SSVEP Competition (Training-Free)

In this topic, electroencephalogram (EEG) dataset were collected from a steady-state visual evoked potential (SSVEP) based brain-computer interface (BCI). This topic aimed at verifying the performance using different training-free algorithms.

Paradigm: Visual Speller

The graphical user interface (GUI) of the system is presented in Figure 1, which is a 40-target BCI speller (including 0-9, A-Z, comma, dot, space and backspace). The frequency range of all targets is 8-15.8Hz with a frequency interval of 0.2Hz and an initial phase interval of 0.5π. The frequency and phase of each target is presented in Table I. During each trial, the GUI presented a cue for 1 second to make subject focus on the highlight target, then all targets flickered for 4 seconds. After stimuli, the result was calculated by training-free algorithm and highlighted on the GUI as a feedback for 1 second. One block contained 40 trials (1 trial for each target), and each experiment contained 6 blocks. All the subjects had a normal or corrected to normal vision. Subjects were asked to avoid eyeblink during stimuli. The event triggers were marked at the beginning of stimuli using light triggers.



Figure 1 GUI of the speller

Table I Parameters of each target

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Label** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| Frequency | 8 | 8.2 | 8.4 | 8.6 | 8.8 | 9 | 9.2 | 9.4 | 9.6 | 9.8 |
| Phase | 0 | 0.5π | π | 1.5π | 0 | 0.5π | π | 1.5π | 0 | 0.5π |
| **Label** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** |
| Frequency | 10 | 10.2 | 10.4 | 10.6 | 10.8 | 11 | 11.2 | 11.4 | 11.6 | 11.8 |
| Phase | π | 1.5π | 0 | 0.5π | π | 1.5π | 0 | 0.5π | π | 1.5π |
| **Label** | **21** | **22** | **23** | **24** | **25** | **26** | **27** | **28** | **29** | **30** |
| Frequency | 12 | 12.2 | 12.4 | 12.6 | 12.8 | 13 | 13.2 | 13.4 | 13.6 | 13.8 |
| Phase | 0 | 0.5π | π | 1.5π | 0 | 0.5π | π | 1.5π | 0 | 0.5π |
| **Label** | **31** | **32** | **33** | **34** | **35** | **36** | **37** | **38** | **39** | **40** |
| Frequency | 14 | 14.2 | 14.4 | 14.6 | 14.8 | 15 | 15.2 | 15.4 | 15.6 | 15.8 |
| Phase | π | 1.5π | 0 | 0.5π | π | 1.5π | 0 | 0.5π | π | 1.5π |

Data Acquisition

EEG data were recorded by an 8-channel wireless EEG acquisition system (Neuracle technology company) with a sample rate of 1000Hz. The 9th channel recorded the onset timepoint of events. The raw data were down-sampled to **250Hz**. It should be noted that **the values of all the event triggers were set to 1 during online test**, which were used only for event synchronization. Except the event trigger, there are other triggers for system control, which are presented in Table III. Contestant can only focus on the event triggers and ignore the others.

Table II Channel Information

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ch No. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Ch Name | ‘POz' | ‘PO3' | ‘PO4' | ‘PO5' | ‘PO6' | ‘Oz' | ‘O1' | ‘O2' | ‘Trigger’ |

Table III System trigger definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Definition | Trial Begin  (Event Trigger) | Trial Stop | Block Begin | Block Stop | System |
| Trigger No. | 1-240 | 241 | 242 | 243 | 244-255 |

During the test, data were provided under a simulated online environment. Every time when calling getData function, the system will provide a new data packet including EEG data and trigger information with a data length of 40 milliseconds. In the same block, data packets were sent in order. If the test data includes multiple blocks and data of current block are all sent out, EEG data of new block will be sent when calling getData function next time. When all data were sent out, the program finishedFlag will be set to 1. When the algorithm detected the finishedFlag changing to 1, run() method should be terminated. It should be noted that data length of the last data packet might not be constant (40 ms) since the EEG data were recorded from real experiments.

Algorithm

The algorithm calls getData to get EEG data. Once the getData called, system will return a new data packet, algorithm can save and process data packets. When the algorithm detected that the data are enough to make decision, the algorithm will call the calculation method and report the result to the system. The system then calculates the data length used in the algorithm based on the usage count of getData, and give out the information transfer rate (ITR) according the data length and accuracy.

The algorithm should meet the following constraints:

1. Trial begin constraint:

For each trial, the algorithm should begin detection after receiving the event trigger, and give out the result before the next event trigger. Otherwise the result will be regard as corresponding to the following trial.

1. Maximum data length constraint:

In this topic, the maximum detection time should be less than 4 seconds (not included). This means that the algorithm can received EEG data with a data length of at most 4 seconds after the event trigger. After that, the algorithm should give out the result immediately, or the result of this trial will be invalid and the result will be set to 0. The detection time of different subjects and different trials can be different.

1. Algorithm termination constraint:

When the Endflag in data packet is equal to 1, all the experiment data are sent out and the algorithm should stop and exit.

Framework

System flowchart



Figure 2 System flowchart

The system framework is presented in Figure 3.



Figure 3 System framework

1. FrameworkInterface

This interface connects the task program and external execution system. The implementation class of this interface is AlgorithmSystemManager, which implement all the functions defined by the interface.

1. ProblemInterface

This interface is faced to the contestant, which is responsible for data transmit and result report. Contestant can receive data from this interface, and report results by this interface. The topic should report the score according to the usage count of getData and the accuracy.

1. TaskInterface

This interface mainly implement the data transmit, score acquisition, data clear and report clear. The Framework call the task by this interface.

1. AlgorithmInterface

This Interface can validate the algorithm. **Contestants should implement this interface.** During the execution of the interface, the algorithm should get data and report results by ProblemInterface. Meanwhile, contestants should control the computation complexity of the algorithm. If the execution time exceed the predefined value, the system will stop the calculation process and the score will be invalid.

DataModel

1. DataModel for contestants
2. data: float type, containing 1 data packet. For instance, for 64-channel EEG data + 1 trigger channel, one data packet contains 65\*10 data points (40ms, 250Hz).
3. startPosition: int type, the start point of each trial in the current block.
4. personID: int type, ID number of the current subject.
5. finishedFlag: bool type, end flag of the test. When calling ProblemInterface.getData() and the finishedFlag in data packet is True, the program should be stopped.

Related function for contestants

1. ProblemInterface

This interface is implemented by the organizer, including getData method and report method. Before running the algorithm, the implementation class of this interface will be imported into the algorithm. When implementing the algorithm, contestants can call this interface to get data and report the results. Organizer will give out the score according to the usage count of getData and the accuracy.

1. def getData(self):

Input：null

Output：DataModel

function：get the next data packet.

1. def report(self, reportModel):

Imput：ReportModel

Output：null

Function：give out the result feedback.

1. AlgorithmInterface

Contestants need to write their algorithm in run function. During the execution of the algorithm, contestants can get DataModel by calling ProblemInterface.getData()and get ReportModel by calling ProblemInterface.report. When calling ProblemInterface.getData() and the finishedFlag in data packet is True, the program should be stopped.

1. def run(self):

Input: null

Output: null

Function: algorithm process

Submit format

Contestants need to submit .pyc file based on python 3.8.

Submit Sample

Refer to the sample code.

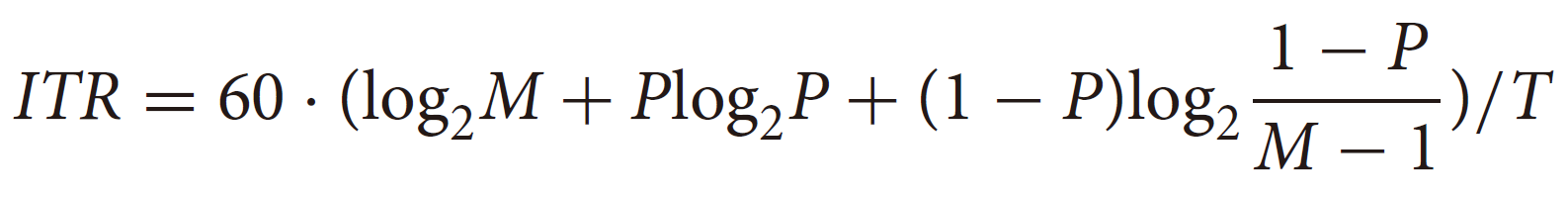
Contestants can accomplish the algorithm by modifying the files in Algorithm folder. To avoid unexpected errors, please not add folders in root directory. After accomplishing the algorithm, contestants need to compress files (Including AlgorithmImplement folder and config.toml) -->Groups --> Group --> Computing elements -->define computing element --> upload the codes --> submit to competition --> select competition --> Implementation --> finish the competition

After implementation, contestants can confirm the scores in the ranking list.

To prevent contestants from cheating by modify the framework codes, protected scoring program will cover the submitted codes except AlgorithmImplement folder and config.toml. When implementing the algorithm, all the codes (Including main.py) are from the server except AlgorithmImplement folder. The scoring program is basically the same as the sample code. It can give out scores and read data from server.

Metric

Simulated information transfer rate (ITR) is used to evaluated the performance of the system.



Where T (in seconds) denotes the average target selection time, M denotes the number of classes, P denotes the classification accuracy. The unit is bits/min. It should be pointed out that T is ideal value without resting time in the experiments.

Performance evaluating method

The algorithm gets data packet by calling getData. When there is an event trigger in the data packet the scoring system will calculate the data length used in the algorithm until calling the report method. The simulated length of the trial is the interval between the event trigger and the time point calling the report method. The average accuracy is calculated according to the output results and the real labels.

It should be noted that the data packet including event trigger is regarded as data of the last trial, and the data length of current trial is calculated from the next data packet. Therefore, the algorithm should not feedback immediately after receiving the data packets including event triggers. The algorithm should wait for at least 1 more data packet.

Abnormal results management

1. Multiple report

If the algorithm reports more than once in one trial, the system will only record the first one.

1. No feedback

If the algorithm does not report any results, then the trial length of this trial will be set to 4 seconds. The result will be set to 0.

1. Feedback timeout

If the algorithm get data more than 4 second in one trial, then the result will be set to 0, and trial length will be set to the time interval between event trigger and the time point calling the report method.

1. Algorithm timeout

To realize a real online BCI system, this topic constraint the computation complexity of the algorithm. The time limitation will be set according to the paradigm and the dataset. If there is a system timeout because of high computation complexity of the algorithm, the score of the algorithm will be invalid.