Using HMMs to boost accuracy in optical character recognition

Prasanna Velagapudi
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
pkv@cs.cmu.edu

Abstract

One of the current trends in the field of optical character recognition (OCR) is the combination of multiple classifiers to produce more robust recognition. In this paper, a kNN classifier, a multi-layer perceptron, and a support vector machine with a radial basis function kernel are tested on a pre-segmented word-based OCR task with and without Viterbi error-correction. I discuss the effects of error-correction on the classification accuracy of each method.

1 Introduction

The field of optical character recognition (OCR) has progressed significantly since machine learning methods were first applied to it almost two decades ago, with the introduction of numerous novel techniques that have reduced computational cost and improved performance. A recent summary of the state of the art subdivides the modern methods of solving segmentation-based OCR—in which data is segmented into characters independently of classification—into three categories: statistical methods, artificial neural networks, and kernel methods (primarily support vector machines) [7].

1.1 Statistical Methods in OCR

Statistical classifiers are derived from Bayes' rule, and can be categorized into parametric and non-parametric methods. Non-parametric methods, like k-Nearest-Neighbor (kNN), store and compare all training samples, making them unusable in the general OCR case. However, there are several popular parametric classifiers for OCR, including quadratic discriminant functions (QDF), linear discriminant functions (LDF), and regularized discriminant analysis (RDA) [7].

1.2 ANNs in OCR

Optical character recognition was historically one of the first "real world" applications of artificial neural networks (ANNs). LeCun et al. first proposed a solution using convolutional neural networks in 1989 [5]. Almost two decades later, the LeNet-5 variant is widely used, and convolutional neural networks are still considered one of the most successful techniques for OCR. However, recent comparisons of techniques for OCR have demonstrated that competitive results should also be obtainable using the much simpler multi-layer perceptron network architecture [7].

1.3 SVMs in OCR

In recent years, support vector machines (SVMs) have also become popular in the field of OCR. Although they have higher computational complexity and memory requirements than ANNs, they have also been shown to have higher classification accuracy over several handwriting recognition databases, specifically in conjunction with a radial-basis function kernel [6].

1.4 Viterbi error-correction

Regardless of the underlying classifier, in the typical cases of single character classification, errors can be introduced by the physical ambiguities of certain characters [2]. Certain handwritten sets of characters such as (“0”, “O”, “o”), (“l”, “i”, “1”), and (“Z”,“z”,“2”) share many optical features, confusing most classifiers. It is therefore beneficial to apply word-level constraints on the occurrence of characters in a sequence to ambiguous characters. Hidden Markov models are one well-studied method of discovering and applying this type of sequential pattern recognition. Numerous bodies of work deal with the integration of HMMs into both ANNs [2] and SVMs [1] for complex tasks such as segmentation. However, there is considerably less work dealing with the much simpler task of using HMMs to simply boost classification accuracy on pre-segmented OCR data.

If we model words as Markov chains of characters, then there exists a well-known method for finding the most likely sequence of characters given some observed sequence of characters and a knowledge of the transitional relationships between characters. This method is the Viterbi algorithm. This algorithm will replace predicted letters with the maximum likelihood estimate for their position given the prediction. If trained properly, correct sequences of letters will remain unchanged, while ambiguous cases can be resolved using the surrounding character sequence as context for the ambiguous letter.

By testing this straightforward and modular error-correction scheme for expressing word-level constraints, we can answer the following questions:

- Can Viterbi error-correction improve OCR accuracy significantly?
- Might certain classifiers be improved more by Viterbi error-correction than others?
- Is it possible to match the performance of more computationally intensive classifiers (like SVMs) using simpler algorithms (such as ANNs) assisted by Viterbi error-correction?

2 Method

2.1 Classification steps

The basic methodology of the error-corrected classification is as follows:

1. The base classifier is trained on a training data set.

2. An HMM is modeled on the transition probabilities of the training data set and the emission probabilities of the base classifier, as seen in Figure 1.

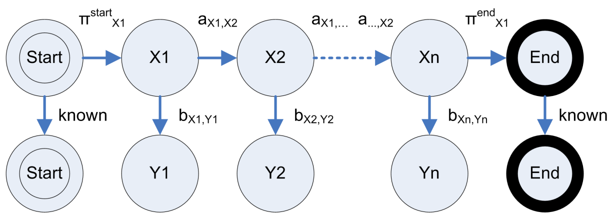

Figure 1: HMM graphical model of word.

3. The base classifier makes predictions on the test data set.

4. The Viterbi algorithm is run on the predictions made by the base classifier, using the HMM built in the previous step to create the maximum likelihood letter sequence for each word.

2.1.1 Training

Due to the small size of the training set, during 10-fold cross-validation certain words would not appear at all in the training set. With so few words to begin with, this skewed the transitional probabilities of the HMM quite considerably. Thus, the HMM's transition probabilities were estimated using the entire data set. It should be noted that the information encoded in the transition probabilities only reflected the occurrence of words and their relative probabilities, both of which would be very reasonable a-priori knowledge in limited-vocabulary OCR cases. Thus, the estimator for transition probabilities is as follows:

$$\begin{aligned} A_{ij} &= \text{\# of transitions from letter i to letter j} && (1) \\ \lambda_A &= \text{regularization constant} && (2) \\ a_{ij} &= \frac{A_{ij} + \lambda_A}{\sum_{j'} A_{ij'} + \lambda_A} && (3) \end{aligned}$$

In addition, multinomial distributions were calculated for the probabilities of starting or ending a sequence on a given letter. Because all sequences were of finite length, an end state with transition probabilities from each letter could be used to improve accuracy. The estimators for these were as follows:

$$\begin{aligned} \Pi_i^{start} &= \text{\# of words that start with letter i} && (4) \\ \lambda^{start} &= \text{regularization constant} && (5) \\ \pi_i^{start} &= \frac{\Pi_i^{start} + \lambda^{start}}{\sum_{i'} \Pi_{i'}^{start} + \lambda^{start}} && (6) \end{aligned}$$

$$\begin{aligned} \Pi_i^{end} &= \text{\# of words that end with letter i} && (7) \\ \lambda^{end} &= \text{regularization constant} && (8) \\ \pi_i^{end} &= \frac{\Pi_i^{end} + \lambda^{end}}{\sum_{i'} \Pi_{i'}^{end} + \lambda^{end}} && (9) \end{aligned}$$

The data was then divided into 10 folds, each containing sets of whole words. Using standard cross-validation techniques, each classifier was trained using the data outside the fold. The predictions made by the classifier on the training data were conditioned by letter, and used to calculate the emission probabilities of the HMM. This yields the following expression for the estimator of emission probabilities:

$$
\begin{aligned}
B_{ik} &= \text{\# of times letter i is classified as letter k} && (10)\\
\lambda_B &= \text{regularization constant} && (11)\\
b_{ij} &= \frac{B_{ik} + \lambda_B}{\sum_{k'} B_{ik'} + \lambda_B} && (12)
\end{aligned}
$$

2.1.2 Classification

Each classifier was then tested on the data included in the fold, and the errors were recorded. The predictions of the classifier were also recorded for processing through the Viterbi algorithm.

2.1.3 Error-correction

The predictions of the classifier were separated into words, and the Viterbi algorithm was run on each word (modeled as a sequence of letters). The output of the algorithm was a new sequence of letters described as the "error-corrected" prediction.

2.2 Classifiers

Several classifiers were selected to test the hypothesis that Viterbi error-correction could be used to improve the accuracy of segmentation-based OCR. A kNN classifier was selected as an offline method of statistical classification. A multi-layer perceptron (MLP) was selected to represent ANN classification. Lastly, a radial basis function kernel SVM was selected as the best representative kernel-based method.

2.2.1 kNN

k-nearest-neighbor classification is a common baseline in many OCR studies. As such, it is included here as a method of baselining the errors generated on this dataset. A generic kNN classifier was implemented in Matlab using a Euclidean distance function. For the purposes of simplicity and comparability, testing was done using 1-NN.

2.2.2 MLP

Using the Matlab Neural Network toolbox, a 2-layer perceptron with 50 hidden nodes and 26 output nodes was constructed. The index of the maximum valued output node was taken to be the output of the network during testing. This model was found through empirical testing to perform the best on the dataset of all attempted MLP designs, although it did not approach the accuracies reported in other literature [7].

2.2.3 SVM+RBF

Using a freely available SVM toolkit for Matlab [3], it was possible to relatively trivially implement a multi-class Vapnik support vector machine with a standard radial basis function kernel. The method used for multi-classing was one-against-one, as it has been shown to perform similarly to 1-of-K (one-against-the-rest) methods, but requires less computation [4].

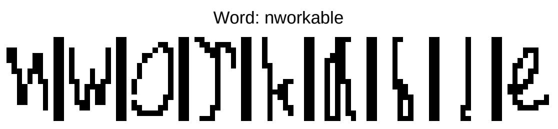

Figure 2: Word sample from dataset: “[U]nworkable”.

Classifier Type	Uncorrected error	Viterbi-corrected error	Reduction in Error
1NN	0.1767	0.1557	0.021
MLP	0.5868	0.4918	0.095
SVM+RBF	0.1614	0.1124	0.049

Table 1: 10-fold cross-validation error rates of classifiers.

2.2.4 Simulated classifier

In order to do more detailed analysis of the Viterbi algorithm’s performance, a simulated classifier was created with a tunable accuracy. For each classification, the simulated classifier randomly selected between the correct choice and a uniform distribution of incorrect choices using a specifiable Bernoulli distribution.

3 Experiment

The dataset used for this experiment is a subset of the dataset used by Robert Kassel at MIT Spoken Language Systems Group for his 1995 PhD thesis. It contains pre-segmented image data of a small vocabulary of handwritten English words. Benjamin Taskar at UC Berkeley selected a ”clean” subset of the words, removed capitalized leading characters, and rasterized and normalized the images of each letter, forming a 6877 word, 52152 character, all-lowercase dataset [8]. Figure 2 is one example of the processed data.

Each classifier was tested using 10-fold cross-validation to calculate overall error rates for each method with and without Viterbi error-correction.

The error-correction proved to be effective in all cases, as shown in Table 1. The SVM and MLP seemed to benefit the most from the correction, even at vastly differing baseline accuracies. The 1-NN, on the other hand, improved much less, suggesting that the type of classifier used significantly affected the ability of the Viterbi algorithm to correct mistakes.

In order to better understand the effects of classifier accuracy on the error-correction, the simulated classifier was tested over a range of accuracies. Figure 3 shows the effect of the classifier’s base accuracy on the Viterbi correction, normalized by the classifier’s base error rate. Interestingly, the proportional reduction in error does not degrade at higher accuracies, implying that the error-correction is beneficial even for very accurate classifiers, although it may be less dramatic.

While this variability may account for the difference in reduction of error between the MLP and SVM, it still does not at all account for the improved performance of the error-correction on these two over the 1-NN and simulated classifiers. The data seems to suggest that the specific errors generated by the former two can be exploited in the emission probabilities of an HMM to improve the performance of the Viterbi algorithm.

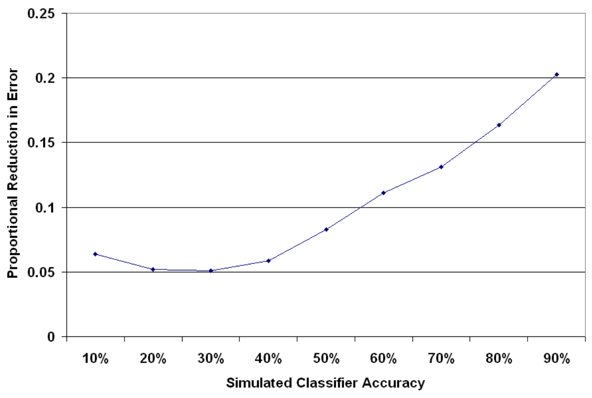

Figure 3: Error reduction divided by error of simulated classifier.

4 Conclusion

I have demonstrated that for limited vocabulary cases, Viterbi error correction is a very useful means of boosting accuracy. Also, preliminary results suggest that the nature of the errors produced by MLP and SVM classifiers are particularly well-suited to Viterbi error-correction, allowing improvements of almost 10% in the former case. However, due to the poor performance overall of the MLP on this dataset, it is unclear whether this might allow a well-trained corrected MLP to consistently match or outperform a well-trained uncorrected SVM.

References

[1] Claus Bahlmann, Bernard Haasdonk, and Hans Burkhardt. On-line handwriting recognition with support vector machines—a kernel approach. In *Proc. of the 8th IWFHR*, pages 49–54, 2002.

[2] Y. Bengio, Y. LeCun, C. Nohl, and C. Burges. Lerec: A nn/hmm hybrid for on-line handwriting recognition. *Neural Computation*, 7(6):1289–1303, November 1995.

[3] G. C. Cawley. MATLAB support vector machine toolbox (v0.55β) [`http://theoval.sys.uea.ac.uk/~gcc/svm/toolbox`]. University of East Anglia, School of Information Systems, Norwich, Norfolk, U.K., 2000.

[4] C. Hsu and C. Lin. A comparison of methods for multi-class support vector machines, 2001.

[5] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems 2 (NIPS*89)*, Denver, CO, 1990. Morgan Kaufman.

[6] C.L. Liu, K. Nakashima, H. Sako, and H. Fujisawa. Handwritten digit recognition: Benchmarking of state-of-the-art techniques. 36(10):2271–2285, October 2003.

[7] Fujisawa H. Liu, C.L. Classification and learning for character recognition: Comparison of methods and remaining problems in neural networks and learning in document analysis and recognition. pages 1–7, August 2005.

[8] B. Taskar. OCR dataset [`http://ai.stanford.edu/ btaskar/ocr/`]. Stanford University, Artificial Intelligence Lab, Stanford, CA, 2003.