Decomposition of Muscle Activity for Sensorimotor Neuroscience Proposal

Daniel King, Jasmine Ortega, Rada Rudyak, Rowan Sivanandam

10/05/2021

Contents

Executive Summary

Muscle movement is facilitated via the nervous system, which transports action potential (electrical signals) from the brain to muscular motor units. The motor units pass these electrical pulses through motor neurons attached to muscle fibers, causing the muscles to contract. The net electrical charge of a muscle can be measured via electromyography, or EMG. Typically, EMG signals are decomposed into several individual electrical signals that can be ascribed to singular muscle units.

In this proposal, we attempt to recreate and improve the blind-source separation algorithm created by Negro, et al (2016) that decomposes action potential spike trains from EMG signals. As is, the blind source algorithm parses EMG signals in a single pass and incorrectly identified action potential peaks are manually removed post hoc. Ideally, the algorithm would instead store the noise patterns as it analyzes, creating a feedback loop that integrates the learned noise patterns.

The final product proposed is a Python package with two main components. The first component is the reconstructed decomposition algorithm, optimized for the UBC Kinesiology Lab use case. The second component is a visualization element that allows users to view the decomposed output of the algorithm.

Introduction

Motor commands, or the coordinated actions between the brain and nervous system, enable the muscular movement of daily life. As the human body ages or suffers from neuromuscular disorders, the brain's motor commands degenerate, leading to difficulty in mobility and communication. Thus, it is imperative to study motor commands and the neural system that facilitates movement in our bodies. In this proposal, we aim to create a Python package that decomposes and visualizes the electrical signals sent from our brains to various muscles.

Electromyography (EMG) is used to quantify the net electrical charge of a muscle as it contracts. In non-invasive electromyography, surface electrodes are placed on the skin and voltage is measured as participants flex and relax a muscle. The collected signal is the sum of all the electric action potential fired from neurons attached to a muscle fiber. A single muscle fiber, along with its connected motor neurons, is known as a motor unit. Typically, researchers are interested in discovering when and which motor unit contributed to the net EMG signal.

Currently, the Sensorimotor Physiology Lab at UBC School of Kinesiology decomposes EMG signals using a free software from OT Bioelettronica, which parses the individual action potential spike trains using a closed-source algorithm. This type of decomposition is known as a convolutive blind source separation algorithm and is based off a paper published by Negro, et al (2016), which provides the mathematical logic and pseudocode used to build the algorithm. The algorithm employs high-level linear algebra and unsupervised learning techniques, such as Latent Component and K-Means classification.

The goal for this project is to recreate the blind-source separation algorithm described in Negro, et al (2016). Additionally, we hope to create a visualization tool that allows users to view the deconstructed EMG signals. To fit the needs of the Sensorimotor Physiology Lab, we will modify the program to deconstruct EMG recordings up to 5 minutes in length. Currently, the Bioelettronica software limits EMG decomposition to 100 second long recordings.

The final data product we will deliver is a Python package, emg-decomPy, that contains the decomposition algorithm and the visualization tool. We aim for emg-decomPy to be open-source and transparent, so future parties can customize or further develop functionalities. To accomplish this, there will be a large emphasis on documentation throughout the development of the package. While made for the Sensorimotor Physiology Lab, it is our hope that emg-decomPy will be a useful package to broadly explore the mechanisms underlying the brain's interactions with our muscles.

Data Science Techniques

There are two primary goals we identified within this project, and the techniques will be chosen based on those components.

1. Build custom software that inputs the raw signals provided by the surface electrode and outputs the decomposed data
2. Visualize the decomposed signals

We also have stretch goals contingent on completing the primary goals in time. First stretch goal is to provide GUI for manually marking false positives. This would include marking a false positive signal on the decomposed graph, and removing it from the visualization and from the raw data. At the same time, we will analyze whether the software can be optimized software based on user's specific needs. To do so, we will review whether anything within algorithm can be hardcoded, skipped, vectorizes, simplified or limited in a way that improves performances. If we still have time left in the project, we can begin to take steps towards the "live learning" upgrade. This would allow the user to mark false positive observations while the decomposition is running, allowing the algorithm to learn in real time.

Techniques and Tools in the original algorithm (Negro et al, 2016)

The primary tool described in Negro et al paper that our software till utilize is Convolutive / iterative blind source separation proprietary algorithm for decomposition optimization. This algorithm, in turn, is based on the general decomposition framework and validation technique presented in earlier works, including Holobar et al 2010, Marateb et al 2011a, and Hu et al 2014. Convolutive blind source separation technique is a combination of latent component analysis (LCA) and iterative fixed-point K-means classification technique. LCA extracts meaningful features out of a large set of EMG signal observations to identify motor unit activities, and K-Means performs classification into individual motor units.

Additional Techniques and Tools

Our project is primarily based on replicating the algorithm from Negro et al paper, so at this point we do not plan to attempt any other techniques for decomposition. However, we may explore possible alternative techniques if we get to the optimization stretch goal. One possible technique to consider is RNNs or bi-direction RNNs since we are dealing with time series type data. We will also check if there were any advances in machine learning techniques since the release of this software to see if any of the new tools that haven't been considered at the time of writing this paper and this algorithm, such as transformer algorithms, can be useful.

We will use a combination of tools for the development setup. We will use GitHub for collaborative version control of our package. We will develop Jupyter notebooks for small intermediate steps and as sandbox to try out contained modules. Once the algorithm has been built, we will use virtual machine by SageMaker lab. This will solve two problems. First, existing black-box software is Windows-only, and using SageMaker VM will allow us to test our algorithm against the original. Additionally, it will provide us with computational power to run longer sequences of muscle unit action potentials (MUAPs.) We will ultimately provide the solution in a python package emg-decomPy. The package will be set up with cookie cutter and utilize poetry for dependency management and configuration

Timeline

Dates in this timeline are tentative as we are still defining the scope of the project:

May 6, 2022: Present the proposal presentation to colleagues and project mentor **May 10, 2022**: Submit a draft written proposal to the project mentor **May 12, 2022**: Revise the report by incorporating feedback from project mentor (in some cases this may involve more than one round of revisions) **May 13, 2022**: After receiving permission from the project mentor, submit the finalized proposal to the capstone partner and project mentor **May 26, 2022**: Complete replicating the original algorithm and present it to the partner **June 1, 2022**: Present visualization component to the partner **June 5, 2022**: Make revisions to the algorithm and visualization based on last meeting **June 7, 2022**: Propose changes to improve and optimize the original algorithm **June 12, 2022**: Implement proposed changes and optimizations **June 14, 2022**: Revisions to the original algorithm **June 17, 2022**: Final Presentations **June 22, 2022**: Draft data product and draft final report **June 29, 2022**: Submit data product and final report

References