Dylan Bannon
Intro to Intelligent Systems
Zach Butler
12/8/2016

Lab 2 Writeup

I am using a regular old decision tree for my predictions. The tree is recursively generated, with nodes at the top of the tree composed of features with the highest information gain. To figure out the best features, I iterate over each value of each feature, split the data into two sets based on that value (one set greater, one less) and calculate the information gain based on the entropy of each of the sets. The best feature-value combination is then used for the next node. This is repeated recursively until the information gain is zero, in which case a leaf node is created with the number of unique results per class for the remaining data as the number of votes for that node.

The features I used were simply the 13 cepstra returned by the MFCC processing of the wav file. Each example was composed of the 13 cepstra and I stored them all in a JSON file. The examples and features don't take too long to generate (roughly 15 seconds on my machine), so it's not too important that they be saved in a file but it does save some time for testing purposes.

As for accuracy, I would say my model is average-good. In my (albeit limited) testing, I saw ~80% accuracy on the training data and ~50% accuracy on novel data. Interestingly, Spanish and Polish predictions were significantly more accurate than English. On the training data, I saw 100% accuracy for Spanish and Polish but only 50% for English. I expect the accuracy to be around 50% for clean test data, probably a bit lower for non-native speakers.

Running the project:

This program requires Python3.4 and the latest versions of `numpy`, `scipy`, and `python-speech-features`. To install with pip, from the root of the project simply run:

```
pip install -r requirements.txt
```

This will install all of the requirements defined in requirements.txt. The program can then be run by the Python interpreter:

```
python3 lab2.py [extract_features | generate_model | predict_language <filename>]
```

The program arguments correspond to which part of the program to run. To extract the features into the JSON file, use the `extract_features` argument. All training data must be in a directory called `audio_data`, which is at the same level as `lab2.py`.

To generate the model, use the `generate_model` argument. There must be an `extracted_features.json` file present in order to generate the model. The resulting model is stored in a binary file called `desision_tree.model`.

To make a prediction, use the `predict_language` argument, along with the filename of the file to predict (relative to the root of the project).

Here is the directory structure of the project:

```
|-Lab2
  \
   |-lab2.py
   |-requirements.txt
   |-decision_tree.model
   |-extracted_features.json
   |-audio_data
    \
     |- [*.wav]
```