Classifying Epileptic Brain States Using Structural Features of Neuronal Sugihara Causation Networks

Kamal Kamalaldin

CONTENTS

Abstract

Epilepsy is a brain disease that profoundly affects the personal lives of its patients as well as the health care system needed to address it. Machine learning is a field of computer science experiencing a renaissance period at a time where computers can do more work in less time than ever before. While industry has moved rapidly towards implementation of machine learning in profitable and futuristic endeavors like autonomous driving and image recognition and creation, some basic and essential societal problems remain relatively unexamined through the lens of machine learning research, an example of which is epilepsy. This work attempts to start to fill this gap and shed light on the ways machine learning can address complex and intricate problems like epilepsy. Through it, several machine learning algorithms will be surveyed, and some simple implementations and tutorials be provided through an external companion. Furthermore, a novel definition of brain communication is introduced in the form of causality, which will also be discussed more broadly. This definition is then used to construct a network of brain communication patterns in the context of which the author attempts to apply machine learning. Finally, discoveries made in this work will highlight a lack of a general understanding and framework of brain communication patterns and the magnitude and scope of future work available in the intersection of neuroscience and machine learning.

I. Introduction

Machine learning is a field that stretches across computer science, mathematics, and even psychology. One of its main purposes is to teach computers how to perform actions without explicitly programming the computers how to perform the actions correctly. For example, instead of programming a computer to play checkers by accounting for every possible situation and move—implementing them with if/else statements—a machine learning approach would be to let the computer play many games and attempt somewhat random moves, keeping track of which moves were beneficial and which were malevolent. Using this approach, the programmer does not need to account for the incomprehensible possibilities for all checkers boards, but rather needs to implement a learning algorithm that would be able to learn from the moves it performs. This is where computer science intersects psychology, since most if not all of the machine learning approaches are inspired by how humans perceive a problem and approach it. These approaches to describe, replicate, and augment the human approach to solving problems are manifested in mathematical frameworks, which is where computer science intersects mathematics in this field.

Since its beginning in the early 19th century, the field of machine learning has been gradually applied to solving many problems in computer science. Some of these problems include optical character recognition, spam detection, weather prediction, fraud detection, and language understanding [1]. Recently, some breakthroughs in neural networks, a specific branch of machine learning, has resulted in an increased interest in machine learning and its real-life applications. This recent development—called deep learning—is being applied to develop autonomous driving vehicles, read people's lips [2], Compress images more efficiently [3], simulate gas and liquid physical spread [4], and literally judge books by their covers [5]. Such projects might be advanced, but the machine learning community is bridging the gap between the highly academic research oriented projects, high profit industry oriented endeavors, and the computer science labor force that is severely lacking in machine learning expertise. Through websites like *Kaggle.com*, *r2d3.com*, *deeplearning.net*, *deeplearningbook.org*, and *neuralnetworksanddeeplearning.com*, computer scientists can learn the different ways that machine learning can be applied, and have access to resources and competitions that gives them experience in the field. During my own journey in the field of machine learning, I came across several problems and ways to approach them with the tool box I developed.

This journey led me to the University of Colorado Colorado Springs (UCCS), where I partic-

ipated in a Research Experience for Undergraduates (REU) under the supervision of Dr. Rory Lewis and Dr. Peter Erdi. This paper is a culmination of the learning and discovery made during and since the time I spent at UCCS in the summer of 2016. As a requirement of the REU, Machine Learning techniques were to be applied to a problem of my choosing, and after much deliberation with my advisors, I decided to explore the topic of Epilepsy and attempt to apply machine learning, in whatever extent possible with my knowledge, to address problems in epilepsy treatment. Through this paper, I guide the reader through the topics of Epilepsy, Machine Learning, and Sugihara Causality, amalgamating them into one coherent piece that is used to describe brain states. This paper assumes a certain level of literacy in readership, mainly that of medium level Computer Science and Mathematics background. Whenever possible, complicated terms and ideas will be distilled to simpler ones that may be more familiar to the reader. The paper will begin by describing epilepsy and putting it into the context of our everyday lives. Then, the topic of machine learning will be introduced, giving the reader a broad (but non-exhausting) overview of some popular techniques used. This section will be the most intensive one in this paper, due to two main reasons: since the majority of the time I spent in UCCS has been in discovering the fascinating land of machine learning, I felt it was only appropriate that this paper fully reflects the majesty I found in my search; since machine learning is a mathematically involved subject that most computer scientists might not be familiar with, I believe it is appropriate to thin the content out for the reader to ensure a proper digestion of the information provided. After Machine Learning, the last of the trio of subjects, Sugihara Causality, will be introduced, shinning light into a subject that is not commonly discussed in Machine Learning, Mathematics, or Computer Science, but nevertheless an important piece of my work in UCCS. After the introduction into these three areas of research, this paper will then walk the reader through my attempt at putting the three areas together to make sense of how areas of the brain communicate with one another, and thereby help diagnose and track epilepsy in the brain.

II. Epilepsy

A. Definition

The medical field defines a seizure as a sudden burst of electricity in the brain that causes a disturbance in its electrical activity. An epileptic seizure is more strictly defined as a ”transient occurrence of signs and/or symptoms due to abnormal excessive or synchronous neuronal activity in the brain.” Until recently, epilepsy has been defined as a brain disorder where the patient suffers from two unprovoked epileptic seizures in a span greater than 24 hours. In 2014, the definition of epilepsy has been revised to encompasses patients who display any of the following symptoms:

- At least two unprovoked (or reflex) seizures occurring in a span greater than 24 hours.
- One unprovoked (or reflex) seizure and a probability of further seizures similar to the general recurrence risk (at least 60%) after two unprovoked seizures, occurring over the next 10 years.
- Diagnosis of an epilepsy syndrome. Epilepsy is considered to be resolved for individuals who either had an age- dependent epilepsy syndrome but are now past the applicable age or who have remained seizure-free for the last 10 years and off antiseizure medicines for at least the last 5 years.

The new definition was designed to allow physicians more freedom in diagnosing cases of epilepsy and take drastic measures in special circumstances where the first occurrence of an epileptic seizure can be greatly suggestive of a problem, necessitating an urgent procedure. It also allows patients to ’outgrow’ epilepsy in cases where the condition is age dependent or has not manifested for a prolonged period of time, indicating the condition no longer affects the patient. Most importantly, the new definition classifies epilepsy as a *disease* rather than a disorder, an action that hopes to spread awareness regarding the ”lasting derangement of normal function” that epilepsy can entail on a patient [6].

There are two main types of epileptic seizures: general seizures and partial seizures. The distinction between the two is where they originate and how greatly they spread in the brain. In general seizures, an electric discharge spreads to both hemispheres of the brain causing the electrical dysfunction that results in the seizure. Similarly, partial (of Focal) seizures are caused by an electric discharge, but this charge spreads only to certain areas of the brain in which it causes a dysfunction leading to the seizure syndrome [7].

B. Syndrome

Epilepsy patients can experience a wide range of physical syndromes, not all of which are observable to a bystander. Some observable syndromes are

- drooling; inability to swallow; difficulty talking.
- repeated non-purposeful movements; rigid or tense muscles; tremors, twitching or jerking.
- pale or flushed skin color; seating; biting of tongue.

Some non-physical signs that the patients report feeling are

- loss of awareness; forgetfulness; distraction; daydreaming
- loss or obscurity of vision, hearing, and sensation
- body feels different, or out of body feeling

C. Epilepsy in Numbers and Life

In the United States alone there are 150,000 new cases of epilepsy a year, or about 48 cases for every 100,000 people. In total, there are about 2.2 million cases of epilepsy in the United States, averaging close to 7 cases for every 1000 people. The disease onset is most common in young children and older adults [8].

A recent review [9] showed that epilepsy ”creates a substantial burden on households” through either loss of productivity or out of pocket costs. While the review noted that the burden could be offset by health insurance, the upfront costs associated with treatment are significant. In Italy, the annual cost per patient is $1,736, while in Spain it was $2,813, not accounting for a $2,924 surgery consultation fee. While these costs represent health system costs in socialized medical care societies, and are therefore shared over a larger population, the case is less forgiving in countries where individual insurance is the norm. In the US, out of pocket cost for a hospital stay is $1,018. The review also considered indirect costs by associating productivity and employment status changes. While employment status did not differ after the diagnosis, productivity losses accounted for an annual average of $2,037 in Spain, and $2,146 per 3 months in Germany. The review concluded that the highest costs of epilepsy treatment come from surgery. A different study [10] showed that hospital charges for epilepsy patients has increased by 137 .9% between 1993 and 2008, rising from $10,050 to %23,909 while the average length of stay decreased from 5.9 to 3.9 days.

Epilepsy also affects patients' lives in other ways. Patients become restricted to their homes due to a fear of having a seizure during their child's recital, work, or even grocery shopping.

Even if the fear of embarrassment subdues, patients take a risk when performing activities that put them in a position in which they can endanger them and the people around them. For example, deciding to ride a bike becomes a more challenging preposition since the patient could suffer from a seizure and endure serious injuries while biking. More importantly, driving is an increasingly risky activity with seizure prone patients. Driving laws make it difficult for patients to live an ordinary individual life with driving restrictions on patients who suffer from seizures. Each state in the United States has its own laws regarding the requirements that patients must meet in order to be allowed to drive on the road. In most cases the patients physician is involved in the decision on whether or not they should be allowed to drive.

D. Treatment

No real cure for the dysfunction of electrical activity in the brain has been developed, therefore epilepsy has no real cure as of yet. Most treatments attempt to mitigate the symptoms (seizure) by using a combination of strategies including the reduction of brain activity through medication, removal dysfunctional parts of the brain, and changing the chemistry of the body through restrictive diets. The following gives a summary of each of these strategies and the ways in which they work.

1) Medication: Medication is the first and most common treatment prescribed by doctors for epilepsy cases. Medications differ for different cases of epilepsy, and there is no *one* specific medication that is commonly prescribed. Such medications, commonly called AED (Anti-Epileptic Drugs) can completely control seizures in 7 our of 10 patients. AEDs work by suppressing seizures that are caused by epilepsy, and do not cure or address the true underlaying causes of epilepsy. Epilepsy medication is known to only be effective if taken regularly. AEDs commonly induce side effects in patients, some serious like mental slowness, hepatotoxicity, dizziness, and drowsiness, and some minor ones like skin rashes and weight gain [11]. AEDs treatments usually continue until the patient is proven to no longer suffer from epileptic seizures.

2) Surgery: Surgery is usually performed on patients who suffer from partial epilepsy and on whom medication has not been affective. However, decisions to have surgery are being made sooner as some correlation has been shown between how early the surgery is performed and its success rate [12]. Epilepsy surgeries are split into two main categories: resection and disconnection surgeries. Resection surgeries entails removing the part of the brain that causes electric dysfunction and therefore seizures, sometimes resulting in a complete 'cure' from epilepsy. Such

a procedure can have the immediate side effect of losing brain functions or memories like being able to play the Piano or the memory of a child's first word. On the other hand, the goal of disconnection surgeries is to cut nervous pathways that are thought to cause epilepsy in the specific patient. Disconnection surgery is often undertaken if seizure is caused by a vital part of the brain and does not provide a 'cure' from epilepsy but rather a relief for the patient. Even more than typical surgeries, brain surgery can be highly risky and physicians advice it as a final resort. This risk draws a sharp contrast to the convenience of medication, since most AEDs can be administered easily and require only an adherence to the drug schedule. Therefore, medication is recommended before any surgical approach is considered.

3) Diet: Dietary restrictions such as fasting have been used to mitigate the onset of epilepsy since biblical times [13], and recent developments in the nutrition perspective on epilepsy have resulted in the resurgence of what is now known as the ketogenic diet. This diet consists of high fat and low carbohydrate items and focuses on mimicking the body's reaction to fasting by using fatty acids as a main source of fuel for the body. The diet works remarkably well on infants who have a mutation that affects the transport of Glucose (a main body fuel source): with the ketogenic diet helps prevent microcephaly, mental retardation, spasticity, and ataxia as a consequence of relative brain hypoglycemia (lack of glucose). This bolsters the evidence that the ketogenic diet regulates the body to use fatty acids as an alternative fuel for the brain. The ketogenic diet was also shown to help Alzheimer Disease patients, though that might have to do with eating less carbohydrates than eating more fatty acids [14].

III. Machine Learning

Machine learning is a buzz word that most computer scientists—and indeed most people in STEM fields—hear around, but only a few have had proper introduction into what exactly the field of machine learning studied, what it aims to accomplish, and how much success there has been in the field (measured in terms of application to industry). In a nutshell, machine learning is the study of developing algorithms that can solve problems on their own without being explicitly told how to solve them or what directions to follow to reach a conclusion. This might seems anti-intuitive so some computer scientists, after all programming—a task most computer scientists perform on daily basis—is literally the process of telling a computer exactly what you want it to do. However, there are some problems and tasks in which humans *can understand* how to perform, but simply *can not explain* how to perform them, at least not in binary code. For example, teaching a child how to identify the number 3 is an intuitive task: a three is a three if it has two moon-like or semicircle shapes on top of one another. However, programming a computer to recognize a three is a completely different task, because you would have to write an algorithm to detect a semi-circle in pixels, and then you would have to tell it to account for two such shapes. But what if the semicircles are not joined? or if the semicircles were not curvy enough? or if the top part of the three was a straight line, a common font type? or what if the three is written in a small font or a big font? A programmer would have to account for all these possibilities with if statements, and that is only for the number 3. When extended to a whole alphabet, the scale of this problem becomes too large for one program to be written. Machine Learning offers a simple solution for such tedious if not impossible problems: show the computer many 3's and many non-3's and let *it* decide what logical statements to take and when.

In a sense, this is inspired by how humans learn: we are put in situations where we read the same letters over and over again, associating them with sounds and concepts. Over enough period of time, we *learn*, mostly by forming associations with our experiences. Machine learning attempts to mimic that natural phenomenon with algorithms that attempt to do everything by doing very little, succeeding at times and failing in others. This section is an overview of the field of supervised machine learning, its history, and a few algorithms that form the baseline education on the field.

A. History of Machine Learning

The first sight of ML is attributed to Arthur Samuel who, in 1952, wrote the first learning computer checkers game. This game learned by playing against itself and against other human players in a supervised setting [15]. After multiple enhancements throughout the years, this program was able to beat novice checker players, demonstrating the potential power of ML. The same basic principles used by Samuel's program more than 60 years ago are still being used (with more optimized algorithms on greatly more powerful computers) today at Deep Brain, beating humans at more and more board games (See alpha Go).

While Dr. Sameul was working on his checkers game at IBM, a major development was happening at College of Medicine at the University of Illinois. There, Warren McCulloch and Walter Pitts proposed the first mathematical model of what they believed to be the structural unit of the brain: the neuron. In their paper *A Logical Calculus of The Ideas Immanent IN Nervous Activity* published in 1943, they described the brain as a network of neurons in which each neuron had excitetory and inhibitory input and with each of these inputs came a weight to imply its importance [16]. Each neruon in the brain had a certain threshold which represented its reluctance to "fire." If the weighted sum of all its excitetory and inhibitory inputs was greater than the threshold, then the neuron would fire. Otherwise, it would not. This was coined as the "all or none" behavior (Fig. 3).

Mathematically, the model describes a set of inputs $X = x_0, x_1, \ldots, x_n$, a corresponding set of weights $W = w_0, w_1, \ldots, w_n$, and a threshold θ the output of the neuron is described as the function

$$f(X, W) = \begin{cases} 1 & \text{if } \sum_{i=0}^{n} x_i w_i \geq \theta, \\ 0 & \text{Otherwise.} \end{cases}$$

For the neuroscience field, the McCollough-Pitts model was a mathematical breakthrough that many have been waiting for to accurately model neurons and the brain. However, the importance of the McCollough-Pitts model to the machine learning field was in that it laid the ground work one of the most important building blocks of today's Artificial Neural Networks: The Perceptron. In 1958 and at the Cornell Aeronautical Laboratory in Buffalo, New York, Frank Rosenblatt introduced this mathematical model that was very similary to the McCollough-Pitts neuron, but had one major difference which was that it had a learning rule that allowed it to adjust the weights of the neurons that feed it. This allowed the perceptron to accept new input as a series of node

inputs and adjust the weights of each of these nodes in such a way that allowed the output of the neuron to be consistent with the true label of the input. This was the first sight of an algorithm that can find classification solutions for linearly separble problems. However, there was one big glaring limitation to the perceptron model: it did not handled complex problems. For example, the perceptron model can only learn linearly separable functions (Fig. 1). This limitation was laid out at length in a scathing critique of the perceptron model in *Perceptrons* written by Minsky and Papert in 1969, which led to a decade-long hiatus in the field of neural-based machine learning algorithms.

After a few critical papers were published on the ways that perceptrons (referred to as learning units as the field continued to grow) could be connected in different layers to solve previously unsolvable problems, the field was revived and work on what then became known as *neural networks* rose into popularity. However, it was not until the early 2000's that neural networks truly experienced a period of renaissance with many of the applications for neural networks being incorporated into real-life projects like search (e.g. Google, Yahoo, Bing), facial recognition (e.g. Facebook Photos, Google Photos), and human sound understanding (e.g. Siri, Google, Cortana).

B. Supervised Learning

Supervised learning can be generally thought of as function approximation where, given a set of inputs and expected outputs, a function from the input to the output is approximated by using a set of learning rules. This approximated function is then used to predict the output of some unseen input.

$$f(x) = y \xrightarrow[learning]{supervised} f^*(x) = y$$

f^* is an approximates function of f that is learned by only having access to x and y.

The input and output spaces of x and y can be either categorical or continuous. Categorical data can be of the label form, such as True/False, or simply a finite set of keywords. In the categorical space, there are no inequality relationship between categories, meaning that no one category can be said to be "greater than" or "less than" the other, something that *can* be said of the space of continuous numbers (e.g. one can say $2 > 1$, but one can not say "green" ¿ "white"). This distinction proves to be important later on when considering how to create mathematical models that can correctly 'learn'. For the cases when the input and output space

is categorical, the learning process is commonly referred to as *label classification*, an for cases when these input spaces are continuous, learning is called *regression*. We will first address label classification, going over four main models—Perceptron, Neural Networks, Decision Trees, and Regression—and it will be clear why a different approach is required for continuous spaces.

1) Perceptron: The simplest form of label classification is often referred to as binary classification, in which the classifier (machine learning algorithm) has to discriminate between two classes. For example, a simple binary classification problem would be to discriminate between sets of points on a Cartesian plane (Fig. 1).

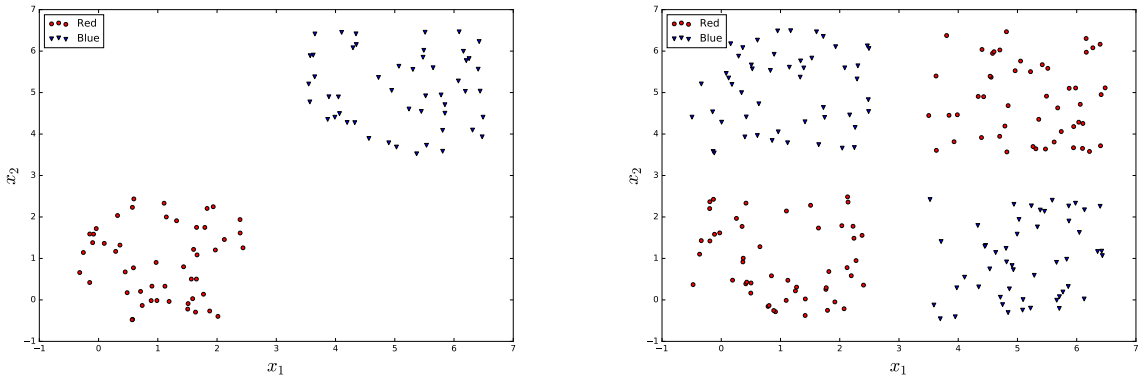


Fig. 1: Two sets of data, one of which is linearly separable (Left) and the other is non linearly separable (Right). Linear separability means that a line can be found that can perfectly segregate the two classes into two sections. A diagonal, horizontal, or vertical line between the blue and red points can be drawn on the left that achieves that goal. However, on the right there is no one line that can be used to separate the red from blue points. This problem of finding a line of separation between two linearly separable data sets can be solved by using a binary classifier, the earliest example of which is the perceptron.

The perceptron was the first machine learning algorithm designed to solve binary classification in linearly separable conditions. Perceptrons were designed to mimic the McCollough-Pitts neuron model which relied on three main components: inputs, weights, and an activation function. In addition to these components, the perceptron had a learning rule that allowed it to learn when to and not to fire. Listing all the components of a perceptron:

- **Inputs**: A dataset of n dimensional points. Each input point x consists of n features such that each input point can be describes as $(x_1, x_2, \ldots, x_n)$.
- **Weights**: n weights described as $w_1, w_2, \ldots, w_n$. These weights are randomly initialized.

- **Activation function**: a function on which the neuron decides to fire or not fire. The first one was the threshold (or step) function where the neuron fires if the weighted sum of inputs and weights is larger than threshold θ, and doesn't fire otherwise.
- **Learning Rule**: A rule by which the weights change, described below.

A perceptron can be trained by passing to it each data point from the the dataset. In our example (Fig. 1), each data point would have a two-dimensional input consisting of the x_1 and x_2 values. Since we are working with two dimensions, the perceptron would have two weights w_1 and w_2. Every time the perceptron encounters a data point it tries to make a prediction by calculating the weighted sum of the inputs S as $\sum_{i=1}^{n} w_i x_i$. If $S \geq \theta$ then the neuron fires, meaning it predicts the category to be $+1$. If $S < \theta$, the neuron doesn't fire, predicting the category to be -1. Given this determination, the perceptron evaluates its predicted output with the actual output of the data point, then manipulating its weights, completing the learning process of the perceptron on that data point.

The learning process for input x, given a true label γ and a threshold θ is summarized by the following updates:

$$S = \sum_{i=1}^{n} w_i x_i > 0 \tag{1}$$

$$\kappa = \begin{cases} +1 \text{ if } S > \theta \\ -1 \text{ otherwise} \end{cases} \tag{2}$$

$$w_{new} = w_{old} + (\kappa - \gamma)x. \tag{3}$$

As can be seen from above, the perceptron learns by changing its weights, as this causes the same input to give a different S which in turn could give a different predicted value κ. Following this learning rule, it can be proven that the perceptron model converges in finite time to a set of weights that result in a perfect classification for all data points in the dataset. A simple implementation of the perceptron was used to solve the sample datasets presented earlier (Fig 2).

The perceptron model did exceptionally well in solving such binary problems when it was introduced, but its lack of versatility when facing non linearly separable datasets resulted in an echoing cry of concern as to how realistic of a learner it could be. In fact, the scathing criticism that the perceptron model received lead to a 10-year hiatus in research on perceptrons [17]. Researchers who worked on the perceptron and neural network fields published under

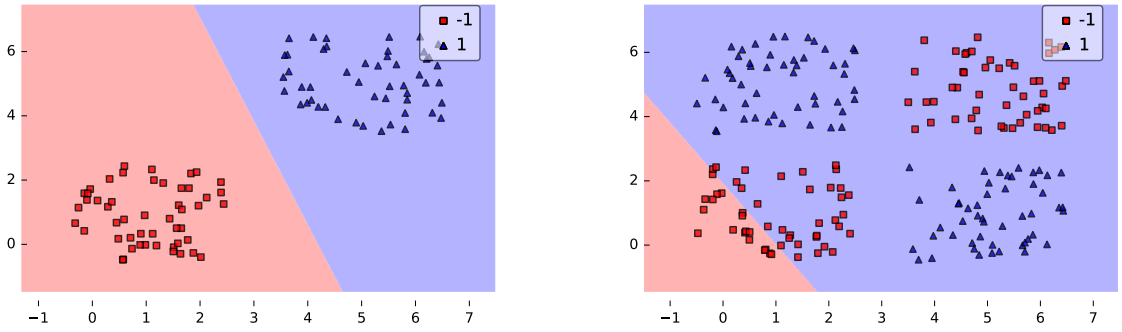


Fig. 2: The perceptron model can be used to solve binary classification in linearly separable datasets as in the left image. After training, all points below the shown line of separation would be classified as -1, and all points above the line would be classified as +1. Seemingly, this behavior would correctly extrapolate to unseen data points from a similar distribution. In the case of non linearly separable datasets, the perceptron keeps adjusting its weights but never reaches a solution which correctly classifies all data inputs, meaning the model quits after a set limit of iterations through the dataset. This results in a having a line that clearly does not separate the two classes (Right).

the umbrellas of ”adaptive signal processing”, ”pattern recognition”, and ”biological modeling” [18]. Slowly but surely, some loyalists to the neural machine learning perspective worked on enhancements to the perceptron model, leading to breakthroughs that skewed public opinion back in favor of what researchers now call neural networks: a group of perceptrons that are linked together to perform learning on datasets that one perceptron alone couldn’t learn.

2) Neural Networks: Neural networks were first implemented by linking perceptrons together to attempt to solve problems that a single perceptron seemed unable to solve. While initial attempts did not indicate that this approach yielded a distinctly superior model, three major breakthroughs were integral to the uphill battle that neural network researchers had to fight in order to regain recognition in the machine learning field. The first breakthrough came when scientists put several neurons together in different topologies (Fig 4), calling them a neural network. In a neural network, layers can be defined as a collection of nodes that are fed data from a previous layer, perform a calculation on this data, and pass it to another layer or as a final result. Although the neural network could approximate more functions, there was no clear way to tune the learning rule from one layer to another. This is when the second and third important ideas came about. The second breakthrough was a realization that choosing a different, continuous activation function (as opposed to the discrete step function) could mean

that a derivative could be taken (Fig. 3). The third was a method by which the chain rule (from calculus) can be employed on the these derivatives that allows error calculations to be passed down from layer to layer reliably, a method called Back Propagation (Fig. 5). Together, these breakthroughs caused a resurgence in the field of neural networks which used perceptrons as their building block. When the activation function changed from a step function, the field moved away from the perceptron naming convention, and used the term *learning units* instead (Fig. 3, 4). The name change is also because the learning rule changes with the change of the activation function, which moves the learning unit further away from the initial conceptualization of the perceptron model.

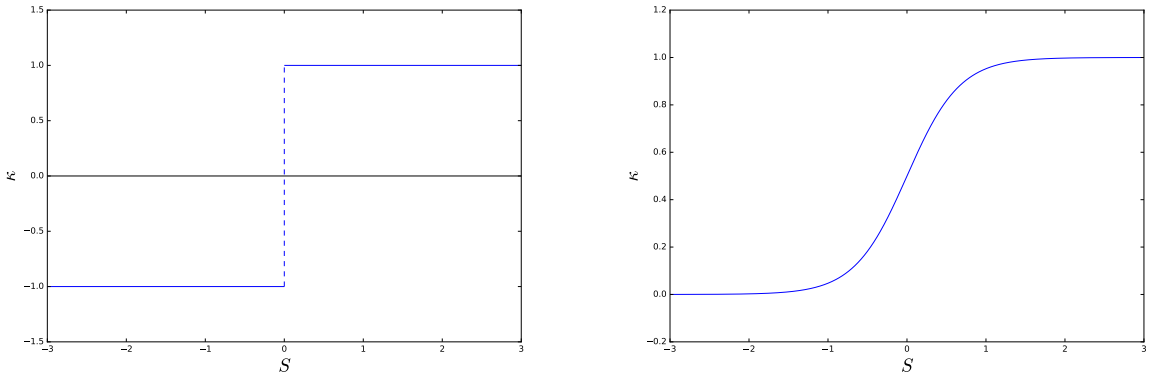


Fig. 3: Different activation functions can be applied to the perceptron model. The simplest and earliest activation function is the step function (Left) which outputs either a $+1$ or -1. A disadvantage of the step function is that small changes in the weighted sum (S) could cause a large change in the output (κ). This is solved by the sigmoid activation function (right) which gives a output (between 0 and $+1$) that is proportional to § within a given range. When a perceptron implements a sigmoid activation unit, it is referred to as a sigmoid learning unit.

A common continuous activation function for learning units is the sigmoid learning unit. Instead of the stepwise function in equation 2, the output function becomes

$$\kappa = \sigma(S) = \frac{1}{1 + e^{-S}} \tag{4}$$

which is bounded by 0 and 1. This function is popular because its derivative is rather simple to compute and leads directly into the theory of back propagation. When the sigmoid function is used, the learning rule becomes a minimization of an error term using gradient descent. For a

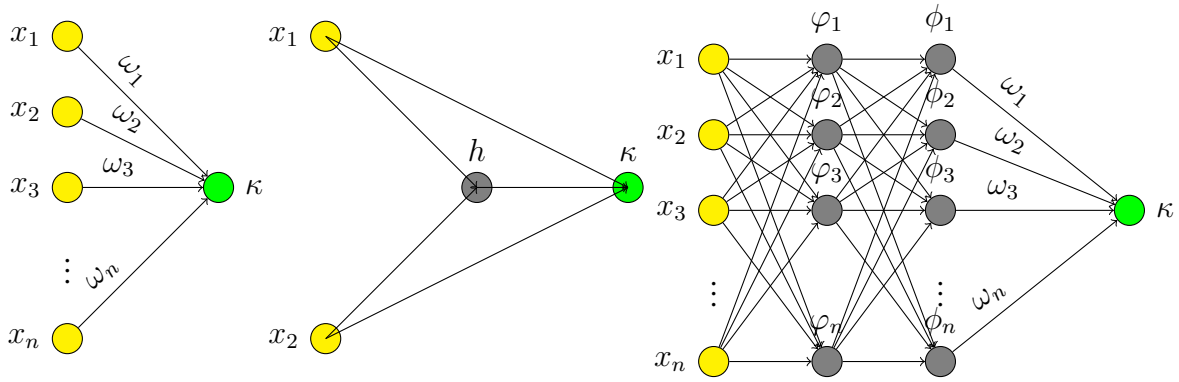


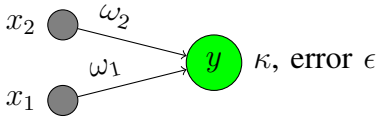
Fig. 4: Perceptrons linked in networks with different topologies. Yellow represents input notes, gray represents hidden nodes, and green represents output nodes. The left network contains only one layer, the input layer, with its associated weights that feed into the output layer. This network essentially simulates a computation that is equivalent to the perceptron model. The center network contains a hidden layer composed of simply 1 node. This network can solve the XOR problem, and is able to solve the linearly inseparable dataset in Fig 1. The more nodes added to the hidden layer and the more hidden layers there are, the more communication happens between the nodes in the previous layer, and therefore the higher the abstraction. The network on the right is a more dense network with two hidden layers and many more hidden nodes.

data point with target classification λ and predicted classification κ, the error term is described as

$$\epsilon = (\lambda - \kappa)^2$$

and is called MSE and is derived in III-B4. Gradient descent is an iterative approach to finding the maximum or minimum of a continuous function by using its derivative (Fig. **??**).

For example, if a learning unit predicts an output of κ with an error ϵ and it has two learning units feeding it



and keeping in mind that κ is $\sigma(S) = \frac{1}{1+e^{-S}} = \frac{1}{1+e^{-\sum_{i=1}^{n} w_i x_i}}$ then the error can be expressed as

$$\epsilon = (\lambda - \kappa)^2 = (\lambda - \sigma(S))^2$$

or in our example

$$\epsilon = \left(\lambda - \frac{1}{1 + e^{-w_1 x_1 - w_2 x_2}}\right)^2$$

Using gradient descent, minimizing ϵ with respect to node x_i requires us to change w_i by the negative of its gradient. This gradient is

$$\frac{\partial MSE}{\partial w_i} = \frac{\partial(\lambda - \sigma(S))^2}{\partial w_i} = \frac{\partial \sigma(S)}{\partial w_i} 2(\lambda - \sigma(S)) \tag{5}$$

because $\sigma(S)$ is the only function of w_i. This is where having an activation function that can be easily differentiated is advantageous. For the sigma function, the derivative evaluates to

$$\frac{\partial \sigma(S)}{\partial w_i} = \frac{\partial \frac{1}{1+e^{-\sum_{j=1}^{n} w_j x_j}}}{\partial w_i} = \frac{x_i e^{-\sum_{j=1}^{n} w_j x_j}}{\left(1 + e^{-\sum_{j=1}^{n} w_j x_j}\right)^2} = \sigma(S)'. \tag{6}$$

The last step is carried for notational simplicity. Combining equations 5 and 6, we find that the weight of the ith node w_i must be changed by the negative of the gradient

$$2\sigma(S)'(\lambda - \sigma(S))$$

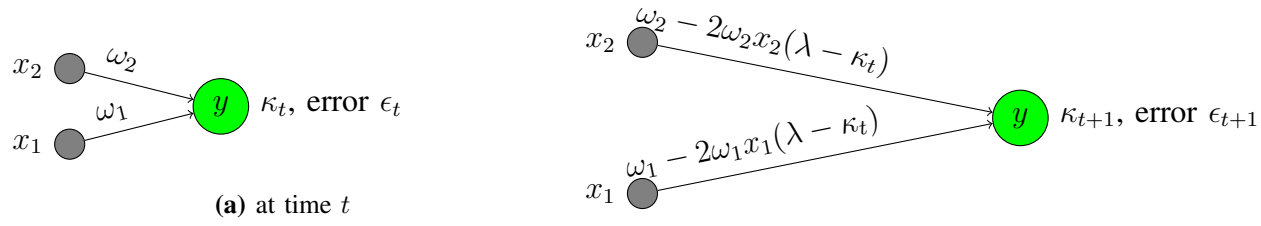
which leads to a weight update rule of

$$w_{i_{t+1}} = w_{i_t} - 2\sigma(S)'(\lambda - \sigma(S)) \tag{7}$$

In this way, the error that is produced at the top layer, where the final classification decision is contrasted with the desired decision, trickles down to all lower layers, with each node changing its weights according to the error of the upper node. Back propagation is worked on out example above in Figure 5.

Some of these networks, if constructed properly, can solve linearly inseparable problems.

3) Decision Trees: A completely different approach to label classification is that of using decision trees to find the features that are most likely to hint at a correct approach towards finding a solution. For example, consider a problem where both the input and the output of the data is categorical. One such dataset is the swimming pool and weather dataset, in which each data point is a weather description of a day and whether or not the tennis court was open on that day (Table I). In this dataset there are several weather-related features such as *Outlook, Temperature, Humidity,* and *Wind*. The *Outlook* feature could contain categories such as sunny, overcast, or rainy; the *Temperature* feature could include categories such as hot or cold; the wind category could include Strong or Weak. All these features contain values from finite set of possibilities.



(a) at time t

(b) at time $t + 1$

Fig. 5: Back propagation works by minimizing the error function using gradient descent. In this case, the error function is the MSE, and its partial derivative with respect to each weight from the ith node in the previous layer is $2w_i x_i(\lambda - \kappa_t$ where λ is the desired outcome and κ_t is the predicted outcome at iteration t. In each iteration, applying gradient descent results in a move towards the opposite of the gradient, meaning partial derivative of the error with respect to that weight is negated from the weight (right). This method is guaranteed to converge, however slowly, over well defined continuous activation functions, an example of which is the sigmoid function (Fig. 3).

Through a simple observation at the table, we can perhaps device our own conditions for when the tennis court is open.

If outlook $=$ sunny $\wedge$ humidity = high, then Yes

If outlook $=$ rainy $\wedge$ wind = Strong , then Yes

Otherwise, No

Although this might be a good rule to follow, it might not generalize well to the rest of the dataset, and inspecting the entire dataset by eye becomes unfeasible beyond 20 data points. This is where the decision tree algorithm comes in, offering an automated method of finding such rules by which we can separate data points into *Yes* and *No* piles. In the simplest decision tree algorithm algorithm, the data points are iteratively separated by the feature that splits the data most evenly (or produces the most homogeneity within a feature split). For example, if we were to take the *Wind* feature and split the Strong and Weak wind conditions into two piles, and observe how mixed or 'pure' the piles are, we would get a sense of how well the *Wind* feature can split up our decision (Fig 6).

This idea of purity or homogeneity is represented in the mathematical formula for entropy (taken from the Information Theory field) as follow

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rainy	Mild	High	Weak	Yes
D5	Rainy	Cool	Normal	Weak	Yes
D6	Rainy	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rainy	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rainy	Mild	High	Strong	No

TABLE I: Weather and Tennis court dataset example [19]. All features are categorical, meaning the values of the features are from a finite, relatively small set of possibilities. The values can not be numerically represented in a formula or compared, nor do they represent values on a range.

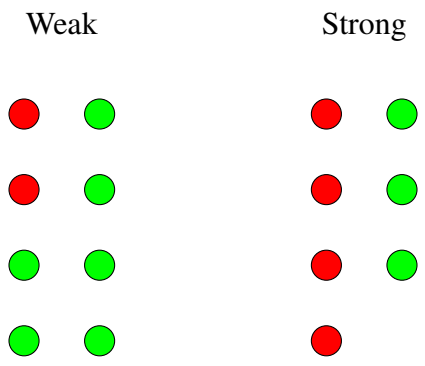


Fig. 6: The decisions to play or not play tennis when the decision is split on wind conditions. Red balls represent No and green balls represent Yes. As we can tell from this graph, splitting on the Wind feature does not provide a homogeneous set (both sides of the split are not purely green or purely red). This means that we can't conclude that whenever the wind is strong tennis can not be played, vise versa. However, It seems that the split caused us to see a relationship between wind conditions and playing tennis, since when the wind was weak most decisions were to play tennis, while when it was wind most decisions were not to play tennis.

$$Entropy = \sum_{i=1}^{n} p_i log_n(p_i) \tag{8}$$

where n is the number of decisions possible and p_i is the fraction of the decisions that were in the ith decision. This definition can be used to calculate the entropy in each of our splits. In our *Wind* feature split example, we can measure the entropy in our Strong and Weak split. In each split our decision possibilities were two (Yes or No); our Strong split had 0.57 fraction No and 0.43 Yes; our Weak split had 0.25 fraction No and 0.75 fraction Yes. So the entropy calculation for the Strong split becomes

$$-\Big(0.57 * log_2(0.57) + 0.43 * log_2(0.43)\Big) = 0.985$$

and for the Weak split the entropy is

$$-\Big(0.25 * log_2(.25) + .75 * log_2(.75)\Big) = 0.811 \ .$$

Figure 7 provides some intuition as to what entropy measures. Using this metric, we can devise an algorithm to iteratively pick the feature that that contains the least entropy when splitting the data upon. However, a more concrete way of using Entropy is measuring *Information Gain* on each split. Information gain is a measure of how much entropy is lost through a transition. The equation that describes information gain is

$$\textit{Information Gain} = Entropy(S) - \sum_{v \in Values(F)} \frac{|T_v|}{|T|} Entropy(T_v) \tag{9}$$

where F is the feature we are splitting on, $|T_v|$ is the number of decisions with split value v, $|T|$ the number of total decisions, and $Entropy(T_v)$ is the measure of entropy on the decisions with split value v. In our wind example, the entropy of the original system (without any splitting on any feature) is $-\big(\frac{5}{14} * log_2(5/14) + \frac{9}{14} * log_2(9/14)\big) = 0.94$. Since we had two splits on *Wind*, let v_1 be the Strong split and v_2 be the Weak split then

$$\begin{aligned}
\textit{Information Gain(Wind)} &= Entropy(Original) - \frac{|T_{v_1}|}{|T|} Entropy(T_{v_1}) - \frac{|T_{v_2}|}{|T|} Entropy(T_{v_2}) \\
&= 0.94 - \frac{6}{14} 0.985 - \frac{8}{14} 0.811 \\
&= 0.054
\end{aligned}$$

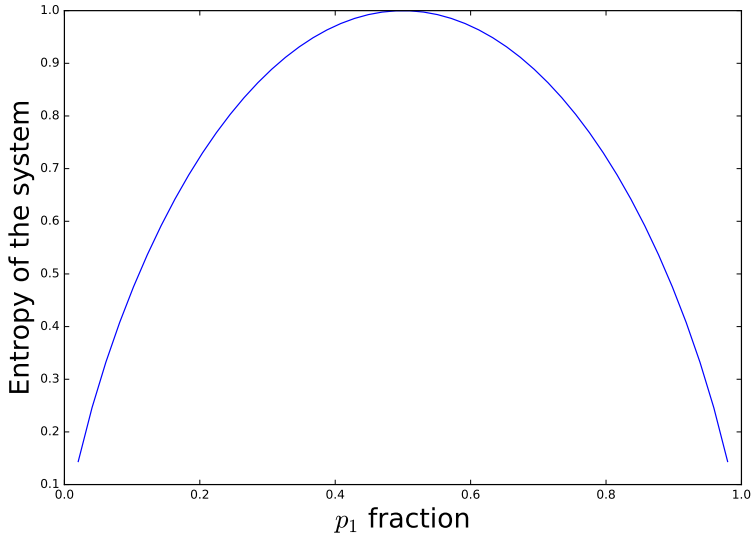


Fig. 7: The entropy of a system with two decisions each of which have a fraction p_1 and p_2 (fraction p_2 is not shown here because it is simply $1 - p_1$). When the system has both fractions in equal proportions, it is perfectly heterogeneous, so it has the highest entropy measure of 1. On the other end, when one fraction dominates the system, the entropy goes to 0. In this way, entropy measures how ”polluted” or pure a system is, giving a higher score to disorganized, polluted systems, and a lower score to homogeneous, pure systems.

In this manner, we can measure the information gain on splitting on each available feature, and choose the feature that results in the highest information gain split. This would give us two to three bins, on which we iteratively carry out this process again choosing another feature and so on. A decision tree is the amalgamation of all these decisions, put in a concise diagram (Fig. 8). This algorithm has the distinct advantage of being able to display its reasoning efficiently, a feat that some machine learning algorithms can not accomplish.

4) Regression Prediction: For cases when the desired output of a function is a number on a meaningful range, label classification is not possible. This is because a supervised label classification problem deals with *choosing* the best label out of a given set, and not generating a label out of the given data. This is where regression prediction comes into play. For functions whose output is a continuous range of numbers, one can not possibly assign a label to each possible output. And even if each output is assignable, there is rarely a dataset big enough to cover possible mappings to each such label.

For example, given an input set of $1, 2, 3, 4, 5, 6, 7$ and an output set $1, 4, 9, 14, 25, 36, 49$, a

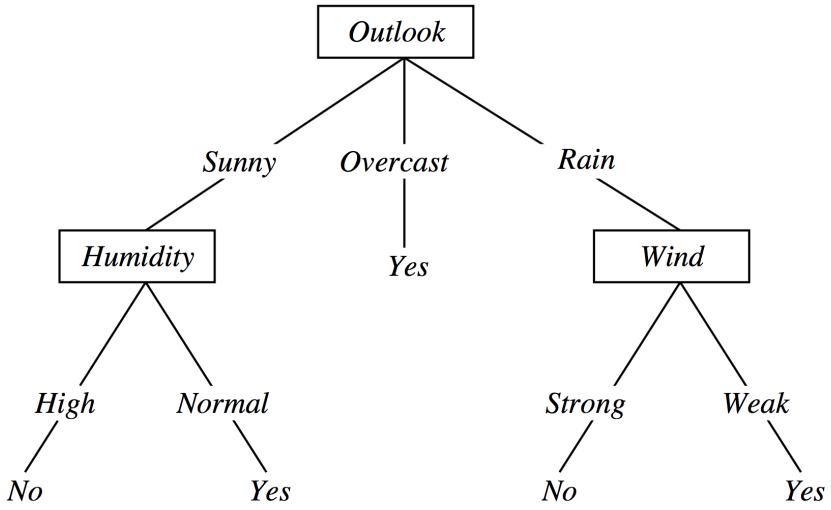
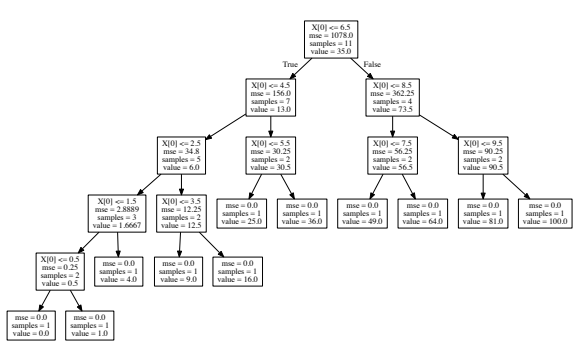


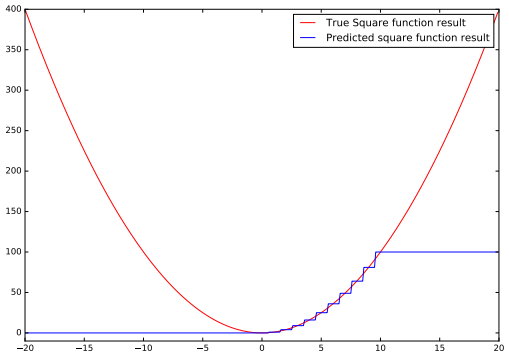
Fig. 8: A decision tree that can be generated by iterating over the tennis-weather dataset using the information gain equation. This is taken from Michel's book on Machine Learning [19].

good—and almost any—supervised learning algorithm should approximate the function $f(x^2)$. However, such a function would have rely on regression learning, and not label classification. To show this, let's delve into a thinking exercise. Say we make a training set for the function $f(x) = x^2$ consisting of input that is all the integers from one to one million and output that is the square of all these input numbers. If we are to use label classification, then we would train the algorithm to map each number to its square label. The first problem arises on the first step of learning, especially in the Decision Tree technique. Because each input is looked at as a categorical input, an input of one (1) will be converted to a label. Since each data point has its own label, the information gain will be maximized and the entropy loss will be minimized when the input is split into one million branches on the first level of the tree. At that point, the function mapping is resolved by simply going through one level. However, this leads to the second problem, which arises when feeding in new test data that the classifier was not trained on. Because the classifier only has access to labels that it trained on, any number other than an integer from one to million will have to map to an already known label from the squares of the integers from one to one million. For example, an input of 1.76 will map to either 1 ($1^2 = 1$) or 4 ($2^2 = 4$), meaning this incurs an accuracy cost. This cost might not be a big problem for the purposes of the algorithm, but the real problem arises when we try to predict the mapping of a number outside the range of our training dataset: numbers below 0 and those above one

million. The nearest value for all numbers below 0 is 1, so all the numbers below 0 will be mapped to the label of 0, while all the numbers above one million will be mapped to the label of one million (Fig 9).



(a) Regression Decision Tree



(b) Regression Decision Test Results

Fig. 9: A Decision Tree trained on a dataset of simple square functions, where the inputs were numbers from 0 to 10 and output is the square of each number. The tree shows a jagged approximation of the square function between 0 and 10, which indicates some level of learning. However, the tree shows very poor generalization outside the learning range. When tested on input that is below 0, the tree matches to the closest label in the decision boundary, which is ”0”. A similar decision is made for input that is greater than 10, where the trees matches all numbers greater than 10 to the label of 10 (”100”).

From this small experiment, the need for a different approach to supervised learning is obvious. Such a technique would need to generalize the mapping not to categorical labels and their limited space, but to an unlimited range of numbers. This method would solve a more conventional mathematical problem of function approximation. Another way to think about the problem is curve fitting: given input and output sets, the curve of the approximated function should pass—or be proximal—the points given in the sets. The best fit for our function would be a curve that passes perfectly through all our data points. As the approximated curve moves further and further away from the given points, the distance between the points and the curve increases, which we can use as a measurement of error (Fig. 10). The sum of this error can be expressed as

$$Error = \sum_{i=1}^{n} (y_i - f(x_i)),$$

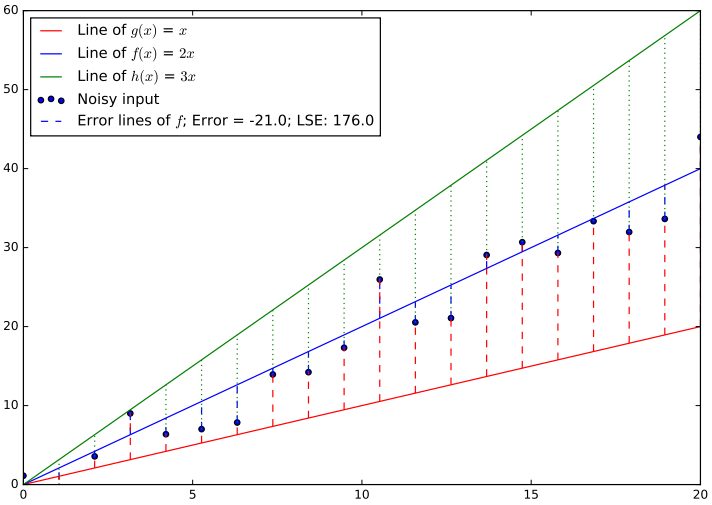


Fig. 10: Different lines of fit and their errors (dashed and dotted lines) with respect to the noisy input data. Because $g(x)$ and $h(x)$ have larger error lines than $f(x)$, $f(x)$ is the best fit for the data out of the three drawn lines. Note that for $f(x)$, the negative and positive errors cancel out, leading to a sum of error of -21. This is allayed by summing the square of the error instead, which results in an equation that can be minimized to 176.

where for the ith data point y_i represents the actual y value and $f(x_i)$ represents the approximation of the regression line at x_i. Minimizing this error term ensures that our resulting function is the closes line fit to our data points. However, notice that in our error function, positive and negative errors will cancel out, leading to the assumption that "two wrongs make a right". To avoid this, we can either take the absolute value of the error, or the square. For mathematical convenience, the squared error is preferred. This results in the method of Least Squared Error (LSE) as given by

$$LSE = \sum_{i=0}^{n}(y_i - f(x_i))^2 = \sum_{i=0}^{n}(y_i - b_0 - b_1 x_i)^2 \tag{10}$$

From this equation it can be shown that

$$b_1 = \frac{\sum_{i=1}^{n} y_i x_i - \frac{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i}{n}}{\sum_{i=1}^{n} x_i^2 - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n}} \tag{11}$$

and

$$b_0 = \bar{y} - b_1\bar{x}$$

where $\bar{y}$ and $\bar{x}$ are the respective means of all y and x values [20]. This formula guarantees a line of best fit that minimizes squared errors for linear data points. For cases when the data points do not display a linear behavior, they can be transformed into a space where they become more linear, linear regression applied, and the regression functions transformed back into the original space (Fig 11).

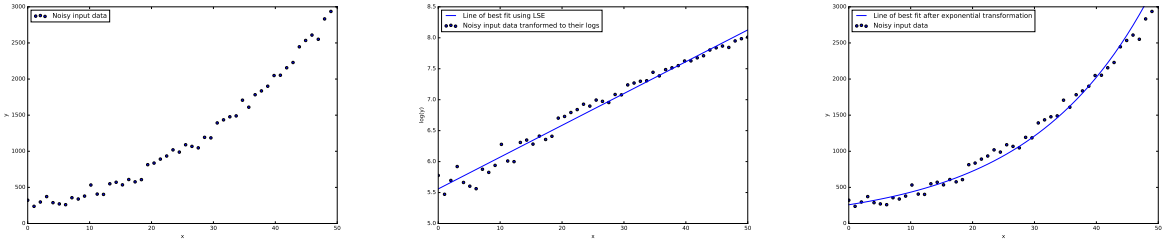


Fig. 11: Finding line of least squared error for data with an exponential trend. First, the data points (Left) are transformed to their logs (Center). Then a line of best fit is found through linear regression (Center). The line is then transformed back to the original space by applying the exponential function (Right).

Using this regression model for out previous example in Figure 9, we are able to attain more realistic learning through a model that extrapolates more correctly to unseen numeric data (Fig. 12)

C. Unsupervised Learning

k-mean

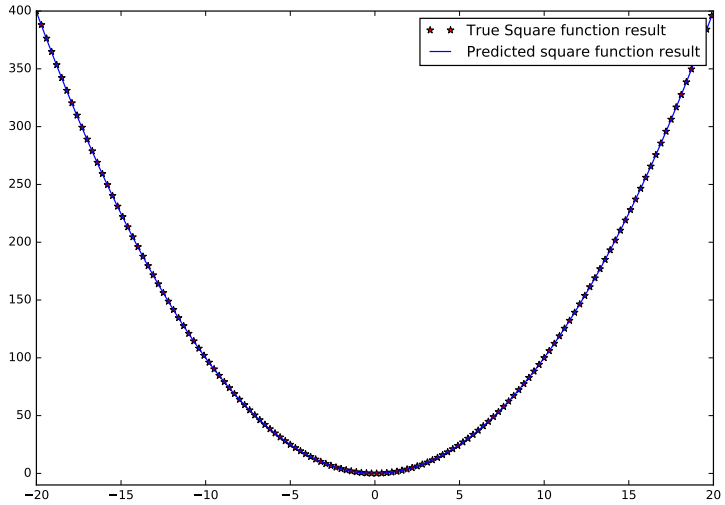


Fig. 12: Using a non-linear regression model for the same data set used on in Figure 9, we are able to build a model that correctly extrapolates to unseen numeric data. Since this is one of the requirements for problems with outputs being continuous numbers, thsi model is highly valuable for prediction in such output spaces.

IV. Sugihara Causality

The concepts of abstract correspondence, correlation and interpreting causation has been discussed in philosophical literature at least as early as Berkley's and Locke's arguments on human perception [21] [22]. Until now, the debate focused on what constitutes a causative effect and how such an effect might be discerned. From philosophy, the debate has moved to empirical science, where different models of causality have been proposed, none of which has been declared the true standard. Causality is often mistaken for correlation because the relationship between the two is not always clear. For example, falling down the stairs could be correlated with breaking a limb, but is falling down the stairs caused by breaking a limb, or is breaking a limb caused by falling down the stairs? One might argue that a person wouldn't break a limb without falling down the stairs, but what if a limb was stressed enough that it broke while a person was normally walking down the stairs, causing the person to lose control of their leg and therefor fall? In this sense, causality is a murky subject that is often considered to be highly subjective and at times nondetachable from the context of the situation. With that being said, there have been many attempts to address the concept of causality in mathematics in the form of the causality of continuous signals on one another. For example, if one could attain the stock prices for hot dogs and hot dogs buns, one would be able to hypothetically see a correlation between the sales of hot dogs and hot dog buns. It is also clear that one can describe a logical causation between the two stock prices, since the products are complementary to one another: if the price for one goes down, the price for the other goes down as well. For the case when substantial fluctuations happen in the stock price for both products, how would one determine which product caused the change in the price of the other? Surely the price drop was caused by *something*, so which of the products caused the other? This is where causality models come into play—this is an example of the use of causality models in the field of Economy.

A particular causality model, Granger Causality (GC), has been widely used in application in the econometric fields [23], and has been the de facto model when causality is concerned. The way Granger Causality works is beyond the scope of this paper, but suffice it to say that Granger Causality behaves best in linear, stochastic systems. However, this causality model carries its own limitations, as, even with extensions to non-linear systems, it has generally not been seen capable of inferring causality in deterministic systems where feedback loops and nonlinearity are a defining feature. New models of causality have been introduced to attempt to go beyond these

limitations. Dynamic Bayesian Networks and, more recently, the Convergent Cross Mapping (CCM) are some such models. The CCM model, also called Sugihara Causality model, relies on the convergence of distance of nearest neighbors in the shadow manifold of pairs of variables [24]. To understand what all this means, we must first discuss manifolds (also called attractors).

A. Manifolds

Given three time series, a manifold of those three series is a new, three-dimensional representation of the state of the three variables at each time point. If the time series are somehow related to a specific concept, then this concept is called a *system*, and the manifold then represents the state progression of the system. To illustrate this, a common example is used, namely the Lorenz Attractor. The Lorenz system is made up of three time series that are governed by the following differential equations

$$\frac{\partial x}{\partial t} = \sigma(y - x) \tag{12}$$

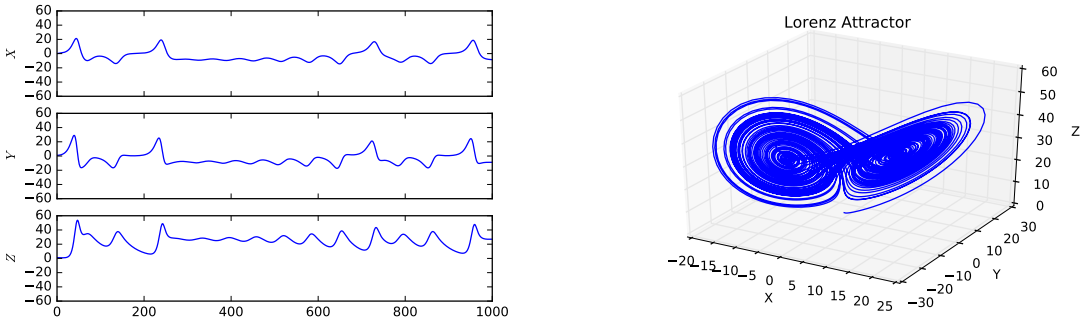
$$\frac{\partial y}{\partial t} = x(\rho - z) - y \tag{13}$$

$$\frac{\partial z}{\partial t} = xy - \beta z \tag{14}$$

where σ, ρ, and β are constants [25]. With an initial condition and some time steps, we can produce three time series like in figure 13a. Given these three time series, a manifold M can be constructed by creating a new time series from the combination of these three series such that each time point t in the manifold, M_t, is described as

$$M_t = (x_t, y_t, z_t).$$

Of course, since this manifold has information about all the three previous time series, it is more highly dimensional than the time series themselves. In this case, since the manifold is of three time series, it is three dimensional (Fig. 13b). Furthermore, since the manifold completely encapsulates the information about the system, the time series x, y, and z can be generated if one has access only to the manifold of the system.



(a) Lorenz system time series

(b) Lorenz Attractor

Fig. 13: A Lorenz system with initial conditions $x = 0$, $y = 1.0$, $z = 1.05$, $\sigma = 10$, $\rho = 28$, $\beta = 2.667$, and time differential of 0.01. The series represent interlinked components in a system, and the manifold is made up of all the three series combined. The manifold is constructed by mapping the combination of x, y, and z at each unique time point to a coordinate in an n-dimensional space, where n is the number of components—$n = 3$ in this case. The resulting mathematical n-dimensional object is called a manifold, and can be used to describe the state of the system in terms of complexity, flux, and stability. In this given example, it can be seen that the system gravitates towards two main "loops", and therefore the system is *attracted* to those points in the three dimensional space.

B. Shadow Manifolds

A shadow manifold of time series ω is an E dimensional reconstruction of E delayed signals of ω. For example, if our ω variable is actually the *sin* function, then to construct E delayed signals of ω we would calculate

$$\omega_1(t) = sin(t)$$
$$\omega_2(t) = sin(t-1)$$
$$\omega_3(t) = sin(t-2)$$
$$\vdots$$
$$\omega_E(t) = sin(t-(E-1))$$

and then to construct a shadow manifold of ω we would construct a manifold of ω_1, ω_2, ... , ω_E (Fig. 14). Since the scalar by 1 increment seems too general, the signals are delayed by

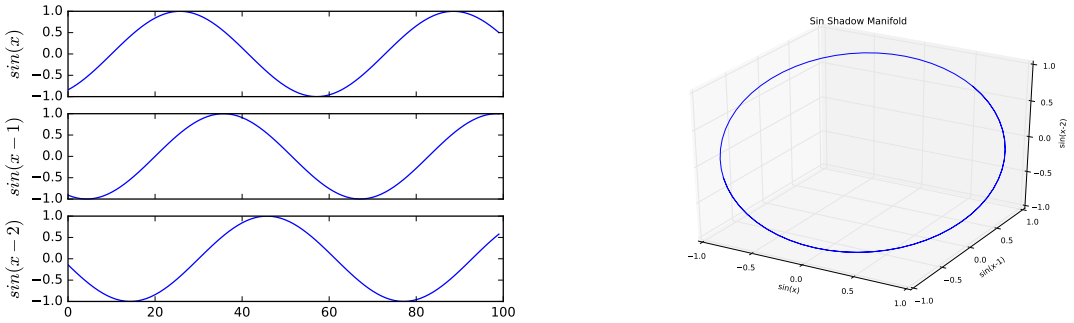


Fig. 14: A shadow manifold of a simple sin function. The delayed signals are created by delaying the series by a scalar constant τ (in this case $\tau = 1$ and $E = 3$ so the function is delayed by $1, 2, 3$). From these delayed signals, a manifold is constructed by mapping each time point of all the delayed signals to an E-dimensional manifold called a shadow manifold.

multiples of τ, where τ is approximated by field knowledge or mathematical analysis. Therefore, the formal way to describe shadow manifolds is that they are signals delayed by a scalar multiple of $E\tau$ such that shadow the manifold of ω, M_ω, is described as

$$\begin{aligned} M_\omega &= f\Big(\omega_1(t), \omega_2(t), \omega_3(t), \ldots, \omega_E(t)\Big) \\ &= f\Big(\omega(t), \omega(t-\tau), \omega(t-2\tau), \ldots, \omega(t-(E-1)\tau)\Big) \end{aligned}$$

By using a theorem called Takens' embedding theorem, it can be shown that each shadow manifold of a variable is a projection of the dynamic system's manifold, M, that preserves the topology of M [26], [27], [28]. For example, in a dynamic system like the Lorenz Attractor where the dynamics of each variable is affected by the other variables in the system, it can be said that each variable subscribes to the overall dynamic of the system. Therefore, the state of one variable could be used to infer the state of another variable if they are dynamically linked. A visual demonstration of this can be seen in Figure 15 where the shadow manifold of each of the variables in the Lorenz system maintains part of the topology of the original whole system. This important property of shadow manifolds imply that there is a one-to-one relationship between a shadow manifold and the original system manifold. More importantly, this also implies that there is a one-to-one relationship between the shadow manifold of one variable and each of the other shadow manifolds in the system. Therefore, the relationship between each shadow manifold is strong. A great demonstration of this concept can be viewed in the online (Science) version of

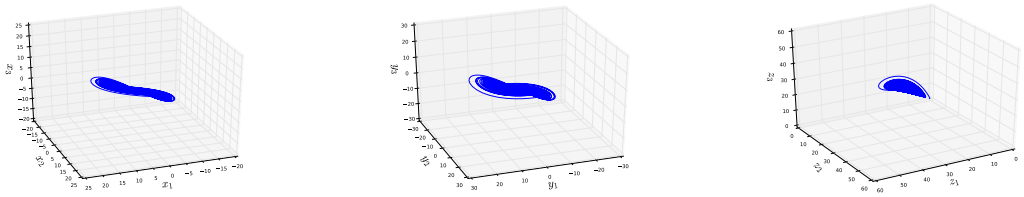


Fig. 15: The shadow manifolds of variables x, y, and z in the Lorenz system. Each of the shadow manifolds retains part of the topology of the original system, as proven by Takens' theorem [28].

the original paper by Sugihara [24] where a video is present to elegantly describe this concept (for ease of access, the reader can also refer to a youtube video called "Takens' theorem in action for the Lorenz chaotic attractor" [29]).

C. Convergent Cross Mapping

Using the mentioned dynamic relationship between shadow manifolds, the Sugihara Causality is calculated as the distance between points of similar time point on each shadow manifold. For variables x and y, the causality at time t is calculated as a measure of the time difference between closest neighbors of y_t. This concept is called cross-mapping, and is illustrated in the supplemental materials on the original paper, and the youtube video referred to previously [24], [29]. The last portion of the Sugihara model is the Convergent in Convergent Cross Mapping, which refers to the convergence of the causality value (cross mapping score) when the library of reference points in x and y, L, is considered. The library L simply refers to the specific range of x and y values to be used in the cross mapping calculation. For example, L could be 20, which means only 20 time-synced data point observations of x and y are taken into consideration, or it could be 100, which means 100 x and y variables are considered. The Sugihara model expects for a causal relationship to *converge* to a value as L increases. This implies that L needs to be sufficiently large to allow an observation of convergence. This convergence is the test used to determine Sugihara Causality, named after its author who describes it as a required but not complete definition of causality [24]. This approach is the first step towards more general and applicable causality models since Granger Causality. Since the introduction of Convergent Cross Mapping, it has been shown to be successfully predictive in biological [30], [31], [24], [32], [33] and cosmological [34] applications while showing weaknesses in others [35].

D. Growth of CCM

Extensions to and amalgamations of the CCM model are beginning to surface in literature. Clark *et al.* proposed an extension to CCM that relies on measuring the smoothness of the mapping (also called flow) function ϕ, thereby reducing the L length requirement[36]. Wismller *et al.* proposed a Mutual Connectivity Analysis framework for the ”analysis and visualization of non-linear functional connectivity in the human brain from resting state functional MRI” [37] which relies heavily on CCM. Tajima *et al.* use the fundamental idea of state space reconstruction to find two measures. The first is *Complexity* which is the best embedding dimension for a certain signal (embedding dimension at which the cross mapping is saturated). The second is *directionality*, the difference in cross map skill or embeddedness between two a pair of signals. With those two measures, they show that the brain exhibits different complexities during conscious and unconscious states. Here, we explore the application of CCM in estimating the causality between neuronal regions by constructing a network of pairwise causality. We then analyze features of such networks during normal and epileptic seizure periods, and use this in conjunction with machine learning to observe if Sugihara Causality can unearth modes of communication in the brain.

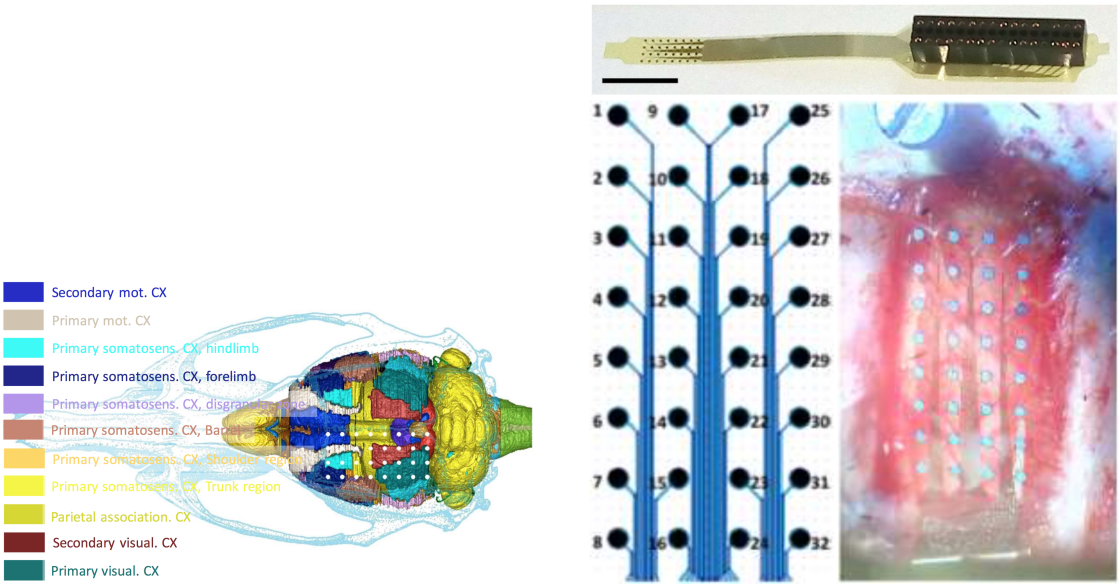
V. Experiment

A. Problem Statement

Given multiple EEG signals from a set of interconnected neuronal regions, can the CCM model identify structural patterns in causal relationships between those regions? Can the patterns and properties of the causality graph resulting from the CCM model be clustered into brain states that represent different stages of epilepsy in the brain? In other words, can a machine learning algorithm be applied to the causal network properties, the model of which could be able to correctly classify epileptic brain states.

B. Methods

1) Data Collection: EEG data was collected from an 4x8 endodermal electrode array at a frequency of 1000 Hz (31 channels, one channel malfunctioned, Fig. 16). The data recording lasts for 500 seconds, during which epileptic seizures were evoked using 4-aminopyridine and EEG data was recorded from the electrode array. The spiking voltage of each recorded electrode is used as a signal and is referred to as a channel (Fig. 17). Each channel has a continuous data series that spans the 500 seconds.



(a) **(b)**

Fig. 16: (a) A composite drawing showing the brain in the scull. The 3D reconstruction of the brain has been made using the maps of the Paxinos atlas, and the localization of cortical areas are indicated by different colors. White points indicate the position of the recording sites of the membrane electrode. Names for the cortical areas are also shown (based on Hjornevik *et al.* and Paxinos *et al.* [38] [39]). (b) Photograph of a membrane electrode shows the construction on the top, the numbering (bottom right) and the surgical implantation (bottom left) is also shown. Electrode 1 malfunctioned during recording.

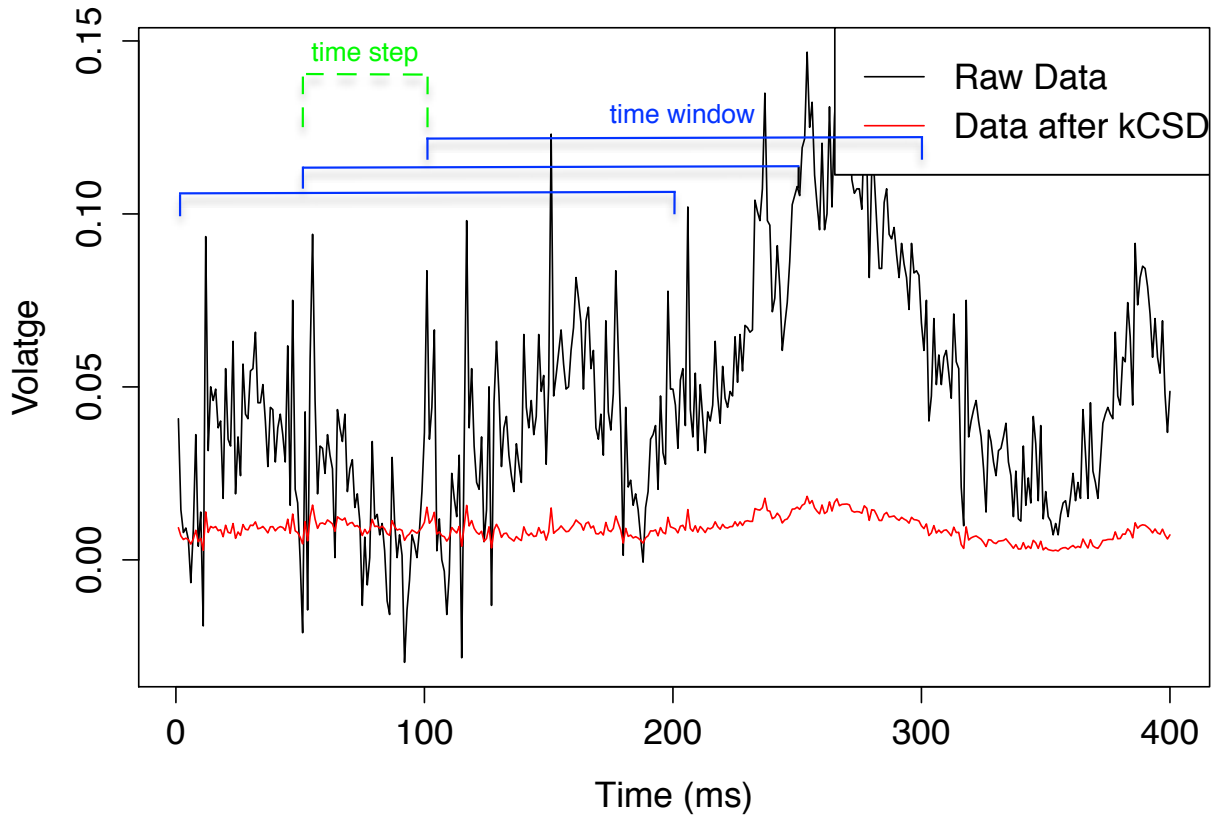


Fig. 17: Sample data from electrode 1, in both its raw form and then in its form after kCSD transformation. Applying kernel Current Source Density extracts the true relationship of neural activity, and in theory eliminates disturbances that might be caused by signal interference. Three sample blue lines are drawn to demonstrate the sliding windows that were used to calculate Sugihara Causality between brain regions. In each time window, the causality was calculated between each brain region, and the time window was moved by an amount *time step*. Different configurations were used for the time window and time step, as reported in table II. Here, the configuration shown has time window as 200 and time step as 50.

TABLE II: Configurations used for the sliding window when calculating Sugihara causality that was used as a measure of brain communication. These configurations inform our model of the granularity of brain communication. Since we do not know exactly how quickly information is being passed down from region to region, arbitrary choices were made to test how well each performs and make a more informed decision in future research. Because we have different time steps, there are a different number of graphs (data points) in each dataset.

Dataset name	Time window	Time step	Number of Data points
dataset 50	200	50	10190
dataset 200	200	200	2547
dataset 250	2000	250	2038

2) EEG Data Preprocessing: Kernel Current Source Density (kCSD) method was used on the grid of channels to account for possible electrical interference with the direct measurement (Fig. 17) [40]. The measured potentials produced by kCSD arise as the linear combination of the trans-membrane currents, which is a more direct and localized quantity to measure the neural activity. Therefore current source density distribution was calculated by the kCSD method and used for the analysis instead of the original EEG signals.

Afterwards, in order to reduce the data size that is operated on, we lumped channels from the same brain regions together by averaging them (Fig. 18). In addition to reducing dimensionality, this process also puts emphasis of causality on functional brain regions instead of a local cluster of neurons.

3) Sugihara Causality Measurement: After lumping the kCSD signals from the same region, we perform Sugihara Causality calculation on the regions by using a sliding windows method (Fig. 17). Because we do not know how granular of a time frame the brain uses to communicate (and how that affects the EEG signal we record), we use three different time scales as reported in table II. This produces 3 datasets with similar characteristics. The important thing to keep in mind is that after this step, each causality measure that resulted from the Sugihara Causality analysis is a measure of how much one region is affecting the other, and we use that as a way to describe brain communication and information flow. Each causality measure is therefore an *edge* of communication from one region to another, meaning that each brain region is a node and each causality measure is a weighted edge. Figure 19 shows a sample graph constructed from joining all the causality measures from a specific time window. This is essentially a (theoretical) snapshot of how each brain region was affecting other brain regions in that time window.

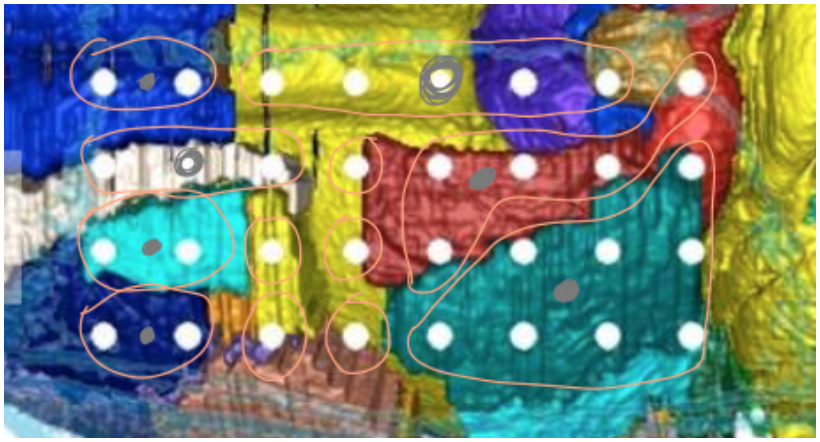


Fig. 18: The distribution and lumping of the brain regions in the brain. A total of 12 region channels were constructed from the initial 31 local channels. The schematic is based on rat brain atlas mapping.

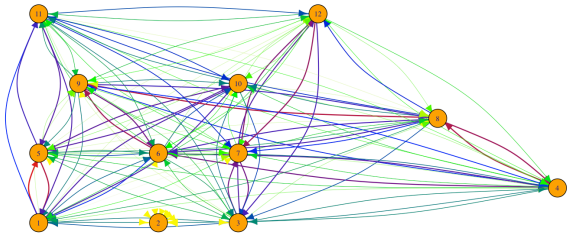
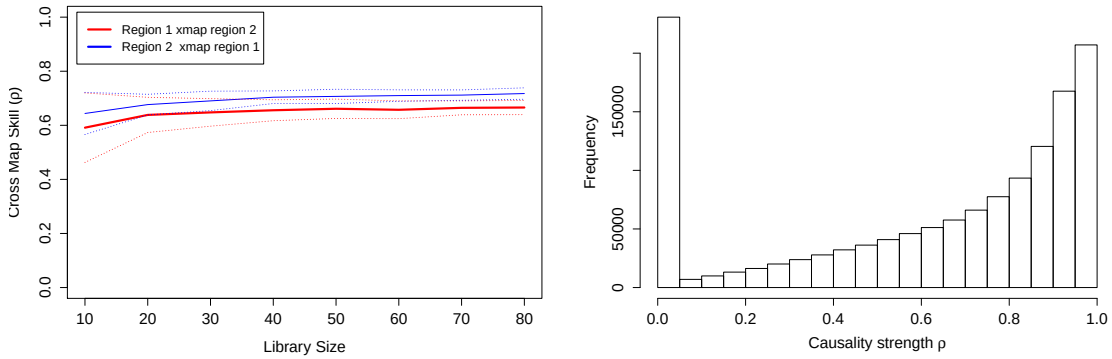


Fig. 19: Sample graph constructed from calculating the Sugihara causality between brain regions. Lumped brain regions correspond to the gray circular mapping in Fig. 18. Edge colors represent the strength of the causal relationship. From weakest to strongest: Yellow, Green, Blue, Red. The edge weight ranges from 0 to 1.

4) Data Exploration and Visualization: The data we produced is a series of complete, directed graphs (all the same topology; each node is a brain region as seen in figure 16; each edge is a causality/communication measure) and their edge weights that we refer to as ρ scores (ρ is the convention name for the score of a Sugihara Causality analysis). Since we processed the data with different time steps, there are a different number of graphs available in each dataset (Tab. II). For each graph, there are 132 ρ scores, all of which range between 0 and 1 (decimals). Each

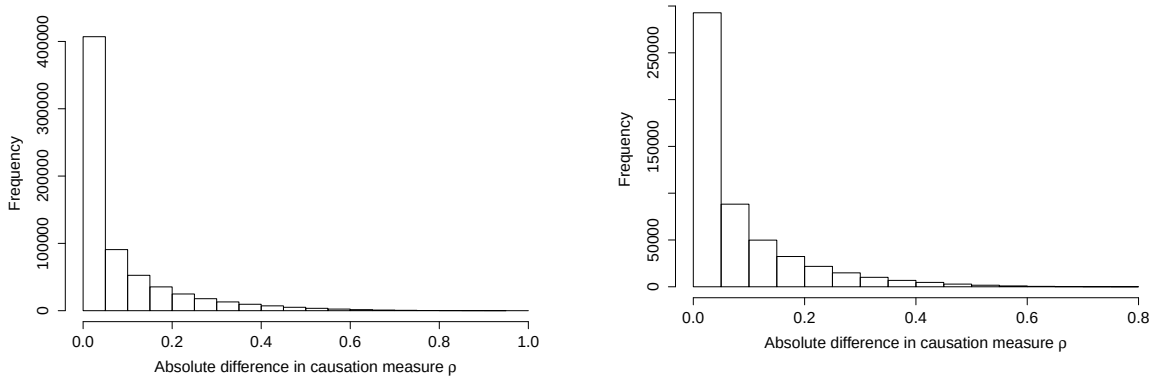
graph was transformed into a vector where each vector is a data point, and all vectors make a dataset. This transformation was required for the data to be entered into sklearn's Gaussian Mixture model function properly.

In all graphs, most of the ρ scores appear to be of high values, suggesting high connectivity between brain regions and therefore a high density graph (Fig. 20b). This indicates that there is constant information flow between regions in the brain. Furthermore, we also notice that many of the ρ scores to be similar within a pair of nodes (Fig. 21). For example, when looking at regions 1 and 2 within the first 200 milliseconds, they appear to be equally communicating to one another (Fig. 20a). This could be indicative of either bidirectional causality or unidirectional forcing, both of which can be detected by the Sugihara model.



(a) **(b)**

Fig. 20: (a) A sample graph plotting the convergent cross mapping skill (ρ) between region (nodes) 1 and 2 during the first 200 ms of the experiment. The skill mapping channel 1 to channel 2 is very similar to the one mapping channel 2 to channel 1. This might infer either bidirectional causality or unidirectional forcing. A similar pattern (close ρ value between pairs) was found for most of the pairs of edges. (b) The distribution of CCM skill (ρ) during the entire experiment. Many relationships appear to be causal in the brain, with equally as many being non-causal throughout the experiment. Causality was calculated from signals of lumped regions after calculating the CSD. Cross mapping was done on every pair of regions with library size of 80, and each pair has two causality directions. Sliding windows of 200 ms were used, with a sliding step of 50 ms.



(a) Absolute difference of ρ between all pairs

(b) Absolute difference of ρ between pair both of which is at least 0.2

Fig. 21: Difference of causality values within a pair is small, even when accounting for non-existing relationships where both causalities are below 0.2. This similarity between directions of causality in a pair could imply a bidirectional relationship between most regions, or could alternatively imply a unidirectional forcing (synchrony) phenomenon. Sliding windows of 200 ms were used, with a sliding step of 50 ms. A library size of 80 was used.

5) Algorithms and Techniques: The machine learning algorithms we use in this experiment are all centered around unsupervised learning, specifically clustering using an altered version of K-means called the Gaussian Mixture model. K-mean algorithms require as input a number n of clusters to find, and a set of data points on which the clusters are to be found. The algorithm then works by choosing n random points in the data set and setting them as centers of the clusters. Each data point that is not a center then becomes in the cluster of the nearest center. The centers are then incrementally reassigned to be the mean of the points in the clusters. This method converges in finite time to a solution, and most of the time (depending on the random initialization) finds good clusters that are well separated and in which each cluster has data points that share common features.

Here, we choose the Gaussian Mixture algorithm as implemented by the Sci-kit learn python library instead of the simple K-means algorithm. Gaussian Mixture Model (GMM) clustering can be thought of as a K-means algorithm that is generalized to account for the covariance of the data and centers of the Gaussian distributions. Because of this feature, while K-means algorithm only clusters data points in spherical shapes, the GMM model allows clusters to take many shapes as described below under the name covariance type. For these reasons, the GMM would be a more suitable algorithm to use in our case. GMM has a few parameters that could

be tuned, the most important of which is the covariance type (full, tied, diagonal, or spherical) which dictates how the cluster edges are expanded and how the clusters grow. This parameter will be fine tuned using grid search.

Furthermore, we use Principal Component Analysis (PCA) to reduce the dimensionality of the dataset. This reduction is important because we have 132 features, and reducing them would result in a simpler analysis of the dataset. PCA works by finding the n axes of most variance in the dataset and transforming the dataset into a space of only these axes. This essentially describes the dataset in a new, lesser set of dimensions than the original dimensions.

Another algorithm that is used is Grid Search, used for fine tunning the Gaussian Mixture model. The grid search algorithm is much simpler than the previous algorithms, since all it does is try to create a model with different parameters and scores each model using a given scoring function. The grid search object then ranks the models based on their scores, and can be used to retrieve the best parameters (the ones that produce the highest score).

6) Metrics and Benchmark: The unsupervised learning used here along with the high dimensionality of the dataset used in this experiment bring with them a unique difficulty in terms of defining a clear scoring method. In the python library we used for analysis (scikit-learn), there are only two methods that can be used to give a score to clusters (assuming no knowledge of *true* labels of each data point): Silhouette Coefficient and Calinsky-Harabaz Index. Both scores ascertain the quality of clustering by measuring how dense, well defined, and separated the clusters are. We are forced to use these metrics due to the nature of the data we are given, as we don't know the true labels of the data points. However, we do know the specific times of when epilepsy was induced, and since epilepsy is *one* state in the brain, then we can use the time indicies for epilepsy to be the one cluster we perform an F1 metric on. The F1 score is calculated as

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

$$precision = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$recall = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

where positive is epilepsy and negative is non-epilepsy.

Since this method has never been attempted before (using Sugihara Causality graphs to identify epilepsy), providing a succinct and complete benchmark is a difficult task. Furthermore, the clustering here is used to explore whether or not there are any extractable information from causal networks in the brain, and whether the brain communicates differently when it is suffering from epilepsy compared to when it is operating normally. Keeping this in mind, we rely on the simple F1 score to provide a simple reliability measure of how well the clustering is separating and predicting the epilepsy cluster. Since we know that epilepsy doesn't happen often in our dataset (only 20 seconds of epilepsy related events in our 500 seconds dataset), we can assign the cluster with the least data points as the epilepsy one, and then measure how exclusive the epilepsy cluster is to the times when epilepsy was *actually* happening by measuring an F1 score. Since our epilepsy happens for about 4% of the dataset, then the baseline score we expect should be 96% of accuracy and 0.99 F1 score. These scores are attained if all the data points were labeled as one (non-epilepsy) cluster.

C. Implementation

1) Data Preprocessing: Some of our edge weights had some Not A Numerical (*NaN*) values, and those were altered to 0, which is the lowest ρ score possible. Once that error was corrected and the data has been put in the graph series format, separated into the three different datasets, we perform Principal Component Analysis (PCA) to reduce the dimensionality of the datasets. This both gives us insight into what components of the graph are logically important for the model and helps decrease the time required to run our model. The results of PCA was used to create three different datasets from each dataset: a dataset that covered 80% of the variance, a dataset that covered 90% of the variance, and a dataset that covered 99% of the variance. This was done to test how dimensionality reduction would affect the final clustering efforts. Furthermore, there were some edges that showed a consistent weight of 0 (for example, all edges from node 2 to node 1 were of weight 0), and having that redundant data is not helpful to distinguishing states. PCA would help us remove such unvarying features from the datasets.

If the scoring for the reduced datasets with only 80% of the variance is similar to the scoring of the reduced datasets with 99% of the variance, then we use the less dimensional reduced datasets even though they give us a less accurate version of the original information. The original dimensionality of the datasets was 132 features per data point, and a list of the PCA datasets and their different variance levels with component numbers are reported in table III. After the

principal component analysis was done on the datasets, the components required to reach the threshold levels of 80, 90, and 99 percents were recorded in table IV (Fig. 22).

TABLE III: Dataset names for the result of PCA analysis. Three different reduced datasets are produced each of which keeps a certain percentage of the original variance. Since we started with three datasets and created three new ones out of each by applying PCA with different levels of components, we arrive at 9 total reduced datasets.

	80% variance	90% variance	99% variance
dataset 50	reduced 50 80	reduced 50 90	reduced 50 99
dataset 200	reduced 200 80	reduced 200 90	reduced 200 99
dataset 250	reduced 250 80	reduced 250 90	reduced 250 99

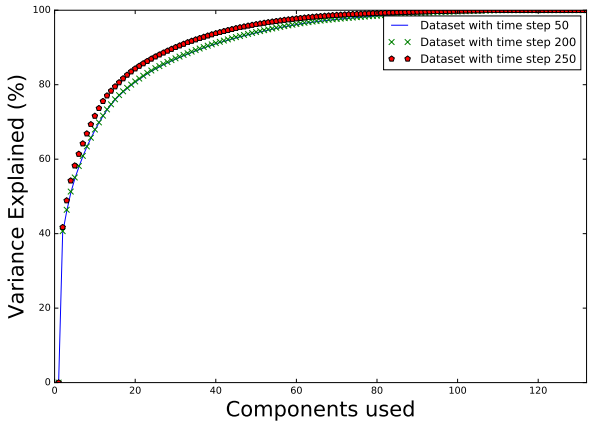


Fig. 22: Principle Component Analysis conducted on the three datasets reveals that the overall variance of the data can be maintained by using only a subset of the edges of the graph rather than all of them. The dataset with a time step of 250 ms needs less components to explain the variance than the other two, while the datasets with 200 and 50 ms time steps showed an almost exact level of variance explained with the number of components. This suggests that the 250 ms dataset is not complex enough, which might indicate that 2000 ms is too big of a time frame to record the granular communication speed of the brain.

TABLE IV: Different component numbers are needed for each dataset to maintain a certain level of variance explained using PCA for dimensionality reduction. These values for components required for reaching a score standard were used to produce reduced forms of the datasets as noted in table III. These reduced forms of the datasets were then analyzed in figure 23.

	80% variance explained	90% variance explained	99% variance explained
Components in dataset 50	20	38	87
Components in dataset 200	20	37	87
Components in dataset 250	16	30	76

2) Clustering: Once the dimensionality of the data was reduced with PCA, a Gaussian Mixture clusterer was initiated for each dataset from table III, and all number of clusters from 2 to 40 were tried and tested by the Silhouette Coefficient and Calinsky-Harabaz Index scorers (Figs. 23, 24).

The complete ordered list of procedures was:

- Removing NaN's with 0's if any exist in the edge weights
- Splitting the dataset into three datasets depending on the time step used
- Vectorizing the graphs so that each graph is a data point
- Running PCA analysis on the three datasets with varying degree of variance kept, creating 9 total datasets
- Clustering the PCA-reduced datasets with cluster numbers from 2 to 40.
- Computing Silhouette and Calinsky scores on the clusters in the previous step

The procedures were implemented successfully, but computing the silhouette and Calinsky scores was a bottleneck as it required calculating the distances between each data point. This could possibly have been avoided by pre-calculating the distances between all data points and passing them to the silhouette and Calinsky scoring functions, a feature available in the sci-kit learn library.

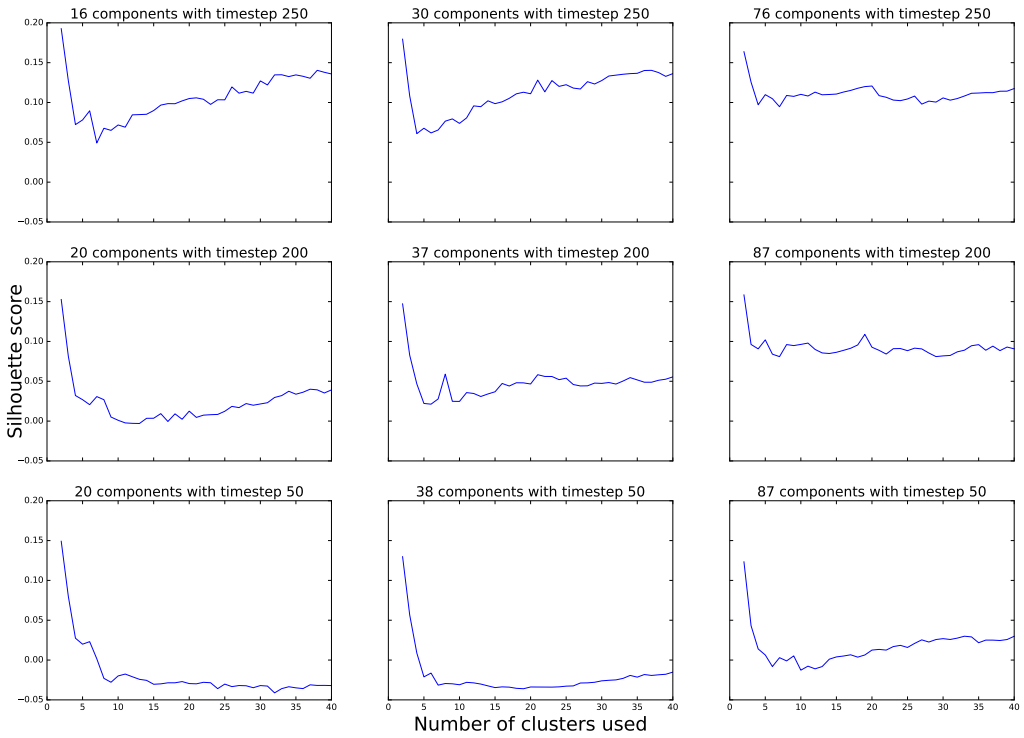


Fig. 23: Silhouette scores for clustering with different cluster numbers. Scores close to 1 suggest clusters that are dense and well separated; scores close to -1 suggest clusters that overlap and are not dense. The best score is consistently when there are only two clusters, and dips greatly when creating more clusters but begins to rise after 30 clusters. With the dataset 250 and dataset 50, PCA with less components (16 and 20 components respectively) showed a better silhouette score than the other, higher component reduced datasets. However, dataset 200 showed a better silhouette score when more components were used. Especially with a rather thin margin of difference in scores between components, it is difficult to conclude whether having more PCA reductionist approach is beneficial.

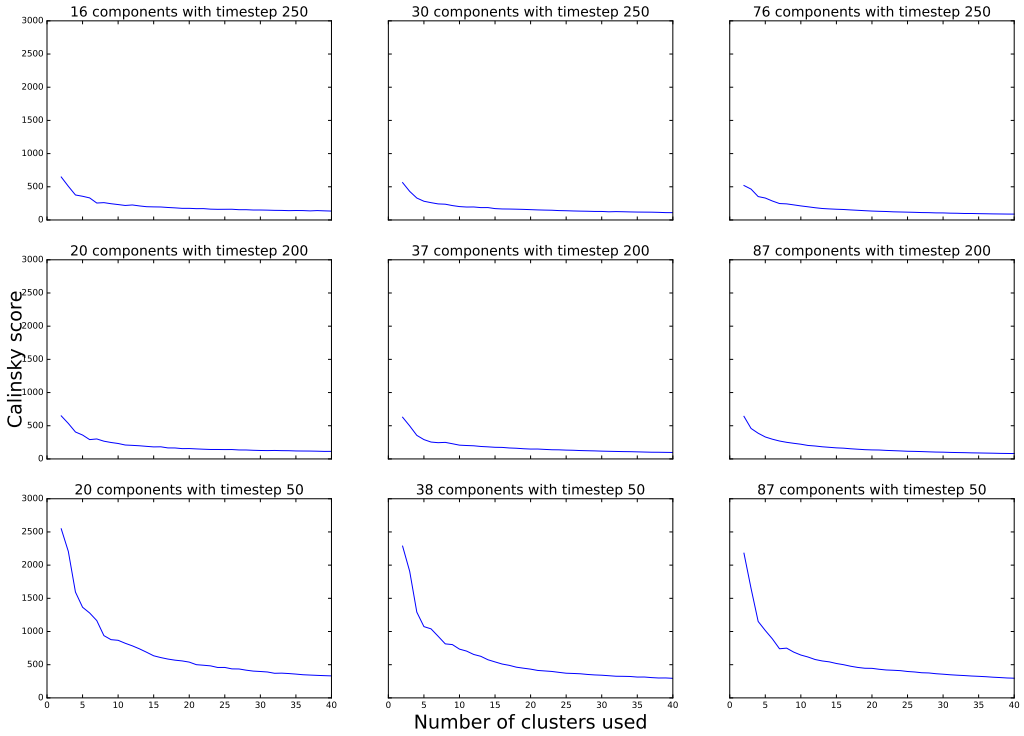


Fig. 24: Calinsky-Harabaz scores for clustering with different cluster numbers. Higher scores mean more separated and dense clusters, which is positive. The score dips greatly when creating more than two clusters, but begins to rise after 30 clusters. The silhouette scores show that having more components when reducing with PCA results in better clustering scores. The scores also show that scores for the reduced data that keep 99% of the variance (right column) are all similar in score and trend.

From figures 23 and 24, it can be seen that the best clustering number is two, and it happens with dataset *reduced 50 80* (dataset that is made with time step of 50 and transformed with PCA). We therefore fine tuned a Gaussian Mixture model with the *reduced 50 80* dataset using Grid Search as implemented by the scikit learn library. The silhouette score was used as a scoring function for the grid search performance measurement. The parameters tuned with grid search was the *covariance type* (with a choice of tied, diagonal, sphere, or full), and the best parameter was 'tied' with a silhouette score of 0.254. This is an increase of 0.05 in the silhouette score, which is a 25% improvement.

D. Results

The results so far indicate that the rat's brain showed the presence of two distinct brain states, as we can cluster the brain's communication network into two clusters. This could be good news, since we can assume that the two brain states are *epilepsy state* and *normal state*. This simplifies our choice of metric to evaluate the model since we can use a simple F1 measurement by assuming the less frequent brain state is the epileptic state, and record the less occurring cluster as the epileptic cluster; we then test the time indicies of this cluster with the time indicies we were given for when epilepsy was evoked. Since we know that the epilepsy happened between seconds 447-457 and 250-260, we set true labels to be 0 for all data points that were in that time period, and 1 for all other data points. We then set the less frequent cluster to be of label 0 (epilepsy cluster) and the more frequent cluster to 1 (non-epilepsy cluster). Our evaluation metric then becomes the F1 score of the predicted labels against the true labels that we produced given the knowledge of our experiment.

Using the fine-tuned model, we achieve an F1 score of 0.788, which seems acceptable at first, but is rather disappointing when considering that the blind benchmark discussed in section V-B6 has a score of 0.99. This is mainly due to the fact that the epilepsy state is infrequent, and so a clustering of non-epilepsy state as epilepsy creates a great error. This indicates that our model does not perform better than blindly guessing that there is only one state in the brain. This does not align with our hopes and expectations for the clustering solution.

To analyze what our model is doing, we plot an annotated time strip that shows what each time window of causality was clustered to (Fig. 25). From this graph, it can be seen that there is no correlation between what we observe in the real-life experiment and our clustering on the graph series constructed from Sugihara Causality measures.

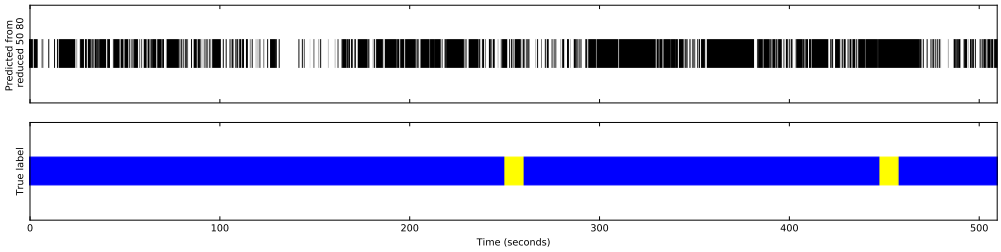


Fig. 25: Time strips that show the epilepsy time track and the results of the respective clustering done on the Sugihara Causality graphs from those time windows. Blue corresponds to true non-epileptic state; yellow to true epileptic state; white to predicted epileptic state; and black to predicted non-epileptic state. There are only two clusters in the top bar. The clustering results does not show any correlation with the actual epilepsy induction time series.

E. Discussion

Initial analysis shows that the causality network is very dense with highly weighted edges. The high density of the graph could have been a side effect of Sugihara's model ability to detect downstream causality. If that is the case, then many of the causal connections detected could in fact be residuals of upstream interactions in the brain network. Put in another way, if region A of the brain was causing region B to fire, and region B causes region C to fire, then the Sugihara causality model would report a causality measure between A and C. Downstream causality measures can be detected by observing both the magnitude of the cross map skill as well as the time lag that produces the greatest cross map skill (Fig. 3 in [41]). Although we could have used this method in our experiment to remove the possibility of picking up extra weights from downstream causality, initial trials showed that the computational power required for that analysis were greater than the capacity available at UCCS.

Concerning the high possibility of the presence of unidirectional forcing from figure 21, Ye *et al.* showed unidirectional forcing can be untangled by inspecting the greatest time lag of the two that produces the highest causality measure (Fig. 2 in [41]). In order to allay the problem of unidirectional forcing, the best lag of each pair is considered. This is a tricky problem because there is no clear range for which to test the lag. This is because the time delay for neuronal activity is yet studied, and how that translates to EEG data could be tricky. We reserve the use of this method due to its computational complexity which would add to the already high time complexity of the analysis.

The theoretical implications of this model could present a novel representation of information flow in the brain and determining causality within the brain. If the graph output of this method id reliable, it could help outline information flow within the brain, much like one would observe in a magnetic wave flowing through an fMRI recording.

F. Future Work

Since various improvements can be made to this work, we break it down by stage of implementation.

1) Methods: It would have been enriching to our knowledge of our true labels if we had access to exactly what the rat was doing during the experiment. A video capture of the rat, or recorded structured interventions to the rat's behavior could have provided more opportunity to label the data with ground truth. For example, we could have shone light into the rat's eyes, and recorded that as a separate brain state; we could have put the rat on a hamster wheel and recorded his running as a separate brain state; eating food could be a brain state; and sleeping is another possible brain state. All these interventions to the rat's environment could enable us to gain more labels rather than the simple epilepsy, non-epilepsy labeling system we worked with. Although this complicates our approach, it would allow us access to better scoring metrics other than the silhouette and Calinksy-Harabaz scores, where the labels have to be known.

2) Sugihara Causality Network: Future work in this area could focus on verifying this model through controlled experiments. Such experiments could be in the form of stimulating a part of the brain (e.g. shining strong light on an the eye to excite the visual cortex) and observing the model's behavior. One would expect a high value of out degrees from the specific region during such an experiment, as it attempts to convey a considerable portion of information to the rest of the brain. Moreover, a clear proof should be presented as to what the most reliable time window and step size ought to be when producing the causality graphs. Such a task can be done by measuring graph similarity of the same time segment as the time window gets shortened incrementally.

3) Data Processing: Apart from collecting the data and applying Sugihara Causality on the dataset (which took 2 days), calculating the current source density using kCSD was by far the most time consuming process. Even though we do not discuss the details of how that was done in this project (we provide already processed kCSD data), it would have been nice to see a GPU implementation of the kCSD, which took advantage of CPU parallelism but not GPU

parallelism. This would massively improve the speed of the implementation and allow for testing more various approaches in the Sugihara Causality step.

4) Cluster Scoring with Higher Dimensions: Due to the high dimensionality of the dataset we worked with, it was difficult to visually measure how well the clustering with different number of clusters took place. Beyond three dimensions, it is impossible to visualize the dataset with labels, and although the PCA reduction resulted in a greatly reduced dimensional feature, it was not enough to allow us to visualize the clusters visually. Following the assumption that downstream causality decreases in magnitude as it travels downstream in the network, we can use this to traverse the graph and rid of any paths that decreases in causality as it goes downstream.

G. Conclusion

The experiment we conducted transformed 32 EEG electrode signals into 12 regional neuronal activity signals, and created a (theoretical) communication network from those signals using Sugihara Causality. These communication graphs were analyzed by unsupervised learning using the Gaussian Mixture model for the goal of gaining some insight into the epileptic state, and test whether unsupervised clustering is able to discriminate between brain states from the communication networks. From our initial results, it is clear that our approach is not successful in distinguishing brain states in a rat's brain. However, the ground work has been laid for future explorations in the topic of communication networks in the brain.

VI. Conclusion

The paper shows promising initial results using the Sugihara CCM model to construct causality graphs between brain regions. We find that the brain network for this experiment is highly causal with a range of time windows. This, however, could be due in part to experimental design limitations, where the electrodes were 1 mm apart which might have caused electrical interference. To to limit this phenomenon we used kCSD to preprocess the data. Initial results show a time varying graph in which information flow can be tracked. At the moment, more analysis is required to make a conclusions on the capabilities of the pairwise causality graph model. Some of most incurring difficulties to overcome are the running complexity of the kCSD preprocessing and CCM algorithm required for the analysis of the amount of pairs in a large network, and the mathematical representation of information flow within the time-dependent graph. We plan on using the Neuroclustering algorithm to discretize the EEG data into epileptic seizures, extract causal network features from the stages, and train a k-means learning algorithm on the created feature set. The implications of these findings could relate more generally to discoverability of causality in modeling scalable natural phenomena. Real world applications manifest in localization of epilepsy in the brain. Furthermore, if the technique of distinguishing causal networks in systems and clustering their properties is successful, it could be a clear indications that the Sugihara causality model is able to detect causation in extremely complex systems comparable to the human brain.

Acknowledgment

Special thanks goes to Dr. Somogivari and Dorrottya R. from Winger RCP, Budapest for providing the neuronal data used in this paper. Furthermore, Dr. Peter Erdi is thanked for his patient guidance of this research topic which was instrumental for the production of this paper.

References

[1] R. Schapire, “Theoretical machine learning,” 2008.

[2] Y. M. Assael, B. Shillingford, S. Whiteson, and N. de Freitas, “Lipnet: Sentence-level lipreading,” *arXiv preprint arXiv:1611.01599*, 2016.

[3] A. C. F. H. Lucas Theis, Wenzhe Shi, “Lossy image compression with compressive autoencoders,” *OpenReview.net*, 2016.

[4] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin, “Accelerating eulerian fluid simulation with convolutional networks,” *arXiv preprint arXiv:1607.03597*, 2016.

[5] S. U. Brian Kenji Iwana, “Judging a book by its cover,” *arXiv preprint arXiv:1610.09204*, 2016.

[6] R. S. Fisher, C. Acevedo, A. Arzimanoglou, A. Bogacz, J. H. Cross, C. E. Elger, J. Engel, L. Forsgren, J. A. French, M. Glynn, D. C. Hesdorffer, B. Lee, G. W. Mathern, S. L. Mosh, E. Perucca, I. E. Scheffer, T. Tomson, M. Watanabe, and S. Wiebe, "Ilae official report: A practical clinical definition of epilepsy," *Epilepsia*, vol. 55, no. 4, pp. 475–482, 2014. [Online]. Available: http://dx.doi.org/10.1111/epi.12550

[7] P. Shafer and J. Sirven. (2013) New terms and concepts for seizures and epilepsy. [Online]. Available: http://www.epilepsy.com/learn/types-seizures/new-terms-and-concepts-seizures-and-epilepsy

[8] P. Shafer and J. I. Sirven. (2010) Epilepsy statistics. [Online]. Available: http://www.epilepsy.com/learn/epilepsy-statistics

[9] K. Allers, B. M. Essue, M. L. Hackett, J. Muhunthan, C. S. Anderson, K. Pickles, F. Scheibe, and S. Jan, "The economic impact of epilepsy: a systematic review," *BMC Neurology*, vol. 15, no. 1, p. 245, 2015. [Online]. Available: http://dx.doi.org/10.1186/s12883-015-0494-y

[10] A. C. Vivas, A. A. Baaj, S. R. Benbadis, and F. L. Vale, "The health care burden of patients with epilepsy in the united states: an analysis of a nationwide database over 15 years," *Neurosurgical focus*, vol. 32, no. 3, p. E1, 2012.

[11] K. S. Walia, E. A. Khan, D. H. Ko, S. S. Raza, and Y. N. Khan, "Side effects of antiepileptics a review," *Pain Practice*, vol. 4, no. 3, pp. 194–203, 2004. [Online]. Available: http://dx.doi.org/10.1111/j.1533-2500.2004.04304.x

[12] H. Weiner and J. Sirven. (2013) Surgery. [Online]. Available: http://www.epilepsy.com/learn/treating-seizures-and-epilepsy/surgery

[13] E. E. Bailey, H. H. Pfeifer, and E. A. Thiele, "The use of diet in the treatment of epilepsy," *Epilepsy & Behavior*, vol. 6, no. 1, pp. 4–8, 2005.

[14] K. W. Barañano and A. L. Hartman, "The ketogenic diet: uses in epilepsy and other neurologic illnesses," *Current treatment options in neurology*, vol. 10, no. 6, pp. 410–419, 2008.

[15] E. B. E. B. Online., "Artificial intelligence (ai)," 2016 25 Sep.

[16] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[17] M. Minsky and S. Papert, "Perceptrons." 1969.

[18] N. Yadav, A. Yadav, and M. Kumar, *History of Neural Networks*. Dordrecht: Springer Netherlands, 2015, pp. 13–15.

[19] T. M. Mitchell, "Machine learning," *Mc G raw ill.(forthcoming)*, 1997.

[20] D. J. Finney, "A note on the history of regression," *Journal of Applied Statistics*, vol. 23, no. 5, pp. 555–558, 1996.

[21] J. Locke, *An essay concerning human understanding*, 1841.

[22] G. Berkeley, *A treatise concerning the principles of human knowledge*. Philadelphia: JB Lippincott & Company, 1874.

[23] C. W. J. Granger, "Investigating causal relations by econometric models and cross-spectral methods," *Econometrica*, vol. 37, no. 3, pp. 424–438, 1969.

[24] G. Sugihara, R. May, H. Ye, C.-h. Hsieh, E. Deyle, M. Fogarty, and S. Munch, "Detecting causality in complex ecosystems," *science*, vol. 338, no. 6106, pp. 496–500, 2012.

[25] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of the atmospheric sciences*, vol. 20, no. 2, pp. 130–141, 1963.

[26] G. S. Paul A. Dixon, Maria J. Milicich, "Episodic fluctuations in larval supply," *Science*, vol. 283, no. 5407, pp. 1528–1530, 1999.

[27] E. R. Deyle and G. Sugihara, "Generalized theorems for nonlinear state space reconstruction," *PLoS One*, vol. 6, no. 3, p. e18295, 2011.

[28] F. Takens, *Detecting strange attractors in turbulence*. Springer, 1981.

[29] F. Villatoro. (2012) Takens' theorem in action for the lorenz chaotic attractor. [Online]. Available: https://www.youtube.com/watch?v=6i57udsPKms

[30] E. R. Deyle, M. Fogarty, C.-h. Hsieh, L. Kaufman, A. D. MacCall, S. B. Munch, C. T. Perretti, H. Ye, and G. Sugihara, "Predicting climate effects on pacific sardine," *Proceedings of the National Academy of Sciences*, vol. 110, no. 16, pp. 6430–6435, 2013.

[31] X. Wang, S. Piao, P. Ciais, P. Friedlingstein, R. B. Myneni, P. Cox, M. Heimann, J. Miller, S. Peng, T. Wang, H. Yang, and A. Chen, "A two-fold increase of carbon cycle sensitivity to tropical temperature variations," *Nature*, vol. 506, no. 7487, pp. 212–215, 02 2014.

[32] J. C. McBride, X. Zhao, N. B. Munro, G. A. Jicha, F. A. Schmitt, R. J. Kryscio, C. D. Smith, and Y. Jiang, "Sugihara causality analysis of scalp eeg for detection of early alzheimer's disease," *NeuroImage: Clinical*, vol. 7, pp. 258–265, 2015.

[33] E. H. van Nes, M. Scheffer, V. Brovkin, T. M. Lenton, H. Ye, E. Deyle, and G. Sugihara, "Causal feedbacks in climate change," *Nature Climate Change*, vol. 5, no. 5, pp. 445–448, 2015.

[34] A. A. Tsonis, E. R. Deyle, R. M. May, G. Sugihara, K. Swanson, J. D. Verbeten, and G. Wang, "Dynamical evidence for causality between galactic cosmic rays and interannual variation in global temperature," *Proceedings of the National Academy of Sciences*, vol. 112, no. 11, pp. 3253–3256, 2015.

[35] J. M. McCracken and R. S. Weigel, "Convergent cross-mapping and pairwise asymmetric inference," *Physical Review E*, vol. 90, no. 6, p. 062903, 2014.

[36] A. T. Clark, H. Ye, F. Isbell, E. R. Deyle, J. Cowles, G. D. Tilman, and G. Sugihara, "Spatial convergent cross mapping to detect causal relationships from short time series," *Ecology*, vol. 96, no. 5, pp. 1174–1181, 2015.

[37] A. Wismüller, X. Wang, A. M. DSouza, and M. B. Nagarajan, "A framework for exploring non-linear functional connectivity and causality in the human brain: Mutual connectivity analysis (mca) of resting-state functional mri with convergent cross-mapping and non-metric clustering," *arXiv preprint arXiv:1407.3809*, 2014.

[38] T. Hjornevik, T. B. Leergaard, D. Darine, O. Moldestad, A. M. Dale, F. Willoch, and J. G. Bjaalie, "Three-dimensional atlas system for mouse and rat brain imaging data," *Frontiers in neuroinformatics*, vol. 1, p. 4, 2007.

[39] G. Paxinos, C. Watson, P. Carrive, M. Kirkcaldie, and K. Ashwell, "Chemoarchitectonic atlas of the rat brain," 2009.

[40] J. Potworowski, W. Jakuczun, S. Łski, and D. Wójcik, "Kernel current source density method," *Neural computation*, vol. 24, no. 2, pp. 541–575, 2012.

[41] H. Ye, E. R. Deyle, L. J. Gilarranz, and G. Sugihara, "Distinguishing time-delayed causal interactions using convergent cross mapping," *Scientific reports*, vol. 5, 2015.