CAP 4628/5627 Project 3 – Emotion Recognition
Due 05/05 by 11:59pm

Project Description

The US Armed Forces has over 2 million soldiers when reserve components are included. It includes the Army, Marines, Air Force, and Coast Guard. New ways to manage soldiers' pain, in fight against opioid abuse, are being investigated.

One way of managing soldier's pain is identifying and treating pain as soon as it occurs. Considering this, they are interested in ways to automatically determine if soldiers are in pain (e.g. wounded), in real-time.

As you have shown them that it is possible to recognize pain with physiological data, as well as images, they are interested in the development of a new system that can identify pain. After some discussion, they concluded that physiological data will be easier to collect in real-time. Due to this, they would like you to develop a new system that **fuses** the physiological data.

Instructions

- The project must be written in Python. You are required to write a script called Project3.py, however, you are free to split up your program into multiple files if you can run python Project3.py <training_data> <testing_data>
- Data types available: Diastolic BP, Systolic BP, EDA, and Respiration
- There are 60 subjects in total (30 in data1.csv and 30 in data2.csv), each having the 5 data types above for both classes (pain and no pain). Data has been collected into a CSV file with the following columns: Subject ID, Data Type, Class, Data. Data is variable length, which is a common problem when working with physiological data.
- You will be required to use both csv files as training and testing in different runs of your program (more details will be given later in document).
- This time, instead of using hand-crafted features, you are required to down sample and normalize the data.
    - Down Sample all signals to a length of 5000. This can be done by calculating a ratio to normalize the data as $ratio = \left\lfloor \frac{total\ number\ of\ frames}{5000} \right\rfloor$. You then take the average of *ratio* number of frames as the new value. For example, if ratio=2, take the average of index 0 and 1 as the new value for index 0. You then take the average and index 2 and 3 as the new value of index 1, and so on. You must do this for all signals in testing and training.
    - Once you down sample the data, you must normalize the signals from [0,1]. (sklearn has a normalize function you can use).
- Given the pre-processed data from the previous step, you will then perform score level fusion. Details/steps are given below.

    - Do not select a data type with command line parameters. Make sure you train and test a random forest on each data type.
    - You will need to save the prediction results for each sample in the testing data, for each run. In total, there are 30 subjects (per csv file) with 4 data types and 2 classes (240 rows per file). For each subject, you need to use majority voting to decide the final class. This means that for each subject/class, look at the 4 predictions that were made and the 1 that has the majority, that is the final class. For example, given a subject, in the testing data, where the predictions are pain, pain, pain, no pain (for the 4 different data types), the final class would be pain. To determine if this is the correct prediction, look at the classes from the testing set to see if it matches. You need to use the same 4 signals from the same class for each subject. So, for each subject, there are 2 prediction from the majority voting (pain and no pain ground truth). If there is no majority (i.e. 2 pain and 2 no pain), you can randomly pick one as the class.
- The only output of your final script is the accuracy of the score level fusion. For your final script that is turned in, only print the accuracy. Your code will be run twice as
    - Project3.py data1.csv data2.csv
    - Project3.py data2.csv data1.csv

Extra Credit

There are 10 points of possible extra credit. Along with majority voting, you can also implement weighted decision fusion. To do this, you will need to use the probabilities of the random forest to assign a weight to the decision of each data type (prediction of random forest). There are functions in python to get these probabilities. As this is extra credit, no other details on this will be given. If you do the extra credit, the output of your script must show the accuracy of the majority voting, as well as weighted fusion. It should be clear which is which in the output.

Final Report

You are required to write a 4-page (plus 1 additional page for references if needed) paper on your final project. Word and Latex templates are available on Canvas. While the output of your script is only the accuracy, you will be required to include more details in the paper. Along with the details listed in the template, you must also include the following (can go in experimental design and results and discussion).

- Confusion matrices for each data type for both training and testing scenarios (i.e. data1.csv is training and data2.csv is testing, and vice-versa).
- Accuracy, precision and recall for each data type for both training and testing scenarios.
- Did majority voting result in the highest accuracy? Why/Why not? If not, which signal was best and why do you think this happened? (This can go in discussion)

Turn in

- Functioning python script(s).
- PDF of final report.

- Turn all of this in a zip file, to Canvas by the due date.

Useful Links

- **Keras -** https://keras.io/
- **Tensorflow -** https://www.tensorflow.org/tutorials
- **Scikit-learn** - https://scikit-learn.org/stable/
- **Python** - https://www.python.org/
- **OpenCV -** https://opencv.org/