International Workshop on Machine Learning in Visualisation for Big Data (2020)
D. Archambault, I. Nabney and J. Peltonen (Editors)

Progressive Multidimensional Projections: A Process Model based on Vector Quantization

E. Ventocilla[1], R. M. Martins[2], F. Paulovich[3] and M. Riveiro[4]

[1]University of Skövde, [2]Linnaeus University, [3]Dalhousie University and [4]Jönköping University

Abstract

As large datasets become more common, so becomes the necessity for exploratory approaches that allow iterative, trial-and-error analysis. Without such solutions, hypothesis testing and exploratory data analysis may become cumbersome due to long waiting times for feedback from computationally-intensive algorithms. This work presents a process model for progressive multidimensional projections (P-MDPs) that enables early feedback and user involvement in the process, complementing previous work by providing a lower level of abstraction and describing the specific elements that can be used to provide early system feedback, and those which can be enabled for user interaction. Additionally, we outline a set of design constraints that must be taken into account to ensure the usability of a solution regarding feedback time, visual cluttering, and the interactivity of the view. To address these constraints, we propose the use of incremental vector quantization (iVQ) as a core step within the process. To illustrate the feasibility of the model, and the usefulness of the proposed iVQ-based solution, we present a prototype that demonstrates how the different usability constraints can be accounted for, regardless of the size of a dataset.

CCS Concepts

• ***Human-centered computing*** $\rightarrow$ ***Visual analytics;***

1. Introduction

A common approach in exploratory data analysis is to construct an overview of a dataset's structure, i.e., a visualization of neighborhood relationships among data points, to identify emerging cluster patterns [Sam69, VR19]. When dealing with high-dimensional data, multidimensional projections (MDPs) can be used to construct 2 or 3-dimensional visualizations of data structure. Examples of these are dimensionality reduction (DR) techniques such as t-SNE [vdMH08] and PCA [Hot33] (visually encoded using scatterplot), and clustering techniques such as SOM [Koh90] and Ward hierarchical clustering (visually encoded with U-Matrices [Ult90] and dendrograms respectively).

Different challenges arise in this exploratory process as data grows in size (number of data points) and dimensionality (number of attributes) [EMK*19]. High-dimensional data, on the one hand, may lead to sparsely populated spaces, which in turn leads to less meaningful values when it comes to computing neighborhood similarities (the so-called *curse of dimensionality*) [AHK01]—hence affecting the representativeness of the outcomes of MDPs. Extensive work has been done to address this challenge, e.g., [PM08, vdMH08, JCC*11, MHM18].

The size of data, on the other hand, even though it appears to have a low correlation to the quality of a projection [EMK*19], can entail challenges for visual analytics systems that aim to comply with different usability constraints, such as, *feedback times* (i.e., waiting times before displaying results from an algorithm) [MPG*14, SPG14], *visual cluttering* (i.e., overlapping elements in the view) [PWR04], and *view interactiveness* (i.e., direct interactive capabilities of the visualization such as brushing and linking). The larger the dataset, the more time it will take to create a projection, the more likely for there to be visual cluttering, and the more likely for the view to have limited interactive capabilities with fast visual feedback.

This work addresses mainly these three usability constraints. Phrased as a question, how can a visual analytics system enable progressive MDPs (P-MDPs) while complying with the outlined usability constraints, regardless of the size of a dataset? Previous work, to the best of our knowledge, has only addressed at most two of the outlined usability constraints, and has only done so tailored to a specific MDP. We propose a process model as an answer, based on incremental vector quantization (iVQ) techniques, i.e., techniques which model the data space through a set of vector representatives, and whose modeling power is improved over time using partial sets of the whole data. Through a prototype implementation of the proposed model, we show that it is possible to comply with the above mentioned usability constraints, even when displaying more than one perspective a dataset's structure (i.e., more than one MDP), despite the size of a dataset. Concretely, our contributions are:

DOI: 10.2312/mlvis.20201099

1. An extended list of design requirements for P-MDP that addresses the outlined usability constraints.
2. A process model that enables P-MDPs for large datasets through iVQ, and outlines the relation of its elements to (a) the different types of user involvements, and (b) the design requirements.
3. A progressive analytics prototype, scalable to large datasets (even if distributed in clusters), that illustrates the flexibility of the model, and its validity in terms of the design requirements.

2. Related work

Different solutions have been proposed to avoid long waiting times before providing visual feedback of an MDP to a user. Some have focused on increasing the performance of existing algorithms [VDM14, CRHC19, PTM*20], while others have enabled progressive training in order to provide fast feedback based on partial results [PM08, MPG*14, SPG14, SASS16, PLvdM*17, KCL*17]. In most cases, if not all, the work has focused on dimensionality reduction (DR) techniques. See Sacha et al. [SZS*17], Nonato et al., [NA18] and Espadoto et al., [EMK*19] for thorough surveys on these techniques.

Progressive training and analysis have gained particular interest in the research community, and different authors have sought to formalize its constructs and implications. Stolper et al., [SPG14] laid out a set of design goals, such as, showing partial results, allowing users to steer the algorithm to areas of interest, avoiding excessive updates that may lead to distractions, among others. Mühlbacher et al., [MPG*14], on the other hand, proposed different types of user involvement (TUI), during progressive training. They characterized TUIs in terms of direction (system feedback and user control), and entities of interest (execution and result), and outlined the different TUIs at the intersection of each, i.e.,: execution feedback, result feedback, execution control, and result control. Schulz et al., [SASS16] proposed a set of *building blocks*—as extensions to the Data State Reference Model [Chi00]—with which incremental visualizations can be developed. The building blocks account for different types of elements such as views, data flows, operators, and mechanisms for sequencing and buffering. Fekete et al., [FP16] defined a Progressive Analytics paradigm in terms of a subsequent set of calls to a learning function F_p, which takes a "growing subset of D", an evolving state S, and a time constraint q. Additionally, they gathered and extended the features that a progressive analytics library should have. We build on these as described in Section 3.

To address visual cluttering, some solutions have relied on different visual cues and metaphors, such as opacity and contours [PLvdM*17], edge bundling [LSL*17, ZYQ*08], and surfaces [PEPM12]. Others have done so through the use of techniques that *compress* (i.e., vector quantize) data into a set of representative units, e.g., SOM [RFZ08], GNG [VR19], CCA [DH97], HiPP [PM08], and HSNE [HVP*19]. Through these, only a limited amount of elements needs to be visually encoded.

View interactiveness has not received as much attention as the previous two challenges. We regard it as the capabilities that a system provides on the *visible* MDP, that is, interactive features over a plot such as linking, brushing, dragging, etc. These do not necessarily entail steering the computation, but provide means to explore the partial or final results. As data grows, such interactive features become more challenging to maintain, especially if the system is to account for continuity-preserving latency, where visual feedback is given under 0.1 seconds [FP16].

3. Design Requirements

The rationale of the proposed model was mainly driven by Fekete et al.'s [FP16] design requirements. According to them, progressive analytics libraries should: (**F1**) provide increasingly meaningful partial results; (**F2**) provide feedback about the state of the computation (e.g., aliveness, absolute and relative progress); (**F3**) provide control over the process (e.g., canceling, resuming, prioritizing); (**F4**) guarantee feedback time constraints (mainly to attention preserving latency, i.e., $< 10s$); (**F5**) allow manipulation of progressive values; (**F6**) allow steering through parameters or other computation components; and (**F7**) allow performing exploratory, analytical computations. For P-MDPs, we add three more requirements which were drawn from the solutions given by [PM08] and [HVP*19]. Concretely, a P-MDP should also: (**F8**) provide an overview of the data structure, while avoiding visual clutter; (**F9**) maintain view interactiveness at a continuity preserving latency ($< .1s$); and (**F10**) allow users to navigate across different levels of detail. To address these requirements, our proposed P-MDP process model leverages from incremental VQ.

4. Vector Quantization

Originally, VQ techniques are intended for reducing the size of digital information that is to be sent through some communication channel, in a manner that can be later reconstructed [Gra84]. Compression is done utilizing a set of representative units or prototypes, i.e., vectors that have the same dimensionality as the original dataset and *stand* in dense regions of the data space [Say17]. These prototypes are then used as a codebook from which data can be reconstructed.

Not all VQ techniques, however, can be used for constructing P-MDPs; VQ techniques for P-MDP should: (1) work in an incremental manner; and (2), retain, to the largest possible extent, the distance relations of the original data, while minimizing the number of required prototypes to do so. These criteria seek to address requirements F8 and F9, which translates to a balance between compression and representation, i.e., too few prototypes will have little representative power for visual assessment, and too many will hinder the usability of the system. Examples that fulfill these requirements are neural-based techniques, such as the Self-organizing Maps (SOM) [Koh90] and the Growing Neural Gas (GNG) [Fri95]; partition-based techniques such as Mini batch K-Means [Scu10]; hierarchy-based techniques such as BIRCH [ZRL96]; and density-based techniques such as D-Stream [CT07].

5. The Process Model

Figure 1 shows the core process model. It is divided into four horizontal groups of elements (delimited by dashed lines): Parameters, Functions, Objects, and States. Objects represent data structures, such as tables, graphs, or vectors; Functions are processes that perform some sort of transformation on an object; Parameters determine the behavior of functions; and States keep a record of the current state of the system. The shaded areas represent the domains of the iVQ and the MDP transformations.

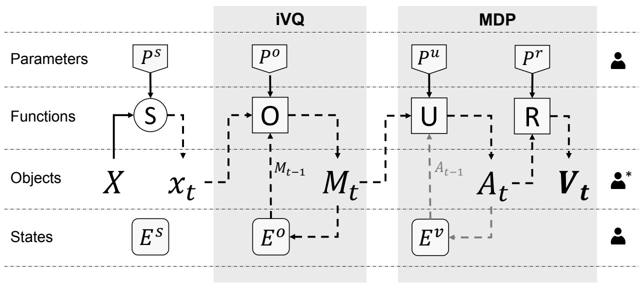

Figure 1: *The VQ-MDP Process Model. User icons (on the right) show which types of elements (Parameters, Functions, Objects, States) can be enabled for user iteraction.*

The process starts by sampling a dataset X, using a sampling function S (i.e., a sequencer [SASS16]), with parameters P^s (e.g., sample size, replacement, or frequency). The sampling function can be seen as a faucet that pushes data into the process pipe. It is through its parameters that a user can filter data points and adjust the speed of the overall training process. The state E^s can hold information about, e.g., the number of samples drawn so far.

A sample x_t, produced by S at a time t, is taken into the iVQ domain which is in charge of modeling the original (non-projected) data space X, by reducing it to a set of representative prototypes. Concretely, the sample is taken as an input by an Optimization function O, which uses it to *fit* (i.e., improve the representative power of) a model M_{t-1} and produce an improved model M_t (when $t = 0$, an initialization strategy for M_{t-1} is used). The behavior of O, in this case, is determined by parameters P^o (e.g., number of units, learning step, or cooling factor). The process within VQ is represented by an internal loop through which progressive optimization is achieved: the improved model M_t becomes the base model for the next optimization. M is, therefore, one of the objects that define the learning state E^o, along with other information such as the number of iterations and goodness of the model.

The optimized model M_t enters the MDP domain, where it is (a) transformed into a visual abstraction A_t, through a function U; and (b) made into a visual element V_t, through a function R. These two functions are, correspondingly, analogous to Chi's [Chi00] Visualization Transformation and Visual Mapping Transformation operators. U can be any MDP technique producing an abstraction object A_t, which can be visualized, e.g., PCA, t-SNE, force-directed placement, Ward HC, or OPTICS [ABKS99]. The outcome of these can be visualized using some function R that produces its corresponding visual element V_t, e.g., scatterplots or graphs in the case of DR techniques, dendrograms in the case of hierarchical clustering, or reachability plots for OPTICS. Another loop (depicted in light gray) shows the possibility to use the previous visual abstraction A_{t-1}, as a baseline object for the next visual transformation. Doing so avoids drastic changes in V_t when using non-deterministic MDPs such as t-SNE. The behaviors of U and R, as with previous functions, are defined by parameters P^u, (e.g., number of iterations per sample, distance metric, perplexity, etc), and P^r (e.g., plot size, color map, etc). Through P^u, users can test different parameter settings, and see their influence on the projection with fast visual feedback. Through P^u users can test different parameter and see their influence on the projection with fast visual feedback.

The user icons on the right of the process model (Figure 1) indicate the groups of elements with which a user can interact, either through control or visual feedback. The asterisk at the objects level indicates that user interaction is mainly towards V_t. Direct interaction with other objects is, if not impossible, less intuitive. We illustrate user involvement through a prototype, and by referring to Mühlbacher et al.'s [MPG*14] TUIs.

6. Prototype

To illustrate one possible implementation of our proposed model, we developed a prototype (see Figure 2) using technologies such as Plotly Dash for the visualization, and PySpark, Scikit-Learn and NumPy as data processing libraries. We used a dataset of methylation array profiles [emt19], which has 787939 data points and 51 features.

Process elements. The prototype uses Spark DataFrame's random sampler S (the `sample` method); Mini Batch K-Means as iVQ, with the `partial_fit` method as optimizer (O); MDS and OPTICS as visual transformations (U); and scatterplot, reachability plot, and parallel coordinates as visual mappers (R)—the latter only requires a transposed x_t as an input, which is why we disregard its U function. The sampler takes a dataset as a Spark DataFrame (X), and three parameters (P^s): number of data points, sample frequency, replacement, and seed. The latter two were left with their default values and not shown to the user.

As an output, the sampler produces samples (x_t) as vector sets. Samples are passed to Mini Batch K-Means (Scikit-learn's implementation) for modeling, which takes several parameters (P^o), e.g., number of clusters, initialization method, maximum number of iterations, etc. Our prototype allows users to set the number of clusters (i.e., prototypes), and the reassignment ratio. By default, the number of prototypes is set to 200. When `partial_fit` is called, it implicitly uses the latest fitted prototypes (M_{t-1}), and produces a new prototype set (M_t). For $t = 0$, Mini Batch K-Means uses 'k-means++' as initialization strategy.

Depending on the wanted visual encoding, fitted prototypes (M_t) need to be preprocessed by a visual transformation U. For the scatterplot view we used MDS, which projects the data to a two-dimensional plane, and for the reachability plot we used OPTICS, which calculates and sorts point distances to core points. Each takes its own set of parameters (P^u): For MDS, the number of initializations and the number of iterations; and for OPTICS, the distance metric (Euclidean, Manhattan, or Cosine), the minimum number of samples, and the Chi value. Parallel coordinates, as previously mentioned, only requires transposing the data matrix.

User involvement. Execution control is enabled through the 'Reset' and 'Run' buttons. The latter executes the entire chain of

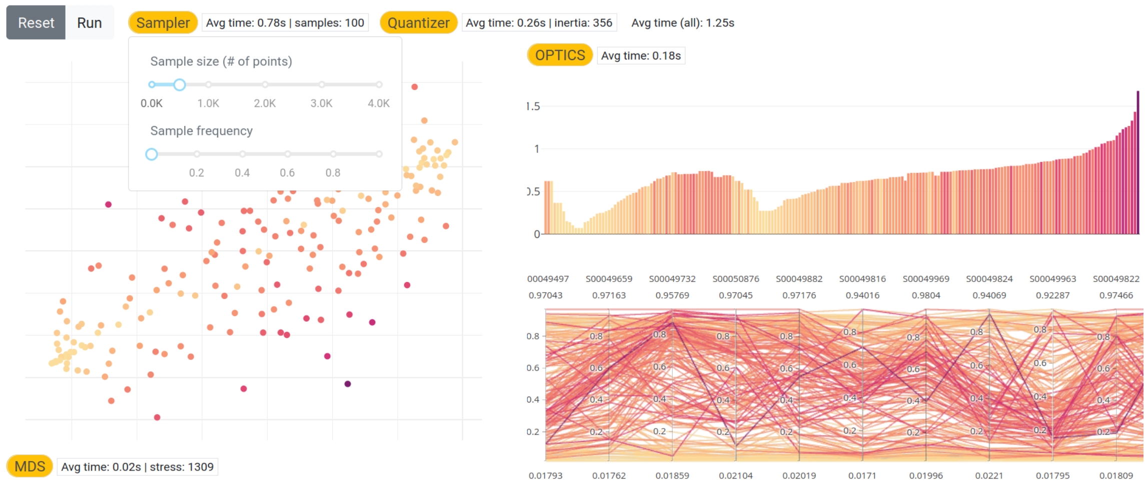

Figure 2: *Prototype of the process model. In the upper part are the control elements: the 'Reset' and 'Run' buttons enable execution control. Sampler, Quantizer, MDS and OPTICS are toggle buttons (in yellow) that provide access to their corresponding parameters. Next to each are their respective state. Below the controls are the visual encodings: an MDS projection in the scatterplot, OPTICS in the bar chart (i.e., reachability plot), and feature values in the parallel coordinates. Each provides a different perspective of the prototypes, i.e., groups of genes with similar behavior in different patients. Color represents the variance of each prototype.*

transformations of the process model until the user decides to stop (using the same button). The former *cleans* the model so that a new set of prototypes is used. Result control can be done through the parameters in iVQ and MDP, which can be updated at any point in time through the dropdown boxes for each component (as an example, Figure 2 shows the Sampler parameters). Users can also *zoom in* into regions of the data space—as compliance with the F10 requirement—using the scatterplot. This translates to three processing steps: (1) predicting to which prototype each data point belongs to (as in clustering), (2) instantiating a new DataFrame X_l (where l represents the zooming level) that filters out those data points that do not belong to the zoomed prototypes, and (3) instantiating a new set of prototypes M_l that will model the zoomed data space. The original prototypes M_{l-1}, and data space X_{l-1}, are kept for zooming out. After zooming in, the transformations of the process model remain the same. The zooming process can be repeated to look deeper into different levels.

Execution feedback is given through continuous updates of state values (E) for each component. These are shown next to each component's toggle button, e.g., 'Avg time', 'samples', 'inertia'. Result feedback, on the other hand, is provided by continuous updates of the visual elements. Finally, brushing and linking are enabled as view interactions—i.e., brushing over one view highlights the corresponding elements on the others.

7. Discussion

Visual feedback, within the attention preserving latency threshold ($< 10s$), is possible through a balance between sample sizes, and the number of VQ and MDP iterations per sample. Tests with our prototype show that the attention latency can be (comfortably) sustained, even when applying the three different MDPs for every M_t. For the case of samples of 500 data points, with 10 VQ iterations, and a 100 MDS iterations per sample (OPTICS does not take an iterations parameter), the average execution time for a 100 samples for all computations was of 1.25 seconds. The most expensive operation was sampling, with an average execution time of .78 seconds.

Visual clutter, on the other hand, has been largely avoided by plotting 200 representative data points, instead of the 787K from the original dataset; while achieving *sensible* and *consistent* visual patterns. Sensible given the nature of the dataset, where a large number of data points have low variance with high and low values (i.e., the yellow clusters), while others lay in between with higher variance (i.e., purple data points). Consistency, on the other hand, comes from our visual assessment of the similarity between the cluster patterns that resulted from different runs.

Finally, view interactiveness was *partially* achieved. Concretely, hovering and brushing did achieve visual feedback at a continuity preserving latency ($< 0.1s$), but took longer time (close to $1s$) to provide visual feedback on linking. This was due, however, to how Dash handles interactions that trigger updates between views.

8. Conclusion

This work presented a process model based on iVQ, through which P-MDPs can be developed, while accounting for different usability constraints. Through the given level of abstraction, the model makes it easy to discern which are the elements with which different types of user involvement can be enabled. To illustrate the model, a prototype was developed and described. The prototype shows that it is possible to provide different perspectives of the structure of a dataset, with incremental and timely results, with low visual cluttering, and with fast view interactiveness, despite the size of the dataset. Future work will address the requirements that arise from streaming data.

References

[ABKS99] ANKERST M., BREUNIG M. M., KRIEGEL H.-P., SANDER J.: OPTICS: Ordering Points to Identify the Clustering Structure. *ACM Sigmod record 28*, 2 (1999), 49–60. 3

[AHK01] AGGARWAL C. C., HINNEBURG A., KEIM D. A.: On the Surprising Behavior of Distance Metrics in High Dimensional Space. In *International conference on database theory*. Springer, 2001, pp. 420–434. 1

[Chi00] CHI E. H.: A taxonomy of visualization techniques using the data state reference model. In *Information Visualization, 2000. InfoVis 2000. IEEE Symposium On* (2000), IEEE, pp. 69–75. 2, 3

[CRHC19] CHAN D. M., RAO R., HUANG F., CANNY J. F.: GPU accelerated t-distributed stochastic neighbor embedding. *Journal of Parallel and Distributed Computing 131* (2019), 1–13. 2

[CT07] CHEN Y., TU L.: Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2007), KDD '07, Association for Computing Machinery, pp. 133–142. 2

[DH97] DEMARTINES P., HERAULT J.: Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *IEEE Transactions on Neural Networks 8*, 1 (1997), 148–154. 2

[EMK*19] ESPADOTO M., MARTINS R. M., KERREN A., HIRATA N. S. T., TELEA A. C.: Towards a Quantitative Survey of Dimension Reduction Techniques. *IEEE Transactions on Visualization and Computer Graphics* (2019), 1–1. 1, 2

[emt19] Methylation array profiling of undifferentiated sarcomas of adults, apr 2019. URL: https://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-6961/?array=A-GEOD-21145. 3

[FP16] FEKETE J.-D., PRIMET R.: Progressive Analytics: A Computation Paradigm for Exploratory Data Analysis. *CoRR abs/1607.05162* (2016). 2

[Fri95] FRITZKE B.: A Growing Neural Gas Network Learns Topologies. In *Advances in Neural Information Processing Systems 7*, Tesauro G., Touretzky D. S., Leen T. K., (Eds.). MIT Press, 1995, pp. 625–632. 2

[Gra84] GRAY R.: Vector quantization. *IEEE ASSP Magazine 1*, 2 (1984), 4–29. 2

[Hot33] HOTELLING H.: Analysis of a complex of statistical variables into principal components. *Journal of educational psychology 24*, 6 (1933), 417. 1

[HVP*19] HÖLLT T., VILANOVA A., PEZZOTTI N., LELIEVELDT B. P. F., HAUSER H.: Focus+Context Exploration of Hierarchical Embeddings. *Computer Graphics Forum 38*, 3 (2019), 569–579. 2

[JCC*11] JOIA P., COIMBRA D., CUMINATO J. A., PAULOVICH F. V., NONATO L. G.: Local Affine Multidimensional Projection. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2563–2571. 1

[KCL*17] KIM H., CHOO J., LEE C., LEE H., REDDY C. K., PARK H.: PIVE: Per-Iteration Visualization Environment for Real-Time Interactions with Dimension Reduction and Clustering. In *Thirty-First AAAI Conference on Artificial Intelligence* (2017). 2

[Koh90] KOHONEN T.: The self-organizing map. *Proceedings of the IEEE 78*, 9 (1990), 1464–1480. 1, 2

[LSL*17] LIU M., SHI J., LI Z., LI C., ZHU J., LIU S.: Towards Better Analysis of Deep Convolutional Neural Networks. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 91–100. 2

[MHM18] MCINNES L., HEALY J., MELVILLE J.: UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv:1802.03426 [cs, stat]* (2018). arXiv:1802.03426. 1

[MPG*14] MÜHLBACHER T., PIRINGER H., GRATZL S., SEDLMAIR M., STREIT M.: Opening the Black Box: Strategies for Increased User Involvement in Existing Algorithm Implementations. *IEEE Transactions on Visualization and Computer Graphics 20* (2014), 1643–1652. 1, 2, 3

[NA18] NONATO L. G., AUPETIT M.: Multidimensional Projection for Visual Analytics: Linking Techniques with Distortions, Tasks, and Layout Enrichment. *IEEE Transactions on Visualization and Computer Graphics* (2018), 1–1. 2

[PEPM12] POCO J., ELER D. M., PAULOVICH F. V., MINGHIM R.: Employing 2D Projections for Fast Visual Exploration of Large Fiber Tracking Data. *Computer Graphics Forum 31* (2012), 1075–1084. 2

[PLvdM*17] PEZZOTTI N., LELIEVELDT B. P. F., V D MAATEN L., HÖLLT T., EISEMANN E., VILANOVA A.: Approximated and User Steerable tSNE for Progressive Visual Analytics. *IEEE Transactions on Visualization and Computer Graphics 23*, 7 (2017), 1739–1752. 2

[PM08] PAULOVICH F. V., MINGHIM R.: HiPP: A Novel Hierarchical Point Placement Strategy and its Application to the Exploration of Document Collections. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1229–1236. 1, 2

[PTM*20] PEZZOTTI N., THIJSSEN J., MORDVINTSEV A., HÖLLT T., VAN LEW B., LELIEVELDT B. P., EISEMANN E., VILANOVA A.: GPGPU Linear Complexity t-SNE Optimization. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (2020), 1172–1181. 2

[PWR04] PENG W., WARD M. O., RUNDENSTEINER E. A.: Clutter Reduction in Multi-Dimensional Data Visualization Using Dimension Reordering. In *IEEE Symposium on Information Visualization* (2004), pp. 89–96. 1

[RFZ08] RIVEIRO M., FALKMAN G., ZIEMKE T.: Improving maritime anomaly detection and situation awareness through interactive visualization. In *2008 11th International Conference on Information Fusion* (2008), pp. 1–8. 2

[Sam69] SAMMON J. W.: A Nonlinear Mapping for Data Structure Analysis. *IEEE Transactions on Computers C-18*, 5 (1969), 401–409. 1

[SASS16] SCHULZ H. J., ANGELINI M., SANTUCCI G., SCHUMANN H.: An Enhanced Visualization Process Model for Incremental Visualization. *IEEE Transactions on Visualization and Computer Graphics 22*, 7 (2016), 1830–1842. 2, 3

[Say17] SAYOOD K.: *Introduction to Data Compression*. Morgan Kaufmann, 2017. 2

[Scu10] SCULLEY D.: Web-scale k-means clustering. In *Proceedings of the 19th International Conference on World Wide Web* (2010), WWW '10, Association for Computing Machinery, pp. 1177–1178. 2

[SPG14] STOLPER C. D., PERER A., GOTZ D.: Progressive Visual Analytics: User-Driven Visual Exploration of In-Progress Analytics. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 1653–1662. 1, 2

[SZS*17] SACHA D., ZHANG L., SEDLMAIR M., LEE J. A., PELTONEN J., WEISKOPF D., NORTH S. C., KEIM D. A.: Visual Interaction with Dimensionality Reduction: A Structured Literature Analysis. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 241–250. 2

[Ult90] ULTSCH A.: Kohonen's self organizing feature maps for exploratory data analysis. *Proc. INNC90* (1990), 305–308. 1

[VDM14] VAN DER MAATEN L.: Accelerating t-SNE using tree-based algorithms. *The Journal of Machine Learning Research 15*, 1 (2014), 3221–3245. 2

[vdMH08] VAN DER MAATEN L., HINTON G.: Visualizing data using t-SNE. *Journal of machine learning research 9* (2008), 2579–2605. 1

[VR19] VENTOCILLA E., RIVEIRO M.: Visual Growing Neural Gas for Exploratory Data Analysis. In *10th International Conference on Information Visualization Theory and Applications* (2019), pp. 58–71. 1, 2

[ZRL96] ZHANG T., RAMAKRISHNAN R., LIVNY M.: BIRCH: An efficient data clustering method for very large databases. *ACM SIGMOD Record 25*, 2 (1996), 103–114. 2

[ZYQ*08] ZHOU H., YUAN X., QU H., CUI W., CHEN B.: Visual Clustering in Parallel Coordinates. *Computer Graphics Forum 27*, 3 (2008), 1047–1054. 2