Cloud-Based Analysis of EEG Signals for BCI Applications

Kathleen Ericson[1], Shrideep Pallickara[1], Charles W. Anderson[1]
[1]*Colorado State University, Colorado, CO, USA*

Correspondence: Kathleen Ericson, Colorado State University, 1873 Campus Delivery, Fort Collins, CO 80523, USA.
E-mail: ericson@cs.colostate.edu

Abstract. In this work we explore the feasibility of using a cloud-based approach to analyzing EEG signals for use in BCI. A cloud-based approach allows us to move analysis away from the user, offering a three-fold benefit. First, it means we can perform more complex analyses as the user transmits data. We are also able to develop algorithms which utilize data from multiple users. This also allows us to leverage demanding analysis algorithms from mobile devices such as smart phones/tablets with limited resources. For this approach to be feasible, we need to ensure that data is processed quickly and reliably, handling any lags that may be incurred due to network communications.

Keywords: EEG, Stream Processing, Motor Imagery, Mental Tasks, Collaborative Classifications, Neural Network

1 Introduction

As EEG collection devices become smaller and more mobile, the need to perform EEG analysis on small-factor devices such as mobile phones or tablets grows. These devices have limited processing power, but are equipped with networking capabilities that allow them to transmit data to a remote resource pool for advanced processing. One approach is to dedicate a single machine to each user wishing to remotely connect for classification, but this leads to wasted resources and does not scale well. We propose interleaving computations as an effective and scalable method of supporting many concurrent users within a pool of limited resources.

2 Approach

In this work we use a feedforward artificial neural network (ANN) with time-delay embedding for the mental task BCI paradigm. We are building off the work introduced in [Anderson and Bratman, 2008; Anderson et al., 2011; Ericson et al., 2010], using similar neural networks and a time-delay of 3 timesteps. Our focus is not on accuracy, instead we mark all classifications as passing or failed based on our ability to classify data in a timely manner.

As we are sending data to be classified in a remote resource pool, we have further compressed signals by sending all data generated for the last 250ms to be classified together. We consider responses which take more than 250ms as failed. We use this measure across all our experiments. This time period was chosen as it is a good fit for our processing footprint, but is fully adjustable. We can classify signals of this length in 10s of milliseconds, leaving plenty of time left over for any communications overheads we may incur from our approach.

2.1 Data

We classify EEG signals generated by an able-bodied adult male, gathered by the CSU BCI lab (`http://www.cs.colostate.edu/eeg`). Four tasks are recorded from the user: imagined right hand movement, imagined left leg movement, counting backwards from 100 by threes, and imagining a computer display tumbling in three dimensions. The dataset is separated into five different recording sessions, where each recording session has 10 five-second recordings for each of the four tasks. We use four of our five datasets for training, reserving the fifth for testing. The EEG data was gathered using a NeuroPulse Mindset 24R amplifier with 19 electrodes arranged using the international 10-20 specifications and a sampling rate of 512 Hz.

2.2 Network Setup

In our experiments we used identical machines to support all processing of EEG signals and to stream pre-recorded EEG signals into the resource pool for classifications. This resource pool consists of machines with four 2.4GHz cores, 16GB RAM, and gigabit ethernet connection, with round-robin load balancing. For classification, we are using a group-of-experts approach, where multiple ANNs are trained with slightly different initial weights, allowing each to learn a slightly different solution. Each user has a dedicated group of experts trained on their own data, which users can tune as needed.

Each machine in the resource pool also has a generic group of experts. This group of experts has access to new training data from *all* users currently utilizing that machine. This allows the system to take advantage of hosting computations for multiple users – it can learn patterns across many users simultaneously, which can lead to increases in performance [Wang and Jung, 2011].

3 Results

We first focused on determining the maximum number of users we could support on a single machine. This sets an upper limit for all further stress tests. As we add more machines and users we put more strain on the communications infrastructure, so we will likely only be able to support fewer users. In our first attempt, we tried to support 30 concurrent users on a single machine. This means that there are effectively 31 unique computations on a machine (one generic, and one dedicated to each user). With 30 users, we only had one message that returned in over 250ms out of 150,000 messages sent, leading to a failure rate of 0.0006%. As the one failed message proved to be one of the first messages sent, we believe that this was purely an initialization overhead. We then tried to support 35 concurrent users on a single machine, which had a failure rate of only 0.2%. When we moved up to 40 concurrent users the failure rate jumped up to 0.5%, but the magnitude of failure went from a 9.5 second delay to 30 seconds. To minimize the magnitude of failures, we decided to cap further tests to 35 concurrent users.

Using the information found in our initial stress tests, we scaled up to a resource pool of 40 machines. We hope to support just as many distinct computations in this setup as we found in the first set of experiments (35 users/machine). As we expect to overload our communications substrate, we started with 25 users per machine, a total of 1000 computations. With 1000 concurrent users, we saw a failure rate of 0.005%, with a worst-case response time of just over one second. We next moved on to support 35 users per machine (1400 computations). When supporting 1400 users, we saw a massive jump in the number of failed computations, up to 0.8%. We also saw a drastic change in the probability density of response times. Looking at these response times (Fig 1), it is clear that attempting to support more users would lead to more failed computations.

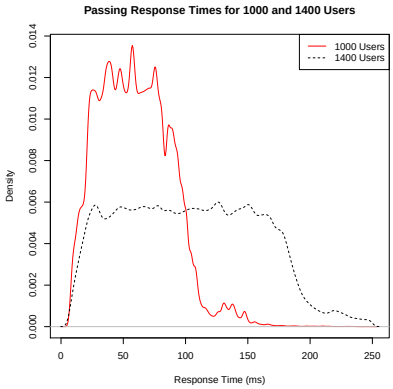

Figure 1: *Density functions of passing response times in milliseconds for 1000 and 1400 users on a cluster of 40 nodes*

4 Discussion

While we have been working with neural networks for the mental task BCI paradigm, there is nothing to preclude the use of another algorithm, or even a different paradigm within our framework. We can support arbitrary processing in a number of programming languages, such as C, C++, C#, Java, Python, and R [Ericson and Pallickara, 2012]. Different algorithms will have slightly different processing requirements and different paradigms will have different communications overheads, but by modifying the rate at which messages are passed to the network for processing we can account for any of these changes.

While we have explored methods of supporting multiple users concurrently, there are cases where it may not be feasible for users to remain constantly connected to a cluster, such as when there are in places with poor wireless connections, or when the user is on a data-limited connection, such as a 3G or 4G network. In these situations, it makes sense to use an offline model such as we see adopted into some mobile voice recognition applications. The phone is regularly updated with new models which have been developed after sampling across multiple users. These are small-form models, which allow quick classifications on the mobile device itself, so users do not require a constant live connection to outside servers. Users would still, however, be able to access models trained with data from multiple users, potentially leading to much better results.

References

Anderson, C., Forney, E., Hains, D., and Natarajan, A. (2011). Reliable identification of mental tasks using time-embedded eeg and sequential evidence accumulation. *Journal of Neural Engineering*, 8(2):025023.

Anderson, C. W. and Bratman, J. A. (2008). Translating thoughts into actions by finding patterns in brainwaves. In *Fourteenth Yale Workshop on Adaptive and Learning Systems*, pages 1–6.

Ericson, K. and Pallickara, S. (2012). Adaptive heterogeneous language support within a cloud runtime. *Future Generation Computer Systems*, 28(1):128–135.

Ericson, K., Pallickara, S., and Anderson, C. W. (2010). Analyzing electroencephalograms using cloud computing techniques. In *IEEE Conference on Cloud Computing Technology and Science*.

Wang, Y. and Jung, T.-P. (2011). A collaborative brain-computer interface for improving human performance. *PloS one*, 6(5):e20422.