Decoding Brainwave Data using Regression

Justin Kilmarx: The University of Tennessee, Knoxville
David Saffo: Loyola University Chicago
Lucien Ng: The Chinese University of Hong Kong
Mentor: Dr. Xiaopeng Zhao
Introduction

Brain-Computer Interface (BCI)

Applications
  - Manipulation of external devices (e.g. wheelchairs)
  - For communication in disabled people
  - Rehabilitation robotics
  - Diagnosis and prediction of diseases (e.g. Parkinson’s disease, Seizure, Epilepsy)

Games

Invasive vs Noninvasive
  - Electrocorticography
    - Fifer et al. (2012)
  - Electroencephalography
    - Mcfarland & Wolpaw (2011)
Background

Invasive

Noninvasive
Sensorimotor Rhythms (SMR)
Steady-State Visual Evoked Potential (SSVEP)
Imagined Body Kinematics
  Continuous decoding the kinematic parameters during imaginary movements of one body part
  Short time of training
  Natural imaginary movement
  Smoother controller system
  Possibility of developing a generalized decoder
  Eliminating Subject dependency
Research Objective and Setup

Objective: The goal of this project was to improve the prediction accuracy for a previously developed BCI model that used linear regression to predict cursor velocity from a subject's thoughts by testing new methods and nonlinear models.

Setup
- Emotiv EPOC for recording EEG signals
- BCI2000 for cursor visualization and data collection
- Matlab/Python for processing
Training

Automated cursor movement on computer monitor in 1D
Subject imagines following movement with dominant hand
10 trials
  5 horizontal
  5 vertical
1 minute each
Cross validation between trials
33 Subjects
Filtering

Raw EEG signals contain a lot of noise
4th order Butterworth lowpass filter with cutoff at 1 Hz
Attempted using bandpass over Mu, Alpha, and Beta bands, but these did not contain useful information for imagined body kinematics
Regression

Predict cursor velocity from EEG data
12 previous points in memory as features
Trial wise cross validation
Average prediction accuracy using goodness of fit on linear regression model
  Horizontal: 70.77%
  Vertical: 44.67%
Results
Results

Other models did not show significant improvements and were more computationally expensive (adaboost regression, ridge regression, kernel ridge regression, support vector regression, and multilayer perceptron)
Channel Importance

Channel-wise identification
Horizontal (top)
  F7 and F8
  Right hemisphere
Vertical (bottom)
  AF3 and AF4
  F3 and F4
Clear pattern between horizontal and vertical
Right hemisphere controls left body
## Results

<table>
<thead>
<tr>
<th>Channels</th>
<th>Horizontal Accuracy</th>
<th>Vertical Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Channels</td>
<td>70.77%</td>
<td>44.67%</td>
</tr>
<tr>
<td>F7, O2, P8, T8, FC6, F4, F8, AF4</td>
<td>71.03%</td>
<td>41.68%</td>
</tr>
<tr>
<td>F7 and F8</td>
<td>69.93%</td>
<td>25.64%</td>
</tr>
<tr>
<td>F7, FC5, T8, FC6, F4, F8</td>
<td>72.73%</td>
<td>36.98%</td>
</tr>
<tr>
<td>AF3 AND AF4</td>
<td>41.93%</td>
<td>30.29%</td>
</tr>
<tr>
<td>AF3, F3, F4, and AF4</td>
<td>49.21%</td>
<td>33.09%</td>
</tr>
<tr>
<td>AF3, F3, F7, F8, F4, and AF4</td>
<td>69.97%</td>
<td>41.61%</td>
</tr>
</tbody>
</table>
Classification

Horizontal vs. Vertical
FFT analysis across 14 channels in 1 second samples
224 total features from 4 bands: Theta (4-7 Hz), Alpha (8-15 Hz), Beta (16-32 Hz), and Gamma (32-40 Hz)
  Mean PSD, median PSD, min PSD, max PSD
Model trained with Random Forest
## Results

<table>
<thead>
<tr>
<th>Channels/Features</th>
<th>Average Accuracy Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Channels/All Features</td>
<td>79%</td>
</tr>
<tr>
<td>All Channels/Means</td>
<td>80%</td>
</tr>
<tr>
<td>Six Channels/All Features</td>
<td>68%</td>
</tr>
<tr>
<td>Six Channels/Means</td>
<td>69%</td>
</tr>
</tbody>
</table>
Serial vs. Distributed

Results were generated using distributed computing

Dask Distributed Library

Cluster setup on Comet at the San Diego Supercomputer Center

<table>
<thead>
<tr>
<th></th>
<th>Adaboost</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial</td>
<td>Adaboost</td>
<td>04:30:00</td>
</tr>
<tr>
<td>Distributed</td>
<td>Adaboost</td>
<td>00:4:29</td>
</tr>
</tbody>
</table>
EEG-Based Control of a Computer Cursor with Machine Learning

Lucien Ng (The Chinese University of Hong Kong),
Justin Kilmarx (University of Tennessee),
David Saffo (Loyola University Chicago)
EEG-Based Cursor Movement Classification

In the sense of machine learning, this is a supervised multiclass classification. The specification as follow:

- **Input**
  - EEG data (time series) with 128 Hz and 14 channels

- **Prediction**
  - Vertical: Left, Right, No Movement
  - Horizontal: Up, Down, No Movement

Direction of movement at any given time point
Objective

• To classify the user indenting cursor movement by using EEG signal with high accuracy, and
• To accelerate the process to acceptable speed
Overview of Models

Raw Data: Current and Past Sample

Preprocessing
- Filter Bank
- Event-related Potential
- Neural Oscillation

Basic Models
- Logistic Regression
- Neural Network

Ensemble Models
- Gradient Boosting

Prediction
- Up/Right
- Down/Left
- No movement
Workflow

Generate massive models setting

Train and test each models on Bridges (16 Cores + GPU)

cross-validation

Observe & Refine

Pick the best combination of models to do run-time prediction
Feature Extraction: Filter Bank

Left low pass filter: Only past time points were used to train and test the model for any given time points.

Applied Low pass frequency: 0.5 Hz, 1 Hz, 2 Hz, 3 Hz, 4 Hz, 5 Hz, 6 Hz, 7 Hz, 9 Hz, 15 Hz, 30 Hz

<table>
<thead>
<tr>
<th>Psychological or Physiological State</th>
<th>Changes in EEG Waves</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep sleep</td>
<td>Predominance of the delta wave</td>
</tr>
<tr>
<td>Concentrated</td>
<td>Suppression of the alpha wave</td>
</tr>
<tr>
<td>Vigilant</td>
<td>Generation of beta wave</td>
</tr>
<tr>
<td>Recognition of sensory stimuli</td>
<td>Changes in gamma wave</td>
</tr>
</tbody>
</table>

Classifying: Multilayer perceptron

A Brief History of Neural Networks. Available: https://www.dtreg.com/solution/view/21
Classifying: Recurrent Neural Network

Models Ensembling: Gradient Boosting

Residual fitting

Experimental Setup

- 12 Subjects’ data were used
- Each of them has 5 trials about horizontal / vertical movements
- Validation:
  - Cross-validation: Reordering the trials, we got 10 different combinations
- The experiment ran on XSEDE-bridges with 16 CPU-cores and GPU (P100)

<table>
<thead>
<tr>
<th>Trials</th>
<th>1st, 2nd and 3rd</th>
<th>4th</th>
<th>5th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Models</td>
<td>Training Data</td>
<td>Validation</td>
<td>Validation</td>
</tr>
<tr>
<td>Ensemble Models</td>
<td>-</td>
<td>2-fold Valid</td>
<td>2-fold Valid</td>
</tr>
</tbody>
</table>
Multithread

All the train-validation sets can run independently.
All the event classifiers can be trained independently
Lets utilize all the CPU-cores!
Visualized results: An Example of Prediction on Horizontal Movement
Results: Horizontal

The best basic model:
- Preprocessed by filter bank
- Neural Network with 32 hidden units

The Accuracy/AUC of subjects

---

AUC

Accuracy
## Results

<table>
<thead>
<tr>
<th>Prediction</th>
<th>AUC</th>
<th>Accuracy</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal</td>
<td>0.91</td>
<td>80%</td>
<td>10.5 hours</td>
</tr>
<tr>
<td>Vertical</td>
<td>0.71</td>
<td>60%</td>
<td>10.5 hours</td>
</tr>
</tbody>
</table>

Time for a training process = 10 minutes
Acceleration: Magma-DNN

MAGMA-DNN: Toward a More Flexible DNN Framework for Low-Level Implementation

Magma is a large, well-supported software package designed for computations in algebra, number theory, algebraic geometry and algebraic combinatorics.

The main operation in neural network is matrix multiplication.

Let's try to use Magma to build a neural network!
Advantage

Open-source

Flexibility: Free to implement any mathematical function for both CPU and GPU with Magma

Fast
Benchmark: MNIST dataset

Number of input size: 28 x 28 = 784

Number of Hidden units: 100

Batch size: 100

Number of iteration: 1000

Precision: Float (32 bits)

GPU: GeForce GTX 1050Ti

![DNN-Framework Speed Comparison Diagram]
Comparison with other DNN frameworks

<table>
<thead>
<tr>
<th></th>
<th>MAGMA-DNN</th>
<th>Caffe</th>
<th>TensorFlow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>Fast</td>
<td>Fast</td>
<td>Relatively Slow</td>
</tr>
<tr>
<td>Input Data Format</td>
<td>Support Native Pointer Array</td>
<td>HDF5 Only</td>
<td>NumPy</td>
</tr>
<tr>
<td>Dependency</td>
<td>MAGMA</td>
<td>Protobuf, HDF5, CUDA, BLAS, OpenCV, Boost…</td>
<td>CUDA, NumPy …</td>
</tr>
</tbody>
</table>
Architecture

Layers

- InputTensor
- OutputTensor
- Forward()
- Backward()
- Update()
- Derivative()

Layers

- InputTensor
- OutputTensor
- Forward()
- Backward()
- Update()
- Derivative()

Layers

- InputTensor
- OutputTensor
- Forward()
- Backward()
- Update()
- Derivative()
Code Example

Layers Initialization

```cpp
InputLayer<float> inputLayer(inputMat);
FCLayer<float> FC1(&inputLayer, n_hidden_units);
ActivationLayer<float> actv1(&FC1, SIGMOID);
FCLayer<float> FC2(&actv1, n_output_classes);
OutputLayer<float> outputLayer(&FC2, labelsMat, BIN_CROSSENTROPY_WITH_SIGMOID);
```

Network construction

```cpp
std::vector<Layer<float>> vec_layer;
vec_layer.push_back(&outputLayer);
vec_layer.push_back(&FC1);
vec_layer.push_back(&actv1);
vec_layer.push_back(&FC2);
vec_layer.push_back(&outputLayer);
```

Training

```cpp
for (int i = 0; i < (int) vec_layer.size(); i++) vec_layer[i]->forward_gpu();
for (int i = vec_layer.size() - 1; i >= 0; i--) {
    vec_layer[i]->update();
    if (i >= 2) vec_layer[i]->backward_gpu(); // fc1 doesn't need to backward
}
Future Works

Explore the EEG data by apply more machine learning techniques on it
Implement Convolutional Neural Network and Recurrent Neural Network on MAGMA
Apply MAGMA-DNN on the EEG data analysis