Writer Demographic Classification Using Bagging and Boosting

Karthik R BANDI, Sargur N SRIHARI
CEDAR, Department of Computer Science and Engineering
State University of New York at Buffalo, Amherst, NY – 14228, USA.
{kkr2, srihari}@cedar.buffalo.edu

Abstract. Classifying handwriting into a writer demographic category, e.g., gender, age, or handedness of the writer, is useful for more detailed analysis such as writer verification and identification. This paper describes classification into binary demographic categories using document macro features and several different classification methods: a single feed-forward neural network classifier and combinations of neural network classifiers based on boosting and bagging. The textual content of each document was the same, consisting of a full page of writing, and eleven macro features were used. Training set sizes for gender, age and handedness populations were 800, 650 and 300 respectively for each class. Test set sizes were 400, 350 and 150 respectively. Classification accuracy using a single neural network was: 73.2%, 82.7%, and 68.5%. Performance improved with the number of classifiers combined, saturating at around ten neural networks: combined by bagging: 74.7%, 83.4%, and 70.1%; and combined by boosting: 77.5%, 86.6%, and 74.4%. Accuracies achieved by boosting are significantly higher than what has been observed before for demographic classification. Results also show the promise of the boosting technique for weak classifiers.

1. Introduction

Handwriting is affected by several factors such as biological, age, social habits, etc. This intuition motivates the study of classification techniques suitable for identifying demographic classes, given a handwritten document. The demographic classes are: male-female, left handed-right handed and different age groups. In most cases where handwriting is used as evidence, few global features extracted from checks, tax forms, etc. are at the disposal of the forensic document examiners. In these cases, the knowledge of the individual features can be used to estimate the validity of the writer verification/identification tasks or to improve their accuracy. Since in some cases partial information about the writer of the query document is available from other sources (his/her gender, approximate age, etc) we also evaluate how useful this information is to perform faster and more accurate writer verification.

In the works of (Tomai, Kshirsagar & Srihari, 2004) there is a detailed analysis of identifying features that best discriminate between demographic classes (using k-nearest-neighbor classification), and then using this information to enhance the writer verification task. In their work, the k nearest neighbor classifier used a set of features called micro features based on the individual characters written by each writer. The micro features use gradient, structural and concavity (Favata & Srikantan, 1996) of each character and encode them in a 512-bit feature vector. Our paper is only concerned with demographic classification and the approach we have used is different: using a neural network classifier and improving performance by combining several neural networks (using bagging and boosting). We have used a set of features called macro features, which are global to the document unlike the micro features, which are specific to a single character occurring in the document.

2. Writing Element Extraction

Writing elements are the discriminating features of handwriting. There are many descriptions of writing elements for questioned document examination, *e.g.*, there are 21 classes of discriminating elements (Huber & Headrick, 1999). In order to match elements between two documents, the presence of the elements are first recognized in each document. Elements, or features, that capture the global characteristics of writer's individual writing habit and style can be regarded to be macro-features and features that capture finer details at the character level as micro-features. In this paper we are concerned with the global or macro features only. The system captures 12 macro features as described by (Srihari, Cha, Arora & Lee, 2002). They are features such as slant, word gap, grey-scale threshold, etc. This study used eleven of the twelve-macro features. The feature "number of black pixels" was not used since it did not contribute towards demographic classification.

3. Handwriting database

The CEDAR letter database consists of more than 3,000 handwritten document images written by more than 1,000 writers representative of the US population. Each individual provided three handwriting samples of the same text. The letter contain all the 26 alphabets in upper case and lower case and the 10 digits. Each writer has contributed to 3 documents. The database is stratified along six categories: gender (G) male, female, handedness (H) right, left, age (A) under 15, 15-24, 25-44, 45-64, 65-84, over 85, ethnicity (E) white, black, Hispanic, Asian and pacific islander, American Indian, Eskimo, Aleut, highest level of education (D) below high school graduate, above, and place of schooling (S) USA, Foreign. The writer's population is unequally distributed in these categories: for example, there are 443 X 3 male documents and 1119 X 3 female documents, 599 X 3 for high

school graduates, 276 X 3 for bachelor's degree holders, 1406 X 3 right-handed, 156 X 3 left-handed, 337 X 3 under the age of 24 and 347 X 3 above the age of 45.

This study considered the following three binary classification problems- *gender*: male *vs* female, *handedness*: left *vs* right, and *age*: below 24 *vs* above 45. The training set size for each class in gender, age and handedness populations were 800, 650 and 300 respectively. The test sets for these were of size 400, 350 and 150 respectively. The choice of data set size was based on the availability of data and for maintaining equal number of samples of either class in a given 2-category classification problem. If this is not done, then the classification could be biased towards the class, which has more number of samples. Fig 1 shows the result of applying principal component analysis on the gender data set. From the plot we can observe that the classification task is hard and a single classifier may not necessarily achieve good performance.

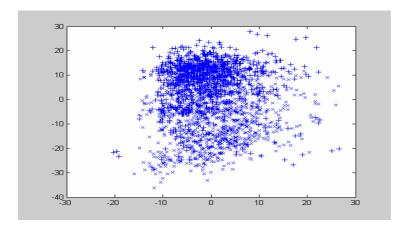


Fig 1: Plot after applying PCA and choosing best two components. '+' represents female population and 'x' represents male population. The two axes correspond to the best two principal components.

4. Classifiers

The performance of a single neural network classifier as well as combined neural network classifiers were evaluated. In order to combine the classifiers both bagging and boosting (AdaBoost.M1) techniques were used. Classifier performance, with the number of combined classifiers ranging from two to ten, were determined. The effect on performance by varying the number of features input to the classifier was also analyzed. The classifier used was a feed-forward neural network classifier with three layers. The transfer function used at every neuron is the log sigmoid function, the number of inputs is the number of input features and the number of outputs is 1. All problems we have considered were 2-class problems. Thus 0.05 represents class A and 0.95 represents class B, for any two classes A and B. In all experiments the size of the test set was roughly half the size of the training set

4.1 Classifier combination: Boosting (Ada Boost)

Boosting is a general technique used to improve the performance of any learning algorithm that consistently generates classifiers with misclassification errors lesser than 50% on a given problem. Schapire (Schapire, 1990) introduced the boosting algorithm. More recently, Freund and Schapire (Freund & Schapire, 1996; Freund & Schapire, 1997) introduced AdaBoost that has undergone intense theoretical study and empirical testing in the last few years. The AdaBoost.M1 algorithm (Freund & Schapire, 1996) was implemented for boosting by resampling, whose pseudo code is given below. AdaBoost.M1 takes as inputs a weak classification method (Weak Learn) and a dataset $D = \{(x_i, y_i), I=1, m\}$ with m examples, where x_i is a feature vector and $x_i \in X$ (instance space), y_i is the corresponding class label and $y_i \in Y$ (set of class labels). Here each problem has only two class labels. In each iteration t, the algorithm repeatedly calls Weak Learn and applies this algorithm on a training dataset D_t. The training set D_t is obtained by using the distribution weight_t for re-sampling from the data set D. Weak Learn finds a new classifier $h_i: X \to Y$ seeking to minimize the training error ε_t on the data set D_t . This process is iterated T times, and the hypotheses h_1, \dots, h_T obtained are combined in a final classifier h final. The re-weighting of examples in D and the way of combining the successive hypotheses h_t are indicated in the algorithm's pseudo code below. The main theme in boosting is, examples that are correctly classified by most previous hypotheses get a small weight, while examples usually wrongly classified get larger weights. Thus, boosting concentrates the efforts of the learning algorithm on those examples that are difficult to learn.

AdaBoost.M1 Algorithm

```
Input: Data set D={(x_i, y_i), I=1,m}, where x_i \in X and y_i \in \forall I \{1...m\}
Base learning algorithm Weak Learn
Initialize: weight _1(i) = 1/m \quad \forall i.
For t = 1 to T:
• Generate data set D<sub>t</sub> by re-sampling m examples from D using distribution weight t
• Train Weak Learn on D_t to obtain the base classifier h_t: X \to Y
• Compute the error of h_t: \varepsilon_t = \sum_{i:ht(xi)\neq yi} weight_t(i). If \varepsilon_t > \frac{1}{2} then set T = t-1 and exit;
• assign \beta_t = \varepsilon_t / (1 - \varepsilon_t)
• update distribution weight <sub>t</sub>:
weight_t+1(i) = Weight_t(i)*\beta_t if h_t(x_i) = y_i;
                          Weight_t(i) \times 1 otherwise;
• normalize weight_t+1 to make it a distribution.
Weight_t+1(i) = weight_t+1(i)/(N_t) for I=1..m; N_t is the normalization constant
N_t = \sum_{i=1}^{m} Weight_{t+1}(i)
Output: The Final hypothesis
   H_{final}(x) = arg max (y \in Y) \sum_{t:ht(x)=y} log(1/\beta_t)
```

4.2 Bagging

Bagging is an ensemble method that creates individual classifiers for the ensemble by training each classifier on a random redistribution of the training set. The training set for each classifier is generated by randomly drawing N examples from the training set – where N is the size of the training set. It is possible that many of the original examples may be repeated in the training sets for each classifier, while others may be left out. Classifier combination by Bagging and how it reduces error has been discussed in (Breiman, 1996; Ali & Pazzani, 1996). Bagging is simple: it basically trains T base classifiers on bootstrap re-samples (Efron & Tibshirani, 1993) of the training data set and outputs a final hypothesis on the basis of plurality voting. In bagging, the re-sampling of the training set is not dependent on the performance of the earlier classifiers, unlike Boosting. For our experimental analysis we have also implemented bagging for the classifier combination task and compared the performance with boosting.

5. Experimental Results

Performance based on number of features considered: The graphs of Figure 2 show how the number of features input to the classifier can affect performance. As expected the performance increases as the number of features considered is greater. The results consider only a single neural network classifier on test data sets. Using a single neural network, performance on gender, age and handedness were 73.2%, 82.7%, and 68.5% respectively.

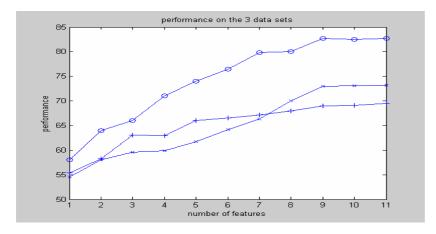


Fig 2: Performance improvement with the number of features on all 3 test data sets (male-female '+', left – right handed 'x', age < 24 – age > 45 'o').

Performance based on all features using Bagging, Boosting (test data sets): Combining ten neural networks (Figs. 3(a)-(c)) produced the following results- bagging: 74.7%, 83.4%, and 70.1%, and boosting: 77.5%, 86.6%, and 74.4%.

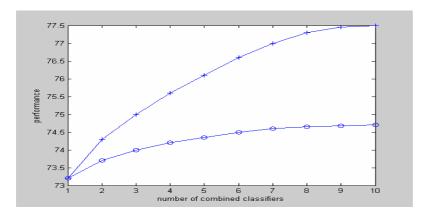


Fig 3 (a): Results of Bagging ('o') and Boosting ('+') based on number of classifiers combined (male – female population).

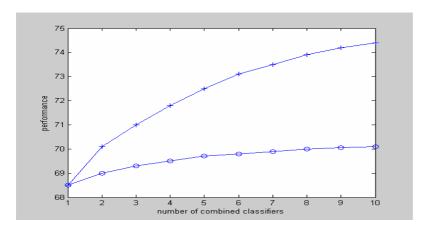


Fig 3 (b): Results of Bagging ('o') and Boosting ('+') based on number of classifiers combined (left handed – right handed population).

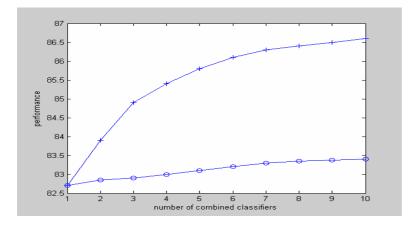


Fig 3 (c): Results of Bagging ('o') and Boosting ('+') based on number of classifiers combined (age < 24 – age > 45 population).

Performance of the Ada-Boost algorithm is clearly better then the bagging approach. From the graphs in Figures 3(a)-(c) it is also evident that increasing the number of classifiers improves performance for both bagging and boosting.

6. Concluding Remarks

Demographic classification of handwriting can be performed with nearly 75% accuracy for gender, age and handedness. Bagging and boosting methods improve classification performance. In our work we have considered documents with same content. Demographic classification considering documents with different content would need to consider micro (character) features since macro features tend to be document-content dependent. The problems we have considered are binary demographic classification problems. In the future we shall extend our work to non-binary categories for demographic classification.

References

Tomai, C.I., Kshirsagar, D., Srihari, S.N. (2004). Group Discriminatory power of handwritten characters. *Document Recognition and Retrieval XI. Proc. of SPIE-IS&T Electronic Imaging*, Vol 5296: 116-123.

Favata, J.T. and Srikantan, G. (1996). A multiple feature/resolution approach to hand printed digit and character recognition. *International Journal of Imaging Systems and Technology*, 7:304–311.

Huber, R.A., and Headrick, A.M. (1999). Handwriting Identification: Facts and Fundamentals, Boca Roton: CRC Press.

Srihari, S.N., Cha, S.H., Arora, H. and Lee, S. (2002). Individuality of Handwriting. *Journal of Forensic Sciences*, 44(4), 856-72.

Schapire, R.E. (1990). The strength of weak learnability. *Machine Learning* 5(2): 197-227.

Freund, Y., and Schapire, R.E. (1997). A decision theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1), 119-139.

Freund, Y., and Schapire, R.E. (1996). Experiments with a New Boosting Algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148-156.

Breiman, L. (1996). Bagging predictors. Machine Learning, 24(2): 123-140.

Ali, K. and Pazzani, M. (1996). Error reduction through learning multiple descriptions. *Machine Learning*, 24(3): 173-202. Efron, B. and Tibshirani, R. (1993). An Introduction to the Bootstrap. Chapman and Hall, New York.