Automatic Speech Recognition with Radial Basis Function Networks

Adam Soltan, Jr.

soltan@kth.se

Supervisor: Kjell Elenius
Approved: 2005-04-25
Examiner: Kjell Elenius

Examenarbete i Talteknologi
(Master Thesis in Speech Technology)
Department of Speech, Music and Hearing
Royal Institute of Technology
S-100 44 Stockholm
Examensarbete i Talteknologi
Automatiskt igenkänning av tal med hjälp av neuronnät med radiala basfunktioner
Adam Soltan, Jr.
soltan@kth.se

<table>
<thead>
<tr>
<th>Godkänt:</th>
<th>Examiner:</th>
<th>Handleare:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005-04-25</td>
<td>Kjell Elenius</td>
<td>Kjell Elenius</td>
</tr>
</tbody>
</table>
Abstract

The current paper describes an implementation and testing of Radial Basis Function networks as applied to speech data. A new quantization algorithm has been developed for this purpose, called Amber. The results are difficult to interpret but it is the view of the author that they show promise for the future.
Preface

Spoken language is the most natural way for humans to communicate with each other. The progressing computerization of everyday life means that it has become increasingly interesting to be able to communicate also with the computers. Up until quite recently this communication has been hampered by the computers’ inability to interpret human speech. With the advent of high performance digital signal processing for audio applications and the appearance of fairly powerful computers accessible to ever wider groups in the society it has become more cost effective to let the computers understand what we are saying.

Artificial neural networks (ANNs) are using the nervous system as inspiration to perform some tasks, usually different forms of pattern recognition operations. This has made ANNs popular objects for studies and attempted implementations of speech recognition devices. This has met with considerable success paving the way for more studies within this field.

The current paper deals with a study of application of radial basis function (RBF) ANNs to the problem of speech recognition. It describes two algorithms developed for the implementation of an RBF network and also the tests performed using them.

All of the software used for this work has been developed and implemented by the author\textsuperscript{1}.

\textsuperscript{1}... of course with the exception of the operating system, the compiler, the standard C libraries and the Tcl/Tk shell.
# Contents

1 Introduction to Automatic Speech Recognition ........................................ 3  
   1.1 Special Features of the Speech Signal ........................................ 3  
   1.2 Feature Extraction ............................................................... 3  
   1.3 TIMIT Database ................................................................. 6  

2 Artificial Neural Networks .............................................................. 6  
   2.1 The Perceptron and the Adaline ............................................. 7  
   2.2 Backpropagation Networks ..................................................... 8  
   2.3 Radial-Basis Functions Networks .......................................... 8  

3 The Implementation ........................................................................... 10  
   3.1 Preprocessing ........................................................................... 10  
   3.2 Quantization Algorithms ....................................................... 10  
   3.3 Testing Procedure .................................................................. 14  

4 Results ............................................................................................ 15  

5 Discussion ....................................................................................... 15  

6 References ...................................................................................... 18  

A Preprocessing and filtering of the data ............................................ 19  
   A.1 TIMIT to df translation .......................................................... 19  
   A.2 Preemphasis ........................................................................... 19  
   A.3 The mfccfilter program .......................................................... 19  
   A.4 Phoneme tag translation table ............................................... 21  

List of Figures

1. An example spectrogram. ................................................................. 3  
2. Mel filter bank - the concept. ....................................................... 5  
3. Example data: input signal (upper left), Haar filtered data (upper right), DFT (lower left), cepstral coefficients (lower right). ........................................ 6  
4. Perceptron ................................................................................... 7  
5. Adaline ....................................................................................... 8  
6. Multilayer perceptron .................................................................. 9  
7. RBF network ............................................................................ 10  
8. Split data configuration ............................................................... 11  
9. Adjusting the centers according to position and according to activity in amber when $l_d < l_c$ .......................................................... 11  
10. Reassigning the centers in amber when $l_d \geq l_c$ ......................... 12  
11. The hard1 algorithm .................................................................. 13  
12. The hard2 algorithm .................................................................. 13  
13. Control panel written in Tcl/Tk .................................................. 14  
14. Confusion matrix ....................................................................... 16  
15. Telescopic window ..................................................................... 20  
16. The initial ('preneural') signal processing .................................... 21
1 Introduction to Automatic Speech Recognition

1.1 Special Features of the Speech Signal

From a signal processing point of view the speech signal can be characterized as being produced by a source and then filtered by the organs of the vocal tract. Thus we can write that the speech signal can be modeled according to the following equation:

\[ s(t) = x(t) * h(t), \]

where \( s(t) \) is the audio signal, \( x(t) \) is the excitation source \(^2\) signal and \( h(t) \) is the filtering performed by the vocal tract. This produces a signal that can be considered to be a result of a weakly stationary stochastic process.

As far as speech recognition is concerned the interesting data is below 8kHz which puts an upper limit on the sampling rate to 16kHz. For today's electronics it is no real challenge and sampling rates of 32 or even 48kHz are not unheard of. A characteristic of the human auditory system is that it has better frequency resolution toward the lower end of the spectrum.

![Example Spectrogram](image)

Figure 1: An example spectrogram.

A very useful and popular way of visualizing the frequency contents of an utterance is a spectrogram (see Figure 1). In a spectrogram the intensity of different frequency components are plotted against time. The darker the point on the plot the more energy is emitted at the particular frequency. For instance, we can see in Fig. 1 that 'sh' doesn’t show any frequency components below approximately 2kHz and looks generally like noise. On the other hand an 'e' contains some well defined frequency components. Also, we can see that an 'e' has a substantial part of its energy below 1kHz. This is a characteristic of voiced phonemes (which a 'sh'-sound is not and thus has little energy in the low portions of the spectrum).

1.2 Feature Extraction

There is a multitude of ways to record a sound onto digital media. There are also numerous formats in which to store the signal data. All this is, however, beyond the scope of this paper and thus will not be dealt with here.

---

\(^2\)such as the vocal cords
The raw speech signal is not very suitable for direct input into a speech recognizer. Some parts of the signal that are of interest need enhancing in order to improve the performance of the system. An important objective is also data reduction which aims at extracting some easy to work with parameters. The process to extract such parameters is called feature extraction and is commonly implemented not only in automatic speech recognition. For speech recognition one of the most common feature parameter types is the so called cepstrum. It is relatively simple to implement while mimicking the processing that goes on in human auditory system. Here follows a description of how the cepstrum transformation of signal data works.

The main rationale behind the real cepstrum\[^{3}\] is the following. As mentioned above, the speech signal can be represented by the following convolution (substituting analogue 't' for digital 'n'):

\[
s(n) = x(n) \ast h(n)
\]

Here, \(x(n)\) is the excitation and \(h(n)\) is the vocal tract response. Usually, these two components can be assumed to vary with different speeds, \(h(n)\) being the slower changing component. Real cepstrum is defined as:

\[
c_c(n) = 1DFT \{ \log |DFT \{ s(n) \}| \}
\]

where DFT stand for the discrete Fourier transform and IDFT stands for the inverse DFT (more on these later on in this chapter). This definition can, however, be taken a couple of steps further:

\[
c_c(n) = 1DFT \{ \log |DFT \{ s(n) \}| \} = 1DFT \{ \log [X(\omega) \cdot H(\omega)] \} = 1DFT \{ \log [X(\omega)] + \log [H(\omega)] \} = c_x(n) + c_h(n)
\]

This operation separates the two components and since they represent parts of the signal pertaining to higher and lower frequencies respectively they will take up different portions of the frequency axis. This enables us to study them as separate entities.

We start with the raw data representing the sampled speech. The first step is to preemphasize the higher frequencies of the signal. This is done because these frequencies have smaller amplitudes than the lower frequency components while they possess a lot of valuable data. This is most readily done by implementation of a first order FIR filter with transfer function in the z-domain given by:

\[
H(z) = 1 - a \cdot z^{-1} \quad 0 \leq a \leq 1
\]

In the time domain it translates into the following equation:

\[
x'(n) = x(n) - ax(n - 1)
\]

A typical value for \(a\) is 0.95.

Next comes the windowing. In order to get a meaningful discrimination of the frequency variations with time, the data is chopped up in frames usually 10-30 ms long. Smaller windows would decrease the frequency resolution too much, while with wider windows the signal could no longer be assumed to be stationary within the frames. Sometimes two frame width are used in the same implementation in order to discriminate temporally fast signal variations well while keeping good resolution in frequency domain. The frames often overlap each other.

Windowing causes distortions in the signal. The more abrupt changes are introduced to the signal the stronger the distortions. That’s why a rectangular window is not used even though it is the simplest window imaginable. The reason is that implementing such sharp cuts into the data stream introduces a lot of distortion to the signal. A popular choice is the so called Hamming window given by:

\[
w(n) = \begin{cases} 0.54 - 0.46 \cos \left( \frac{2\pi n}{N-1} \right) & n = 0, \ldots, N - 1 \\ 0 & \text{otherwise} \end{cases}
\]

The processing described up until now belongs really to the preparatory stages. Now comes the core part of the transformation of the data: the Fourier transform\[^{4}\]. In this case we use the discrete Fourier transform (DFT) which for an \(N\)-point sequence \(\{x(n)\}\) is defined as:

\[
X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N} \quad k = 0, 1, \ldots, N - 1
\]

where \(j = \sqrt{-1}\).

\[^{3}\]Cepstral analysis is a part of a broader class of methods known as homomorphic signal processing.

\[^{4}\]The Fourier transform is in the author’s opinion the most important engineering tool in use today.
Since we are computing the real cepstrum we only use the magnitudes of the Fourier coefficients. The arguments which contain the phase information in the imaginary output of the Fourier transform are of less interest for speech recognition since the phase generally carries little relevant information.

Next in the processing chain come the Mel scale filters. As mentioned above, the human auditory system has better resolution at lower frequencies. It is thus common practice to use filter banks that produce higher resolution output toward the lower parts of the spectrum. To implement a Mel scale filter bank the data can be translated into the Mel scale and then run through uniformly spaced filters. For the Mel scale we can use the following mapping function:

$$f_{Mel} = 2585 \cdot \log_{10} \left( 1 + \frac{f}{700} \right),$$

where $f_{Mel}$ is the resulting Mel scale frequency value and $f$ is the input frequency. Another commonly used scale is the so-called Bark scale given by the equation:

$$f_{Bark} = 700 \cdot \text{asinh} \left( \frac{f}{700} \right)$$

where $f_{Bark}$ is the resulting Bark scale frequency and $f$ is as above the input frequency. Both scales are based on the human perception of different intensities at different frequencies. Once

![Figure 2: Mel filter bank - the concept.](image)

the frequency data is translated into the new scale the signal is sent to the filters which deliver the energy levels for each interval. The principle is depicted in Figure 2. In this study the number of Mel filters used was 24.

At this point in the processing a logarithm is applied to the data in accordance with the suggested deconvolution of the original signal.

The final stage of feature extraction is the computation of the IDFT. However, since we are dealing only with real values we can use the so-called cosine transform defined as:

$$c_i = \sqrt{\frac{2}{N}} \sum_{k=1}^{N} o_k \cos \left( \frac{n}{N} (k - 0.5) \right)$$

where $N$ is the number of filters and $o_k$ is the output from the filter number $k$. $c_i$ are the cepstrum parameters. Commonly, the first thirteen cepstrum parameters are used starting with $c_0$ which is the amplitude of the frame.
The cepstral parameters of a frame are often compared to the cepstral parameters of the neighboring frames through the application of first and second order time derivatives. This is done in order to catch the short-time dynamics of the signal. The first derivative can be computed using the following formula derived from linear regression analysis:

\[
\overrightarrow{d} = \frac{\overrightarrow{\tau}}{M} (2\overrightarrow{\tau}_{t+2} + \overrightarrow{\tau}_{t+1} - \overrightarrow{\tau}_{t-1} - 2\overrightarrow{\tau}_{t-2})
\]

where \( \overrightarrow{d} \) and \( \overrightarrow{\tau} \) are the derivative and the cepstrum parameter vectors respectively.

1.3 TIMIT Database

The DARPA TIMIT acoustic-phonetic continuous speech corpus (Garofolo, 1993) was used for training and validation of the system. It has been developed under the sponsorship of Defense Advanced Research Projects Agency - Information Science and Technology Office (DARPA-ISTO). It has been collected by Massachusetts Institute of Technology (MIT), Stanford Research Institute (SRI) and Texas Instruments (TI) and prepared for CD-ROM distribution by the National Institute of Standards and Technology (NST). The database used for the current work is that from the prototype version of the CD-ROM.

The corpus contains sentences read by speakers representing 8 major dialect regions of the United States. These are: New England, Northern, North Midland, South Midland, Southern, New York City, Western and Army Brat (persons who have moved around). The speech data is sampled at 16 kHz. For the purpose of the thesis a system for cepstrum parameter extraction has been implemented. More on this later (see 3.1 and A).

2 Artificial Neural Networks

There is a wide variety of different approaches in the field of speech recognition. They can be divided in two main streams: connectionist and mathematical\(^\text{5}\). Connectionist systems are the

---

\(^{5}\)There are of course many other ways to classify the different algorithms.
various kinds of neural networks that take their inspiration from the nervous systems of the animal kingdom. Mathematical methods are represented e.g. by the hidden Markov models which take no notice of how nature works but in stead rely on the powerful mathematical tools to model the underlying functions. As with many branches of science there is no clear cut division between the two approaches. The radial-basis function networks are an example. Even though they are built up of autonomous units, called neurons, their function is really far removed from biological systems and the original inspiration for this type of architecture comes from statistics.

Here follows a presentation of models used for this thesis work.

2.1 The Perceptron and the Adaline

One of the first practical models of a neural unit, called the perceptron, was developed by Frank Rosenblat in the late 1950’s. The original layout got subsequently simplified into its currently most popular version called the single layer perceptron (see Figure 4).

![Figure 4: Perceptron](image-url)

The input data is fed in from the left as the vector $\mathbf{x}$. These inputs are weighted and summed. If the result exceeds a given threshold the perceptron output is set to ‘1’ (i.e. it fires), otherwise to ‘0’. The $\varepsilon$ is the error output and is used when training the perceptron. There is also a so called bias input (feeding in from the top in the figure). Its presence adds to the stability of the training procedure.

Perceptron’s function is to classify vectors into two linearly separable clusters. In order to do this the weight vector, $\mathbf{w}$, must be modified accordingly. The procedure for this is:

$$w(k+1) = w(k) + \lambda \varepsilon(k),$$

where $\lambda \in [0;1]$ is a learning parameter and $\varepsilon = d - y$ is the error output with ‘$d$’ being the desired output and ‘$y$’ being the actual output. Larger $\lambda$ values increase the speed of learning while smaller $\lambda$s improve the accuracy.

Since training of a perceptron requires presenting the unit both with input data and the desired output the training procedure is classified as supervised.

A variation on the perceptron is the adaline\(^6\) (see Figure 5). The difference between the two

---

\(^6\) Originally known as the ADaptive Linear NEuron it later became ADaptive LINear Element when the skepticism toward neural networks was widespread.
Figure 5: Adaline

is where the error signal is taken. The adaline makes the error calculation before the thresholding takes place. This way it also takes into account the magnitude of the error not just its sign. The training procedure is the same as for the perceptron. There are two problems with this approach. Since the error is calculated before the step function there is some ambiguity as to what is the desired output. The current work uses the sign (whether Σ has the same sign as the desired output or not). The other problem is that when only the sign is checked the weights can grow out of bounds, i.e. the relations between the weights are right but the absolute values can grow too large for the computer to be able to handle them. To mitigate this behavior the weights are normalized to 1 between each application of training vectors.

The perceptron and other similar (single layer) configurations have one major drawback. They only work for linearly separable patterns.

2.2 Backpropagation Networks

Backpropagation networks (also known as multilayer perceptrons or MLP) are unrelated to radial-basis function networks but this type of artificial neural networks has been extensively used for speech recognition with considerable success. For this reason a backpropagation network has been used as a reference in this study.

Figure 6 depicts the general layout of a backpropagation network. The units in the hidden and output layers sum the weighted input signals and run them through a sigmoid or other similar non-linear function. The training is supervised and usually requires a fairly large training database.

One of the strengths of MLP is its ability to work with patterns that are not linearly separable.

Since backpropagation network is not the subject of this work there will be no more discussion of its function and properties. The reader is referred to a host of literature on this popular type of neural networks, for example Haykin (1994) and Freeman and Skapura (1991).

2.3 Radial-Basis Functions Networks

The radial-basis function (RBF) networks are very much on the borderline between neural networks and vector quantization. The only real difference is in the way the quantization is performed, i.e. the learning paradigm.
2.3.1 Cover’s Theorem on Separability of Patterns

In 1965 T. M. Cover (Cover, 1965) presented a theory stating that a complex pattern classification problem that does not lend itself to linear separation is more likely to be linearly separable if cast non-linearly in space with higher dimension. Such procedure could for instance be Fourier transforming a signal and thus adding a large number of dimensions to the data in a nonlinear fashion.

2.3.2 Building Radial Basis Functions Networks

The RBF networks derive their name from the fact that they are built around units that can be viewed as a set of basis vectors and that the units usually employ the Gaussian (normal) distribution as the activation function\(^7\).

The data is fed to each RBF unit through unit weights. The weights are being operated on during the training. In the end the weight arrays act as vectors and together with the Gaussians of the RBF units they function as centers of a vector quantizer. The output of the system will ideally show which input vector lies closest to which center making linear separation of patterns a relatively simple task.

When employing unsupervised learning for training of an RBF network the centers are assigned to the data clusters in a random manner. This approach calls for some kind of perceptron like units as output layer\(^8\). For this work an adaline network was mostly used. Once stability is achieved within the RBF layer the supervised training of the output layer can commence.

For supervised training the output layer can be substituted by simpler statistical functions since the outputs of the RBF units are assigned to different phoneme tags from the start and there is less complicated to figure out which phoneme’s center (or centers) are the most active.

\(^7\) which is symmetrical and can thus be seen as having spherical form in spaces with higher dimensions, hence \textit{radial}.

\(^8\) The output layer translates activity levels of the RBF units into phoneme tags.
3 The Implementation

3.1 Preprocessing

The entire preprocessing system including the loading of raw data, the preemphasis and the Fourier transformation with cepstrum analysis have been implemented by the author. The software developed for this purpose constitutes together with the ANN parts proper a very versatile, modular and rather complete lab system for software experiments with neural networks. For more technical details on the preprocessing stages used for this work see Appendix A.

In all the experiments whole utterances were loaded for each run.

3.2 Quantization Algorithms

Two approaches have been tested with varying results. One of the implementations (amber) uses self organization and the other (represented by functions hard 0,1 and 2) uses supervised learning. It seems, however, that they are generally comparable in performance.

3.2.1 Split data configuration

In order to test some aspects of high dimensionality a more flexible scheme was introduced. An input data vector can be split up and the different parts of it can be presented to a selected portion of the RBF units (see Figure 8).

The point was to see whether this approach could influence the result through introducing a measure of control over the dimensionality. Splitting a vector in such a way lowers dramatically the number of degrees of freedom.

3.2.2 Amber and the unsupervised training

The main algorithm tested, called amber, was developed especially for this project. Its main guiding principle was to create a quantization procedure which would require no prior knowledge of the statistical characteristics of the input data.

The algorithm analyses the distances between the input data and the neural units and activity levels of the neurons. In particular it compares the distance between the input vector and the closest unit, $l_d$, with the shortest distance between any two units, $l_c$. If $l_d < l_c^9$ the centers are

---

9 $l_d$ stands for distance to data and $l_c$ stands for distance between closest centers.
adjusted, i.e. the winning center is moved a bit closer to the input vector while all the other units are moved away (see Figure 9a). The winning vector’s activity count is increased\textsuperscript{10}. This is done once for every frame. The magnitude of the adjustment falls off exponentially with the distance from the input vector. At the end of each run, i.e. once per utterance, another move is performed. The least active center is deleted while the most active center is split in two. The two new centers are separated by the average distance between the centers in the entire setup and each receives half the activity count of the deleted center (see Figure 9b). This action will not be performed if the winning center is also the least active one.

The activity counts are subject to decaying once per utterance. This semi Hebbian behavior has a stabilizing effect on the network. The activity count is just the amount a given unit has been a winning one. In the current implementation it is an integer. The decay value is a value by which to multiply the activity counts between every utterance. This way the units connected to the more common phonemes are hindered from reaching excessively high values. For this work a decay value of 0.9 has been used throughout.

\textsuperscript{10}As every run works with whole utterances each center’s activity count can be increased multiple times with each pass.

Figure 8: Split data configuration

Figure 9: Adjusting the centers according to position and according to activity in amber when $l_d < l_c$

If, on the other hand, $l_d \geq l_c$, one of the centers that are the closest to each other will be...
reassigned to the position of the input vector, the assumption being that a new cluster of data has been found. The center that is closer to the input data is assigned the values of the input vector (see Figure 10).

Throughout the project the amber algorithm has proven to be stable and very reliable as far as locating the data clusters is concerned. Even clusters separated spatially by other data sets but belonging to the same data set got classified correctly as belonging to the same set.

![Figure 10: Reassigning the centers in amber when \( l_d \geq l_c \)](https://example.com/figure10.png)

### 3.2.3 Algorithms for supervised training

The main feature of supervised training as opposed to the unsupervised one is that the network is not allowed to organize its units on its own. For instance, in case of phoneme recognition, different phoneme tags are assigned to units in an á priori manner. When input data is presented to the network the desired output is already known and the units are adjusted accordingly. Thus supervised training requires all of the data to be labeled.

**3.2.3.1 Supervised training algorithm #0 (hard0)** This function implements supervised learning of an RBF network. The crudest variety possible. The algorithm looks for a matching id and moves the unit closer to the input data. This means that one unit per possible phoneme tag is the only reasonable thing. This was just a concept demonstrator and was not extensively tested.

**3.2.3.2 Supervised training algorithm #1 (hard1)** Supervised learning of an RBF network, M1. This algorithm looks at the input data and compares the distance to the centers with the desired ids with the closest center in general. If the centers with the desired ids are too far a new center is created (see Figure 11a). The new center gets its coordinates from the input vector. Otherwise the closest center is moved a bit closer to the input data (see Figure 11b). In this way more centers are created near the borders between the cells thus increasing the accuracy where it is really needed. The obvious weakness is that this approach tends to produce inordinate amounts of centers thus making the algorithm rather slow. Fortunately, an equilibrium is reached after a while so the number of centers can be kept within manageable limits.

There is a parameter called the closeness parameter which controls how far the closest center with the right id has to be for a new center to be created. The condition for a new center is that the distance to the closest center with the right id has to be more then average distance between the center of the cell times the closeness parameter. Thus with decreased closeness parameter the accuracy should increase at the expense of an increased number of centers.

**3.2.3.3 Supervised training algorithm #2 (hard2)** This is a modified version of hard1. Here no new centers are created. Instead, if a center is needed, i.e. no center with the right id is close enough to the input data, the least active center is moved to this position. Of course, if the
least active center happens to be the last in an id group it is left alone and the next least active center is utilized. See Figure 12 for graphical cues.

**Figure 11:** The hard1 algorithm

**Figure 12:** The hard2 algorithm

Due to restraints on time the testing of this algorithm is referred to future studies.

### 3.2.4 Output Layer

The output layer used in this study consists of 61 adaline units as there are 61 phoneme tags used in the model employed. (See also A.4.) For obvious reasons only supervised training was used to teach this layer. It is, however, not a part of the RBF network proper.

The training paradigm used for this layer was the popular least-mean-square (LMS) method.

It might sound like a good idea to use a backpropagation network as the output system but since many of the speech recognizers are built entirely on this type of artificial neural networks it seemed like using it would be missing the point.
3.3 Testing Procedure

The training was performed using a random selection of 500 sentences from the TIMIT database. Since there are scores of parameters to tweak\(^\text{11}\) the author put together a control panel in Tcl/Tk (see Figure 13). This decreased the risk of mistakes since it simplified the procedure considerably. The validation used 50 sentences from the above database. These sentences were different from the ones used for training.

Input vectors had 78 dimensions\(^\text{12}\). However, the data was presented as split into smaller chunks according to the method outlined in section 3.2.1. The dimensions tested were 13, 26, 39 and 78.

The recognition value was defined as the percentage of frames for which the phoneme tags were correctly classified. See also A.4.

\(^{11}\)Things like epoch numbers, lambdas, amounts of units, input dimensions, etc.

\(^{12}\)26 cepstrum parameters as the telescopic window was used (see A.3.1) plus first and second order derivatives. See A.3.8 for further details.
4 Results

The original set of the phonemes used to label the TIMIT transcript contains 61 tags. Over time it has turned out the many of them are so similar that they can be merged together resulting in 39 tags. The blueprint for this procedure is outlined in A4. It first appeared in Lee and Hon (1989). In the current work the 61 phoneme-set was used for training and testing while the merging of the phonemes was performed when the statistics were put together.

The results achieved point in the right direction (See Table 1). While 44% is not too impressive there is no doubt that results were not reached by chance (a random picking of tags would in this case result in 2.5%). Studying the confusion matrix in Figure 14 gives a more detailed picture ('sil'

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Results Achieved</th>
<th>Splitting of Input Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>amber</td>
<td>43.8%</td>
<td>2x39</td>
</tr>
<tr>
<td>hard</td>
<td>29.1%</td>
<td>1x78</td>
</tr>
<tr>
<td>BPN</td>
<td>23.9%</td>
<td>1x78</td>
</tr>
</tbody>
</table>

stands for silence). On vertical axis we see the phonemes that were presented to the network and on the horizontal the suggestions given by the system. The darker a square the higher percentage of guesses. For example, 76% of times the tag was sil the guess was correct.\(^{13}\) Can also see that 52% of the times there was a z the network suggested it was an s.

The biggest problem is, however, to decide which configurations offer best results. There is not enough data to clearly point in any direction. It does seem, however, that the number of RBF units should be between 600 and 150 and that the input dimension of the units should not exceed 39. The author wants to stress, however, that very similar setups could end up with very different results. Also, fairly little energy was devoted to the study of how the output layer should be set up.

A note on the seemingly poor results achieved by the BPN. Very little time was devoted to optimization of the BPN. The set up used was merely following the definition of a multilayer perception.

5 Discussion

The results seem to show rather low levels of recognition considering what is usually achieved (between 60% as in Elenius and Takács (1990) and 75% as in Wettschereck and Dietterich (1992)). Since both the RBF networks and the BPN behaved rather similarly it would seem that the quality of data was too low, i.e. the preprocessing stages work poorly or the feature extraction is too simplistic. It is safe to assume, though, that there is something to this system since it is clearly not working at random. On the plus side we can mention that:

- when tags get mixed up they often do so with tags of similar phonemes (g and k, i and iy, s and z, etc)
- some misclassifications may be attributed to errors and ambiguities in the labeling

On the other hand we have that:

- different kinds of 'silences' are very common and since recognizing them is probably easier than the (other) sounds it drives up the recognition statistics

\(^{13}\) Out of all the times the network suggested 'sil' 27.3% of the classifications were incorrect. Considering that some of the most common misses were at phonemes 'b', 'dh', 'k', 'ng' and 't', which are plosives, one might suspect that the misalignment of phoneme labels has a role here.
there does not seem to be any consistency as to which RBF configurations (the number of hidden units, input vector splitting, etc) are the most successful\textsuperscript{14}

Still, it is the view of the author that the main problem lies in the data presented\textsuperscript{15} to the networks rather than with the algorithms tested. It is therefore advisable to devote more time to validate the methods presented in this paper.

\textsuperscript{14}For instance, even though the most successful setup split the input vector in two, the second most successful was trained and tested on vectors split six ways. And the configuration that came third didn’t split the input vectors at all.

\textsuperscript{15}i.e. the feature extraction part of the setup.
6 References


Appendix

A Preprocessing and filtering of the data

The processing of the data is performed by different programs which all can take input from standard input and send their output to standard output. In this way the handling and processing of the data can be made swift and easy.

A.1 TIMIT to df translation.

This project uses DARPA TIMIT acoustic-phonetic speech database prototype. The data represents raw sound waves using integers. The sampling frequency is 16kHz. The data format is called adc. The entire database comprises 4200 sentences read out by different speakers. To translate the adc files into the native data format tmt2df is used.

A.2 Preemphasis.

The preemphasis is implemented in the program preemph. The current implementation performs the following operation on the data:

\[ x'(n) = x(n) - ax(n - 1) \]

The default value for a is 0.95 and this is the value that was used throughout.

A.3 The mfccfilter program.

The final stage before the data is presented to the network is the mfccfilter. This program’s main purpose is to produce cepstral parameters and their derivatives, mfccfilter takes a host of parameters that control its behavior.

A.3.1 The basic and the telescopic window types.

In order to gain a better picture of transient characteristics of data versus more “stable” features a double window can be employed. One of the window components spans over a larger chunk of data than the other, inner window does. This approach is called here a telescopic window. The standard way of doing things, i.e. a single window is the default setting, though. Both types were tested and similar results have been attained. The reader is referred to section 4 which deals more extensively with the achieved results.

A.3.2 The Hamming window.

Only two types of windowing operations are implemented in mfccfilter: the square and the Hamming window. For the Hamming window the impulse response is a raised cosine:

\[ w(n) = \begin{cases} 
0.54 - 0.46 \cos \left( \frac{\pi n}{N} \right) & n = 0, \ldots, N - 1 \\
0 & \text{otherwise}
\end{cases} \]

where \( N \) is the window width. In all the relevant calculations the Hamming window was used. In the telescopic window the Hamming windowing is applied separately to each subwindow.

A.3.3 The FFT.

Perhaps the most fundamental function in all digital signal processing is the digital Fourier transform or its more efficient variant the fast Fourier transform, mfccfilter uses a recursive implementation of radix-2 FFT. In order to compute the cepstral coefficients the absolute value of the FFTs are taken (since we are not interested in the phase information).
A.3.4 Mel filter banks.

Next step in the signal’s road to cepstrum parameters are the Mel filters. The Mel scale is computed according to the formula:

$$f_M = 2595 \cdot \log \left(1 + \frac{f}{700}\right)$$

where $f$ is the frequency, $f_M$ is the Mel transformed frequency and $\log()$ is the 10 logarithm. Banks with 24 filters were used in all the experiments.

A.3.5 The logarithm.

For improvement of the dynamic characteristics of the data and smoothing of the spectrum the data is filtered through natural logarithm. This is a straightforward procedure on every output from the Mel scale filter.

A.3.6 Spectrogram.

mfccfilter can also produce a spectrogram of the signal. The 'graphical' representation is rather simplistic and uses ASCII characters to convey the information. If the telescopic window is used the spectrogram is based on the outer window.

A.3.7 Cosine transform.

Next comes the cosine transform. This stage produces the actual cepstral parameters. The formula used here is:

$$c_i = \sqrt{\frac{2}{N}} \sum_{m=0}^{N} k_m \left(\frac{\pi i}{N} (m - 0.5)\right),$$

where $i$ is the cepstrum parameter order, $N$ is the window width and $k_m$ is the $m$th output from the Mel filter bank (after the logarithm). The standard used here was to calculate 13 coefficients.
including the \( c_0 \). When using the telescopic window each of the subwindows got its own set of cepstrum parameters, which in our case makes a total of 26 parameters.

### A.3.8 The deltas.

The final procedure before the data is ready for the neural processing is to calculate the deltas. The options are not to use any deltas, to include the first derivative and to include the second derivative. All the options were tested in the course of the network operations. For the first order deltas the following formula is applied:

\[
d_t = 0.5(c_{t+1} - c_{t-1})
\]

For the second order deltas \texttt{mfccfilter} uses the following procedure:

\[
d_t^2 = (c_{t+1} - c_t) - (c_t - c_{t-1})
\]

![Diagram of signal processing](image)

**Figure 16:** The initial (‘preneural’) signal processing.

### A.3.9 The passing of the tags.

In order to capture the correct phoneme tag for every window the most common tag within the processed window is passed to the network.

### A.4 Phoneme tag translation table

There were originally 61 phonemes used for the tagging of speech data. Many of these are so similar that when calculating the accuracy of the recognition they were merged together resulting in 39 phoneme tags as in Lee and Hon (1989). For details see Table 2.
Table 2: The reduced phoneme set and the merged phonemes.

<table>
<thead>
<tr>
<th>Phoneme tags</th>
<th>Incorporated phoneme tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>aa</td>
<td>ao</td>
</tr>
<tr>
<td>ae</td>
<td></td>
</tr>
<tr>
<td>ah</td>
<td>ax, ax-h</td>
</tr>
<tr>
<td>aw</td>
<td></td>
</tr>
<tr>
<td>ay</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
</tr>
<tr>
<td>ch</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
</tr>
<tr>
<td>dh</td>
<td></td>
</tr>
<tr>
<td>dz</td>
<td></td>
</tr>
<tr>
<td>eh</td>
<td></td>
</tr>
<tr>
<td>er</td>
<td>axr</td>
</tr>
<tr>
<td>ey</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td></td>
</tr>
<tr>
<td>hh</td>
<td>hv</td>
</tr>
<tr>
<td>ih</td>
<td>ix</td>
</tr>
<tr>
<td>iy</td>
<td></td>
</tr>
<tr>
<td>jh</td>
<td></td>
</tr>
<tr>
<td>k</td>
<td></td>
</tr>
<tr>
<td>sil</td>
<td>bcl, dcl, epi, gcl, h#, kcl, pau, pcl, q, tcl</td>
</tr>
<tr>
<td>l</td>
<td>el</td>
</tr>
<tr>
<td>m</td>
<td>em</td>
</tr>
<tr>
<td>n</td>
<td>en, nx</td>
</tr>
<tr>
<td>ng</td>
<td>eng</td>
</tr>
<tr>
<td>ow</td>
<td></td>
</tr>
<tr>
<td>oy</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td></td>
</tr>
<tr>
<td>r</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td></td>
</tr>
<tr>
<td>sh</td>
<td>zh</td>
</tr>
<tr>
<td>t</td>
<td></td>
</tr>
<tr>
<td>th</td>
<td></td>
</tr>
<tr>
<td>uh</td>
<td></td>
</tr>
<tr>
<td>uw</td>
<td>ux</td>
</tr>
<tr>
<td>v</td>
<td></td>
</tr>
<tr>
<td>w</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td></td>
</tr>
<tr>
<td>z</td>
<td></td>
</tr>
</tbody>
</table>

The label 'sil' was not present in the original set of phonemes.