Localization of Neural Sources from Simulated EEG Data

Kamilla Ida Julie Sulebakk

June 7, 2023

Agenda

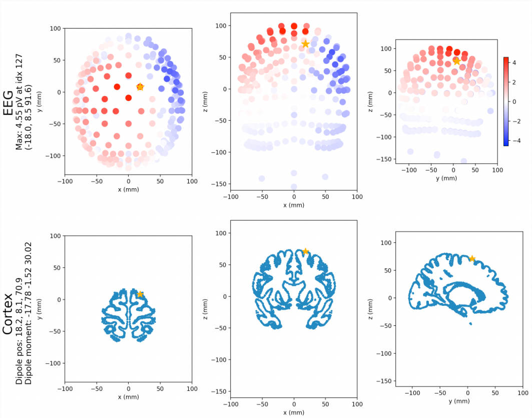
- Background – the EEG inverse problem and its relevance for medical diagnosis
- NY Head Model and the generation of EEG signals
- Basic neural network architectures
- Results and analysis
- Personal takeaways

Background

- EEG inverse problem
  - Localize the neural populations that are generating specific EEG signal components
- Seizure zone in EEG recordings from patients with epilepsy
- Neural networks for the purpose of localizing abnormal activity, can serve as a supplement for analysis

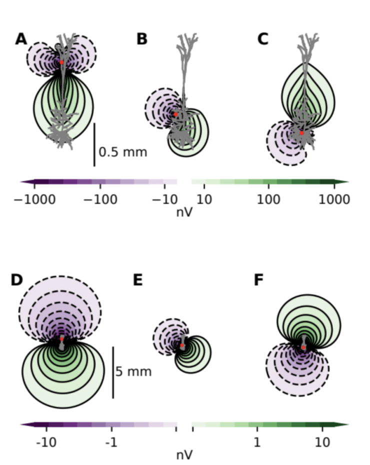
Simulating EEG Data

- Substantial amount of data is necessary to:
  - Capture complicated relationships
  - Make accurate predictions
  - Generalize well to unseen examples
- True EEG data for training a NN doesn't exist
- Solution: Simulate realistic data using LFPy
- New York Head Model
  - Volume conductor, computer model of the human head
  - Simulates electrical activity in the brain
  - Based on the anatomical and electrical characteristicsof MRI data
  - Provides detailed information on geometry and electrical properties
  - Generates predictions of EEG signals at 231 electrodes for 75K cortical locations
  - Utilizes the "Lead Field Matrix"



Current Dipole Approximation

- When simulating EEG signals, neural sources are treated as current dipoles
- Electrical potentials stemming from neural activity of a population of neurons tend to look like current dipoles
- By doing a multipole expansion the extracellular potential can be approximated by the dipole contribution alone

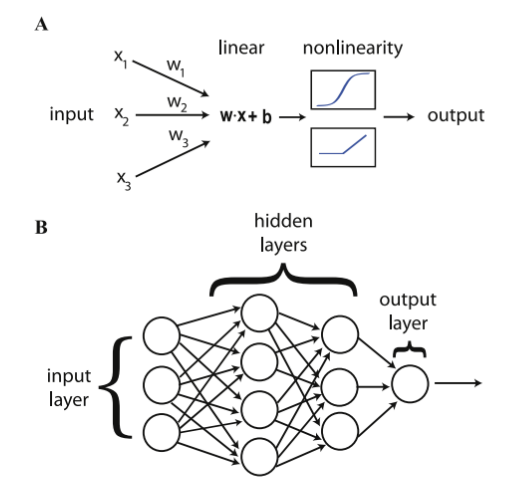


Code snippet

```python
def calculate_eeg(nyhead: NYHeadModel, A: int = 1.0):
    """
    Calculates the eeg signal from the dipole population

    returns:
        eeg_i : array of length (231)
            Combined eeg signal from the dipole population for a single patient
    """
    M = nyhead.get_transformation_matrix()

    # Dipole oriented in depth direction in the cortex
    p = np.array(([0.0], [0.0], [A])) * 1E7 # [nA* mu m]

    # Rotates the direction of the dipole moment so that it is normal to the cerebral cortex
    p = nyhead.rotate_dipole_to_surface_normal(p)

    # Generates the EEG signal that belongs to the dipole moment
    eeg_i = M @ p * 1E3 # [mV] -> muV unit conversion (eeg_i between 10 og 100)

    return eeg_i
```

The Feed Forward Neural Network

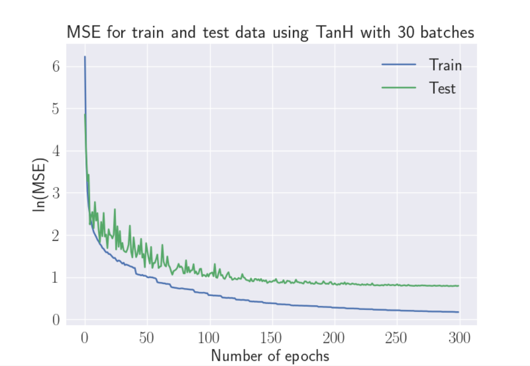
- "Learns from experience"
  - Processing and analyzing data
  - Uncover patterns linking input features to their corresponding output values
- FFNN: Information is only processed forward
  - Neurons
  - Linear transformation that weights the importance
  - Non-linear activation functions

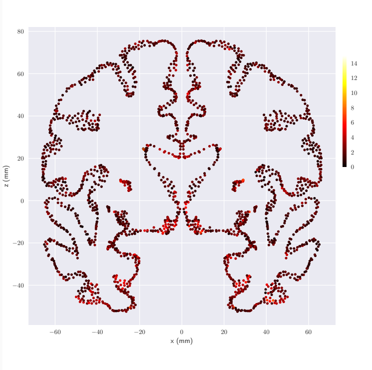


The FFNN

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class Net(nn.Module):
    def __init__(self, N_dipoles: int, determine_area: bool = False, determine_amplitude: bool = False):
        super().__init__()
        self.determine_area = determine_area
        self.determine_amplitude = determine_amplitude
        self.fc1 = nn.Linear(231, 180)
        self.fc2 = nn.Linear(180, 120)
        self.fc3 = nn.Linear(120, 84)
        self.fc4 = nn.Linear(84, 16)
        if determine_amplitude:
            self.fc5 = nn.Linear(16, 5*N_dipoles)
        elif determine_area:
            self.fc5 = nn.Linear(16, 4*N_dipoles)
        else:
            self.fc5 = nn.Linear(16, 3*N_dipoles)

    def forward(self, x: torch.Tensor):
        x = F.relu(self.fc1(x))
        x = torch.tanh(self.fc2(x))
        x = torch.tanh(self.fc3(x))
        x = torch.tanh(self.fc4(x))
        x = self.fc5(x)

        return x
```

Results (1) - Predicting coordinates



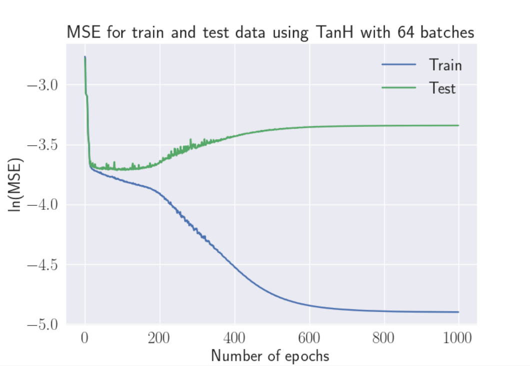


Results (2)

	MAE for different models		
Predictions	**Location**	**+ Amplitude**	**+ Radius**
x-coordinate [mm]	0.891	1.707	5.964
y-coordinate [mm]	0.903	1.923	6.295
z-coordinate [mm]	0.896	1.861	5.409
Amplitude [nA]	-	0.619	339.9
Radius [mm]	-	-	2.094

Results (3) - Predicting coordinates for multiple dipole populations

- Typical example of overfitting
- NN are overly specialized to the specific training data and fail to generalize effectively
- Solution: Change architecture of NN (?)



Summary

- Stronger fundation in managing, producing and analyzing large quantities of data
- Required proficiency in creative thinking and patience
- Further developed my programming skills
- Given the opportunity to developed and defeated bugs