N-Tuple Features for OCR Revisited

Dz-Mou Jung, *Member*, *IEEE Computer Society*, M.S. Krishnamoorthy,
George Nagy, and Andrew Shapira

Abstract—N-tuple features for optical character recognition have received only scattered attention since the 1960s. Our main purpose here is to show that advances in computer technology and computer science compel renewed interest. N-tuple features are useful for printed character classification because they indicate the presence or absence of a given rigid configuration of n black and white pixels in a pattern. Desirable n-tuples fit each pattern of a specified (positive) training set of characters in at least p different shift positions, and fail to fit each pattern of a specified (negative) training set by at least $n - q$ pixels in each shift position. In this work we prove that the problem of finding a distinguishing n-tuple is NP-complete, by examining a natural subproblem with binary strings called the missing configuration problem. The NP-completeness result notwithstanding, distinguishing n-tuples are found automatically in a few seconds on contemporary workstations. We exhibit a practical search algorithm for generating, from a small training set, a collection of n-tuples with low class-conditional correlation and with specified design parameters n, p, and q. The generator, which is available on the Internet, is empirically shown to be effective through a comparison with a benchmark generator. We show experimentally that the design parameters provide a useful tradeoff between distinguishing power and generation time, and also between the conditional probabilities for the positive and negative classes. We explore the feature probabilities obtainable for various dichotomies, and show that the design parameters control the feature probabilities.

Index Terms—Backtracking, character features, classification, decision trees, distinguishing string, missing configuration, n-tuples, OCR, simulated parallelism.

◆

1 INTRODUCTION

> We have an interest in decision-making circuits with the following qualities: 1) measurable high reliability in decision making, 2) either a high or a low reliability input, and 3) possibly low reliability components [2].

WITH these words in 1959, Bledsoe and Browning introduced n-tuples ("decision-making circuits") for recognizing typewritten characters, hand-blocked print, and handwritten script.

For our purposes, an n-tuple is simply a collection of n pixels with distinct locations. Fig. 1 shows a 7-tuple T_1 comprising four black pixels and three white pixels. The tuple "fits" the *c* exemplar and the *e* exemplar, and does not fit the *n* exemplar or the *r* exemplar. That is, if we copy T_1 onto a transparency and superimpose the transparency onto the *c*, it is possible to shift the transparency so that each pixel of the 7-tuple is the same color as the corresponding pixel of the *c*. Here each character is considered to be embedded in a sea of white pixels, and the tuple is moved only by shifting, and not, for example, by rotating or reflecting. In this way, we find that T_1 fits the *c* and *e*, but does not fit the *n* or *r*. We say that the 7-tuple T_1 is designed for the $ce - nr$ dichotomy.

In the early sixties, researchers at IBM's T.J. Watson Research Center conducted a massive series of experiments, combining n-tuple features with various types of statistical classifiers for high-performance typewritten and hand-printed character recognition [1], [6], [7], [9]. Simple hardware can determine whether or not a given n-tuple fits a given character, so n-tuples were sometimes called *decision-making circuits*. N-tuples were mentioned only briefly in Levine's 1969 survey of feature extraction [8], but Nadler praised them in his 1972 State of the Art in Optical Character Recognition [10]. The sensitivity of n-tuples to sample size was investigated at the UK National Physical Laboratories [20], and Stentiford exploited n-tuple independence for reading printed postal addresses in 1985 [19]. Since then, the quest for magic universal features has meandered in other directions. Our main purpose here is to show that advances in computer science and computer technology compel serious reexamination of the applicability of n-tuples to OCR.

In this reprise of an earlier strand of research, we present a generator that finds a collection of distinct n-tuples for any specified dichotomy of binary character patterns. The generator is intended for the automatic construction of OCR systems where little or no prior information is available about the nature of the symbol shapes. We prove that the problem of finding a distinguishing tuple is NP-complete, by showing that a natural subproblem with binary strings called the *missing configuration problem* is NP-complete. The NP-completeness of tuple generation suggests that in general, generating tuples may be difficult. Despite this, for practical problems the generator finds distinguishing n-tuples in a few seconds on contemporary workstations. The quality of the generator is established by looking at absolute execution times, and by comparing the generator to a simple benchmark generator. We give a set of design parameters that provide control of the class-

---

- *D.-M. Jung is with Caere Corporation, Los Gatos, CA.*
- *M.S. Krishnamoorthy, G. Nagy, and A. Shapira are with Rensselaer Polytechnic Institute, Troy, NY 12180. E-mail: nagy@ecse.rpi.edu.*

Manuscript received June 26, 1995; revised Apr. 15, 1996. Recommended for acceptance by J.J. Hull.
For information on obtaining reprints of this article, please send e-mail to: transpami@computer.org, and reference IEEECS Log Number P96043.

conditional feature probabilities associated with a given dichotomy and n-tuple. We also present statistical evidence that for a given dichotomy, tuples can be generated that are not highly correlated and can therefore be used with simple classification methods.

The n-tuple generators in this paper are part of a C language software library for generating n-tuples. This library is available on the Internet [16].

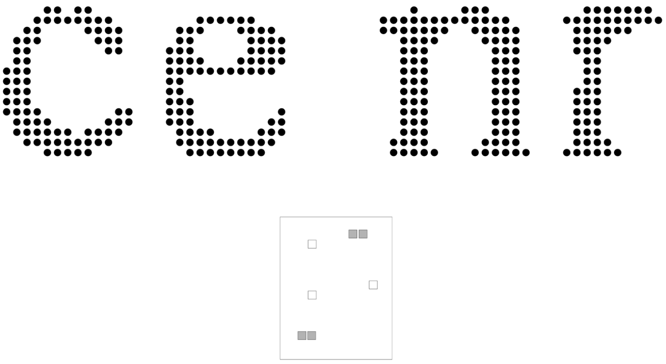


Fig. 1. A 7-tuple T_1 that fits the *c* and *e* and does not fit the *n* or *r*.

Returning to Fig. 1, if the exemplars (the training samples) are representative of other instances of *c*, *e*, *n*, and *r*, then we can expect the tuple T_1 to behave similarly on new exemplars (the test samples). We say that T_1 is designed to fit the positive class C^+ of *c* and *e*, and to misfit the negative class C^- of *n* and *r*. Such a tuple is an elementary two-category classifier. Several tuples can be used in combination to build multicategory classifiers, or to improve the accuracy of two-category classifiers. Minimum-distance, weighted linear network, quadratic, nearest neighbors, and decision tree classifiers have all been used successfully in conjunction with n-tuples.

A tuple is associated with the presence or absence of a specific configuration of black and white pixels in a given pattern. At one extreme, a 2-tuple detects a pair of pixels of given color and relative displacement. At the other extreme, a tuple with as many elements as the entire pattern array constitutes a matched filter (or mask, or template) for a specific binary pattern. For OCR, tuples are sought that match most instances of several designated alphanumeric pattern classes, and do not match most instances of other designated classes.

Given a set of character classes, it is easier to find n-tuple features for some dichotomies than for others. For instance, the *ceo-rnm* dichotomy is easier than the *rnc-eom* dichotomy. The formulation of suitable dichotomies, which is a difficult problem in itself, is addressed in [21].

We are interested in n-tuples because they are uniquely suited for the automated design of single-font OCR systems when only a small number of labeled samples are available. Limited sample availability is characteristic of in-the-field adaptation. We consider field adaptation desirable, even to only a single document. For a given document, an adaptive system using features that discriminate between the classes of the document's actual typeface can perform at least as well as a static multifont system. Possibly, the adaptive system can do better.

Accordingly, throughout this paper we make the following assumptions:

- only one font is to be recognized,
- only a small number of labeled samples are available, and
- the dichotomy is given.

1.1 Classification Using N-Tuples

Regardless of the particular type of classifier used, the error rate is closely linked to the class-conditional feature vector probabilities [3]. These probabilities are

$$P(\mathbf{x} \mid C^+) = P(x_1, x_2, \ldots, x_r \mid C^+) \text{ and } P(\mathbf{x} \mid C^-) = P(x_1, x_2, \ldots, x_r \mid C^-).$$

Here r is the length of the feature vector $\mathbf{x}$, and each element x_i is an indicator random variable that is true iff feature i is present in the character under consideration. If x_i is true we can write the logical statement x_i = true, or simply x_i; if x_i is false we write $\overline{x_i}$. In our case, i identifies a tuple we are interested in, and x_i indicates whether the tuple fits the character under consideration.

For accurate classification, we want to maximize the marginal probabilities $P(x_i \mid C^+)$, and minimize the marginal probabilities $P(x_i \mid C^-)$. That is, we would like

$$P(x_i \mid C^+) \approx 1 \text{ and } P(x_i \mid C^-) \approx 0, \text{ for all } x_i.$$

For accurate classification and ease of classifier design, we want the individual features x_i to be class-conditionally independent; we would like the following to hold for all feature vectors $\mathbf{x}$:

$$P(\mathbf{x} \mid C^+) = P(x_1 \mid C^+)P(x_2 \mid C^+) \cdots P(x_r \mid C^+), \text{ and} \qquad (1)$$

$$P(\mathbf{x} \mid C^-) = P(x_1 \mid C^-)P(x_2 \mid C^-) \cdots P(x_r \mid C^-). \qquad (2)$$

Given a small training sample of representative patterns, we wish to generate tuples that have the above properties. Our scheme is based on the following propositions.

P1 Small values of n are most desirable for generating tuples that resist variations in positive test patterns.

P2 Large values of n are most desirable for generating tuples that resist variations in negative test patterns.

P3 The probability of a generated tuple fitting a positive test pattern increases with p, if we generate tuples that fit each positive training exemplar in p or more shift positions.

P4 The probability of a generated tuple fitting a negative test pattern decreases with q, if we generate tuples that agree with each negative training exemplar at q or fewer pixels in each shift position.

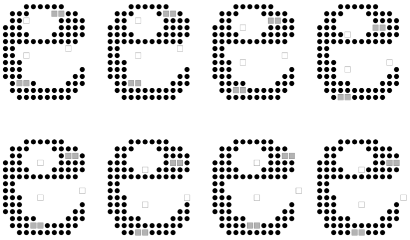
P5 Tuples with appropriate values of (n, p, q) can be found without using excessive computing resources.

The validity of these propositions is demonstrated in subsequent sections.

The first two propositions imply a trade-off between the two conditional probabilities that depends on n. Our experiments support earlier findings [6] that $n = 7 \pm 2$ is usually acceptable. In view of the third and fourth propositions, for a specified value of n, for each q we select the largest p for which tuples can be found. Different (p, q) combinations bias reliability towards either the positive or negative class. For each dichotomy, we can generate tuples according to (n, p, q) values appropriate for the dichotomy's class probabilities.

1.2 Outline

An outline of the remainder of the paper is as follows. In Section 2, we discuss the p and q parameters and the partial order that these parameters impose on the set of all tuples. Section 3 shows that the tuple generation problem is NP-complete. Section 4 describes the tuple generator and explores the generator's speed and the relation between speed and the design parameters. In Section 5, we investigate the optimal tuple size; the dependence of the feature probabilities on the design parameters; the statistical correlation of the generated tuples; and the effect of the choice of dichotomy on the feature probabilities. We conclude with a discussion of the implications of our improved method of tuple generation for printed character classification.



2 *p*-*q* STABILITY

Before discussing the p and q parameters, let us solidify a few concepts.

Earlier, we stated that an n-tuple is a set of n pixels at distinct locations. Here we define a *pixel* to be an integer triple (r, c, v), where the row r and column c give the pixel's location, and $v \in \{0, 1\}$ gives the pixel's color (0 is white and 1 is black). The *order* of a tuple T is the number of pixels $|T|$.

A *character* E with $r_{\max}$ rows and $c_{\max}$ columns is an array $A[1 \ldots r_{\max}, 1 \ldots c_{\max}]$ with elements having value 0 or 1. We embed E in a sea of white pixels by defining an auxiliary array $B[-\infty \ldots +\infty, -\infty \ldots +\infty]$, so that $B[i, j] = A[i, j]$ if $1 \le i \le r_{\max}$ and $1 \le j \le c_{\max}$, and $B[i, j] = 0$ otherwise. A tuple T *fits* E (exactly) if there exists some *shift* $(\Delta r, \Delta c)$ such that for each pixel $(r, c, v) \in T$, $B[r + \Delta r, c + \Delta c] = v$.

For the remainder of this section let us fix a set E^+ of positive class exemplars, a set E^- of negative class exemplars, and a tuple size n.

Given a tuple T, we define p_T to be the maximum value p such that for each character $E \in E^+$, there are at least p shifts that give an exact fit between T and E. Similarly, we define q_T as the minimum value q such that for each character $E \in E^-$, for no shift do T and E agree in q or more pixels. A tuple T is said to be *p-q-stable*, or to *satisfy the (p, q) constraint*, if $p_T \ge p$ and $q_T \le q$. We restrict our attention to tuples T with both $p_T \ge 1$ and $q_T \le n - 1$.

The idea behind *p*-*q*-stability is that the larger p_T is, the more T resists noise in positive test samples; the smaller q_T is, the more T resists noise in negative test samples. Fig. 2 shows the characters c and e and the 7-tuple T_1 from Fig. 1. For each of the two characters, T_1 is shown superimposed in all positions for which a match exists. This shows that $p_{T_1} = 7$. Illustrating q_{T_1} is more difficult. In this example, $q_{T_1} = 5$, i.e., when T_1 is superimposed in each shift position on either of the negative characters n or r, there are always at least $7 - 5 = 2$ pixels of T_1 that mismatch the character.