Introduction to Data Visualization

Hasso-Plattner-Institut
Fachgebiet Computergrafische Systeme
Prof. Dr. Jürgen Döllner

Winter Term 2021/2022 – Assignment Sheet # 4

Objectives

With these assignments you will learn

- how to evaluate sample visualizations;
- general placement strategies for glyphs;
- how to implement a treemap using D3.js;
- how to apply the dimension reduction techniques PCA and t-SNE for visualization tasks.

For your programming tasks, you will use Python and its Machine Learning library scikit-learn, and JavaScript and its library D3.js for creating visualizations. You are expected to comment and document your code in a focused and clear manner.

Your solutions must be uploaded via Moodle by **January 20, 2022, 9am (UTC+1)** as one ZIP file that contains all answers and source files. The naming convention for this ZIP file is **sheet4_<group_name>.zip**.

Instructions

Implementation As visualization framework we use *D3.js* and its respective language JavaScript; for details we refer to its documentation[1].

The implementations with D3.js should be implemented as a web page based using the framework. Make sure that relative paths are used. Supplementary exercise-specific data should be organized in a sub-folder called "data". Color schemes, in particular, can be explored and loaded via colorbrewer2.org[2]. Local provisioning for debugging and testing can be done using a local web server such as the http-server by node[3]. The server can be installed with the following command:

- npm install http-server -g

In order to load e.g. CSV files you need to start a local server by

- http-server -a 127.0.0.1 -o

in your working directory containing both the sources and the data.

Pair Programming On these assignments, you are encouraged (not required) to work with a partner provided you practice pair programming. Pair programming "is a practice in which two programmers work side-by-side at one computer, continuously collaborating on the same design, algorithm, code, or test." One partner is driving (designing and typing the code) while the other is navigating (reviewing the work, identifying bugs, and asking questions). The two partners switch roles every 30–40 minutes and, on demand, brainstorm.

Violation of Rules A violation of rules results in grading the affected assignments with 0 points.

- Writing code with a partner without following the pair programming instructions listed above (e.g., if one partner does not participate in the process) is a serious violation of the course collaboration policy.
- Plagiarism represents a serious violation of the course policy.

[1] D3.js documentation
[2] COLORBREWER 2.0
[3] Node.js®

Exercise 4.1: Evaluation of Visualization Examples (4 Points, Theory)

In the zip file of this exercise sheet you will find four screenshots of visualization examples from different applications and domains. Describe and explain a selected aspect that conceptually represents a weak point in the respective visualization example.

Your submission should contain a single PDF-file.

Exercise 4.2: Glyph Placement Strategies (4 Points, Theory)

In the paper "A taxonomy of glyph placement strategies for multidimensional data visualization", Matthew O. Ward systematically summarizes strategies for designing visualisations using glyphs. Read the paper included in the zip folder and answer the following questions:

a) What is the difference between structure-driven and data-driven placement strategies?

b) Describe one advantage and one disadvantage of glyph overlapping.

c) What is the main reason for applying distortion techniques?

Your submission should contain a single PDF-file.

Exercise 4.3: 2D Treemap for Visualizing Hierarchical Structure of a Software Project (8 Points)

Treemaps are a class of 2D layout techniques for visualizing hierarchically structured data, where each data point is represented as 2D geometry; in this case rectangles. Because files of a software project are organized according to a certain folder structure, treemaps can be used for software visualization tasks.

The goal of this task is to generate a 2D treemap for the open source project *globjects*[4]. The data set *BasicMetrics-Teaching-glojbects.csv* contains two basic software metrics, stored in the following columns:

- *name* contains the name of the source code file;
- *LoC* stores the Lines of Code of each source code file;
- *NoC* stores the number of commented lines in each file.

Treemaps generated with *D3.js* can be found in the "D3 Gallery"[5]. To create a treemap, proceed with the following steps:

a) For each file compute the position, width and height of a rectangle according to a treemap algorithm of your choice using a *D3.js* layout generator.

b) Extend your implementation with a visual mapping to the color of each rectangle.

c) Allow the user to explore the visualization by integrating a tooltip for each rectangle, that appears when hovering over a rectangle.

Your submission to this exercise should contain a single HTML-file.

Exercise 4.4: Dimension Reduction of Fashion-MNIST dataset (6 Points)

In the last exercise session, the generation of a 2D layout for the *MNIST handwritten digit database*[6]was presented and discussed. In this task, you will implement a Python script using *scikit-learn* for generating a 2D layout that displays the local structure of the elements of the dataset *MNIST-Fashion.csv*[7]. It consists of 60000 greyscale pictures of size 28x28 pixels; that is, each image can be considered as a 784 dimensional vector. Figure 1 shows ten samples of the dataset. To create a 2D layout, proceed as follows:

[4] `https://github.com/cginternals/globjects`
[5] `https://observablehq.com/@d3/gallery`
[6] `https://tensorflow.org/datasets/catalog/mnist`
[7] `https://github.com/zalandoresearch/fashion-mnist`

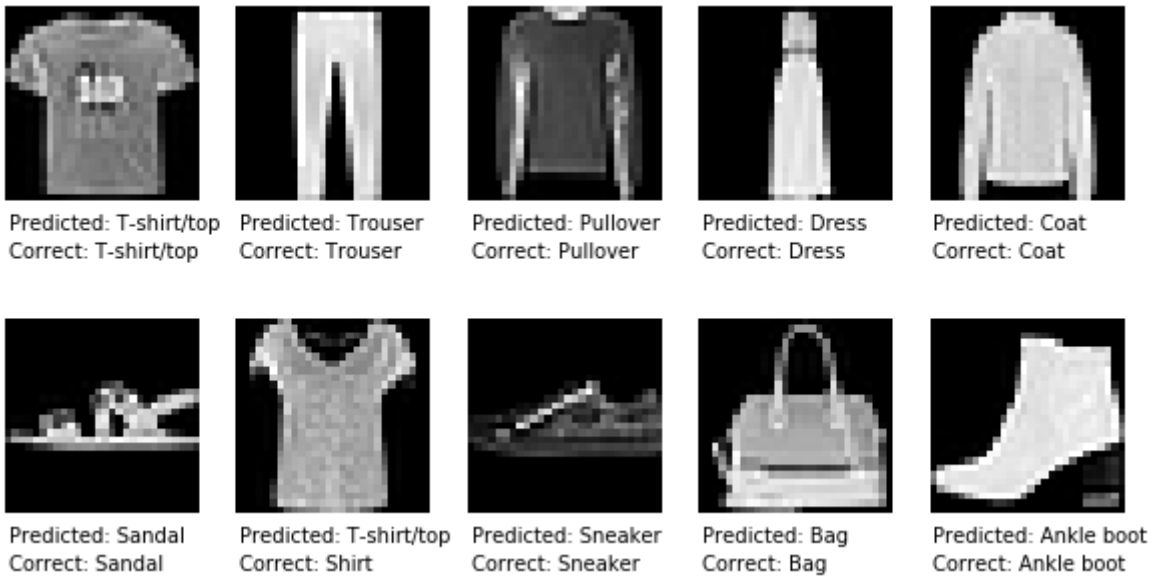

Figure 1: Examples for data points of the *Fashion-MNIST* dataset.

a) Reduce the initial size of dimension using Principal Component Analysis (PCA). Ensure to choose the minimum number of principal components to obtain at least 95% of variance in the data.

b) Modify the parameters of the t-SNE algorithm to represent each data item as a tuple of real numbers. Pay attention that your solution keeps local properties of the dataset, that is, data points of the same category should be placed close to each other.

c) Visualize your results in a 2D scatter plot using *matplotlib*. Apply an appropriate color scheme that reflects the label of each data point.

Your submission should contain a single Python script together with a pipenv, so that your implemenation can be executed. Your final script must generate a 2D scatter plot.