Report Activity Monitoring

The implementation of Sleep Monitoring described in this report was prepared by Laura Liebenow (635637), Ricardo Stolze (645740) and Jonas Franke (638210).

The main goal of this program is analyzing data, sampled from sleeping people using different sensor modalities, to learn a classification matrix for the different phases of human sleep. Therefore a Multi Layer Perceptron gets trained. Valuation is done by using a k-fold cross validation.

# Pattern Recognition Chain

First sensor data of one pair EEG channel and one EOG channel was used, but in order to achieve bigger training datasets, more sensor channel are used now, resulting in better accuracy in the end.

As preprocessing we downsampled the data to 25Hz and deleted the timestamps of all data files. The resulting arrays were saved as .npy files, in purpose to save time.   
Beneath the main python-code (mds\_excercise3.py), which loads and downsamples the .csv-files, another file is given (mds\_excercise3\_fast.py), which loads the preprocessed .npy-files to save time.

Also the label data had to be changed from ‘char’ to ‘int’ for training purpose, so we numbered the different sleep stages from 0 to 4.

The preprocessed data then is combined in one array per patient.  
Since some of the data-lengths were not matching to the amount of labels given, for 4 patients we cut the last label and a corresponding part of the sensor data of, resulting in matching data sizes.  
For data segmentation the sliding time window method without overlapping windows is used. Every window includes data measured in 30 seconds. The segmentation is done by splitting each sensors dataset into sets of 750 values (25Hz \*30sec) followed by concatenating the resulting datasets of each sensor. Using 6 different sensors this results into datasets of 4500 values per label.

In order to use a 3-fold-cross validation method, the data of each patient is mixed together randomly and divided into three partitions of same size. For each run of the MLP, one of those partitions is used as testing dataset, while the other two are used as training data, which leads to three different f1-scores.  
The mean f1-score is used as final score.

# Multi-Layer-Perceptron

To process the data, a multi-layer perceptron is used, based on the MLP implemented in exercise one. The MLP implemented in this approach consists of 3 hidden layers using rectified linear units as activation function.   
After every hidden layer a dropout is performed to diminish overfitting: 20% of the neurons in a layer are turned off and are not regarded in the next layer. Finally a softmax activation shifts the results into a value range of [0,1].  
For the model compilation it turned out in practice that the Adam optimizer leads to the best results and inheres a high accuracy. For the loss a sparse categorical crossentropy is used, since it is most suitable for multiclass characterization.  
Training is done in 15 epochs with a batch size of 256, which leads to a good performance in without taking too much time.

# Results

As mentioned before, the mean f1score calculated using a 3-fold-cross validation is used for evaluation. Most times it is between 50% and 60%. Notable is that most times there is a discrepancy between the three runs. Often, two of three k-fold-runs resulting into a f1-score which is higher than 60%, but the third run results into a score, which is lower than 50%.

The results are okay for a MLP, but better results may be achieved by using other deep learning techniques, like a convolutional neural network.