Fast Parzen Density Estimation Using Clustering-Based Branch and Bound

Byeungwoo Jeon and David A. Landgrebe

School of Electrical Engineering

Purdue University, W. Lafayette, IN 47907, U.S.A.

Telephone: (317) 494-3486, FAX : (317) 494-3358

landgreb@ecn.purdue.edu

FAST PARZEN DENSITY ESTIMATION USING CLUSTERING-BASED BRANCH AND BOUND[1]

ABSTRACTS

This correspondence proposes a fast Parzen density estimation algorithm which would be specially useful in the non-parametric discriminant analysis problems. By pre-clustering the data and applying a simple branch and bound procedure to the clusters, significant numbers of data samples which would contribute little to the density estimate can be excluded without detriment to actual evaluation via the kernel functions. This technique is especially helpful in the multivariant case, and does not require a uniform sampling grid. The proposed algorithm may also be used in conjunction with the data reduction technique of Fukunaga and Hayes [4] to further reduce the computational load. Experimental results are presented to verify the effectiveness of this algorithm.

Key Words: Parzen density estimation, branch and bound, pre-clustering, non-parametric discriminant analysis

---

[1] This work was supported in part by NASA under Grant NAGW-925

Submitted August 21, 1992, Revised April 8, 1993

I. INTRODUCTION

Applying statistical pattern recognition techniques often requires an estimation of probability density functions of data samples. If the distribution of the data is known to follow a certain form with a few parameters, such as that of the Gaussian distribution, then the probability density can be easily evaluated using the estimated parameters of the distribution function. However, it is not always possible to assume a density function in a parametric form without causing significant error. In this case, a non-parametric approach must be taken by employing density estimation techniques [1]. Density estimation is usually very computationally intensive, and because of that, it may be less attractive in many applications, e.g., especially for an on-line application. Therefore, it is often desirable to lessen the computation load in estimating densities.

A previous approach to reducing the computational requirement of density estimation is based on the *k* -nearest neighbor method [2,3]. It functions by saving the number of distance evaluations for finding the k nearest neighbors. As for kernel-based density estimates, such as the Parzen density estimate [1], by noting that the amount of computation is directly related to the number of training samples, Fukunaga and Hayes [4] extracted a representative subset of the training samples to achieve computational saving. The reduced subset of training samples was selected in such a way that the Parzen density estimate with the reduced set matches as closely as possible with that of the full data set in the sense of an entropy measure of similarity between the two estimates.

Silverman [5] proposed an efficient algorithm based on the fast Fourier transform (FFT) for evaluating univariate Parzen density estimates on regular grids. In the case of plotting a density estimate, for example, the estimate would be evaluated over equally spaced locations. Silverman noted that the Fourier transform of the density estimate can be considered as a product of the Fourier transforms of the kernel function and the data. A modified discretization scheme was employed later by Jones and Lotwick [6] to reduce errors related to the Fourier transform of data. Notice that this fast algorithm based on FFT can not be applied to the general cases of density

estimates over irregularly spaced locations. For instance, suppose Parzen density estimates are used for non-parametric classification [7], then the density values must be estimated for the samples to be classified, and in general, the samples are not regularly spaced.

In a manner similar to the efficient density estimate based on the *k*-nearest neighbors [3], in this correspondence, a simple branch-and-bound procedure is applied to the Parzen density estimation to reduce the number of kernel evaluations. Note that the contribution of a training (or design) sample on the density estimate being evaluated rapidly diminishes if it is far away from the location of evaluation. Therefore, without causing significant error, some of the training samples could be left out in evaluating kernel functions if the distances from the location of evaluation to those samples are large enough to result in negligible kernel function values. Once a kernel function for the Parzen density estimate and the maximum allowable error produced by leaving out distant samples are determined, a certain critical distance or threshold can be found. The computation for evaluating the kernel function for training samples (*i.e.*, distances of the samples to the location of evaluation) is significantly saved by applying the branch-and-bound procedure to the pre-clustered training data.

Experimental results are presented to show the effectiveness of the proposed approach in reducing the computation of Parzen density estimation. Notice that to further reduce the computational burden, this proposed algorithm also can be used in addition to the data reduction algorithm in [4].

II. FAST PARZEN DENSITY ESTIMATION

Suppose the set **Y** consists of N q-dimensional ($q \geq 1$) samples with which to estimate a probability density function. The Parzen density estimate $\hat{f}_x(x)$ of the unknown probability density function at x, x R^q, may be obtained as a sum of kernel functions placed at each sample y in **Y** as [1],

$$\hat{f}_X(x) = \frac{1}{N h^q} \sum_{y \in \mathbf{Y}} K\left(\frac{x - y}{h}\right) \qquad (1.a)$$

where K(•) is a selected kernel function and h is the smoothing parameter (or, window size). The kernel function satisfies the following condition,

$$\int_{x \in R^q} K(x)\, dx = 1 \qquad (1.b)$$

Since the estimate $\hat{f}_X(x)$ will inherit properties of the selected kernel function, the kernel function is often chosen in such a way that it has mathematically tractable properties, such as continuity or differentiability. Some examples [1] include the Gaussian kernel function, the Epanechnikov kernel function, or a rectangular kernel function. The value of the kernel function rapidly decreases as the distance from the origin increases. Therefore, the contribution of a sample $y \in \mathbf{Y}$ to the density estimate at x, $\hat{f}_X(x)$, will become negligible if the distance between x and y becomes large. In many situations, it is possible to select a "critical distance," D_c so that, without introducing significant error, the contribution of a sample $y \in \mathbf{Y}$ in eq. (1.a) can be assumed to be zero if the distance between x and y is more than D_c. This is equivalent to employing a truncated and rescaled version of the original kernel function for density estimation. Suppose the truncation level is denoted by β, $0 \le \beta \le 1$, then, the critical distance D_c for the window size h = 1, can be found by solving,

$$\beta = \int_{x^T x \le D_c^2} K(x)\, dx \qquad (2.a)$$

The truncated kernel function with truncation level β is denoted by $K'(x; \beta)$ and defined as,

$$K'(x;\beta) = \frac{K(x)}{\beta} \qquad x^T x \le D_c^2 \qquad (2.b)$$

$$= 0 \qquad \text{otherwise}$$

Note that even though there is no truncation (*i.e.*, with $\beta = 1$), due to the finite numerical precision of computation, there is always an implicit truncation determined by the degree of precision of the computer being used. Depending on the specific application and the degree of permissible trade-off between accuracy and speed, an appropriate value for β, $0 \le \beta \le 1$, in eq. (2.a) can be selected. Some kernel functions such as the Epanechnikov kernel function [1] or the rectangular kernel function have finite support only where the kernel functions have non-zero values. In these cases, it is straightforward to select a value D_c without losing any accuracy, and there is no need for truncation and normalization. The critical distance with window size h is then obtained by multiplying h with the D_c calculated in eq. (2.a).

Note that kernel functions can be written as functions of an appropriate distance measure, and the evaluation of the distances are often time-consuming since they are usually quadratic. Denote the distance between two samples, x and y as L(x,y) where L(•) is an appropriate distance measure, e.g., the Euclidean distance measure, defined as,

$$L(x, y) = \sqrt{(x-y)^T (x-y)} \qquad (3.a)$$

If different smoothing parameters should be used in different coordinate directions, then, the following distance measure can be used with the kernel covariance matrix Σ,

$$L(x, y) = \sqrt{(x-y)^T \Sigma^{-1} (x-y)} \qquad (3.b)$$

Note that this distance measure in eq. (3.b) is equivalent to using the Euclidean distance measure in eq. (3.a) after an linear transformation with $\Sigma^{-0.5}$ applied to the samples x's and y's [7]. Assume an appropriate linear transformation has been already been applied to the set **Y** and

sample x's as required to deal with the different smoothing parameters. For simplicity, only the Euclidean distance measure will be used in the subsequent discussion, without loss of generality.

Suppose the Parzen density estimate is evaluated at x with the truncated kernel function in eq. (2.b). Then the sample y's in **Y** which do not satisfy,

$$L(x, y) < D_c \tag{4}$$

can be excluded in the summation of eq. (1.a). The number of evaluations of distance L(x, y) in eq. (4) could be significantly reduced if the branch and bound procedure [3] is applied to the pre-clustered samples in **Y**; *i.e.*, after grouping y samples which are adjacent in R^q space into clusters, the distances from x to the clusters can be checked as described below to find any clusters whose members are all distant from x by more than D_c. The samples grouped to those distant clusters can be omitted entirely from the summation in eq. (1.a).

To pre-cluster the samples in **Y** into groups, unsupervised clustering [7] is performed with the Euclidean distance measure in eq. (3.a). Note that an appropriate linear transformation with $\Sigma^{-0.5}$ is assumed to have been performed as necessary to the data in the set **Y**. Employing the Euclidean distance measure after the linear transformation is equivalent to using the Mahalanobis distance measure [9] with the same covariance matrix Σ for all clusters.

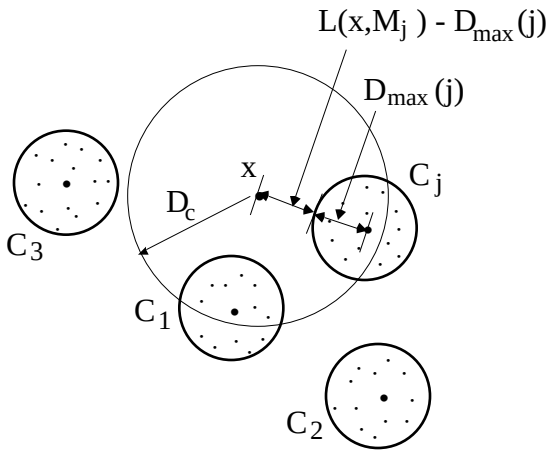

Figure 1. Efficient computation of Parzen density estimation using clustering. Samples grouped into clusters other than C_1 and C_j in this figure need not be considered in the computation of the Parzen density estimate.

To each cluster, for example, to the j^{th} cluster, C_j, three variables, $\{I_j, M_j, D_{max}(j)\}$ are associated. M_j is the cluster mean and I_j is the index set of cluster C_j where i I_j only if y_i belongs to the cluster C_j, y_i **Y**. $D_{max}(j)$ denotes the maximum distance from the cluster mean, M_j to the samples in cluster C_j as,

$$D_{max}(j) \equiv \max_{i \in I_j} \left\{ L(x_i, M_j) \right\}$$

Since the distance from x to any sample in C_j should be larger than $L(x, M_j) - D_{max}(j)$, all the samples belonging to cluster C_j which do not satisfy the inequality,

$$L(x, M_j) - D_{max}(j) < D_c \qquad (5)$$

can be excluded in evaluating the density estimate at x as shown in Figure 1. Thus, the calculation of distances from x to each sample in **Y** can be significantly reduced by checking the inequality in eq. (5) and deleting clusters as a whole appropriately. This same idea can also be applied to reduce the number of clusters which should be checked for the inequality in eq. (5) by creating a hierarchical grouping of the clusters, but we will not elaborate on this here.

The unsupervised grouping of the samples in **Y**, is not sensitive to the particular clustering algorithm chosen; basically any available clustering algorithm [7], such as the well-known ISODATA procedure [8], can be used. In selecting a clustering method, however, it is important to remember that clustering itself can be time-consuming unless carefully applied, and its objective is to group samples in **Y** into distinct clusters to apply the branch-and-bound method. There is not a strict requirement to come up with clusters which are optimal or near optimal in any sense, and only a few iterations should satisfy the need for grouping samples.

In clustering, the creation (and/or deletion), or merging of clusters can be cumbersome. A very simple algorithm is considered in this correspondence as follows. A new cluster is created

whenever the minimum distance to the existing clusters exceeds a certain user-defined value, denoted by $\sqrt{q}T_{create}$, otherwise the sample under consideration is assigned to the nearest cluster. After one or two iterations, the creation option can be disabled so that every sample is assigned to one of the existing clusters. For computational reasons, the merging of clusters is not allowed.

The computation required for the pre-clustering may not be trivial, but it is required only once for the (training) data set **Y**. If the number of location's x on which the probability density $\hat{f}_x(x)$ is to be computed is large, then this one-time extra computation for clustering should be worthwhile. When the probability density in eq. (1.a) is evaluated, there exists another extra computation required for the distances from x to each cluster center. However, considering the savings due to skipping a subset of distant samples in **Y**, this will be quite negligible unless the number of clusters is comparable to the number of total samples in **Y**. Suppose there are M clusters for **Y**, and each cluster has n samples on the average (therefore, N = n M). Through the branch-and-bound procedure with eq. (5), suppose only the sample y's in m, $1 \le m \le M$, clusters are selected on the average, to be included in the summation in eq. (1.a); then, one needs M (for distances to the clusters) plus n m (for distances to the samples in the m selected clusters) distance computation. Therefore, the ratio of saving in computing distances over the regular Parzen density estimate is,

$$\frac{M + n\,m}{N} = \frac{1}{n} + \frac{m}{M}$$

The variable n and M are dependent on T_{create}, and m is dependent on both T_{create} and D_c. To achieve a maximal efficiency in reducing the computational load of clustering and speeding the Parzen density estimate, care must be exercised in selecting a proper value of T_{create}. Too small a value of T_{create} would result in a large number of small clusters with only a few members in each; in this case the overhead of clustering and checking the inequality in eq. (5) could surpass the savings obtained by skipping the samples grouped to distant clusters. On the other hand, a few clusters with large number of samples in each of them due to using too large a value of

T_{create} might not result in much deletion of clusters in evaluating the density estimate. The value of T_{create} thus must be selected in relation to the critical distance D_c.

III. EXPERIMENTS AND DISCUSSION

To explore the effectiveness of the proposed fast Parzen density estimation algorithm, an experiment with simulated data was performed. For the (training) data set **Y**, 1000 samples of bivariate (q = 2) Gaussian data were generated. The mean and covariance matrix were set to $[0, 0]^T$ and to the identity matrix, respectively. The Parzen density estimate was evaluated at four different groups of locations. That is, 4 sets of bivariate Gaussian samples, each containing 100 samples, were generated with the means at $[\pm 1.5, 0]^T$ and $[0, \pm 1.5]^T$. All covariance matrices were set to the identity matrix.

In pre-clustering the data samples in **Y**, if the *squared* distance to the nearest cluster was more than qT_{create}^2, then, a new cluster was generated, otherwise the sample under consideration was assigned to the nearest cluster. Therefore, the maximum distance $D_{max}(\bullet)$ in eq. (5) was $\sqrt{q}T_{create}$. To see the effect on the efficiency of this algorithm, the parameter for a new cluster generation, T_{create}, was selected as,

$$T_{create} = \alpha D_c \tag{6}$$

and the value α was varied to see its effect on the effectiveness of the proposed algorithm. The effectiveness of this algorithm can be measured in terms of the percent of distance computations actually evaluated during density estimation, *i.e.*,

$$R \equiv 100 \times \frac{\text{average number of distance computation}}{\text{number of training samples}} \tag{7}$$

In the numerator in eq. (7), the number of distance computations to the cluster centers was also included even though it might be negligible in most of the cases. The average was calculated by

dividing the total number of distance computations with that of the density estimate evaluations. In the case of conventional Parzen density estimation, R in eq. (7) is 100. If the overhead of computing distances to the cluster centers surpasses the savings acquired by deleting some of the distant clusters, R can be greater than 100.

The Epanechnikov kernel function [1] was considered in the first experiment since it is straightforward to choose the critical distance D_c, which is equal to the window size h. As suggested in ([1] p.86), the window size h was set to 0.56. Under this setting, only 4.24 % of the training samples on average actually contributed non-zero kernel function values in the density estimation. The value of α in eq. (6) was varied from 0.01 to 8 to see its effect on deleting the distant clusters. Only one iteration of clustering was performed since a crude grouping of the samples was sufficient.

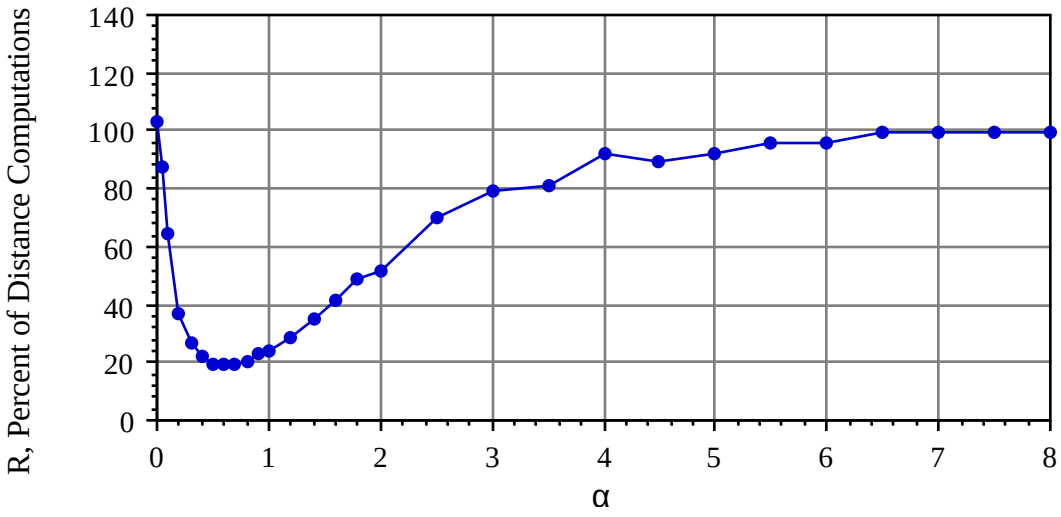

Figure 2. R, the average percent of distance computations, as in eq. (7), using the Epanechnikov kernel function with cluster creation conditions as in eq. (6), where α is varied from 0.01 to 8.; window size h = 0.56, critical distance D_c = 0.56.

As α in eq. (6) decreased (i.e., as the number of clusters increases), the savings in distance computation increased up to a certain point, after which the overhead of distance computation to the cluster centers overwhelmed the savings attained by skipping some of training samples as seen in Figure 2. Unless α is extremely small (unless $\alpha < 0.02$ in this experiment), the overhead

was negligible. A savings of about 80 % was observed in distance computation with $0.5 < \alpha < 1.0$.

The same experiment was performed with the Gaussian kernel function, which had non-zero values over the entire feature space. The window size was set to h=0.304 as suggested in ([1] p.86). Truncation was performed as in eq. (2.a,b) with a truncation level β which was varied from 0.8 to 0.999. Note that even if there is no truncation, there are some training samples which would not make any contribution to the density estimate because of the numerically finite precision. That is, the value of the exponential function in the Gaussian kernel becomes (numerical) zero when its argument is too small. In this correspondence, the *implicit* truncation due to the numerical finite precision is considered as the case with $\beta = 1.0$.

There must be error introduced due to the truncation of the kernel function, and the amount of error can be measured by the average percent difference between the two density estimates obtained with and without truncation as,

$$\text{Average percent difference} = 100 \times E_X \frac{\left| \hat{f}_X(x; \beta) - \hat{f}_X(x) \right|}{\hat{f}_X(x)} \qquad (8)$$

where $\hat{f}_X(x)$ denotes the density estimate without truncation and $\hat{f}_X(x; \beta)$ denotes the density estimate with the truncation level β. The expectation in eq. (8) is obtained by computing the average over the given 400 density estimate evaluation locations.

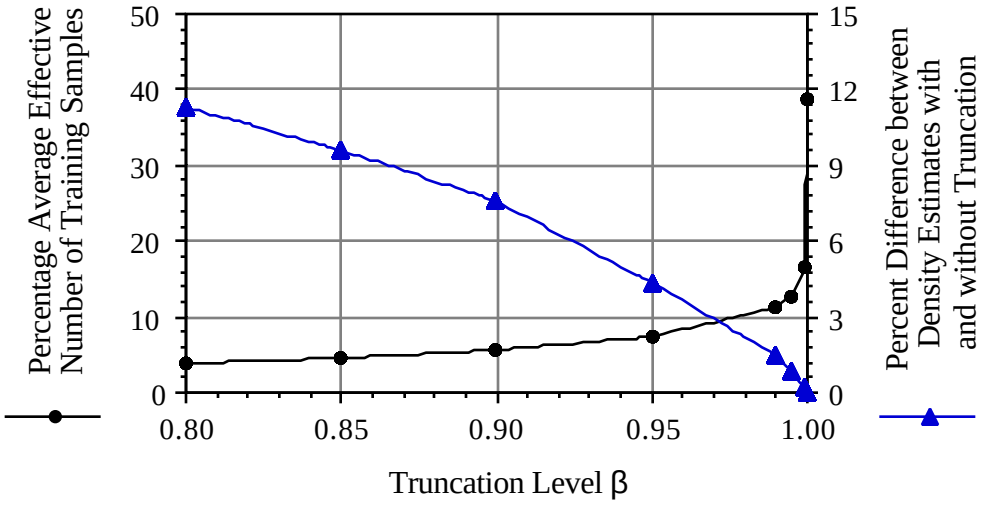

Figure 3. Percent average effective number of training samples which have non-zero contribution to the density estimate and the corresponding average percent difference between density estimates obtained with and without truncation.. A truncated Gaussian kernel function is used with different truncation level β's.; the window size is set to h = 0.304.

Figure 3 shows the average number of effective training samples which give non-zero values for the exponential function when the truncated Gaussian kernel function when truncation level β is used. As described before, the number without truncation was considered to be β = 1.0. As seen in Figure 3, even when β = 1.0, only about 38% of the training samples contributed non-zero kernel function values to the density estimate due to the numerical finite precision. When β = 0.999, the effective number of training samples dropped to 16%, but there was only 0.19 % of a percent difference on the average between $\hat{f}_x(x)$ and $\hat{f}_x(x; \beta)$. If β = 0.99, the percent difference was 1.47% with 11% of effective training samples. Whether or not this error due to the truncation is acceptable depends on the particular application of the estimated density in mind.

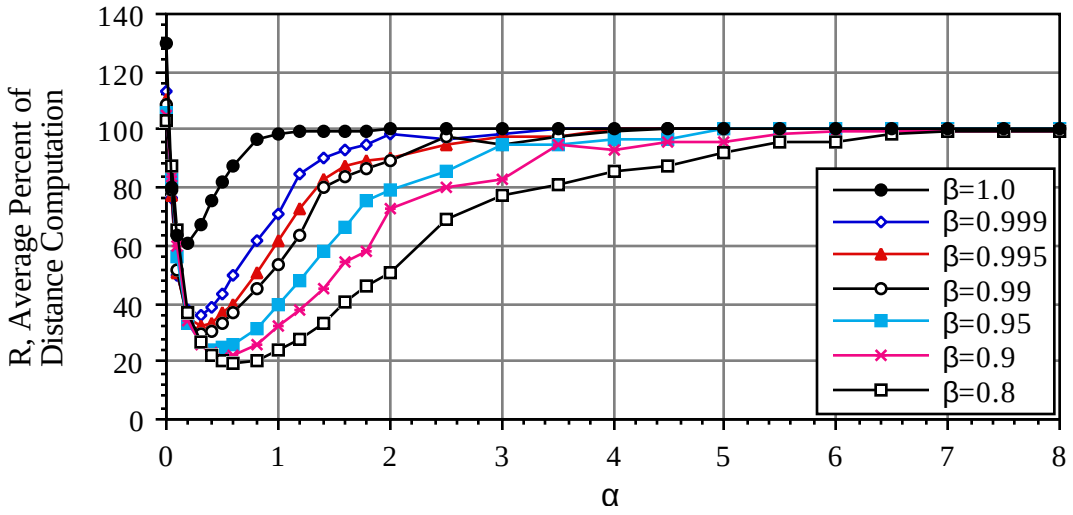

Figure 4. R, the average percent of distance computations, as in eq. (7), with a truncated Gaussian kernel function and truncation level β. The parameter α in the cluster creation condition of eq. (6) is varied from 0.01 to 8.; window size h = 0.304.

As before, when the parameter α in eq. (6) is varied from 0.01 to 8, the average number of actual distance computations is presented in Figure 4. As expected from Figure 3, as the truncation level β became larger, the amount of savings in distance computation increases. Note that, with b=0.99, which results in only about 1.47% change of the density estimate from the case without truncation, as much as about 70% of the distance computations can be saved, as seen in Figure 4. As seen before with the Epanechnikov kernel function, extremely small or large values of α were not acceptable since they produced too many small clusters or just only a few large clusters. With α in the range of 0.2 ~ 1.0, it was observed that about 40 ~ 80 % savings in distance computation could be achieved.

IV. CONCLUSION

In this correspondence, a computationally efficient Parzen density estimation algorithm is developed by utilizing a simple branch and bound procedure applied to the pre-clustered (training) data samples. Not only those kernel functions having finite support for non-zero

values, such as the Epanechnikov kernel function, but also those kernel functions having non-zero values over the entire feature space were applicable to this algorithm through truncation. By choosing the proper parameter value for cluster generation, substantial savings in computation could be realized. Values which were found to be satisfactory were those close to the critical distance D_c. Experimental results verified that savings were significant. To further enhance the computational efficiency, this proposed algorithm can be used in conjunction with the data reduction technique [4].

REFERENCES

[1] B. W. Silverman, Density Estimation for Statistics and Data Analysis, Chapman and Hall, 1986.

[2] P. E. Hart, "The Condensed Nearest Neighbor Rule," IEEE Trans. Inform. Theory, IT-14, pp.515-516, 1968.

[3] K. Fukunaga and P. M. Narendra, "A Branch And Bound Algorithm For Computing K-Nearest Neighbors," IEEE Trans. on IEEE Computers, C-24, pp. 750-753, 1975.

[4] K. Fukunaga and R. R. Hayes, "The Reduced Parzen Classifier," IEEE Trans. on Pattern Anal. and Machine Intell., PAMI-11, pp.423-425, 1989.

[5] B. W. Silverman, "Kernel Density Estimation Using The Fast Fourier Transform," Statistical Algorithm, AS176, Appl. Statist. 31, pp.93-97, 1982.

[6] M. C. Jones and H. W. Lotwick, "A Remark on Algorithm AS 176. Kernel Density Estimation Using The Fast Fourier Transform," Remark, AS R50, Appl. Statist. 33, pp.120-122, 1984.

[7] K. Fukunaga, Introduction to Statistical Pattern Recognition, 2nd edition, Academic Press, New York, 1990.

[8] G. H. Ball and D. J. Hall, "Isodata, A Novel Method of Data Analysis and Pattern Classification," Stanford Research Institute Technical Report, (NTIS AD 699616), Stanford, CA, 1965.

[9] C. W. Therrien, Decision, Estimation, And Classification ; An Introduction To Pattern Recognition And Related Topics, John Wiley & Sons, New York, 1989.