Probability Table Compression Using Distributional Clustering for Scanning N-Tuple Classifiers

Jianying Hu   Eugene Ratzlaff
IBM T.J. Watson Research Center
1101 Kitchawan Road, Route 134 Yorktown Heights, NY 10598
{jyhu, ratzlaff}@us.ibm.com

Abstract

A method for compressing tables of probability distributions using distributional clustering is presented and applied to shrink the look-up tables of a scanning n-tuple handwritten character recognizer. Lossy compression is realized by clustering n-tuples that are observed to induce similar class probability distributions. A new distance metric called "weighted mean KL divergence" is introduced to assess similarity and account for the cumulative effect of merging two distributions. After compression, cluster membership is rebalanced in an annealing-like process. The proposed method is evaluated on three isolated-character subsets of the UNIPEN database. Compression ratios in excess of 2000:1 are demonstrated for 5-tuple classifiers.

1. Introduction

The scanning n-tuple (sn-tuple) classifier first introduced by Lucas and Amiri [7] provides accurate, high speed recognition for offline or online character data [7, 8] at the cost of large memory requirements. The sn-tuple is a maximum-likelihood classifier applied to chain code feature sequences, where the probability of observing the complete code is given by the ensemble probability for observing all of the scanned n-tuples derived from the chain code.

For each sample of a character class c_i, the sn-tuple algorithm generates a variable length chain code that is subsampled into tuples of length n with features $f_1, f_2, \cdots, f_n$, where each code f ranges from 0 to $\sigma - 1$. In training, we assume a uniform distribution of the class prior probabilities $p(c_i)$ for the set of Q character classes $C = \{c_1, c_2, \cdots, c_Q\}$ and estimate the probability distribution $P(C|T_i)$ of the observed n-tuples at each $i = \{1, 2, \cdots, \sigma^n - 1\}$. In decoding, given a sequence of observed n-tuples $\tau = (t_1, t_2, \cdots, t_M)$, where $t_k \in \{T_1, T_2, \cdots, T_{\sigma^n - 1}\}$, $k = 1, 2, \cdots, M$, the sn-tuple classifier assumes that the n-tuples are mutually independent. Using the Bayes rule and assuming a uniform distribution of class prior probabilities, it can be shown that the posterior probability of the input belonging to class c_i, $p(c_i|\tau)$, is determined by the product of the conditional probabilities of class c_i given each individual n-tuple. Thus the classifier selects the character class with highest posterior probability as given by:

$$c = argmax_i \prod_{k=1}^{M} p(c_i|t_k). \tag{1}$$

where each $p(c_i|t_k)$ is drawn from the $\sigma^n \times Q$ probability look-up table generated in training.

Unfortunately, these look-up tables can become very large with commonly used values of $n \geq 5$ and $\sigma = 8$, making it impractical for embedded applications. Hoque *et al.* have addressed this issue using chain code decompositions [5]. Comparing a compressed 8-tuple with an uncompressed 5-tuple, they demonstrate a 12X memory savings with improved ensemble accuracy [5, 4]. Cho *et al.* have applied quantization and EM mixture models with discrepancy tables for 21:1 compression and nominal accuracy loss [2]. In this paper we present a new method using distributional clustering that performs with nominal accuracy loss at 20:1 compression, but which can scale to compressions of more than 5000:1 with only moderate increases in the error rate.

The rest of the paper is organized as follows. In Section 2 we first briefly review the concept of distributional clustering. We then introduce a new distance measure called *weighted mean KL divergence* to quantify the effect of merging two different distributions, followed by a two-stage clustering procedure to group n-tuples into a given number of events. Experimental results on standard UNIPEN data [3] are presented in Section 3 and we conclude in Section 4.

2. Distributional Clustering of N-Tuples

Consider the random variable over character classes, C, and its distribution given a particular n-tuple T_i, denoted $P(C|T_i)$. The idea behind distributional clustering of n-tuples is that if two distinct n-tuples, T_i and T_j induce similar class distributions, they can be clustered together and represented by a single distribution which is the weighted average of the individual distributions:

$$P(C|T_i \vee T_j) = \frac{P(T_i)P(C|T_i) + P(T_j)P(C|T_j)}{P(T_i) + P(T_j)}. \quad (2)$$

To be more general, from now on we will use the notion of class distribution given a particular event, E_i, denoted $P(C|E_i)$. Tuples belonging to the same cluster are treated as identical events and induce the same class distribution. Since we now only need to store one distribution per event as opposed to one per distinct n-tuple, this paradigm leads to a compression ratio of $\sigma^n : M$, where M is the number of events. The small overhead of a look up table mapping any n-tuple to an event is in most cases negligible compared to the size of the probability table.

2.1. The Effect of Merging Two Distributions

Given two distributions $P(C|E_i)$ and $(C|E_j)$, the information theoretic measure for the difference between them is the Kullback-Leibler (KL) divergence measure defined as:

$$D(P(C|E_i)\|P(C|E_j)) = -\sum_{k=1}^{Q} p(c_k|E_i) \log\left(\frac{p(c_k|E_i)}{p(c_k|E_j)}\right).$$

Unfortunately this measure has two undesirable properties: it is not symmetric, and it is infinite when a class has non-zero probability in the first distribution and zero probability in the second.

Baker and McCallum [1] introduced a related measure called "KL divergence to the mean", defined as:

$$\frac{P(E_i)}{P(E_i \vee E_j)} \cdot D(P(C|E_i)||P(C|E_i \vee E_j)) + \frac{P(E_j)}{P(E_i \vee E_j)} \cdot D(P(C|E_j)||P(C|E_i \vee E_j)). \quad (3)$$

In information theoretical terms, this measure can be understood as the expected amount of inefficiency incurred if, instead of compressing two distributions optimally with their own code, we use the code that would be optimal for their mean. This measure not only avoids the two undesirable properties of the classic KL measure, but is also more suitable for clustering as it measures directly the effect of merging two distributions into one.

For the purpose of n-tuple clustering in the context of character recognition, we wish to further modify this measure to take into account the cumulative effect of merging two distributions on the final classification. As shown in Equation 1, each n-tuple encountered in the input character is treated as an independent event and the class likelihood of all the events are accumulated to produce the final score. Thus, the true cost of merging two distributions should be further weighted by the prior probability of the joint event – the less frequently two events are likely to occur, the smaller the impact of merging their distributions. We call this new measure the "weighted mean KL divergence", defined as:

$$D_f(E_i, E_j) = P(E_i) \cdot D(P(C|E_i)||P(C|E_i \vee E_j)) + P(E_j) \cdot D(P(C|E_j)||P(C|E_i \vee E_j)). \quad (4)$$

This is the distance measure we will use to cluster the n-tuple distributions.

2.2. Clustering Algorithm

Given the distance metric defined above and the desired number of clusters M, the optimal solution to the n-tuple clustering problem is the one which minimizes the total within-cluster distance out of all possible permutations. Unfortunately this is a combinatorial problem with exponential complexity. Various techniques have been proposed before to achieve a locally optimal solution in polynomial time, including the well known K-means and hierarchical clustering methods [6]. We have developed a method based on the greedy agglomerative approach proposed by Baker and McCallum [1] (referred to as the Baker method from now on). Our method contains an initialization stage and a refinement stage, requiring a total computation time of $O((M+1)^2(\sigma^n - M) + \alpha M \sigma^n)$, where α is a small integer (e.g., 3).

The first stage, initialization, is very similar to the Baker method, the main difference being the distance measure used (Equation 4 vs. Equation 3):

- Sort the n-tuple distributions into a list ordered by decreasing mutual information with the class variable, defined as: $I(C|T_i) = H(C) - H(C|T_i)$, where $H()$ represents entropy and $H(C)$ is constant for all tuples.
- Initialize $M + 1$ clusters as singletons with the top $M + 1$ tuple distributions on the list
- Iterate until the last tuple distribution on the list has been processed
  - Merge the two clusters with minimum distance (Equation 4).
  - Create a new cluster consisting of the next tuple distribution in the sorted list.

In the second stage, refinement, the cluster membership of each n-tuple is adjusted using the following annealing-like process, repeated α times:

- Compute the within-cluster distance between each n-tuple distribution and its cluster center using Equation 4.

- Sort the n-tuple distributions in the order of decreasing within cluster distance.
- For each n-tuple T_k in the sorted list, suppose its cluster assignment is E_i:
  - Compute its distance, $D_f(T_k, E_k)$ to all cluster centers.
  - Find the cluster E_j with the minimum distance.
  - If $j \neq i$ then move T_k to cluster E_j and update distributions $P(C|E_i)$ and $P(C|E_j)$.

3. Experimental Results

The proposed method was evaluated on the UNIPEN isolated handwritten characters in categories 1a (digits), 1b (upper case) and 1c (lower case), using the complete Train-R01/V07 subsets for training and the full DevTest-R01/V02 subsets for testing [3]. 5-tuples from static (offline) bitmap chain code features were used with $\sigma = 9$ and offset $\delta = 7$ [8]. For the digits, uppercase and lowercase categories we used 14, 133, and 170 allographs, respectively. Uncompressed probability distributions were first trained, then subsequently compressed using distributional clustering, then further compressed 2-fold by quantizing [2] 4-byte floats to scaled 2-byte log-probabilities.

Figure 1 shows the error rate for digits classification as a function of the combined compression from both distributional clustering and quantization. All compression ratio reported here reflect only the size reduction in the probability look-up tables and do not account for the added overhead of mapping sn-tuple addresses to event addresses.

Results using three different clustering methods are plotted in Fig. 1: the Baker method, the weighted mean KL divergence without refinement (referred to as ”Weighted Mean”), and the weighted mean KL divergence with refinement (referred to as ”Refined Weighted Mean”). Results of Refined Weighted Mean were all generated with 3 iterations of the annealing process defined in Section 2.2 (i.e. $\alpha = 3$) and reflect the full method proposed herein. As shown in the plot, the proposed method using weighted mean KL divergence measure performs significantly better than the Baker method. With a total compression of 20:1 the drop in accuracy is statistically insignificant. The error rate gradually increases almost linearly with the log of the compression ratio until reaching about 5000:1. At this point the number of n-tuple distributions begins to approach the number of character classes and the error rate rises precipitously as the number of classes exceeds M as seen with the last two points.

The available 5000:1 compression provides two orders of magnitude more compression than that reported by other methods [2, 5], though without the potential for improving accuracy as with Hoque *et al.*. At a compression of 5905:1 the 7.0% error rate using distributional clustering is slightly lower than the 7.6% error rate of a 2-mixture model with a compression of only 146:1 (without discrepancy table) [2].

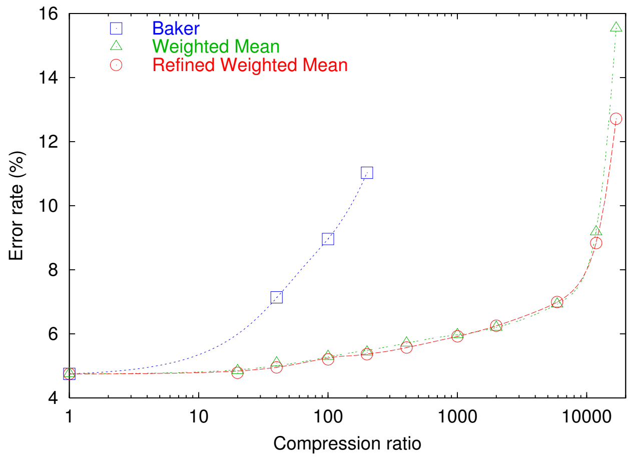

Figure 1. Error rate vs. compression ratio for digit recognition.

Results for compression of the uppercase and lowercase probability tables are shown in Fig. 2. The fact that the behavior of the algorithm in both cases is very similar to that shown in the digit recognition case demonstrates that the algorithm scales remarkably well to problems with much larger numbers of classes. This is further demonstrated by the plots in Fig. 3, where the relative error rate (additional error normalized by uncompressed error) is plotted against compression ratio for all three tasks.

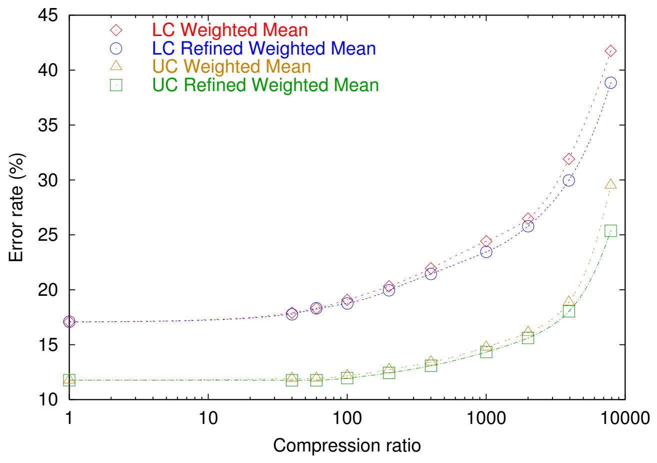

Figure 2. Error rate vs. compression ratio for uppercase and lowercase character recognition.

Another interesting observation that can be made from Figures 1 and 2 is that the refinement procedure results

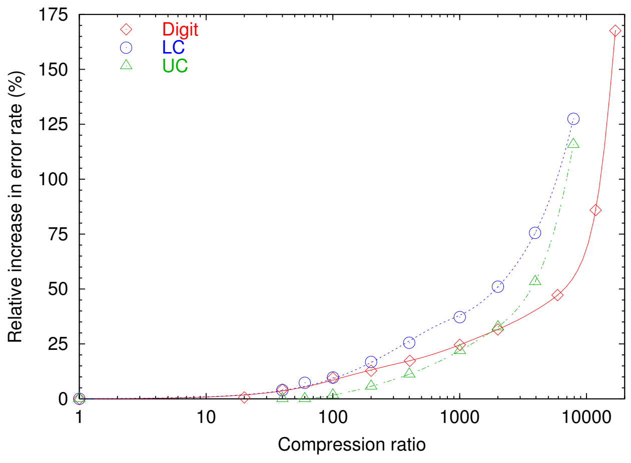

Figure 3. Relative error increase vs. compression ratio for digit, uppercase and lowercase character recognition.

in only marginal performance improvement, especially at lower compression ratios. This is somewhat surprising, since our calculations show that in virtually every case refinement reduces the total within-cluster weighted mean KL divergence by about 10%. One reason why this consistent reduction in total distribution distortion is not reflected in the final performance could be that there are other factors, such as mutual information, which are not considered in the refinement process.

It has been shown that ensemble classification of online handwriting with different sn-tuple features results in increased accuracy [8], and that this advantage holds true despite compression using mixture models [2]. We also created an ensemble classifier by pairing the static sn-tuple method with a dynamic feature scanning 5-tuple method and observed similar results using compression by distributional clustering. For example, at a compression of 200:1 the static+dynamic ensemble error rate for digit recognition was only 3.3% (up from 3.0% without compression), whereas the separate error rates for the static and dynamic feature sn-tuple classifiers were 5.4% (up from 4.7%) and 6.2% (up from 5.6%), respectively.

4. Conclusions and Future Work

Distributional clustering with weighted mean KL divergence provides several advantages for compressing probability tables such as those used by scanning n-tuple classifiers. This compression method incurs a negligible speed penalty and works for small as well as large numbers of classes. A desired compression ratio can be selected from a wide range, up to as much as 2000:1 or more for 5-tuples. Developers can target a particular memory footprint by evaluating the ratio of the sizes of the original and the desired compressed tables to determine the required compression ratio. Furthermore, the synergies of combining multiple scanning n-tuple classifiers having different feature types to achieve higher accuracy are maintained with compression by distributional clustering; in fact, combining classifiers somewhat ameliorates compression lossiness.

The proposed method also has a few limitations. The time to compute grows significantly with increasing n and M. In addition, the method's lossiness can be quite different for different tasks. and appears to rise dramatically with increasing compression as M approaches the number of fundamentally unique classes. This suggests that compression might be significantly more lossy when smaller values of n are used with a large number of classes.

In principle this compression method can work for any probability table, especially those that have some distributions that are similar or are infrequently observed or sparsely populated. For the scanning n-tuple, we believe this method may well work in conjunction with the chain code decompositions of Hoque *et al.* [5], or the mixture models or discrepancy tables of Cho *et al.* [2], or various combinations of these three. In addition, further compression using log-probability quantization to one byte instead of two should be examined. Future work in this area should focus on combining and extending combinations of these different compression methods.

References

[1] D. Baker and A. McCallum. Distributional clustering of words for text classification. In *SIGIR'98*, pages 96–103, Melbourne, Australia, August 1998.

[2] S. Cho, M. Perrone, and E. Ratzlaff. EM mixture model probability table compression. In *Proc. ICASSP'03*, Hong Kong, 2003.

[3] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. Unipen project of on-line data exchange and recognizer benchmarks. In *Proc. ICPR'94*, Jerusalem, Israel, August 1994.

[4] S. Hoque. Personal communication. 2004.

[5] S. Hoque, K. Sirlantzis, and M. Fairhurst. A new chain-code quantization approach enabling high performance handwriting recognition based on multi-classifier schemes. In *Proc. ICDAR'03*, Edinburgh, England, August 2003.

[6] A. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[7] S. Lucas and A. Amiri. Recognition of chain-coded handwritten characters with the scanning n-tuple method. *Electronic Letters*, 31(24), 1995.

[8] E. Ratzlaff. A scanning n-tuple classifier for online recognition of handwritten digits. In *Proc. 6th ICDAR*, Seattle, US, September 2001.