[6] T.K. Ho, H.S. Baird, Perfect Metrics, *Proceedings of the Second International Conference on Document Analysis and Recognition*, Tsukuba Science City, Japan, October 20-22, 1993, pp. 593-597.

[7] T.K. Ho, H.S. Baird, Asymptotic Accuracy of Two-Class Discrimination, *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, April 11-13, 1994, pp. 275-288.

[8] T.K. Ho, Selection of Image Features for Distribution-Map Classifiers, *SPIE Proceedings Vol. 2422, Document Recognition II*, San Jose, February 5-10, 1995, pp. 105-114.

[9] T.K. Ho, E.M. Kleinberg, Building Projectable Classifiers of Arbitrary Complexity, *Proceedings of the 13th International Conference on Pattern Recognition*, Vienna, Austria, August 25-30, 1996, pp. 880-885.

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

[11] E.M. Kleinberg, An overtraining-resistant stochastic modeling method for pattern recognition, *Annals of Statistics*, **4**, 6, December 1996.

[12] 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, pp. 386-405.

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

a similar reason, we did not find the method of *projection pursuit* [5] in exploratory data analysis useful in this context.

4 Conclusions

We described a method for inferring 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). We have shown that a classifier can be constructed using a metric defined on these distribution maps.

Our method requires unusually large and representative training sets, which we provided through pseudo-random generation of training samples using a realistic model of printing and imaging distortions. We illustrated 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 showed that it is superior to a conventional parametric classifier using comparable resources. The reject behavior, permitted by the distinctive distributions of distances to correct or incorrect classes, is also very reliable.

We have also shown a way to construct similar feature transformations and classifiers for arbitrary domains. The features and the metric were derived with minimum heuristics. Moreover, the classifier's accuracy can be improved with additional training data. In this study we have concentrated on linear mappings as features. Future studies may focus on investigating other families of features, especially those that can be used under weaker conditions on the class-conditional distributions.

Acknowledgements

We would like to thank Eugene Kleinberg for the communications of his theory, and Robert Haralick, David Ittner, George Nagy, and Theo Pavlidis 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, pp. 546-561.

[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] J.H. Friedman, J.W. Tukey, A Projection Pursuit Algorithm for Exploratory Data Analysis, *IEEE Transactions on Computers*, **c-23**, 9, September 1974, pp. 881-890.

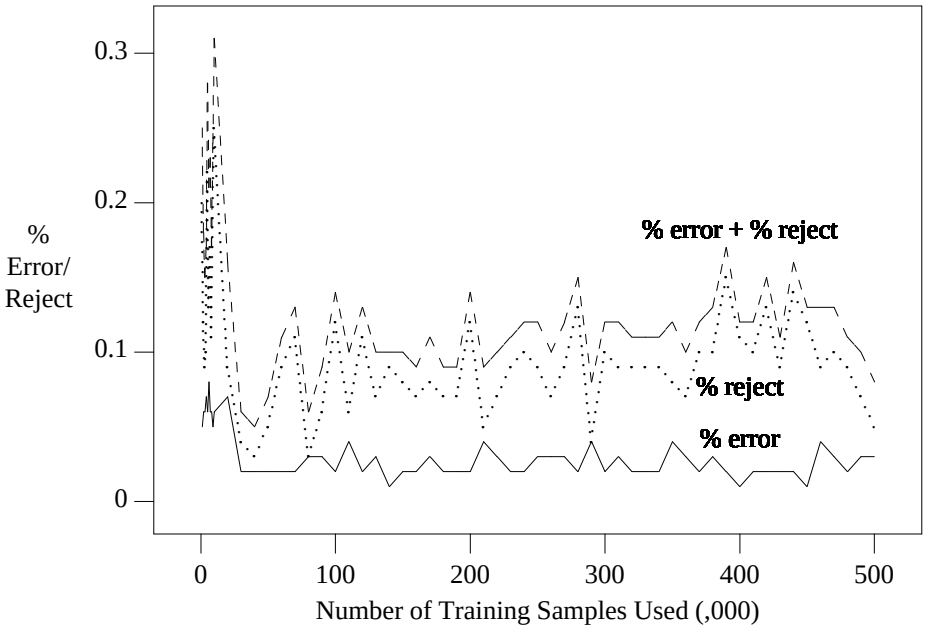

Figure 6: Changes in error and reject rates as training samples were added.

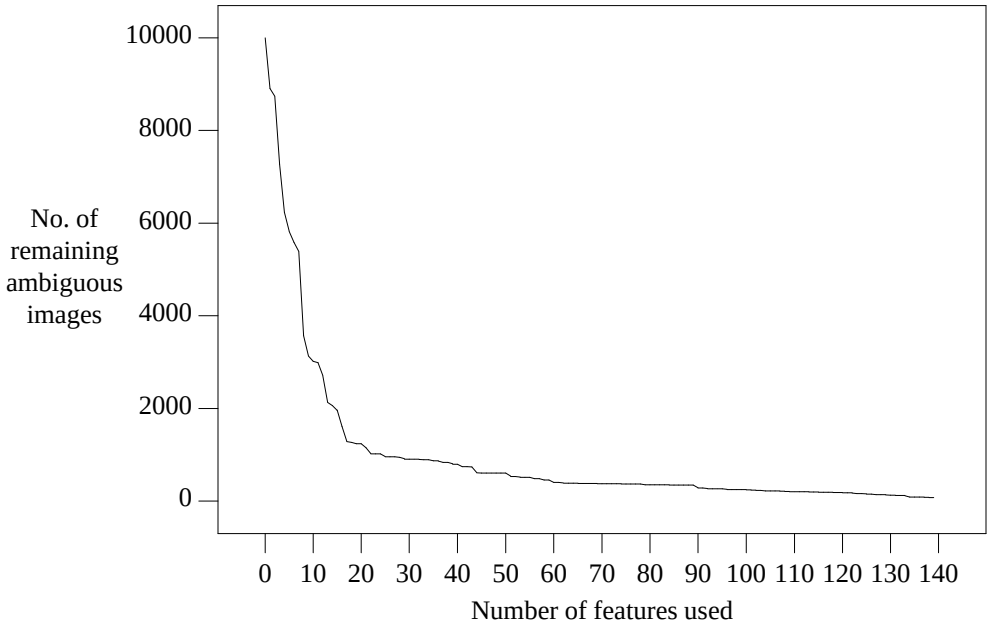

Figure 7: Efficiency of individual features in distinguishing ambiguous images.

Table 4: Changes in classifier accuracy and size

training set added	total #images for training	#images correct	% correct	#images erroneous	% erroneous	#images rejected	% rejected	#new features	total # features
1	1,000	99,749	99.75	52	0.05	199	0.20	1	1
2	2,000	99,828	99.83	56	0.06	116	0.12	2	3
3	3,000	99,848	99.85	60	0.06	92	0.09	1	4
4	4,000	99,836	99.84	68	0.07	96	0.10	2	6
5	5,000	99,718	99.72	59	0.06	223	0.22	2	8
6	6,000	99,792	99.79	77	0.08	131	0.13	2	10
7	7,000	99,775	99.78	60	0.06	165	0.17	2	12
8	8,000	99,828	99.83	60	0.06	112	0.11	2	14
9	9,000	99,759	99.76	50	0.05	191	0.19	0	14
10	10,000	99,691	99.69	56	0.06	253	0.25	1	15
11	20,000	99,841	99.84	66	0.07	93	0.09	2	17
12	30,000	99,932	99.93	23	0.02	45	0.04	2	19
13	40,000	99,953	99.95	22	0.02	25	0.03	2	21
14	50,000	99,931	99.93	22	0.02	47	0.05	3	24
15	60,000	99,890	99.89	21	0.02	89	0.09	2	26
16	70,000	99,867	99.87	23	0.02	110	0.11	3	29
17	80,000	99,941	99.94	28	0.03	31	0.03	2	31
18	90,000	99,913	99.91	25	0.03	62	0.06	3	34
19	100,000	99,860	99.86	20	0.02	120	0.12	3	37
20	110,000	99,903	99.90	37	0.04	60	0.06	3	40
21	120,000	99,872	99.87	16	0.02	112	0.11	2	42
22	130,000	99,900	99.90	25	0.03	75	0.07	2	44
23	140,000	99,893	99.89	14	0.01	93	0.09	3	47
24	150,000	99,895	99.89	22	0.02	83	0.08	3	50
25	160,000	99,909	99.91	16	0.02	75	0.07	3	53
26	170,000	99,890	99.89	29	0.03	81	0.08	2	55
27	180,000	99,918	99.92	16	0.02	66	0.07	3	58
28	190,000	99,905	99.91	21	0.02	74	0.07	3	61
29	200,000	99,868	99.87	16	0.02	116	0.12	3	64
30	210,000	99,909	99.91	38	0.04	53	0.05	3	67
31	220,000	99,905	99.91	27	0.03	68	0.07	3	70
32	230,000	99,893	99.89	20	0.02	87	0.09	2	72
33	240,000	99,886	99.89	18	0.02	96	0.10	3	75
34	250,000	99,888	99.89	26	0.03	86	0.09	3	78
35	260,000	99,909	99.91	26	0.03	65	0.07	2	80
36	270,000	99,884	99.88	27	0.03	89	0.09	3	83
37	280,000	99,854	99.85	20	0.02	126	0.13	3	86
38	290,000	99,924	99.92	35	0.04	41	0.04	2	88
39	300,000	99,874	99.87	23	0.02	103	0.10	2	90
40	310,000	99,879	99.88	27	0.03	94	0.09	4	94
41	320,000	99,888	99.89	23	0.02	89	0.09	2	96
42	330,000	99,898	99.90	16	0.02	86	0.09	3	99
43	340,000	99,893	99.89	16	0.02	91	0.09	2	101
44	350,000	99,881	99.88	37	0.04	82	0.08	3	104
45	360,000	99,910	99.91	25	0.03	65	0.07	2	106
46	370,000	99,881	99.88	16	0.02	103	0.10	2	108
47	380,000	99,873	99.87	26	0.03	101	0.10	2	110
48	390,000	99,834	99.83	16	0.02	150	0.15	4	114
49	400,000	99,877	99.88	15	0.01	108	0.11	2	116
50	410,000	99,883	99.88	17	0.02	100	0.10	2	118
51	420,000	99,845	99.84	23	0.02	132	0.13	3	121
52	430,000	99,890	99.89	21	0.02	89	0.09	2	123
53	440,000	99,839	99.84	23	0.02	138	0.14	3	126
54	450,000	99,864	99.86	15	0.01	121	0.12	3	129
55	460,000	99,871	99.87	41	0.04	88	0.09	3	132
56	470,000	99,872	99.87	25	0.03	103	0.10	2	134
57	480,000	99,892	99.89	21	0.02	87	0.09	3	137
58	490,000	99,906	99.91	25	0.03	69	0.07	2	139
59	500,000	99,919	99.92	33	0.03	48	0.05	2	141

The test set consists of 50,000 'c's and 50,000 'e's (100,000 samples in total). The same test set is used to measure the error rate throughout the experiment, regardless of changes in the training set and the features. Each sample image is binarized, size-normalized to 48×48 pixels. The binary values of the pixels are used as input.

At each iteration, linear projections were found using the fixed increment perceptron training algorithm. For each projection, the range of values was divided into 25 segments. Once sufficient features were found to fully discriminate a given training set, the next training set was added and the algorithm searched for new features to resolve new ambiguities. Because of difficulties in implementation, we considered ambiguities in the new training set only and did not backtrack and re-examine previous training samples.

When the algorithm finished with a training set, we tested the accuracy of the classifier with the fixed test set. Since we have an unlimited source of training data, in principle this could continue indefinitely. However, seeing no new pattern on the results, we terminated the experiment after 500,000 training samples were used. Table 4 shows the changes in classification accuracy and the number of features added with each new training set.

From Table 4 it is not difficult to see that the accuracy of the classifier stabilized early (when 100,000 training samples were used). This is more obvious from the graphs in Figure 6.

The results suggest that the initially derived features contribute most to classification accuracy. We can measure the efficiency of each feature by the number (or %) of samples separated from other classes by using that feature. As an example, we examined the 139 features obtained before training set 59 was added. Figure 7 shows the numbers of samples distinguished by each of those 139 features, starting from the features derived using training set 1. This graph illustrates the ability of the method in selecting the most useful features first. Since the features are ordered by their efficiency, the sequence can be pruned according to requirements on accuracy and availability of resources.

It is unclear what has caused the persistent errors and rejects. Our conjecture is that the class-conditional distributions have nonzero and overlapping tails, so that there are certain intrinsic ambiguities in the data – relative to the quantization of values we have adopted.

3.5 Alternative Procedures

A few other trials have been carried out using several alternative procedures. The trivial method of obtaining mappings by Euclidean distance, produced very inefficient features, as expected. The method of projection on the line connecting cluster centroids also yielded many inefficient features, with low error rates but high reject rates. Using the same perceptron method with value ranges divided into 50 segments instead of 25 resulted in slower improvements in accuracy, though after 50,000 training samples were used the differences were no longer obvious.

The method of *principal components* (Karhunen-Loéve expansion) [3] was considered for finding useful directions for linear projections. However, maximization of the variability of the entire data set does not contribute much to the *discrimination* between classes. For

3.3 Adaptation to Unseen Samples

So far we have been concerned with separating all samples in a given training set. Given a fixed training set of a small size, this is easily achievable. However, the usefulness of the classifier is determined by its accuracy on previously unseen samples. In this section we discuss ways to adapt the classifier to new samples.

The quantization of the value ranges of the features provides limited invariance of classification among samples projected onto the same line segment. By the definition of the metric used by the classifier, an error occurs when a test sample has a feature value that has not previously occurred among training samples of that class. To prevent this from happening, the training set must be large enough relative to the range and quantization of the feature values, so that the projections can be fully saturated. Yet too many training samples could also lead to the problem of over-saturation – in an extreme case where all class-conditional distributions have infinite nonzero tails, any projection will eventually have all values marked for all classes, and when this happens no discrimination will be possible. In the previous discussions we have excluded these cases by assuming that each sample has a unique classification. But in practice this often cannot be guaranteed.

It is therefore difficult to predict the quantity of training samples needed. An adaptive procedure is preferred, so that the classifier can improve itself by using additional training samples. It is possible that new training samples are projected to values that are marked for other classes but unmarked for their classes, and thus introduce new ambiguities. The classifier will need new features to resolve such ambiguities. If we keep adding in new training samples and find no further need to include new features, the classifier can then be considered stable. Errors will also be unlikely since newer samples are then less likely to turn on values unmarked for their classes.

The way new features are found is similar to the selection of other features. We first project the new training samples using existing mappings, and identify those that overlap with other classes. Then we look for hyperplanes that separate those samples from other classes. Older training samples of other classes also need to be re-examined for overlaps with the new samples.

3.4 Results of Feature Selection

We tested the procedure in an experiment on a constrained problem in optical character recognition. We chose the problem of distinguishing between images of the symbols 'c' and 'e' in the Adobe Times Roman typeface, with noise introduced by a parameterized model of document image defects [1]. The choice was motivated mainly by the practical importance of the problem and the unambiguous shapes of the ideal prototypes. The use of the defect model gives us a way to precisely define the scope of the problem, and also an indefinitely large source of data for training and testing. The values of the defect model parameters are identical to those used in a previous experiment [7], and will be omitted here for brevity.

Using the defect model and the ideal shape prototypes, we generated 500,000 training samples that are divided into 59 sets. There are 1,000 (500 'c's and 500 'e's) samples in each of the first ten training sets, and 10,000 (5,000 'c's and 5,000 'e's) in the subsequent ones.

samples, i.e., projections that can separate a larger set of samples from others. We will now focus on a condition on the class-conditional distributions that would allow discrimination by more efficient projections. We require that for a given problem in a d-dimensional feature space, and for each sample x, there exists one or a number of parallel $(d-1)$-dimensional hyperplanes that can separate x from samples of all other classes. We call these classes *partially linearly separable*. Note that this condition is weaker than linear separability, which requires all samples of one class be separable from samples of all other classes by a single hyperplane. It occurs in practice that, in many applications with high-dimensional input, different classes rarely spread over nested regions, and the classes are often partially linearly separable. Figure 5 shows an example of two partially linearly separable classes.

3.2 Progressive Elimination of Ambiguities

We now describe an algorithm that searches for projections that fully discriminate between partially linearly separable classes. The algorithm removes ambiguities progressively by introducing additional features that are targeted towards separating current ambiguous samples.

The algorithm is applied to each class in turn. Suppose we start with a particular class c. At the beginning, all training samples are first assigned to two sets S_1 and S_2, where S_1 consists of all samples from class c, and S_2 consists of samples from all other classes. The sample mean m_i of each set S_i $(i = 1, 2)$ are computed. A line $\overline{m_1 m_2}$ is drawn passing through m_1 and m_2. All samples are then projected onto this line. The range of the projection is then divided evenly into a fixed number of segments. A segment is marked for a class if there is any sample of that class projected onto that segment. A distribution map of the classes along this line is thus obtained.

If there is no segment marked for classes from both sets, then we have obtained a discriminating feature for class c. Otherwise, S_1 is pruned and only those samples overlapping with S_2 are retained (call the pruned set S_1'). The procedure is then repeated using S_1' and S_2. In case that all samples in S_1 overlap with those from S_2, S_1 is split into two halves and the same procedure is applied to each half. This continues until either S_1 is empty, or it is impossible to separate S_1 and S_2 by any further projections.

Notice that when each new projection is made, some samples that are previously removed from S_1 may overlap with S_2 when they are projected onto the new line. But there is no need to backtrack to those samples because there has already been one projection on which they can be separated. After S_1 is finished, we put class c into S_2 and take another unprocessed class from S_2 to S_1, and the algorithm iterates until all classes are processed.

In this procedure, the directions of the projections are determined by the geometry of the spread of the classes, over which we have little control. An improvement can be made by introducing an optimization procedure to choose projections that lead to as few ambiguities as possible. The fixed increment perceptron training algorithm can be used in this context to find a hyperplane between S_1 and S_2, so that the number of samples falling on the opposite side is minimized. The only change from the previously described procedure is to replace the line $\overline{m_1 m_2}$ by the line perpendicular to the resultant hyperplane.

not introduce classification errors. The worst case is when distributions of all classes are projected onto identical values, so that the projection does not introduce any difference in the values of the metric.

By the definition of the metric, the classifier is more prone to indecisions due to ambiguities than to errors. A sample has a unique classification when the metric is minimized for one single class. This occurs when there is at least one value range in which the sample is outside all but one class model. We now describe a method to obtain projections such that this condition is satisfied as far as possible. Following the convention in image recognition, we will call each of these projections a *feature* of the sample.

3.1 Efficiency of Features

Given a fixed set of training samples of n classes, we need a set of projections so that each sample of each class is separated from samples of all other classes by at least one projection. We assume that there is no intrinsic ambiguities in the samples, i.e., no two classes share an identical sample.

A set of projections can be obtained trivially as follows: pick an arbitrary metric in the feature space, say, the Euclidean distance $d(x, y)$ between two samples x and y. For each sample x, compute $d(x, y)$ between x and all other samples $y(y \neq x)$. Set the value range to be $[0, max(d(x, y))]$. Then we have a mapping from the feature space onto this value range, which is simply the distance from this particular sample. Do this for all samples, and each sample will then be separated from all others by the distance measured from itself.

The projections thus obtained guarantee the discrimination between training samples of different classes, but they are often not satisfactory to be used in the distribution-map based classifier. The projections are too specific to particular samples, and too many of them are needed. For this reason we call these projections *inefficient*. A constant scaling factor may be used to compress the value ranges, so that the projected value remains invariant for samples in the immediate neighborhood of the sample from which distance is measured. But this remedy may not work in application domains where samples are far apart in a large space, or when they are at almost the same distance from each other.

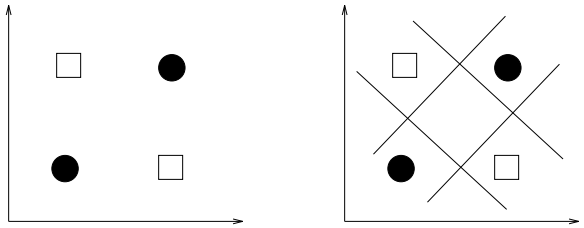

Figure 5: Two classes (c_1: black circles, c_2: white squares) that are not linearly separable but are partially linearly separable.

What we need are projections that better exploit local cluster characteristics of the

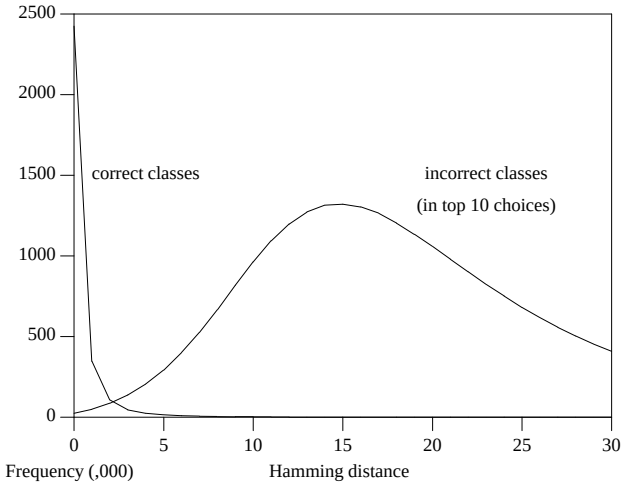

Figure 4: Distribution of Hamming distances in top 10 choices.

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.

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

3 Feature Transformations for Arbitrary Domains

In the character recognition experiment we project the samples using some well-known feature transformations. In other application domains, such transformations may not be readily available. In the sections below we discuss a way to automatically construct projections suitable for this classifier.

Our method imposes no restriction on the form of the projections, given that they satisfy several criteria. First, the value range has to be bounded and discretized, and has a suitable cardinality. The cardinality (number of discrete values) needed depends on the number of available training samples. It has to be small enough so that training samples limited in quantity (say, a few hundreds) can form a dense model. It has to be large enough to reveal differences between samples of different classes. For this reason, binary values are usually not sufficient to differentiate the classes. As an example, in the previous experiment, we found that a cardinality of 25 for each projection was useful with 800 training samples for each class.

A second criterion is that the projections should be different from each other. A projection need not be strictly orthogonal to all others. It may add to the discriminating power of the classifier as long as it is not identical to others. In principle, a projection does

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.

2.5 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.

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 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 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

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.

2.4 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 distance	
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

bytes of storage. Note that this representation can express multi-modal distributions, and is not limited to parametric approximations.

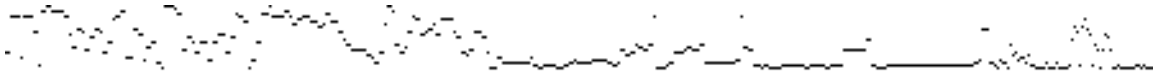

Figure 2: The feature values for one sample image, shown as a distribution map. The horizontal coordinate ranges from 0 to 447, representing each of the 448 features. The vertical coordinate ranges from 0 to 24, representing the value of each feature.

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

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 circumstances, 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.

On the test set, we cannot expect the metrics to be perfect, but, as we shall see, they can be 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

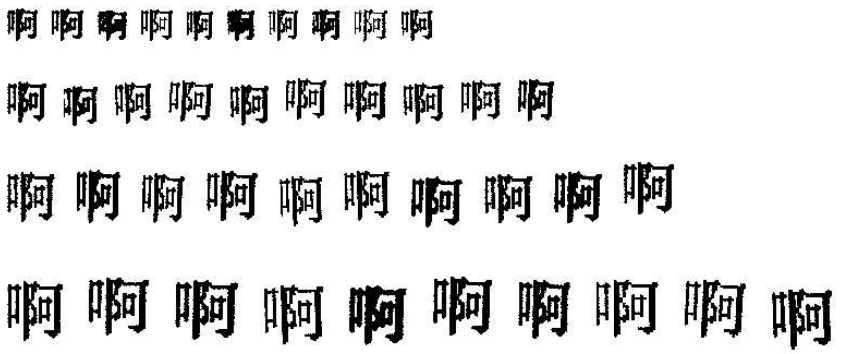

Figure 1: Examples of distorted images generated by the defect model.

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.

2.3 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

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 projected to 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 detectable, 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.

2.2 Generation of 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] & [13]). 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,

The experiments embrace all 3755 character classes of the GuoBiao Coding GB2312-80, Level 1 [2].

2.1 The Features

In this experiment we are concerned with the best use of given feature transformations. Therefore we have chosen, somewhat arbitrarily, some features commonly used in published Chinese recognition systems [12]. The binary image of an input character is first size-normalized to a 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

1 Introduction

One of the challenges in pattern recognition is to find faithful but concise descriptions of the class-conditional distributions of the samples. The samples are typically expressed as points in a high-dimensional feature space, and they often spread over regions that are in general non-simply connected, multi-modal, and nonconvex. This irregular geometry adds to the difficulty of finding compact expressions; in particular, 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 [9] [10] [11]. 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 empirical class-conditional distributions by first exerting some 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. The image generator provides a source of indefinitely many samples which are projected down to finite and discrete ranges of feature values to produce dense one-dimensional maps. A series of such maps together give a distinctive profile of each class.

The distinctive profiles can be used to construct a metric for classification. For an arbitrary test sample, the metric counts the number of projected value ranges of each class that do not include that sample. The metric is zero when the sample fits the profile of a particular class perfectly. Larger values of the metric indicate discrepancies of the sample from the ranges of feature values of the corresponding classes.

We will first illustrate the method with an experiment in which the classifier is applied to the recognition of 3755 classes of machine-print Chinese characters in four typefaces. In this experiment we use feature transformations designed specifically for character images. The feature extractors transform a binary-valued image into an integer-valued vector with component values in limited ranges. The class-conditional distributions in this feature space are mapped. For applications to other domains, it is necessary to have feature transformations that can be used similarly. We will describe a method to select suitable linear transformations that can project the classes into subspaces where they are partially separable[8], so that a similar classifier can be constructed.[1]

2 Distribution Maps of Printed Characters

To illustrate the mapping process, we designed an experiment to 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.

[1] Preliminary versions of this paper have appeared in [6] [8].

Pattern Classification with Compact Distribution Maps

Tin Kam Ho, Henry S. Baird
Bell Laboratories
Lucent Technologies

March 3, 1998

Abstract

A difficult problem in classification is representing the class-conditional distributions concisely and faithfully. We propose a way of mapping such distributions and its use in constructing a similarity metric. A classifier using this metric can achieve low error rates and useful confidence scores permitting reliable reject behavior. We illustrate the method by an application in a challenging character recognition problem with thousands of classes. For applications to arbitrary domains, we present a method to automatically construct feature transformations that are suitable for such mappings.