HMM:s in speech recognition

Emina Karic
D01, Lund Institute of Technology, Sweden
d01ek@efd.lth.se

January 15, 2007

Abstract

This paper describes the work done in my project in ASR and the theory behind it, including signal processing, probability theory, and Hidden Markov Models in speech recognition.

1 Introduction

Speech recognition involves the conversion of a speech signal into a sequence of words in textual format. In practice this process will involve signal processing and decoding as well as the use of a viable language model. The aim of this paper, and the project preceding it, has been to study the mechanisms involved and in a simple and understandable fashion describe the lessons learned.

This project has been limited to a small portion of what needs to be done for isolated word recognition with a small and limited vocabulary. The work that has been done in includes the implementation of the signal processing part, and the decoding, both of which will be explained in more detail in the sections to come.

2 Speech recognition

The different components that make up a speech recognition system will usually include: signal processing, acoustic models, classifier and pattern decoder. In some cases language modeling will also be included, which of course won't be necessary for isolated word recognition.

2.1 Signal processing

The signal processing part of the ASR-system will be responsible for not only the sampling the speech waveform, but also feature extraction, thereby reducing the resources necessary to store and manipulate the representation of the data.

What was studied in the project was the use of cepstral coefficients. Cepstral coefficients can be regarded as features extracted from the signal.

The cepstral coefficients can be found by essentially taking the Fourier analyses of the decibel spectrum which in turn can be computed by taking the log of the FT of the windowed signal. The windowing function is in most cases a Hamming or Hanning window, which will reduce spectral leakage and assure that the signal can be viewed as short-time stationary.

```
FT(log(abs(FT(sig[0:n] * hamming[m]))))
```

In practice this means that each frame, consisting of n samples of data are multiplied by the windowing function sampled at m points and zeroed outside of this interval. This part of the process is illustrated by Figure 2, to be compared with the original data which can be found in Figure 1.

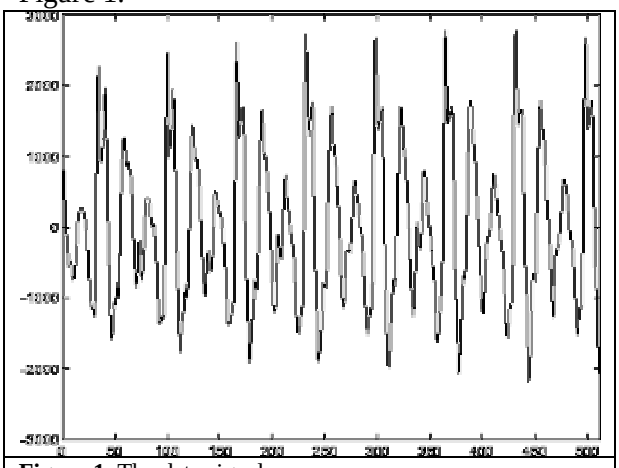

Figure 1: The data signal

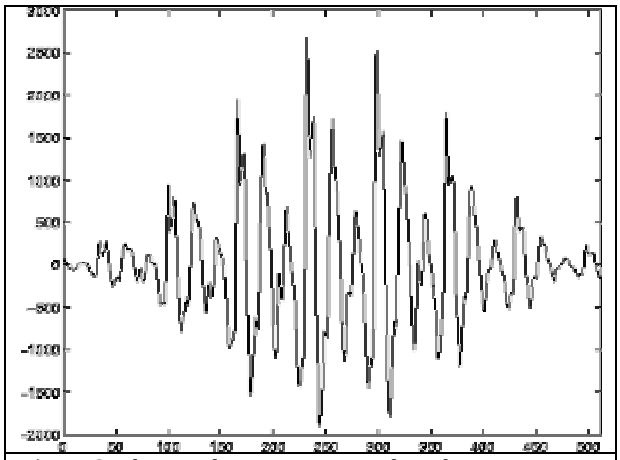

Figure 2: The signal is Hamming windowed to minimize leakage and

Since multiplication in time domain corresponds to convolution in frequency domain, once the Fourier analysis of the signal is carried out we will have the convolution of the signals in frequency domain, as shown in Figure 3.

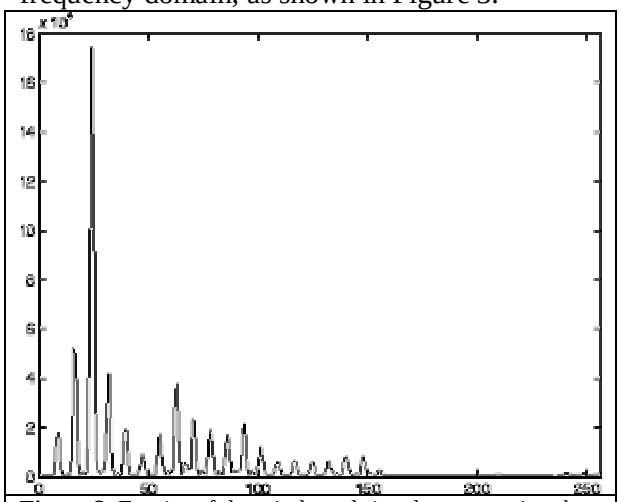

Figure 3: Fourier of the windowed signal, representing the amplitude spectrum.

Taking the log of this signal's absolute value will give us the power spectrum, and a subsequent Fourier analysis will discover the harmonics in the signal, as represented by the cepstral coefficients to be seen in Figure 4.

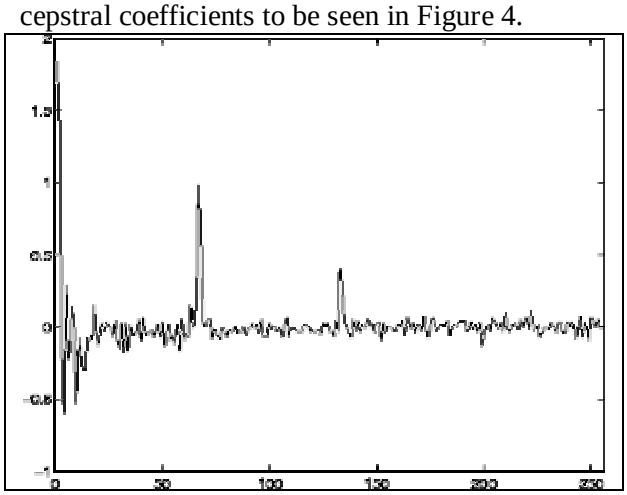

Figure 4: Fourier of the log of the absolute on the signal in Figure 3. This represents the full real cepstrum.

These coefficients can be used to represent the signal in a more compact way, and as input to the next portion of the system.

2.2 Acoustic model

An acoustic model for single word recognition will be based on the recognition of phonemes where each phoneme is characterized by its feature vector. The probability will need to be estimated using some form of training algorithm in combination with relevant training data.

In most speech recognition systems nowadays the acoustic model utilized is the Hidden Markov Model.

The Hidden Markov Model (HMM) is a generalization of the Markov chain. The ordinary Markov chain is a time discrete stochastic process in which the state at time t_{i+1} depends only at the state at time t_i, and the states at all times are observable, as shown in Figure 5. Each transition occurs with a certain probability.

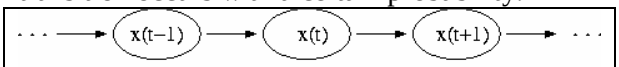

Figure 5: Sequence of states in Markov model

In most applications this will be a simplification, but one necessary to make in order to be able to model the system, and hypothesize about its behavior. In some cases though, as with ASR-systems this won't be enough, and thus a more general model is needed.

The HMM is a system where we have similar conditions as in the Markov chain, but the state it self is not observable. What in stead is observable is the output generated by the state. This output will be dependant on the current state and some output probability for that unobservable state, see Figure 6.

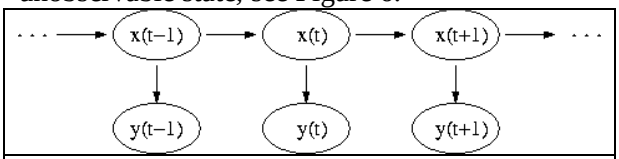

Figure 6: Sequence of states with the associated outputs for a HMM

Any HMM is fully described by the following parameters:

$\prod$ = the vector of initial distributions
A = the transition probabilities
B = the observation probabilities

The parameters are learned using a sequence of observations for which the hidden states are known, and determine the HMM that most likely

generated the sequence. For this purpose the forward-backward algorithm is used. Once the parameters are learned any sequence of observations may be decoded into the sequence of states that most likely generated it using the Viterbi algorithm. An algorithm related to the Viterbi algorithm is the Forward algorithm which determines, given a HMM, the probability of the sequence observed. The forward algorithm is defined as follows:

$\alpha_1(1) = \pi_1 b_1(Y_1)$	$1 \le i \le N$
$\alpha_t(j) = [\sum_{i=0}^{n} a_{t-1}(i)] a_{ij} b_j(X_t)$	$2 \le t \le N$, $1 \le j \le N$
$P(Y \mid \prod, A, B) = \sum_{i=0}^{n} a_T(i)$	

For each possible state sequence of length T, the same as the number of observations, we sum all probabilities given by the product between the total probability of reaching state *i* at time t_{t-1} multiplied bi the transition probability from state *i* to *j* multiplied by the probability of when in state *j* observing the output Y(t).

As mentioned earlier, a related algorithm is the Viterbi algorithm, which finds the most probable path given a set of observations. The definition of the Viterbi algorithm is as follows, where the best sequence is given by S^*:

$V_1(1) = \pi_1 b_1(Y_1)$	$1 \le i \le N$
$B_1(i) = 0$	
$V_t(j) = \underset{1 \le i \le N}{Max} [V_{t-1}(i) a_{ij}] b_j(X_t)$ $B_t(j) = \underset{1 \le i \le N}{Arg\max} [V_{t-1} a_{ij}]$	$2 \le t \le N$, $1 \le j \le N$
Best score $= \underset{1 \le i \le N}{Max} [V_{t-1}(i)]$ $s_T^* = \underset{1 \le i \le N}{Arg\max} [V_T(i)]$	
$s_t^* = B_{t+1}(s_{t+1}^*)$ $S^* = (s_1^*, s_2^*, ..., s_T^*)$	$t = T-1, T-2, ..., 1$

Both of the discussed algorithms naturally require a trained HMM, which brings us to the hardest of problems in the domain of HMM: the forward-backward algorithm. It is more complex in composition than the previously mentioned ones, but in theory what is done is the computation of forward $\alpha_t(i)$ as well as backward $\beta_t(i)$ probabilities as defined below.

$\alpha_t(i) = [\sum_{i=0}^{n} a_{t-1}(i)] a_{ij} b_j(X_t)$	
$\beta_t(i) = [\sum_{i=1}^{N} a_{ij} b_j(X_{t+1}) b_{t+1}(j)]$	$t = T-1, T-2, ..., 1$

Along with these, the probability of at time t transitioning from state *i* to state *j*, $\gamma_t(i,j)$.

$$\gamma_t(i,j) = \frac{a_{t-1}(i) a_{ij} b_j(X_t) b_t(j)}{\sum_{k=1}^{N} a_T(k)}$$

Following this the parameters are reestimated. These new values are $\hat{a}_{ij}$ and $\hat{b}_j(k)$ as given by

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} g_t(i,j)}{\sum_{t=1}^{T} \sum_{k=1}^{N} g_t(i,k)}$$

$$\hat{b}_j(k) = \frac{\sum_{t \in X_t = o_k} \sum_{i} g_t(i,j)}{\sum_{t=1}^{T} \sum_{i} g_t(i,j)}$$

These steps are then repeated until the values converge.

3 Work done and future development

The system described above has in only been implemented in part: signal processing and decoding. What is left to be done is mainly the implementation of the forward-backward algorithm.

References

Xuedong Huang et al. 2001. *Spoken language processing*. Prentice-Hall PTR, New Jersey, USA

http://svr-www.eng.cam.ac.uk/~ajr/SpeechAnalysis/