Michael Lengel

IS550

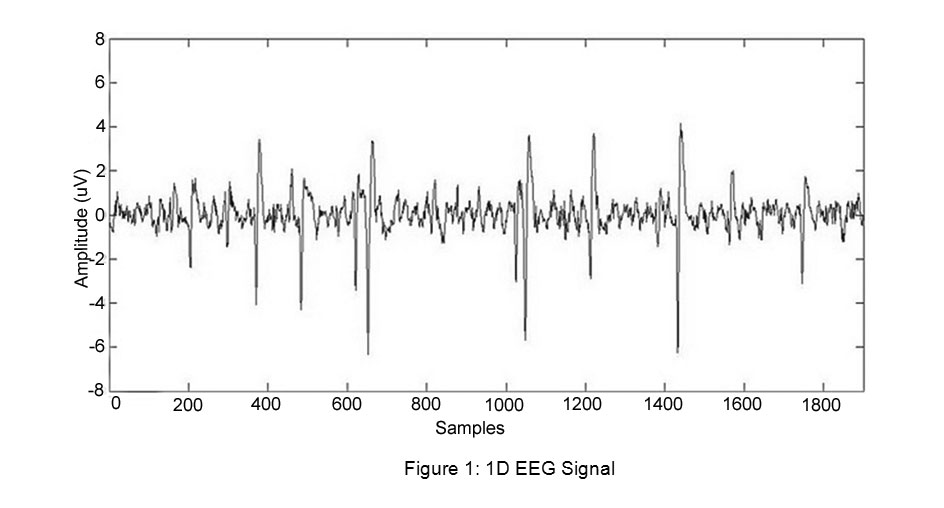
Final Project

7/18/2018

Predicting Schizophrenia Through EEG Signals

Project Background:

Brain cells talk to each other through electrical impulses. An electroencephalogram (EEG) is a test that is used to detect this electrical activity in your brain. As a result, an EEG can often be useful in detecting any anomalies or problems with this activity. The EEG uses electrodes that are attached to your scalp in order to collect the signal that corresponds with the electrical impulses in your brain. These electrodes are hooked up to a computer which records the timestamp of the signal and the magnitude of the signal. (1) A normal EEG signal is riddled with both noise as well as sharp increases (peaks) and sharp decreases (valleys). An example of an EEG Signal has been listed below for reference.



In this particular project I will be aiming to classify patients who undergo an EEG exam into two different groups based on a number of features. I will go into more detail about the features in the dataset portion of my report. It is a binary classification problem where the target variable is whether or not a patient is schizophrenic. Schizophrenia is a mental disorder, with its most common symptoms being auditory hallucinations, false beliefs, and confused thinking (2).

The patients in the dataset I am using for analysis have been exposed to a certain experiment. During the experiment each of the patients is given three different trials. Each trial last 3 seconds, and EEG signals are captured during the entirety of these trials. The first trial the patients press a button which generates an auditory tone. The second trial the patient will listen to the same tone without pressing the button. The last trial the patient presses the button but there is no auditory tone generated. This third trial is used as type of control trial. The general idea behind this experiment is since schizophrenic patients have difficulty distinguishing between internal (auditory hallucinations) and external sounds (the generated tone) there should be a pattern or difference in their EEG signals vs. the EEG signals of a patient without schizophrenia (2). This is something I intend to put to the test using a RNN in order to attempt to predict which patients are schizophrenic based on the EEG signals are every time stamp. The value-add of a successful implementation of this project would be significant. If an accurate model can be trained it’s possible that doctors could give this experiment to a patient, feed their data into the neural net, and have it output a predicted diagnosis to serve as an input for doctor’s who would make a diagnosis based on both the model’s output and any other information they have available to them.

The Dataset:

As previously mentioned this project is a binary classification, with a target variable of a one or a zero. A one represents the patient is schizophrenic, and a zero will represent the patient is not-schizophrenic. In addition to this the dataset contains metadata about the patients such as age, gender, and number of years of education. In addition to this metadata, the dataset also contained nine different EEG signals from electrodes placed on different areas of the patient’s skull. It’s important to note that the dataset which contained the demographics metadata was in a separate .csv file from the dataset that contained the EEG signals. As a result- the metadata had to be left joined to EEG dataset using a primary key of patient number. The EEG Signals data contained 81 patients, each with 3 second trials. The EEG signal at each of the 9 different locations was captured every millisecond. As a result there were roughly 3000\*3\*81 (~730K) rows in the final dataset.

The Model:

As previously discussed the model I selected to analyze this dataset was a Recurrent Neural Network or RNN. My particular dataset is a sequenced dataset which RNN’s are particularly well equipped to handle in comparison to more traditional machine learning models. The results from previous time stamps are taken into account when addressing the current timestamp that is being fed through the model.

More specifically a Long Short Term Memory or LSTM RNN was chosen. A LSTM has significant advantages over the standard Recurrent Neural Network. (3) While a regular RNN has the ability to take only recent timestamps into account making future predictions, a LSTM has the ability to take into account timestamps that are further in the past. This is particularly useful in my project, because if the model picks up a certain pattern that is associated with a patient being schizophrenic in the first trial, it needs to be able to remember that and take it into account when making predictions for that same patient in the third trial. This is where the Long Memory acronym comes from in LSTM.

Now that I’ve explained the LSTM model and the justification of using it for this application I will go into further detail regarding the model specifics. Just like any machine learning problem the dataset needs to be split into both training and test subsets. Unlike a typical machine learning problem where you may be able to feed the data into sci-kit learns train\_test\_split function the dataset needed to be split in a more nuanced way. The challenges with more typical splitting methodologies is it could potentially separate data from the same patient into both the train and test split. Since a RNN is dependent on all of the data points in the sequence this would be problematic for this application. Instead random numbers needed to be generated to decide which patients would be contained in the training data and which would be contained in the testing data. I used around a 75/25 training test ratio. That meant ~75% of the data, or 61 patients, were contained in the training set while 25%, or 20 patients, were contained in the set used for testing. Something that’s important to note is to feed your data into a LSTM model it needs to be in a 1 by 3 shape. The first element representing the number of unique patients, the second number representing the length of the sequence, and the third element representing the number of features in your dataset. In my case the training data shape was (61,9216,12). Each trial consisted of 3072 ms and there were 3 trials per patient. As a result I felt a max length of 3072\*3 made sense in this application. This way I didn’t lose any data, nor did I need to pad any of the values with 0. I was able to do this because of how clean of a dataset I had. In real life applications you may have to do some more data cleansing or zero padding. In addition, I selected the batch size of 3072 so the LSTM would feed each trial into model while training. I saw through trial and error that it was important to keep the continuity of each trial while training the model or it would have a negative impact on model performance. I experimented with 5, 10, and 15 epochs but while I didn’t hit the point of overtraining where the model decreased in accuracy, I did notice a large diminishing return in the amount of time it took to train vs. the increases in performance I was seeing. I kept the activation function the same because I wanted my model to output a probability (0-1) that the patient was schizophrenic at every time stamp throughout the experiment. Please use the table below as reference.

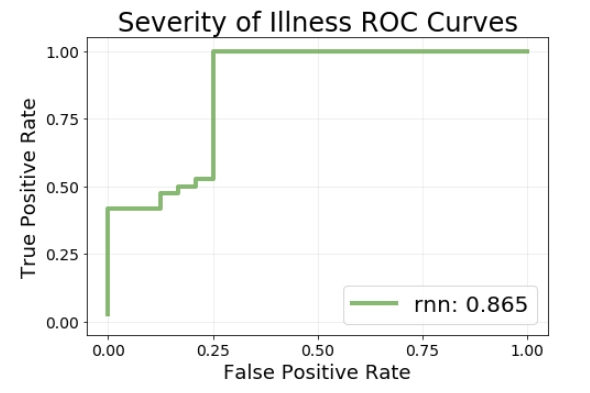
|  |  |
| --- | --- |
| Parameter | Value |
| neurons | 528 |
| batch size | 3072 |
| epochs | 5 |
| learning rate | 0.005 |
| activation | sigmoid |
| optimizer | binary crossentropy |
| dropout\_w | 25% |
| dropout\_u | 10% |

Results:

As I previously mentioned, for each line in the test dataset I wanted to return the probability that the patient is schizophrenic. It’s important to have this prediction be returned every timestamp because as the patient is undergoing the different trials the model may recognize a certain pattern during the course of the experiment. Since it is a LSTM model this model may remember said pattern when classifying data from the same patient but further in the future. In the table shown below you can see for almost every subject in the test set each patient was at one point throughout the three trials classified as both with schizophrenia and without schizophrenia. Seeing as the classification may change throughout the course of the experiment as the model gets more data on the patient, in order to make the classification it’s important to take the model’s predicted probability of schizophrenia at the last prediction of the last trial. By taking the last prediction you are ensuring the model has made its choice based on all of the information available to it & as a result will be the most accurate prediction.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| subject | min pred | max pred | final prediction | classification | accuracy |
| 4 | 0.222 | 0.642 | 0.485 | 0 | correct |
| 8 | 0.188 | 0.644 | 0.486 | 0 | correct |
| 12 | 0.207 | 0.636 | 0.496 | 0 | correct |
| 16 | 0.171 | 0.646 | 0.486 | 0 | correct |
| 20 | 0.213 | 0.647 | 0.487 | 0 | correct |
| 24 | 0.263 | 0.658 | 0.496 | 0 | correct |
| 28 | 0.382 | 0.647 | 0.554 | 1 | correct |
| 32 | 0.404 | 0.648 | 0.545 | 1 | correct |
| 36 | 0.387 | 0.683 | 0.580 | 1 | correct |
| 40 | 0.504 | 0.701 | 0.565 | 1 | correct |
| 44 | 0.471 | 0.744 | 0.589 | 1 | correct |
| 48 | 0.536 | 0.718 | 0.590 | 1 | correct |
| 52 | 0.547 | 0.744 | 0.625 | 1 | correct |
| 56 | 0.585 | 0.762 | 0.621 | 1 | correct |
| 60 | 0.583 | 0.772 | 0.587 | 1 | incorrect |
| 64 | 0.582 | 0.775 | 0.582 | 1 | incorrect |
| 68 | 0.582 | 0.777 | 0.583 | 1 | correct |
| 72 | 0.582 | 0.785 | 0.582 | 1 | correct |
| 76 | 0.591 | 0.798 | 0.598 | 1 | correct |
| 80 | 0.592 | 0.808 | 0.603 | 1 | correct |

As you can observe above, the model produced a good range of predictions- as well as a solid % of correct predictions (18/20). It’s important to note, that the two incorrect predictions are false positives. Here the model is saying the patient is schizophrenic when they are not. This type of inaccuracy in this application is better than a false negative, because in this case it gives the Doctor a chance to review these patients and make the final call. If there were false negatives, the Doctor may never bother to look at patients who actually have schizophrenia because the model says they were not schizophrenic. It’s a very good sign that there were exactly zero false negatives in this model. With these results the precision would be 14/16 or 87.5% while the recall would be 100% or 14/14 due to zero false negatives. The AUC curve can be observed below.



Conclusion:

In conclusion, since the model is producing a high precision, and perfect recall the LSTM model effective for this application of sequenced data. Specifically, the EEG signals produced when putting patients under this experiment seem to have predictive value on whether or not a patient is schizophrenic. This comes as no surprise, as the reason this dataset was published and shared was because the researchers found predictive value in analyzing these variables through a different type of model. My project makes a valuable contribution because a Doctor can now feed the data from future patients undergoing this same experiment through the model to help make an initial determination of which patients might be schizophrenic. With no false negatives the Doctor can reliably use this model as first pass to determine which patients he needs to take a closer look at while making a diagnosis.

References:

(1) Swartz, Barbara E. (1998). "The advantages of digital over analog recording techniques". Electroencephalography and Clinical Neurophysiology. **106** (2): 113–7. [*doi*](https://en.wikipedia.org/wiki/Digital_object_identifier):[*10.1016/S0013-4694(97)00113-2*](https://doi.org/10.1016/S0013-4694%2897%2900113-2). [*PMID*](https://en.wikipedia.org/wiki/PubMed_Identifier) [*9741771*](https://www.ncbi.nlm.nih.gov/pubmed/9741771).

(2)  American Psychiatric Association. Diagnostic and Statistical Manual of Mental Disorders (5th ed.). Arlington: American Psychiatric Publishing. pp. 101–05. [*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [*978-0-89042-555-8*](https://en.wikipedia.org/wiki/Special:BookSources/978-0-89042-555-8).

(3) Li, Xiangang; Wu, Xihong (2014-10-15). "Constructing Long Short-Term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition". [arXiv](https://en.wikipedia.org/wiki/ArXiv):[1410.4281](https://arxiv.org/abs/1410.4281) Freely accessible[[cs.CL](https://arxiv.org/archive/cs.CL)].

Supplementary Objective:

Based on the feedback I received during the in class portion of my presentation- I tried to take my analysis one step further. I was asked to try to come up with a way to forecast the future EEG signals based on the previous EEG values. This is an extremely relevant question that is applicable in many types of modern day forecasting. The best way to accomplish this is by introducing what we refer to as a lagged variable. This lagged variable or variables become a new column or columns in your dataset. The lag part refers to a time/sequence offset you’re introducing to the variable. For instance, in this project I tried various different lengths of lag. I started with a large lag- because this makes it possible for you to predict EEG signal farther into the future, but eventually started to pair this down in order to increase the forecasting performance. Naturally, the further you are looking away from the current timestamp- the hard it will be to predict correctly. For instance, with this project each row was 1 millisecond. At first, I used a lag of 500 ms. This means my model would attempt to predict what the EEG signal would be in 500 ms based on the signals that were happening at that instantaneous moment. Colloquially, this would be referred to as a lag of 500. Doing this required me to re-design the dataset into a shape that had the current EEG signal (target variable) & the EEG signal from 500 ms prior (feature) on the same row.

Another unique part of this problem is the way you have to shift. If you were to look 500 rows above the first row for patient 2, you would be looking at patient 1’s EEG signal. As a result, the lag has to be partitioned by each patient. This creates a unique problem, because at the start of patient 2’s data- there is no signal 500 ms before for us to observe in the dataset. As a result, depending on how big your lag is, you are potentially dropping 81 \* 500 records from the dataset- because they will have no features to predict the target variable. Luckily in a dataset with 750K rows dropping 40K becomes an easier pill to swallow.

Anyways after creating the dataset I adjusted my model accordingly- most notably in the modification of the activation layer. Previously the activation layer was sigmoid which outputs a probability anywhere from 0-1. Now that the target was no longer a classification and was now an unbounded regression with values > 1 and < -1 the sigmoid activation layer no longer made sense. As a result, I had to change my activation function to linear which would return unbounded results from negative infinity to positive infinity. As a result- instead of the model outputting a probability- it was outputting the actual predicted EEG signal.

After making the modifications to my model and testing it I was struggling with performance issues. My first thought to remedy this was to turn the lag down. This is because it’s easier to forecast something when you are forecasting only a day in advance rather than a week in advance or a month in advance. Therefore, I turned the lag down from 500 ms to 250. I was still experiencing the same issues so I again brought it down to 100 and then 50 and then eventually to 10. Once I was still having problems predicting the EEG signal 10 milliseconds in advance I started to brainstorm possible reasons this prediction wasn’t working.

After taking a step back I realized what the most likely reason for the performance issues was. The dataset was comprised of 81 patients over three trials. I already mentioned how this affected the way I create the lagged dataset, but I realized it created an additional problem I hadn’t thought of. Each of the three trials are different. The first trial a tone is played when the patient hits a button, the second trial a tone is played without the patient hitting the button, and the third trial no tone is played even when the patient hits the button. While just labeling which of the three trials did help the model, the most important predictive feature of this exercise was not captured in the dataset. That was the timing of the auditory tone. Since the tone did not start at the same time for each of the patients, and there is no indication of when the tone was generated in the first two trials, or when the button was pressed in the third- we are missing vital information about the most deterministic variables on the EEG Signal. Without those in the dataset, and with the level of noise generated in capturing an EEG signal, it wouldn’t matter how far ahead I was trying to predict the model was going to have a poor performance.

While I was not able to get a solid performing model / decent results I still felt I learned a lot from this exercise. Most specifically, I got good hands on experience in creating a lagged dataset as well as good experience adjusting a RNN’s activation functions based on the investigatory question at hand. In addition, I feel like attempting this has given me a good perspective on the core concepts of machine learning. Machine learning isn’t just about feeding numbers into a model and spitting out an answer. A lot of thought and input from SME’s need to be included when creating a model to figure out if the features you have are likely to be predictive. In the supplementary portion of my project, they were unfortunately not- although it has opened my eyes into be aware of the necessity of this, and is something that I will carry forward with me in my career.