Neural Engineering
Final Project
NER 2015 BCI Challenge

Nicolas Berberich
Electrical Engineering and Information Technology
Technische Universität München
Email: n.berberich@tum.de

Andreas Wiedemann
Mathematics in Science and Engineering
Technische Universität München
Email: andreas.wiedemann@tum.de

Claas Brüß
Electrical Engineering and Information Technology
Technische Universität München
Email: claas.bruess@tum.de

I. Introduction

The outlined project provides a coarse survey of methods in machine learning and preprocessing applied to EEG data. The data was provided in context of an now archived Kaggle challenge with the goal of improving error correction in brain-computer interfaces based on human response to machine feedback.

II. Challenge

A. Kaggle Platform

Founded in 2010 Kaggle offers an online platform for data analytics and machine learning competitions. The competitions are held by making prepared data sets available to all participants, which is usually done through the Kaggle platform itself. All submissions are ranked on leader boards with top positions usually being awarded with varying sums of cash, if the submission was entered before a given deadline. After the deadline challenges are archived with most data sets remaining available to interested platform users.

B. BCI Challenge @ NER 2015

This particular challenge has already been archived and was posed as part of the IEEE Neural Engineering Conference 2015 ($NER2015$) with the goal to flag errors in brain-computer interface responses through analysis of EEG data of the test subjects.

The experiment tasked test subjects with spelling out a given 5 letter word by the focusing on single characters in a 6 by 6 matrix. For this the correct character was highlighted initially to draw the attention of the test subject. This was followed up by sequential flashing animations of subsets of the 36 characters shown.
The system tried to identify all flashes that included the character in focus of the test subject by looking for a P300 wave response in the EEG data that was recorded simultaneously.
The machine determined character was shown enlarged to the test subject for 1.3 seconds as feedback. The challenged asked to analyze the EEG data that was recorded in this 1.3 second time span after the feedback to evaluate whether the system picked the character the test subject intended.

For this a test data set consisting of EEG recordings of 10 subjects and a labeled training data set consisting of EEG recordings of 16 subjects was provided. The EEG recordings include data from 56 channels with electrode placement following the extended 10-20 layout.

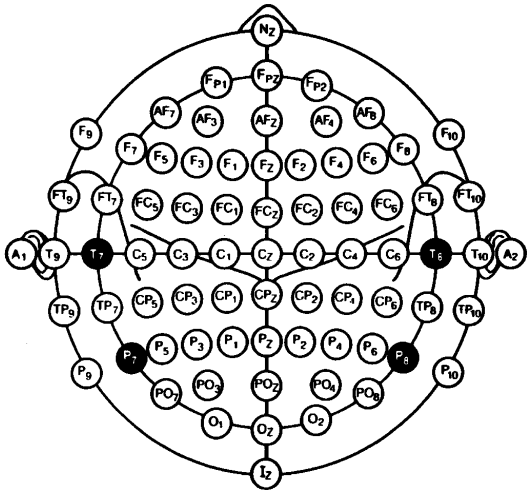

Fig. 1: Extended 10-20 Electrode Placement Layout

III. Course Relevance

This challenge blends several aspects of neural engineering and incorporates multiple topics that were addressed in this course.
Through this challenge we were tasked to analyze brain activity data obtained through EEG which touches on the neuroscientific aspects of the field.
Given the complexity of the data, machine learning is the natural approach to the posed problem. Some of the algorithms such as neural networks are brain inspired and showcase one

of the topics in neural engineering that have had the most impact in the last years.
Finally the motivation to improve classification accuracy is to enhance the detection of misinterpretations of human intend in brain-computer interface setups, which have shown promising potential in applications such as neuro prosthetics.
Bringing all of these points together

IV. Dataset

File	Size
ChannelsLocation.csv	952 B
train.zip	2.18 GB
TrainLabels.csv	100.96 KB
test.zip	3.76 GB
SampleSubmission.csv	63.11 KB

The given **training data** consists of the raw EEG records of 16 subjects. Every subject's EEG measurements were captured in five sessions, which were registered in separate CSV files. So the quantity of CSV files amounts to 80 with a size of 8.7 GB. In every session there are 60 feedbacks, only in the fifth session 100 feedbacks were provided.

The header of one CSV file has the following structure: Time, (Channel 1), ..., (Channel 56), EOG, FeedbackEvent.

- The data was recorded at a frequency of 200 Hz. Consequently the column time increases at a rate of 0.05s.
- The EEG samples of 56 channels are given in the following columns.
- The ensuing value is the EOG derivation for eye movements.
- Finally the column Feedback Event is a zero vector except for the time stamps for which the provision of feedback to the subject occurred.

The file **TrainLabels.csv** characterizes all the feedback. If the selected item differs from the expected one, the corresponding feedback is labeled 0, otherwise 1.

In the file **ChannelsLocation.csv** the polar coordinates are saved for all EEG electrodes (Channel 1, ..., Channel 56), to be more precise the radius r and angle ϕ values. These locations are the same for all sessions and subjects.

The **test files** (5.1 GB) have the same inner structure as the train files. They consist of five sessions of ten other subjects.

V. Data Preprocessing and Feature Extraction

In order to analyze the brain activity response to the machine feedback, we need to extract all the data points within the mentioned 1.3 second time frame from the complete recorded time series. This time frame encompasses 260 time series data points due to the 200Hz sampling rate. Since there were, in total, 5440 feedback events given in the training data, this lead to a training matrix of 5440x260 for every of the 56 electrodes/channels.
The obtained data set is then further processed with the following methods. This also allows for a comparison in performance to raw data and superficial investigation of the impact of the feature on different machine learning algorithms.

A. Butterworth Filter

In order to focus on the brain activity we want to omit all DC components as well as high frequency noise. We do this by applying a Butterworth filter with a pass band of 1-40Hz which safely encompasses the frequency spectrum of all relevant brain activity in the alpha and beta range. Using the Butterworth filter to obtain only the alpha waves between 8 and 13 Hz proved to be inferior. The same was true for using only the beta waves (13-30 Hz).

B. Averaged Butterworth Filter

After subjecting the data to the filter described above we additionally averaged the amplitude of the filtered signal over 1.3 seconds. This approach had the advantage, that we could then use the data from all channels instead of from only a single channel at a time without making the training matrix too large which would have lead to impracticable long training times.

C. Splitting Training Data for Cross Validation

Cross validation strategies can help accelerate the parameter tuning process during algorithm implementation. The idea is to split the provided training data into subsets and to use part of those for training and others for validation.

A range of strategies for the division of data are being used throughout the community. Here we used the n-fold cross validation method, which cuts the data into n equal chunks leaving one of them for testing while all the others for training. This repeated n times with every data chunk being used for testing once.

VI. Machine Learning Algorithms

The different supervised machine learning approaches were implemented with various Python libraries and environments in order to be able to compare accuracy and applicability of various methods for a binary classification based on EEG time series data.

A. Random Forests

Our random forests approach is implemented through the random forest classifier that is included in the sklearn.ensemble module of the scikit-learn library. This algorithm fits several decision trees (constructed with either the 'gini' impurity criterion or the 'entropy' information gain maximization criterion on sub-samples of our training data and averages them. The main tuning parameter used is the number of trees in the forests $(500 - 1000)$.

B. Support Vector Machines

Similar to the section above we use scikit-learn to implement this approach. Specifically the C-Support Vector Classification with svm.SVC. The kernel mode was set to linear with a precision parameter C set to 5. A low C makes the decision surface smooth, while a high C aims at classifying all training examples correctly. Whilst testing this approach we ran into

problems with run time on our machines (>15h) even though we down sampled the data by factors of up to 10, since the fit time complexity scales more than quadratic.

C. Gradient Boosting Machine

The scikit-learn library was also used in this implementation. The number of estimators n was either set to n=500 or n=1000. Since we have a binary classification, the sklearn Gradient Boosting Classifier only induces a single regression try which is fit on the negative gradient of the binomial deviance loss function.

D. Neural Network

As a third approach we use neural networks for interpreting and classifying the EEG data. The Python library Keras offers a simple API for setting up neural networks which can use the powerful Tensorflow backbone from Google. Keras supports not only convolutional networks, but also recurrent networks and a combination of the two as well. To enable fast computations, Keras can also access the GPU.

1) Fully Connected Neural Network: The first neural network, which we set up, is a sequential model, a linear stack of layers. As input data we utilize only one channel, giving features of length 260, as described in section preprocessing. It is a fully connected network, having three layers. Since we have not found a pretrained model which we could adapt to our task by fine tuning, we initialize the weights by small random numbers generated from a uniform distribution. As optimization algorithm we use adam, an efficient SGA. However the results are very poor and we achieve an accuracy rate of 0.53. We blame mainly the used input data, which probably lost too much information through preprocessing, and the configuration of the model. Possibly it is not large enough to capture the structure of the problem.

2) Convolutional Neural Network: Our next idea was to use a convolutional neural network. In order to do so the input data has to be transformed, such that depending on the positions of the electrodes n-channel images are generated, where a channel is depicted by a frequency band.

The library enables to convert and filter the raw EEG data in various ways and to create informative visualizations.

In the first step EEGrunt [4] is utilized to calculate theta, alpha and beta waves, which are subsequently averaged over the period of 1.3s. The averaging process is performed to obtain exactly one image per label.

As next step images must be created by transforming the polar coordinates of the electrodes into two dimensional, spatial coordinates and computing the pixel values on the basis of neighboring frequency band values.

The generation of the images is performed by an auxiliary function of the library EEGLearn [5].

Unfortunately our convolutional neural network performs poorly on the test data. Averaging of the frequency bands could be problematic.

Fig. 2: Averaged EEG images

3) : Recurrent Convolutional Neural Network For analyzing time series of EEG data Bashivan proposes a method basing on recurrent convolutional neural networks [5]. The input data has to be transformed into time series of 3-channel images, where the channels represent theta, alpha and beta waves of every electrode. Of course the location of the electrodes is of importance for creating these images, where the pixel values are inferred by interpolation. To make the representations (convolution) invariant to inter-and intra-subject differences Bashivan suggests to add noise to all images.

The corresponding library EEGLearn [5] is built upon the libraries Theano and Lasagne. EEGLearn itself simply facilitates the set up and configuration of the recurrent convolutional neural network, the rest, e.g. training, has to be implemented with the aid of the other two libraries.

VII. Instructions for Usage of the Programs

Our code is available on GitHub [6].

A. Prerequisites

- Python 2.7.11+
  - scikit-learn
  - numpy
  - tensorflow
  - keras
  - lasagne
  - theano

B. Execution

In order to run the preprocessing python programs, we downloaded train.zip and test.zip and stored them both uncompressed in the same folder.

Preprocessing files are:

- `X_train_filtered_channel.py`
- `X_train_average.py`
- Analog for testing.

For applying the various machine learning algorithms the first part represents the used input data and the second part defines the algorithm:

- `*_gbm.py`: Gradient Boosting Classifier
- `*_rf.py`: Random Forest
- `*_snn.py`: Sequential Neural Network
- `*_svm.py`: Support Vector Machine.

The programs write `predictions.csv` according to the format of `SampleSubmission.csv` which can then be uploaded to the kaggle platform to obtain the accuracy score on their test data.

In the folder **EEGLearn** more scripts basing on neural networks are stored. Preprocessing is here done by the command `py WriteDataMat.py -f(ormat) <raw|avg_img>` For both formats the script writes the file `Data.mat`, storing the variables *featMat, labels, featMatTest*.

For classifying the data there exists:

- `snn.py`: Fully Connected Neural Network
- `cnn.py`: Convolutional Neural Network
- `rcnn.py`: Recurrent Convolutional Neural Network (in work).

VIII. Results

After consulting the competitions forum we decided to focus our efforts on a limited subset of channels for training purposes. The choice made here reflects the advice given by top competitors in the competition.
All our results sorted by feature and classifier can be found below with highlighted top results for each feature type.

We discovered inconsistencies in our neural network results. While our fully connected neural network showed the encouraging result of 0.84 when tested in a 1000 fold split cross validation test case, it demonstrated random behavior when tested on the Kaggle platform. We havn't been able to resolve those issues yet, but hope to do so for future competitions.

Overall the combination of random forest classifiers with Butterworth filtered data showed the most promising results for the provided data set with accuracies of up to 70%. Within our trials, we found the data recorded from electrode 39 (CP2) to yield the most information with respect to the given task. Using n=500 decision trees in our random forest seemed to be enough. Further increasing this number did not result in better scores.
Using random forests had the advantage of being fast to train, usually between one and two minutes. The training time of our Support Vector Machine on the other hand increased exponentially with the number of data points we used. When training it with all 5440 data points, the algorithm ran for over 15 hours. After completion, the result was still worse than our results obtained with random forests.

Feature	**Classifier**	**Channel, Electrode**	**Parameter**	**Result acc**
Raw Time Series	Gradient Boosting	channel 46	n=500	0.573
		channel 46	n=1000	0.58
		channel 40	n=500	0.582
		channel 39	n=500	0.609
	Random Forest	channel 46	n=500	0.577
		channel 46	n=1000	0.574
		channel 40	n=500	0.595
		channel 39	n=500	**0.647**
		channel 40	n=1000	0.639
Filtered Time Series				
	Gradient Boosting	channel 46	n=500	0.584
		channel 39	n=500	0.0.644
	Random Forest	channel 46	n=500	0.653
		channel 46	n=1000	0.678
		channel 39	n=500	**0.697**
		channel 39	n=1000	0.684
		channel 39	n=1500	0.682
	Support Vector Machine	channel 39	5xdown, n200	0.623
		channel 39	5xdown, n500	0.636
		channel 39	10xdown, n200	0.630
		channel 39	10xdown, alle	0.624
	Fully Connected Neural Network	channel 39	Epochs = 3000	0.5 (**0.84** cross reference)
Filtered Average	Gradient Boosting	channel 39	n=500	0.518
	Random Forest	channel 39	n=500	**0.533**
Averaged three channel image	Convolutional Neural Network	Averaged theta, alpha, beta	Epochs = 3000	**0.542**

12	↑33	ThibaultV	0.70152	20	Mon, 26 Jan 2015 10:57:45 (-5.5d)	
13	↑136	A.M.	0.69865	11	Wed, 11 Feb 2015 23:39:57 (-13.5d)	
-		**Erzlearner**	0.69659	-	**Tue, 05 Jul 2016 22:36:18**	**Post-Deadline**

Post-Deadline Entry
If you would have submitted this entry during the competition, you would have been around here on the leaderboard.

14	↑29	os	0.69468	12	Sat, 21 Feb 2015 18:02:02 (-3.9d)
15	↑12	Black Swan (NER)	0.69364	39	Tue, 20 Jan 2015 01:02:22 (-6.9d)

Fig. 3: excerpt of the challenge leaderboard

To improve our score even higher we made the observation, that statistically positive feedback is given more often than negative feedback. Taking this into account in our random forest classifier, we obtained a score of 0.697 resulting in rank 14 out of 260 participants (top 6%) on the official leader board of the challenge as shown in Figure 3.

IX. Summary and Outlook

Like it was outlined in section ”III. Course Relevance” working on this challenge felt very relevant. But this fun and interesting project was not without its challenges. For algorithms such as support vector machines or different neural network longer training times were a concern and made it tougher to work on them part time.

Besides this we ran into problems with the implementation of neural networks, with some of us being new to certain libraries and tools, causing us to only provide two accuracy ratings for our efforts in neural net based approaches.

Prior knowledge about EEG systems and the characteristics of the data they produce allowed us to preprocess the data in meaningful ways which had substantial impact on the classification accuracy. We hope to bring our new insights with us into upcoming projects further exploring the possibilities for machine learning applications in EEG based brain-computer interfaces with perhaps more complicated classification schemes.

References

[1] MARGAUX, Perrin, et al. *Objective and subjective evaluation of online error correction during P300-based spelling.* Advances in Human-Computer Interaction, 2012, 2012. Jg., S. 4.

[2] https://www.kaggle.com/c/inria-bci-challenge [03.07.2016]

[3] http://scikit-learn.org/ [03.07.2016]

[4] https://github.com/curiositry/EEGrunt [04.07.2016]

[5] Pouya Bashivan, Irina Rish, Mohammed Yeasin and Noel Codella. *Learning Representations from EEG with Deep Recurrent-Convolutional Neural Networks*, 2015, https://github.com/pbashivan/EEGLearn arXiv:1511.06448.

[6] https://github.com/A-Wiedemann/Neural-Engineering