Lab 2

In this lab, you will use the MIT/TIDIGITS corpus (Leonard and Doddington, 1993) to learn about recurrent neural nets. We will improve our earlier strategy of using a fixed length signal and extract words represented by feature sequences of variable length. You will design a simple voice activity detector (VAD), also known as a speech endpointer, which will produce variable length sequences that are classified by a many to one sequence analyzer using recurrent neural networks and compare this to a feed forward network.

Your VAD will be an unsupervised learner based on Gaussian mixture models (GMMs). When you are certain that there is both speech and noise in a signal, they are usually distinct enough that a simple root mean square (RMS) energy feature is enough to partition between the speech and noise. There are numerous ways that this can be designed, but we will design a class that takes all files in the corpus as input and constructs a 2 mixture GMM. Speech and silence frames are then identified by finding the mixture that produces the highest likelihood score. If done correctly, one mixture will have a mean corresponding to speech (higher mean) and the other noise (lower mean).

We will package this into a class called UnsupervisedVAD that will reside in package myclassifier. Details on method signatures can be found in the skeleton file on Blackboard. The constructor should create an RMS stream (10 ms advance 20 ms length) over all files and collect their RMS values. Use scikits learn’s Gaussian model (sklearn.mixture.GaussianMixture) to train a 2 mixture model. Note that the GaussianMixture class expects a column vector and np.array will create row vectors by default. You can use the array methods’ reshape method with the argument [-1, 1] to put it into a column vector that the GMM expects.

Write UnsupervisedVAD.classify which labels frames as speech or noise. You can use the GMM’s predict method to find the mixture class with the maximum posterior likelihood. This can be converted into True’s when the frame is more closely associated with speech and False when it is more closely associated with noise. Access the GMM’s list of means (.means_) and determine whether 0 or 1 is speech by picking the arg max of the vector.

There are many algorithms for deciding when something is speech or noise. For a more advanced version of the GMM endpointer that adapts the means based on varying signal energy, see Huang et al. (2001, pp. 422-424). It is common to pad the detection by a small number of frames as the start and end of speech are sometimes quiet.
The isolated digits that you are working with in the TIDIGITS corpus allow us to discard most spurious detections in a simple manner. As our data consists of a single word, identify the longest set of contiguously labeled speech frames and then explore from either side until you see two frames that are not labeled as speech. The signature of features.get_features has been modified to accept a voice activity detector object. Modify the function such that if a voice activity detector has been passed in, the speech will be endpointed using the VAD. Find the longest sequence of frames that have been labeled as speech and expand it such that until there are 2 consecutive frames that are not labeled as speech.

Example:

![Diagram of speech frames](image)

**Figure 1 – Longest set of frames identified as speech is 5 frames. We expand until two consecutive noise frames are found or we reach the beginning or end of the signal.**

Test this with a feed forward network with your best network from lab 1. Produce a plot showing the tensorboard scalars for the run. Produce a plot showing the mean error without the endpointer compared to the mean error with the endpointer. Be sure to produce the 95% confidence intervals for the mean.

For the recurrent neural network portion of the lab, you have two tasks.

1. Modify classifier.recurrent to use the batchgenerator.PaddedBatchGenerator class to set up minibatches that are no longer the longest sequence in a minibatch. When you fit the model, use Keras’s mode fit generator method which permits you to specify an iterable object that returns minibatches of examples and labels. While TIMIT is a small enough data set that it can all fit comfortably in memory, it is important to know how to handle larger datasets that will not fit.

2. Create a regularized recurrent model (e.g. using LSTM) that achieves no worse than a 10% error rate (you can do better). LSTMs train more slowly than feed forward nets. If you are pressed for time, you may reduce the number of epochs to 50 or reduce the number of folds to 4. On the same plot as your feed-forward network, plot the results. Please note that I have observed a problem with
tensorboard monitoring on recurrent networks after the first fold is processed. You are not required to monitor with tensorboard.

As it is the end of the semester, no formal written report will be required for this lab. What to turn in:
1. Performance and tensor board plots. Be sure to write captions that adequately describe your plots.
2. Table showing results for each experiment. Again, a table caption is required.
3. Hard and soft copies of code.

**Reading Summary 2**

**REFERENCES**

