Erkennung ereigniskorellierter Potenziale eines Elektroenzephalogramms durch eine KI

Alexander Reimer Matteo Friedrich

4. Dezember 2021

Inhaltsverzeichnis

1 Zusammenfassung

In diesem Projekt wollen wir einen Roboter mithilfe bloßer Gedankenkraft steuern.

Um Daten über das Gehirn zu bekommen, nutzen wir einen Elektroenzephalographen, kurz EEG, welches durch Elektroden an der Kopfhaut die Spannungen innerhalb des Gehirns misst. Dies werten wir mithilfe eines Neuronalen Netzes aus, welches wir vorher darauf trainiert haben, Muster in diesen Daten zu erkennen. So können wir bestimmte Ereignisse anhand der EEG-Daten ableiten, z.B. ob jemand geblinzelt hat oder sich gerade konzentriert.

Das Ziel ist es dann, durch das Erkennen verschiedener dieser sogenannten ereigniskorrelierten Potentialen (EKPs) einen Roboter nur mit Gedanken steuern zu können.

2 Einleitung

In diesem Projekt wollen wir einen Roboter mithilfe bloßer Gedankenkraft steuern.

3 Methode und Vorgehensweise

3.1 Materialien

- EEG
  - 4 Channel Ganglion Board von OpenBCI ⋆
  - 2x Spike Electrodes ⋆
  - 2x Flat Electrodes ⋆
  - Electrode Gel
- Roboter
  - Lego Mindstorms EV3 Brick
  - 2x EV3 großer Motor
  - Raspberry Pi 4B 8GB
  - Diverse Legoteile
- Software
  - Flux.jl für das neuronale Netz
  - BrainFlow.jl als Schnittstelle zum EEG
  - FFTW.jl für die Fast Fourier Transformation
  - CUDA.jl zum effektiven Nutzen einer NVIDIA GPU
  - PyPlot.jl zum plotten
  - BSON.jl zum Speichern und Laden von Netzwerken

3.2 Vorgehensweise

Die Idee eines BCI – Brain-Computer Interface – ist nicht neu, sondern wird intensiv erforscht. Aufgrund bereits durchgeführter Experimente wissen wir, dass es möglich ist. Besonders informativ bei unserer Recherche war der Artikel „Brain-computer interfaces for communication and rehabilitation“ [1].

Wir hoffen, neben dem Erlangen von Erfahrung in diesem interessanten Bereich auch selbst dazu beizutragen. Dies wollen wir erreichen durch *a)* eine allgemeine Anwendung, bei der kein vorheriges Trainieren für eine fremde Person benötigt wird, *b)* das Verwenden von günstiger, für viele bezahlbare Hardware, sowie *c)* ein performantes Programm welches *d)* leicht für die eigenen Zwecke anpassbar ist.

Unser Projekt lässt sich grob in 3 Teile unterscheiden.

Zum einem gibt es den neurobiologischen Teil. Dieser besteht aus der Messung von Gehirnaktivität und der Umwandlung dieser Aktivität in für uns nutzbare Daten.

Der zweite Teil besteht aus der Verarbeitung dieser Signale. Hierfür nutzen wir ein neuronales Netz, welches Muster in den Gehirnaktivitäten erkennen kann.

Der letzte Teil von unserem Projekt beinhaltet die Konstruktion und Steuerung eines EV3 Roboters. Je nachdem, was das Neuronale Netz ausgibt, soll sich dieser Roboter anders verhalten und so über Gedanken steuerbar sein.

3.2.1 Elektroenzephalographie

Bei der Elektroenzephalographie werden Elektroden an der Kopfoberfläche platziert. Diese können die sehr kleinen Spannungen messen, die durch Reize im Gehirn entstehen und durch den Schädel dringen. Diese Spannungen werden nicht von einzelnen Nervenzellen erzeugt, sondern geben die Summe aller lokalen Spannungen wieder. Man kann also auch nur ungefähr sagen, wo genau im Gehirn ein bestimmter Reiz ausgelöst wurde, je mehr Elektroden desto höher die Genauigkeit.

Wir untersuchen dabei Ereigniskorrelierte Potentiale (EKPs). Dies sind bestimmte Spannungsschwankungen („Potentiale“), welche in Zusammenhang mit einem (beobachtbaren) Ereignis stehen, wie z.B. Blinzeln oder Armbewegungen.

Weiter ist es möglich, nur durch Gedanken eine Steuerung auszuführen. Dies funktioniert jedoch in den von uns gefundenen Beispielen meist durch Instrumentelle oder Klassische Konditionierung, also durch das Bestrafen und Belohnen auf Basis der Messungen des EEG. So kann das Gehirn darauf trainiert werden, auf Verlangen bestimmte, vorher festgelegte Aktivität auszulösen, die dann gemessen und ausgewertet werden kann.

Wir wollen dies nicht machen, da die Konditionierung Zeit benötigt, nicht unser Ziel einer allgemeinen Anwendbarkeit erfüllt, und voraussichtlich bessere Ausrüstung erfordert als wir haben.

Jedoch wollen wir in Zukunft versuchen, mithilfe unseres Neuronalen Netzes bereits bei allen Menschen vorhandene, selbstkontrollierbare Gehirnaktivität zu finden und sicher zu erkennen. Solche Gehirnaktivität könnte z.B. der allgemeine Gedanke an “Rechts” sein, was womöglich mit erhöhter Aktivität auf der linken Hirnhälfte in Verbindung stehen könnte.

Zuerst probieren wir es aber mit EKPs, da diese deutlich leichter zu erkennen sind und wir so erstmal das Konzept eines Neuronalen Netzes in einem solchen Kontext ausprobieren können. Denn wenn wir EKPs noch nicht sicher erkennen könnten, wären komplexere Zusammenhänge erst recht nicht möglich.

(Hier eine Abbildung von einem EEG mit Blinzeln)

Unser EEG-Gerät, das Ganglion Board, hat vier Elektroden, mit einer zeitlichen Auflösung von jeweils 200 Herz (200 Messungen pro Sekunde). Diese Elektroden haben wir am Okzipitallappen platziert. Grund dafür ist, dass wir mit unserem Gerät primär Alphawellen (Wellen im Bereich zwischen 7 und 13 Herz) gut erkennen können und dieser dort besonders ausgeprägt sind.

Unser EEG-Gerät besteht aus einem Klettband mit Löchern für die Elektroden, einer Platine, in welche die Elektroden eingesteckt werden, und einem USB-Dongle, mit welchem die Signale der Platine kabellos empfangen werden können.

Um die Signale in Julia empfangen, in Dateien speichern, und laden zu können, haben wir BrainFlow benutzt.

3.2.2 Neuronales Netz

Was ist ein neuronales Netzwerk?

Zuerst wollten wir die Grundstruktur eines neuronalen Netzwerkes programmieren. Ein neuronales Netzwerk besteht aus drei Teilen: dem Input Layer, den Hidden Layers und dem Output Layer.

Der Input Layer ist nur eine Liste aus Zahlen zwischen 0 und 1. Er gibt an, welche Eingaben (Inputs) das Netzwerk bekommen soll, z. B. die Grauwerte der Pixel eines Bildes.

Die Hidden Layers sind eine Ansammlung von in mehrere Layer (Schichten) unterteilten Neuronen. Jedes Neuron besitzt eine Aktivierung (Activation), die als Zahl zwischen 0 und 1 angegeben werden kann, und einen Bias (Verzerrung), der eine beliebige Zahl sein kann. Die Neuronen verschiedener Layer sind alle durch sogenannte Gewichte (Weights) verbunden, die ebenfalls einen beliebigen Wert haben können.

Forward Pass

Um nun die Aktivierungen eines Neurons zu berechnen, gibt es den sogenannten Forward Pass. Dabei beginnt man im ersten Hidden Layer damit, für alle Neuronen den sogenannten Netz Input (auch net input) zu berechnen. Um den Netz Input eines Neurons zu berechnen, werden alle Aktivierungen des vorherigen Layers mit den von dem Neuron dorthin führenden Gewichten multipliziert und summiert. Der Bias ist eigentlich auch ein Gewicht, jedoch ist er mit einem Neuron verbunden, das immer die Aktivierung (a_i) 1 hat, weshalb man einfach $+b_i$ statt $+b_i * a_i$ rechnen kann.

Um aus diesem Netz Input nun die Aktivierung zu berechnen, benötigt man eine Aktivierungsfunktion, die dafür sorgt, dass die Aktivierung zwischen 0 und 1 liegt. Wir haben dafür eine Sigmoidfunktion benutzt, die eine Zahl nimmt und einen Wert zwischen 0 und 1 ausgibt.

$$sig(x) = \frac{1}{1 + e^{-x}}$$

$$a_i = sig\left(netzinput_i\right)$$

wobei a_i = die Aktivierung des Neurons i und $netzinput_i$ = der Netzinput des Neurons i.

Dies wird dann für jedes Neuron in jedem Layer wiederholt. Da man aber immer die Aktivierungen des vorherigen Layers benötigt, muss man das ganze vom Input Layer zum Output Layer machen. Daher auch der Name Forward Pass. Die Outputs sind dann lediglich die Aktivierungen der Neuronen im Output Layer. Die Interpretation dieser hängt von den Trainingsdaten ab (s. Backpropagation - Theorie). Die allgemeine Formel für die Aktivierung eines Neurons lautet also:

$$a_j = sig\left(\sum_L (a_L * W_{Lj}) + b_j\right)$$

wobei a_j = die Aktivierung des Neurons j, L = der nächste Layer, a_L = alle Aktivierungen des Layers L , W_{Lj} = alle Gewichte zwischen dem Neuron j und den Neuronen des Layers L, und b_j = der Bias des Neurons j.

Loss/Cost

Für den Trainingsprozess muss das Programm jedoch wissen, wie gut eine bestimmte "Konfiguration" (Gewichte und Biases) des Netzwerkes ist. Dafür gibt es die sogenannte Cost-Funktion (auch Loss- Funktion genannt), die mithilfe von Trainingsdaten funktioniert. Trainingsdaten bestehen aus einer Liste aus Trainingsdatensätzen. Jeder dieser Datensätze beinhaltet Inputs für das Netzwerk und die richtigen Outputs dafür. Machine Learning, das mit solchen Trainingsdaten arbeitet, wird Supervised Learning genannt. In diesen Datensätzen liegt meist auch die Schwierigkeit von Machine Learning (s. Konfiguration der Trainingsdaten).

Die Funktion für die Cost des Output Layers und somit gesamten Netzwerkes (für einen Trainingsdatensatz) lautet wie folgt:

wobei C0 = die Cost des Output Layers, L = der letzte Layer (Output Layer), a(L) = die Aktivierungen des Layers L, und y = die richtigen Outputs für die Inputs, mit denen die Aktivierungen berechnet wurden.

Um die Cost zu berechnen muss man also für alle Output Neuronen die Differenz der gegebenen und der richtigen Aktivierungen bilden. Danach muss man diese Differenzen quadrieren und am Ende alle Ergebnisse aufsummieren. Dies kann man für alle Trainingsdatensätze wiederholen und von allen Costs den Durchschnitt nehmen, um die allgemeine Performance eines Netzwerkes zu überprüfen. Diese Art der Cost-Funktion wird Mean Squared Error (MSE) genannt. Zusammenfassend kann man also sagen, dass die Cost die Abweichung von den berechneten und den richtigen Outputs angibt.

Aufgrund des Trainingsdatensatzes weiß man nun, wie der Output Layer verändert werden

muss.

Backwardpass

Doch wie verändert man nun die Aktivierungen des Output Layers? Es müssen alle Gewichte und Biases davor angepasst werden. Um nun zu wissen, wie ein Gewicht verändert werden muss, gibt es folgende Funktion:

$$\Delta W_{ij} = \epsilon * \delta_i * a_j$$

wobei ΔW_{ij} = um wie viel das Gewicht W zwischen den Neuronen j und i verändert werden muss, ϵ = die Lernrate (meist ein kleiner Wert wie 0.001), δ_i = die Ableitung der Cost des Neuronen i im Verhältnis zum Weight W_{ij}, und a_j = die Aktivierung des Neurons j. Was dabei oft verwirrend ist: j bezeichnet das Neuron, welches zuerst kommt, und i das Neuron, welches danach kommt (Reihenfolge im Forward-Pass), obwohl es bei W_{ij} andersherum steht.

Für den Bias wird die gleiche Formel benutzt, mit der Ausnahme, dass aj immer 1 ist und so wegfällt. Der Grund dafür liegt darin, dass der Bias, wie in der Struktur beschrieben, eigentlich nur ein Gewicht ist, das mit einem Neuron verbunden ist, welches immer eine Aktivierung von eins hat.

Um nun δ_i für den Output Layer zu berechnen, gibt es folgende Gleichung:

$$\delta_i = sig'\left(netzinput_i\right) * \left(ai(soll) - ai(ist)\right)$$

wobei $sig'(x)$ = die Ableitung von $sig(x)$, also $sig'(x) = sig(x) * (1 - sig(x))$, $netzinput_i$ = der Netzinput des Neurons i , $a_i(soll)$ = die Aktivierung, die das Neuron haben sollte (also das gleiche wie y), und $a_i(ist)$ = die Aktivierung, die das Neuron hat.

Um ein neuronales Netzwerk zu erstellen haben wir Flux benutzt. Wir haben immer wieder mit der Struktur experimentiert, momentan ist sie ...

Zum Trainieren haben wir Supervised Learning benutzt.

3.2.3 Roboter

4 Ergebnisse

Zuerst haben wir versucht, eine KI zu trainieren, welche erkennen kann, ob eine Person gerade geblinzelt hat oder nicht. Diese könnte man zum Beispiel nutzen, indem man einen Roboter immer dann nach vorne fahren lässt, wenn eine Versuchsperson blinzelt. Hierbei gibt es zwei sofort erkennbare Probleme. Zum einen ist Blinzeln ein Menschlicher Reflex, den man nicht abstellen kann. Das heißt, dass es in unserem Beispiel mit dem Roboter nicht möglich wäre den Roboter langfristig stehen bleiben zu lassen, weil man irgendwann blinzeln müsste. Außerdem erkennen wir beim Blinzeln keine Gehirnaktivitäten, sondern nur ...

5 Diskussion

6 Danksagung

6.1 Finanzierung

Danke an den Förderverein „Gesellschaft der Freunde des Gymnasium Eversten e.V.“ für die Finanzierung des EEG-Geräts. Alle finanzierten Teile sind in der Materialliste mit ⋆ gekennzeichnet.

7 Quellen & Referenzen

Literatur

[1] Ander Ramos-Murguialday Ujwal Chaudhary Niels Birbaumer. „Brain-computer interfaces for communication and rehabilitation“. In: *Macmillan Publishers Limited* 12 (Sep. 2016), S. 513–525.