# Distributed Classification of EEG Signals for Automated Sleep Stage Scoring

Application in MapReduce running on Hadoop

TAYLOR D. SMITH, B.S., University of Kentucky
SATRIO HUSODO, M.S., University of Kentucky
XI CHEN, B.A., University of Kentucky

Analysis of electroencephalography (EEG) signals can provide invaluable information about a variety of systems/processes with counterpoints in the cortical brain. Clinicians and researchers can make various conclusions based on the features present in EEG recordings. In particular, it has been demonstrated that features such as the amplitude and frequency of oscillating signals correspond to activity in the brain and highly correlate with various factors including general information, such as the patient's cognitive state, as well as more specific and/or transient events such as seizures or performing certain tasks. Furthermore, EEG has been used to classify sleep into 4 stages, based primarily on the dominant frequency(s) present, and each of these stages is implicated in different activities that take place during sleep, such as memory consolidation or dreaming. While automated methods for sleep stage scoring exist, sleep experts are still frequently employed to perform this scoring. The aim of this project is to develop a distributed algorithm for classification of sleep stages from raw, unlabeled EEG in order to streamline this process for extremely large datasets. Ultimately, the goal is to provide a faster method to quickly extract the segments (corresponding to sleep stage) where analyses will be performed.

CCS Concepts: • **Information systems** → *Collaborative filtering*; *Clustering*; • **Computing methodologies** → *Classification and regression trees*; *Neural networks*; *Cross-validation*; • **Applied computing** → *Bioinformatics*;

Additional Key Words and Phrases: hadoop,MapReduce,distributed data,clustering,eeg,sleep stage

**ACM Reference format:**
Taylor D. Smith, B.S., Satrio Husodo, M.S., and Xi Chen, B.A.. 2017. Distributed Classification of EEG Signals for Automated Sleep Stage Scoring. 1, 1, Article 1 (December 2017), 5 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Several methods for monitoring brain activity exist, each having advantages and disadvantages. For example, functional magnetic resonance imaging (fMRI) is useful for visualizing neuronal activation in different regions within the brain or central nervous system as a whole. Similarly, both traditional MRI and computed tomography (CT) are particularly well suited for analyzing spatial structure of the brain and can be used in conjunction with fMRI to accurately determine which structures are activated given an arbitrary task or state of mind. These methods, however, have a number of drawbacks when more intricate analyses are required. Both the cost and operational logistics of


XXXX-XXXX/2017/12-ART1 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

these methods render them impractical for longitudinal analyses except where the desired time resolution is very low. As a more direct method of monitoring brain activity, the electrical field generated by neuronal activity may be measured.

Since the electrical potential of neurons changes based on their current activation state, neuronal activity results in the creation of local electric fields. By placing receiver electrodes on the human scalp, electroencephalography (EEG) may be used to monitor the consensus of cortical activity in terms of changes in voltage. One of the key advantages of EEG is typically very high time-resolution, allowing more useful analysis of changes over short periods of time (or subsequently the frequency domain) with little technical difficulty or additional time required. In particular, basal brain activity is predominantly oscillatory in nature, where the characteristics of this oscillatory behavior are correlated with conscious state of mind, as well as other factors. EEG is particularly informative during sleep, when artifacts due to motion and other confounding factors are less prevalent, however modern brain-computer interfaces (BCI) also utilize EEG for task recognition and interactivity.

The different sleep stages include the stages of non-rapid eye movement (NREM) sleep and REM and are predominantly defined by the dominant frequency band in the EEG. Identification of the sleep stage and its progression over time can be very informative itself; however, there are certain features/tasks that are associated with the various sleep stages. A common example would be the presence of 'sleep spindles', predominantly in Stage 2 NREM sleep. A sleep spindle is characterized in the EEG as a burst of oscillatory activity in the range 9-16 Hz with a recognizable waxing-waning shape lasting greater than 0.3 seconds. While a discussion of the characteristics of sleep spindles and their implications are beyond the scope of this paper, it suffices to say that selection of specific sleep stages from recorded data can facilitate further analysis.

While tools for automated analysis of EEG are readily available, and have been for some time, the complexity of the information present in EEG signals, along with frequently low signal-to-noise ratios, the data is largely manually curated by domain experts. There do exist automated methods to complete various tasks when analyzing EEG data; however, most are developed for detection/analysis of discrete events (i.e. picking up a pencil) or for extracting more general features for statistical comparison, such as average amplitude, dominant frequency, or power analyses. Therefore, implementing these methods along with machine learning algorithms in a parallel fashion is clearly both feasible and valuable.

The goal of this project is to implement a distributed machine learning algorithm for automated sleep stage scoring from raw EEG in order to facilitate quick sleep stage extraction for further investigation.

## 2 DATA

The dataset used for training/validation of the model is the Massachusetts Archive of Sleep Studies (MASS) sleep spindle dataset. This set is composed of 19 full night EEG recordings with a sample rate of 256 Hz. Each is a whole-night recording taken from a different individual. In addition to expert annotated sleep spindles, a hypnogram containing the sleep stage in 30 second epochs is provided. The sleep stage scoring was performed on the central Cz channel, however 20 channels (19 primary, 1 reference) are provided for each.

## 3 METHODS

### 3.1 Generation of Training Data

Figure 1 shows an overview of the preprocessing steps. For each of the 19 excerpts, hamming windows were used to segment the raw signal from all channels into 30 second segments with an

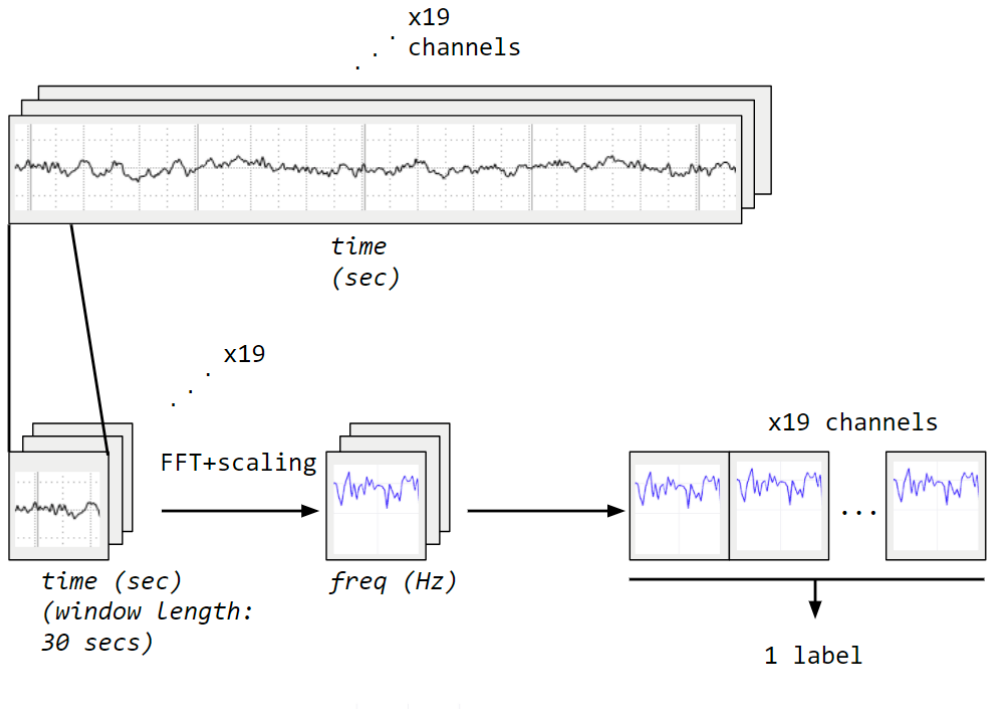


Fig. 1. Preprocessing steps showing how a patient's EEG signal files are decomposed into individual windows, transformed via FFT, and assigned with the label.

overlap rate of 50%. The purpose of the hamming window was to reduce artifacts introduced by abrupt signal changes (such as the beginning or end of a window). Each window was then labeled the appropriate sleep stage, as indicated by the provided hypnograms. The window duration was chosen to align with the hypnogram resolution, however due to the 50% overlap, windows that are in-between different stages will be excluded from this analysis.

### 3.2 Feature Extraction

The feature extraction methods most commonly employed for analysis of EEG typically involve frequency decomposition of the signal. Briefly, a short-time fast-fourier transform (STFFT) was applied to each 30 second training example on each channel separately and the power spectral density (PSD) was used to construct a 2 dimensional frequency-channel domain image. Each channel is then separately normalized to the range [0..1] across all subjects/frequencies. Note that during normalization, the data is first clipped to the range [mean - 3*st.dev., mean + 3*st.dev.] to avoid outliers dominating the rest of the signals.

The resulting feature matrices contain a large amount of redundant information. Following normalization, the entropy for each feature (channel-frequency pair) is calculated across all subjects as a means of identifying features with little significance in terms of differentiating the input data. At this point, the features with the highest entropy are selected and the resulting matrices flattened, resulting in a 1-D feature vector for each window.

Currently, much of the preprocessing code is already implemented and available, however none of it is in a parallelized form. So, we plan to work on implementing a parallelized/MapReduce-based

version of the preprocessing steps so that we can handle large amounts of data. The classification and prediction models outlined below will also be implemented as such.

### 3.3 Classifiers

Once the feature vectors have been generated and feature selection has completed, an established classification model will first be trained on a large portion of the data, before then being evaluated on the remaining data. There are three main classes of supervised learning models we plan on testing/implementing.

*3.3.1 Support Vector Machine (SVM).* Given the training set, we can build a linear or non-linear classifier using SVM. An input sample consists of EEG features paired with the annotated sleep stage. SVM will find a subset of the most important features by maximizing the hyperplane that separates the output classifications for the training dataset. We plan to use Apache Spark's SVM implementation.

*3.3.2 Random forest.* The random forest method builds a classifier consisting of multiple decision trees. The use of subsets of the training data (bootstrapping) and random selection of features will result in the randomization of the decision trees. Classifying the test dataset involves aggregating the results of the random trees by consensus voting or averaging.

*3.3.3 Neural Network.* Neural network (NN) has been showing promises in handling high dimension data for classification. After the FFT and data concatenation, the output data dimensions will increase dramatically, which is a typical problem of the curse of dimensionality. In the essence, the inputs for neural network will be all the features that are from the output of the concatenation after FFT as shown in Figure 1. Within the NN, we plan to construct 3-7 layers with pooling between layers, which will decrease the feature dimension. After training, the weights within the NN will act as secondary, more abstract feature extractors without having to engineer the feature extractors by hand. If an external package is used for NN implementation, we will most likely use either TensorFlow or Apache MXNet, however we may implement a simple multi-layer perceptron (MLP) instead.

## 4 EVALUATION PLANS

We plan to use k-cross validation to train and evaluate the classifiers. With the 8 patient samples, we will iteratively leave one patient out, train on the rest of the patients, and measure the classifier accuracy. The overall accuracy will be the average of the accuracies from the different iterations. We will also generate confusion matrices and or ROC-like curves to visualize the relationship between true positive and true negative and the overall distribution of correct vs. incorrect predictions.

## 5 ALTERNATIVE STRATEGIES

Particularly due to the fact that our applications will be written in Python and therefore it is not unreasonable to imagine a scenario in which it is run on a system that does not contain additional Python libraries utilized by our commands. While this could technically be resolved by passing the source for the necessary libraries to the systems running the mapper and reducer programs during initialization, direct implementation of the required methods would undoubtedly be much more efficient. Since numpy and scipy are typically included with any installation of Python, these are presumed to be available, however this is one of the primary motivations for developing an MLP rather than utilizing TensorFlow or Apache MXNet.

## 6 TIMELINE

| Week | Dates | Task |
|---|---|---|
| 1 | 10/30-11/5 | Data preprocessing; study data; learn to use tools |
| 2 | 11/6-11/12 | Start implementing classifiers and NNs; evaluate data and identify challenges |
| 3 | 11/13-11/19 | Continue implementations; tackle challenges; evaluate running results |
| 4 | 11/20-11/26 | Continue implementations; evaluate results; compare methods; start writing |
| 5 | 11/27- | Finish project; write report; make presentation |

## 7 TEAM MEMBER ASSIGNMENTS

The group plans to work together to parallelize the preprocessing steps. Xi Chen and Satrio Husodo will first work together on understanding the non-parallelized programs that have been implemented by Taylor Smith and also understand the intricacies of the dataset prior to working on implementing the distributed algorithms. We also have a subset of features that we can work on for the classification methods. For the classifications, each person will implement one of the algorithms by using existing frameworks. Taylor Smith has the raw datasets and has implemented the preprocessing steps in sequential form. Taylor will work on parallelizing the preprocessing steps and implement MLP. Satrio Husodo will work on parallelizing the preprocessing and implement SVM. Xi Chen will work on parallelizing the preprocessing steps and implement Random Forest. All group members will contribute to the planning, writing, and presentation of the project.