

Vector interpolation for time alignment in speech recognition

E. A. Yfantis and J. D. Ellison

Department of Computer Science, University of Nevada, Las Vegas, 4505 S. Maryland Pkwy, Las Vegas, NV 89154-4019, USA

Email: yfantis@cs.unlv.edu and elisonj@cs.unlv.edu

Abstract

In this paper, a new algorithm for time alignment in speech recognition is presented. This new algorithm, Vector Interpolation, has a classification rate comparable to that of Dynamic Time Warping. While Vector Interpolation is able to match Dynamic Time Warping for recognition accuracy, it requires significantly less computation, making it much faster than DTW based algorithms. Both algorithms are presented and a comparison of the two is made.

1 Introduction

Automatic speech recognition has applications in many areas. From continuous recognition for dictation systems to isolated word spotting systems for information retrieval and command and control, speech recognition is being used in more and more ways. The ability to successfully apply ASR technology depends on having algorithms that can perform accurate classifications in as efficient a manner as possible. Many portions of speech processing are very computationally intensive and gains in any portion of the system are a benefit to the system as a whole. Alpha-digit recognition is of particular importance for telecommunication applications of ASR for purposes such as hands free dialing of a phone or automatic retrieval of directory information. Current implementations for such systems have been based on either Markov chains or templates and dynamic programming techniques, such as those discussed in Loizou [1]. While Markov based methods and dynamic



programming methods are both capable of making accurate classifications, they are both highly complicated and thus, computationally expensive to implement. This paper presents another alternative, Vector Interpolation, for performing classifications.

2 Background

For the tests of the performance of the algorithms, five recordings of each of the ten digits, zero through nine, were made. The first recording of each digit was used for casual template training and the other four recordings to serve as classification problems. Each recording was sampled at 8 kHz using 8 bit Pulse Code Modulation. The recordings were then processed by a preemphasis filter of the form $H(z) = 1 - \tilde{a}z^{-1}$ where $\tilde{a}$ was defined to be 0.9375. The purpose of this preemphasis filtering is to flatten the spectra.

After performing the preemphasis, the PCM data was broken into analysis frames. Each frame was composed of 300 PCM samples and consecutive frames begin every 100 samples, giving 200 samples of overlap between frames. A Hamming window was used to minimize discontinuity at the start and end of each frame.

Following the windowing, an 11th order autocorrelation analysis is performed for each frame. The autocorrelations were used to compute the Linear Prediction Coefficients using the Levinson-Durbin method.

The LPC values were then used to obtain 16 cepstral coefficients for each frame, using the method described by Rabiner & Juang [2]. The cepstral coefficients were used as the recognition data.

3 The algorithms

The cepstral coefficients represent the feature used to perform the recognition. There is a high degree of similarity between a vector of cepstra from one frame of a recording to the next. There is also a high degree of similarity between the cepstra of different recordings of the same word.

One of the problems to overcome for accurate recognition is the proper time alignment of the recognition data for different utterances. Every time a word is spoken, even if the speaker is trying to minimize the differences in the utterances, there will be some discrepancy in

the time taken to complete the word. In small vocabulary systems, this problem is usually handled by Dynamic Time Warping methods. DTW based systems allow for a time alignment to be computed and based on this alignment, to estimate the difference between two patterns.

3.1 Dynamic Time Warping

Given two speech patterns, $\mathcal{X}$ and $\mathcal{Y}$, these patterns can be represented by a sequence $(x_1, x_2, \dots, x_{T_x})$ and $(y_1, y_2, \dots, y_{T_y})$ where x_i and y_i are the feature vectors, in our case by the cepstra found via the LPC analysis. As we have noted, in general the sequence of x_i 's will not have the same length as the sequence of y_i 's. In order to determine the distance between $\mathcal{X}$ and $\mathcal{Y}$, given that some distance function $d(x, y)$ exists, we need a meaningful way to determine how to properly align the vectors for this comparison. Dynamic Time Warping is one way that such an alignment can be made. We define two warping functions, ϕ_x and ϕ_k , which transform the indices of the vector sequences to a normalized time axis, k . Thus we have,

$$i_x = \phi_x(k), \quad k = 1, 2, \dots, T \quad (1)$$

and

$$i_y = \phi_y(k), \quad k = 1, 2, \dots, T. \quad (2)$$

This gives us a mapping from $(x_1, x_2, \dots, x_{T_x})$ to $(x_1, x_2, \dots, x_T)$ and from $(y_1, y_2, \dots, y_{T_y})$ to $(y_1, y_2, \dots, y_T)$.

With such a mapping, we are able to compute $d_\phi(\mathcal{X}, \mathcal{Y})$ using these warping functions, giving us the total distance between two patterns as

$$d_\phi(\mathcal{X}, \mathcal{Y}) = \sum_{k=1}^T d(\phi_x(k), \phi_y(k))m(k)/M_\phi \quad (3)$$

where $m(k)$ is a path weight and M_ϕ is a normalization factor. See Figure 1 for an example of such a time alignment.

Thus, all that remains is the specification of the path ϕ indicated in Equation 3. The most common choice is to specify that ϕ is the minimum of all possible paths, subject to certain constraints, giving

$$d(\mathcal{X}, \mathcal{Y}) \triangleq \min_{\phi} d_\phi(\mathcal{X}, \mathcal{Y}). \quad (4)$$

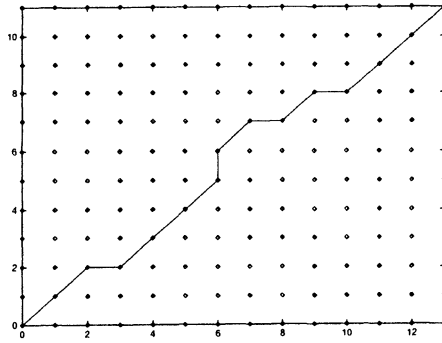


Figure 1: A time alignment using a warping function.

For a more complete description of the Dynamic Time Warping, see Sakoe [3].

Dynamic Time Warping has been very successful at making appropriate time alignment for recognition. However, DTW based algorithms are expensive. Although careful optimization can reduce the computational complexity, a DTW based algorithm will still essentially require $O(n^2)$ distance computations for finding the proper time alignment.

3.2 Vector Interpolation

We propose an alternative to the use of DTW based algorithms. Despite the fact that DTW based algorithms perform accurate recognition, it is desirable to reduce the computational complexity. As mentioned previously, DTW based algorithms require $O(n^2)$ distance computations to perform an alignment. While optimization techniques can be used to reduce the search space for the optimal alignment, they produce only a linear factor of improvement. The Vector Interpolation technique that we propose can perform a time alignment and estimate the distance between two sequences with $O(n)$ distance computations.

The Vector Interpolation algorithm uses a linear time normalization, defined as

$$d(\mathcal{X}, \mathcal{Y}) = \sum_{i_x=1}^{T_x} d(x_{i_x}, y_{i_y}), \quad (5)$$

where i_x, i_y are related by

$$i_y = \frac{T_y}{T_x} i_x. \quad (6)$$

This establishes the time normalization. However, most values of i_y will be real numbers, rather than integers. Since i_y is an index into a sequence of vectors for $\mathcal{Y}$, how can i_y be interpreted? Previous treatment of this problem in the literature has dealt with it by using a round off rule. We propose the novel solution of data interpolation. A vector at time i_y can be interpolated from the adjacent, known vectors by the following equation

$$y_{i_y} = y_{\lfloor i_y \rfloor} \frac{T_y}{T_x} + y_{\lceil i_y \rceil} (1 - \frac{T_y}{T_x}). \quad (7)$$

The use of this technique increases the computational load by requiring the generation of the interpolated vector. However, this can be done with $O(n)$ multiplications and additions. Then, only $O(n)$ distance comparisons need be calculated, rather than the $O(n^2)$ distance comparisons that would be required for a DTW technique. This represents significant savings.

4 Results

As was outlined in the first section, five recordings of each of the digits were made, giving a total of 50 recordings. For each digit, the first recording served as a model and the other four as classification problems. In each implementation, the cepstral data was pre-computed and read from a file for comparisons.

Both the Dynamic Time Warping implementation and the Vector Interpolation algorithm were able to correctly classify all of the 40 test cases. On average, the Vector Interpolation algorithm took only 0.25 milliseconds to make a classification, while the Dynamic Time Warping algorithm took 4.64 milliseconds to make a classification. Thus, a classification using the DTW algorithm took over 18 times as long to complete as a classification using the Vector Interpolation algorithm, yet Vector Interpolation was able to make the classifications just as accurately.

Both algorithms are successful for making the time alignments, as evidenced by the 100% success rate each exhibited. However, the

Vector Interpolation algorithm is more than an order of magnitude improvement in the time required to make the classification. While further experimentation is necessary to ensure the robustness of the Vector Interpolation algorithm, these preliminary results suggest that it is far superior to Dynamic Time Warping Techniques.

References

- [1] Loizou, P. & Spanias, A., High performance alphabet recognition, *IEEE Trans. on Speech and Audio Processing.*, **4**(6), pp. 430-445, November, 1996.
- [2] Rabiner, L., & Juang, B., *Fundamentals of Speech Recognition*, Prentice Hall, New Jersey, 1984.
- [3] Sakoe, H. & Chiba, S., Dynamic programming optimization for spoken word recognition, *IEEE Trans. on Acoustics, Speech, and Signal Processing.*, **34**(1), pp. 43-49, February 1978.