Proceedings of The Institute of Acoustics 7D2

DYNAMIC PROGRAMMING VARIATIONS IN AUTOMATIC SPEECH RECOGNITION

R K MOORE

ROYAL SIGNALS AND RADAR ESTABLISHMENT, MALVERN

INTRODUCTION

Non-linear time warp pattern matching by dynamic programming has established itself as one of the most useful tools available for constructing an automatic speech recogniser. Using such techniques, present day recognisers achieve significantly better performance than their predecessors. This paper describes three basic dynamic programming algorithm variations. In the first, the cost of time compression/expansion is simply set to a constant value. In the second, this cost is made a function of the relationship between the items being matched. In the third, the cost is related to the rate at which the speech spectrum shape changes in time. In addition, two efficiency modifications are described which speed up the matching process.

DYNAMIC PROGRAMMING ALGORITHMS

The three dynamic programming algorithm variations represent three ways of expressing the non-linear time scale distortion necessary to match one speech sample with another. The simplest algorithm (1) states that the cost of distorting the time scale is constant per unit time. Mathematically the algorithm is expressed as:

$$g(i, j) = \min \begin{Bmatrix} g(i-1, j) & + K \\ g(i-1, j-1) & + d(i, j) \\ g(i, j-1) & + K \end{Bmatrix} \quad \ldots\ (1)$$

where $d(i, j)$ is the distance between the i'th frame of one speech sample and the j'th frame of the other (where a frame might be a spectrum), and $g(i, j)$ is the cumulative distance between the first i frames of the first sample and the first j frames of the second sample. If i is in the range 1 to I and j is in the range 1 to J, then $g(I, J)$ is the distance between the two speech samples. K is the cost of distorting the time scale by one frame time unit. When $K = \emptyset$, complete time distortion is allowed, and anything will match with anything. As K gets larger, so the amount of time scale distortion allowed gets progressively lower. This algorithm is equivalent to the one presented by Velichko and Zagoruyko [2].

The second algorithm (2) states that the cost of distorting the time scale at a particular instant of time within a speech sample is a function of the relationship between the two speech samples at that instant.

$$g(i, j) = \min \begin{Bmatrix} g(i-1, j) & + & d(i, j) \\ g(i-1, j-1) & + K* & d(i, j) \\ g(i, j-1) & + & d(i, j) \end{Bmatrix} \quad \ldots\ (2)$$

This algorithm is equivalent to the one used by White and Neely [3] when K = 1. When K = 2 it is equivalent to the best algorithm described by Sakoe and Chiba [1].

The third algorithm (3) states that the cost of time scale distortion is a function of rate of change of spectrum of the speech sample. That is, if the spectrum is changing slowly (for example, within a vowel, nasal, fricative or silence region) then time distortion may take place with little cost. If, however, the spectrum is changing rapidly (for example, during stop or CV/VC transitions) then the time scale must be preserved.

$$g(i, j) = \min \left\{ \begin{array}{l} g(i-1, j) + K^* s(i, i-1) \\ g(i-1, j-1) + d(i, j) \\ g(i, j-1) + K^* s(j, j-1) \end{array} \right\} \quad \ldots\ (3)$$

s(i, i-1) is the distance between the i'th frame of one speech sample and the previous frame of the same speech sample.

For each of the three algorithms, the overall similarity score calculated may be normalised by the lengths of the speech samples being matched. This is done by dividing g(I, J) by I + J.

Each algorithm may also be subjected to an efficiency modification whereby a restriction is placed on the values of i and j. In the first modification, i and j are restricted to values which maintain g(i, j) within a distance W of the diagonal of the matrix G. With high values of W i and j can take any value. With low values the calculations will be confined to the diagonal region. Hence by varying W the algorithms can be changed from linear time warping to less and less restricted non-linear time warping. Also, since fewer distances have to be calculated for smaller values of W so the faster the algorithm becomes.

The second efficiency modification is equivalent to the method known as 'beam search'. In this case the restriction is made dependent upon the quality of the partial matches in the matrix G. It is effected by thresholding each row in the matrix such that only values of g(i, j) less than the lowest value in that row plus the variable T are retained. This means that there is effectively a low value of W in regions where there is a good match between the speech samples, and a high value where there is uncertainty.

EXPERIMENTS

To investigate the behaviour of these algorithms, two sets of data were used in a series of recognition experiments. The first set consisted of 22 repetitions of the ten digits (zero-nine) spoken in isolation, and the second set comprised 25 repetitions of the eight cardinal vowels. Both sets were spoken by a phonetician and the parameterisation was effected by a 16 channel filter bank using a 10 ms frame rate. One example of each digit and one example of each vowel were taken from the centre of the recordings and used as templates.

Each algorithm (normalised and unnormalised) was tested over a range of values of K, W and T. Over 400 experiments were conducted, most of which

each took two hours of computer time.

RESULTS

The results for variations in K are as follows:

For algorithm 1, normalised digits peak at K = 50 and 60 with a recognition rate of 96.4%. At this value of K the unnormalised digits achieve 83.2% rising to 93.2% at K = 140. Normalised vowels also peak at K = 50 and 60 and again at K = 110 and 120 with a recognition rate of 98.5%. Unnormalised vowels are marginally better with a rate of 99.0% at K = 40 and 50. Thus, the optimum value for K appears to be 50.

For algorithm 2, normalisation makes little difference with values of K greater than 15. In the region K = 1 to K = 15 normalised digits gradually increase in performance from 91.8% at K = 1 (White and Neely), 92.3% at K = 2 (Sakoe and Chiba) to a peak of 94.1% at K = 7 and 9. Unnormalised digits are fairly constant in this region with a minor peak of 91.8% at K = 1 and 10. For vowels, both the normalised and the unnormalised versions suffer a drop in performance in the K = 2 and 3 region. Normalised vowels attain 93.0% at K = 1 and this drops to 84.0% at K = 2. Performance then recovers to reach a peak of 92.0% at K = 10. Unnormalised vowels fare slightly worse at the peaks of K = 1 and 10 with a rate of 91.8%, but the figure drops to only 86.5% at K = 2. For valuesof K less than 1, all performance suffers a steady decline. The optimum value for K is thus 1.

For algorithm 3, normalisation improves the performance for digits over most values of K. Above K = 8 normalisation also improves vowel recognition, but below K = 8 it depresses the performance. Normalised and unnormalised digits peak at K = 8 with recognition rates of 96.4% and 93.6%. Normalised vowels peak at K = 4 and 8, and unnormalised vowels peak at K = 2, 4 and 8 all with a rate of 97.5%. The overall optimum is thus K = 8.

Comparing the three algorithms at optimum values of K, algorithm 1 comes out best (digits 96.4%, vowels 98.5%), with algorithm 2 slightly worse (96.4%, 97.5%), and algorithm 3 some way behind (94.1%, 93.0%).

For the variations in W and T only the normalised versions of the algorithms were tested at optimum values of K. The results for W are as follows:

For algorithm 1, as W is reduced below 25 the recognition rate for the digits drops off with a slight peak at W = 3 (linear time normalisation). For vowels the recognition rate is maintained as W is reduced to 3 where there is a slight peak to 99.0%.

For algorithms 2 and 3 the behaviour is much the same as for 1. None degrade until W is less than 30, hence each can be speeded up by a factor of 1.6 without loss of recognition accuracy. In general, digits fair worse than vowels as W approaches linear time normalisation.

The tests for T were only applied to algorithms1 and 2. For algorithm 1, both digit and vowel scores fall at approximately the same rate as T is reduced below 1700. A test revealed that this value of T enabled the whole matrix to be included in the beam search. Hence this variation can provide no increase

in search efficiency.

For algorithm 2, the vowel performance degraded with lower values of T in much the same way as with algorithm 1. However, the digit score increased to 92.7% at T = 1000 and then remained above the 91.8% obtained at large values of T down to T = 600. Below this value the score falls. The same test revealed that at T = 800 this algorithm was achieving a speed up factor of 4.

CONCLUSIONS

Three dynamic programming algorithms have been shown to be open to optimisation in terms of recognition accuracy and speed of computation. For accuracy the algorithm based on a constant time distortion cost is comparable with the one based on a cost related to the rate of change of spectrum. Both algorithms are shown to be superior to the one based on either White and Neely's or Sakoe and Chiba's. All algorithms can be speeded up by a factor of 1.6 without loss of accuracy by using a simple restriction on the dynamic programming matrix. However, White and Neely's variation can be speeded up by a factor of 4 using beam search.

REFERENCES

1. H SAKOE and S CHIBA 1978 IEEE ASSP 26, 43-49. Dynamic Programming Algorithm Optimisation for Spoken Word Recognition.

2. V VELICHKO and N ZAGORUYKO 1970 Int J Man-Machine Studies, 2, 223-234. Automatic Recognition of 200 Words.

3. G WHITE and R NEELY 1976 IEEE ASSP 24, 183-188. Speech Recognition Experiments with Linear Prediction, Bandpass Filtering, and Dynamic Programming.