Appendix

0.1 NEAT Algorithm

0.2 Balancing Algorithm

The flow of Algorithm 1 is as follows: the main function `BalanceData` receives the datasets S structured by subjects and a minimum tolerated balancing error `balanceVTR`. First, we determine how many records are minimally available from each subject (line 4). Half of this number (`minSamples`) is randomly chosen from each person's records and added to the balanced dataset `BD` (lines 5-8). In this process, the class probabilities from the target sizes of the datasets are summed (variable c). Thus, the datasets are balanced with respect to the subjects*.

However, because of random subsampling, the classes may not be uniformly balanced, i.e., individual classes appear disproportionately often in the dataset. Therefore, one dataset is now randomly selected in turn from each person's dataset. If the data set improves the balance of the classes, it is added to the balanced data set BD, otherwise it is discarded. Depending on the design of the datasets and the class probabilities, this section does not terminate reliably, i.e., it may not be possible to achieve a complete balance. Therefore, `maxIter` ensures that the algorithm terminates eventually.

0.2.1 Choice of hyperparameters for NEAT

The NEAT algorithm can be influenced by a large number of parameters (hyperparameters). Table 1 lists the most important parameters and their values used here.

The most important hyperparameters are the population size, i. i.e. how many neural networks are evolved in parallel, and the number of generations the algorithm runs through. Here, 500 was chosen as the population size. The rule of thumb "ten times the dimension" [1, 2] was interpreted more generously here because of the additional diversity of possible mutations. The maximum number of generations was chosen to be $G_{max} = 2000$, which in preliminary tests proved to be just about feasible with respect to the runtime on the cluster.

Algorithm 1 Balancing the training data

Precondition: abs(x) Absolute value of x
Precondition: len(x) Number of elements in x

```
1: function BalanceData(S, balanceVTR = 0.01, maxIter = 100000)
2:     c ← []
3:     BD ← []                                              ▷ The balanced data
4:     minSamples ← MinSamples(S) / 2

5:     for subject in S do                                  ▷ Balance the participants
6:         D ← chose random minSamples from subject
7:         c ← c + D_out                                    ▷ Sum up the class probabilities
8:         BD ← BD ∪ D

9:     balance ← Balance(c)
10:    while balance > balanceVTR and maxIter ¿ 0 do▷ Balance the classes
11:        maxIter ← maxIter - 1
12:        for subject in subjects do
13:            D ← chose random sample from subject
14:            c_neu ← c + D_out
15:            if balance > Balance(c_neu) then    ▷ If D contributes to better balance...
16:                balance ← Balance(c_neu)
17:                c ← c_neu
18:                BD ← BD ∪ D
19:    return BD

20: function MinSamples(S)      ▷ What is the minimum number of samples each person provides?
21:    minSamples ← len(S_0)
22:    for i = 1 to i = N − 1 do
23:        minSamples ← min(minSamples, len(S_i))
24:    return minSamples

25: function Balance(c)
26:    diff ← 0
27:    sum ← c_0
28:    for i = 1 to i = N − 1 do
29:        diff ← diff + abs(c_i − c_{i−1})
30:        sum ← sum + c_i
31:    return diff / sum
```

Table 1: Used hyperparameters for the NEAT algorithm in the YAHNI implementation.

Parameter	**Werte**	**Beschreibung**
num.generations (G_{max})	5000	Number of generations that the algorithm runs
popul.size (NP)	1000	Number of different ANN that the algorithm evolves
weight.max	50	Upper limit for initialization of connection weights
weight.min	-50	Lower limit for initialization of connection weights
add.neuron.mutation.rate	0,10	Probability with which a new neuron is added
add.connection.mutation.rate	0,30	Probability with which a new connection is added
training.evalSplit	0,10	Proportion of training data that is reserved for evaluation
recurrent	disallowed	Cycles in the network are not allowed
learningRate	0,01	The learning rate used in the backpropagation method
fullyConnected	true/false	Initial networks with or without complete connections between neurons
num.hidden.neurons	Number	Initial number of hidden neurons

References

[1] Chen, S., Montgomery, J., Bolufé-Röhler, A.: Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution. Applied Intelligence **42**(3), 514–526 (2015). doi:10.1007/s10489-014-0613-2

[2] Storn, R.M.: On the usage of differential evolution for function optimization. Proceedings of North American Fuzzy Information Processing, 519–523 (1996). doi:10.1109/NAFIPS.1996.534789