SEMESTER PROJECT

473877-FS2022-0: AI FOR MEDICAL TIME SERIES

Shunyu Wu, Jiahui Yu

I. INTRODUCTION

An electroencephalogram (EEG) is a test that detects electrical activity in people's brain using electrodes attached to their scalp. Human brain cells communicate via electrical impulses and are active all the time, even when they're asleep. This activity shows up as wavy lines on an EEG recording.

The goal of this project is to classify EEG responses when people see different images.

II. DATASET

This dataset consists of 21 subjects, 64 channels, 384 time points. There are 6 classes representing EEG responses to images with different familiarity and pleasantness *(familiar and neutral (FN), familiar and pleasant (FP), familiar and unpleasant (FU), novel and neutral (NN), novel and pleasant (NP), novel and unpleasant (NP) pictures)*. [1] For more detail about the dataset please check here.

III. METHOD

A. Extracting the dataset

The dataset has 22 subjects, each stored as one epoch with (TRIALS, CHANNELS, TIMES) structure. The event id label marked as follows:

Label	FN	FP	FU	NN	NP	NU
ID	0	1	2	3	4	5

B. Train a classifier

The goal of the project is to train a classifier for the above dataset that can classify Familiar (FN, FU and FP) and Novel (NN, NP and NU). Many different kinds of

algorithms can be consider to achieve the goal, such as support vector machine, neural network, decision tree, naive Bayes, k-NN, etc. But for time series data the input features are not independent, therefore SVMs and Naive Bayes would not be a good choice since they assume that the input features are independent. The k-NN algorithm could still work, and we measure the similarity between two time series using dynamic time warping. The second classification model is neural network(RNN), and it performs much better on large data set as in this task.

C. Evaluate performance

We will evaluate the performance of our model via Accuracy, Precision, Recall & F1 Score.

TABLE I: Evaluation Method

	True Values		
Predicted Values		Positives	Negatives
	Positives	True positive	False positive
	Negatives	False negative	True negative

$Accuracy = \frac{TP+TN}{TP+FN+FP+FN}$, $Precision = \frac{TP}{TP+FP}$
$Recall = \frac{TP}{TP+FN}$, $F1\text{-}score = \frac{TP}{TP+\frac{1}{2}(FP+FN)}$

And for multi-classification problem, we use the AUC *(Area Under The Curve)* ROC *(Receiver Operating Characteristics)* curve.

$TPR/Recall/Sensitivity = \frac{TP}{TP+FN}$
$Specificity = \frac{TN}{TN+FP}$
$FPR = 1 - Specificity = \frac{FP}{TN+FP}$

ROC is a probability curve and AUC represents the degree or measure of separability. The ROC curve is plotted with TPR against the FPR where TPR is on the y-axis and FPR is on the x-axis.

IV. Classification I

A. Data Process

Since the EEG data set is very huge and DTW algorithm performs bad at long time series. I just average the channels of one subject to reduce the data size. This is bad implementation(should have used PCA to eliminate redundant features) but even so

I cannot run all 21 subjects on my computer. 2 subjects of data took nearly an hour, so all work below just show an idea.

In order to classify familiar vs novel, new event label 1 to familiar and 0 to novel are reassigned manually. Different epochs of data are combined together to produce one numpy array and the event labels are added to the last column. As a result generating a 2D array of dimension $row \times (384 + 1)$, where row is the sum of all trials of included subjects.

The data set is then split into *train* and *test* using *train_test_split* from *sklearn*, in which 33% will be test set and 67% will be train set.

B. Dynamic Time Warping

Dynamic time warping finds the optimal non-linear alignment between two time series. It is much reliable than euclidean distances which is suspicious due to distortion in the time axis. However, it has a very high time cost because DTW running time is quadratic to the time series's length.

Consider two time series X and Y of the same length n where $X = x_1, x_2, ..., x_n$ and $Y = y_1, y_2, ..., y_n$. Then we construct an $n \times n$ matrix whose i, j^{th} element is the *Euclidean distance* between x_i and y_j. What we want is to find a path through this matrix that minimizes the cumulative distance. This path then determines the optimal alignment between the two time series.

Let path W be $W = w_1, w_2, ..., w_K$ where each element of W represents the distance between a point i in X and a point j in Y i.e. $w_k = (x_i - y_j)^2$. And we want to know $W^* = argmin_W(\sqrt{\sum_{k=1}^{K} w_k})$.

The optimal path is found via dynamic programming using this recursive function. $\gamma(i, j) = d(x_i, y_j) + min(\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1))$

1) Locality constraint: DTW has a complexity of $O(nm)$ where n and m are the lengths of two time series. Since we are performing dynamic time warping multiple times on very long time series data, this can be extremely expensive. However, it is intuitive that if the two points are far apart they are unlikely to match. In order to reduce running time, we can enforce a window size w, so that only mappings within this window are considered and thus speeds up the inner loop.

2) LB Keogh: LB Keogh lower bound can also speeds up dynamic time warping. It is defined as $LBKeogh(X, Y) = \sum_{i=1}^{n}(y_i - U_i)^2 I(y_i > U_i) + (y_i - L_i)^2 I(y_i < L_i)$, where U_i and L_i are upper and lower bounds for time series X which are defined as $U_i = max(x_{i-r} : x_{i+r})$ and $L_i = min(x_{i-r} : x_{i+r})$ for a reach r and $I(\cdot)$ is the indicator function.

C. k-NN algorithm

We use the k-NN algorithm for classification and choose $k = 1$ empirically. And we manually appended ***familiar: 1*** and ***novel: 0*** labels to *train* and *test* dataset to fit the function input . Although we tried the above two methods to reduce time cost, the dataset is still too big for our own computer to run. So we just classified two subjects... The window size is set to 4. And the ratio between *train* and *test* dataset size is 0.67 : 0.33.

TABLE II: classification report

	precision	recall	f1-score	support
novel: 0	0.24	0.24	0.24	91
familiar: 1	0.79	0.78	0.78	322
accuracy			0.66	413
macro avg	0.51	0.51	0.51	413
weighted avg	0.66	0.66	0.66	413

1) Result: As shown above using just 2 subjects data and average channels is not a good practice, the resulting accuracy is only 66%. But it is enough to present that DTW and KNN did their jobs, so the idea works. Just need better CPU or GPUs to compute the whole data set.

V. Classification II: Neural Network

A. Data Preprocess

The EEG data consists of 21 subjects, 64 channels, and 384 time points with six labels representing EEG responses to images. It is an enormous and high-dimensional dataset with a high computation cost. So before defining a model, the first thing is to eliminate some useless features, which could be channels that do not record any valuable information about brain activities.

Principle Component Analysis (PCA) is then performed to maintain the first twenty most valuable channels. The method can roughly be described as follow:

1) Normalize all the data points
2) Calculate the covariance matrix X of data points
3) Calculate eigenvectors and corresponding eigenvalues
4) Sort the eigenvectors according to their eigenvalues in decreasing order
5) choose the first k eigenvectors, and that will be the new k dimensions
6) Transform the original n-dimensional data points into k dimensions

It is implemented by the sklearn python library, using the function StandardScalar() to normalize data points and function PCA() to find the most valuable channels.

The dataset is separated into three parts using the function train_test_split() from sklearn. 60% will be the trainset, and 40% will be the test-set. Cross-validation will be performed using 20% of the data from the trainset.

The goal of the classifier is to classify the data into two categories (familiar (FN, FP, FU) and novel (NN, NP, NU)). Familiar is labeled into 0, and the novel is labeled into 1 for the task.

B. Model

Neural Network is used for modeling this task; more specifically, gate recurrent unit (GRU), a particular kind of recurrent neural network (RNN), is implemented to model the task.

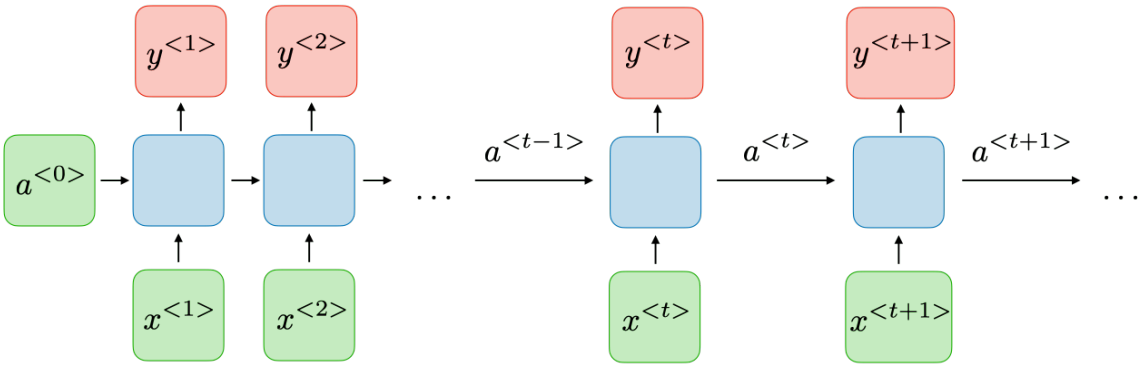


Fig. 1: architecture of a RNN

The architecture of a RNN is shown in figure 1, which allow previous outputs to be used as inputs while having hidden states.

For each time step t, the activation $a^{<t>}$ and the output $y^{<t>}$ are calculated as follows:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \tag{1}$$

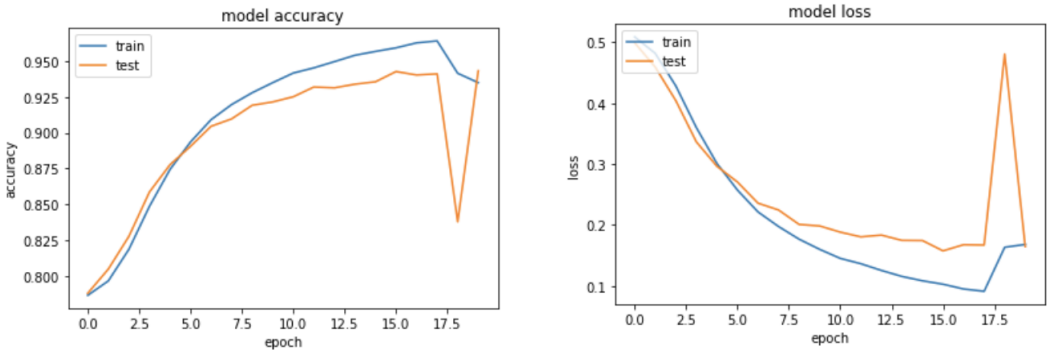
$$y^{<t>} = g_2(W_{ya}a^{<t>} + b_y) \tag{2}$$

where W_{aa}, W_{ax}, W_{ya}, b_a and b_y are coefficients that are shared temporally and g_1, g_2 are activation functions.

EEG data is a time-series data and has a large input. The advantage of RNN is that it can process the input of any length, and its computation considers historical data, which is suitable for model EEG data.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 20, 1)]           0

 gru (GRU)                   (None, 20, 256)           198144

 flatten (Flatten)           (None, 5120)              0

 dense (Dense)               (None, 2)                 10242

=================================================================
Total params: 208,386
Trainable params: 208,386
Non-trainable params: 0
_________________________________________________________________
```

Fig. 2: architecture of the model



(a) (b)

Fig. 3: Train process

The architecture of the model is shown on figure 2. It takes the data points of 20 channels and 384 time points in objects 1 and 2, and the labels as input. GRU is a particular type of RNN that deals with the vanishing gradients when the network becomes too deep. It has a update gate Γ_u to measure how mush past should maintain, and a relevance gate Γ_r to measure how much past should drop.

$$\Gamma = \sigma(W_x^{<t>} + U_{a^{<t-1>}} + b) \tag{3}$$

where W, U, b are coefficients specific to the gate and σ is the sigmoid function. The flatten layer is used to reshape the dimension of the data and the dense layer

implements the activation function to output the result (a two dimensional array only includes 0 and 1 to represent the classification).

C. Result

Train the dataset with the model for 30 epochs. It finds a nearly optimal result after 20 epochs as shown on figure 3.

The test accuracy is 94.225%.

TABLE III: Classification report

	precision	recall	f1-score	support
novel: 0	0.95	0.98	0.96	150907
familiar: 1	0.91	0.81	0.86	41247
accuracy			0.94	192154
macro avg	0.93	0.90	0.91	192154
weighted avg	0.94	0.94	0.94	192154

REFERENCES

[1] P. G. K. Athina Tzovara, "Ai medical timeseries 2022 project," 2022.