OPEGRA Platform

Readme for Dummy Matlab Algorithm

Simon Ruffieux (simon.ruffieux@hefr.ch)

This document shortly describes the general structure of the Matlab dummy algorithm and emphasizes some important points for its compatibility with the platform.

The project is based on several files

- **DummyAlgo.m**: the entry point for managing, testing and optimizing the algorithm
- **Test.m**: the entry point for the Opegra platform (file referenced in algorithm.info)
- **LoadTrainSet.m**: function to load selected features from the Trainset (with ground truth)
- **LoadTestSet.m**: function to load selected features from the Testset (no ground truth)
- **TrainModel.m**: function to train the model of the algorithm
- **Recognize.m**: function to recognize classes from a dataset using a model

The project contains two main files; the *DummyAlgo.m and the* Test.m *files. The DummyAlgo.m* file is used during development to manage all the tasks (loading of data, training, optimizing and testing). the *Test.m* file is only used to load the trainset and the model, perform the recognition and output the result in the 'results.txt' file.

The function *Test()*, shown below; is automatically called by the platform during the evaluation. The full path of the test set will be provided by the platform at runtime. A set of specific features to extract is selected in this example using a vector. Ensure those match the features you selected when you trained your algorithm! Note that the labels are converted back from [1-11] to [0-10] because Matlab arrays do not like 0 indexes (see *LoadTrainSet()).*

```
function [ predicted ] = Test(testset_path)
    FeaturesToExtract = [1:9 18:29];
    %load('myModel.mat', 'model'); this dummy example does not use a model
    model = [];
    testSet = LoadTestSet(testset_path,FeaturesToExtract); %When the dataset is loaded, labels are converted from [0-10] to [1-11]
    predicted = Recognize(model, testSet);
    predicted = int32(predicted)-1; % In the original dataset, classes are labelled from 0 to 10 and not from 1 to 11
    filename = 'results.txt';
    fid = fopen(filename, 'w');
    dlmwrite(filename,predicted,'-append',...
     'delimiter',' ',...
     'newline','pc');
    fclose(fid);
end
```

Code 1 : The function Test() outputs the predicted labels in the 'results.txt' file (Test.m). This function is called by the Opegra platform during evaluation.

The function *Recognize()* recognizes all occurrences present in the dataset using the algorithm model given in parameters. You may have to initialize your algorithm using the model, depending on the type of algorithm you are using. This example only returns a random label for each gesture occurrence of the dataset.

```
function [ predicted ] = Recognize( model, evalSet)
    disp('--- Starting Recognition ---');
    % --- Initialize variables -----------------------------------
    predicted = zeros(size(evalSet.occurence,2),1);


    % --- Load and init the algorithm using the model -----------------------------------
    % No model is used in this dummy example


    % --- Iterate through all Testset and foreach occurence, predict its class -----------------------------------
    % --- Note that in our scheme, Matlab works with classes labeled from 1 to 11 (and not 0 to 10 as in the dataset)
    for i=1:size(evalSet.occurence,2)
        randomNumber = randi([1 11]); %Predict a class randomly
        predicted(i) = randomNumber;
    end
    disp('--- Finished Recognition ---');
end
```

Code 2 : The function Recognize() return the list of predicted labels (Recognize.m)

The functions *LoadTrainSet()* and *LoadTestSet()* are very similar. They load a dataset according to the path given in parameters and extract the selected features. The main difference is that *LoadTrainSet() orders the dataset into classes (required for certain algorithms) using the ground truth while LoadTestSet() does not order by classes (it can't as there is no ground truth). Note that the range of the labels are modified from [0-10] to [1-11].*

```
function [ trainingSet ] = LoadTrainSet(path, featuresToExtract)
    % LoadTrainSet
    %   loadTrainSet() loads the training set and prepares it
    %
    %   Input arguments:
    %       - path: The path of the training set file (.mat file)
    %       - featuresToExtract: a vector containing the features to extract (Time x Features)
    %
    %   Returns:
    %       - A training set ordered by class
    %          - trainingSet.class(i) contains all the occurences of a
    %          specific class
    %          - trainingSet.class(i).occurence(i).feature contains the
    %          selected features for a specific occurence (Features x Time)


    loadedSet = load(path);
    loadedSet = loadedSet.Dataset;


    for i=1:11
        trainingSet.occurenceCount(i) = 0;
    end


    for i=1:size(loadedSet.Data.occurence,1)
        currentLabel = loadedSet.Data.occurence(i).label+1; %the classes go from 0 to 10 but matlab only accepts 1 to 11 (so we
need to handle that)
        currentOccurence = trainingSet.occurenceCount(currentLabel) + 1; %increase the current occurence of the gesture
        %In the following line we do the feature extraction and transpose the matrix as HMM require F x T and the current matrix
is given as T x F (T is time and F is feature(s))
        features = loadedSet.Data.occurence(i).sensor(1).observation(:,featuresToExtract);
        trainingSet.class(currentLabel).occurence(currentOccurence).features = features';
        trainingSet.occurenceCount(currentLabel) = currentOccurence; %update the number of occurence of the gesture with
currentLabel
    end
end
```

Code 3 : The LoadTrainSet() function which returns a training set ordered by classes and containing only the selected features (LoadTrainSet.m)

You can use a completely different organisation in your files; I only showed here the most important steps which are listed below

- Have a specific file/function that will perform the recognition when using the Opegra platform. This file must be specified in the algorithm.info file
- Saving and loading the model of the algorithm
- Loading dataset from path given in parameter and converting it to features
    - Distinguish Trainset (contains ground truth) from Testset (ground truth is null)
- Output your predicted labels in the 'results.txt' file as this will be read by the platform upon termination of your algorithm.
    - One label per line!

A final word on how to **create the .zip file to upload the algorithm** on the platform. You should zip all files required by your project (.m files, model.mat, etc.). These files should be at the root of your compressed document. Note that this is the exact same structure as the zip you downloaded.