**Localization of Neural Sources from Simulated EEG Data**

**­**

First of all, thank you for the opportunity to present today. I am very excited to share some of the key experiences and skills I have developed during my studies that I believe are relevant to this position.

I am currently in the final stages of completing my master’s thesis, which I feel has greatly challenged me to combine many of the academic skills I have acquired throughout my studies. The project has demanded a high level of proficiency in structuring, programming, time management, creative thinking, and patience - and I hope that some of these experiences have equipped me with some of the necessary tools and mindset to tackle the challenges that lie ahead. I would therefor like to present to you, an overview of my master’s thesis, showcasing the progress I have made thus far.

*Background*

An important issue in EEG signal analysis is so-called EEG inverse problem where the goal is to localize the source generators, that is, the neural populations that are generating specific EEG signal components. An important example is the localization of the seizure zone in EEG recordings from patients with epilepsy (or other brain disorders causing abnormal electrical activity). By accurately localizing the source of epileptic activity, healthcare professionals can better understand the underlying mechanisms of epilepsy and make informed decisions regarding treatment options.

My thesis focuses on the utilization of machine learning techniques to train a neural network for the purpose of localizing current dipoles in the human brain, through the input of simulated EEG data.

*The Feed Forward Neural Network*

In order to “solve” the inverse problem, I have in my thesis used a FFNN. The feedforward neural network is a machine learning technique that learns from experience, by processing and analyzing data so to uncover pattern that associate characteristics of the data to the corresponding true value. The feed forward neural network is the simplest form of neural networks, as information is only processed forward, from the so-called input nodes, through the hidden nodes and to the output nodes.

FIGURE OF STRUCTURE OF NEURAL NETWORKS

The basic components of a neural network are neurons consisting of a linear transformation that weights the importance of various inputs, followed by a non-linear activation function. Activation functions are mathematical functions that make the network able to capture complex relationships and patterns that cannot be represented by simple linear functions. Without activation functions, a neural network would essentially be a linear model, capable only of representing linear relationships between inputs and outputs.

*Simulating EEG data*

Neural network algorithms need a substantial amount of data to effectively learn and understand complex patterns. Easily explained, the more data they have, the better they can capture the complicated relationships and make accurate predictions. Moreover, enough data is important not only for training the networks, but also enabling them to generalize well to new, unseen examples.

However, there simply does not exist enough, if any, EEG data, where the location of the neural current sources responsible for generating the signal is known. Therefore, the first thing I had to do even before constructing the neural network, was to find a way to simulate a great amount of realistic EEG data. This was done using the LFPy Python module, where the head model named New York Head is implemented.

The New York Head is a volume conductor, computer model of the human head used to simulate the electrical activity of the brain. It is based on the anatomical and electrical characteristics of MRI-data from 152 adult human brains and provides detailed information on the geometry and electrical properties of the different tissues and structures within the head.

The New York Head model can generate predictions of EEG signals recorded at 231 electrodes in response to a given set of source currents, out of approximately 75 000 cortical locations. The model does this by using the “lead field matrix”, which is a mathematical representation of the relationship between the electrical activity in the brain and the electrical potentials recorded on the scalp.

*Current Dipole Approximation*

To analyzing EEG signals, the neural sources are treated as current dipoles. This is because the electric potentials stemming from the neural activity of a population of neurons will tend to look like the potential from a current dipole when recorded at a sufficiently large distance, as in EEG recordings. This can be shown by utilizing the technique of multipole expansion which are beneficial in order to see that often only the first few terms are needed in order to provide an accurate approximation of the original function:

EQUATION 1

EQUATION 2

The terms denoted monopole, dipole and quadrupole represents contributions to the extracellular. Assuming that we are sufficiently far away from the source distribution, all terms above the dipole contribution vanish. As for the monopole contribution, we have that the net sum of currents over a neuronal membrane is always zero. This means that also the monopole term vanishes, and the expression for the extracellular potential is approximated by the dipole contribution alone.

FIGURE OF DIPOLES:

Electricpotentialsfromsynapticinputlookslikepotentialsfromasingledipole far away. A-C: Extracellular potential in the vicinity of a neuron receiving a single synaptic input for three different synaptic input locations. Because of the complex distribution of the membrane currents, the extracellular potentials will have a complex, but mostly dipole-like shape. A-C: Same as panel A-C, but seen from a greater distance, so that the dipole contribution is dominating the extracellular potential.

*Code snippet*

FIGURE SIMPLE DIPOLE  
FIGURE CALCULATE EEG

*Feed forward neural network:*

The provided code is a snippet of the feed forward neural network. The model is implemented using the PyTorch framework which is a library for deep learning. The network has been constructed by the use of “trial and error”- method – and finally consists of four fully connected, linear layers, which perform linear transformations on the input data. The activation functions used between the linear layers is the relu and hyperbolic tangent function, which introduces non-linearity into the network. Without going into any details, maps all negative input values to zero and keep positive values unchanged. TanH on the other hand, maps the input values between -1 and 1, providing a smooth non-linear transformation. Depending on the complexity of our problem (if we want to predict only coordinates, or radii and amplidues as well) – the number of output nodes ranges from 3 to 5.

The data set of ours consist of 75 000 samples, with 231 features each. For the training of the network, we use 50 000 samples, where 20 % of these are used as test data. The last 25 000 samples are held back, and never introduced for the network, in order to validate the performance of the network after finish training.

It is also worth mentioning that we have introduces a 10% noise to the training data in order to prevent overfitting, which occurs when a neural network becomes overly specialized to the specific training data and fails to generalize effectively to unseen datasets. By incorporating this noise, we aim to ensure that the network learns to generalize well and is capable of making accurate predictions beyond the training data.

*Results:*

The figure to the left provides the MSE for the network against epoch. An epoch here refers to a complete iteration over the entire training dataset. In other words, it is the number of times the learning algorithm has seen and processed the entire training dataset. During each epoch, the neural network processes the input data, calculates the loss, and updates the model’s parameters using an optimization algorithm. The goal is to minimize the loss, which is our case is the MSE, and improve the network’s performance on the training data.

MSE provides a measure of how well the model’s predictions align with the actual values. A lower MSE indicates that the model’s predictions are closer to the true values.

*Summary:*

To sum up, this was a short presentation of some of the things that I have worked with on my master’s thesis. However, there is still some pieces missing and results to improve – nevertheless, I wanted to give you a little insight of what I have been working with this year. And through this experience, I have developed my foundation in managing, producing, and analyzing large quantities of data, while also deepening my understanding of neuroscience.