

# Optimizing Binary Feature Vector Similarity Measure using Genetic Algorithm and Handwritten Character Recognition

Sung-Hyuk Cha, Charles C. Tappert  
Department of Computer Science  
School of Computer Science and Information System  
Pace University, Pleasantville, New York, U.S.A.  
{scha,ctappert}@pace.edu

Sargur N. Srihari  
Center of Excellence for Document Analysis and Recognition (CEDAR)  
University at Buffalo, State University of New York  
Buffalo, New York, U. S. A.  
srihari@cedar.buffalo.edu

## Abstract

*Classifying an unknown input is a fundamental problem in pattern recognition. A common method is to define a distance metric between patterns and find the most similar pattern in the reference set. When patterns are in binary feature vector form, there have been two approaches to improve the performance over the equal-weighted Hamming distance metric. One is to give different weights to different features using an optimization technique, and the other is to use a similarity measure that gives full credit to features present in both patterns and the less credit to those absent from both patterns. Both approaches have been reported to perform better than the naïve Hamming distance approach. In this paper, we propose to combine these two approaches using a genetic algorithm to optimize weights. Experimental results show that this method is superior to conventional measures in an OCR application.*

**Key words:** Handwriting Recognition, Nearest neighbor, Genetic Algorithm, Similarity Measure

## 1 Introduction

A common method for classifying an unknown input vector involves finding the top  $k$  similar vectors in a reference set. The  $k$ -nearest neighbor, or simply  $k$ -nn, has wide acceptance in character recognition (see [1, 2] for extensive surveys). There are two important aspects in this approach. One is selecting important features from the character image, and the other is selecting an appropriate similarity measure.

*Gradient, Structural, and Concavity*, or simply *GSC*, features have been found and utilized to recognize a character. Since Favata, Srikanthan, and Srihari introduced this multiple feature/resolution approach [3, 4, 5, 6], it has undergone a period of reappraisal of its effectiveness, and alternatives or modifications have been considered to further improve the

performance. While GSC features are considered significant ones, relatively little study has been carried out to select and design a good similarity measure for GSC recognizers. In this paper, we introduce a way to evaluate the similarity measure and to find the optimal similarity measure according to the evaluation.

There are many definitions of distance measures encountered in various fields such as information retrieval and biological taxonomy [7]. Common definitions include *Euclidean*, *Minkowski*, *cosine*, *dot product*, *Tanimoto distance*, etc. Here, we examine most of the currently used similarity measures on a database of mixed hand-printed/cursive characters after extracting *GSC* binary features.

The *hamming distance* is widely for binary features. To further improve the performance, different weights can be applied to features [7, 8] using common optimizing techniques such as gradient descent or genetic algorithms [11, 12]. Another approach is to use a similarity measure that gives full credit to features present in both patterns, less credit to those not present in either pattern, and no credit to those present in only one of the patterns to be matched [3]. Both approaches have been reported to perform better than the naïve Hamming distance approach. In this paper, we propose to combine these two approaches using a genetic algorithm to optimize the weights. We present experimental results that demonstrate its superiority over conventional approaches in an OCR application.

The subsequent sections are organized as follows. Section 2 gives a general review of GSC features used in off-line character recognition. Section 3 discusses ways to evaluate the similarity measures. In section 4 we propose an additive compound feature distance metric, and in section 5 we present experimental results in an off-line character recognition problem to show the effectiveness of this metric. Finally, section 6 concludes the paper.

## 2 Review of GSC features

Among many classifiers, the GSC (*Gradient, Structural, and Concavity*) classifier has been shown to have the highest accuracy in off-line character recognition problems [4]. It is based on the philosophy that feature sets can be designed to extract certain types of information from the image [3, 4, 5, 6]. These types are gradient, structural, and concavity information. Gradient features use the stroke shapes on a small scale, structural features use stroke trajectories on an intermediate scale, and concavity features use stroke relationships at long distances.

All the features are listed in Table 1. The input character image is a binarized and slant-normalized image. A bounding box is placed around the image and it is divided into  $4 \times 4$  grid as shown in Figure 1. This approach is known as a quasi-multiresolution approach. For each grid, all directional, rules and various concavity features are checked. Therefore, Gradient, Structural, and Concavity have 192, 192, and 128 features, correspondingly. In all, there are 512 features. A sample vector for a character "A" is given in Figure 2. See [3] for a detailed description of the rules.

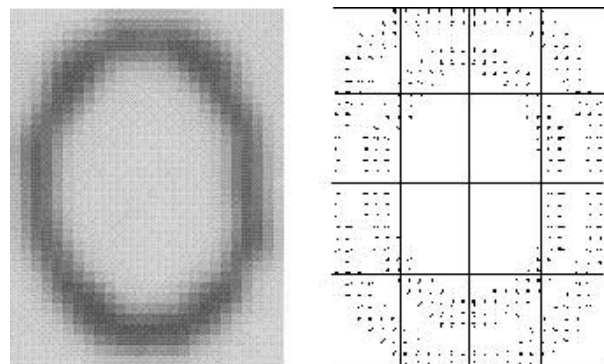


Figure 1.  $4 \times 4$  grid

Grid	Gradient		Structural		Concavity Features	
	ID	Directional	ID	Rule	ID	Concavity
(0,0)	G01-00	$1^\circ \sim 30^\circ$	S01-00	r <sub>1</sub>	C-CP-00	coarse pixel density
...	...	...	...	...	...	...
(3,3)	G01-33	$1^\circ \sim 30^\circ$	S01-33	r <sub>1</sub>	C-CP-33	coarse pixel density
(x,y)	G02-xy	$31^\circ \sim 60^\circ$	S02-xy	r <sub>2</sub>	C-HR-xy	horizontal run length
(x,y)	G03-xy	$61^\circ \sim 90^\circ$	S03-xy	r <sub>3</sub>	C-VR-xy	vertical run length
(x,y)	G04-xy	$91^\circ \sim 120^\circ$	S04-xy	r <sub>4</sub>	C-UC-xy	upward concavity
(x,y)	G05-xy	$121^\circ \sim 150^\circ$	S05-xy	r <sub>5</sub>	C-DC-xy	downward concavity
(x,y)	G06-xy	$151^\circ \sim 180^\circ$	S06-xy	r <sub>6</sub>	C-LC-xy	left concavity
(x,y)	G07-xy	$181^\circ \sim 210^\circ$	S07-xy	r <sub>7</sub>	C-RC-xy	right concavity
(x,y)	G08-xy	$211^\circ \sim 240^\circ$	S08-xy	r <sub>8</sub>	C-HC-xy	hole concavity
(x,y)	G09-xy	$241^\circ \sim 270^\circ$	S09-xy	r <sub>9</sub>		
(x,y)	G10-xy	$271^\circ \sim 300^\circ$	S10-xy	r <sub>10</sub>		
(x,y)	G11-xy	$301^\circ \sim 330^\circ$	S11-xy	r <sub>11</sub>		
(x,y)	G12-xy	$331^\circ \sim 360^\circ$	S12-xy	r <sub>12</sub>		

Table 1. GSC features where  $x, y = 0 \dots 3$

In order to classify an input vector into one of the 26 letters of the alphabet, the  $k$ -nearest neighbor ( $k$ -nn) approach



Figure 2. A sample character and its GSC features

is used. To compute the distance, the following definition of similarity was previously used.

$$S[x, y] = x^t y + \frac{\bar{x}^t \bar{y}}{\sigma} \quad (1)$$

where  $\sigma$  is the contribution factor, and usually  $1 < \sigma < 5$ . The term  $x^t y$  indicates the number of 1 bits that match between  $x$  and  $y$  and the term  $\bar{x}^t \bar{y}$  denotes the number of 0 bits that match between them.

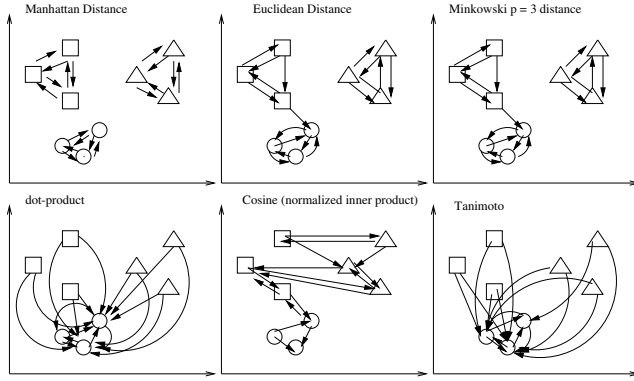
Various similarity measures, such as *Euclidean*, *Minkowski*, *cosine*, and *dot product*, have been examined, but the measure given in equation (1) is superior to these other distance or similarity measures. Also, we observed the change in recognition performance for different values of  $\sigma$ , and found the optimal value to be  $\sigma = 1.9$ , whereas  $\sigma = 2$  was used in the previous GSC classifier on a particular test set [3, 4, 5, 6]. Also, the measure given in equation (1) performs better than the *hamming distance*

$$S_H[x, y] = x^t y \quad (2)$$

## 3 Similarity Measure Evaluation

It is important to choose a suitable similarity measure between feature vectors. In this section, we discuss how to evaluate the performance of a similarity measure. One can test the similarity measure by using a test set but this method is test-set dependent. A more general method is to divide samples into subsets randomly. Tuning is performed repetitively to avoid an accidental extremity in one set, thereby averaging out extreme cases. If we use a single tuning set, tuning can be misled by random errors or coincidental regularities within the set. However, obtaining many reasonably large subsets is quite costly and tedious.

For this reason, we use a different evaluating method for  $k$ -nearest neighbor classifiers. It is the "all  $k$ -nearest neighbor" approach. We will perform the *all k-nearest neighbor* algorithm for the reference set and count mismatches, where



**Figure 3.** All  $k$ -nearest neighbor graph

the all  $k$ -nearest neighbor algorithm determines the  $k$ -nearest neighbors for each point of the reference set  $R$ . Since we use the entire reference set to classify input vectors of the unknown class, we would like to choose the similarity measure that best clusters reference samples of the same class.

Fig. 3 which shows two-dimensional continuous features, and, for  $k = 2$ , shows every pattern pointing to its two nearest neighbors. Depending on the distance measure used in each case, the nearest neighbors can be considerably different. Now we can count the errors for each measure using the evaluation measure

$$E(x) = \frac{\text{Number of errors}}{k \times n}$$

where  $n$  is the number of samples. For example,  $E(\text{Manhattan}) = 0$ , while  $E(\text{Euclidean}) = 0.0556$ . Clearly, in this example, we would like to use the Manhattan distance.

Although computing all  $k$ -nearest neighbor takes only  $O(n \log n)$  [9], binary feature vectors or non-Euclidean space features cannot be solved in this manner. However, they can certainly be solved in  $O(n^2)$  by apply the the nearest neighbor search for each element in the reference set.

The simplest way to select the evaluation function is to choose the one with the minimum error rate. Another approach is to use the error versus reject percentages of the recognition graph [10], as shown in Fig. 4, where the  $x$  and  $y$  axes represent the error rate and reject rate, respectively. A good recognizer must be as close to the origin as possible. Practically, however, the measure minimized in the range between 5% to 20% error is the best measure. We will use this approach to evaluate and optimize the similarity measure. Since computing the exact size of the area is difficult, we simply measure the heights in the 5% to 20% error range.

## 4 Compound Feature

The number of weights is enormous if each feature is given its own weight. It creates not only a huge space to search for the optimum, but also the recognition execution

time is excessive. In order to make our claim clear, we simplify the features into three compound features.

In contrast to the similarity measure given in equation (1), we divide the feature set into three homogeneous groups: gradient, structural, and concavity feature sets, and find the optimal similarity measure for each set. Then we use the additive model to combine these measures into one as follows:

$$\begin{aligned} S[x, y] &= S^I[x_g, y_g] + S^{II}[x_s, y_s] + S^{III}[x_c, y_c] \quad (3) \\ &= \left(x_g^t y_g + \frac{\bar{x}_g^t \bar{y}_g}{\sigma_g}\right) + \left(x_s^t y_s + \frac{\bar{x}_s^t \bar{y}_s}{\sigma_s}\right) \\ &\quad + \left(x_c^t y_c + \frac{\bar{x}_c^t \bar{y}_c}{\sigma_c}\right) \end{aligned}$$

where  $x_g$ ,  $x_s$ , and  $x_c$  are the subsets of  $x$  consisting of only the gradient, structural, and concavity parts of  $x$ , so that  $x = x_g \cup x_s \cup x_c$ . First, we consider each individual set of GSC features. For  $S^I[x_g, y_g]$ ,  $S^{II}[x_s, y_s]$  and  $S^{III}[x_c, y_c]$ , we use the similarity of equation 1.

Next we associate weights to each feature group of (3).

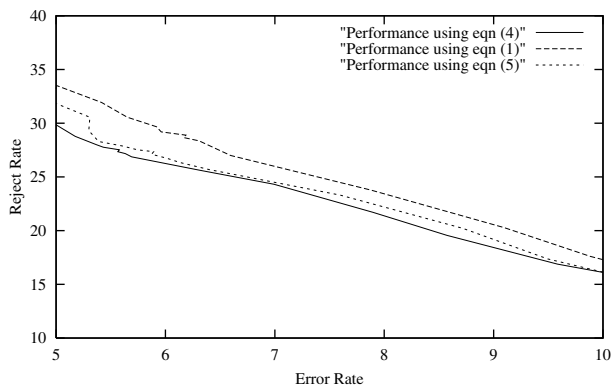
$$\begin{aligned} S_w[x, y] &= w_g \left(x_g^t y_g + \frac{\bar{x}_g^t \bar{y}_g}{\sigma_g}\right) + w_s \left(x_s^t y_s + \frac{\bar{x}_s^t \bar{y}_s}{\sigma_s}\right) \\ &\quad + w_c \left(x_c^t y_c + \frac{\bar{x}_c^t \bar{y}_c}{\sigma_c}\right) \\ &= w_{g1} x_g^t y_g + w_{g2} \bar{x}_g^t \bar{y}_g + w_{s1} x_s^t y_s \\ &\quad + w_{s2} \bar{x}_s^t \bar{y}_s + w_{c1} x_c^t y_c + w_{c2} \bar{x}_c^t \bar{y}_c \quad (4) \end{aligned}$$

Here  $w_g$ ,  $w_s$ , and  $w_c$  are the weights for the Gradient, Structural, and Concavity feature groups. Now we have six coefficients to optimize in this new similarity measure. This is a multi-dimensional, space optimization problem. We use a genetic algorithm to determine the feature weights from the preclassified training set. A genetic algorithm can be a general optimization method that searches a large space of candidate objects to find one that performs near optimal according to the fitness function [11]. Genetic algorithms offer a number of advantages: they search from a set of solutions rather than from a single one, they are not derivative-based, and they explore and exploit the parameter space (Goldberg 1989) [12]. For the weight adaptive model, we create a numerical optimization model that depends on a set of weights  $u$  defined as follows:  $u = \{w_{g1}, w_{g2}, w_{s1}, w_{s2}, w_{c1}, w_{c2}\}$ .

Note that if  $w_{g2} = w_{s2} = w_{c2} = 0$ , we have the simple weighed hamming distance,

$$S[x, y] = w_{g1} x_g^t y_g + w_{s1} x_s^t y_s + w_{c1} x_c^t y_c \quad (5)$$

It is already known that the recognition performances of equations (1) and (5) are better than the naïve hamming distance given in equation (2). Our claim is that the recognition performance of the similarity measure given in equation (4) outperforms those in equations (2), (1), and (5). The following section proves this claim experimentally.



$$\text{Error rate} = \frac{\text{Number of Errors}}{\text{Number of Accepted Queries}} \times 100$$

$$\text{Reject rate} = \frac{\text{Number of Rejections}}{\text{Number of Total Queries}} \times 100$$

**Figure 4.** Error vs. Reject Percentage Graph.

## 5 Experiment

The fitness criterion can be defined as the area of error versus reject percentage graph for a specific weight set. We used a typical *Generic Algorithm* [8] to optimize the weight coefficients:  $w_{g1}$ ,  $w_{g2}$ ,  $w_{s1}$ ,  $w_{s2}$ ,  $w_{c1}$ , and  $w_{c2}$ . The experiment was performed on a 300MHz UltraSparc, SunOS 5.6, 4x782 MIPS, with 2,048 Memory. Our GSC reference set contained 21800 characters (A-Z), and approximately 800 references per character.

To validate that our proposed similarity measure is superior to the other ones, as a validation set we used the *bd-testing data set* consisting of 1681 mixed hand-printed/cursive characters. This validation set served as a safety check for improved performance. Figure 4 shows the improvement on this validation set. The dashed line shows the performance using equation (1) that was previously used in the GSC classifier. The dotted line shows the performance using equation (5), the weighted hamming distance. And finally, the solid line indicates the performance using equation (4), the proposed optimized distance metric. Clearly, the size of the area of the optimized distance metric is smaller than that of the other metrics.

We conclude that the method of combining the two approaches performs better than either one alone, and we demonstrated improved performance of the GSC classifier on off-line character recognition data by changing the similarity measure.

## 6 Conclusion

To conclude, we emphasize that selecting and designing a similarity measure is as important as finding significant fea-

tures. Although the characteristic of the feature vector is one of the design components to be considered, a poor choice of similarity measure can result in unsatisfactory performance in recognition accuracy. In designing a similarity function, it is necessary to have a tuning set in addition to a reference set, and a validation set if coefficients are associated with it. The major contribution of this paper is proposing an improved similarity measure for a GSC recognizer and demonstrating its improved performance on off-line character recognition data.

## References

- [1] B. V. Dasarathy, "Visiting nearest neighbors - a survey of nearest neighbor pattern classification techniques," in *Proceedings of the International Conference on Cybernetics and Society*, pp. 630-636, IEEE, 1977.
- [2] B. V. Dasarathy, "Nearest neighbor pattern classification techniques," in *IEEE Computer Society Press*, 1991.
- [3] J. T. Favata and G. Srikantan, "A multiple feature/resolution approach to handprinted digit and character recognition," *International Journal of Imaging Systems and Technology*, pp. 7:304-311, 1996.
- [4] J. T. Favata, G. Srikantan, and S. N. Srihari, "Hand-printed character/digit recognition using a multiple feature/resolution philosophy," in *IWFHR-IV*, pp. 57-66, December 1994.
- [5] G. Srikantan, S. W. Lam, and S. N. Srihari, "Gradient-based contour encoding for character recognition," *Pattern Recognition*, vol. 29, no. 7, pp. 1147-1160, 1996.
- [6] G. Srikantan, D.-S. Lee, and J. T. Favata, "Comparison of normalization methods for character recognition," in *Proceedings of the Third ICDAR 95*, vol. 2, pp. 719-722, IEEE Computer Society Press, August 1995.
- [7] R. O. Duda and P. E. Hart, *Pattern classification and scene analysis*. New York: Wiley, 1st ed., 1973.
- [8] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. John Wiley & Sons, Inc., 2nd ed., 2000.
- [9] F. P. Preparata and M. I. Shamos, *Computational geometry: an introduction*. Springer-Verlag, 1997.
- [10] C. K. Chow, "On optimum recognition error and reject tradeoff," in *IEEE Transactions on Information Theory*, pp. 16:41-46, 1970.
- [11] M. Mitchell, *An introduction to genetic algorithms*. Cambridge, MA: MIT Press, 1996.
- [12] L. Davis, *Handbook of genetic algorithms*. New York: Van Nostrand Reinhold, 1991.