K-Means Clustering for Hidden Markov Models

Michael P. Perrone and Scott D. Connell[a]
Pen Technologies Group, IBM T.J. Watson Research Center
mpp@us.ibm.com

Abstract

An unsupervised k-means clustering algorithm for hidden Markov models is described and applied to the task of generating subclass models for individual handwritten character classes. The algorithm is compared to a related clustering method and shown to give a relative change in the error rate of as much as 8% on a 30,000-word vocabulary, unconstrained-style, on-line, writer-independent handwriting recognition task.

1 Introduction

One aspect of unconstrained-style handwriting which makes recognition more difficult is the existence of a large degree of variation among instances of any given handwritten word or character. These variations can be categorized as either noise inherent to the writing process, or differences among writing styles. Models (*e.g.*, character models) to be used in classification must represent both of these variations as accurately as possible, to achieve good recognition accuracy. Due to the existence of disparate styles of writing any given character, individual character classes may be more adequately represented by a set of character subclass models, rather than a single model per character class. The problem of automatically determining these character subclasses, which we will refer to as *allographs*, remains difficult.

Previous methods for allograph identification fall into two categories: those which cluster and perform classification in different spaces[1,2,3] and those which cluster and perform classification in the same space[4,5]. Those algorithms which use the same space for clustering and classification rely on very simple classifiers such as k-nearest neighbors and template matching and therefore may not be appropriate for large, real-world problems. Those algorithms which cluster and classify in different spaces are free to use more sophisticated classification techniques; however they cannot be certain that the allographs found in their clustering space will correspond to well-formed clusters in their recognition space.

[a]This author is now at the Dept. of Comp. Sci. & Eng., Michigan State University: connell@cse.msu.edu

In: L.R.B. Schomaker and L.G. Vuurpijl (Eds.), Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition, September 11-13 2000, Amsterdam, ISBN 90-76942-01-3, Nijmegen: International Unipen Foundation, pp 229-238

One example of such a method was used by Bellegarda *et al.* and Subrahmonia *et al.*[1,2] to initialize HMM training:[b] Each exemplar from each character class is size-normalized and resampled to a fixed number of equispaced points along the strokes. Local features are extracted for each point and concatenated to make a high-dimensional feature vector for each exemplar. The top few principal components are used to generate a subspace for clustering. A fixed, but large, number of initial clusters is selected and k-means clustering is performed to identify allographs. After each clustering, the mean corresponding to the cluster with the smallest population was removed and the whole set was reclustered using the remaining means. This mean removal process was iterated until the iteration's drop in the probability of the training data exceeded a preset threshold. Visual inspection of the resultant allograph classes shows them to be reasonable though not perfect. Note that this algorithm was used to generate allograph labels for data which was subsequently used to train HMMs using a different feature space. Therefore, this algorithm has the problem that it does not perform clustering in the same space as classification.

In this paper we avoid this problem by proposing an allograph identification method that uses the same space for clustering and classification while at the same time using a more sophisticated classification method than, for example, k-nearest neighbors. The core idea of this approach is to use a standard k-means clustering algorithm in which the distance metric used is the probability that a given allograph model generates a given instance of a character. In our case, the same HMMs are used in clustering and classification.

2 K-Means HMM Allograph Clustering Algorithm

The clustering algorithm can be summarized:

1. Generate initial allograph labels for training data.
2. Train allograph HMMs using labeled data [1,2].
3. Relabel training data using newly-trained HMMs.
4. Compare the new labels to the previous labels.
5. Terminate if:
   (a) No labels have changed (*i.e.*, converged to a fixed point), or
   (b) The new labels match the labelings of any previous iteration (*i.e.*, converged to a limit cycle).
6. Go to Step 2.

[b] This method was used to label one of the data sets used in this paper (*HandSeg*, see Sec. 4.)

In Step 1, initial labels may be chosen randomly or generated by some other clustering algorithm. The relabeling in Step 3 is performed for each character, C, by choosing the label of the most probable allograph model, thus at time t, the label, $l_t(x)$, for some exemplar, $x \in C$, is given by

$$l_t(x) = \arg\max_j P(x|M^C_{j,t-1}), \quad \forall\, j = 1, \ldots, J^C \tag{1}$$

where $M^C_{j,t}$ is the j-th allograph model at time t from character C, $P(x|M^C_{j,t})$ is the probability of exemplar x given allograph model $M^C_{j,t}$, and J^C is the number of allograph models for character C.

3 Clustering Experiments

To test the algorithm, we performed the following experiments.

3.1 Single-Character Allograph Clustering

For a single character class and a fixed number of allographs, we randomly assigned allograph labels to each exemplar. This labeling was used as the initial training set for our clustering algorithm. The results in Fig. 1 show the typical behavior of the average over exemplars of the log of the probability density of the most likely allographs

$$\frac{1}{N}\sum_{i=1}^{N} \max_j P(x_i|M^C_{j,t}) \tag{2}$$

for t at convergence. Note that the single allograph case has no variance since it consists of a single possible arrangement of labels (*i.e.*, all the same). It is included to show the log-prob improvement due to multiple-allograph character models relative to a single character model.

3.2 Differentiating Two Character Classes

Another basic question is how well this algorithm separates categories that humans distinguish. To test this, we considered four different character pairs: "1"/"0", ")"/"(", "R"/"B" and "O"/"U". For each paired set, each exemplar was randomly assigned an allograph, labeled as if it had come from a single character class with two allographs. Allograph clustering was performed to test the clustering algorithm's ability to "discover" character class distinctions. For the 1/0 and)/(pairs, clustering always resulted in allographs that exactly corresponded to the original character classes. For the more confusable R/B and O/U pairs, clustering resulted in allographs that had on average 98.9% and 97.4% correspondence, respectively, to the original character classes. The

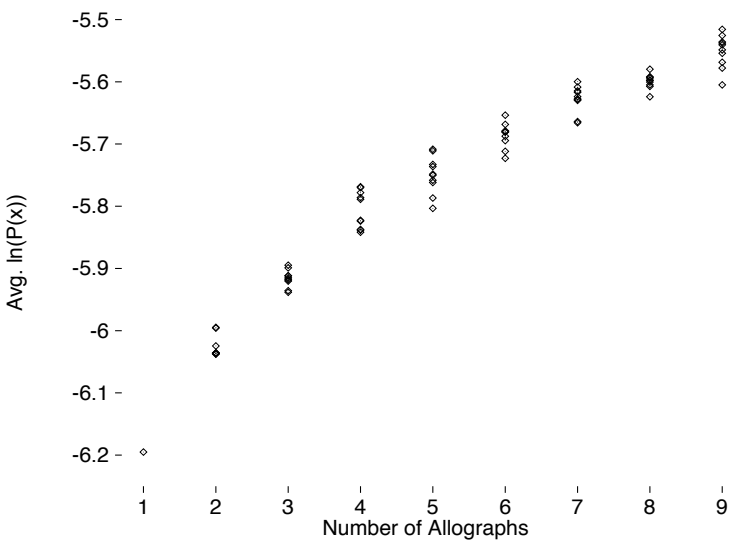

Figure 1: Average log probability density of a set of 655 examples of a single-stroke "B" character written by 68 writers. Results are shown for 9 random initializations for each number of allographs. Point spread corresponds to convergence to various suboptimal local maxima.

characters that were mapped to a different character class correspond to cases where the character exemplars were visually more ambiguous. This suggests that the models are sensitive to subtle differences that human labeling disregards.

3.3 Choosing the Number of Allographs

One effect of increasing the number of allograph models is the corresponding slowdown in recognition speed as more models must be evaluated. Therefore in order to trade off between speed and accuracy, one may look for a sharp "shoulder" in the models' log-probability curve. To check whether the proposed algorithm would generate a sharp shoulder when one is clearly expected to exist, we combined three character classes, "1", "0" and "R", chosen since they are visually very distinct and therefore are probably well-separated in probability space. Randomly-initialized allograph clustering resulted in Fig. 2, which shows a clear shoulder at three allographs. It is interesting to note that the two-allograph solutions found three local minima corresponding to the three different possible groupings of these three characters into two groups: (1)(0R), (0)(1R), and (R)(01). However, distinct shoulders were not generally

observed (see for example Fig. 1) making it difficult to select an optimal number of allographs and suggesting that the optimal number is potentially much higher than the numbers that we investigated. For this reason, we turn to cross-validation to help us determine the optimal number of allograph models.

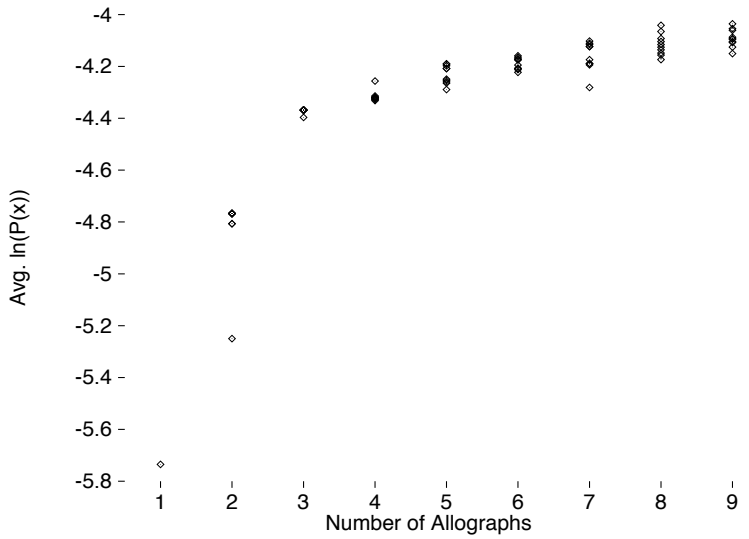

Figure 2: Optimal cluster number selection from a set of 9 random initializations for each number of clusters. Shoulder occurs at 3 clusters. Additional clusters increase the log probability further due to additional variability within each of the three constituent character classes.

3.4 Cross-Validation

When recognition speed is not an issue, one can use cross-validation to choose the optimal number of allographs per character class. Fig. 3 shows training and testing results for the "–" (hyphen) character class with 2-, 5- and 10-fold cross-validation. In each case, the data were randomly split into the appropriate number of folds and trained from random initialization with each subset held out in turn. Each line on the graph corresponds to an average over 30 training runs versus the number of allograph clusters. A surprising observation from Fig. 3 is that even for a character as simple as a hyphen, the optimal number of allographs is at least eight and possibly higher.[c] One question which needs further investigation is whether the improved modeling

[c]For technical reasons unrelated to the algorithm, we did not run cross-validatory allograph clustering experiments with more than nine allographs per character.

within a character will lead to improved recognition accuracy when compared to other models.

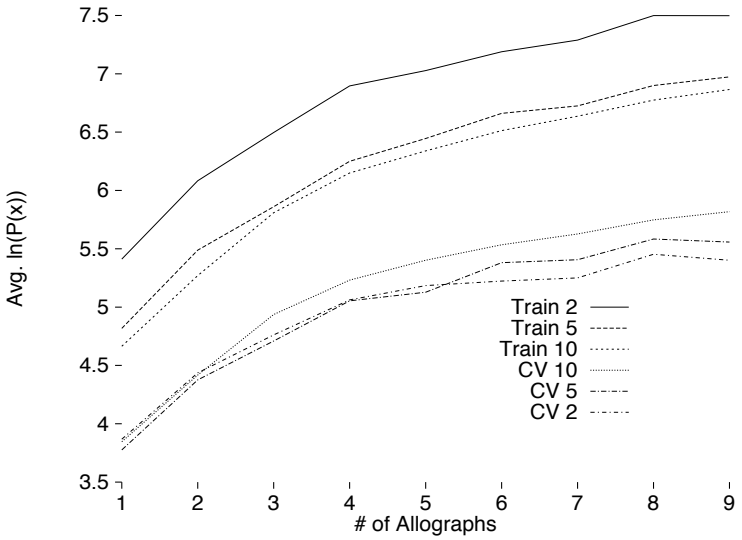

Figure 3: Cross-validation graphs for a set of 639 examples from 68 writers of the "–" character for fixed numbers of allographs ranging from 1 to 9. The sequence of graphs is in the same order from top to bottom as in the graph key.

3.5 Algorithm Convergence Properties

For the data sets examined, we found that convergence of the clustering algorithm was fairly quick. Fig. 4 shows the dependence of the mean number of training iterations on the number of allographs and on the data. The figure shows that convergence speed slows slightly as the number of allographs increases; however convergence appears to depend more strongly on the complexity of the character data being clustered. Note that single-allograph models always have a single iteration and that 17.3% of these convergence results are runs which converged to limit cycles. The average length of these limit cycles was 2.3 iterations. The number of exemplars per allograph (see Fig. 5) varied greatly with the average percentage difference between the largest and the smallest being 18.7% of the total population and was fairly independent of the number of allographs and the character class. Out of 324 random initializations of clustering. no clusters were observed to have zero membership. Thus the number of allographs generated can never increase and usually do not decrease.

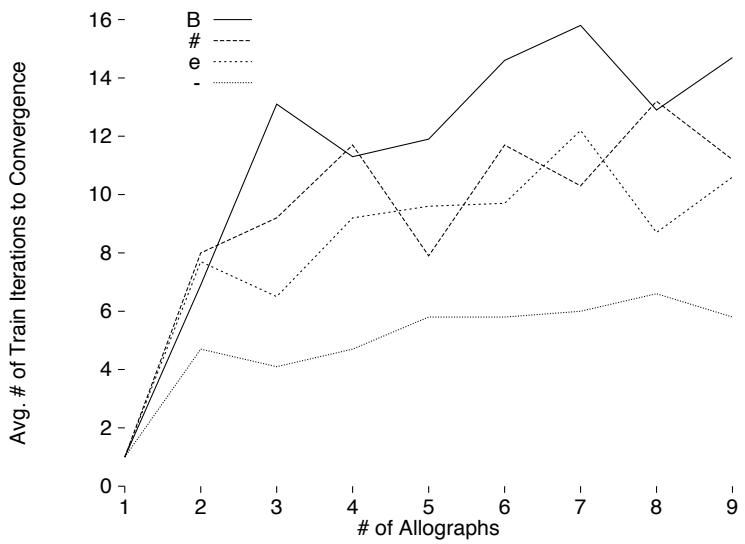

Figure 4: Average number of clustering iterations to convergence as a function of the number of allograph clusters for four different character classes.

4 Large Vocabulary Experiments

The goal of these preliminary, large-vocabulary experiments was to determine whether iterative k-means allograph clustering can find allograph models which improve recognition accuracy on real-world, handwriting data. The large-vocabulary experiments used the following training data sets, where "isolated" means not written as part of a word:

- *UncWords* - 52,212 unconstrained-style words from 100+ writers
- *HandSeg* - 122,423 isolated exemplars of 93 character classes and exemplars from hand-segmented unconstrained-style words from 100+ writers
- *Natural* - 55,883 isolated exemplars of 93 character classes from 125 writers in their "natural" styles
- *Predefined* - 84,960 isolated exemplars of 93 character classes from 136 writers, written to match a predefined set of 326 visually distinct allographs
- *Cursive* - 42,068 isolated cursive exemplars of 93 character classes from 98 writers

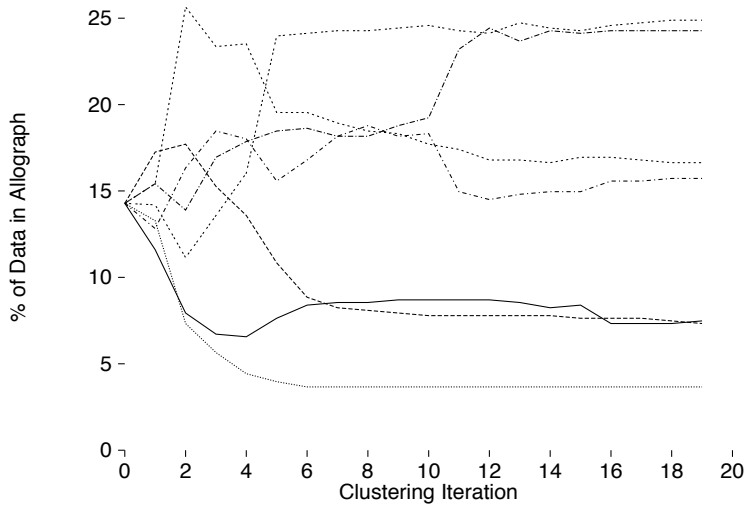

Figure 5: Population of 7 clusters vs. number of clustering iterations for a set of 629 exemlars from 125 writers of the "B" character.

- *Discrete* - 101,293 isolated printed exemplars of 93 character classes from 224 writers

The test set consisted of 4920 unconstrained-style words written as sentences from 188 writers. Each experiment included the *UncWords* data set for HMM training but not for allograph clustering. All results are reported on the Test set.[d] The zeroth iteration corresponds to the first set of models trained.

The results in Fig. 6 marked as *Baseline* correspond to using the *HandSeg* data set with allograph labels from the allograph clustering algorithm used by Bellegarda *et al.* and Subrahmonia *et al.*[1,2] as the initialization to the k-means allograph clustering algorithm. The result labeled *Natural* corresponds to combining the *Natural* and *HandSeg* data sets, using the HMMs from the zeroth iteration of the *Baseline* experiment to label the combined data set, and then using those labels as the initialization to the k-means HMM allograph clustering. An analogous process produces the results labeled *Predefined*, *Cursive* and *Discrete*.

The results mentioned so far depend on another allograph clustering algorithm for their initial allograph labels. One question to ask is how dependent

[d]For expedience, none of the large vocabulary runs were allowed to converge and no checking for cycling of the allograph labels was performed.

the proposed k-means HMM clustering algorithm is on these initial labelings. To answer this question, we ran an experiment which was independent of any other clustering algorithm: The result labeled *Random* corresponds to using the *HandSeg* data set with uniformly distributed random allograph labels, where the number of allographs per character class is the same as for the *Baseline* results, and then using those random labels as the initialization to the k-means HMM allograph clustering.

The results in Fig. 6 show that for all but one data set, allograph relabeling improves the recognition accuracy on an independent test set. It is interesting to note that random labeling in the *Random* experiment performs much better than the labeling used in the *Baseline* experiment, which suggests that if an initial labeling is near a suboptimal solution it may be difficult to escape from it, as appears to be the case for *Baseline*. This further suggests that the algorithm might benefit from the introduction of noise in the labels, as in simulated annealing. It is not clear why the *Natural* data set performed poorly, but it seems reasonable to assume that it may be because that data set does not well represent the allographs found in the test set. This point deserves further investigation. Also, there appears in some cases to be a subsequent drop in accuracy if too much training occurs, suggesting possible over-fitting.

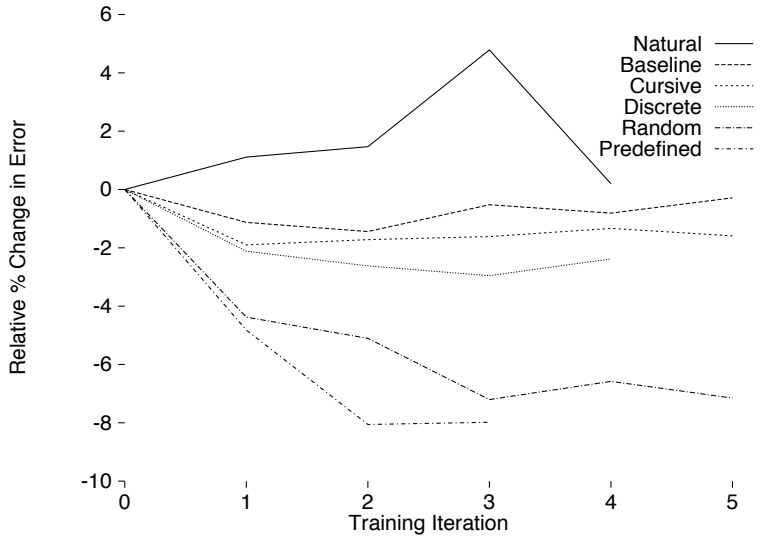

Figure 6: Relative % change in character error rate as a function of training iteration for several training sets.

5 Discussion

This paper describes a method for combining k-means clustering and HMMs to address the problem of handwritten character allograph determination. The method has been shown to work well in simple cases, and preliminary results on a large-vocabulary, writer-independent, unconstrained-style, handwriting recognition task show that this method can improve recognition accuracy. In particular, we were pleasantly surprised to see that in the large-vocabulary recognition tests, random initialization may perform better than starting from a previously chosen "good" initial-condition allograph labeling. This will be investigated further in conjunction with the use of cross-validation to select the optimal number of allograph models. Also, we plan to investigate replacing our current use of $\max P(x|M)$ approximation with the unapproximated form $\sum P(x|M)P(M|C)$.

6 Acknowledgements

The authors gratefully acknowledge the help of Millie Miladinov, John Pitrelli, Gene Ratzlaff and Jayashree Subrahmonia.

7 References

1. J. Bellegarda, D. Nahamoo and K. Nathan, "Supervised Hidden Markov Modeling of On-Line Handwriting recognition," *ICASSP'94*, vol. 5, pp. 149-152, 1994.
2. J. Subrahmonia, K.S. Nathan and M.P. Perrone, "Writer Dependent Recognition of On-Line Unconstrained Handwriting," *ICASSP'96*, vol. 6, pp. 3478-3481, 1996.
3. S. Connell and A.K. Jain, "Learning Prototypes for On-Line Handwritten Digits," *Proc. 14th International Conference on Pattern Recognition*, Brisbane, Australia, pp. 182-184, Aug. 1998.
4. L. Prevost and M. Milgram, "Non-supervised Determination of Allograph Sub-classes for On-line Omni-scriptor Handwriting Recognition," *Proc. 5th International Conference on Document Analysis and Recognition*, Bangalore, India, pp. 438-441, Sept. 1999.
5. P. Scattolin and A. Krzyzak, "Weighted Elastic Matching Method for Recognition of Handwritten Numerals," *Vision Interface'94*, pp. 178-185.