An Improvement to k-Nearest Neighbor Classifier

P. Viswanath and T. Hitendra Sarma
Department of Computer Science and Engineering,
Rajeev Gandhi Memorial College of Eng. and Technology
Nandyal-518501, A.P., India.
Email: {viswanath.p, hitendrasarma}@ieee.org

Abstract—**Non-parametric methods like Nearest neighbor classifier (NNC) and its variants such as k-nearest neighbor classifier (k-NNC) are simple to use and often shows good performance in practice. It stores all training patterns and searches to find k nearest neighbors of the given test pattern. Some fundamental improvements to k-NNC are (i) weighted k-nearest neighbor classifier (wk-NNC) where a weight to each of the neighbors is given and is used in the classification, (ii) to use a bootstrapped training set instead of the given training set, etc. Hamamoto *et. al.* [1] has given a bootstrapping method, where a training pattern is replaced by a weighted mean of a few of its neighbors from its own class of training patterns. It is shown to improve the classification accuracy in most of the cases. The time to create the bootstrapped set is $O(n^2)$ where n is the number of training patterns. This paper presents a novel improvement to the k-NNC called k-Nearest Neighbor Mean Classifier (k-NNMC). k-NNMC finds k nearest neighbors for each class of training patterns separately, and finds means for each of these k neighbors (class-wise). Classification is done according to the nearest mean pattern. It is shown experimentally using several standard data-sets that the proposed classifier shows better classification accuracy over k-NNC, wk-NNC and k-NNC using Hamamoto's bootstrapped training set. Further, the proposed method does not have a design phase as the Hamamoto's method, and this is suitable for parallel implementations which can be coupled with any indexing and space reduction methods easily. It is a suitable method to be used in data mining applications.**

Index Terms—**Pattern classification, nearest neighbor rule, bootstrapping, k-nearest mean classifier.**

I. Introduction

Pattern classification deals with developing a model or method which classifies the given query data item in to one of predefined classes. A data item in most of the cases is a vector of feature values and is called a pattern. Training set is a set of patterns where for each pattern we know its class. The model or method to do pattern classification is developed by using the training set. There are various pattern classification methods, like, the Bayes classifier, nearest neighbor classifier, Perceptron, multi-layer Perceptron, support vector machines, etc [2]. There is no *the best classifier* suitable for all domains (except in a very rare situation where one has the complete probability structure of the problem like known class conditional densities, apriori probabilities *etc*., in which case the Bayes classifier is the best). A classifier which performs well with image data (like classifying a hand written character) may not perform well to recognize a speaker based on his/her speech. For details refer to "No Free Lunch Theorem" given by Wolpert [3], [2]. So, each classifier has its own value and is applicable in some domains. Nearest neighbor based classifier, even-though was first invented in 1950s [4], [5], it is still in active research areas. This is partly because there are still improvements over the basic method and other improvements which reduces space and time requirements.

Nearest neighbor classifier (NNC) and its variants like k-nearest neighbor classifier (k-NNC) are simple to use and often shows good performance [6], [7], [8], [9]. It is shown that when infinite number of training patterns are available, k-NNC is equivalent to the Bayes classifier and the error of NNC is less than twice the Bayes error [2]. Performance of k-NNC or NNC with a larger training set is better than that with a smaller training set. It has no significant design (training) phase (except like finding k value in case of k-NNC), hence it is highly adoptive to dynamically varying data-sets. Hence NNC or k-NNC or its variants are suitable to work with data mining applications where training data-sets are large.

Two prominent problems with nearest neighbor based classifiers are (i) the space requirement, which is $O(n)$ where n is the number of training patterns, and (ii) its classification time is also $O(n)$. That is, one needs to store the entire training set and search it to classify a given pattern. The space requirement problem can be solved to some extent by prototype selection [10], [11], [9], or by reducing the dimensionality by using feature selection and extraction methods [2], or by using a compact representation of the training data like PC-tree [12], FP-tree [13], CF-tree [14], etc., or by using some editing techniques [7] that reduce the training set size without affecting the performance much. The classification time requirement problem can be solved by finding an index over the training set, like R-Tree index [15].

Two fundamental improvements over k-NNC are, (i) weighted k-nearest neighbor classifier(wk-NNC) [16], where a weight for each training pattern is assigned and is used in the classification, and (ii) generating an artificial training set by applying a bootstrapping method and to use the bootstrapped training set in the place of the original training set. Bootstrap method given by Hamamoto [1] is shown to work well with nearest neighbor based classifiers. Any of these methods can be coupled with any space reduction and or with any of the indexing methods. There exists in Literature another bootstrap method used for k-NNC by Vijaya Saradhi *et al.* [17] where two methods called REST and MREST are given which basically reduces the training set size, but does not improve the performance. Hence this is only a space reduction technique.

978-1-4244-9477-4/11/$26.00 ©2011 IEEE

The present paper proposes a novel and fundamental improvement over k-NNC called k-nearest neighbor mean classifier (k-NNMC). k-NNMC finds k nearest neighbors for each class of training patterns separately, and finds means for each of these k neighbors (class-wise). Classification is done according to the nearest mean pattern. Experimental studies are done using several standard data-sets where it is shown that the proposed classifier performs better than k-NNC, wk-NNC and k-NNC using Hamamoto's bootstrapped training set. Further, the proposed method is suitable for parallel implementations and also can be coupled with any indexing and space reduction methods easily. It is a suitable method to be used in data mining applications.

The rest of the paper is organized as follows. Section II describes Notation and definitions used throughout the paper. Section III describes about NNC, k-NNC and wk-NNC. Section IV deals with the proposed classifier, *viz.,* the k-nearest neighbor mean classifier (k-NNMC). Section V gives some of the empirical results and finally Section VI gives some of the conclusions along with a few future directions of research.

II. Notation and Definitions

1) **Pattern:** A pattern (data instance) is a d dimensional vector of feature values. For example, $X = (x_1, \ldots, x_d)^T$ is a pattern.
2) **Set of Features:** The set of features is $F = (f_1, \ldots, f_d)$. $X[f_i]$ is the value of the feature f_i in the instance X. That is, if $X = (x_1, \ldots, x_d)^T$ then $X[f_i] = x_i$. Domain of values for feature f_l, for $1 \leq l \leq d$, is D_l.
3) **Feature Space:** It is the set of all patterns denoted by $D = D_1 \times D_2 \times \ldots D_d$.
4) **Set of Classes:** $\Omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$ is the set of classes. We say that the class label of a pattern X is ω_i to mean the same thing as X belongs to class ω_i.
5) **Set of Training Patterns (Training Set):** $\mathcal{X} = \{(X_1, l_1), (X_2, l_2), \ldots, (X_n, l_n)\}$ is the set of all training patterns available by using which the classifier can be found. Here, each of X_i, for $1 \leq i \leq n$ is a pattern and l_i is the class to which it belongs. So, $l_i \in \Omega$.
   $\mathcal{X}_j$ is the sub-set of training patterns for which for each pattern its class label is ω_j. Here, $1 \leq j \leq c$. Size of $\mathcal{X}_j$ is n_j.
   $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \ldots \cup \mathcal{X}_c$. Size of $\mathcal{X}$ is n. That is, $n = \sum_{j=1}^{c} n_j$.
6) **Set of Test Patterns (Test Set):** This is an independently drawn data-set from the same source as the training set and is denoted by $\mathcal{Y}$. For each pattern in $\mathcal{Y}$ we know its actual class label.
7) **Distance Function:** Distance measure used is the Euclidean distance. Between two patterns, $X = (x_1, \ldots, x_d)^T$ and $Y = (y_1, \ldots, y_d)^T$ the distance is $\mathit{dist}(X, Y) = \sqrt{\sum_{j=1}^{d}(x_j - y_j)^2}$.
8) **Classifier:** It is a function $f : D \rightarrow \Omega$. This function is found by using the training set.
9) **Loss Function:** For a pattern Y, let its actual class be ω_Y. Then the loss incurred by the classifier for the pattern Y is the function $L_f(Y)$ which is defined below.
   $$L_f(Y) = \begin{cases} 0 & \text{if } \omega_Y = f(X) \\ 1 & \text{if } \omega_Y \neq f(X) \end{cases} \quad (1)$$
   This loss function is traditionally called 0-1 loss function. This is used to measure the performance of the classifier over the test set.
10) **Classification Accuracy (CA):** This is the accuracy of the classifier over the test set. This is normally measured as a percentage. That is, CA is the percentage of patterns in the test set that are correctly classified by the classifier. Which is
   $$\frac{\sum_{\forall Y \in \mathcal{Y}}(1 - L_f(Y))}{|\mathcal{Y}|} \times 100.$$

III. Nearest Neighbors based Classifiers

The nearest neighbor classifier (NNC) assigns a class to the given test pattern which is the class of its nearest neighbor in the training set according to the distance function. Let $Y \in \mathcal{Y}$ be the test pattern. Let $X_N \in \mathcal{X}$ be a pattern such that $\mathit{dist}(X_N, Y) \leq \mathit{dist}(X, Y)$ for all $X \in \mathcal{X}$. Then we say X_N is a nearest neighbor. X_N need not be unique. There can be more than one pattern in the training set which are at equal distance from Y. Let the class of X_N be ω_N. Then the class assigned to the test pattern Y is ω_N. In case, if there are several patterns qualifying to be X_N then a tie occurs. Ties are broken arbitrarily. Let the actual class label of Y which is available with the test set be ω_Y. Then, if $\omega_N = \omega_Y$, we say that the pattern Y is correctly classified, otherwise we say it is wrongly classified.

The k-nearest neighbor classifier (k-NNC) where k is an integer such that $k \geq 1$ is a generalization over NNC. k nearest neighbors for the given test pattern Y are found in the training set. The class information of the each of the k nearest neighbors is preserved. Let the k nearest neighbors along with their class information be $\{(X^1, l^1), (X^2, l^2), \ldots, (X^k, l^k)\}$ where the class to which the training pattern X^1 belongs is l^1, the class to which X^2 belongs is l^2, and so on. Then the class label assigned to Y is according the majority voting among the labels $l^1, l^2, \ldots, l^k$. That is, for example, assume $k = 7$, let the class labels of the 7 nearest neighbors be $\omega_2, \omega_1, \omega_2, \omega_2, \omega_2, \omega_3, \omega_3$. Then, since ω_2 is occurring 4 times, where as ω_3 occurred for 2 times and ω_1 occurred for 1 time, the majority vote winner is ω_2. Hence the class label assigned to the test pattern Y is ω_2. If there are more than two majority vote winners then a tie occurs. Ties are broken arbitrarily.

The weighted k-nearest neighbor classifier (wk-NNC) which is also known as *the modified k-nearest neighbor classifier* [16] is an improvement over k-NNC. This is achieved by giving weights to each of the k nearest neighbors. Let $w^i = (h^k - h^i)/(h^k - h^1)$ be the weight for the i^{th} nearest neighbor for $1 \leq i \leq k$, where h^i is the distance of i^{th} nearest neighbor from the test pattern. That is, $h^i = \mathit{dist}(Y, X^i)$. Total weight for each class is found by summing up the weights of nearest neighbors belonging to that class. If there are no nearest neighbors for a particular class, then the total weight assigned to that class is 0. Now, the test pattern is assigned to the class which has maximum total weight. k-NNC can be seen as a special case of wk-NNC where weights for all k nearest neighbors is equal to 1.

Another improvement found in the Literature is to generate an artificial training set from the given training set. Then k-NNC or NNC uses the artificial training set instead of the originally given training set. This process of generating a new training set from the given training set is often known as *bootstrapping*. A Bootstrapping method given by Hamamoto *et al.,* [1] generates bootstrap samples (artificial samples) by locally combining original training samples. For a given original training pattern, it first finds r nearest neighbors within the class of the given pattern and then finds a weighted average of these neighbors. Let X be a training pattern and let $X^1, \dots, X^r$ be its r nearest neighbors in its class. Then $X' = \sum_{i=1}^{r} w_i X^i$ (where $\sum_{i=1}^{r} w_i = 1$) is the bootstrapped pattern generated for X. Either all patterns in the original set are considered or else a random sample is taken with replacement for generating bootstrapped patterns. The weights w_i are either chosen to be equal for all i (equal to $1/r$) or else randomly assigned a value (such that $\sum_{i=1}^{r} w_i = 1$). So, four different techniques for generating bootstrapped training sets are given. Out of these four methods, it is shown that considering all training patterns by giving equal weight to the r neighbors is best. So, this method is used for our comparative study. Experimentally it is shown that NNC with bootstrap samples outperforms conventional k-NNC in most of the cases.

It is worth noting that there is no theoretical explanation which proves the superiority of wk-NNC or k-NNC which uses the bootstrapped data-set.

IV. k-NEAREST NEIGHBOR MEAN CLASSIFIER (k-NNMC)

This section describes the proposed improvements to k-NNC called *k-nearest neighbor mean classifier(k-NNMC)*. The proposed improvement is orthogonal to the space reduction techniques and classification time reduction techniques. Any of the space reduction, as well as classification time reduction techniques can be applied along with the proposed improvements. Similarly these improvements are independent of the bootstrap methods.

k-NNMC works with each class of training patterns separately. For each class of training patters $\mathcal{X}_i$, for $1 \leq i \leq c$, it finds k nearest neighbors of the test pattern. Let these k nearest neighbors for class ω_i be $\mathcal{X}_i^k = \{X^1, X^2, \dots, X^k\}$. Let the mean (centroid) of these k nearest neighbors be M_i. Like wise, this process is done for each class of training patterns, *i.e., i* is varied from 1 to c. We get $M = \{M_1, M_2, \dots, M_c\}$. If M_j which is in M is at minimum distance from Y, then the class assigned to Y is ω_j. Ties are broken arbitrarily. Algorithm 1 describes this.

Figure 1 shows for a two class problem the 3 nearest neighbors for each of the class. The dimensionality, *i.e.,* $d = 2$. Patterns from class ω_1 are shown with '*', and patterns from ω_2 are shown with 'o'. 'Y' is the test pattern. Observe that NNC will assign class ω_2 to the test pattern. k-NNC with $k = 3$ will assign class ω_1 to the test pattern. wk-NNC will assign class ω_2. k-NNMC will assign class ω_2, since M_2 is closer than M_1.

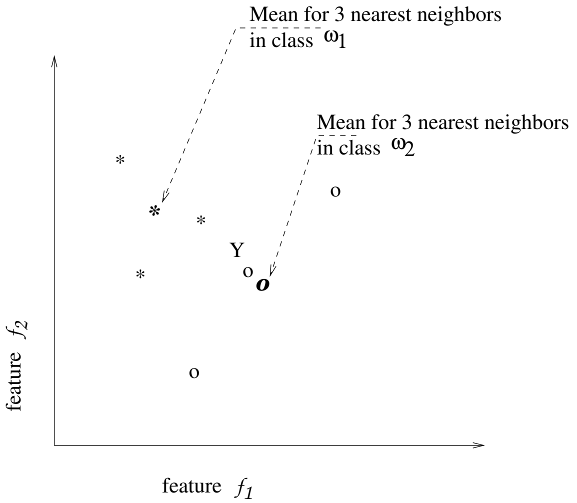
Mean for 3 nearest neighbors in class ω1
Mean for 3 nearest neighbors in class ω2
Y
feature f_2
feature f_1

Fig. 1. For a two class problem showing 3 nearest neighbors for each class

Algorithm 1 can be easily implemented in a parallel form. We require one machine to find k nearest neighbors of Y for each class of training patterns. This can be done in parallel for c classes. Each machine will find mean of its k nearest neighbors and the mean's distance from Y. This distance is communicated and pooled at a machine which will find the nearest among these c distances. Finally the co-ordinating machine will classify Y according to the nearest mean. The time taken for doing classification can be reduced by a factor of c (of course, communication overheads need to be taken in to account). Also index building structures or space reduction techniques when it is limited only to a class of training patterns, it will be efficient. So the proposed method is suitable to work with large training sets and therefore is suitable for data mining applications.

V. EXPERIMENTAL STUDY

This section describes the experimental studies performed.

We performed experiments with five different data-sets, *viz.,* OCR, WINE, THYROID, PENDIGITS and HANDWRITTEN-SYMBOLS. Except the OCR and HANDWRITTEN-SYMBOLS data-sets, all others are from the UCI Machine Learning Repository [18]. OCR data-set is also used in [19], [12], [20]. For OCR, THYROID and PENDIGITS data-sets, the training and test sets are separately available. For the WINE data-set 100 patterns are chosen randomly as the training patterns and the remaining as the test patterns. For HANDWRITTEN-SYMBOLS data-set, sixteen hand-drawn symbols, as shown in Figure 2 are used. Ten different persons are asked to draw each symbol for ten times. So, a total of 1600 hand-drawn symbols database is created. Each symbol's coordinates are collected by drawing it on a 512 × 512 writing pad. 400 randomly chosen symbols (out of 1600 symbols) are separated by doing

Algorithm 1 k-NNMC($\mathcal{X}$, Y)

```
{Note: $Y$ is the test pattern.}
for each class $\omega_i \in \Omega$ do
    Find $\mathcal{X}_i^k$ = $k$ nearest neighbors of $Y$ in $\mathcal{X}_i$.
    {Note: $\mathcal{X}_i$ is the subset of training patterns for which class label is $\omega_i$.}
    Find $M_i$ = mean of patterns in $\mathcal{X}_i^k$.
end for
Find minimum in $\{\text{dist}(Y, M_1), \text{dist}(Y, M_2), \ldots, \text{dist}(Y, M_c)\}$. Let this be $\text{dist}(Y, M_j)$.
Output the classification decision $\omega_j$.
```

Data-set	Number of features	Number of classes	Number of training examples	Number of test examples
OCR	192	10	6670	3333
WINE	13	3	100	78
THYROID	21	3	3772	3428
PENDIGITS	16	10	7494	3498
HANDWRITTEN-SYMBOLS	8	16	1200	400

TABLE I
PROPERTIES OF THE DATA-SETS USED

Data-set	NNC	k-NNC	wk-NNC	k-NNC(HBS)	k-NNMC
OCR	91.12	92.68	93.37	92.88	94.09
WINE	94.87	96.15	97.44	97.44	98.71
THYROID	93.14	94.40	94.81	94.54	95.07
PENDIGITS	96.08	97.54	97.63	97.54	97.88
HANDWRITTEN-SYMBOLS	78.20	87.80	88.40	86.50	88.90

TABLE II
A COMPARISON BETWEEN THE CLASSIFIERS (SHOWING CA (%))

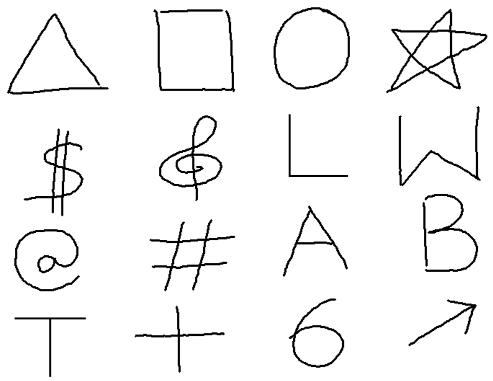

Fig. 2. Handwritten symbols used for experimental study.

sampling without replacement, and is used as the test set. Remaining 1200 symbols constitutes the training set. Zernike moments [21] of order 8 are used to represent each symbol after doing normalization against scale and translation as described in [22].

The properties of the data-sets are given in Table I.

All the data-sets have only numeric valued features. The OCR data-set has binary discrete features, while the others have continuous valued features. Except OCR and HANDWRITTEN-SYMBOLS data-sets all other data-sets are normalized to have zero mean and unit variance for each feature.

Table 2 compares classification accuracies obtained for various classifiers. Classifiers compared are NNC, k-NNC, wk-NNC, k-NNC using Hamamoto's bootstrapping (k-NNC(HBS)) and the proposed classifier k-NNMC. The parameters like k value and r value (used in Hamamoto's bootstrapping) are found by employing a 3-fold cross validation [2].

From the presented results, it is observed that the proposed k-NNMC performs consistently better than the other related classifiers.

VI. CONCLUSIONS AND FUTURE WORK

The paper presented a novel fundamental improvement to the well known k-nearest neighbor classifier. It finds k nearest neighbors for each class of training patterns separately and then finds mean of these k nearest neighbors class-wise. The class assigned is that for which the distance of the mean pattern is minimum with the test pattern. The proposed method can be easily implemented in parallel form and can be easily integrated with any space reduction or indexing structure.

Future work is to give a theoretical explanation which formally proves why the proposed method is working well. Error can be divided in to bias and variance. An analysis which compares bias and variance for each classifier can give a better insight in understanding why a particular classifier is doing well when compared with other related classifiers.

ACKNOWLEDGMENTS

This work is supported by an AICTE Project under RPS Scheme with reference: "F.No: 8023/BOR/RID/RPS-51/2009-10".

REFERENCES

[1] Y. Hamamoto, S. Uchimura, and S. Tomita, "A bootstrap technique for nearest neighbor classifier design," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 1, pp. 73–79, 1997.

[2] R. O. Duda, P. E.Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. John Wiley & Sons: A Wiley-interscience Publication, 2000.

[3] D. H. Wolpert, "The relationship between PAC, the statistical physical framework, the bayesian framework, and the VC framework," in *The Mathematics of Generalization*, D. H. Wolpert, Ed. MA: Addison-Wesley, 1995, pp. 117–214.

[4] E. Fix and J. Hodges, Jr., *Discriminatory Analysis: Non-parametric Discrimination: Consistency Properties*, Report No. 4, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

[5] E. Fix and J.L. Hodges, Jr., *Discriminatory Analysis: Non-parametric Discrimination: Small Sample Performance*, Report No. 11, USAF School of Aviation Medicine, Randolph Field, Texas, 1952.

[6] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[7] B. V. Dasarathy, "Nearest neighbor (NN) norms: NN pattern classification techniques," B. V. Dasarathy, Ed. Los Alamitos, California: IEEE Computer Society Press, 1991.

[8] Belur V. Dasarathy, "Data mining tasks and methods: Classification: Nearest-neighbor approaches," in *Handbook of data mining and knowledge discovery*. New York: Oxford University Press, 2002, pp. 288–298.

[9] V. S. Babu and P. Viswanath, "Rough-fuzzy weighted k-nearest leader classifier for large data sets," *Pattern Recognition*, vol. 42, no. 2009, pp. 1719–1731, 2009.

[10] Q. Xie, C. Laszlo, and R. Ward, "Vector quantization technique for nonparametric classifier design," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 12, pp. 1326–1330, 1993.

[11] V. Susheela Devi, *Optimal Prototype Selection for Efficient Pattern Classifiers*, Ph.D Thesis, Department of Electrical Engineering, IISc, Bangalore, 2000.

[12] V. Ananthanarayana, M. Murty, and D. Subramanian, "An incremental data mining algorithm for compact realization of prototypes," *Pattern Recognition*, vol. 34, pp. 2249–2251, 2001.

[13] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *Proceedings of ACM SIGMOD International Conference of Management of Data*, Dallas, Texas, USA, 2000.

[14] Z. Tian, R. Raghu, and L. Micon, "BIRCH: an efficient data clustering method for very large databases," in *Proceedings of ACM SIGMOD International Conference of Management of Data*, 1996, pp. 103–114.

[15] A. Guttman, "R-trees: a dynamic index structure for spatial searching," in *13th ACM SIGMOD Int. Conf. Management Data*, vol. 2, Boston, MA, 1984, pp. 47–57.

[16] S. Dudani, "The distance-weighted k-nearest-neighbor rule," *IEEE Transactions on SMC*, vol. SMC-6, no. 4, pp. 325–327, 1976.

[17] V. Vijaya Saradhi and M. N. Murty, "Bootstrapping for efficient handwritten digit recognition," *Pattern Recognition*, vol. 34, pp. 1047–56, 2001.

[18] P.M.Murphy, *UCI Repository of Machine Learning Databases [http://www.ics.uci.edu/mlearn/MLRepository.html]*, Department of Information and Computer Science, University of California, Irvine, CA, 2000.

[19] T. R. Babu and M. N. Murty, "Comparison of genetic algorithms based prototype selection schemes," *Pattern Recognition*, vol. 34, pp. 523–525, 2001.

[20] P.Viswanath, M. Murty, and S. Bhatnagar, "Partition based pattern synthesis technique with efficient algorithms for nearest neighbor classification," *Pattern Recognition Letters*, vol. 27, no. 2006, pp. 1714–1724, 2006.

[21] J. Flusser, T. Suk, and B. Zitova, *Moments and Moment Invariants in Pattern Recognition*. UK: John Wiley & Sons, 2009.

[22] H. Hse and A. R. Newton, "Sketched symbol recognition using Zernike moments," in *Proceedings of ICPR-2004*, vol. 1. Los Alamitos, CA, USA: IEEE Computer Society, 2004, pp. 367–370.