High-performance OCR preclassification trees

Henry S. Baird & Colin L. Mallows

AT&T Bell Laboratories
600 Mountain Avenue, Room 2C-322
Murray Hill, NJ 07974-0636

ABSTRACT

We present an automatic method for constructing high-performance preclassification decision trees for OCR. Good preclassifiers must prune the set of alternative classes to a small number without erroneously pruning the correct class. We build the decision tree using greedy entropy minimization, using pseudo-randomly generated training samples derived from a model of imaging defects, and then ''populate'' the tree with many more samples to drive down the error rate. We describe a refinement of the method of [BM94] that approaches the user-specified accuracy more closely and thus allows higher pruning. The essential technical device is a leaf-selection rule based on the Good-Turing Theorem [Good53]. Such a preclassifier, constructed for a pan-European polyfont classifier, attains a 1% error rate and a 3.8 pruning factor, in tests on synthetic images. On real pages printed in ten European languages, the preclassifier sped up the page reader by a factor of 2.2, with no measurable increase in error.

Keywords: decision tree, classification, character recognition, image defect models, learning, population

1. Introduction

Non-backtracking decision trees promise fast classification. Inferring optimal trees, under various measures, has been shown to be computationally infeasible [Bun87] in the worst case. However, when applied to OCR problems, suboptimal heuristics often build roughly balanced, strongly pruning trees [CN84,WS87]. Many such heuristics — including the one we use — have the serious practical drawback that error accumulates rapidly as the tree deepens, when using a training set of fixed size.

We examine the role of decision trees as preclassifiers in a multi-stage decision strategy without backtracking. Preclassification trees prune the set of classes to a smaller number: that is, each leaf of the tree owns a subset of the classes. Pruning the true class is an irrecoverable error. True classifiers, able to prune the set of classes to a single choice, are generally slower: when executed downstream of the preclassifier, it will be presented with fewer classes to be distinguished and, in our case, will run proportionally faster. Thus a good preclassifier is one

This is an expanded and corrected version of a paper presented at the IS&T/SPIE Symposium on Electronic Imaging: Science and Technology, February 5-10, 1995, San Jose, CA, and to be published in *IS&T/SPIE Proceedings Vol. 2422, Document Recognition II.*

that rarely prunes the true class but otherwise prunes as many classes as possible.

It is often possible to construct shallow preclassification trees with an acceptably low probability of error. In our trials, we build the tree initially using a greedy entropy-minimization heuristic, using as many sample images as our computing resources comfortably admit (in this latest trial, 1,066,639 samples), with a pruning factor of 9. Trees as deep and strongly pruning as this usually exhibit unacceptable error: in our case, 15% on a distinct test set.

We have experimented with massive generation of sample images to drive down this error rate by brute force. In an early stage of this research [Bai93], we built preclassification trees that achieve an error rate of less than 1.0% and a pruning factor of ×5.2, on polyfont ASCII text printed at type sizes 10 point and higher and at a spatial sampling rate of 300 pixels/inch (ppi). In [BM94], we described a refinement that guarantees any user-specified upper bound on the error rate. Viewing the decision tree as a black box, this applied a statistically motivated stopping rule which in practice was rather conservative, achieving 1/7 the target error.

We now describe an alternative method that achieves the user-specified error bound more tightly and thus improves the pruning rate of the resulting tree. The method exploits the structure of the tree and does not require a stopping rule.

We describe the engineering context in Section 2, and the derivation of a leaf-selection rule in Section 3. Experimental trials, on both synthetic and real images, are described in Section 4. Section 5 contains conclusions and discussion.

2. The Engineering Context

We now briefly sketch the engineering context of this work, including the application, the source of image samples, features and classifiers, binary decision trees, and building and testing the trees.

2.1. The Application

The context of this effort was a project to build a classifier subsystem for a family of machine-print page readers for ten European languages [BGI94]. This required distinguishing 209 character symbols (all of ASCII and Latin-1, plus a few Turkish symbols), in 20 typefaces, over the range of type sizes 8-12 point (at 300 ppi).

2.2. Image Samples

We used a quantitative model of imaging defects [Bai92] with parameters for type size, spatial sampling rate (digitizing resolution), blur, binarization threshold, pixel sensitivity variations, jitter, skew, stretching, height above baseline, and kerning. Associated with it is a pseudo-random defect generator that reads one or more sample images of a symbol — in this trial, these are high-resolution noiseless artwork purchased from typeface manufacturers — and writes an arbitrarily large number of distorted versions exhibiting a user-specified distribution over the model parameters. These distributions have been roughly calibrated on image populations occurring in printed books and typewritten documents. During training, including tree-building, we use only synthetic images. We test on both synthetic data (for consistency checking and debugging) and on real data from printed and scanned paper documents.

2.3. Features and Classification

The classifier technology used here is described in [Bai88]. Briefly, it extracts local geometric shapes from the input image of an isolated symbol, maps this diverse collection of shapes into a feature vector with binary components. This binary feature vector is designed to be insensitive to location and type size; the preclassifier will examine only this vector. From a training set labeled with true class, type size, etc, we infer a single-stage

Bayesian classifier under an assumption of class-conditional independence among the features. This is the ''main'' classifier for which we need a preclassifier. Its runtime is $O(\ C\ (F + \log C)\ )$, where C is the number of classes to be distinguished at runtime, and F the number of binary features.

In this trial, $C = 4032$ classes (one for each <typeface,symbol> pair for which we have artwork), and $F = 704$ features.

2.4. Decision Trees

At each node of our decision trees, a single feature is tested. Thus these are binary decision trees. Each leaf of the tree contains a subset of the classes. An input image, represented by its feature vector, is said to be correctly preclassified if the leaf it arrives at contains its true class.

2.5. Building a Tree

The tree-growing heuristic is to execute a sequence of *splits* (<leaf,feature> pairs) until a stated pruning factor is reached. At each step, examine all possible next splits and choose the one which most decreases the expected entropy of the tree. The tree's expected entropy and pruning factor are estimated on the assumption that the distribution of the training data is representative: this depends on the validity of the image defect model. The method is greedy, with no look-ahead: multiple splits are not examined at each step.

Even with the short-cuts of this sub-optimal heuristic, the large scale of our problem strained our computing resources. We use a Silicon Graphics Computer Systems Challenge XL, with 150 MHz R4.4k processors, running time-shared UNIX. In order for the program to terminate in reasonable time, it was necessary throughout the tree-building phase to hold in main memory all training data and the growing tree itself. As a result, the number of training samples was effectively limited to about a million: precisely, 1,066,639, or 3/4 of the number used to build the Bayesian classifier. After 32.5 CPU hours of tree-growing, the pruning factor reached ×9. The ×9 tree contained 932 decision nodes and 933 leaves. Before the pruning factor could reach ×10, the size of the running process exceeded 512 Mbytes, a hard system limit, and the run terminated abnormally.

2.6. Testing the Tree

The trees are perfect on the training set by construction. Testing on a distinct set of 1,030,000 synthetic samples revealed a 15% error rate (1000 for each <typeface,size,symbol> triple for the single typeface Avant Garde-Book Oblique and the five integer type sizes 8-12 point). In the same test, the pruning factor was measured as ×9.3, slightly higher than on the training data.

2.7. Populating the Trees

In ''population'' of the tree, we use a sample set distinct from the training set to change the contents of the leaves without changing the structure of the decision nodes. We first empty the contents of the leaves (throwing away the class sets left over from the training data). Then, we generate a large number of distinct population samples N (10000 for each <typeface,symbol> pair) and assign them to the leaves, counting the number assigned to each leaf. Finally, we select a subset of the leaves and mark them with the symbol class, in order to achieve as nearly as possible the desired error rate.

3. A Leaf-Selection Rule

Consider a single symbol class. Given sample-occurrence counts at the leaves, we wish to select the smallest possible number of leaves to be marked with this class. Formally, let p_i denote the probability that a sample (from this class) is assigned to leaf i. In our application n = 933 and there are very many small p's. The problem is to identify a set S of leaves, with $|S|$ as small as possible, such that the total probability content of the leaves in S is close to a given target t (actually $t = .99$). If we knew the p's, we would simply sort them and take the largest ones first; but if only sample data is available it is not clear how to proceed. We have considered both sequential and non-sequential versions of the problem. In the simplest sequential version, discussed in [BM94], we want a stopping rule such that when we stop, the total probability of the leaves that have been visited is close to the target t. In a modified sequential version, we allow sampling to proceed further than this, and apply some selection rule to decide which leaves to include in S. In a non-sequential version, we simply use knowledge of the structure of the tree to choose a sample size N.

We have not seen any previous attack on this problem. Good [Good53], following up an observation by Turing, showed that after N cases have been processed, giving for each k a number N_k of leaves that have been visited exactly k times, then the total probability of the leaves that have been seen exactly k times can be estimated by $(k+1)N_{k+1}/N$. We prove this result below. This result suggests a procedure for the non-sequential version of the problem, i.e. when the number of test cases N is fixed, but does not immediately help in assessing the expected performance of the procedure.

In our application, it seems that as sampling proceeds, the sum of the probabilities of the leaves that have been visited (at least once) increases along a roughly exponential curve. We suggest a simple one-parameter model that has this feature. Simulation suggests that the resulting procedure works well in our application.

3.1. The Good-Turing theorem

We use the notation set up above. Fix N, and suppose we run N test cases. Let Z_j be the number of cases that are assigned to leaf i. Let B_k be the set of leaves that are each hit exactly k times:

$$B_k = \{i : Z_i = k\}$$

The true probability content of this set of leaves is

$$P_k = \sum_{j \in B_k} p_j$$

The expectation of this is

$$E(P_k) = \sum_j p_j \binom{N}{k} p_j^k (1-p_j)^{N-k}$$

However the empirical coverage of this set of leaves is $Q_k = k|B_k|/N$, which has expectation

$$E(Q_k) = \frac{k}{N} \sum_j \binom{N}{k} p_j^k (1-p_j)^{N-k}$$

so that we have

$$E(Q_{k+1}) \approx E(P_k)$$

If we denote the number of test cases by a superscript, we have exactly

$$E(Q_{k+1}^{(N+1)}) = E(P_k^{(N)}).$$

3.2. The Dirichlet model.

Suppose a sequence of m test cases is observed. We describe the realization by the notation

$$C_1, Y_{11}, C_2, Y_{21}, Y_{22}, C_3, Y_{31}, Y_{32}, Y_{33}, \cdots, Y_{KK} \tag{1}$$

where C_1 is the index of the first leaf to be visited, Y_1 is the number of times this leaf is repeated before a different leaf is seen, C_2 is the index of the second leaf to be visited, Y_{11} is the number of hits on the first leaf between the first time the second leaf is seen and the first time a third leaf is seen, Y_{22} is the number of times the second leaf is seen before a third leaf appears, and so on; Y_{ij} is the number of times the j-th leaf that appears is seen between the times the i-th and $i+1$-th leaves appear. The number of different leaves that are ever seen is K, which is a random variable. The probability of this realization is

$$p_{C_1}^{Y_{+1}} p_{C_2}^{Y_{+2}} \cdots p_K^{Y_{KK}}$$

where $Y_{+j} = \sum_{i=j}^{K} Y_{ij}$.

Since we do not know the p's, we will assume they have a prior distribution. It is very convenient to make this symmetric, since then in our analysis we will not need to keep track of the identities of the various leaves. There is no loss of generality in this assumption if we relabel the leaves arbitrarily. We postulate a Dirichlet distribution with parameters n, α for the n unknown cell-probabilities p_i, i.e.

$$f(p_1, \cdots p_n) = c_n(\alpha) p_1^{\alpha - 1} p_2^{\alpha - 1} \cdots p_n^{\alpha - 1}$$

where the constant $c_n(\alpha)$ is $\Gamma(n\alpha)/\Gamma(\alpha)^n$. This distribution is well-known to have some very elegant properties, in particular: the joint distribution of $(p_1, ..., p_j)$, conditional on their sum $s = p_1 + \cdots + p_j$, is the same at that of $sr_1, \cdots, sr_j$ where the joint distribution of $r_1, \cdots, r_j$ is Dirichlet with parameters j, α.

Assuming this prior, we can write down the joint probability density for the event that $K = k, p_{C_1} = p_1, p_{C_2} = p_2, \cdots p_{C_k} = p_k$, and the observations are as in eq (1) above:

$$p_1^{\alpha + Y_{+1} - 1} p_2^{\alpha + Y_{+2} - 1} \cdots (1 - p_1 - \cdots - p_k)^{(n-k)\alpha - 1}$$

Let $cumP_k = \sum_{j=0}^{k} P_j$. Then for the Dirichlet prior, the posterior distribution of $cumP_k$ is simply beta with parameters

$$K - 1, n\alpha + N - K - 1$$

where

$$K = \sum_{j=0}^{k} (\alpha + j) \left| B_k \right|$$

which is just what it would be if we started with a uniform prior for $cumP_k$, executed $n\alpha + N$ binomial trials with success probability $cumP_k$, and observed K successes. Thus we may use the usual binomial formula for the for the variance of the estimate $\hat{cumP}_k$ of $cumP_k$, namely

$$\mathrm{var}(\hat{cumP}_k) = \hat{cumP}_k (1 - \hat{cumP}_k)/N$$

We may expect this result to be a good approximation for both sequential and non-sequential procedures, provided we use a suitable smoothing procedure to interpolate between integer values of k. We choose to estimate P_k by a weighted straight-line smooth of Q_{k+1}, with the weights depending on the variances of the individual Q_k's. Then we sum to get an estimate $\hat{cumP}_k$ and estimate the (real-valued) cutoff k so that $\hat{cumP}_k = 0.99$. If

$k = n + f$ with n an integer, we mark all the leaves in the sets $B_0, \cdots B_n$ and a random fraction f of the leaves in B_{n+1}.

4. Experimental Trials

In these trials (and in [BM94]), the accuracy target $t = 0.99$. We populated trees for each of 186† symbols of a pan-European Latin alphabet, in each of ten typefaces (see [BGI94]). Thus we applied the population rule separately to each of 1860 <typeface,symbol> pairs, letting type size vary during sampling randomly uniformly in the interval [7.5,12.5] point. Each of the resulting trees is specific to a <typeface,symbol> pair. For each typeface, we then merged all their symbols' trees into one by computing the set union of their leaf contents, giving a tree populated for the entire typeface. Each of these typeface-specific trees was then tested using a distinct set of synthetic images from that typeface.

In comparing our results with [BM94], we expect that the error rate of the tree, for each <typeface,symbol> will be $\simeq$ 1.0%, rather than a small fraction of it. Since the minimum number of leaves is chosen, consistent with this target, we expect the pruning factor to be higher. We also hope — but at our current state of understanding we cannot ensure — that the number of samples required will be less.

The results on a typical typeface, Avant Garde-Book Oblique, are as follows. The stopping-rule method of population described in [BM94] required 8,161,999 samples to populate the tree When tested on a distinct set of 1,005,000 synthetic samples (1000 for each <typeface,size,symbol> triple for the five sizes 8-12), the error rate was measured to be 0.15%, about 1/7th of the target. The pruning factor on the same test data is $\times$2.58.

In applying the present leaf-selection method, we chose to generate $N = 10000$ samples for each <typeface,symbol> pair, so 1,860,000 samples were used altogether to populate the tree. When tested on the same 1,005,000 samples used above, the tree's error rate was measured to be 1.05%, close to the target of 1.0%. The tree's effective pruning factor was $\times$3.83.

The superiority of the leaf-selection method over the stopping-rule method is clear. It achieves the user-specified error target more closely, improves the pruning factor of the tree, and, at least in these tests, does not require as many population samples to be generated.

We also tested on ''real'' data. Six hundred pages were printed and then scanned (at 400ppi) containing text in each of ten European languages, twenty typefaces, and three type sizes (8, 10, and 12 point). The ten typeface-specific trees were merged into a single tree. The test exercised not only the preclassifier and classifier, but other stages of the complete page reader including geometric layout analysis, shape-directed resegmentation, and contextual analysis (both typographic and linguistic, specific to each language).

The results, averaged over all 600 pages, were that the page reader as a whole (not merely the classification stage) was sped up by a factor of 2.2. Accuracy was unaffected: actually, an improvement of 0.02% was measured, which is not significant at 95% statistical confidence. The lack of extra error is perhaps surprising in light of the fact that our error target was fully 1.0 per cent. One possible explanation is that our image defect model is pessimistic in the sense that it generates a superset of the defects that occur in our printing/scanning aparatus.

† For these tests, we used a subset of the original 209 symbols for which artwork existed in all of the typefaces. The omitted symbols are rare punctuation which do not occur in the European text used in the tests (described below) on real printed pages.

5. Discussion

We have described a method for automatic construction of preclassification decision trees for OCR that closely achieve a user-specified target for the extra error that they contribute to a multi-stage decision procedure. The essential technical device is a leaf-selection rule for tree-population, based on the Good-Turing Theorem (1953). Compared to a previously described stopping-rule procedure, this method improves (in fact, it maximizes) the pruning factor consistent with the given error target. In large scale trials on actual printed pages, a preclassification tree built in this way sped up the entire page reader by better than a factor of two without contributing any measurable extra error.

These results suggest that our methodology, which is somewhat controversial due to its total reliance on synthetic training data, has passed a significant test of its relevance to practice.

There remain several interesting open questions. The fact that $N = 10000$ samples per <typeface,symbol> pair produced good results is already an improvement over [BM94]: perhaps further improvements are possible. More precisely, what is the minimum number of synthetically generated image samples required for population that will ensure that the variance of the error is bounded to within a user-specified limits? We optimize each <typeface,symbol>-specific tree separately and then merge them: can we do better by attempting to optimize the merged tree? Is there some way to improve on our method for constructing the decision nodes of the tree?

6. Acknowledgement

Stimulating conversations with Tin Ho and David Ittner are much appreciated.

7. References

[BM94] H. S. Baird & C. L. Mallows, ‘‘Bounded-Error Preclassification Trees,’’ *Proc., IAPR 1994 Workshop on Structural and Syntactic Pattern Recognition*, Nahariya, Israel, October 4-6, 1994. (To appear in D. Dori & A. Bruckstein (Eds.), *Shape, Structure and Pattern Recognition*, World Scientific, Singapore, 1995.)

[Bai88] H. S. Baird, ‘‘Feature Identification for Hybrid Structural/Statistical Pattern Classification,’’ *Computer Vision, Graphics, and Image Processing*, Vol. 42, No. 3, June 1988, pp. 318-333.

[Bai92] H. S. Baird, ‘‘Document Image Defect Models,’’ in H. S. Baird, H. Bunke, and K. Yamamoto (Eds.), *Structured Document Image Analysis*, Springer-Verlag: New York, 1992, pp. 546-556.

[Bai93] H. S. Baird, ‘‘Document Image Defect Models and Their Uses,’’ *Proc., IAPR 2nd ICDAR*, Tsukuba, Japan, October 20-22, 1993.

[BGI94] H. S. Baird, D. Gilbert, D. J. Ittner, ‘‘A Family of Eurpoean Page Readers,’’ *Proc., IAPR 12th ICPR*, Jerusalem, Israel, October 9-13, 1994.

[Bun87] W. Buntine, ‘‘Learning Classification Trees,’’ *Statistics and Computing*, vol. 2, 1992, pp. 63-73.

[CN84] R. G. Casey and G. Nagy, ‘‘Decision Tree Design Using a Probabilistic Model,’’ *IEEE Trans. Information Theory*, Vol. IT-30, No. 1, Jan. 1984, pp. 94-99.

[Good53] I. J. Good, ‘‘The Population Frequencies of Species and the Estimation of Population Parameters,’’ Biometrika 40, pp. 237-264, 1953.

[WS87] Q. R. Wang and C. Y. Suen, ‘‘Large Tree Classifier with Heuristic Search and Global Training,’’ *IEEE Trans. PAMI*, **PAMI-9**, No. 1, Jan. 1987, pp. 91-102.