Speech recognition using multilayer perceptron artificial neural network

Tobias Bengtsson
Undergraduate Student
Department of Computer Science
Lund University
tobbe@tobbe.nu

Abstract

We implement the part of automatic speech recognition that recognizes phonemes using standard signal analysis methods Discrete Fourier Transform and Linear Predictive Coding. The System is speaker independent aiming at large vocaulary but with discontinuous speech. A multilayer perceptron artificial neural network is trained. No attempt to find out which words the phonemes is part of is done.

1 Method

1.1 Tools

Most of the code is written in the programming language *python* with numpy and scipy. The *GNU Compiler Collection* (GCC) was used for C++ code. *Weka*, the machine learning toolkit, was used to train classifier models. *Audacity* was used for audio editing. *Gnuplot* was used to plot debug information. The *Snack* sound toolkit from KTH was used to read the Waxholm recordings.

1.2 Signal Analysis

A formant is a peak in the frequency spectrum that are resonant from the vocal tract. We want to find these formant frequencies. They are the distinguishing components of human speech. The first formant is the one with the lowest frequency, which is called f_1. There are usually three where the two lowest is considered the most important. (Wikipedia, 2006)

1.2.1 Fast Fourier Transform

Given the input signal $\tilde{x}_n \quad n = 0, \ldots, N-1$ we use the Hamming window to reduce what is called *spectral leakage* which occurs due to the discreteness of the transform.

$$w(n) = (0.53836 - 0.46164 \cos\left(\frac{2\pi n}{N-1}\right)$$

$$\forall n \quad x_n = w(\tilde{x}_n)$$

then take the DFT (Discrete Fourier Transform):

$$\hat{f}_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} nk} \qquad k = 0, \ldots, N-1.$$

Using the Fast Fouier Transform which is an efficient algoritm to compute this. To get a spectrum in dB we set

$$\forall k \quad g_k := \log |\hat{f}_k|$$

Then we scale it and use only the coeffiecents below 3500 Hz since no formant lies above this frequency.

1.2.2 Linear Predictive Coding

LPC approximates the voice apparatus with a tube with a buzz in one end. The tube models the mouth and the buzz is generated by glottis (the space between the vocal cords). LPC assumes the signal is somewhat stationary and model it as a linear combination of its previous samples:

$$X_i = \sum_{k=1}^{n} a_k X_{i-k}$$

The task is to find the coefficients a_k. To do this we use the algoritm called Durbin-Levinson's recursion[1]. I ported an implementation of it in C found in Ogg Vorbis sourcecode (Vorbis,). We will only use the first 10 coefficients. Given the coefficients we use them as a filter to get a frequency spectrum.

1.3 Machine Learning

I use the machine learning toolkit *Weka* to test whether my feature vectors is recognizeable. The Naive-Bayes method was used.

A multilayer perceptron artificial neural network was written in both C++ and python. It learns using backpropagation i.e. generalized delta rule. It handles multiple hidden layers and can (de-)serialize its state (from)to XML.

[1] Invented by N. Levinson 1947, modified by J. Durbin in 1959

2 Corpora

2.1 High quality recordings

A corpus was recorded at home under near studio conditions (high quality, no background noise). The Swedish vowels a, o, u, å, e, i, y, ä, ö were recorded in 90 samples from 8 different speakers. Because of the need for people to come to my home out of town to get these optimal conditions it was impossible to get the amount of people needed to produce this kind of corpus.

2.2 Dictaphone recordings

A second corpus was collected using a dictaphone. Students were recorded at the hospital in Malmö. Unfortuneatly the recordings suffered from noisy background, clipping and overall poor quality. So these were unusable.

2.3 Waxholm database

The Waxholm database is a corpus of 4468 recordings from 66 different speakers. About half of them speaks the Stockholm dialect. It has more than 600 words. Each recording is tagged at phoneme level using 40 phonemes in a variant of the Swedish Technical Alphabet (STA). A parser was written for these tag files.

3 Results

The first attempt made to train a classifier used my high quality recordings with the nine vowels. The neural net trained fast but could not recognize the test set. Weka using the Naive Bayes method was trained as well but couldn't classify correctly anything better than pure chance.

The same experiment was tried with the dictaphone recordings but it did not work.

I chose to pick a narrow subset of the waxholm database to get some results quickly as I was now running out of time. The phonemes `E0` (as in 'ligger'), `'A:` as in 'var', `'I` as in 'till' and `']:` as in 'skärgården' were chosen. These whole phonemes were cropped from the sound files yielding roughly 6000 new files. Generating feature vectors from FFT only and using Wekas NaiveBayes with 10-fold cross validation gave 65% correctly classified. Instead using LPC gave 63% correct.

It appeared that in the beginning and the end of the phonemes there are a transition from/to the surrounding phonemes which makes it harder to classify. Given this the first and last 10 ms was skipped. Also to get a more stationary signal within each frame the remaining middle part was divided in 20ms frames. After these steps we now have 20 621 frames. Weka was retrained with a FFT feature vector and results were boosted to 69%, a 4% increase. Given a vector of both FFT and LPC gave 74% correct. We get the following confusion matrix which shows that `'A:` is very hard to differentiate from `']:'`

`'A:`	`'I`	`E0`	`']:`	<– classified as
4219	28	822	989	`'A:`
6	2617	393	21	`'I`
459	658	3536	185	`E0`
1574	15	296	4803	`']:`

Figure 1: Confusion matrix

Presenting that best vector to a series of neural nets I achieved only 62% even when using only the training set. Finding the optimal number of hidden nodes and layers took several weeks but could be automated somewhat.

4 Future Improvements

Given more time one could use recurrent neural networks (Hopfield, 1982) e.g. Boltzmann Machines (Ackley et al., 1985) to model temporal features. Recurrent networks will improve the rate of successful classification (Elenius and Blomberg, 1992). We could use optimal brain damage (LeCun et al., 1990) to increase learning speed and improve generalization or go the other way and use a constructive network i.e. Cascade Correlation (Fahlman and Lebiere, 1990). Using more advanced features like Perceptual Linear Prediction and Mel Frequency Cepstral Coefficients will most certainly boost the performance. Further improvements that could be made are classifying gender before classifying phonemes (Abdulla and Kasabov, 2001).

References

W.H. Abdulla and N.K. Kasabov. 2001. Improving speech recognition performance through gender separation. In *Artificial Neural Networks and Expert Systems International Conference*, pages 218–222, Dunedin, New Zealand.

DH Ackley, GE Hinton, and TJ Sejnowski. 1985. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169.

Kjell Elenius and Mats Blomberg. 1992. Comparing phoneme and feature based speech recognition using artificial neural networks. In *ICSLP-1992*, pages 1279–1282, Banff, Canada.

Scott Fahlman and Christian Lebiere. 1990. The cascade-correlation learning architecture. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*. Morgan-Kaufmann.

John Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558.

Y. LeCun, J. Denker, S. Solla, R. E. Howard, and L. D. Jackel. 1990. Optimal brain damage. In *Advances in Neural Information Processing Systems 2*, pages 598–605, San Mateo, CA. Morgan Kaufmann.

Ogg Vorbis. - `/lib/lpc.c`. http://www.vorbis.com.

Wikipedia, 2006. *Formants*. http://en.wikipedia.org/wiki/Formants, December.