Joint Feature and Classifier Design for OCR

Dz-Mou Jung and George Nagy
Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute
Troy, NY, 12180-3590, USA

Abstract

Shift-invariant, custom designed n-tuple features are combined with a probabilistic decision tree to classify isolated printed characters. The feature probabilities are estimated using a novel compound Bayesian procedure in order to delay the fall-off in classification accuracy with tree size due to a small sample set. On a ten-class confusion set of eight-point characters, the method yields error rates under 1% with only 3 training samples per class.

1 Motivation

We address the restricted problem of automating the design of a recognition system for isolated single-font, single-size characters. Such a kernel algorithm is essential for adaptive recognition along the lines investigated in [14, 1].

In view of our ultimate objective of truly omnifont (multilingual) recognition, we need to automate the design not only of the classifier, but also of the features, so that the system can accommodate *any* typeface that it might encounter. And, in order to satisfy the requirement of adaptivity on short segments of text, we must be able to design the entire system using only a few training samples of each class.

Features based on expansions of the patterns, such as moment invariants or orthogonal kernel transforms (Fourier, Haar, Walsh, Hadamard), are optimal for pattern *reconstruction*, but reveal *differences* between the character classes only incidentally. On the other hand, intuitively attractive structural features fail the criterion of complete automation. We are not aware of *any* method for completely automatic feature design.

We can, however, design good pixel-based classifiers for two categories, which can be used as features in a binary tree classifier. Pixel-based classifiers called *n-tuples* were introduced by Bledsoe and Browning at the Eastern Joint Computer Conference [2], investigated extensively in [9, 10, 11, 12, 3], reviewed favorably in [13], and revived recently in [19]. These researchers applied n-tuples directly to multicategory classification, without exploiting the considerable advantage to be gained by designing each tuple explicitly for two-category classification. Yet the superiority of n-tuples resides precisely in the possibility of configuring them to discriminate between two arbitrary sets of prototype characters. This possibility is best realized through hierarchical classification.

In a binary tree classifier, the patterns arriving at a node are partitioned into two categories, each of which is usually a mixture of several character classes. Each node decision is determined by whether the n-tuple assigned to the node fits or does not fit the unknown pattern. The leaves of the tree carry the label of a single character class. Several leaves may correspond to the same character, and leaves may also be labeled as "reject". Since the average depth of the tree is typically only 12 or 14, only a few n-tuples must be tested to classify an unknown character.

Tree-based classification also has a long pedigree. Distinguished antecedents include, among many others, fuzzy trees [15, 6], CART (Classification and Regression Trees) [4], inductive learning trees [16], feature-based trees [20, 21], Olivetti's OCR system for office documents [7], and Shlien's multi-trees [17, 18]. Our trees derive from the probabilistic design algorithm presented in [5], and the excellent information-theoretic analysis of [8].

2 N-tuples for two-category discrimination

A typical 7-tuple for a c-e vs. n-r dichotomy is shown in Fig. 1a. The n-tuple fits the design prototypes "c" and "e" (Fig. 1b) in $p = 4$ different shift positions (for instance, if the coordinate of the upper left corner of the "e" prototype is (0,0), with x increasing to the right and y down, then the tuple fits "e" when the topmost 'b' of the tuple moves to (3,5), (4,5), (0,13), (1,13), and (2,13).), and it fails to fit ei-

0-8186-7128-9/95 $4.00

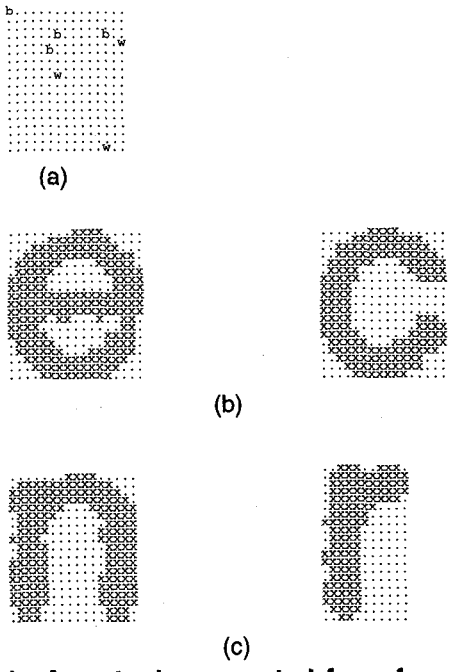

Figure 1: An n-tuple generated from four prototypes. (a) A 7-tuple with 4 black ('b') pixels and 3 white ('w') pixels. (b) Prototypes of the positive classes "e" and "c", where 'x' represents a black pixel and '.' white. (c) Prototypes of the negative classes "n" and "r".

ther the prototype "n" or "r" (Fig. 1c) in any position (in the best-fit position, only $q = 5$ of the 7 elements of the tuple fit either the "n" or the "r", An example best-fit position is when the topmost 'b' of the tuple is shifted to (3,2) of the "n".). It is clear that a one- or two-pixel variation in any of the four characters would not prevent a fit on "c" and "e", or cause a false fit for "n" and "r": the tuple is "noise-resistant."

We say that this n-tuple was designed for *positive* prototypes "c" and "e" and *negative* prototypes "n" and "r", with design parameters $n = 7, p = 4$, $q = 5$. In general, having a high value of n decreases the probability of a (perfect) fit of characters of the classes represented by the positive prototypes, but increases the probability of a misfit of characters of the classes represented by the negative prototypes.

High values of p generally result in a high probability of fitting any member of the positive classes. Low values of q yield a low probability of fitting characters from the negative classes. Good n-tuples therefore have high p and low q.

The tree-design procedure requires that n-tuples that appear on any path be class-conditionally independent. Computations of pairwise correlations indicate that our tuple generation procedure does indeed satisfy the independence criterion.

The input to the n-tuple generator program consists of the design parameters (n, p, q) and of the bitmaps of the positive and negative prototypes. The program either produces an n-tuple that satisfies the specifications or returns after exhausting its time allocation. (In that case, it is called again with a different set of prototypes.) During the design of a single classification tree, the n-tuple generator may be called several thousand times.

3 Decision tree design

In principle, the design of a probabilistic tree requires only the iteration of two steps [5]:

1. Choose the next leaf node to be expanded.
2. Select the best feature for the chosen node.

The first step is accomplished by choosing the node with the maximum entropy, calculated from the class probabilities at each of the current leaves.

In the second step, the positive and negative classes are determined by clustering the design samples corresponding to the prevalent node class probabilities. One or more prototypes are then chosen from each positive character class and each negative class to serve as input to the n-tuple generator. The class-conditional feature probabilities corresponding to the returned n-tuple are estimated, and the class probabilities of the two child nodes are computed.

A well-known phenomenon with decision trees is that their classification accuracy declines when the trees are extended beyond a certain point that depends on the number of design samples. This problem arises because as the tree grows we must estimate an increasing number of leaf class probabilities using a constant number of design samples. (The a priori class probabilities of the classes at the root are essentially distributed among all the leaves.)

Breiman et al. advocate using only part of the training samples in tree design and saving the rest for cross-validation [4], but we adopt a different approach, called Compound Bayesian Estimation. The parameters to be estimated are the class-conditional feature probabilities for each tuple X used in the tree.

The value of the parameter of the Bernoulli process that models whether a tuple X fits an independently drawn sample of a given class depends both on the prototypes on which the tuple was generated and on the character class.

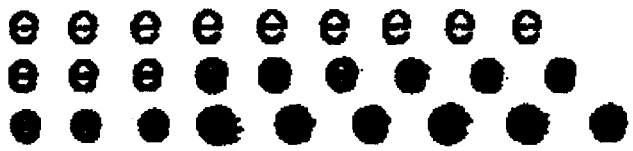

Figure 2: Enlarged character bitmaps sampled from Helvetica 8-point e's. The top, middle, and bottom rows are originals, 2nd-, and 5th-generation photocopies, respectively.

Instead of using a fixed a priori probability distribution for the Bernoulli parameter, we *estimate* the a priori distribution by generating additional (predictor) n-tuples, which we test on the design samples. The underlying assumption is a form of ergodicity, which stipulates that the ensemble distribution of the fit-probabilities of predictor n-tuples generated by identical processes on the design sample is the same as that of the n-tuples on the (unavailable) test samples. The a priori distribution itself is assumed to be a Beta distribution, whose two parameters are estimated using the predictor n-tuples and the design samples. The a priori a priori (sic) distribution of the parameters of the Beta distribution are assumed to be uniform over $(-1, 9] \times (-1, 9]$.

4 Experimental data

We prepared three "stress" Helvetica data sets for designing and testing our n-tuple-based decision trees. Each data set is of one of three print qualities. Fig. 2 shows examples of e's. The top, middle, and bottom rows contain originals, 2nd-generation, and 5th- generation copies, respectively.

Each of the 3 data sets contains 10 alphabetic classes of 1,000 8-point characters each. The total number of samples is therefore 90,000. All samples are scanned at 300 dpi, which is generally considered the limiting sampling rate for 8-point characters. The 10 classes are: a, c, e, n, o, r, s, u, x, z. We expect the major confusion classes to be a-s, c-e-o, r-n- u, and x-z.

For each print quality, 100 characters per class are randomly selected and reserved for training purposes. The remaining 900 characters of each class constitute the test set. In the experiments to be described, each tree was designed on only a very small fraction of the training sets.

Table 1: Average error rate on sets of nine trees.

	Compound Bayes	Maximum Likelihood
Hel-0	0.009	0.031
Hel-2	0.044	0.095
Hel-5	0.248	0.336

5 Experimental results

Table 1 shows the average accuracy of nine trees designed with compound Bayesian estimation (called Compound Bayes Trees), compared with those designed with maximum likelihood estimation. Each tree was designed on different samples of only *three* characters of each class, and tested on the 9000-sample test set. The size of the training set was set at three because at each node one sample is required for designing the n-tuple, and at least two are necessary to estimate the two parameters of the Beta distribution. The expansion of the nodes was halted when the tree size reached 1024 leaves. The median error-rate compound Bayes tree for the original characters made only 36 substitution errors of the following 7 types:

e → o	22	(22 e's were mistaken as o's)	
o → c	5	a → s	3
o → e	2	o → u	2
a → u	1	c → r	1

The Maximum Likelihood Trees reached zero estimated error at every node long before this limit was reached, and therefore could not be expanded further. This is an intrinsic disadvantage of small-sample maximum likelihood estimation. Although the variances in the error rate are large, the Compound Bayes Tree maintains an advantage over the Maximum Likelihood Tree on every pair generated from the same training set. Both the Bayesian and the Maximum likelihood trees are well balanced.

The most encouraging aspect of the results shown in Table 1 is that they confirm our hypothesis: predictor features improve the accuracy of the estimation of the node class probabilities sufficiently to construct large trees from very small training samples. Indeed, the error rate decreases nearly monotonically for each doubling of the tree size. The errors are concentrated at very few leaf nodes.

As a check on the contribution of the information from the predictor n-tuples, we also carried out similar experiments using simple Bayesian estimation based on the uniform a priori density for the parameter of the Bernoulli process. The resulting error rate was much higher than with either of the above procedures.

In total 81 trees (including simple Bayes trees), 56,401 node-n-tuples, and 156,200 predictor n-tuples

were generated. On average the design of each Compound Bayes Tree for original characters took 13.5 hours of CPU time on a SUN SPARC 10 with 32 MB RAM, and about 70% longer on fifth-generation copies. Testing each tree on 9000 isolated characters took 90 seconds, but this included considerable overhead for collecting statistics.

6 Conclusion

Our experiments suggest that Compound Bayesian Estimation often allows extending the useful size of probabilistic classification trees beyond that possible with either maximum likelihood estimation, or with simple Bayesian estimation.

Even at its current rate of development, the method yields respectable classification accuracy on a confusion set of noisy characters of small point size. The design is completely automated and produces both features and classifier custom-tailored to a small design sample. Our current research addresses the many issues, including segmentation, that must be resolved for practical application of the method.

Acknowledgments

This work has been conducted in the New York State Center for Advanced Technology (CAT) in Automation, Robotics and Manufacturing at Rensselaer Polytechnic Institute. The CAT is partially funded by a block grant from the New York State Science and Technology Foundation. The sponsorship of the Central Research Laboratory, Hitachi, Ltd. is gratefully acknowledged. We are indebted to A. Shapira for the n-tuple generation program, and H. S. Baird, R. G. Casey, and C. N. Liu for valuable suggestions.

References

[1] H.S. Baird and G. Nagy, "A Self-Correcting 100-Font Classifier," *Proc. SPIE Symp. Document Recognition*, SPIE Vol. 2181, 106-115, Feb. 1994.

[2] W.W. Bledsoe and I. Browning, "Pattern Recognition and Reading by Machine," *Proc. EJCC*, 225-233, 1959.

[3] R. Bakis, N.M. Herbst, G. Nagy, "An Experimental Study of Machine Recognition of Handprinted Numerals," *IEEE Trans.* SMC-4, 2, 119-132, July 1968.

[4] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, *Classification and Regression Trees*, Wadsworth & Brooks, Monterey, CA 1984.

[5] R.G. Casey and G. Nagy, "Decision Tree Design Using a Probabilistic Model," *IEEE Trans.* IT-30, 93-99, 1984.

[6] R.L.P. Chang and T. Pavlidis, "Fuzzy Decision Tree Algorithms," *IEEE Trans.* SMC-7, 1, 28-35, Jan. 1977.

[7] G. Ciardiello, M. T. Degrandi, M. P. Roccotelli, G. Scafuro, M.R. Spada, "An Experimental System for Office Document Handling and Text Recognition," *Proc. ICPR-9*, Rome, 739-743, 1988.

[8] R. M. Goodman and P. Smyth, "Decision Tree Design from a Communication Theory Standpoint," *IEEE Trans.* IT-34, 979-994, 1988.

[9] L. A. Kamentsky and C. N. Liu, "Computer-Automated Design of Multifont Print Recognition Logic," *IBM J. Res. and Dev.* 7, 1, 2-13, 1963.

[10] L. A. Kamentsky and C. N. Liu, "Theoretical and Experimental Study of a Model for Pattern Recognition," *Computer and Information Sciences*, Spartan, Washington, DC 194-218, 1964.

[11] C. N. Liu, "A Programmed Algorithm for Designing Multifont Character Recognition Logics," *IEEE Trans.* EC-13, 586-593, Oct. 1964.

[12] C. N. Liu and G. L. Shelton, "An Experimental Investigation of a Mixed Font Print Recognition System," *IEEE Trans.* EC-15, 6, 916-925, December 1966.

[13] M. Nadler, "The State of the Art in Optical Character Recognition," in *Machine Perception of Patterns and Pictures*, Inst. of Physics, Teddington, Middlesex, 3-18, Apr. 1972.

[14] G. Nagy and G.L. Shelton, "Self-corrective Character Recognition System," *IEEE Trans.*, IT-12, 2, 215-222, Apr. 1966.

[15] T. Pavlidis and F. Ali, "Computer Recognition of Handwritten Numerals by Polygonal Approximation," *IEEE Trans.* SMC-5, 610-614, 1975.

[16] J.R. Quinlan, "Induction of Decision Trees", *Machine Learning* 1, 1, 81-106, 1986.

[17] S. Shlien, "Multiple Binary Decision Tree Classifiers," *Pattern Recognition* 23, 7, 757-763, 1990.

[18] S. Shlien, "Nonparametric Classification Using Matched Binary Decision Trees," *Pattern Recognition Letters* 13, 83-87, 1992.

[19] F.W.M. Stentiford, "Automatic Feature Design for Optical Character Recognition Using an Evolutionary Search Procedure," *IEEE Trans.* PAMI-7, 3, 349-355, May 1985.

[20] Q.R. Wang and C.Y. Suen, "Analysis and Design of Decision Tree Based on Entropy Reduction and its Application to Large Character Set Recognition," *IEEE Trans.* PAMI-6, 406-417, 1984.

[21] Q.R. Wang and C.Y. Suen, "Large Tree Classifier with Heuristic Search and Global Training," *IEEE Trans.* PAMI-9, 1, 91-102, 1987.