

Probability Table Compression for Handwritten Character Recognition

Sung-Jung Cho*, Michael Perrone and Eugene Ratzlaff
IBM T.J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY 10598, USA
sung-jung.cho@samsung.com,
{mpp, ratzlaff}@us.ibm.com

Abstract

This paper presents a new probability table memory compression method based on mixture models and its application to N-tuple recognizers and N-gram character language models. Joint probability tables are decomposed into lower dimensional probability components and their mixtures. The maximum likelihood parameters of the mixture models are trained by the Expectation Maximization (EM) algorithm and quantized to one byte integers. Probability elements that mixture models do not estimate reliably are kept separately. Experimental results with on-line handwritten UNIPEN uppercase and lowercase characters show that the total memory size of an on-line scanning N-tuple recognizer is reduced from 12.3MB to 0.66MB bytes, while the recognition rate drops from 91.64% to 91.13% for uppercase characters and from 88.44% to 87.31% for lowercase characters. The N-gram character language model was compressed from 73.6MB to 0.58MB with minimal reduction in performance.

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 (SNT) [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.

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 [3], 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 [4, 5]. 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. Under ideal conditions, the compression ratio is 841:1 for

*Current address: Sung-Jung Cho, Ubiquitous Computing Lab, Samsung Advanced Institute of Technology (SAIT), Korea.

character prediction and 72.9:1 for character recognition. The method 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 convenience, define $X_{a,b} \equiv (X_a, \dots, X_b)$ to be the sequence of random variables X_i for $i = a, \dots, b$. Thus $P(X_1, \dots, X_N) = P(X_{1,N})$.

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

$$P(X_{1,N}) = \sum_{l=1}^M P(\mu_l, X_{1,N}) \quad (1)$$

$$= \sum_{l=1}^M P(\mu_l, X_{1,k}, X_{k+1,N}) \quad (2)$$

$$= \sum_{l=1}^M P(X_{k+1,N} | \mu_l, X_{1,k}) P(\mu_l, X_{1,k}). \quad (3)$$

We now assume that

$$P(X_{k+1,N} | \mu_l, X_{1,k}) = P(X_{k+1,N} | \mu_l). \quad (4)$$

Thus

$$P(X_{1,N}) = \sum_{l=1}^M P(X_{k+1,N} | \mu_l) P(\mu_l, X_{1,k}). \quad (5)$$

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 (4) 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 (5), 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 class has its own joint probability table, the number of elements, T_1 , in the probabilities tables is given by

$$T_1 = CF^N. \quad (6)$$

Similarly the number of elements, T_2 , of the mixtures and mixture coefficients from the model in Eqn (5) is given by

$$T_2 = CM(F^k + F^{N-k}) \quad (7)$$

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

$$\frac{T_2}{T_1} = M(F^{k-N} + F^{-k}) \quad (8)$$

In our work, we set $k = \lfloor N/2 \rfloor$ since for given M and N , this value of k gives optimal compression. For instance, when $N = 5$, $F = 9$, which are the typical configurations in our SNT recognition system, the maximum compression ratio is 72.9 : 1 with $M = 1$ and $k = 2$; and when $N = 4$, $F = 29$, which are the typical configurations in our N-gram character predictor, the maximum compression ratio is 841:1.

2.2 Training Algorithm

Since we have introduced the mixtures as hidden variables, we use the EM Algorithm [6] to optimize the parameters of our mixture model under the constraints that

$$1 = \sum_l^M P_t(\mu_l | X_{1,k}) \quad (9)$$

$$1 = \sum_{X_{k+1,N}} P_t(X_{k+1,N} | \mu_l) \quad (10)$$

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

$$P_{t+1}(\mu_l | X_{1,k}) = \frac{\sum_{X_{k+1,N}} P(X_{1,N}) P_t(\mu_l | X_{1,N})}{\sum_{X_{k+1,N}} P(X_{1,N})} \quad (11)$$

$$P_{t+1}(X_{k+1,N} | \mu_l) = \frac{\sum_{X_{1,k}} P(X_{1,N}) P_t(\mu_l | X_{1,N})}{\sum_{X_{1,N}} P(X_{1,N}) P_t(\mu_l | X_{1,N})} \quad (12)$$

where

$$P_t(\mu_l | X_{1,N}) = \frac{P_t(\mu_l | X_{1,k}) P_t(X_{k+1,N} | \mu_l)}{\sum_j P_t(\mu_j | X_{1,k}) P_t(X_{k+1,N} | \mu_j)} \quad (13)$$

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.

Figure 1(a) shows the procedure for compression. Mixture models are trained by the EM algorithm from joint probability tables and its elements are quantized into one-byte integers. Then the difference between each original probability and its corresponding mixture-model estimate is calculated. When the difference exceeds a predetermined limit, the original probability is stored with its corresponding probability table address in the discrepancy table. Finally, the discrepancy table elements are quantized into one-byte integers.

Figure 1(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.

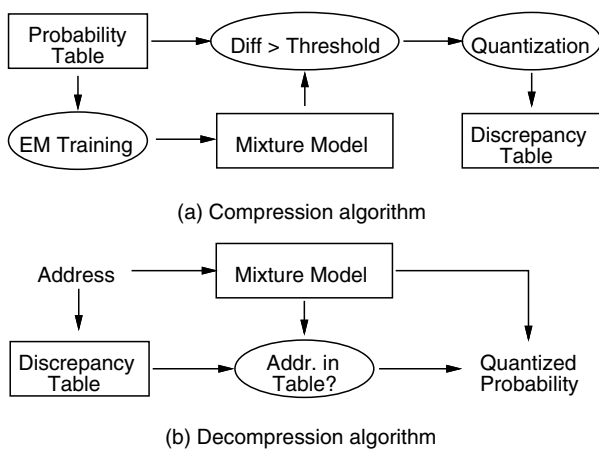


Figure 1. Compression and decoding of probabilities

4 Experiments

To test our compression algorithm, we used it to compress the probability tables generated by a charac-

ter N-gram predictor model and the scanning N-tuple recognizer described in the sections below.

4.1 The Character N-Gram Predictor

The proposed method was validated by compressing a simple 5-gram character language model and comparing the compressed and uncompressed models. A 25-million word corpus of English text was first filtered by eliminating all symbols except apostrophe, converting all digits to a single class, and converting all uppercase characters to lowercase characters. This filtering reduced the remaining characters to a 29-class set of 26 lowercase characters classes, one digit class, and the apostrophe and space character classes. The corpus was then parsed into “stream” and “word” N-grams. The stream processing treated the entire corpus as a single string from which all 5-grams were extracted. The word processing broke the corpus into words, prepended four spaces to each word and then extracted all 5-grams from each resulting string. The stream processing retains some neighboring word context while word processing does not. The stream and word sets each had 139,318,386 5-grams, of which 451,137 and 217,214 were unique, respectively. Each data set of 5-gram strings was randomly split, 99:1, into training and test sets, respectively. Full and compressed probability tables were created from the training set to predict the likelihood of any character following a given 4-character history.

To test the character language model, the 4-gram history of each test 5-gram was first used to create a complete set of 29 possible 5-grams queries. The probability of each 5-gram query was extracted from the model and the 5-grams were ordered by probability into an N -best list. Results for predicting the correct 5-gram given the 4-gram history were recorded for the $N = 1$ to 10 top N -best lists.

Figure 2 compares the predictive performances of the uncompressed (Baseline) and compressed 5-gram language models. The compressed models represent a 125:1 compression of the original probability tables at a small cost in accuracy. For purposes of comparison, we also constructed a (lossless) look-up table of all observed 5-tuples probabilities (*i.e.*, we compressed the original table by removing all zero probability entries.) Using 4 bytes per 5-tuple and 4 bytes for each floating-point probability resulted in a compression of 12:1.

4.2 The Scanning N-Tuple Recognizer

The scanning N-tuple (SNT) technique is applicable to variable length discrete feature sequences (e.g. chain

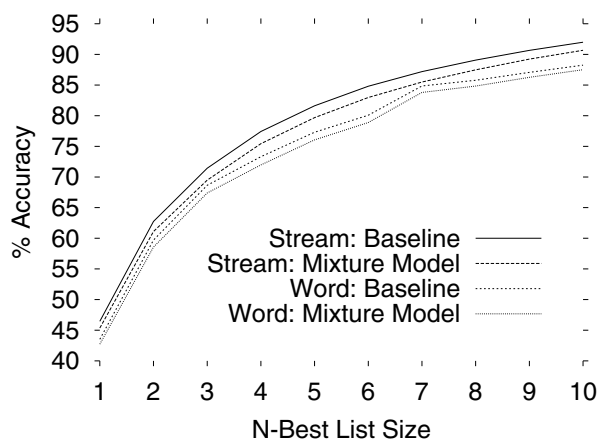


Figure 2. N-gram prediction performance of 6 mixture $P(X_2X_3|X_1X_4)$ compression models using a 0.00004 discrepancy table threshold on stream and word data compared to their respective baselines.

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 sequence. A handwritten character is recognized by choosing the class that yields the highest joint probability.

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

$$X_{1,N}^i = (f_i, f_{i+k}, f_{i+2k}, \dots, f_{i+(N-1)k}) \quad (14)$$

where $i = 1, \dots, L - (N-1)k$, 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

$$P(\cup_i X_{1,N}^i | C) = \prod_i P(X_{1,N}^i | C). \quad (15)$$

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 $P(X_{1,N}^i)$.

4.3 SNT Experimental setup

In order to evaluate the proposed system, we use the on-line handwritten digits from the UNIPEN consor-

tium [7]. Training and testing was done with the Train-R01/V07 and DevTest-R01/V02 sets, respectively, using subsets 1b-uppercase and 1c-lowercase.

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.

4.4 SNT Compression Performance

Fig. 3 shows the size of the uppercase and lowercase mixture models and discrepancy tables as a percentage of the size of the original uncompressed models for various numbers of mixtures. Note that there is a trade-off between the size of the discrepancy table and the size of the mixture model: As the mixture model becomes more complex, there is less discrepancy between it and the uncompressed table. In this case, the total memory size is a minimum at around 6 mixtures. Memory usage increases linearly with the number of mixtures; similarly, recognition time increases linearly with the increasing number of addition and multiplication operations for calculating probabilities. The relative recognition times for 2, 4, 6, and 8 mixtures are 2.9, 3.8, 4.6 and 5.5, respectively, relative to the uncompressed model. The mixture model algorithm trades speed for size.

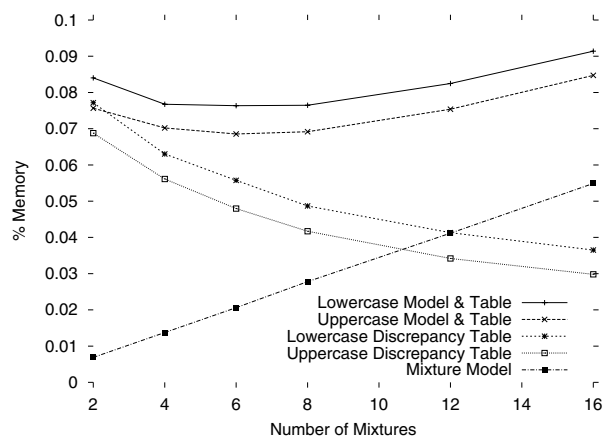


Figure 3. Relative memory sizes of mixture models (2, 4, 6, 8 and 16) and discrepancy tables (threshold: $6 \cdot 10^{-5}$) for combined static and dynamic SNT features on uppercase and lowercase characters.

4.5 SNT Recognition Performance

Figure 4 shows the uppercase and lowercase character recognition rate using static features for the uncompressed model (Baseline) and the mixture model with and without a discrepancy table. As expected, when more mixtures are used, the recognition accuracy increases because the mixture models more accurately reproduce the original (non-mixture) model. Surprisingly, much of the accuracy is recovered by using a discrepancy table with a mixture model consisting of only two mixtures. The quantization of probability values

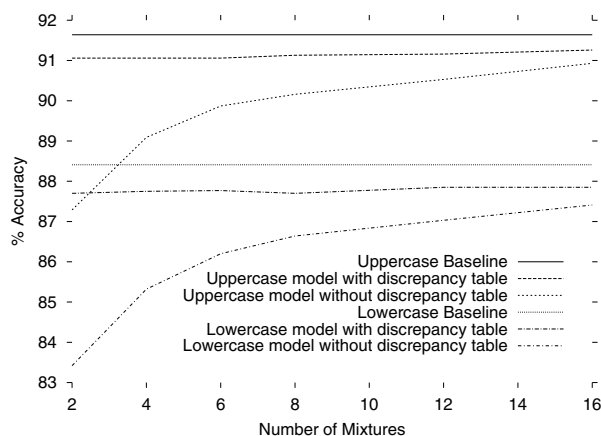


Figure 4. Uppercase and lowercase character recognition accuracy with and without discrepancy tables (threshold: $6 \cdot 10^{-5}$) vs. the number of mixture (2, 4, 6, 8 and 16) for combined static and dynamic SNT features.

from floats to bytes does not significantly degrade the recognition accuracy, even though it reduces memory usage by 1/4.

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 that 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.

Proposed mixture models were evaluated with a scanning N-tuple recognizer on on-line handwritten

UNIPEN uppercase and lowercase character 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 the difference threshold of $6 \cdot 10^{-5}$. The memory size was reduced from 12.3MB to 0.66MB bytes, while the recognition rate drops from 91.64% to 91.13% for uppercase characters and from 88.44% to 87.31% for lowercase characters. The recognition speed dropped from 860 char./sec to 160 char./sec. on a 1.13GHz Pentium CPU. The N-gram character language model was compressed from 73.6MB to 0.58MB with minimal reduction in performance. It is interesting to note that the uncompressed static and dynamic feature uppercase models individually have accuracies of 88.5% and 83.5% while the combined mixture model is both smaller and more accurate (91.1%), similarly for lowercase (86.1% and 83.3% vs. 88%). Thus, on memory-constrained devices, the mixture model can both increase accuracy and reduce memory.

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] Eugene H. Ratzlaff, "A scanning n-tuple classifier for online recognition of handwritten digits," in *ICDAR, Seattle*. IEEE, Sep. 2001.
- [3] Thomas M. Cover and Joy A. Thomas, *Elements of Information Theory*, John Wiley & Sons, 1991.
- [4] E. W. D. Whittaker and B. Raj, "Quantization-based language model compression," in *EUROSPEECH*, 2001.
- [5] Joshua Goodman and Jianfeng Gao, "Language model size reduction by pruning and clustering," in *ICSLP, Beijing, China*, 2000.
- [6] Jeff 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, Univ. of Berkeley, 1997.
- [7] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet, "Unipen project of on-line data exchange and recognizer benchmarks," in *12th ICPR, Jerusalem, Israel*, Oct. 1994, pp. 29-33.