Document Image Analysis

Lecture 8: Scaling Up Classifiers

Henry S. Baird
Richard J. Fateman

Xerox Palo Alto Research Center
University of California – Berkeley

The course so far….

- DIA overview, objectives, measuring success
- Isolated-symbol recognition:
  - symbols & glyphs
  - models, features, classifiers
  - image metrics for classification
  - ***this lecture***: ***scaling up, e.g. to 100 fonts of full ASCII***
  - *next lecture: 'best' classifier?, combination of classifiers*
- *Soon: word recognition, character segmentation, context, …*

Scaling up: or, how to build a full-ASCII, 100-typeface classifier (+ Tibetan)

- We're still focused on *isolated* symbols
- What strategies are good for extending classifiers to:
  - many symbols
  - many typefaces
  - many sizes
  - varied image degradations
- Anatomy of a particular system:
  - how to use 'structural' features in a statistical setting
  - how to train (semi-)automatically
  - how to improve systematically
- Versatility: applications to non-ASCII symbol sets

Polyfont Classification

Image recognition algorithm:

printed & typewritten text
>100 fonts of ASCII symbols

Performance analysis:

accuracy
uniformity: symbols × fonts × sizes × defects

Assess "generalization"

accuracy on fonts *not* trained on
rich variety of font-symbol shapes
large-scale experimental data

Next lecture….

- Is there a dominant classifier?
  - how costly is it to build the best?
- How do classifiers compare?
- Combining classifiers:
  - voting methods
- When are combinations better?
- Randomized classifiers

The Fonts of Greatest Interest

"Fonts" are distinguished by shape:

roman and *italic* are different
size variations don't count
moderate width and weight variations don't count:

ne ne ne ne
ne ne ne ne
ne ne ne ne
ne ne ne

Body text fonts

not display/decorative/script/swash

Widely used for English prose

not classified ads

100 fonts = 50 "font-families"

Most Commonly Occurring Fonts

For maximum "coverage" of:

20th C. American
books, magazines, newspapers, letters

Scholars:

Alexander Lawson, *The Anatomy of a Typeface*

"The growth of typefaces has overwhelmed attempts to track type usage."

Manufacturers / Designers:

Linotype – Bruce Leahnert
Adobe – Ned Bunnell
IBM – John Schappler (Courier, *etc.*)
Chuck Bigelow (Lucida, *etc.*)

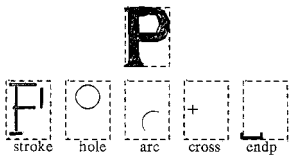
Industry watch:

American Institute of Graphics Arts
Annual 50 Best Books list (1925-1980)
Type World Newsletter – Frank Romano

Consultants:

Tytell Typewriter Co. & Questioned Document Lab. (NYC)

'Structural' Shape Analysis



Shape types —
incomparable
redundant
elementary

Hypothesis: simple 'shapes' may
-- capture human intuition
-- generalize across font styles

'Structural' analysis was popular in late 70's through early 90s

However, not clear how to learn them (don't want to hand-code)

Competing statistical methods seemed to provide a better framework for coping with noise or all sorts

Structural & Statistical

Structural shape analysis

yeilds: set of component shapes

"Feature identification" —

find a **mapping from** sets of shapes **to** bit-vectors

Statistical classification

accepts: bit vector

Combining structural features with statistical classification

Structural shape analysis:

- represent by list of simpler shapes
- "intuitively" good features
- information-preserving
- unordered, variable-length list

"Feature identification"

- mapping:
  - from list of shapes
  - to bit-vector

Statistical classification

- represent by list of numbers (bits)
- requires a few, independent features
- ordered, fixed-length list

← This is the glue between the two approaches

Choosing a parameterization of the shapes

Examples (strokes):

strokes <x,y> <r,i>

Visually similar shapes lie close together.

strokes <x,y> <r,i>

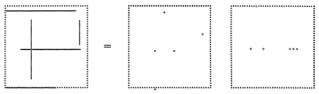
Dissimilar shapes lie far apart.

Crucial choice (manual)

Exploit the fact that, for all shape types:

- can find a low-dimensional parameterization with certain "continuity" properties under a suitable metric

Clustering Shapes in their parameter spaces



strokes <x,y> <r,i>

Parameterization of strokes of a 'P'.

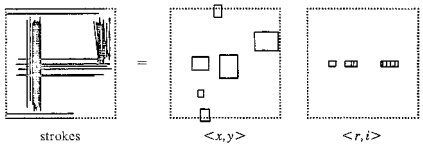
strokes <x,y> <r,i>

Superimposed 'P' strokes, showing clusters.

Note how tightly strokes cluster in this space

The variation of strokes due to noise of all kinds is easier to analyze

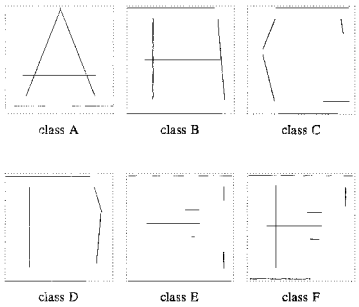
Idealizing the shape clusters



Clusters approximated by L_∞-spheres

(Comparable results with Euclidean norm)

Idealized shape clusters seem to capture some intuition

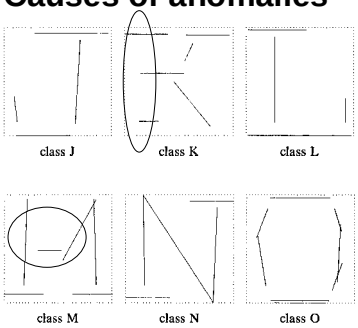


Most classes are as familiar as the 'A' shown here

Roughly correspond to intuition, but there are irritating anomalies

Similar results for the other shape types: holes, endpoints, etc

Causes of anomalies



Some shapes, such as vertical stroke in 'K', are difficult to extract reliably without 'higher-level' models

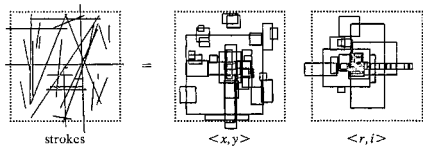
So, even intuitively 'low-level primitives' challenge the earliest stages of feature extraction

As image degradations worsen, these instabilities become pervasive

Merging clusters

There are lots of similar clusters among the classes
Reduce their number by a greedy merging heuristic:

- merge the next pair of clusters whose merged volume increases the least,
- until have a target number of clusters.



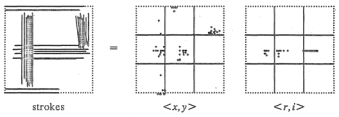
Stroke clusters after merging (1/3 of alphabet).

Each final shape-cluster gives a 'feature' for classification

- Assign a binary feature to each shape-cluster
- Given an unknown isolated symbol image,
  - perform shape analysis into strokes, holes, etc
  - if any of these 'matches' (falls inside) a shape-cluster (in its parameter space), then set that shape-cluster's feature bit to 1
  - if no shape matches a shape-cluster, its feature bit remains 0
- So, have a bit-vector each bit of which indicates the presence (one or more times) of a shape-cluster
- Finally, classify this vector
  - using, e.g., the linear classifier assuming class-conditional independence among the features

Some simpler feature mappings

k-partitioning (for comparison)



A 3-partition (giving 81 disjoint regions).

Experiments with 3-, 4-, and 5-partitioning:

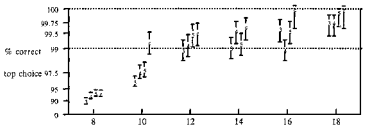
3 288
(c 300)
4 864
5 2050

Cheaper to compute than the shape-clustering feature mapping

But, gives many more features

Which is better? Must run an experiment

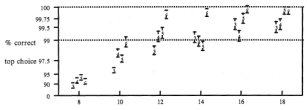
Small-scale experiments: single-font



Better than 99.5% top choice, above 12 point.

Top choice performance of classifiers trained on the Times Roman font and mixed sizes (8-18 point), and tested on the same font at the specific sizes shown (2100 samples per datum).

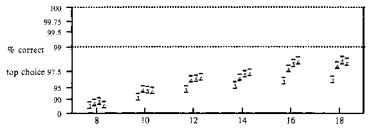
Small-scale experiments: mixtures of similar fonts



Clustering superior to *k*-partitions for *similar* fonts.

Top choice performance of classifiers trained on a mixture of three similar but not identical fonts (Times Roman, Palatino, and Malibu Roman), and on mixed sizes (8-18 point), then tested on the same font mixture at the specific sizes shown (for 3-, 4-, and 5-partitioning and clustering feature-identification methods; 6300 samples per datum).

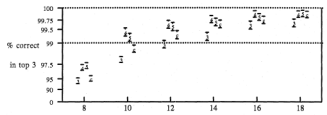
Small-scale experiments: mixtures of dissimilar fonts



Top choice bad for mixtures of dissimilar fonts.

Top choice performance of classifiers trained on a mixture of six dissimilar fonts (Times Roman, *Times Italic*, **Times Bold**, Helvetica Light, *Helvetica Italic*, and **Helvetica Black**), and on mixed sizes (8-18 point), then tested on the same font-mixture at the specific sizes shown (12,600 samples per datum).

Small-scale experiments: top-3 correct



Top 3 performance is still good.

Performance (correct in top 3 choices) of classifiers trained on a mixture of six dissimilar fonts (Times Roman, *Times Italic*, **Times Bold**, Helvetica Light, *Helvetica Italic*, and **Helvetica Black**), and mixed sizes (8-18 point), then tested on the same font-mixture at the specific sizes shown (12,600 samples per datum).

How to improve classifier?

'split' classes into sets of similar shapes

a.1 *a a a a a a etc.*
a.2 *a a a a a a etc.*
a.3 *a a a a a a etc.*
a.4 *a a*
a.5 *a*
a.6 *a*

"Variant" = Bayesian "class" = statistical prototype

Splitting classes

more variants:

larger, slower classifier
more accurate on given fonts
less accurate on other fonts

We manually chose 612 variants:

= 1 variant (no splitting)
J 18 variants
6.5/symbol on average

Best & Worst….

Fonts:

Best:

Garamond Roman: Pack my box with five dozen liquor jugs.
Textype Italic: Pack my box with five dozen liquor jugs.
Plantin Light Italic: Pack my box with five dozen liquor jugs.
Aster: Pack my box with five dozen liquor jugs.

Average:

Typewriter Gothic: Pack my box with five dozen
Bell Italic: Pack my box with five dozen liquor jugs.
Serifa: Pack my box with five dozen liquor jugs.
Caslon Old Face: Pack my box with five dozen liquor jugs.

Worst:

Bengulat Book: Pack my box with five dozen liquor jugs.
Gill Sans Italic: Pack my box with five dozen liquor jugs.
Weiss: Pack my box with five dozen liquor jugs.
Avant Garde Italic: Pack my box with five dozen liquor ju

Characters:

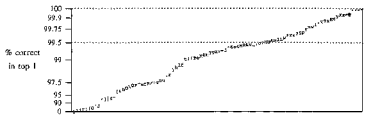
Best:

+ : y > L X w s d A …

Worst:

… (} ' ^ I / 1 | 1 0

Accuracy by symbol

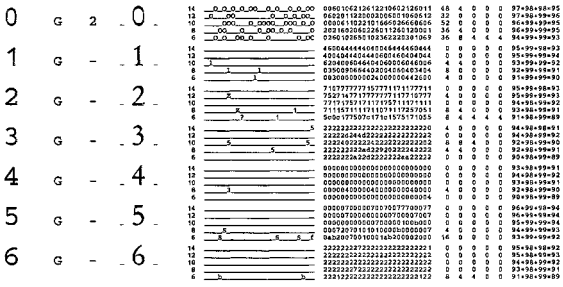


Top-1 accuracy of each of the 94 symbols, averaged over all 100 fonts and over the sizes 10p, 12p, and 14p (at 300 ppi), assuming that all fonts and symbols are equiprobable. The 94 symbols are plotted in ascending order of accuracy. (About 7000 samples per datum; fewer for symbols not present in all fonts.)

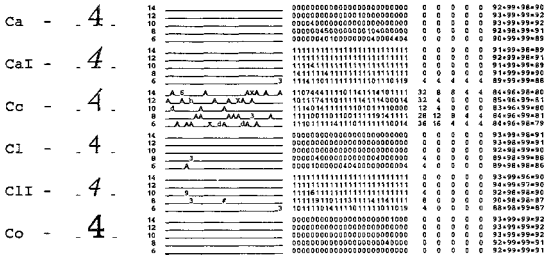
75% of top-1 confusions:

0O 1lIJ]|}{ GC S5 `'

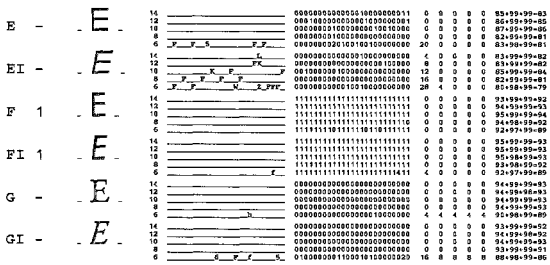
Accuracy by font-symbol…



Accuracy by font-symbol



Accuracy by font-symbol



Space requirements

Space required to represent all:

fonts × symbols × sizes × defects

Space ∝ no. shape variants

Each shape variant represents:

a set of font-symbols, *e.g.* {a, a, a, a, a, a}
× all sizes × all defects

'Compression'

Compression = no. font-symbols / no. variants

9400 / 612 = 15.4

~ 15 font styles, on average, are
represented by *one* statistical prototype,
in spite of size and defect variations

Implies that larger classifiers can be built
with only a *fractional* increase
in computing resources

Generalization

Extrapolation to fonts *not* trained on

We trained on *only* 2416 of the 9400 font-symbols,
and still did well on *all* of them

"Generalization factor" ≈ 3.9

This has held roughly constant:

25-, 40-, 60-, 80-, 100-font trials

Implies that a 100-font classifier will
do well on many more than 100 fonts

Feature identification

Choosing a mapping —

- automatic inference from training data
- uniformly applicable to all shape types
- "simple"
  - controlled by a few, natural parameters
  - exact values of parameters not critical

Applying a mapping —

- usably fast
- exploit multi-dimensional search algorithms

Effect on classification —

- superior to simpler fixed mappings
- demonstrated in large-scale experimental trials

Summary of characteristics

Simple, fully automated method —

- a minimum of hand-crafted rules
- applied uniformly to all shape-types
- guided by a few parameters
  - "natural" statistical trim-factors
  - exact values not critical

Tested in large-scale trials —

- results hold with 95% statistical confidence
- showed improvement above 99.5% (hard)

Fairly general technique —

- requires that shape types be elementary
- "continuous" parameterization
- not strongly specialized to OCR

Training on size ranges