188.413 Self-Organizing Systems(VU 3.0)

HandsOn Assignments: SOM – Data Exploration

This is the description of assignment 3. It is focused on extensive analysis of data using the SOM toolbox in Python (other toolboxes may be used as well, but hardly any provide the number and quality of visualizations required to explore data solidly)

Goal of this exercise is to get in-depth knowledge on training and interpreting Self-Organizing Maps (SOMs) using and combing a variety of visualizations and understanding the impact of different parameter settings and (semi-)automatically documenting the provenance. At the same time we aim at reducing the effort required for preparing reports, focusing on the core insights to be documented (and included in the provenance documentation), forming a kind of electronic lab notebook.

Note that documentation (and grading) of your experiments is predominantly performed based on the provenance information and descriptions thereof that you provide as integral part of your notebook in PROV-O and automatically logged in the provenance knowledge graph for this course. (Note that the focus in grading will not be on the correctness of the usage of the ontologies. Similarly, if you fail to find a specific concept in an ontology to model information using a free-form field instead, this is also ok. You should aim for as much machine-actionability as reasonably possible, but it should not turn into an exercise on ontology exegesis.)

In addition to the provenance information collected you will have to submit a **short report** based on the structure in this assignment paper which summarizes the key aspects of each processing step, and which can be (semi-)automatically created from your provenance logs in the graph database (if you want to) by exporting the respective information into e.g. a LaTeX document, augmented with any additional observations and discussions, specifically interpretations of results if they are not logged in the knowledge graph.

A) SOM Toolbox or Python / Jupyter SOM Toolbox

Download

- the Python version of the SOM Toolbox, with an associated Jupyter notebook (*training_assignment.ipynb*), available at https://github.com/smnishko/PySOMVis.
- The template Jupyter Notebook showing how to set-up and document the provenance of your range of experiment runs from your notebook in a semi-automatic manner, is available in TUWEL.
- Use PROV-O to automatically document all experiments / runs of individual cells in your notebook in a **Knowledge Graph (KG)** via the infrastructure provided, relying on suitable ontologies wherever possible. Specifically, we recommend using the ontologies listed below. If you find that some information you want to represent cannot be properly represented by these, feel free to use other ontologies that you are aware of, or use controlled vocabularies or free-form text to represent this information. Recommended ontologies include:
    - Provenance
        - PROV-O:
            - doc: https://www.w3.org/TR/prov-o/
            - serialization: https://www.w3.org/ns/prov-o
    - Data:
        - schema.org:
            - doc: https://schema.org/Dataset
            - serialization: https://schema.org/version/latest/schemaorg-current-https.ttl
        - Crossaint
            - doc: https://docs.mlcommons.org/croissant/docs/croissant-spec.html
            - serialization: https://github.com/mlcommons/croissant/blob/main/docs/croissant.ttl
        - Units of Measurement:
            - SI Digital Framework
                - doc: https://github.com/TheBIPM/SI_Digital_Framework/blob/main/SI_Reference_Point/docs/README.md
                - doc: https://si-digital-framework.org/
                - doc: https://si-digital-framework.org/SI

            - serialization: https://github.com/TheBIPM/SI_Digital_Framework/blob/main/SI_Reference_Point/TTL/si.ttl
        - Quantities and Units
            - doc: https://www.omg.org/spec/Commons
            - serialization: https://www.omg.org/spec/Commons/QuantitiesAndUnits.ttl
- ML Experiments:
    - MLSO:
        - doc: https://github.com/dtai-kg/MLSO
        - doc: https://dtai-kg.github.io/MLSO/#http://w3id.org/
        - serialization: https://dtai-kg.github.io/MLSO/ontology.ttl
- Terminology:
    - ISO22989: Artificial intelligence concepts and terminology (use as controlled vocabulary)

B) Dataset

1) **Select a data set** from the OpenML Machine Learning Repository (http://www.openml.org) with the *following requirements:*
    a. minimum 1000 instances,
    b. minimum 20 attributes,
    c. minimum 4 class labels (for visualizing class distributions on the map).

    Alternatively, you can also
    - opt to create an **artificial dataset**, preferably via parameterized scripts (in Matlab, Java, R, Python…) similar to the 10-Gaussians dataset, creating data of different densities combining
        i. Data on a finite area of a 1-d (line), 2-d, 3-d, 5-d hyperplanes
        ii. Data on (hyper-)spheres with different radius as well as Gaussians
        iii. Linear data sets in different intertwined settings
        iv. Other cluster characteristics that you find interesting
2) ***Register* the dataset** you picked with your group number in the TUWEL Wiki. You must make sure that your dataset is unique, i.e. no two groups may take the same data set! (first come, first serve - do it early to get a data set that you also find interesting to work.)
3) ***Create a machine-actionable description*** of the dataset following Croissant / Schema.org descriptions for datasets (c.f. Croissant: https://neurips.cc/virtual/2024/poster/97627, https://docs.mlcommons.org/croissant/docs/croissant-spec.html; schema.org: https://schema.org/Dataset, c.f. the JSON example provided at https://schema.org/Dataset#eg-0478)
4) **Analyze and describe the characteristics** of the dataset (size, attribute types as discussed in class, value ranges, sparsity, min/max values, outliers, missing values, correlations, ...), and describe this in the provenance graph. Also, describe any hypotheses you might have concerning the distribution of the data, number of clusters and their relationship, majority/minority classes as rdf comment field in the provenance graph.
5) **Preprocessing**: Get the data into the form needed for training SOMs. Describe your preprocessing steps (e.g. transcoding, scaling), *why* you did it and *how* you did it. Specifically, if your dataset turns out to be extremely large (very high-dimensional and huge number of vectors so that it does not fit into memory for training SOMs) you may choose to apply subsampling for the training data.

C) SOM Training and Analysis

1) Train a reasonably sized „regular“ SOM
    - Train a SOM with „regular“ size (i.e. number of units as a certain fraction of the number of data items) and reasonable training parameters (sufficiently large initial neighborhood, learning rate; provide a justification for the selection of the parameters. NOTE: Learning rates for SOMs differ from those usually encountered in Deep Neural Networks, c.f. lecture)

- Analyse in detail the class distribution, cluster structure, quantization errors, topology violations. a) Can you identify the border effect and magnification factors. b) How well do class distribution and cluster structure match? c) Which classes fall into sub-clusters, which classes are split across clusters, which classes mix in clusters. d) How is the quantization error distributed on the map, how does this correspond with perceived cluster separation and quality?
- **Describe and compare the structures found** (providing detailed info on visualizations and parameters)

2) Analyze different **initializations of the SOM**:
   - Train one further „regular-sized“ SOM using the same training parameters as above, but using a different random seed for initializing the SOM.
   - **Show and describe** a) how the cluster structures and class distributions shift on the two SOMs, b) the effect on topology violations, cluster relationships, etc. c) Which clusters show a stable relationship, which ones change their relative position? d) Which data instances are stably mapped with similar data instances, which change a lot? Are they part of the same clusters?
   - **Describe and compare the structures found** (providing detailed info on visualizations and parameters)
3) Analyze different **map sizes**:
   - Train 2 additional SOMs varying the size (very small / very large) (provide reasons for choice of sizes)
   - Train each map with rather large neighborhood radius and high learning rate (provide reasons for the definition of „high“!)
   - Analyse in detail the a) class distribution, b) cluster structure, c) quantization errors, d) topology violations. Also, e) analyze how clusters shift, change in relative size, and how their relative position to each other changes or remains the same. f) Check for aspects such as magnification factors. What is the resulting granularity of clusters visible on the small and large maps? Are the same clusters visible in the very large map as in the regular map?
   - **Describe and compare the structures found** (providing detailed info on visualizations and parameters)

4) Analyze different **initial neighborhood radius settings**:
   - Train the very large SOM as specified above, but with a much too small neighborhood radius.
   - Analyse the a) cluster structure, b) quantization errors, c) topology violations. d) In how far does this map differ from the very large map trained with a correct/high initial neighborhood radius?
   - **Describe and compare the structures found** (what is the effect of a „too small“ neighborhood radius? How to detect it?)

5) Analyze different **initial learning rates**:
   - Train the regular-sized SOM as specified above, but with a (I) much too large / (II) much too small learning rate (provide justification for the setting of the parameter)
   - Analyse for both (I) and (II) a) cluster structure, b) quantization errors, c) topology violations. d) In how far do these two maps differ from the well-trained map analyzed above?
   - **Describe and compare the structures found** (how can you detect „too small“ learning rates? When do they start to make sense?

6) Analyze different **max iterations**:
   - Train a regular SOM using 2, 5, 10, 50, 100, 1000, 5000, 10000 iterations
   - Analyse cluster structure. a) When do cluster structures start to emerge? b) After how many iterations do they stabilize? c) How can you tell from the quality measures whether the map is stable? d) Which visualizations help you discover not-yet stable SOM mappings?
   - **Describe and compare the structures found** (what is the effect of a „too low“ number of iterations, when does it start to converge properly/lead to reasonable structures?)

7) Detailed analysis of an „**Optimal SOM**“:

- Train a SOM using what you consider to be „optimal parameters“ based on sub-tasks 1-6.
- Describe the final model following MLSO.
- Provide a detailed interpretation of the cluster/class structures using a combination of visualizations and their parameter settings. Describe the findings in detail, specifically analyzing and providing rationale for
    a. Cluster densities / cardinalities, shapes: what can you tell about the cluster sizes shapes, their cardinalities and densities? Can you observe areas of higher/lower densities? Compare different visualizations that support (or contradict) your hypothesis and reason/explain why they do so.
    b. Hierarchical cluster relationships: can you detect any hierarchies in the data? How do they seem to be structured? Which clusters are similar, which are very distant, how could they be related? Compare different visualizations that support (or contradict) your hypothesis and reason/explain why they do so.
    c. Topological relations / violations: in which areas can you observe topology violations? What types of violations do you observe in which areas of the map (i.e. actual violations due to bad training or the inherent structure of the data vs. cluster data that is mapped onto the plane). In how far do different visualizations agree on these violations? Compare different visualizations that support (or contradict) your hypothesis and reason/explain why they do so.
    d. Class distribution: Which classes are mapped onto which parts of the map? How do they relate to each other? In how far does the class distribution match the cluster structure? Which classes are well-separated, which ones less so? What might be the reason for these overlaps? Is the mapping less correct in these regions (e.g. higher error measures)? Are these areas well-separated. Which classes form homogeneous clusters, which form sub-clusters, how similar are these sub-clusters?
    e. Quality of the map in terms of vector quantization and topology violation: is the quality homogeneous, are there certain areas or classes where the quality of the mapping is lower, others where it is higher?

D) Summarize your findings

Usually, you will not need to log this into the provenance graph. Add this only in the final report. You will be able to create a large part of this report automatically via the documentation provided in the provenance graph. Once you have generated the according LaTeX output (c.f. template notebook provided) you can save the file and manually edit it using any editor or upload it into Overleaf (*https://overleaf.com*), using your TUWIEN account via the single-sign on (SSO) option to continue working on it. Alternatively, you can, of course, also simply export plain text output and use any other word processing software of your choice to prepare the final report.
Add comments, explanations beyond the comments added into and retrieved from the provenance graph.

1. Summarize your overall findings and lessons learned:
    a. Which parameters have what kind of influence on the SOM?
    b. How sensitive is the setting of these parameters
    c. Which visualizations are most useful to reveal what kind of information? Which combination of visualizations is most useful? Which visualizations turned out to be less useful and why so?
2. (**optional**) Provide feedback on the exercise in general: which parts were useful / less useful; which other kind of experiment would have been interesting, … (this section is, obviously, optional and will not be considered for grading. You may also decide to provide that kind of feedback anonymously via the feedback mechanism in TISS – in any case we would appreciate learning about it to adjust the exercises for next year.)

Submission guidelines:

- **Upload ONE [zip/tgz/rar] file** to TUWEL that **contains all your files** (all scripts/programs you wrote, JSON-files, and subsidiary information for repeating experiments), the **Jupyter Notebook**, the **report**

as a PDF file including the source files (Word/OpenOffice/LibreOffice/LaTeX source files) **inside that zip file.** You must follow this naming convention:

  - *SOS2025_ExSOM-Exp_group<groupno>_<studentID1>_<studentID2>.zip*
  - Example: A submission of group 99 with 2 students (ids: 019999, 029999) looks like this: SOS2025_ExSOM-Exp_group99_0019999_0029999.[zip/tgz/rar]
  - Apply the same naming convention to the PDF report (but obviously with pdf extension)

- https://www.acm.org/publications/proceedings-template. (Conference Proceedings Style File, 2-column layout) LaTeX recommended (you may use the Overleaf Template provided at https://www.overleaf.com/latex/templates/acm-conference-proceedings-primary-article-template/wbvnghjbzwpc), but Word/OpenOffice template is obviously also ok. Those submitting a Jupyter notebook (with all textual content/analysis/explanations enumerated and described in-line) must submit a PDF export of the fully computed notebook in addition to the notebook.
- **Put your names and your student-IDs in the report** (as author info)
- **Use graphs** to visualize findings. Do not just show graphs, also describe what they mean. (This also applies to the Jupyter-NB PDF exports!)
- **Use tables** to combine findings and other information for maximum overview whenever possible. Describe what the figures show and explain the data. Clarify, don't mystify.
- **Enumerate and label ALL figures, equations and tables** and refer to them in the report. Describe, explain and integrate them with the text. It must be clear to the reader what information can be learned from them.
- Ensure that all figures, experiments etc. are provided with sufficiently detailed information to re-create them based on the information provided. (reproducibility!)
- Make sure the **structure of the report** follows the **structure of the tasks** provided here.

General advice:

- Reserve plenty of **time for "playing" with the data** and start early.
- **Collaboration between groups** is welcome, **but** ensure your group uses a **unique data set.**
- **Collaboration inside the group**: Try to perform at least part of the experiments within the group together. Specifically discuss the results amongst each other. Subdividing and solving tasks alone will cost you more time and not meet the goals of the exercise. Specifically, we discourage splitting the tasks for part C onto the group members. Collaborate, brainstorm and discuss what you find.
- Try to **understand your results** and note down any peculiar observations you make, try to provoke "wrong" behavior of the algorithms (over-learning, strange parameters, test wrong encodings, absurd scalings,...) and report these findings as well. Work with the data and with the settings.