Perfect Metrics

Tin Kam Ho, Henry S. Baird

AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974, USA

Abstract

We describe an experiment in the construction of "perfect metrics" for minimum-distance classification of character images. A perfect metric is one that, with high probability, is zero for correct classifications and non-zero for incorrect classifications. These promise excellent reject behavior in addition to good rank ordering. Our approach is to infer from the training data faithful but concise representations of the empirical class-conditional distributions. In doing this, we have abandoned many usual simplifying assumptions about the distributions: e.g. *that they are simply-connected, unimodal, convex, or parametric* (e.g. *Gaussian). Our method requires unusually large and representative training sets, which we provide through pseudo-random generation of training samples using a realistic model of printing and imaging distortions. We illustrate the method on a challenging recognition problem: 3755 character classes of machine-print Chinese, in four typefaces, over a range of text sizes: in a test on over three million images, the perfect-metric classifier achieved better than 99% top-choice accuracy. In addition, we show that it is superior to a conventional parametric classifier.*

1 Introduction

A critical problem in the design of classifiers is constructing faithful but concise descriptions of the class-conditional distributions of the training samples. Such distributions in high-dimensional vector spaces are in general non-simply connected, multi-modal, and non-convex. Thus the abstractions provided by simple parametric models are often unjustified. The problem is exacerbated by the fact that training data collected in an *ad hoc* manner is often uselessly sparse.

The complex interplay among the representativeness of samples, the discriminating power of features, and mutual dependencies among features, is analyzed in detail in a theory of asymptotically perfect classifiers recently proposed by Kleinberg [5] [6]. He suggested that nearly perfect accuracy is achievable, provided that one is given training data which is, in a certain precise sense, sufficiently representative relative to the expressive power of the features.

Motivated by this analysis, we attempt to study these issues by exerting control over the statistical properties of the training data. In a machine-print character-recognition setting, this can be done by the use of a pseudo-random image generator based on a realistic model of document image distortions.

An outline of the body of this paper is as follows:

1. Select features, and generate training data, so that certain engineering requirements are met.
2. Infer detailed but space-efficient descriptions of the empirical class-conditional distributions.
3. Construct a set of metrics, using these descriptions, that permits a nearly perfect match (distance of nearly zero) between a sample and the ideal prototype for its class.
4. Show that a classifier using these metrics achieves high accuracy, and furthermore that it improves on a conventional parametric — but otherwise comparable — classifier.

2 Design of the Experiment

We will build a classifier for the four most commonly used fonts in printed Chinese: Song, Fang Song, Hei, and Kai. The text size will range from 7 point to 14 point, at a spatial sampling rate of 400 pixels/inch. The experiments embrace all 3755 character classes of the GuoBiao Coding GB2312-80, Level 1 [2].

The Features

Since our principal interest here is how to make the best use of given features — not how to select the best features — we have chosen, somewhat arbitrarily, some features commonly used in published Chinese recognition systems [7]. The binary image

of an input character is first size-normalized to 48x48 binary-valued pixel matrix by simple scaling and centering. That is, each image is mapped to a point in a binary-valued vector space of $48 \times 48 = 2304$ dimensions, containing at most $2^{2304} \approx 10^{694}$ distinct points.

For reasons that will become clear later, we choose not to use this binary-valued space directly. Instead, we will use three integer-valued feature sets: vertical and horizontal projection profiles, distances from outer contours to the bounding box, and distributions of stroke directions.

The projection features are computed as follows. The image area is divided into upper and lower halves, and a vertical projection profile (counting the number of black pixels in each column), is computed for each. Similarly, two horizontal projection profiles are obtained for the left and right halves. These four profiles are then concatenated to form a vector with 48x4=192 dimensions; each projection feature's integer value lies in the range [0,24].

The contour features are distances from each of the four edges of the bounding box to the character's outer contour. For each column, we calculate the distance from the upper edge of the box to the first black pixel of the column, and from the lower edge to the last black pixel. Similarly for each row, we calculate the distance from the left edge to the leftmost black pixel, and from the right edge to the rightmost black pixel. These distances form a vector of $48 \times 4 = 192$ dimensions; each contour feature's integer value lies in the range [0,48].

The stroke-direction features are computed by run-length analysis as follows. From each black pixel, we compute the length of the black runs containing that pixel as they are extended in four directions (horizontal, NE-SW diagonal, vertical, and NW-SE diagonal). The pixel is then labeled with the direction in which the run length is the maximum. Then we partition the image area into 16 (12×12) square regions and count the number of pixels of each of the four types in each region. These counts are stored in a vector of 16x4=64 dimensions; each stroke direction feature's integer value lies in the range [0,144].

Each character image is thus mapped to a point in an integer-valued vector space of $192 + 192 + 64 = 448$ dimensions, containing at most $25^{192} \times 49^{192} \times 145^{64} \approx 10^{731}$ distinct points.

Now of course the actual distributions of images (points) of different character classes and fonts in either of these vector spaces (binary-valued or integer-valued) can be expected to be sparse compared to the total volume of the space, and complex in detail. An important aspect of our approach is that we will avoid making many of the usual simplifying assumptions about these distributions: in particular, we will not assume that they are simply-connected, unimodal, convex, or approximately parametric (*e.g.* multi-dimensional Gaussian). However, we will tentatively assume that the distributions are "locally dense" in suitable subspaces. Intuitively, we mean that images belonging to the same class, when observed in the right subspace, will cluster together in (possibly more than one) local region. Choosing such subspaces and exploiting the resulting distributions for classification is the strategic goal towards which this paper is a first tactical step.

We proceed by treating each of our integer-valued features as such a subspace, of dimension one: that is, we project the whole space onto each dimension, and examine the distributions there. In order that clustering will be easily detected, the cardinality of values in each subspace should be neither too small nor too large.

If the cardinality of a feature's values is less than $1/n$ the number of training samples for a class, then some value must occur more than n times, and further we can expect that clusters, if any, will form densely and be easy to extract automatically. In fact, we hope for more: that every feature value possible for a class will occur at least once in the training set.

A cardinality of two — as occurs in each component of the binary-valued vector space — is often too small to reveal details of distributions. On the other hand, if the cardinality of the subspaces is too large, there will be too many variations in each subspace to be represented by a training set of reasonable size.

With this in mind, we compress the integer-valued ranges of both the contour and the stroke-direction features to lie in [0,24], matching the range of the projection features. We will generate training sets with 800 samples per class, so that for each feature, we have 32 times as many samples as we have feature values. We describe the creation of such a data set in the next section.

Generation of the Distorted Samples

We use an explicit, quantitative, parameterized model of defects due to printing, optics, and digitization, and a pseudo-random image generator that implements the model. The model is based on approximations to the physics of the printing and imaging process ([4] & [8]). A detailed description is in [1]; here, we give a brief overview. The input to the generator is an "ideal" back and white image at high

resolution: in practice, we use bitmaps or scalable outline descriptions purchased from typeface manufacturers. The model parameters specify the nominal text size of the output (in units of points), the output spatial sampling rate (digitizing resolution in pixels/inch), the point spread function (the standard error of its Gaussian blurring kernel in units of output pixels), the digitizing threshold (in units of intensity, where 0.0 represents white and 1.0 black), the distribution of sensitivity among the pixel sensors (a noise term added to the threshold), the distribution of jitter among the pixels (*i.e.* discrepancies of the sensor centers from an ideal square grid, in units of output pixel), rotation (skew angle), stretching factors (both horizontally and vertically), and translation offsets with respect to the pixel grid.

Nominal text sizes of the training set data are 7, 9, 11, and 13 point, and for the test set 8, 10, 12, and 14 point. The pseudo-random generator accepts a specification of a distribution on these parameters; each parameter is randomized independently. The distributions used in these experiments are as follows. The digitizing resolution was fixed at 400 pixels/inch. The standard error of the Gaussian blurring kernel varied, from image to image, normally with mean 0.7 and standard error 0.3 (output pixels). The binarization threshold varied, from image to image, normally with mean 0.25 and standard error 0.04 (intensity). Pixel sensor sensitivity varied, from pixel to pixel, normally with mean 0.125 and standard error 0.04 (intensity). Jitter varied, from pixel to pixel, normally with mean 0.2 and standard error 0.1 (output pixels). Skew varied, from image to image, normally with mean 0 and standard error 0.7 degrees. The multiplicative factor affecting width varied uniformly in the interval [0.85,1.15], and the multiplicative factor affecting height varied normally with mean 0 and standard error 0.02. Translation offsets were chosen uniformly in [0,1], in units of output pixel.

Fifty samples were generated for each font/size/symbol triple, for a total training/testing set of 200 for each font/symbol pair and so 800 total for each symbol. To help the reader appreciate the kind of defects generated, Figure 1 shows a selection.

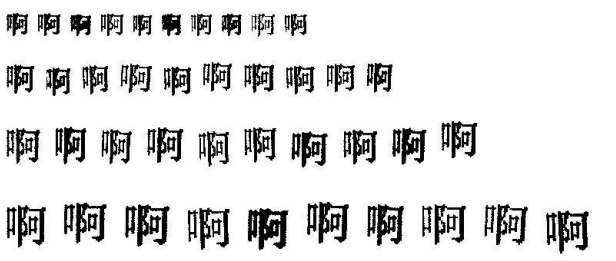


Figure 1: Examples of distorted images generated by the defect model.

Construction of Metrics and the Classifier

The feature extractors were applied to each training sample. We can consider the result as either an integer-valued vector of 448 dimensions, or, equivalently, as a binary-valued vector of $448 \times 25 = 11200$ dimensions, called a *distribution map*. In a distribution map for a single sample, each feature is represented by 25 bits, and for a single sample a single bit is set to 1 indicating the value of the feature. Such a distribution map is illustrated in Figure 2.

For each class, the distribution maps for 800 training samples are combined into one map by computing their Boolean union. In such a *class distribution map*, each feature value that occurs at least once in the training set is represented by a bit set to 1, and 0 bits represent feature values that never occur. We choose to combine all four fonts' training data into a single set of class distribution maps. The distribution maps of the first 3 classes used in our experiments are shown in Figure 3. The classifier is completely described by the set of all 3755 distribution maps, for a total of $3755 \times 11200 \approx 42.1$M bits, or 5.26M bytes, of storage.

Note that this representation can express multimodal distributions, and is not limited to parametric approximations.

During testing, for each character image, we extract its features and, for each class, match the features to the class distribution map, by computing a 448-bit vector in which each feature's bit is set to 1 if and only if the feature's value occurs in the class distribution map. Finally, the "distance" to that class is simply Hamming distance of this vector to an ideal vector containing all '1's. It is important to note that each class has its own peculiar metric. By contrast with some other minimum distance classifiers, there is no single metric that applies to all classes.

If all feature values occur for a class, then the resulting bit-vector is all 1's, and Hamming distance is 0. Thus, by construction, all samples in the training set have distance 0 to their correct class. And if, for each pair of classes, there is at least one feature for which the distribution maps have no '1's in common, then all samples in the training set will have non-zero distance to all but their correct class. Under these cir-

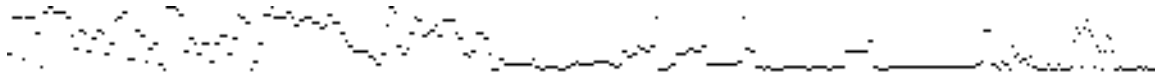


Figure 2: The feature values for one sample image, shown as a distribution map. The horizontal coordinate ranges from 0 to 447, representing each of the 448 features. The vertical coordinate ranges from 0 to 24, representing the value of each feature.



Figure 3: Comparison of distribution maps for 3 classes (GB1601 – GB1603). Note the frequent occurrence of multi-modal distributions.

cumstances, we say that the metrics are "perfect" on the training set. So, perfection in this sense depends in part on the discriminating power of the features, which is a topic outside the scope of this paper.

On the test set, we cannot expect the metrics to be perfect, but, as we shall see, they are remarkably close to perfect. Naturally, for classification we do not rely on zero distance, but rank the classes in ascending order on distance.

In addition to a large-scale trial of this perfect-metric classifier, we will compare it to a conventional parametric classifier built with the same training data and under similar assumptions and computational constraints. For this we choose minimum Mahalanobis distance. During training we estimate, for each font/class/feature triple, the mean and standard error of the feature values. Note that by simply adding the contribution of each feature during the computation of Hamming distance, we have made an implicit assumption of independence among the features. The independence assumption, applied to the Mahalanobis classifier, implies that the off-diagonal terms in the cross-correlation matrix are 0. The storage requirements per class are $4 \times 448 \times 2 = 3584$ numbers, which — assuming parsimoniously one byte per number — is still 2.5 times greater than for the perfect-metric classifier. Also, we estimate that the computation time required by the perfect-metric classifier is no greater than for the Mahalanobis classifier. For n classes and m features, the perfect-metric classifier requires nm lookups in distribution maps, nm additions, and n comparisons, for a total of about $4nm+n$ arithmetic operations; by contrast, the Mahalanobis classifier evaluates $4nm$ expressions of the form $((x - mean)/serr)^2$ (3 arithmetic operations each), plus $4nm$ additions and $4n$ comparisons, for a total of $16nm + 4n$ arithmetic operations.

In summary, it is safe to say that the perfect-metric classifier does not consume greater CPU time and space resources than the Mahalanobis classifier; arguably, the opposite is true. Also note that we do not combine the statistics of the four fonts. This might be expected to give the Mahalanobis classifier an advantage over the perfect-metric classifier, for which the data for all four fonts were combined into a single distribution map.

3 The Experimental Results

Comparison with a Mahalanobis-distance classifier

Our first experiment was to compare the performance of the perfect-metric classifier with that of a Mahalanobis-distance classifier. Because of limitations in computing resources — a Mahalanobis classifier for all 3755 classes would require over 50M bytes of main memory in our naive implementation — we compared them using the first 300 classes (GB1601 – GB1918). Both classifiers were trained using 800 training samples and tested with 800 distinct samples for each class. A total of 240,000 (800×300) testing samples were classified. Table 1 summarizes the classification results.

Table 1: Comparison of results for 300 classes.

		perfect metrics		Mahalanobis dist.	
point size	no. of samples	no. of errors	% correct	no. of errors	% correct
8	60,000	239	99.60	399	99.33
10	60,000	46	99.92	134	99.78
12	60,000	31	99.95	187	99.69
14	60,000	33	99.94	259	99.57
font					
Song	60,000	102	99.83	193	99.68
FangSong	60,000	49	99.92	263	99.56
Hei	60,000	151	99.75	189	99.69
Kai	60,000	47	99.92	334	99.44
total	240,000	349	99.85	979	99.59

Table 1 shows, as expected, that the classification results are better for images printed in larger point sizes, and worse for smaller point sizes. Comparing the figures in Table 1, we can see that perfect metrics yield substantially fewer errors than those given by Mahalanobis distance: in fact, the error rate is lower by a factor of 2.4, at 95% statistical confidence.

The advantage of the perfect metrics is even more impressive when we compare the number of errors in neighborhoods of the top choice. For the perfect metrics, the number of errors drops rapidly as the size of the neighborhood increases, which is not the case with the Mahalanobis-distance classifier. In fact, 274 (78.5%) of the 349 errors by the perfect metrics can be corrected if we take the top two choices, whereas only 164 (16.8%) of the 979 errors by the Mahalanobis distance classifier can be corrected similarly.

Classification performance of perfect metrics on 3755 classes

Our second experiment was to evaluate the perfect-metric classifier for the 3755 classes in full GB2312-80 Level 1. The classifier was trained on 800 samples of each of the 3755 classes, and tested on another 800 samples of each class. A total of $800 \times 3755 = 3,004,000$ samples were tested. Table 2 summarizes the classification results. Table 3 shows the number of errors and correct rates in neighborhoods of various sizes.

To judge the degree of perfection of the metrics on the test set, we count the occurrence frequencies of each value of the Hamming distance for the correct and incorrect classes respectively. Figure 4 shows these frequencies. Figure 4 suggests that the metrics are "nearly perfect" on the test set since the input matches the correct class with a zero distance in 80.75% of the cases, while incorrect classes have zero distance in fewer than 0.0003% of the cases. The same point can be made, more strikingly, by picking a threshold distance of two: then, 95.8777% of distances

Table 2: Classification results by perfect metrics for 3755 classes.

point size	number of samples	number of errors	% correct
8	751,000	14,650	98.05
10	751,000	4,464	99.41
12	751,000	5,347	99.29
14	751,000	4,692	99.38
font			
Song	751,000	7,707	98.97
FangSong	751,000	5,116	99.32
Hei	751,000	11,136	98.52
Kai	751,000	5,194	99.31
total	3,004,000	29,153	99.03

to correct classes are $\leq$ threshold, while 99.9986% of distances to incorrect classes are above threshold. In another interesting statistic, if we reject (refuse to commit on a decision) when the top choice distance is greater than six, then we miss only 0.62% of the correct decisions and avoid 11.93% of the substitution errors.

For 22,142 images including 11,200 (38.42%) of the errors, the feature vector was matched to more than one classes with zero distance. This could be caused by two reasons: (1) the input image is identical to certain instances of an incorrect class under a certain set of distortion parameters; and (2) the feature set is not sufficient to distinguish between certain pairs of classes. While errors of the first type are intrinsic to the problem, those of the second type can be corrected by enhancing the feature set.

4 Conclusions

We have shown that, for a challenging multi-font, multi-size, large-alphabet character recognition problem, distribution maps of one-dimensional projections capture many details of the class-conditional distributions in an integer-valued feature space. These maps can be used to construct a set of metrics that are perfect on training samples and nearly perfect on testing samples. A classifier built using these metrics compares favorably to a conventional parametric classifier in top-choice accuracy, reject behavior, and computational efficiency.

We have presented a method for making good use of any set of features which satisfy these criteria:

- the cardinality of feature values should be *small enough*, compared to the number of training samples per class, so that every value occurring in the test set is likely to occur in the training set; and

- the cardinality of feature values should be *large enough* so that for every pair of classes there is at

Table 3: Correct rates in various size neighborhoods, for 3755 classes.

	N: size of neighborhood									
	1	2	3	4	5	6	7	8	9	10
number of errors	29,153	6,919	4,419	3,423	2,819	2,411	2,103	1,873	1,728	1,566
correct rate (%)	99.03	99.77	99.85	99.89	99.91	99.92	99.93	99.94	99.94	99.95

least one feature for which it is likely that their distribution maps are disjoint.

As we have shown, the first criterion can be met by the application of realistic pseudorandom image distortion generators — and, we conjecture, perhaps in no other way. The second criterion is harder to satisfy with presently-available methods for proposing and evaluating features.

Further research on perfect metrics and other uses of distribution maps will probably include: (a) methods of constructing and pruning usable features; (b) fast algorithms for evaluating perfect metrics; and (c) applications to problems of larger scale.

Acknowledgements

We would like to thank Eugene Kleinberg for the communications of his theory, and George Nagy and David Ittner for their helpful comments. The font descriptions used in the experiments were supplied by William Sun along with his software package "GB2PS" distributed on Internet. We are thankful for his contribution.

References

[1] H.S. Baird, Document Image Defect Models, in H.S. Baird, H. Bunke, K. Yamamoto (Eds.), *Structured Document Image Analysis*, Springer-Verlag, 1992.

[2] *Code of Chinese Graphic Character for Information Interchange, Primary Set (GB2312-80)*, National Standards Bureau, Beijing, China, 1980.

[3] R.O. Duda, P.E. Hart, *Pattern Classification and Scene Analysis*, Addison-Wesley, New York, 1973.

[4] M.P. Ekstrom, *Digital Image Processing Techniques*, Academic Press, Orlando, 1984.

[5] E.M. Kleinberg, Stochastic Discrimination, *Annals of Mathematics and Artificial Intelligence*, **1**, 1990, 207-239.

[6] E.M. Kleinberg, The Theory of Stochastic Modeling in Pattern Recognition, in preparation.

[7] S. Mori, K. Yamamoto, M. Yasuda, Research on Machine Recognition of Handprinted Characters, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-6**, 4, July 1984, 386-405.

[8] W.F. Schreiber, *Fundamentals of Electronic Imaging Systems*, Springer-Verlag, Berlin, 1986.

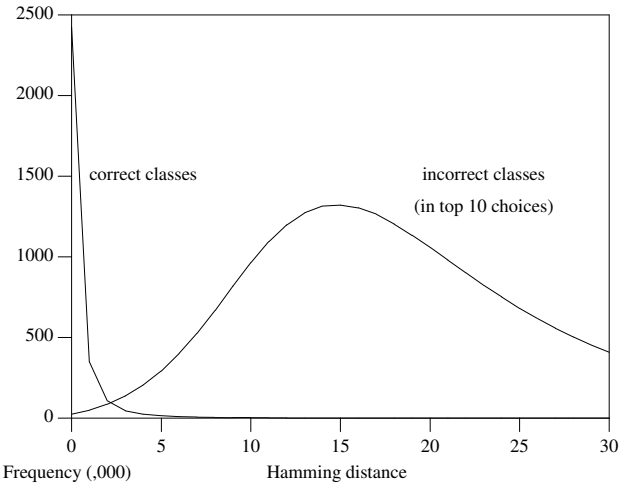


Figure 4: Distribution of Hamming distances in top 10 choices.