Homework 3: Human Activity Recognition using Data from Smartphones.

Solutions to this assignment are to be submitted in myCourses via Assignment (formerly known as Dropbox). The submission deadline is **Sunday December 15, 2019 at 11:59PM**. You should submit a zipped file containing a pdf (preferable latex generated) with your written answers and the Jupyter notebook with any code needed to complete the assignment. Use comments to explain your code. All code should be in the notebook while descriptions/explanations should be in the PDF.

A group of 30 volunteers performed six activities (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING) wearing a smartphone (Samsung Galaxy S II) on the waist. 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz were recorded and labeled manually. The obtained dataset is partitioned into two sets, where 70/30 for training/testing respectively.

1. **Activity recognition using the hidden Markov model (HMM)** Your goal in this problem is to perform data classification for activity recognition, using the Hidden Markov Model (HMM). Instead of working with all six classes, we use only three to save time and resources. You should therefore train three HMMs, one for each class in the training set.For the testing dataset, we compute the log_probability of each sample belonging to each of the three classes and assign that sample to the model which gives the highest log_probability value. The overall performance of the entire system should be evaluated.

   You are provided with a Jupyter notebook, `HMM_Classification.ipynb`, containing skeleton code for this question. The numbers in section below roughly correspond to the sections in the notebook. Because `scikit-learns` has deprecated its HMM module, we are using the `Pomegranate` library for the HMM functions and `Pandas` for data management. You are free to use a different library if it is more familiar but state this clearly in your report.

   Data for this homework can be downloaded from `https://www.dropbox.com/sh/cak135xdjx0ppps/AAC6wpmWh9AxASLSxaEoAJCxa?dl=0`

   (a) Read in the data from files `Train_q2.xlsx` and `Test_q2.xlsx`

   (b) Split the training and test datasets into three parts, one for label. The labels we are interested in correspond to WALKING, WALKING_UPSTAIRS, and SITTING, label values 1, 2 and 4 respectively. If using pandas, remember to drop the label column after successfully loading the data into frames.

   (c) Create an HMM for each class using the Baum-Welch algorithm to find the parameters of the model. We can assume the data is distributed according to a 3-component Gaussian Mixture Model; the command for training one of such HMMs can be written as:

   ```
   model1 = HiddenMarkovModel.from_samples(NormalDistribution, n_components=3, X=df1_train)
   ```

   where `df1_train` is the dataframe containing label 1 samples.

   (d) Now, we will test how the models perform against each sample in the test datasets in the function `get_model_accuracy`. The arguments to the function should be a dataframe and a label number.

   For each test sample we will evaluate the log_probability of that sample belonging to each of the three classes and assign the sample to the model which gives the highest log_probability value. That model becomes the predicted class value while the input to the function is the target class value.

The output of the function should be an array of counts where each count position represents the number of samples assigned to that position. The count array will be used to generate the confusion matrix.

(e) Lastly, generate the confusion matrix for your system of HMMs and discuss your findings in your PDF report. Note from the confusion matrix which classes are getting confused and see if you can share some insight into this.

2. **Activity recognition using the long short-term memory (LSTM) networks** Similar to the previous problem, you are to perform data classification for activity recognition, but this time using the Long short-term memory (LSTM) neural networks. Again, we will use only three classes. The training data samples and their target labels are passed to the network for several epochs until the training error stabilizes at a low value.

   You are provided with a different Jupyter notebook, `LSTM_Classification.ipynb`, containing skeleton code for this question. The numbers in section below roughly correspond to the sections in the notebook. We are using the `PyTorch` deep learning library which might require some initial learning and familiarizing.

   (a) Read in the data from files `Train_q2.xlsx` and `Test_q2.xlsx`. Convert the data into input tensors expected by the `Torch.nn` module.

   (b) Create a subclass `Model` to implement the LSTM. Write the `forward` function here. Also, since this is a classification problem, use the loss function `CrossEntropyLoss` - although this is actually the *Softmax* loss. Use the `Adam` optimizer.

   (c) Load the data into the `Torch DataLoader` class

   (d) Set your number of epochs and train the LSTM network. Plot the training loss and display this in your PDF.

   The TA Parikshit had to play a trick to get the network to converge, please reach out to him if your network is not converging quickly. He successfully trained this model in only 10 epochs!.

   (e) Now, we will apply the model to the test data to evaluate its performance. Again, for this classifier, display the confusion matrix and compare the results here with what you obtained for the HMM data. Discuss your observations in your PDF report.

   NOTE: There is steep learning curve to working with deep learning libraries, so start early. Also, feel free to view examples online and from Github, and to work with other classmates to accelerate your learning.

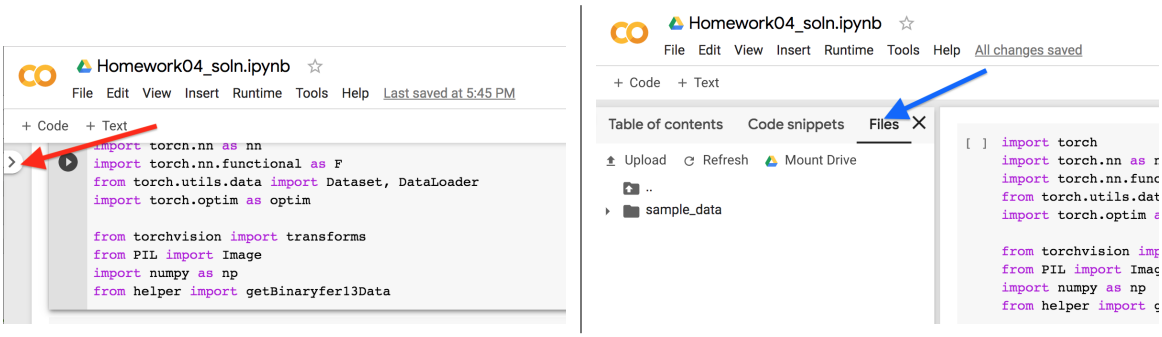
See information on the next page on how to access a free Google resource in the cloud for implementing deep learning algorithms

Using Google Colab

Google Colab will allow you to use high-end resources including access to a GPU for free, for up to 12 hours, at which time the system will refresh and you will lose any code or data not stored in another location. Ensure you save constantly and download your notebook often to avoid loss of data.

To access Colab, follow this:

- Log on to Google with your personal gmail id.
- Go to `https://colab.research.google.com/`
- From the Menu, select `File`→`Upload notebook` and upload `Homework4.ipynb`
- Next, from the Menu, select `Edit`→`Notebook settings` and check that the `Runtime type` is Python 3 and select `Hardware accelerator` to GPU.
- Click on the arrow at the left of the window (see red arrow) as shown in the image below, to expand the tab as shown on the right image.



- Select the sub-menu on the tab named `Files` (blue arrow on right image); note that this is different from the high-level option.
- You can drag-and-drop your data files there to make them accessible to the notebook. Uploading the data files is VERY SLOW; You might want to save them in your google drive and simply mount when needed.