DURATION MODELING RESULTS FOR AN ON-LINE HANDWRITING RECOGNIZER

Andrew Senior, Jayashree Subrahmonia and Krishna Nathan

IBM T.J. Watson Research Center, P.O. Box 218,
Yorktown Heights, NY 10598, USA.

ABSTRACT

This paper describes a series of experiments that have been conducted to investigate the effect of duration modeling in a hidden Markov model (HMM) based on-line handwriting recognition system. The issues discussed include parametric vs. non-parametric distributions to model duration, different methods for training transition probabilities and the effect of weighting on duration terms.

1. INTRODUCTION

A variety of approaches have been taken to the problem of recognizing on-line handwriting — handwriting recorded in the form of coordinates on the time-sampled trajectory of an electronic stylus. We have developed a system based upon hidden Markov models [3], a method successfully applied to the recognition of other time series signals, notably speech.

The goal of the recognizer is to find the word W that has the maximum *a posteriori* probability given the observed output D, i.e., the decoder chooses the word W such that

$$\hat{W} = \arg\max_W p(W \mid D) = \arg\max_W p(D \mid W)p(W) \quad (1)$$

In a hidden Markov model system, we compute $p(D \mid W)$ by considering the likelihoods of different paths through the model. Different paths are represented by *state sequences* that show which frames of the data, D, are associated with which states of the model. The set of possible state sequences for D in the model, M_W, for word W, is denoted by $\mathcal{S}_{M_W}$. Thus we obtain an estimate, $\hat{W}$, for the correct word as follows

$$\begin{aligned} &\hat{W} = \\ &\arg\max_W p(D \mid W)p(W) = \arg\max_W p(D \mid M_W)p(W) \\ &p(D \mid M_W) = \sum_{S \in \mathcal{S}_{M_W}} p(D \mid S)p(S \mid M_W), \end{aligned} \quad (2)$$

or for the Viterbi decoding used here,

$$p(D \mid M_W) \approx \max_{S \in \mathcal{S}_{M_W}} p(D \mid S)p(S \mid M_W). \quad (3)$$

The data likelihoods, $p(D \mid S)$, are generated from a mixture-of-Gaussians model described elsewhere [2]. This paper deals with the calculation of $p(S \mid M_W)$.

Very often a simple duration modeling scheme is used in a HMM, which determines $P(S \mid M_W)$ as the product of transition probabilities, one for each frame of input data, the probability values being determined by the model states where the transition associated with a frame starts and ends. This gives a computationally simple way of calculating $P(S \mid M_W)$. However, a variety of researchers in both speech and handwriting have found that imposing a more complex duration model, with characteristics closer to that of the system generating the data, gives improved recognition results [4, 5, 1].

We have investigated the application of duration models to our handwriting recognition system, both in extending the simple duration model to a more complex, histogram-driven model, and by investigating a variety of ways of estimating that histogram.

2. SYSTEM OVERVIEW

The IBM unconstrained, on-line handwriting recognizer is a hidden Markov model-based system. The system has been described more fully elsewhere [2], but here we give a brief overview of the system, concentrating on the aspects that are affected by duration modeling.

The system recognizes words in a two stage process. The first stage uses simple, fast models with one state per letter to search quickly through the lexicon, resulting in a short list of likely words. These words are then passed to the detailed match model that has multi-state models for each letter. The detailed match model rescores the short list and the most likely word is chosen as the answer. Results presented here are the first word recognition accuracy for either the fast match model alone or in combination with the detailed match. All results presented are for isolated words which include punctuation and capitalized words; a 22,000 word lexi-

con is used, but the system's stochastic language model is not. Except where described, the system is tested on data written by writers not in the training set.

Most commonly, with a single-state model, simple transition probabilities are used, determining the probabilities of remaining in a state or leaving at each time instant. This is known to give a geometric duration distribution, which does not match the observed durations of letters. To overcome this, in the fast match model, characters are expanded for multiple frames ignoring transition probabilities, e.g., we hypothesize a letter '*a*' with a duration of five frames. We then multiply the data probability by a duration probability for the entire character. This probability is estimated from a histogram of observed lengths or from a parametric distribution.

The detailed match similarly hypothesizes multiple-frame character instances, but the most likely state sequence within the multi-state model for a character is found by the Viterbi algorithm. Here, the more conventional duration model based on transition probabilities is used. Figure 1 shows a single letter model. The transition probabilities of the three transitions from each state can be set independently to give the desired duration characteristics. Because of the complexity of this model, the duration characteristic is no longer the geometric distribution. In fact, it can be seen that if $p(\text{next}) > \text{p(self)}$ and $p(\text{next}) > \text{p(null)}$, then the most likely duration is the same as the number of states in the model, and the duration distribution will peak at this value.

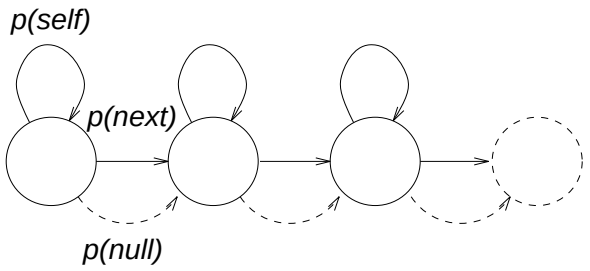

Figure 1: A three-state detailed match model showing the transition probabilities for the first state. The dashed null transition consumes no input frame. The first state of the next model is shown dashed.

3. FAST MATCH IMPROVEMENTS

The fast match model requires a non-parametric duration distribution to calculate the probability of each state sequence. The distribution gives the probability of each letter lasting a given number of frames. By changing the way we calculate the duration distribution, we can improve the recognition performance of the model.

3.1. Lexeme duration modeling

It can be seen that even within a single writer's handwriting, a given letter may be written in different ways. For instance, letters may have different ligatures depending on the surrounding letters, or a capital '**I**' may be written differently at the start of a word, in the middle of an acronym or for the personal pronoun. When looking at a pool of writers, letter shape variability increases tremendously, but we may be able to identify a few, commonly used, standard shapes for each letter. We term these shapes *lexemes* and build a separate model for each, rather than clumping the heterogenous shapes together in a single letter model.

Table 1 shows the recognition results when duration characteristics are shared across all lexemes for a given character, and when lexeme durations are modelled separately. As seen from the Table, modeling is improved (at no speed cost and negligible storage increase) if lexemes are modelled separately.

Fitting method	Accuracy (%)
Character	68.9
Lexeme	69.7

Table 1: A comparison of fast-match accuracies with character and lexeme-based duration models.

3.2. Parametric distributions

The fast-match duration distributions are all created from histograms of observed durations from a corpus of data. The observed statistics are of course noisy, especially when there is a limited amount of training data on which to estimate the distributions, such as in writer-dependent training. When data are limited, better distributions can be obtained by fitting a parametric distribution to the observed data. This smooths the noisy observed counts and can provide more robust recognition.

We have compared normal, gamma and Poisson distributions to the original histogram. It can be seen that the Poisson distribution most closely matched the desired distribution, giving improved performance over the un-smoothed histogram. The other distributions did not perform as well.

Fitting method	Accuracy (%)
Histogram	80.4
Normal	77.2
Gamma	77.9
Poisson	81.6

Table 2: A comparison of parametric distribution fits to lexeme durations. Accuracy results are for a fast match model trained and tested on independent sets of writing by the same person.

3.3. Discrete vs. cursive data

The basic system used only discretely written data (i.e. characters written in isolation) to estimate the character length duration statistics since segmentation of a cursive word into its component characters is difficult, and the lengths of the characters are thus unknown. However, when a model has been trained, the maximum likelihood state sequence through the correct model gives a segmentation that can be used to collect statistics for the durations of letters in cursive words. We now have the option of combining statistics from the two sources (discrete and cursive) to give better estimates of the distributions. In our pre-processing, frames are generated at turning points of the pen trajectory. Different numbers of frames are generated for letters if they are discretely written or written as part of a cursive string, because of differences in ligatures and end-points. We find that estimating the duration histograms from discrete data alone gives us the best performance on a test set of mixed discrete and cursive writing.

4. DETAILED MATCH IMPROVEMENTS

4.1. Baseform length

Experiments have shown that the rule for choosing the number of states in the detailed match models significantly affects recognition performance. In many handwriting and speech recognition systems, the number of states in a model is fixed for all models [6]. However, we have found a marked improvement by adjusting the number of states to match that observed in practice. Future experiments will compare a variety of methods of choosing the number of states, including choosing the mean or mode length of observed lexemes, as well as stretching or shrinking the baseforms. Shrinking the baseform has the advantage of reducing computation and memory usage, but extended baseforms avoid the use of the self-transitions which conveniently model stationarity in speech systems, but have no such interpretation with the features that we use.

4.2. Trained transition probabilities

Other users [7] of hidden Markov models find that the transition probabilities are unimportant, and that training for the parameters does not improve performance. With limited training data, this may be the case, but we find that training does improve the accuracy. We compared the effect of uniform, untrained distributions with distributions chosen before training and then fixed, and with fully-trained distributions.

In the uniform distribution case, $p(\text{self}) = p(\text{next}) = p(\text{null}) = 1/3$

The 'chosen' distributions were chosen to have the expected duration for each lexeme model equal to the mean observed duration of that lexeme. The expected value of the duration per state can be computed as follows

$$\begin{aligned} &E[\textit{duration per state}] = \\ &0.\text{p(null)} + \\ &1.(p(\text{next}) + \text{p(self)p(null)}) + \\ &2.(p(\text{self})\text{p(next)} + \text{p}^2(\text{self})\text{p(null)}) + \\ &3.(p^2(\text{self})\text{p(next)} + \text{p}^3(\text{self})\text{p(null)}) + \\ &\ldots \\ &= \tfrac{p(\text{next})+\text{p(self)p(null)}}{(1-\text{p(self)})^2} \end{aligned} \tag{4}$$

Since, $p(\text{next}) + \text{p(self)} + \text{p(null)} = 1.0$,

$$E[\textit{duration per state}] = \frac{p(\text{next}) + \text{p(self)}}{(1 - \text{p(self)})} \tag{5}$$

Given the mean observed duration per state from the training data, one can compute $p(\text{next})$, $p(\text{self})$ and $p(\text{null})$ by choosing a value for one and using the relationships shown above.

Assignment method	Accuracy (%)
Fixed, uniform values	76.2
Fixed, chosen values	77.6
Fixed, estimated values	78.3
Trained	79.6

Table 4: Detailed match recognition scores according to transition probability training method.

4.3. Weighting

In both the fast match and detailed match models, we combine the state sequence probabilities from the duration model with the data likelihoods from the mixture-of-Gaussians model. Although strictly, we should simply multiply the probabilities, in the past it has been

FM Weight	DM weight.						
	0.0	0.5	0.75	1.0	1.5	2.0	3.0
0.0	71.9	76.3	77.1	77.4	77.6	77.7	76.6
0.5	73.4	76.6	77.0	77.3	77.8	77.3	76.4
0.75	73.4	76.7	77.2	77.4	77.8	77.5	76.6
1.0	73.6	76.9	77.5	77.6	77.9	77.7	76.8
1.5	74.0	76.9	77.5	77.7	77.7	77.9	
2.0	74.1	77.1	78.0	78.1	77.9	78.0	76.9
3.0	73.8	76.6	77.3	77.4	77.8	77.4	76.3

Table 3: Detailed match recognition scores when decoding with different duration model weighting factors.

found [5] that weighting the different types of probabilities differently can improve results. This can be interpreted as giving a confidence measure to these probability estimates, weighting more heavily those probabilities we consider to be better estimated. With a duration model weight α, equation 3 becomes:

$$p(D \mid M_W) \approx \max_{S \in \mathcal{S}_{M_W}} p(D \mid S) p(S \mid M_W)^{\alpha}. \quad (6)$$

The weighted probabilities are normalized to sum to one. In this study, we have experimented with weights for both the fast and detailed match probabilities. Table 3.3 shows the results as these are changed independently. A small improvement is obtained by weighting the fast-match duration probability.

5. CONCLUSIONS

This paper has reviewed a number of techniques for duration modeling in a hidden Markov model system. Duration models can be seen to have a significant effect on the overall recognition rate of the system, affecting both the detailed match and fast match performances. Improvements in the system performance have been achieved by modeling lexemes separately, by using a histogram technique for the fast match probabilities and by weighting these probabilities in relation to the data likelihoods. The detailed match performance has been improved by adjusting the length of the detailed match models and by training the transition probabilities of these models.

Acknowledgments

The authors would like to thank the other members of the IBM Handwriting recognition group.

6. REFERENCES

[1] M. M. Hochberg. *A Comparison of State-Duration Modelling Techniques for Connected Speech Recognition.* PhD thesis, Division of Engineering, Brown University, October 1992.

[2] K. S. Nathan, H. S. M. Beigi, J. Subrahmonia, G. J. Clary, and H. Maruyama. Real-time on-line unconstrained handwriting recognition using statistical methods. In *ICASSP95*, volume 4, pages 2619–2623, 1995.

[3] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP magazine*, 3(1):4–16, January 1986.

[4] M. Schenkel, I. Guyon, and D. Henderson. On-line cursive script recognition using time delay neural networks and hidden Markov models. In *International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 637–640, 1994.

[5] A. W. Senior. *Off-line Cursive Handwriting Recognition using Recurrent Neural Networks.* PhD thesis, Cambridge University Engineering Department, September 1994.

[6] T. Starner, J. Makhoul, R. Schwartz, and G. Chou. On-line cursive handwriting recognition using speech recognition techniques. In *International Conference on Acoustics, Speech and Signal Processing*, volume 5, pages 125–128, 1994.

[7] Y. Zhao. A speaker-independent continuous speech recognition system using continuous mixture gaussian density HMM of phoneme-sized units. *IEEE ASSP Magazine*, 1(3):345–361, July 1993.