cairo.thws
Center für Künstliche Intelligenz und Robotik
Technische Hochschule Würzburg-Schweinfurt

Portfolio

Learning of Structured Data – Sheet 3 –

Fakultät Informatik und Wirtschaftsinformatik

Lecturers:
Prof. Dr. Dominik Seuß
Prof. Dr. Andreas Lehrmann

Portfolio design:
Prof. Dr. rer. nat. habil. Frank-Michael Schleif

Release date: 27/12/24

name(s) & matr. no.:

Portfolio 3 consists of 2 exercises and is related to a real-world classification challenge.

The data consists of motion sequences of 33 elderly people playing charades and is divided into a training set (27 subjects) and a test set (6 subjects). Each file consists of one subject performing one of 5 possible actions: "boxing", "drums", "guitar", "rowing" and "violin". See `data/README.md` for a detailed description of the file structure.

Your goal is to leverage your knowledge of encoding techniques, proximity measures, and classification methods to train a model that predicts the actions in the test set as accurately as possible. Each exercise is worth 10 points, so you need to finish **both exercises** for full points.

The solution consists of theory (submit as PDF file; sufficiently detailed and explained), coding (submit as Python file; follow good coding practices), and a video (submit as MP4; ensure proper encoding).

For this third sheet you are expected to submit in teams of 2-3.

Exercise 1: (Data) Loading & Visualization

- Write an import script that loads the *.csv* files in `data/train/` and `data/test/` into a Python `Dataset` class that enables access to individual sequences. At the minimum, your class must support the following methods:
  - `getSequence(self, dataset, subject, action, iteration)`: returns a tuple `(seq,label)`, where `seq` contains the specified sequence (iteration i of action j by subject k in dataset 'train' or 'test') and `label` the associated label.
  - `random(self, dataset)`: returns a random tuple `(seq,label)` from `dataset` ('train' or 'test'), where `seq` contains a sequence and `label` the associated label.

  Demonstrate your dataset class by accessing and printing (a) the sequence/label corresponding to file *"p0_boxing_01.csv"*; and (b) a random sequence.

- Augment your `Dataset` class with a method
  - `visualize(self, dataset, subject, action, iteration, frame)`

  that saves a visual representation similar to Figure 1 of the specified frame to disk. Save the 125 frame visualizations corresponding to file *"p0_boxing_01.csv"* to disk and convert the individual frames into an *.mp4* video (frame rate: 30 frames per second). Submit the video.

  Hint: `ffmpeg` might be useful to convert an image sequence to a video.[1]

[1] https://hamelot.io/visualization/using-ffmpeg-to-convert-a-set-of-images-into-a-video/

Due date:
13/01/25

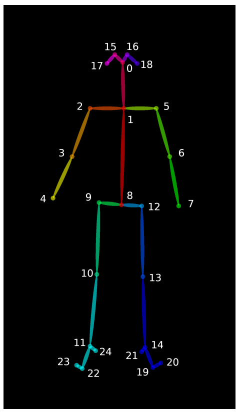

Figure 1: **Skeleton Visualization.**

Exercise 2: (Model) Design and Evaluation

In class we discussed numerous data representations (e.g., vectorial/non-vectorial, observable/latent, graph-based, proximity-based, etc.) and model architectures (e.g., classification, regression, clustering, etc.). Build a system that is appropriate for the type of data and task at hand.

Remember that the test set cannot be used during any part of the training process; if you want to optimize hyperparameters you are free to use a subset of the training set as a validation set (or perform cross validation).

Exercise 2.1 (Design):

- Preprocess the data:
  - do you need to remove outliers from the training set?
    [*you are not allowed to remove data from the test set!*]
  - do you need to perform normalization on the training/test set?
- Choose a data representation and process the data accordingly. There is a wide spectrum of possible representations: simple features/statistics (e.g., joint locations, histogram counts), latent embeddings/codes (e.g., spectral, variational), proximity measures (e.g., kernels, dissimilarities), etc. Do what is necessary to maximize test set performance (except using the test set during training).
- Use your data representation to train a classification model on the 1167 training sequences. Once trained, your model should be able to process a motion sequence s – either at once or frame by frame – and predict its class $c \in \{\text{boxing}, \text{drums}, \text{guitar}, \text{rowing}, \text{violin}\}$. Where applicable, this step should also include hyperparameter optimization.
- Provide a **detailed** documentation of the entire pipeline, including pre-processing steps, data representation, and model architecture. Motivate the individual components (why did you choose them?), explain how they work at the technical level, and describe how you selected the hyperparameters.

Exercise 2.2 (Evaluation):

- Perform a theoretical and practical analysis of the runtime and memory consumption of your approach. Distinguish between training and testing.
- Evaluate your model on the 305 test sequences. Compute confusion matrices and total classification accuracies over five independent training/evaluation runs and report mean±standard deviation.