iris	89.93%	93.29%	3585	3111	340	444
statlog	21.06%	41.27%	65743	62631	112	415
wdbc	86.27%	96.3%	22820	18776	79	98
wine	72.32%	93.22%	7703	7389	201	384
yeast	27.44%	28.32%	44547	46156	2613	2765

Table 16:: Key values for MSS for each dataset

dataset	DSC_{start}	DSC_{end}	CD_{start}	CD_{end}
ecoli	63.88%	63.39%	11636	11729
iris	89.93%	89.93%	3585	3585
statlog	21.06%	22.51%	65743	57698
wdbc	86.27%	91.37%	22820	17566
wine	72.32%	100%	7703	5641
yeast	27.44%	24.81%	44547	43926

Table 17:: Key values for PSS for each dataset