**AI for IoT: Domestic Audio Classification**

Aim

Develop a system that is able to achieve real-time audio classification at home of various common scenes (watching TV, cooking, walking around, etc.) at reasonably high accuracies.

Process (Updated 18 April)

The data set DCASE2017 Task 1 were downloaded and extracted. The data set gave samplings of 15 non-domestic audio scenes (buses, beaches, cities, etc.) and was used as baseline to figure out how to perform audio classification. The data files were .wav and were 10 seconds in length, sampled at 44.1 kHz resolution. There were 312 10-second segments per scene for a total of 4680 samples.

A Jupyter notebook was used to extract features from the .wav files. The files were loaded and audio features were extracted with a library called librosa. Audio files were loaded at their recorded rate of 44.1 kHz, and features were extracted with a frame length of 2048 and hop length of 512.

The features extracted were as follows:

1. Root-mean-square energy
2. Zero crossing rate
   * The amount of times the signal crossed from positive to negative and vice versa.
3. Spectral centroid
   * The frequency that divides the spectrum into 2 equal parts.
4. Rolloff factor
   * The frequency in which 85% of the spectrum lies below.
5. Spectral flux
   * Indicates how quickly the energy of the spectrum changes.
6. Mel-frequency cepstral coefficients (20)
   * A representation of the short-term power spectrum of a sound.

As each feature is calculated using a frame of the whole audio sample, a vector of length 862 (44.1 kHz \* 10 seconds / 512 hop length) is returned. In order to streamline the training process and avoid having too many features, the mean and standard deviation of each feature is calculated across all the frames. This gives a total of 50 features.

The features were extracted using a helper function written to return a feature dictionary and the extracted features were stored in a pandas Data Frame and pickled to use. Extraction took 27 minutes and 46 seconds and the resulting pickle was 1.8 MB.

The pickle file was loaded in a separate Jupyter notebook and the sklearn library was used to perform basic classification techniques to the data. The data was split into 75% testing data and 25% testing data. Two pre-processing methods were also used, standard scaling (used to scale data to fit a standard normal distribution {0 mean and unit variance}) and min-max scaling (used to scale data to fit in the range [0, 1]). The scaling methods were fitted on the training data and then used to scale the test set.

The first classification algorithm used was the K-nearest-neighbours classifier. Graphs were plotted showing the accuracy of the classifier whilst varying k. All 3 sets of training data were used, in order to compare the effectiveness of the different pre-processing methods. The graphs showed the highest accuracy when k=1, with an overall downward trend as k increased. Of the 3 pre-processing methods, the min-max scaled data had the highest accuracy of 95.4%. The standard-scaled data had an accuracy of about 94%, while the raw testing data had only 54.5%.

A helper function was written to display the accuracy, average precision, recall and f-score of the classifier, as well as the individual precision, recall and f-score of each label. The 1-Nearest-Neighbour classifier on the min-max scaled data had an average precision and recall of 95.5% and 95.6% respectively, for an average f-score of 95.5%.

Next, a logistic regression model was tested. The gradient descent algorithm used was the L-BFGS and the maximum iterations for this algorithm were set at 200. Results were decidedly worse, giving an accuracy of 85.7% for standard scaling and 80.6% for min-max scaling. The non-pre-processed data had an accuracy of 78.5%.

Next, a standard Gaussian Naïve-Bayes model was tested. Results were also not very good. Standard and min-max scaling had an accuracy of 66.2%, while non-pre-processed data had an accuracy of 66.9%.