Adaptive Feature Extraction with Spike Coding for ANN Based Phoneme Recognition

Danijel Koržinek, Polsko-Japońska Wyższa Szkoła Technik Komputerowych
(Prof. Krzysztof Marasek, Polsko-Japońska Wyższa Szkoła Technik Komputerowych)

Abstract
Most speech recognition systems rely on spectral based features or others derived from them (e.g. Mel frequency cepstral coefficients). These features are usually adjusted according to the psychoacoustic properties of the human or human-like auditory systems obtained by real-life experimentation. However, since the auditory system is not completely understood, these modifications are not entirely accurate and further improvements can be achieved. Several attempts of finding optimal features for speech recognition have already been made, either through artificial evolution or gradient descent of different parameters in conventional feature sets. This research explores the possibility of deriving such features in a single process together with an ANN model for recognizing phonemes, commonly used in hybrid speech recognition. Special attention is paid to features extracted with spike coding mechanisms, which are known for their sparseness and efficiency and exhibit many similarities with auditory processing in the brain. The adaptation of these features is inspired in how our own auditory system learns to recognize sounds, but it is also supported by a lot of research that shows the similarity of features derived by adaptation and their counterparts derived from physiological and psychological experimentation. The research is still a work in progress so final results are yet unknown.

1. Introduction
Feature extraction is the first step in the process of speech recognition. As such, it is highly influential in the systems performance. Any drawback in that phase can have considerable and insurmountable consequences on the final result. That is why feature extraction has been a separate subject of study for decades.

Initial attempts centered on using well known signal processing techniques to extract seemingly significant chunks of information used elsewhere in the industry. The emergence of the digital telecommunication era provided many techniques in efficient speech coding useful also for speech recognition. These were mostly based on spectral analysis of the data and provided such well known feature sets as LPC [1], filterbanks and even cepstral features.

They were later improved using the information about the psychoacoustic properties of our auditory system. These were established through numerous experiments to quantitatively determine how humans perceive certain sounds. Several perceptual scales were found that way, like the mel scale [13] that is used to improve the location and shape of the filters used in previously mentioned feature sets.

One of the most popular feature sets consists of the so-called Mel-Frequency Cepstral Coefficients (MFCCs). They are computed by first extracting the logarithm of the outputs of a mel-filterbank, which is a set of filters spaced linearly in the mel-scale (logarithmically in the spectrum). These outputs are then further processed by the DCT to get the cepstral coefficients. Cepstrum is a play on words of the term spectrum and it stands for the Fourier transform of the logarithm of a fourier transform of a signal. DCT can be used instead of the Fourier transform, when only the magnitude is necessary.

To measure the quality of the different feature extraction techniques for the task of speech recognition several test scenarios can be attempted. Most commonly, however, the simplest possible test is chosen, e.g. phoneme recognition on a well known data set, for example the TIMIT corpus [2] which contains the transcribed recordings of short utterances in American English. Indeed, it is very unlikely that a feature set would perform better on the phoneme recognition task, but worse on full vocabulary speech recognition (and vice versa). Still, this method of evaluation has gained much criticism over the years (e.g. [3]) mainly due to the lack of representative testing datasets. In other words, some features that performed better in simple tasks would drastically loose in quality when noise or other type of distortion was introduced. Nevertheless, the simple experiment is still useful in providing key
information on the quality of the different aspects of the speech recognition process.

2. Phoneme recognition task

Phoneme recognition is most commonly performed by some form of machine learning. The most common rivaling methods are the Gaussian-Mixture Models [4] and Artificial Neural Networks [5]. The experiments in this paper are based on the latter approach.

For a baseline experiment, the previously mentioned TIMIT corpus was used. A Recursive Neural Network with approx. 175k weights trained to classify consecutive frames to one of the phoneme classes. These were later converted into phoneme sequences using the forward algorithm [4]. The sequences were compared to correct manual transcriptions to acquire the accuracy measure: number of correctly positioned segments divided by the number of segments in the original sentence. The number of correctly positioned segments is calculated by subtracting substitutions, insertions and deletions from the segment count. The substitutions, insertions and deletions are calculated using the Levenshtein distance method.

Here are the statistics for some common feature sets:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>12 mel-scale filterbank outputs</td>
<td>39.66</td>
</tr>
<tr>
<td>filterbank+delta+acceleration coeff.</td>
<td>32.09</td>
</tr>
<tr>
<td>filterbank+energy+delta+acc.</td>
<td>31.67</td>
</tr>
<tr>
<td>MFCC</td>
<td>28.28</td>
</tr>
<tr>
<td>PLP</td>
<td>28.60</td>
</tr>
<tr>
<td>gammatone filterbank</td>
<td>29.39</td>
</tr>
</tbody>
</table>

Most of the features were calculated using the HTK toolkit and the details of their implementation can be found in [6]. The gammatone filterbank feature set was calculated using a method described in section 3.

The same experiment was repeated on a polish corpus of our making. The results were similar with respect to the order of the features (from best to worst). This means they are equally representative of the task at hand and thus we can use our corpus to compare the results between the languages.

3. Sparse coding methods

Sparse coding methods have been getting a lot of attention recently. It has been shown that they are not only efficient from the statistical point of view, but there is also a strong correlation between them and certain biological sound analysis processes [7]. The basic idea of sparse codes is to use a more general time-frequency representation of signals than the traditional Fourier-like spectral analysis. This representation is known as the sparse, shiftable kernel representation [8]:

\[ x(t) = \sum_{m=1}^{M} \sum_{\nu=1}^{n_m} s_i^m \phi_m(t - \tau_i^m) + \varepsilon(t) \]  

(1)

The signal \( x(t) \) is represented as a weighted sum of kernels \( \phi_m(i) \) each multiplied by a weight given by \( s_i^m \) and shifted in time by \( \tau_i^m \). The error \( \varepsilon(t) \) denotes the fact that this representation is not perfect and given a finite code length (finite number of kernels) there can always be some residue between the original and the encoded signal.

This method of encoding has one major advantage over the standard Fourier-based ones and that is the ability to use arbitrary kernel shapes. Fourier analysis decomposes the signal into frequency bands that are described by filters whose coefficients are sine-wave functions in the time domain. The sparse, shiftable-kernel method allows for any shape of the filter in the time domain.

There are, however, several obstacles to overcome before this becomes a viable method. Assuming we have a dictionary of kernel shapes to work with, the primary problem is how to encode signals. We need to define the amplitude \( s_i^m \), location in time \( \tau_i^m \) and the chosen kernel shape \( \phi_m(i) \) for each instance of the kernel in any encoded signal. There are three possible methods. The most accurate and slowest of all is the gradient descent method. It is currently unfeasible for any practical use, so it will not be investigated in this work. Second way is to use some sort of greedy iterative approach, like the matching pursuit algorithm [9]:

\[ R^0x = x \]
\[ R^nx = \left( R^{n-1}x, g_m \right) g_m + R^{n-1}x \]  

(2)

\[ g_m = \arg \max_{g_m \in D} \left( R^nx, g_m \right) \]

This algorithm is described above using a recursive formula where \( R \) is the residue or error of the current encoding. The parameters (amplitude, location and kernel) are stored in the \( g \) variable. It can be shown that \( R \) decays with each step and the more steps, the better the encoding. Given, this is a greedy method (we choose the best fitting kernel at each step), the final result is obviously suboptimal. This brings up several other problems, like choosing the appropriate amount of iterations (stop criterion) and also whether the criterion for choosing the best parameters at each step is correct. With the current criterion, the phonemes with higher energy (e.g. vowels) will always be preferred to the ones with
lower energy (e.g. plosives). This means that some phonemes will be represented by a considerably lower amount of kernels than others, which makes them much harder to recognize. Even though this was the favorite encoding method in [7] it is still not good enough to compete with the traditional features in the phoneme recognition task.

The third way of eliciting the parameters for sparse code signal encoding is known as the filterNthreshold method. In [7] it is used only as a preliminary estimate of the true parameters and is shown to perform much worse than other methods with regards to sparseness and overall encoding efficiency. Nevertheless, it is often chosen when sparse codes are used for speech recognition tasks (e.g. [12]). This method was also used to perform the gammatone filterbank experiment mentioned in section 2. In fact, assuming the threshold is set to 0, the method is equivalent to a simple filterbank, where the kernel shapes describe the filters of the filterbank. For the experiment, 64 kernel shapes were generated according to the gammatone filter specifications [10]. The result is very similar (only slightly better) to the mel-scale filterbank (calculated from the FFT), which is not surprising given they are both using very similar perceptual scales.

3. Kernel functions adaptation

As already mentioned, one of the main advantages of the sparse coding methods is the ability to adapt the kernel function to the task at hand. To do this we will employ the standard gradient-descent technique on the function performing the filterbank-threshold procedure. The formula for the output of a given filter is:

$$O_f(t) = \sum_{i=0}^{L_f} \phi_f(i)x(t+i),$$

$$t = 1..T, f = 1..N$$

(3)

Where T is the length of the output and N the number of the filters. \(\phi_f(i)\) are the values of consecutive coefficients of the kernel \(f\) of length \(L_f\) and \(x(t)\) is the signal.

The difference between the targets \(\Omega\) and the outputs above is given by:

$$\Delta = O - \Omega$$

(4)

Which makes the error function, given the MSE criterion:

$$E = \Delta^2$$

(5)

The partial derivative of the error function with respect to the given kernel coefficient is given by:

$$\frac{\partial E}{\partial \phi_f(i)} = -4 \sum_{i=1}^{T} \Delta_f(t)O_f(t)x(t+i),$$

$$f = 1..N, i = 1..L_f$$

(6)

Finally, the kernel coefficients can be updated iteratively, given a certain learning rate \(\eta\):

$$\phi_f^{X+1}(i) = \phi_f^X(i) + \eta \frac{\partial E^X}{\partial \phi_f^X(i)}$$

(7)

As shown in [7] this form of adaptation may even be present in the biological counterparts of the systems being developed. The adaptation of such features is not a brand new idea and has already been studied before [11]. The novel approach here is the idea to feed the targets for this adaptation straight from the network used for the phoneme recognition task. In other words, the kernels used in the feature extraction are being trained together with the phoneme recognizer to find the optimal parameter set.

4. Conclusion and future work

This paper describes a novel adaptive feature extraction technique for use in speech recognition. The work is still in progress, so the results were not known at the time of writing the paper.

Sparse code features were used in the task of speech recognition. A few attempts have already been recently made with these features, but the success was insubstantial when compared to the traditional approaches. Actual progress may come from the utilization of adaptation of kernel functions.

The idea of adaptive features for speech recognition is not new and has been already discussed in literature. The difference in this approach is that the targets for the adaptation would come directly from the ANN used for recognition. Also, as mentioned in the introduction, different types of testing scenarios should be explored, e.g. robustness to noise or other distortions.

Finally, the feasibility of using this method in a real-life scenario needs to be further examined. As it stands now, the method is quite slow: at least several times slower than realtime compared to traditional methods which are often tens maybe hundreds time faster than realtime. Fortunately, the method seems quite susceptible to massive parallelization and given current hardware trends it may become feasible pretty soon. Other, heuristic optimization may also be possible.
Bibliography


Author:
MSc. Danijel Koržinek
Polsko-Japońska Wyższa Szkoła Technik Komputerowych
ul. Koszykowa 86
02-008 Warszawa
tel. (+48) 022 58 44 500
fax (+48) 022 58 44 501
email: danijel.korzinek@pjjustk.edu.pl