Accent Recognition with Neural Network

Matthew Seal, Matthew Murray, Ziyad Khaleq

Training Data

Our primary goal for our training data was to obtain audio clips of individuals with a variety of accents saying the same phrase. While we also hoped to obtain a large number of training samples, we were focused on obtaining training samples containing the same words for our initial sample so that our features would be the same for each training sample.

After some research, we found a university-sponsored site that contained exactly the type of data we were looking for: http://accent.gmu.edu/. This site had clips of numerous different accents saying the following phrase:

"Please call Stella. Ask her to bring these things with her from the store: six spoons of fresh snow peas, five thick slabs of blue cheese, and maybe a snack for her brother Bob. We also need a small plastic snake and a big toy frog for the kids. She can scoop these things into three red bags, and we will go meet her Wednesday at the train station."

Due to the academic nature of this site, all of the audio clips contained essentially no background noise, making our preprocessing simpler. Furthermore, since the phrase is contains a variety of words, we had a large number of features to use in our algorithm. The next step was to preprocess this data into useful training samples.

Preprocessing

There were two steps to our preprocessing. The first step was to convert the audio clips from George Mason's website from .mov files into something more easily manipulated in Matlab. We chose to convert them to .wav files, and we used the Audio Converter tool from iOrgSoft to easily convert the .mov files into .wav format.

The second step was to split the .wav files into clips of individual syllables. We planned to use each syllable as an input feature to the algorithm. In order to determine the syllables, we used a magnitude window filter to select samples from the original clips. In speech, there is important information at the beginning and the end of each syllable that can be used to distinguish between accents. Consequently, we desired to produce samples that captured the entire length of the syllable. After experimenting with a number of different sample lengths, we ultimately decided, based on performance results, to produce samples clips that were a minimum of 100ms and a maximum of 500ms in length. Samples in this range of lengths seem to fully encapsulate most syllables.

Once we have split original sound clips into samples, we normalize the magnitudes of each sample to account for any differences in natural speech volume. Also, we shift each sample by the center frequency of the original speaker's voice (determined by examining the frequency content of the original clip). The sample is shifted so that it is centered around the average speaking frequency of humans. The combination of normalization and shifting is designed to eliminate any extraneous noise not related to accent that might arise from a speaker's gender or natural speaking volume.

The shifted and normalized sample is then divided into buckets according to the frequency content of the sample. Experimental results indicate that approximately 8 buckets produce optimal results. These buckets containing information about the frequency content of the sample are then the inputs to the neural network.

<u>**Neural Network Design**</u>

Our neural network design was built from scratch in Python using no external libraries, except numpy for preprocessing the wav files. To get all of the elements of our final network complete we had a large coding task ahead of us. We started with a simple three layer feed-forward network that could take any set of inputs to the first layer and give any output we desired in the final layer. We defined links between nodes and between input/output and nodes to have an explicit transfer function which defaulted to a sigmoid function. The network thereby defaulted to taking in arbitrary data and returning a confidence value from -1 to 1 for each discrete output.

We then analyzed the various options we had for choosing our learning algorithm. The traditional method of training a neural network is to implement back-propagation. Back-propagation however is prone to local minima problems, which we believed accent recognition has in abundance. We looked at several approaches and eventually decided to try and implement a genetic algorithm to perform our learning based initially on Training Feedforward Neural Networks Using Genetic Algorithms (David J. Montana and Lawrence Davis, 1989). There is statistical and empirical evidence that this approach avoids many of the localization problems found in high dimensionality state spaces, though convergence can sometimes be slower than back-propagation.

To produce our genetic algorithm we had to first create a way of obtaining a neuron “fitness” indicator. The indicator needed to provide information about how much the given neuron helped or hurt our output results. A simple approach would be to run the network once for each neuron without the contribution of the given neuron and compare the distance from training results against the original forward propagation. This however makes learning an $O(n^2)$ operation.

To avoid this problem we made a neuron method which recursively removes its effective input from the network without disrupting neuron’s that aren’t forward attached to the original neuron. The results of running this fitness algorithm allow us to compare the change in distance from training values with and without a neuron present in O(1) time for a fixed number of output neurons. The overall algorithm then becomes O(n) time to determine the fitness of every hidden neuron on a network. The same method can be applied to input neurons on trained networks to perform principle component analysis on our input data and shrink our overall network size.

<u>**Genetic Algorithm**</u>

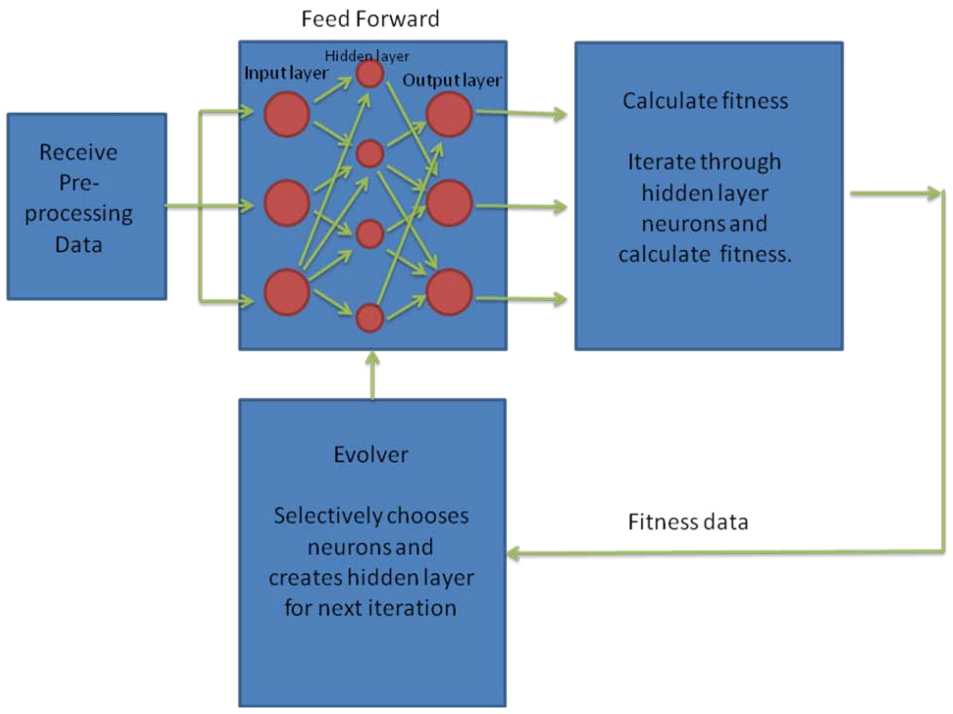
In implementing our genetic algorithm we had many choices in what elements we could include in order to enhance performance. The primary feature of genetic updates comes from a cross-over technique where new neurons are made by combining parent links and weights in interesting ways – much like chromosomes in cell Meiosis.

However, there are a wide range of other techniques employed, including elitism, immigration, mutation, colony separation, multiple parent cross-over which can improve performance in difficult problems. We chose to implement all of the aforementioned techniques except colony separation.

The choices for our genetic algorithm resulted in a network where evolving the network caused the following set of operations to occur. First we gathered the fitness of each neuron in our hidden layer. Then we randomly picked by weighted fitness a percentage of our neurons to continue onto the next iteration as elite nodes. And finally for the remaining neurons, up to the number of neurons that were present in our network before, we either introduced an entirely new immigrant neuron with new weights and links or performed cross-over to generate a neuron from our existing network. This produced an entirely new hidden layer for our network which was based on the prior hidden network and the fitness of our neurons.

Some of the process of evolving our network was enhanced to improve convergence and local minima avoidance. For example, when we performed cross-over we used four parents rather than biologically common two parents to generate a new neuron. Additionally we added mutations to each cross-over event to allow weights to change from their original parental values. These mutations were allowed in rare cases to change the entire transfer function between nodes from a linear relationship to a more exotic function such as an exponential or other non-linear operator. Variations on a Simple Genetic Algorithm (Douglas, 2009) provided a good set of initial parameter values for these additional algorithms and gave us an indication of which parameters could help with which type of problems as they appeared in our training process.

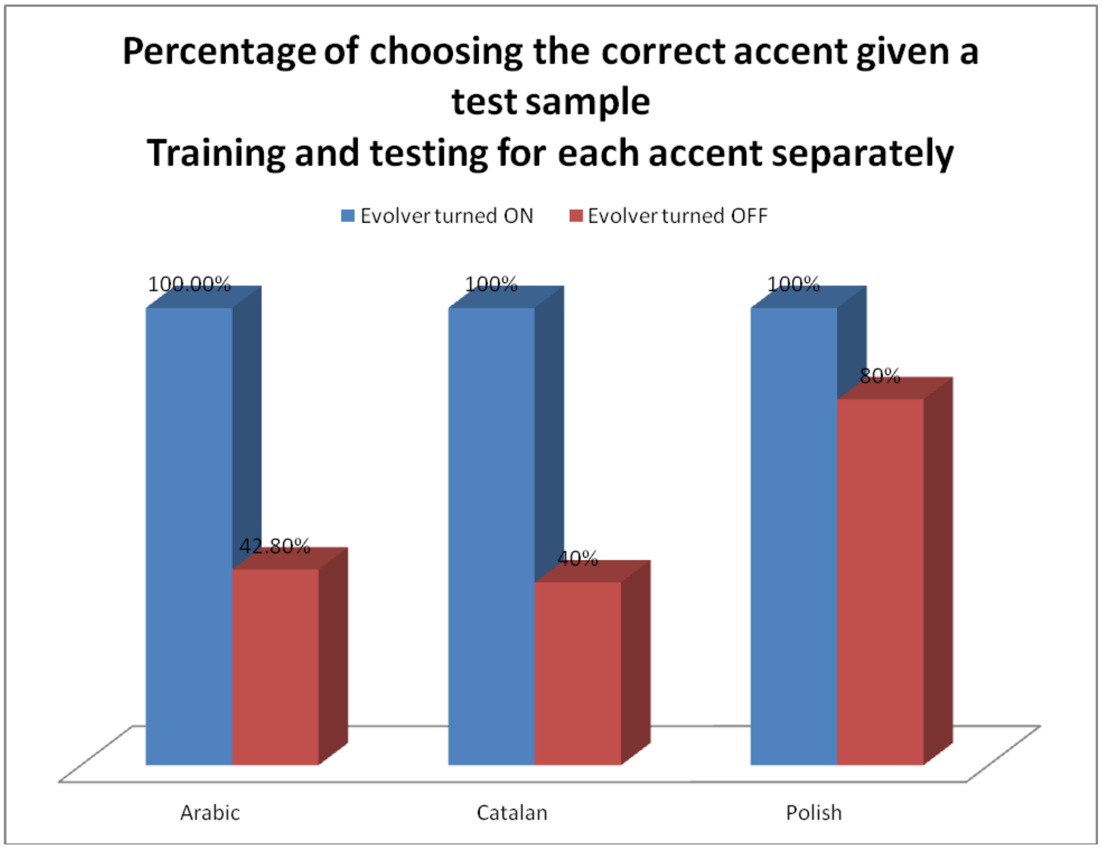
Complete Network Topology


Feed Forward
Input layer
Hidden layer
Output layer
Receive Pre-processing Data
Calculate fitness
Iterate through hidden layer neurons and calculate fitness.
Evolver
Selectively chooses neurons and creates hidden layer for next iteration
Fitness data

Accent Recognition with Neural Network
Matthew Seal, Matthew Murray, Ziyad Khaleq

Training and Initial Results

Once we had our network build, our training data categorized and preprocessed, and debugged our codebase we implemented a variety of training processes to attempt accent recognition on a set of foreign English speakers. We started with stochastic updates to our network, where we learned on a single speaker for an extended learning period before attempted to introduce a new speaker. This process resulted in the network recognizing a single speaker with 100% accuracy but which was unable to effectively learn to differentiate two out of three speakers from each other. The network would over-fit the particular training samples it had last received and forget how to recognize speakers from earlier training. The process did show that our evolver was very effective at matching a single speaker at a time though and could reach near 100% identification with just 10 syllable training inputs.



Given that our network had difficulty with stochastic updates, we decided to apply batch updates to allow the network to see many speakers in short succession. This resulted in our network never converging and eventually – once it had seen many samples – to select a state which miscategorized all speakers, but with less total error than correctly categorizing a single speaker. Once the network was in this fixed state its accuracy was close to random guesses. The rate at which the network reached a steady state was heavily influenced by which data we used for hold-out and what data we used for training. Certain samples seemed much more influential than others.

Input Changes and Conclusions

With the knowledge that our network seemed unable to find a solid differentiation between speakers and that specific samples had large affects on our network performance, we re-approached our input methodology. Our original input was 100 -500 ms samples of syllables which were slit into sets of 8 fft-buckets from time samples of the beginning, middle and end of the sample. We tried increasing the number of buckets and changing the size of our samples to include more phonemes per sample. Changing the number of buckets did not help the network, but using larger samples seemed to provide some minor improvements in some cases. Our best results were around 40% accuracy in identification of various speaker accents on our hold out data.

Given the difficulty of the task and the fact that we made certain assumptions about the information needed to resolve accent differences it makes sense to us that our network would have difficulty identifying the subtle differences in accents. We think from our data that there isn't sufficient information in the sample formats we were providing to conclusively distinguish various accents. The neural network was learning more to recognize speech than to recognize a style of speech. To resolve such problems we believe that a more detailed analysis of speech patterns is required before our network could better resolve the input data. Detailing the difficulties which foreign English speakers have with pronouncing certain words could lead to a more structured input of data into our network.

Overall we were glad to be able to create a machine learning algorithm which was outside of the scope of this course and apply the theory taught during the quarter to better understand where our algorithm would do well and why it was not working as well as we would have liked. We didn't want to just use a prebuilt library to analyze a task we knew we could already solve, but instead see what limitations are present in applying new techniques to hard problems and to learn how to build and alter machine learning algorithms to our focus problem. We succeeded in finding where the algorithm was limited and what we needed to focus on if we were continuing this work into another quarter. With better pre-processed data and variations on the network parameters we think that our network could be trained to accurately predict a speaker's accent, but that with the data we gathered the problem space was not separable.

References

Douglas, Scott H. "Variations on a Simple Genetic Algorithm." *Unpublished AI Work*. Web. 26 Nov. 2011. <http://www.scottdouglas.net/projects/ga/paper1.pdf>.

Montana, David J., and Lawrence Davis. *Training Feedforward Neural Networks Using Genetic Algorithms*. *International Joint Conferences on Artificial Intelligence*. 1989. Web. 26 Nov. 2011. <http://ijcai.org/>.

Shibata, Katsunari, and Koji Ito. "Gauss-Sigmoid Neural Network." *Neural Networks, 1999. IJCNN '99. International Joint Conference*. 16 July 1999. Web. 2 Dec. 2011. <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=831131&tag=1>.