EE5608 Sequence Modeling

- Please use Google Classroom to upload your submission
- Your submission should comprise of a single file (ZIP), named <Your Roll No> Prog_Assign, with all your solutions data-set used.
- You should use Python for the programming assignments.

Part 1 : Dynamic Time Warping DTW

1. A set of training examples and test samples of isolated words (two word classes, 0 and 1) can be downloaded from the following link: https://github.com/Jakobovski/free-spoken-digit-dataset/tree/master/recordings

   (a) Write a code for computing the DTW distance between a pair of recordings using the Type II constraint, with type (c) slope weighting and euclidean distance.

   (b) Pick any one of the training recordings as the reference for each class and compute the DTW distance between the reference and all the test samples. Is the intra class distance lower than the inter class distance always ?

   (c) Repeat for a different choice of reference sample. Will the classification change if I use a different reference from the training set ?

   (d) Will it improve the DTW method with Type III constraint using a cosine distance and type (d) slope weighting. ?

Part 2 : Implementing HMM

1. For the set of training examples and test samples of isolated words (two word classes, 0 and 1) from part 1: Implement HMM

   (a) Find Mel-Frequency Cepstral Coefficients (MFCCs) from the raw speech samples. Pick 25 ms worth of speech samples with a 10 ms overlap to find MFCCs. Use the basic 13 element version of MFCC as the feature vector representing 25 ms of speech. You are free to use “Librosa” Python library to find this feature.

   (b) Write a code to implement the likelihood computation using the forward variable after assuming a uniform flat start based initialization with 5 states per HMM and GMM with 3 mixture components per state.

   (c) Write a code to implement the Viterbi algorithm to decode the best state sequence using the existing model
   (d) Use the Baum-Welch re estimation method to train HMMs with examples from class 0 and class 1

   (e) Implement a basic two-class classifier using the HMMs constructed in the previous step. In practise, test samples come from a continuous speech waveform. Here

however, test your classifier using samples from the database. Your classifier should simply construct the likelihood of the test sample and choose the phone with higher likelihood. Classify the test examples and report the performance. How does the performance change for different number of states per HMM, different number of mixture components per GMM?.

2. For a 5 state HMM with two non-emitting states and given an observation sequence

(a) Develop the forward-backward algorithm for HMM with non-emitting states

(b) Develop the Viterbi Decoding algorithm.

Part 3: Character-level Recurrent Neural Network (RNN).

1. Find and Load Training Text :

(a) Choose the text you want your neural network to learn, but keep in mind that your data set must be quite large in order to learn the structure! RNNs have been trained on highly diverse texts (novels, song lyrics, Linux Kernel, etc.) with success, so you can get creative. **Gutenberg Books** is a source of free books where you may download full novels in a .txt format.

(b) We will use a character-level representation for this model. To do this, you may use extended ASCII with 256 characters. As you read your chosen training set, you will read in the characters one at a time into a one-hot-encoding, that is, each character will map to a vector of ones and zeros, where the one indicates which of the characters is present: char $\rightarrow$ [0, 0, $\cdots$, 1, $\cdots$, 0, 0] Your RNN will read in these length-256 binary vectors as input.

2. Implement an RNN (without using inbuilt libraries):

(a) **Backpropagation Through Time :** In an RNN with a single hidden layer, you should have three set of weights: W_{xh} (from the input layer to the hidden layer), W_{hh} (the recurrent connection in the hidden layer), and W_{ho} (from the hidden layer to the output layer). Suppose you use Softmax units for the output layer and Tanh units for the hidden layer, show:

i. Write the equation for the activation at time step t in the hidden layer and the equation for the output layer in the forward propagation step.

ii. Write the equation for the weight update rule at time step t for W_{xh} , W_{hh} , and W_{ho} in a vectorized notation. Suppose the backpropagation goes back to time step k (0 < k < t).

(b) **Network Training**: Train your recurrent neural network using the dataset you created in 1(b). You are free to choose learning parameters (sequence length, learning rate, stopping criteria, etc.). Complete the following task:

i. Report your training procedure. Plot the training loss vs. the number of training epochs.

ii. During training, choose 5 breaking points (for example, you train the network for 100 epochs and you choose the end of epoch 20,40,60,80,100) and show how well your network learns through time. You can do it by feeding in the network a chunk of your training text and show what is the output of the network. Report your Result.

(c) **Experiment with Network Structure:** As before, we want to explore how the network learns when we change parameters:

i. **Number of hidden units**. Try doubling and halving your number of hidden units. Like in 2(b), plot the training loss vs. the number of training epochs and show your text sampling results. Discuss your findings.

ii. **Sequence length**. Try doubling and halving your length of sequence that feeds into the network. Like in 2(b), plot the training loss vs. the number of training epochs and show your text sampling results. Discuss your findings.

3. Repeat the Character level RNN on your native language text data (need not be Hindi)

Hindi database link is: http://www.cfilt.iitb.ac.in/iitb_parallel/iitb_corpus_download/
Get the unicode representation of each characters and train the model.
You can use inbuilt libraries like keras, tensorflow for this programm

4. Music Data Generation using RNN (can use inbuilt libraries like keras, tensor flow):

You can use any MIDI datasets available as open source. One of the link would be http://deeplearning.net/datasets/
Use Python toolkit Music21 for to acquire the musical notation of MIDI files. Additionally, it allows us to create Note and Chord objects so that we can make our own MIDI files easily.
Train your recurrent neural network using the dataset you created using MIDI files.
Now use the neural network to generate music notes. Chose to generate 500 notes using the network since that is roughly two minutes of music and gives the network plenty of space to create a melody.
Now that we have a list of Notes and Chords generated by the network we can create a Music21 Stream object using the list as a parameter. Then finally to create the MIDI file to contain the music generated by the network we use the *write* function in the Music21 toolkit to write the stream to a file.

For RNN experiments, save the weights so that it can be readily used.