Improving Off-line Handwritten Character Recognition with Hidden Markov Models

Elie Krevat
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, PA 15213
ekrevat@cs.cmu.edu

Elliot Cuzzillo
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, PA 15213
ecuzzill@andrew.cmu.edu

Abstract

A method for the off-line recognition of handwritten characters with hidden Markov models (HMMs) is described. Performance of our HMM is compared to a baseline Naive Bayes classifier. Experimental results are given for different variants of our HMM algorithm, with optimizations for ignoring the effects of inter-word transitions when applying the Viterbi algorithm to return the most likely character sequence. Finally, our HMM algorithms are compared to a dictionary creation and lookup algorithm that is less generally applicable but for our particular dataset achieves almost perfect classification results.

1 Introduction

The field of Optical Character Recognition (OCR) has been extensively researched in the past 60 years, and due to its many different applicable environments, it continues to be a rich area for active research. In this paper, we focus specifically on offline handwritten character recognition. The document is available in its entirety, represented as a pixel or bit-map representation of the handwritten characters. Unlike the online domain, this domain lacks information on the time sequence of individual line-segments made for each character. Furthermore, for the purposes of this paper, we will assume that the segmentation problem of separating each character from its neighbors has been solved for us, as well as any noise filtering and normalizations of scale.

Since we can make use of the entire document at once, it is possible to exploit correlations between adjacent characters and words. One way to do this is through contextual knowledge of syntax and a dictionary of possible words, which has been shown to be successful for reading handwritten address information of postmarked mail [10]. However, an alternative and less data-intensive approach for when the data is not easily mapped to a dictionary representation is to make use of a hidden Markov model (HMM) to learn common sequences of character orderings and other inter-letter correlations.

We tackle the problem of character recognition with two main goals. Our primary goal is to quantify the marginal benefit of using a HMM to learn correlations between adjacent characters. Our secondary goal is to learn an accurate classifier for our specific data set, and for this secondary goal we will not restrict ourselves to Markov models. By building a series of "smarter" classifiers, adding optimizations and variants to previous models, we can compare just how well each of our algorithms perform.

2 Related Work

There is a large corpus of research on the application of character recognition in many different domains, but a much smaller subset of literature on the off-line handwritten character recognition using HMMs. HMMs combined with statistical grammar rules are very attractive for online cursive handwriting recognition because training is easy and avoids segmentation issues [7]. HMMs have been commonly used for off-line cursive handwriting [1, 3], and combined with dictionaries of limited size and cooperative writers it has achieved up to 98% accuracy [2].

Previous research has differed in their approach by either learning an HMM over whole dictionaries of words as states [8], or learning a much smaller set of states over the individual letters [6, 9]. The former approach with words as states is much more applicable for domains of small dictionary sizes or where there are other restrictive syntactic rules [2]. The latter approach with letters as states, which we have used, has the desirable property that only one model is needed for the entire language and that model is much less complex. Some modern papers on printed handwriting incorporate the use of Markov Random Fields and multi-dimensional dynamic programming to determine when to merge pixels and regions for local feature extraction [4].

Much of the related HMM work focuses on a number of specific features extracted from the image sample, which is usually more helpful for cursive handwriting recognition [2]. These type of global features are described in depth in [11] for extracting moments and invariants of the entire image. Some examples of these features include measures of curvature between endpoints of an edge and the percentage of pixels lying in certain regions of the image. Other work uses local feature extraction to group windows around the pixels of an image [5].

3 Problem Definition

The high-level task in off-line handwriting recognition is to classify an ordered sequence of images of handwritten characters. The dataset we used, found at `http://ai.stanford.edu/ btaskar/ocr/`, adheres closely to this model. Each character is represented by a scaled, denoised, segmented 16 by 8 array of 1-bit pixels. This means that the problem of character segmentation, in which the image of a whole page of text is split into images of individual characters, has been previously solved for this dataset. To simplify the problems posed by capitalized letters, the dataset does not include the first letters of the original words. An example of an actual observed word, displayed by concatenating each segmented letter to form the original word, is shown in Figure 1.

The data was collected from 159 different subjects with wide variance in handwriting style, and it appears that some of the characters have been segmented from cursive handwriting. Therefore, our data set might be considered a mixed-modal environment, which is a much harder domain compared to a small group of collaborative writers that mimic a specific style of writing as in [2].

Each character entry includes, along with the sampled bitmap image, a label for the correct character classification, word information (to indicate boundaries between words), and ordering information. The ordering information and the word ID has been crucial to our approach to the problem. Finally, the data is preprocessed into ten evenly distributed cross-validation folds.

4 Proposed Method

Hidden Markov models effectively capture correlations in the sequence of characters in each word, and since they may be combined with other generative learning algorithms we expect an HMM classifier will provide a significant improvement in accuracy. HMMs are actively used in unsupervised learning environments along with expectation maximization algorithms such as Baum-Welch, but labeled observations provided by our dataset allow us to explicitly train our model. Specifically, the prior probabilities and the transition matrix for each letter is assigned to be the frequency of that letter in the training data, To compensate for transitions that were not represented in our training data, we applied a small prior probability to each letter and transition by 'halluci-

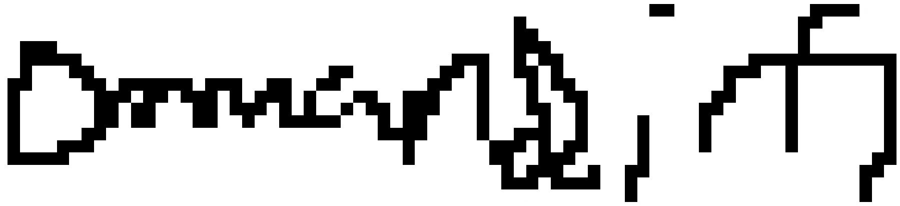

Figure 1: Example observation from data set: actual word is “commanding” with the first letter removed

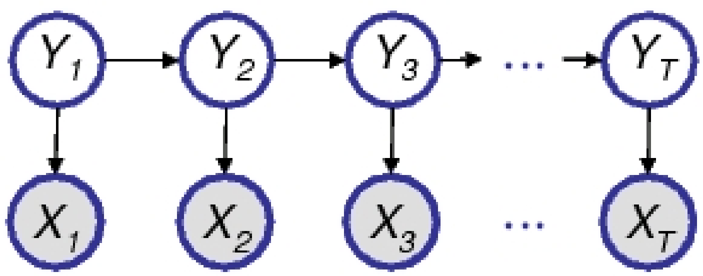

Figure 2: In our Markov model, the hidden variables *Yt* are the 26 letters of the English alphabet and the observed variables are the bitmap images

nating’ one example of each before training. Once our model has been trained, the problem of predicting the true character sequence of a series of bitmap observations is solved by application of the Viterbi algorithm, which predicts the most probable sequence of states that produces a group of observations.

In addition to the transition matrix, there are many possible approaches for the representation and the learning algorithm of the emission probabilities for our HMM classifier. Our HMM’s latent variables represent the letters corresponding to each image, and our observed variables are the image bitmaps. Because our image representation generally has a smaller number of pixels than other datasets of higher resolution, we did not believe that extracting a separate set of global features for training in our HMM was necessary, and the Naive Bayes approach over each pixel would be justified. Therefore, we made a Naive Bayes assumption that our pixels are conditionally independent given the letter. Clearly, the Naive Bayes assumption is not valid in this setting, but the resulting classifier, even without an HMM, still has a 62.7% accuracy on the test data. We use the latter classifier as a baseline to compare against the performance of our final HMM.

One difficulty with Naive Bayes is that it isn’t robust to translations of the image– an O shifted to the left and an O shifted to the right appear completely unrelated under its assumptions. To combat this, we experimented with translating the image in four different directions during classification to find the maximally probable orientation. As our results show, this variant of the Naive Bayes algorithm actually hurts performance, and the experimental results are included as a second algorithm forcomparison.

One important optimization which we made to the Viterbi algorithm for application in our domain was to eliminate the influence of character transitions for the last characters of each word. The intuition for this change is that we want to focus on learning the more correlated intra-word relationships and not be misinformed by the largely noisy and uncorrelated transitions between the last character of one word and the first character of the next word.

As a benchmark for our final optimized HMM approach, we also explored a dictionary-based method that did not use any hidden Markov assumptions. For this final algorithm, we expected that because of the very small dictionary size of the data set (55 distinct words), we would achieve our most accurate results, but if we were given a data set with a larger number of words this approach might quickly become intractable. During training, a hash table is created containing all words found in the training set. The Naive Bayes representation is still used for the distribution of possible images given the letter class. This dictionary approach necessitates classification of entire words. To classify each word, we compute the probability of the observation given the word (assuming that all the character images are conditionally independent given the word), and choose the word that maximizes this probability.

4.1 Additional Algorithms Considered but not Implemented

We also considered several other methods both for emission and transition probability representation. One possibility was Joint Bayes parameter estimation, but the number of parameters which would have to be estimated makes this approach intractable. The number of prior probabilities of each character that needs to be learned is the same for a naive estimator as for a joint, and is just equal to the number of possible characters (26). The main problem with a Joint Bayes estimator is that because our observed character bit arrays are 128 bits, we would have to learn on the order of 2^{128} joint probabilities for each of the 26 possible states. When compared to the 3,328 parameters (128x26) that need to be estimated for the Naive Bayes approach, the Naive Bayes is a much more attractive classifier.

Another possibility for emission probability representation was a Markov random field, which can represent significant conditional independence assumptions. The problem here is that Markov random fields are very inefficient for representing joint distributions– evaluating joint distributions in an MRF involves computing an exponentially-difficult normalization term.

Finally, we began to implement an nth-order hidden Markov model, that is one which does inference on a graphical model with an nth-order Markov assumption, but had to stop for performance and time reasons. For instance, when using a second-order Markov assumption, the transition matrix changes from a matrix of size 676 to one of size 17576. Worse, while running the Viterbi algorithm on a first-order HMM maintains only 26 most-probable-paths at any given time, which are each about 5000 states long, the Viterbi algorithm on a second-order model needs 676 most probable paths, which makes the algorithm prohibitively expensive.

5 Experiments

All of our experiments were run with 10-fold cross-validation, and the results are given as the averages across all runs.

Our experiments to date first involved a test of the Naive Bayes assumption, which we use as a baseline classifier and then combine into the emission probabilities of our HMM. It was somewhat surprising that this approach achieved a 62.7% accuracy rate with a standard deviation of 1.1%, which is about a 15 times better accuracy than a completely random classifier would do over the 26 possible characters.

When we implemented our shifting strategy during classification, choosing the maximum probable letter over the set of five one-pixel shifted translations, we were even more surprised to find that this algorithm actually performed worse than the baseline Naive Bayes classifier. It achieved a 55.9% success rate, almost 7% less than our baseline, with a 1.0% standard deviation. To explain these results, we hypothesize that by shifting to add more common white background pixels and possibly subtracting the less common black pixels, it meant that our shifted images with less black pixels were more likely to be chosen, and these shifted images contain less data to make a better informed decision. We speculate that if, in training, we made a larger white buffer around each image, and shifted each training image to the position most likely given the previous training images, then we would have a “sharper” set of probabilities. Another variant which might help, which is also left as a future work item, is by shifting the observations during the learning phase of the Naive Bayes classifier, or perhaps preprocessing the data by overlaying all the shifted images on top of each other, creating a thicker boundary around each letter. This latter approach is similar

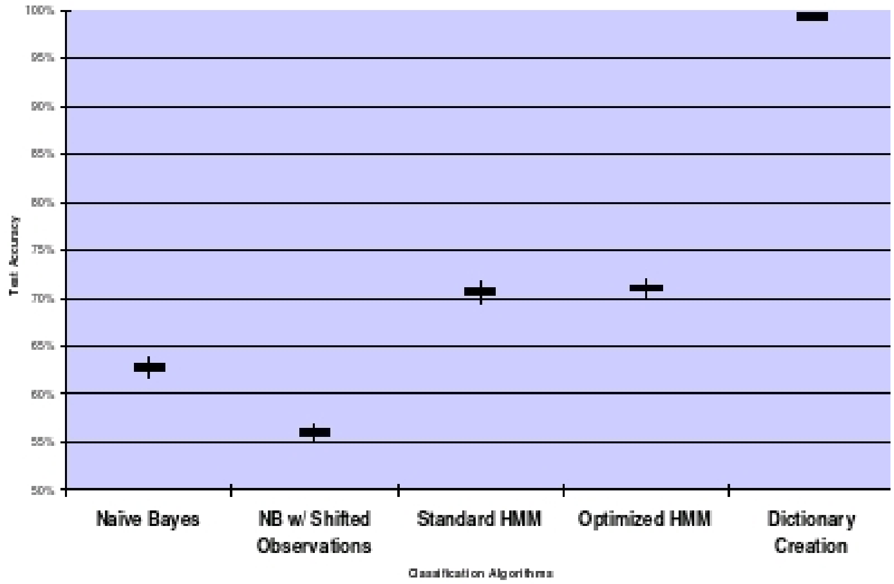

Figure 3: Experimental Results

to the localized feature extraction by expanding pixels, except that a smarter localized feature extraction would only expand pixels that are more important for achieving better classification results.

To measure the improvement provided by the hidden Markov model, we learned the model parameters as described above, and then implemented the Viterbi algorithm. Our results show an average accuracy of 70.6% and standard deviation of 1.3%, a significant improvement over our baseline algorithm. We were disappointed to see only a marginally better improvement by applying the optimizations to the Viterbi algorithm to handle the noisy data of character transitions across words, achieving a final accuracy of 71.0% and standard deviation of 1.1%. We hypothesize that due to the very small dictionary size of the words in our data set, that the noise contributed by inter-word transitions is not significant enough for our optimizations to make a big impact. When applied to another dataset with a larger and more varied dictionary size, we could possibly see a better marginal performance of our optimized algorithm.

Just as the very small dictionary size of our dataset might have limited the marginal benefit of our optimized HMM algorithm, it also resulted in an extremely accurate performance of the dictionary creation and lookup algorithm. With close to perfect accuracy, this final approach achieves an average of 99.3% accuracy on the test data with a standard deviation of 0.3%.

6 Conclusions

These results, first and foremost, demonstrate that a Naive Bayes classifier can perform remarkably well on images of handwritten characters even though the Naive Bayes assumption is not wholly accurate. The Naive Bayes classifier accounted for a significant portion of our solution's accuracy, without applying any complex feature extraction or pre-processing on the data set.

These results also show a significant benefit derived from exploiting correlations between adjacent letters in words when a dictionary is infeasible to create or unavailable for the testing set. Our optimized hidden Markov model was shown to improve classification accuracy from 62.7% to 71.0% compared to the baseline Naive Bayes algorithm. However, an HMM does not capture the correlations between letters nearly as well as a dictionary does, when the dictionary is small enough to make this approach feasible, and the dictionary is available. As one would expect, when the domain is constrained enough to the point that grammatical or syntactical rules can be predicted and used by a learning algorithm, that algorithm does particularly well for that domain. However, for a simple scalable model that is applicable to many domains of variable dictionary size, the hidden Markov model can achieve impressive results on a mixed-modal data set of handwritten printed and cursive data.

References

[1] N. Arica and F. Yarman-Vural. An overview of character recognition focused on off-line handwriting. *IEEE transactions on systems, man and cybernetics - part C: Applications and reviews*, 2001.

[2] H. Bunke, M. Roth, and E.G.Schukat-Talamazzini. Off-line cursive handwriting recognition using hidden markov models. *Technical Report IAM-94-008, University of Bern*, 1995.

[3] R.G. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. In *IEEE transactions on pattern analysis and machine intelligence*, 1996.

[4] S. Chevalier, E. Geoffrois, and F Preteux. A 2d dynamic programming approach for markov random field-based handwritten character recognition. In *IAPR International Conference on Image and Signal Processing*, 2003.

[5] T.H. Hildebrandt and W. Liu. Optical recognition of handwritten chinese character: advances since 1980. *Pattern Recognition*, 1993.

[6] A. Kundu, Y. He, and P. Bahl. Recognition of handwritten word: first and second order hidden markov model based approach. In *Computer Society Conference on Computer Vision and Pattern Recognition*, 1988.

[7] J. Makhoul, T. Starner, R. Schwartz, and G. Chou. On-line cursive handwriting recognition using hidden markov models and statistical grammars. In *Human Language Technology Conference, Proceedings of the workshop on Human Language Technology*, 1994.

[8] R. Nag, K.H. Wong, and F. Fallside. Script recognition using hidden markov models. In *Proceedings of ICASSP*, 1988.

[9] S.N. Srihari, I.J. Hull, and R. Chowdhury. Integrating diverse knowledge sources in text recognition. In *ACM Trans. on Office Automation Systems*, 1983.

[10] R.W.S. Tregidgo and A.C. Downton. High performance character recognition for off-line handwritten british postcodes. In *IEE Colloquium on Binary Image Processing - Techniques and Applications*, 1991.

[11] O.D. Trier, A.K. Jain, and T. Taxt. Feature extraction methods for character recognition - a survey. *Pattern Recognition*, 1996.