HandsOn Assignments: SOM

This is the description of the "data exploration" option for assignment 3, i.e. for groups preferring an extensive data exploration / evaluation session over an implementation assignment. It is focused on extensive analysis of data using the SOM toolbox in Java or Python (other toolboxes may be used as well, but hardly any provide the number and quality of visualizations required to explore data solidly) Students wanting to focus on coding rather than experiments can take the coding option provided separately, which focuses on implementing a specific set of visualizations.

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.

A) SOM Toolbox or Python / Jupyter SOM Toolbox

Download either

- the SOM Toolbox (implemented in Java) and read the Step-by-Step Guide available at http://www.ifs.tuwien.ac.at/dm/somtoolbox.
- or the Python version of the SOM Toolbox, with an associated Jupyter notebook (*training_assignment.ipynb*), available at https://github.com/smnishko/PySOMVis. (if you choose this option you may submit the Jupyter notebook (with detailed textual explanations, as required for the report) instead of the report document. Just ensure that all content requested for the report is also present in the notebook (i.e. multiple figures comparing different parameter settings, are displayed next to each other for comparison and the differences analyzed and described, etc.). In this case, please submit both the notebook as well as a PDF print-out of the notebook.)
- or SOMbrero (based on R), available at http://sombrero.nathalievialaneix.eu/index.html, with documentation being available at http://sombrero.nathalievialaneix.eu/articles/c-doc-numericSOM.html

Play around with some of the pre-trained maps (there are libraries to read the pre-trained maps from the Java Toolbox into the Python framework), exploring different visualizations, getting a feeling for parameters, representation of clusters, and interpreting visualizations.

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 in
     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
   - use a **dataset of your own choice** that fulfills the above criteria (please contact me briefly on any alternative data sets that you plan to use/generate and make sure the data can be shared)
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) ***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 report. Also, describe any hypotheses you might have concerning the distribution of the data, number of clusters and their relationship, majority/minority classes.
4) **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.

HandsOn Assignments: SOM

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)
   - Analyse in detail the class distribution, cluster structure, quantization errors, topology violations. Analyze the border effect and magnification factors. How well do class distribution and cluster structure match? Which classes fall into sub-clusters, which classes are split across clusters, which classes mix in clusters. 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 two further „regular-sized“ SOMs using the same training parameters as above, but using two different random seeds for initializing the SOM.
   - **Show and describe** how the cluster structures and class distributions shift on the 3 SOMs, the effect on topology violations, cluster relationships, etc. Which clusters show a stable relationship, which ones change their relative position? Which data points are stably mapped with similar data points, which change a lot? Are they part of the same clusters? Where do more changes happen?
   - **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 class distribution, cluster structure, quantization errors, topology violations. Also, analyze how clusters shift, change in relative size, and how their relative position to each other changes or remains the same. Check for aspects such as magnification factors. What is the resulting granularity of clusters visible on the small and large maps? Can the according granularity also be found on the larger 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 two SOMs („regular“ / very large) with much too large / much too small neighborhood radius
   - Analyse the cluster structure, quantization errors, topology violations. In how far do these two maps differ from the maps analyzed above?
   - **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 two SOMs („regular“ / very large) with much too large / small learning rate
   - Analyse cluster structure, quantization errors, topology violations. In how far do these two maps differ from the maps 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 **scalings**:
   - Train a „regular“ SOM with obviously wrongly scaled data

- Analyse cluster structure, quantization errors, topology violations. In how far does this map differ from the maps analyzed above?
- **Describe and compare the structures found** (providing detailed info on visualizations and parameters)

7) Analyze different **max iterations**:
   - Train a regular SOM using 2, 5, 10, 50, 100, 1000, 5000, 10000 iterations
   - Analyse cluster structure, quantization errors, topology violations. When do cluster structures start to emerge? After how many iterations do they stabilize? How can you tell from the quality measures whether the map is stable? Which visualizations help you discover not-yet stable SOM mappings? Which quality measures provide good indicators?
   - **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?)

8) Detailed analysis of an „**Optimal SOM**“:
   - Train a SOM using what you consider to be „optimal parameters“ based on sub-tasks 1-7. Provide a detailed interpretation of the cluster/class structures using a combination of visualizations and their parameter settings. Describe these 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

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.)

HandsOn Assignments: SOM

HandsOn Assignments: SOM

Submission guidelines:

- **Upload ONE [zip/tgz/rar] file** to TUWEL that **contains all your files** (all scripts/programs you wrote and subsidiary information for repeating experiments), 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:
  - *SOS2022_ExSOM-A_group<groupno>_<studentID1>_<studentID2>_<studentID3>.zip*
  - Example: A submission of group 99 with 3 students (ids: 019999, 029999, 039999) looks like this: SOS2022_ExSOM-A_group99_0019999_0029999_0039999.[zip/tgz/rar]
  - Apply the same naming convention to the report (but obviously with pdf extension)
- Follow the **ACM formatting guidelines**, using the templates provided at 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/gallery/tagged/acm-official#.WOuOk2e1taQ), but Word/OpenOffice template is obviously also ok. Those submitting a Jupyter notebook (with all textual content/analysis/explanations in-line) must submit a PDF export of the fully computed notebook in addition to the notebook.
- **Put your names and your studentIDs in the report** (as author info)
- **Use graphs** to visualize findings. Do not just show graphs, also describe what they mean.
- **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.

Hints on the toolbox:

The following hints are aggregations and quotes from postings/discussions in the last year(s). Please use the TUWEL forum to discuss issues found, specifically those relating to heterogeneous OS environments.

(1) Installer vs. Source code:

The installer provided for Windows platforms does not seem to work for Windows 10. Please use the source code distribution instead.

(2) Mac OS X users: somtoolbox.sh throwing an error:

```
Unable to find feedback application.
Oct 29, 2016 7:39:03 PM at.tuwien.ifs.somtoolbox.apps.SOMToolboxMain main
SEVERE: Runnable "somtoolbox.sh" not found.
```

somtoolbox.sh uses the linux function "readlink". This functions differently on Mac OS X. The result is a blank $REALPATH. This causes the if/else "Check for the executable" to choose the wrong case.

For Mac OS X users, replace
```
REALPATH=$(readlink -f "$0" || true)
```

with
```
REALPATH=`perl -e 'use Cwd "abs_path";print abs_path(shift)' $0`
```

(2) ARFF file:
Quite a lot of machine learning datasets are distributed in ARFF format.
The SOMtoolbox contains a converter that can parse ARFF and write SOMLib input files.
It is available in the toolbox in the "Helper" Section, 3rd item there: InputDataFileFormatConverter.
(The same can easily be achieved also via any text editor, but the converter is more convenient)

Parsing ARFF files: ensure that there are no quotation marks around the attribute labels:

i.e. NOT: `@attribute 'y-box' integer`
but rather: `@attribute y-box integer`

Also, note that the SOM cannot work with nominal values. Thus, all attribute values need to be converted (1-to-n coding) to independent attributes first. Otherwise, calling the ARFF converter will result in throwing a NumberformatException when calling e.g.
```
./somtoolbox.sh InputDataFileFormatConverter --inputFormat ARFF som/XXX.arff --
outputFormat SOMLib som/XXX
```

(4) Scalability
The SOM toolbox works in 32bit mode. It can handle a maximum of roughly 10000-15000 vectors (depending on dimensionality, obviously) before it runs out of memory with the default of 3.6GiB. Also more than 50000 vectors will result in NegativeArraySizeException errors with certain visualizations which are only logged to the terminal and not shown in a window. If one does not look at the terminal output for a while, those are easy to miss and can cause quite some additional work later on. The visualization then simply does not appear.