EM MIXTURE MODEL PROBABILITY TABLE COMPRESSION

Sung-Jung Cho[*]*, Michael Perrone and Eugene Ratzlaff*

IBM T.J. Watson Research Center,
Yorktown Heights, NY 10598, USA
sjcho@ai.kaist.ac.kr, {mpp, ratzlaff}@us.ibm.com

ABSTRACT

This paper presents a new probability table compression method based on mixture models applied to N-tuple recognizers. Joint probability tables are modeled by lower dimensional probability mixtures and their mixture coefficients. The maximum likelihood parameters of the mixture models are trained by the Expectation-Maximization (EM) algorithm and quantized to one byte integers. The probability elements which mixture models do not estimate reliably are kept separately. Experimental results with on-line handwritten UNIPEN digits show that the total memory size of an N-tuple recognizer is reduced from 11.8M bytes to 0.55M bytes, while the recognition rate drops from 97.7% to 97.5%.

1. INTRODUCTION

There is a continuing public desire for improved handwriting recognition (HWR) systems, particularly for handheld devices such as PDAs and smart phones. Such embedded HWR systems should provide high accuracy and real-time speed with a small memory footprint. The scanning N-tuple recognizer [1] has demonstrated the potential for excellent speed and accuracy for on-line HWR [2], but consumes significant memory resources. This paper describes methods for significantly reducing the memory use of the SNT through the use of mixture models.

1.1. The Scanning N-Tuple Recognizer

The scanning N-tuple (SNT) technique is applicable to variable length discrete feature sequences (e.g. chain codes). The probability of observing the complete feature sequence is represented by the joint probability of observing all of the partial feature sequences, where training data are used to generate look-up tables of the estimated probabilities of each partial feature sequences. A handwritten character is recognized by choosing the class that yields the highest joint probability.

[*]Current address: Sung-Jung Cho, CS Div., EECS, KAIST, Kusong-dong, Yousong-ku, Daejon, 305-701, Korea

For each character sample of a character class C, the SNT algorithm generates a variable length sequence of features, $f_1, \ldots, f_L$. We define the i-th N-tuple of a given feature sequence to be

$$X^i_{1,N} = (f_{i+k}, f_{i+2k}, \ldots, f_{i+Nk}) \quad (1)$$

where $i = 1, \ldots, L - Nk$, and k is the subsampling distance. The SNT assumes that the N-tuples are all independent, thus the probability of observing a given sequence of N-tuples is given by

$$P(\cup_i X^i_{1,N}|C) = \prod_i P(X^i_{1,N}|C). \quad (2)$$

The joint probability $P(X_{1,N}|C)$ is modeled by a lookup table of the normalized frequency counts of each of the possible N-tuples observed in the N-tuples for all the data for a given class C. For the remainder of the paper, we will take the conditioning on C as given and simply use $P(X^i_{1,N})$.

1.2. Compression of Joint Probability Tables

As with the SNT, conditional and joint probability tables are incorporated in many other on-line handwriting recognition systems for representing relationships between discrete random variables. N-gram language models and Bayesian networks are two such examples. One of the practical problems with such tables is that the table size grows exponentially with the number of random variables.

When such joint probability tables must be compressed, three factors should be considered. First, a compression algorithm should have a high compression ratio. Second, it should not severely degrade recognition accuracy. Third, it should not slow the recognizer so as to compromise real-time responsiveness.

Many algorithms have been introduced for image and data communications compression (e.g. arithmetic coding, JPEG). These methods are generally inappropriate for probability tables because the table data must be randomly (not sequentially) accessed with minimal computational cost. In the literature of language model compression, quantization

and pruning methods are used [3, 4]. Quantization allows probability terms to be represented with only one or two bytes rather than four. With pruning methods, high order conditional probabilities are approximated with low order ones. Those probability elements that can be approximated reliably are pruned away from tables.

In this paper, we present a new compression algorithm based on mixture models. Joint probability tables are decomposed into lower-dimensional components and their mixtures. Then, model parameters are quantized into one byte integers. This algorithm satisfies the three criteria for practical application. It has a high compression ratio, as much as 1/292 under ideal conditions. It classifies quickly because only linear operations are employed using integer math. Finally, experimental results show that it does not severely degrade recognition rates, even at high compression ratios.

2. MIXTURE MODELS

For notational convenience, define $X_{a,b} \equiv (X_a, \ldots, X_b)$, the sequence of random variables X_i for $i = a, \ldots, b$. Thus $P(X_1, \ldots, X_N) = P(X_{1,N})$.

We want to compress the scanning N-tuple joint probability table $P(X_{1,N})$. We do so by using a mixture model to approximate $P(X_{1,N})$. In particular, we introduce a complete set of mixtures represented by a hidden variable μ as follows

$$\begin{aligned} P(X_{1,N}) &= \sum_{l=1}^{M} P(\mu_l, X_{1,N}) && (3)\\ &= \sum_{l=1}^{M} P(\mu_l, X_{1,k}, X_{k+1,N}) && (4)\\ &= \sum_{l=1}^{M} P(X_{k+1,N}|\mu_l, X_{1,k}) P(\mu_l, X_{1,k}) && (5)\\ &= \sum_{l=1}^{M} P(X_{k+1,N}|\mu_l) P(\mu_l, X_{1,k}). && (6) \end{aligned}$$

In Eqn (5) and (6), we assume that

$$P(X_{k+1,N}|\mu_l, X_{1,k}) = P(X_{k+1,N}|\mu_l). \quad (7)$$

Note that $P(X_{k+1,N}|\mu_l)$ is a mixture and $P(\mu_l, X_{1,k})$ is a mixture coefficient.

In general the assumption in Eqn (7) is only an approximation; however for any finite, discrete, joint probability table, it is easy to show that there exists a finite M_{exact} such that the model presented above is exact. Given the model in Eqn (6), we can now tune the amount of compression by varying M between 1 and M_{exact}. In general, the compression will be lossy.

2.1. Memory Usage Comparison

In this paper, the X_i's are discrete random variables that can have one of F different values. Thus, the joint probability table, $P(X_{1,N})$, has F^N distinctive probability elements which implies that the memory size grows exponentially with N. When there are C classes and each of them has its own joint probability table, the total memory size T_1 of the scanning N-tuple classifier is given by

$$T_1 = CF^N. \quad (8)$$

Similarly the memory usage, T_2, of the mixtures and mixture coefficients from the model in Eqn (6) is given by

$$T_2 = CM(F^k + F^{N-k}) \quad (9)$$

The memory compression ratio T_2/T_1 is determined by the number of mixtures, M, and the number of conditional variables k:

$$\frac{T_2}{T_1} = M(F^{k-N} + F^{-k}) \quad (10)$$

In our work, we set $k = \lfloor N/2 \rfloor$ since for given M and N, this value of k minimizes the number of model parameters. For instance, when $N = 5$, $F = 9$, which are the typical configurations in our recognition system, the best possible compression ratio is $1/73$ with $M = 1$ and $k = 2$.

2.2. Training Algorithm

Since we have introduced the mixtures as hidden variables, we use the EM Algorithm [5] to optimize the parameters of our mixture model under the constraints that

$$\begin{aligned} 1 &= \sum_{l}^{M} P_t(\mu_l|X_{1,k}) && (11)\\ 1 &= \sum_{X_{k+1,N}} P_t(X_{k+1,N}|\mu_l) && (12) \end{aligned}$$

where the t subscript indicates the iteration index of the EM Algorithm. The resulting parameter update rules are

$$\begin{aligned} &P_{t+1}(\mu_l|X_{1,k}) \\ &\quad = \frac{\sum_{X_{k+1,N}} P(X_{1,N}) P_t(\mu_l|X_{1,N})}{\sum_{X_{k+1,N}} P(X_{1,N})} && (13)\\ &P_{t+1}(X_{k+1,N}|\mu_l) \\ &\quad = \frac{\sum_{X_{1,k}} P(X_{1,N}) P_t(\mu_l|X_{1,N})}{\sum_{X_{1,N}} P(X_{1,N}) P_t(\mu_l|X_{1,N})} && (14) \end{aligned}$$

where

$$\begin{aligned} &P_t(\mu_l|X_{1,N}) \\ &\quad = \frac{P_t(\mu_l|X_{1,k}) P_t(X_{k+1,N}|\mu_l)}{\sum_j P_t(\mu_j|X_{1,k}) P_t(X_{k+1,N}|\mu_j)} && (15) \end{aligned}$$

Note that in the above equations, the sums over $X_{1,k}$ imply sums over all possible k-tuples, similarly for $X_{1,N}$.

3. SYSTEM OVERVIEW

The proposed memory system has two elements. One is the parameter set for mixture models comprising mixture components and mixture coefficients. The other is a discrepancy table that stores probability values that are not reliably estimated by the mixture models.

Fig. 3(a) shows the procedure for compression. Mixture models are trained by the EM algorithm from joint probability tables. Then the difference between each original probability and its corresponding mixture-model estimate is calculated. When the difference exceeds a predetermined threshold, the original probability is stored with its associated N-tuple address attribute in the discrepancy table. Finally, the mixture models and the discrepancy table are quantized into one-byte integers.

Fig. 3(b) shows the procedure for decoding probabilities. If an address appears in the discrepancy table, the associated probability is fetched. If not, its probability is estimated from mixture models.

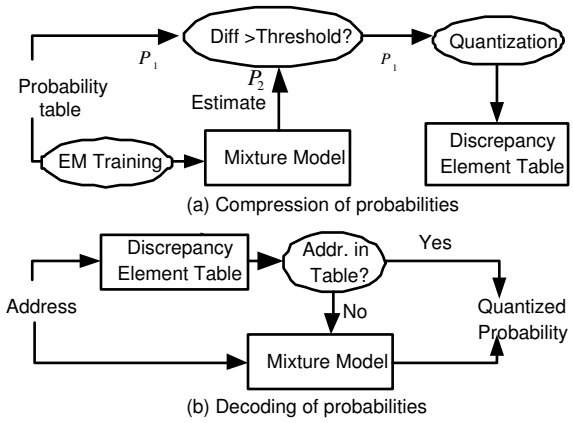

Fig. 1. Compression and decoding of probabilities.

4. EXPERIMENTAL RESULTS

4.1. Experimental Setup

In order to evaluate the proposed system, we use the online handwritten digits from the UNIPEN consortium [6]. The 15,953 characters of the Train-R01/V07 subset 1a digit set were used for training; the 8,598 digits of the DevTest-R01/V02 subset 1a were used for testing.

Both dynamic and static features are incorporated into the N-tuple recognizer [2]. The dynamic features follow the dynamic trace of a handwritten character. The static features are derived by mapping the dynamic traces to a bitmap image. Both methods generate 9-element values in 5-tuples ($N = 5, F = 9$). Either feature type can be used alone, or the features can be combined by arithmetically merging the recognition results for each feature type.

4.2. Recognition Performance

Figure 2 shows the recognition rate, relative entropy, relative memory size and relative recognition time as a function of the number of mixtures using static SNT features. As expected, when more mixtures are used, the recognition accuracy increases because the mixture models more accurately reproduce the uncompressed model. For 4, 8, and 16 mixture tables, the recognition rates are 94.0%, 94.9%, and 95.3% respectively, compared to the uncompressed 95.9%. The relative entropy, a dissimilarity measure between mixture models and the uncompressed model, decreases. Its values are 0.57, 0.31 and 0.14, respectively, for 4, 8, and 16 mixtures. Memory usage increases linearly with the number of mixtures; similarly, recognition times linearly increase with the increasing number of addition and multiplication operations for calculating probabilities. For 4, 8, and 16 mixtures, the relative memory sizes compared to the uncompressed model are 0.05, 0.11, and 0.22, respectively. The relative recognition times are 3.7, 6.7, and 11.1, respectively.

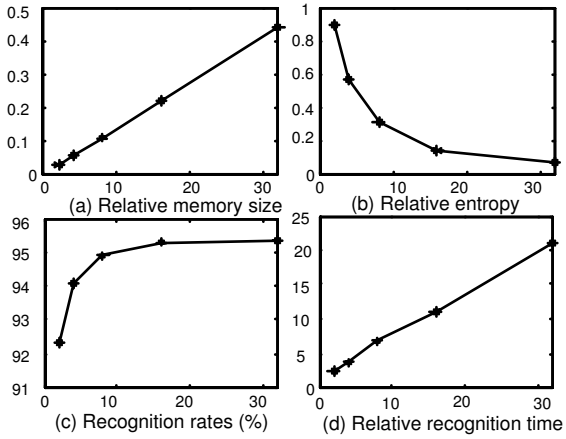

Fig. 2. Performance of mixture models on digit sets with static SNT features as a function of the number of mixtures (2, 4, 8, 16, 32). X-axis: the number of mixtures.

The quantization of probability values from floats to bytes does not significantly degrade the recognition accuracy, even though it significantly reduces memory usage by $1/4$. It increases recognition errors in the uncompressed model by only 1.2% and in the mixture models by less than 3.0 % (3.0%, 2.2%, and 1.9% for 4, 8, and 16 mixtures).

Figure 3 shows the recognition rates of quantized original probability tables and 3 different mixture models when

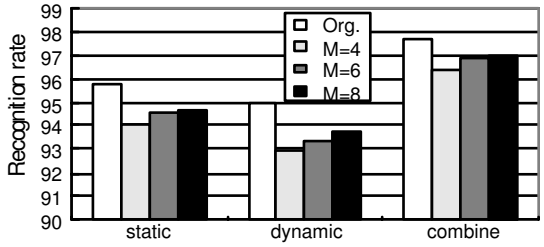

Fig. 3. Recognition rates of quantized mixture models.

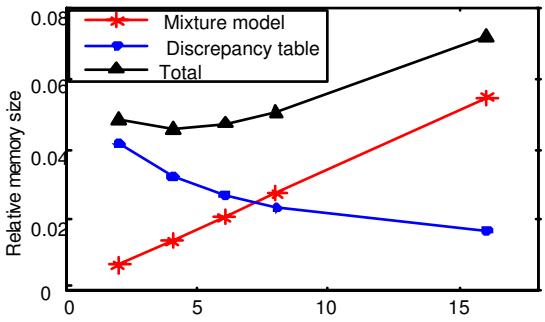

Fig. 4. Relative memory sizes of mixture models (2, 4, 6, 8 and 16) with discrepancy tables (threshold: $4 \cdot 10^{-5}$).

different feature combinations are used for recognition. Mixture models have a less negative impact on accuracy when features are combined. When an 8-mixture model is adopted, accuracy drops 1.0% with static features and 1.2% using dynamic features. However, when both features are combined the accuracy drops only 0.7% (from 97.7% to 97.0%).

Discrepancy tables complement mixture models. As the number of mixtures increase, the mixture model sizes increase linearly. However, the discrepancy table sizes decrease because original probabilities are more accurately estimated. Fig. 4 shows that the total memory size becomes minimum around 4 mixtures.

Table 1 demonstrates how recognition rates significantly improve with discrepancy tables, from less than 97.0% up to 97.5%. The optimal balance between accuracy and memory size is obtained with six mixtures. The memory size is reduced to 0.56M bytes (1/21 compression ratio) and the recognition rate becomes 97.5% (0.2% drop in accuracy).

5. CONCLUSIONS

In this paper, we propose a new framework based on mixture models and quantization for compressing large probability tables. A joint probability table is decomposed into two small probability tables: a mixture coefficient table and a mixture component table. A discrepancy table is constructed for probability elements which mixture models do not estimate reliably. The proposed method has a high compression ratio, fast decoding speed, and only a small degradation in recognition accuracy.

	without dis. tb.		with dis. tb.	
	Rec. rate	Mem.(k)	Rec. rate	Mem. (k)
Mix. = 4	96.5%	162	97.4%	540
Mix. = 6	96.9%	243	97.5%	559
Mix. = 8	97.0%	324	97.4%	599

Table 1. Recognition performance of mixture models and discrepancy tables with the threshold of $4 \cdot 10^{-5}$ and combined features (The recognition rate and the memory size of the uncompressed model: 97.7% and 11.81M bytes).

Proposed mixture models were evaluated with a scanning N-tuple recognizer on on-line handwritten UNIPEN digit sets. As more mixtures are used, the original probability values are more accurately reproduced and recognition rates are nearly restored. Relative memory sizes and recognition times increase almost linearly with increasing number of mixtures. Model parameters are quantized into one byte integers (1/4 memory size reduction) with less than 3% recognition error increase. When multiple features are combined, the degradation of recognition accuracy becomes smaller. The discrepancy tables increase recognition accuracies significantly.

The most effective compression was obtained with 6 mixtures and a difference threshold of $4 \cdot 10^{-5}$. The memory size was reduced from 11.81M bytes to 0.55M bytes (1/21 compression ratio) and the recognition accuracy dropped from 97.7% to 97.5%. The recognition speed dropped from 860 char./sec to 160 char./sec. on a 1.13GHz Pentium CPU.

6. REFERENCES

[1] S. Lucas and A. Amiri, "Recognition of chain-coded handwritten characters with the scanning n-tuple method," *Electronics Letters*, vol. 31, no. 24, pp. 2088–2089, 1995.

[2] E.H. Ratzlaff, "A scanning n-tuple classifier for online recognition of handwritten digits," in *ICDAR, Seattle*. IEEE, 2001.

[3] E. W. D. Whittaker and B. Raj, "Quantization-based language model compression," in *EUROSPEECH*, 2001.

[4] J. Goodman and J. Gao, "Lanuage model size reduction by pruning and clustering," in *ICSLP, Beijing, China*, 2000.

[5] J.A. Bilmes, "A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," Icsi-tr-97-021, U. Berkeley, 1997.

[6] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet, "Unipen project of on-line data exchange and recognizer benchmarks," in *ICPR, Jerusalem*, Oct. 1994, pp. 29–33.