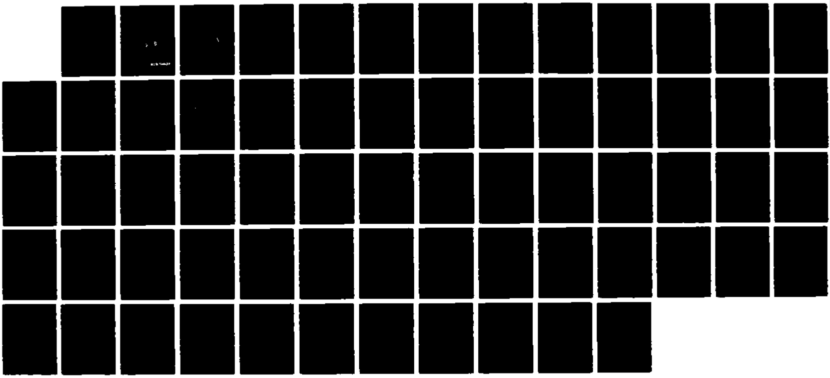
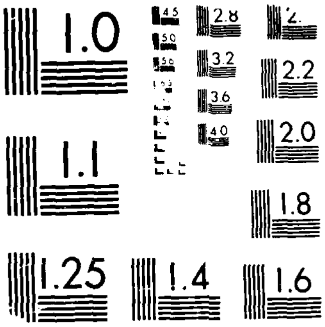
AD-A193 069 CHARACTERIZATION OF ALGORITHMS FOR PASSIVE INFRARED 1/1
(CAPIR)(U) NORTHROP RESEARCH AND TECHNOLOGY CENTER
PALOS VERDES PENINSULA CA 01 FEB 87 DAAL02-85-C-0151
UNCLASSIFIED F/G 17/5.1 NL





MICROCOPY RESOLUTION TEST CHART

[illegible] BUREAU OF STANDARDS 1963-A

CAPIR FINAL REPORT
Contract: DAALO2-85-C-0151

Northrop Research and Technology Center
One Research Park
Palos Verdes Peninsula, CA 90274

NORTHROP
Research and Technology Center

1

CAPIR FINAL REPORT
Contract: DAALO2-85-C-0151

Northrop Research and Technology Center
One Research Park
Palos Verdes Peninsula, CA 90274

CAPIR FINAL REPORT
Contract: DAAL02-85-C-0151

Northrop Research and Technology Center
One Research Park
Palos Verdes Peninsula, CA 90274

February 1, 1987

/

Contents

1 Introduction

This is the final report for the "Investigation of Classifier Algorithms" project of the DARPA/NVEOL Characterization of Algorithms for Passive Infrared (CAPIR) program. This report covers all activity performed by the Northrop Research and Technology Center during the period from 17 October 1985 to 17 October 1986.

The overall objective of this effort is to develop and study an optimal classifier so that future ATR designers can concentrate on feature extraction. The specific tasks of this project are:

A. Perform feature extraction and analysis on ERIM-supplied segmentor outputs.

B. Develop probability density estimates and figure-of-merit computations for kernel-based classifier.

C. Develop estimates for confidence in the probability density estimates.

D. Evaluate and demonstrate classifier using above items.

This report is divided into nine sections. Section 2 details the feature extraction task and includes information about the exchange of data with ERIM. Section 3 describes the classifiers used in this study as well as the window-width optimization problem in producing density estimates with the kernel-based classifier. Section 4 describes the confidence measures used to attack task C. Section 5 presents the results of our study on the development set of images, while Section 6 reports results for the characterization set of images and comparisons of these results to the development set results. Section 7 gives the results of our window-width optimization study while Section 8 presents our confidence measure experiments. Finally, Section 9 gives the conclusions of this study.

2 Feature Extraction

In our original contract statement, this feature extraction task was not included. After some discussion with NVEOL personnel, it was decided to perform the extraction of nine simple shape-based features on the ERIM-supplied database. Such a simple set of features should be adequate for the job of evaluating various classifiers.

2.1 Data Exchange

Software was developed to read the ATRWG-style tape formats. The ERIM development set was read from tape and placed on disk. The total number of development set images read onto disk was 136.

For each image file on the three magnetic tapes received from ERIM, three disk files were created. One file consists of the grey-level image, the second contains the binary mask image, and the last contains the ATRWG header1 and header2 information.

Software was written to decode the header2 portion of the ATRWG information. Currently, this software supports "list-format" header2. Included with this is the ability to decode both the ground-truth list and the target-metric list. Other list types, e.g. image-truth, have not been decoded.

The results of classifier output for the development set were sent to ERIM.

The characterization set of images was received from ERIM. It included a set of five (5) image tapes and one experiment definition tape. There was no classifier experiment defined on the tapes. NVEOL suggested using all of the image files present in the development set in a single experiment. The initial characterization set included no segmentor output. This problem was resolved.

The results of using the classifiers trained on the development set to classify object in the characterization set were sent to ERIM.

2.2 Feature Definitions

The nine features suggested for use by ERIM are shape-based features. Their definitions are as follows. For any object in any image, let R be the set of (x, y) coordinate pairs in the bounding rectangle defined in the ERIM supplied ground-truth data. Let $P(i, j)$ be the (i, j)th pixel in the segmented binary (0 or 1) mask image. The nine features are defined as follows:

- $Feature(1) = area = \sum_{(i,j)\in R} P(i,j)$
- $Feature(2) = perimeter$ where the perimeter is estimated by the following procedure. First, let $perimeter := 0$. Then for every $(i,j) \in R$, first let $s_x = P(i-1,j)+P(i,j)+P(i+1,j)$ and $s_y = P(i,j-1)+P(i,j)+P(i,j+1)$, then if $s_x = 1$ and $s_y = 0$, set $perimeter := perimeter + 1$ or if $s_x = 0$ and

$s_y = 1$, set *perimeter* := *perimeter*+1 or if $s_x = 1$ and $s_y = 1$ and $P(i,j) = 0$, set *perimeter* := *perimeter* + $\sqrt{2}$.

- *Feature*(3) = *ratio of the eigenvalues* = λ_1/λ_2. To define this, and the following features, let

$$avg_x = \sum P(i,j)i/\sum P(i,j)$$

$$avg_y = \sum P(i,j)j/\sum P(i,j)$$

$$u_{11} = \sum P(i,j)(i - avg_x)(j - avg_y)$$

$$u_{20} = \sum P(i,j)(i - avg_x)^2$$

$$u_{02} = \sum P(i,j)(j - avg_y)^2$$

$$b = u_{20} + u_{02}$$

$$c = u_{20}u_{02} - u_{11}^2$$

$$\lambda_1 = \sqrt{(b + \sqrt{b^2 - 4c})/2}$$

$$\lambda_2 = \sqrt{(b - \sqrt{b^2 - 4c})/2}$$

  where each sum is over $(i,j) \in R$.

- *Feature*(4) = *Feature*(2)^2/*Feature*(1).
- *Feature*(5) = $|u_{11}/b|$.
- *Feature*(6) = $-c/b^2$.
- *Feature*(7) = $(u_{20} - u_{02})/b$.
- *Feature*(8) = $(\sqrt{u_{20}} + \sqrt{u_{02}} - 1)$*Feature*(1)/*Feature*(2).
- *Feature*(9) = *Feature*(1)/($\lambda_1\lambda_2$ − *Feature*(1)).

Throughout the rest of this report, we will refer to features by the above numbers.

It is important to note here that the segmentor output provided by ERIM is the result of segmentation by hand rather than by any particular segmentor algorithm. Thus, the segmented objects on which the above features are computed are *ideal*.

2.3 Development Set Pattern Matrix

A pattern matrix consisting of the nine features for each segmented object in the set of images was computed. A total of 378 segmented objects were found and processed. Class information contained in the header2 target metric field "Target Type Observed" (T-18) is also retained.

A number of inconsistencies between the ground-truth data and the target-metric data on the number of objects in the field-of-view were found. In all cases, the target-metric data was used. Also, one file (2029075z) had a target-metric specification for two targets. However, only the first appeared valid, therefore, the second target was ignored. No additional problems were encountered.

2.3.1 Fourteen-Class Problem

We have been tasked with two classification task: the "detection problem", which refers to classifying the patterns into one of two classes, and the "target identification problem", which refers to classifying the patterns into one of the *a priori* target types, e.g. M-60 tank, M-133 tank, etc.

For the multi-class target identification problem, we use the original class labels from the ATRWG header. In the development set, there were fourteen such classes. Table 1 shows the breakdown of the class labels on the patterns that are present in the fourteen-class pattern matrix.

In the remainder of this report, we will refer to classes by their class number.

2.3.2 Two-Class Problem

The "detection" problem is set up as the problem of classifying tracked vehicles versus non-tracked vehicles. A pattern matrix was built with all of the original 378 patterns partitioned into these two pattern classes. Tracked vehicles consist of initial target types in the ranges (10-25) and (200-219) while non-tracked vehicles consist of target types of (100-108) and (300-316). This resulted in a split of 182 patterns in class 1 (tracked) versus 196 patterns in class 2 (wheeled).

Class Number	Object Label	Number of Patterns
1	11	19
2	17	30
3	24	27
4	100	24
5	104	21
6	200	14
7	204	46
8	208	9
9	218	37
10	300	34
11	302	23
12	306	31
13	315	26
14	316	37

Table 1: Breakdown of class labels in the development set

2.3.3 Feature Analysis

Simple feature analysis was done on the development set data using standard techniques. The mean and covariance matrix of the original nine features are shown in Table 2.

The same analysis was done for each of the pattern classes in both the two- and fourteen-class problems. That analysis is too lengthy to present here. It is sufficient to report that the classes appear highly intermingled.

To show this data, we resort to the principle components projections of the data down to two dimensions. Figures 1 through 4 show the scatter plot of the data on the first two eigenvectors. The first two plots show the two-class breakup of the data, while the second two show the same data, but for the fourteen-class problem. The two plots for each problem show the data at two different resolutions. The scales are given on the plots. Very little separation between the classes can be seen. Numerous outliers occur in the data.

The next four plots (Figures 5 through 8) are scatter plots of the data in feature space with features 1 and 6 taken as basis vectors. This space yielded the best

```
                         Global Mean Vector:
0.39E+03  0.44E+02  0.23E+01  0.75E+01  0.38E-01 -0.15E+00  0.58E+00  0.28E+04  0.42E+00

                       Global Correlation Matrix:
             1.00  0.95 -0.15  0.51  0.43 -0.11 -0.15  0.97 -0.06
             0.95  1.00 -0.17  0.68  0.48 -0.11 -0.13  0.85 -0.08
            -0.15 -0.17  1.00  0.03 -0.38  0.96  0.84 -0.10  0.09
             0.51  0.68  0.03  1.00  0.42  0.09  0.11  0.40 -0.01
             0.43  0.48 -0.38  0.42  1.00 -0.37 -0.41  0.37  0.02
            -0.11 -0.11  0.96  0.09 -0.37  1.00  0.91 -0.08  0.05
            -0.15 -0.13  0.84  0.11 -0.41  0.91  1.00 -0.12  0.04
             0.97  0.85 -0.10  0.40  0.37 -0.08 -0.12  1.00 -0.04
            -0.06 -0.08  0.09 -0.01  0.02  0.05  0.04 -0.04  1.00

                       Global Covariance matrix:
 0.56E+06  0.33E+05 -0.87E+02  0.12E+04  0.15E+02 -0.46E+01 -0.28E+02  0.57E+07 -0.15E+03
 0.33E+05  0.21E+04 -0.61E+01  0.99E+02  0.10E+01 -0.29E+00 -0.15E+01  0.31E+06 -0.12E+02
-0.87E+02 -0.61E+01  0.63E+00  0.83E-01 -0.14E-01  0.42E-01  0.16E+00 -0.66E+03  0.25E+00
 0.12E+04  0.99E+02  0.83E-01  0.10E+02  0.60E-01  0.16E-01  0.36E-01  0.10E+05 -0.10E+00
 0.15E+02  0.10E+01 -0.14E-01  0.60E-01  0.21E-02 -0.95E-03 -0.46E-02  0.14E+03  0.35E-02
-0.46E+01 -0.29E+00  0.42E-01  0.16E-01 -0.95E-03  0.31E-02  0.13E-01 -0.34E+02  0.10E-01
-0.28E+02 -0.15E+01  0.16E+00  0.86E-01 -0.46E-02  0.13E-01  0.61E-01 -0.24E+03  0.34E-01
 0.57E+07  0.31E+06 -0.66E+03  0.10E+05  0.14E+03 -0.34E+02 -0.24E+03  0.63E+08 -0.11E+04
-0.15E+03 -0.12E+02  0.25E+00 -0.10E+00  0.35E-02  0.10E-01  0.34E-01 -0.11E+04  0.12E+02

                  Eigen Values of Global Covariance Matrix
0.735E-04 0.121E-02 0.116E-01 0.589       2.26       12.3       64.6      0.362E+05 0.636E+08
```

Table 2: Analysis of 378 patterns with nine features

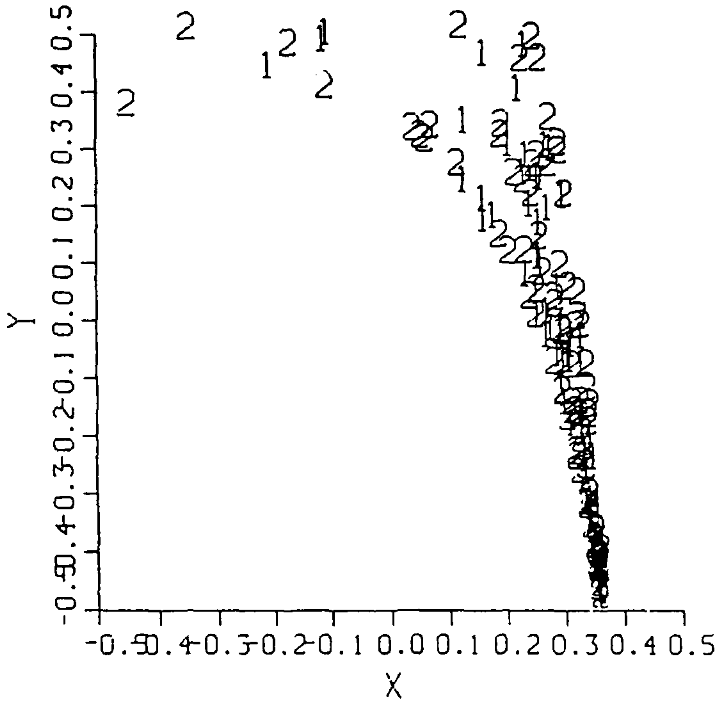


Figure 1: Development set projected on largest two Eigenvectors - full scale, two class labels

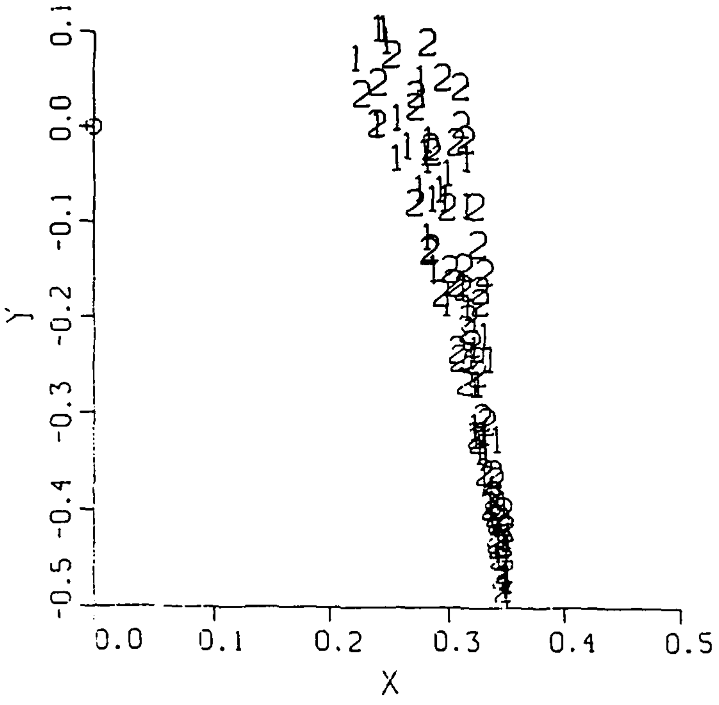


Figure 2: Development set projected on largest two Eigenvectors - partial scale, two class labels

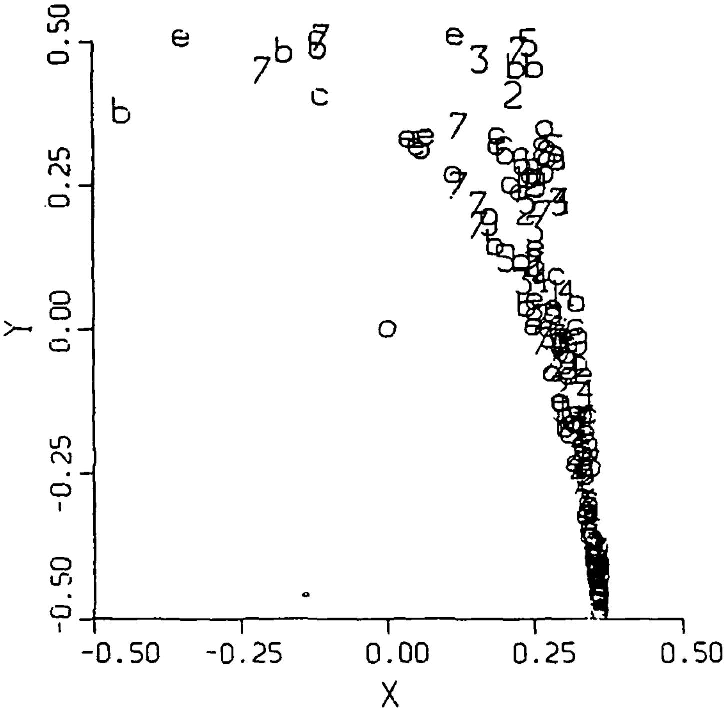


Figure 3: Development set projected on largest two Eigenvectors - full scale, fourteen class labels

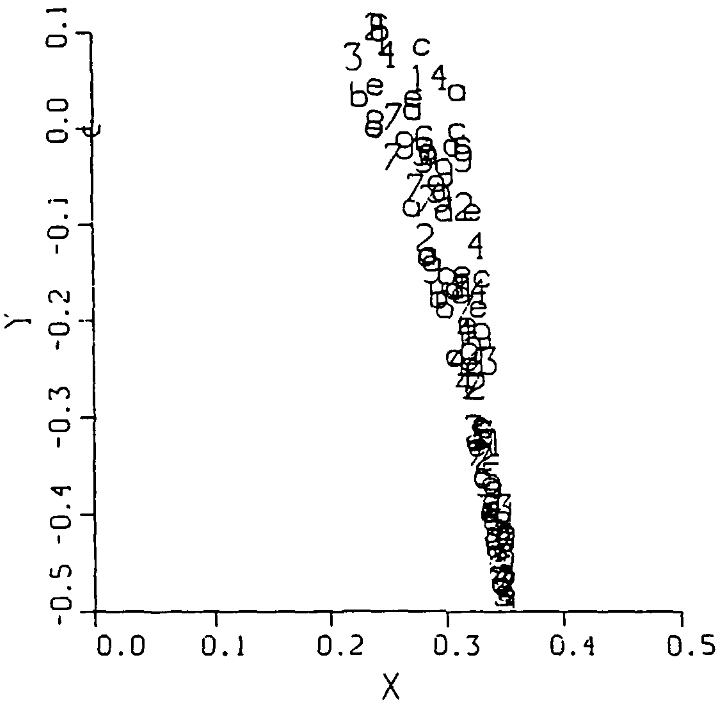


Figure 4: Development set projected on largest two Eigenvectors – partial scale, fourteen class labels

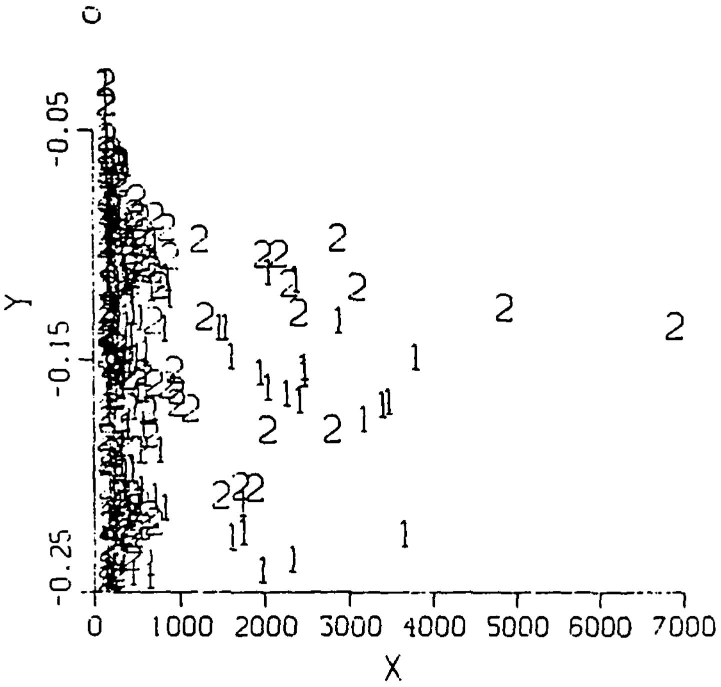


Figure 5: Development set projected on Features 1 and 6 – full scale, two class labels

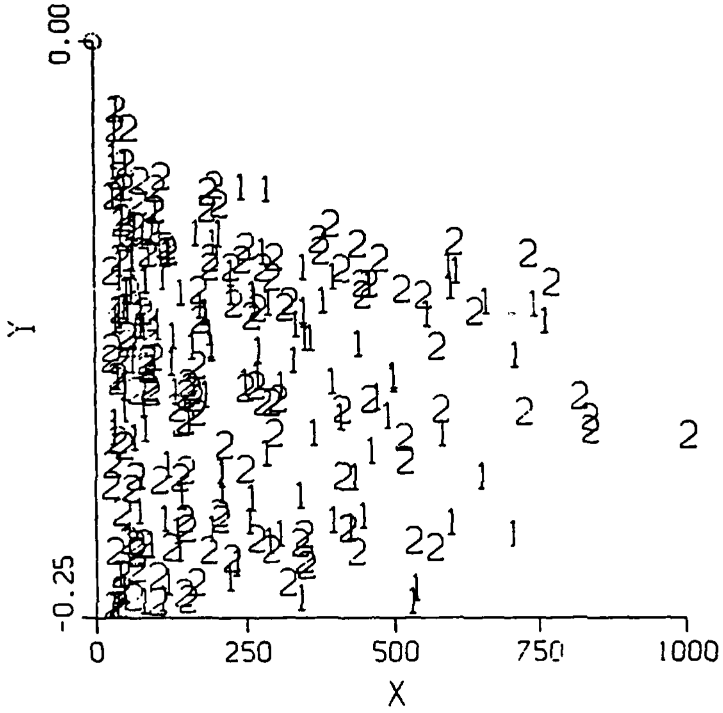


Figure 6: Development set projected on Features 1 and 6 – partial scale, two class labels

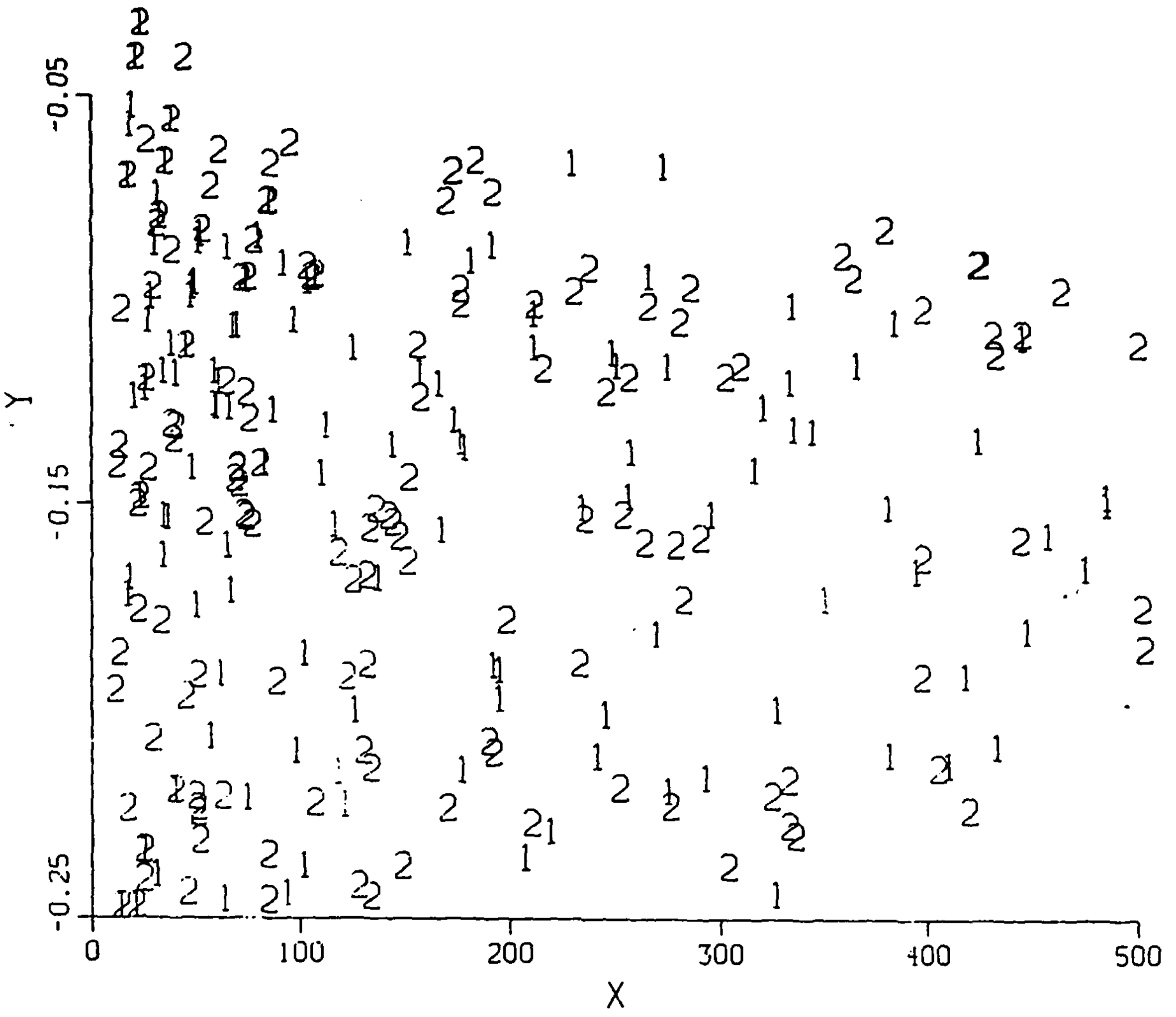


Figure 7: Development set projected on Features 1 and 6 – partial scale, two class labels

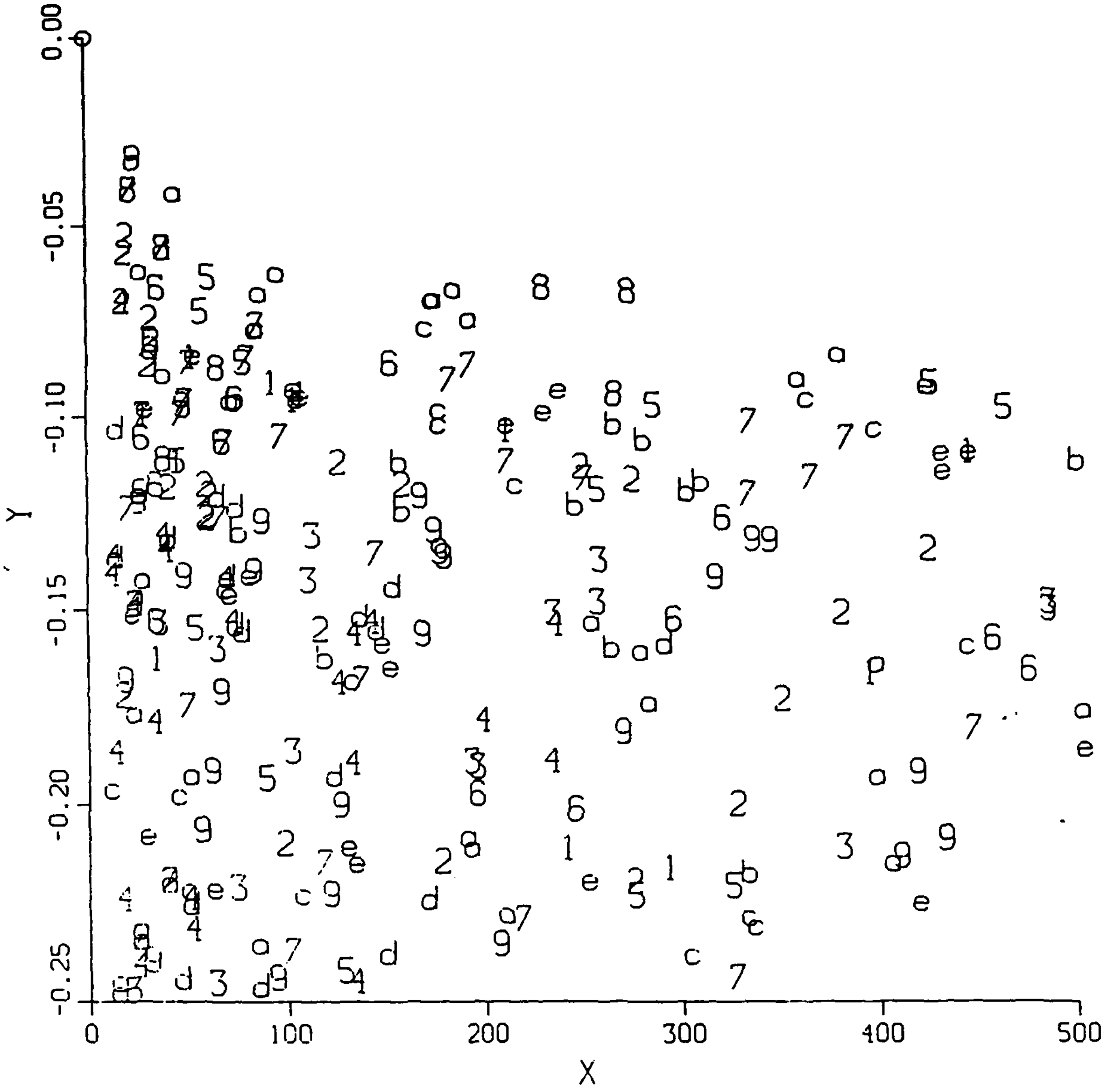


Figure 8: Development set projected on Features 1 and 6 – full scale, fourteen class labels

Gaussian classifier for the two-class problem with number of correct classifications of 218. The classes estimated had mean vectors and covariance matrices of mean(1)=(432.3 −.2), cov(1)=[(5.4X10^5 9.0) (9.0 0.0)] mean(2)=(354.1 −.2), cov(2)=[(5.6X10^5 −.5) (−.5 0.0)]. The confusion matrix showed that 52 patterns in class 1 were classified as class 2 and 88 patterns in class 2 were classified in class 1. The first three plots show the data at various resolutions with two class labels, while the final graph shows the data with fourteen class labels. A number of instances of two samples from the same class lying very close together in the space can be seen. This may mean that the training data contains non-independent samples, e.g. two images taken close together in time.

Additional analysis of this data appears in Section 5 of this report.

2.4 Characterization Set Data

We attempted to build a pattern matrix consisting of all the objects in the characterization set with each of our nine features. A few problems were found in building this matrix. These were:

- The truth file for image 2007064c had errors reading from tape. We have deleted this image set from subsequent processing.
- The second target in image 2003016c had only a few target pixels in its bounding rectangle. This caused a feature (lambda2) for this target to be computed as zero. Since this feature is used as a denominator in a subsequent feature, we arbitrarily set the lambda2 feature value for this target to 0.001.
- The larger images (1280x240 versus 496x320) on the last two tapes (3041, 3042) caused some problems in our tape reading software. We have corrected these reading problems. However, we found when computing the features for the targets in these images that many of the bounding rectangles of the targets did not contain any target pixels. We decided to ignore these images.

In total, a set of 373 targets have been processed and our set of nine features computed for each of them. This is approximately the same number of targets as the development set.

3 Classifier Algorithms

Three basic classification techniques are to be studied in this project. These are a classifier based on the kernel density estimation technique, the K-Nearest Neighbor (KNN) classifier, and a parametric Gaussian classifier. Further, for the kernel-based technique, various methods of choosing the window-width parameter are to be studied. We now report the details of these classifiers.

3.1 Kernel-Based Classifier

The kernel-based classification technique is based on the Parzen nonparametric probability density estimator. This density estimation technique consists of defining a variable kernel with unit volume in feature space and convolving this kernel over a data set. To make this clearer, the mathematical form for the kernel estimate of a density of each point X in a feature space is given in Equation 1, where the kernel is a Gaussian form (though any other kernel could be used) with covariance matrix estimated from the samples from each class. This is the form used in the rest of this study.

$$P[X|C_j] = \frac{1}{M_j}\sum_{i=1}^{M_j}\left\{\frac{1}{(2\pi)^{d/2}}\frac{1}{h^d|\Phi_j|^2}\,Exp\left[\frac{-1}{2h^2}(X-Z_i)^t\Phi_j^{-1}(X-Z_i)\right]\right\} \tag{1}$$

where

$$\begin{array}{rcl} X &=& \text{random variable in feature space} \\ C_j &=& \text{Class } j \\ Z_i &=& i\text{th sample from class } C_j \\ M_j &=& \text{cardinality of class } C_j \\ d &=& \text{dimensionality of feature space} \\ \Phi_j &=& \text{covariance matrix for class } C_j \\ h &=& \text{window} - \text{width parameter} \end{array}$$

Once the class conditional densities are estimated, classification proceeds by a standard minimum risk criterion, where risk can be based on class prior probabilities and cost of misclassification. In our work, we assume that these are equal for each class, and thus we use a maximum probability classifier. The covariance matrices for each class are estimated from the training data for that class.

3.2 Window-Width Selection

A sub-part of this project is the investigation of various methods to choose the window-width parameter, h, in kernel-based density estimation. Three techniques are to be studied. The final result will be a graph showing the resultant criterion function against the classification performance by the leave-one-out technique.

The comparison of these techniques on the ERIM-supplied ATR data is given in Section 8 of this report.

3.2.1 Leave-One-Out

The leave-one-out technique for window-width selection is intuitively simple. It involves searching the window-width parameter, h, for that value which maximizes the final classification performance, where performance is measured using the leave-one-out technique of error estimation. This technique will be used as the baseline with which to compare the other techniques.

3.2.2 Density Overlap

In this method, the overlap between two class-conditional densities is computed by numerical integration for a given value of h. The criterion function is the value of the overlap. For a multi-class problem, various combinations of the two-class overlap can be used, including min-max and average.

The implementation of the numerical integration is done as follows. Due to the large dimensionality of the feature space, the standard grid method was not chosen. Rather, the midpoint between every pair of points from unequal classes was chosen as an evaluation point. At each of these midpoints, the class conditional densities were estimated and the overlap (which for two classes is just the minimum of the two probabilities) is summed. The final criterion value is normalized by the total probability mass (of all classes) seen.

3.2.3 Bhattacharya Metric

This is similar to the density overlap technique, but here the criterion function is the well-known Bhattacharya metric B defined as

$$B = -\ln \int \sqrt{P[X|Class1]P[X|Class2]}$$

where the integration is over all of the feature space.

As for the overlap method, the numerical integration is actually computed over all midpoints of between class vectors. Again, the final statistic is normalized by the total sum of the probabilities of all classes.

3.2.4 MEISER Method

Let x_1, $x_2, ..., x_n$ be a random sample from an $\Re^d$-valued random variable having unknown density function f. The kernel estimate of the density is then

$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) \tag{2}$$

where $K_h(x) = V_h^{-1}K(x/h)$; h is the window-wiath parameter, $h \in \Re$, $h > 0$; $V_h = h^d$ is the window's volume; and K is a functional on $\Re^d$ having integral one (along with other mild restrictions).

The integral square error of the estimate is $\int (f - \hat{f}_h)^2$. It can be shown that this leads to an *expected* square error measure of

$$L_h = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} K_h^*(x_i - x_j) - \frac{2}{n(n-1)}\sum_{i}\sum_{j \neq i} K_h(x_i - x_j) \tag{3}$$

where K_h^* is the convolution of the kernel at window-width h. The MEISER is then to select the window-width h such that L_h is minimized.

In our case, where the kernel is a Gaussian form, the convolution of the kernel reduces to another Gaussian whose covariance matrix is twice the original kernel's matrix.

3.3 K-Nearest Neighbor

The K-nearest neighbor classification method is well known. It involves assigning a sample, X, whose class in unknown into the class with majority vote among the classes of the K closest training samples to X. If there is no majority vote, we assume a random assignment into one of the classes of the set of classes having tied votes.

3.4 Parametric Gaussian Classifier

The parametric Gaussian classifier used in our study is defined by assuming that each of the pattern classes has a Gaussian form with unknown mean vector and covariance matrix. The mean and covariance matrix are estimated from the class' training data and these parameters are "plugged into" the parametric form of the classifier.

4 Confidence Measures

Although the probability of correct classification is, by itself, a very useful indicator of the quality of a decision, there are other higher-level considerations concerning the degree of certainty in the probability estimates. In standard statistical parameter estimation, it is considered good practice to report a confidence interval on any estimated parameter. One of our tasks is to look at confidence measures estimates for probability density estimators. In the rest of this section, we give a brief overview of the techniques used for confidence measure estimation.

4.1 Non-linear Scaling

Since confidence estimates are formed based on the number of samples used to make the parameter estimate, a direct approach to determining the confidence in a probability estimate for every point in feature space is based on the non-linear scaling on the number of points which contributed to the density estimate. We choose to base our scaling on the Mahalanobis distance to points in the training set.

More formally, the non-linear confidence measure at each pattern in a data set is defined as the minimum, over all of the classes, of the number of other patterns which contributed to the density estimate divided by the total number of possible patterns which might have contributed. A pattern contributes to the density estimate of the pattern of interest if it is within some threshold in Mahalanobis distance from that pattern. In our experiments, the threshold distance is set at 1.0.

While this method has intuitive appeal, its statistical properties are unknown.

4.2 Bootstrap

Of course, the "best" approach to confidence measure estimation is to go to the field and gather hundreds of sets of data and use each to estimate the probability density functions (pdfs). The resultant pdfs could then be used to form confidences in the pdfs at every point in feature space. Unfortunately, this procedure is unworkable due to the high cost in obtaining even a limited set of ATR training data.

A new statistical technique called *bootstrapping* has been used with some success to form confidence intervals for a single real-valued parameter without taking more data. One of our tasks is to apply the bootstrapping paradigm to the more difficult problem of a confidence interval about a functional estimate, in this case a probability density function for the ERIM ATR data.

The bootstrap technique can be described generically as follows. It is a resampling technique for nonparametric estimation of bias, variance and more general measures of error. First, we are given a statistic $\theta(X, F)$ of interest, where $X = (x_1, ...x_n)$ indicates the entire n point independent, identically distributed (i.i.d.) F sample, where F is some distribution function. On the basis of observing $X = x$, we wish to estimate some aspect of θ's distribution, for example $E(\theta)$. The bootstrap procedure is then:

(1) Fit the nonparametric Maximum Likelihood Estimator (MLE) of F,

$$\hat{F}: \ mass\ 1/n\ at\ x_i,\ i = 1, 2, ..., n \tag{4}$$

(2) Draw a bootstrap sample from $\hat{F}$,

$$x_1^*, x_2^*, ..., x_n^*\ distributed\ i.i.d.\ \hat{F}, \tag{5}$$

and calculate $\theta^* = \theta(x_1^*, ..., x_n^*, \hat{F})$.

(3) Independently repeat step 2 - a large number, M, times obtaining bootstrap replicants $\theta_1^*, \theta_2^*, ..., \theta_M^*$ - and calculate whatever aspect of theta's distribution in which we are interested. For instance, if we wish to estimate $E(\theta)$, we calculate,

$$E(\hat{\theta}) = \frac{1}{M} \sum_{m=1}^{M} \theta_m^* \tag{6}$$

In all cases, we are calculating a Monte-Carlo approximation to the nonparametric MLE for the quantity of interest; the approximation being that M is finite rather than infinite.

Bootstrapping is applied to pdf estimation in the following manner. For each pattern, X_i, in the dataset, M bootstrap replicants of the dataset, minus the given pattern X_i, are computed. The deletion of the pattern of interest (X_i) from the bootstrap samples ensures that leave-one-out type classifiers are used in subsequent processing. For each of these M bootstrap samples, the estimated probability that X_i belongs to each of the two classes in a two-class problem $(P_m[X_i \in C_l], l = 1, 2,$ m=1,...,M) is computed using the kernel-based density estimation method. The bootstrap confidence measure at X_i is then defined to be

$$cboot(X_i) = \sum_{m=1}^{M} \sum_{n=1}^{M} I(P_m[X_i \in C_1)/P_n[X_i \in C_2])$$

where

$$I(x) = \begin{cases} 1/M^2 & \text{if } x \geq 1 \\ -1/M^2 & \text{otherwise} \end{cases}$$

This is nothing more than computing the value of (the number of times the probability ratio would decide class 1) minus (the number of times the probability ratio would decide class 2) normalized by the number of possible probability ratios. Note that if every computed probability for class 1 (class 2) at X_i was greater than all the computed probabilities for class 2 (class 1) then $cboot(X_i) = 1(= -1)$. This full process is then repeated for each X_i in the dataset. The number of bootstrap replicants, M, used in our experiment is 100.

5 Evaluation – Development Set

This section describes the results of our study of the development set of images.

5.1 KNN Results – Development Set

The two-class pattern matrix was analyzed for class separability by a leave-one-out KNN classifier. This was performed by searching over all feature subsets of a given size made up of nine original features and, for each subset, recording the number of correct classifications (out of 378). The value of K in the KNN classifier was varied from 1 to 10. All ties between the classes when classifying a pattern are resolved randomly.

Subset Size	Number Correct	Feature Subset	K
1	232	7	7
2	248	5,9	7
3	253	5,6,9	1
4	246	3,5,7,9	1
5	246	3,5,6,7,9	1
6	218	1,2,4,5,6,8	8
7	218	1,2,3,4,5,6,8	8
8	217	1,2,3,4,5,6,7,8	8
9	215	all	8

Table 3: KNN results for the two-class problem

5.1.1 Two-Class Problem

The best feature subsets, along with their performance in terms of the number of correctly classified patterns, are shown in Table 3.

5.1.2 Fourteen-Class Problem

We performed the same classification analysis on this fourteen-class matrix as for the two-class matrix. In Table 4, for feature subsets of size 1 through 9, we show the best subset found, for all K from 1 to 10, and the number of correctly classified samples out of 378.

5.2 Gaussian Results — Development Set

As for the KNN classifier, a search was performed for a standard Gaussian classifier over all possible feature subsets for both the two-class and fourteen-class problems. Here each class is assumed to have a class-conditional Gaussian distribution with an unknown mean vector and covariance matrix. These are estimated for each class' data using the maximum likelihood technique and then these estimates are “plugged-in” to the standard Gaussian decision rule.

Subset Size	Number Correct	Feature Subset	K
1	80	1	9
2	93	1,7	1
3	92	1,5,6	1
4	98	1,5,6,9	1
5	99	1,5,6,8 9	1
6	97	1,2,5,6,8,9	1
7	95	1,2,3,5,6,8,9	1
8	93	1,2,3,5,6,7,8,9	1
9	84	all	3

Table 4: KNN results for the fourteen-class problem

5.2.1 Two-Class Problem

For the two-class problem, the best feature subsets, along with their performance in terms of the number of patterns correctly classified are shown in Table 5.

5.2.2 Fourteen-Class Problem

We did the same procedure as the previous item using the fourteen-class problem. The results are shown in Table 6.

Note that in the subset of size 9, a singular covariance matrix could not be inverted to form the Gaussian classifier.

5.3 Kernel Results – Development Set

Both the two-class and fourteen-class problems were analyzed using the kernel-based classifier. Unlike the analysis for the KNN and Gaussian classifiers, the analysis of the kernel-based classifier was made only in the full nine-dimensional feature space. In this space, the pdf for all the classes was estimated at various values of h and the error rate was computed.

In the next two sections, we show the values of h (sampled at values 0.1 through 2.0 in steps of 0.1) versus the number of correctly classified samples out of 378. Undecided samples are those samples that had the *same* probability for each class.

Subset Size	Number Correct	Feature Subset
1	202	2 and 5
2	218	1,6
3	228	2,5,6
4	234	2,4,5,6
5	234	2,3,4,5,7
6	230	1,2,3,4,5,6
7	214	1,2,3,4,5,6,7
8	205	1,2,4,5,6,7,8,9
9	205	all

Table 5: Gaussian results for two-class problem

Subset Size	Number Correct	Feature Subset
1	44	1
2	68	3,5
3	75	3,4,5
4	79	2,3,4,5
5	82	2,3,4,6,7
6	93	3,4,5,6,7,9
7	100	1,2,4,5,6,7,8
8	105	1,2,3,4,5,7,8,9
9	*	*

Table 6: Gaussian results for the fourteen-class problem

Value of h	Correct Classifications	Undecided
0.1	209	69
0.2	234	22
0.3	244	12
0.4	234	8
0.5	233	4
0.6	233	3
0.7	223	2
0.8	226	1
0.9	217	1
1.0	214	1
1.1	217	1
1.2	215	1
1.3	215	1
1.4	218	0
1.5	217	0
1.6	213	0
1.7	208	0
1.8	207	0
1.9	202	0
2.0	198	0

Table 7: Kernel results for the two-class problem

For small h values, these are usually points that are "outliers", which had no point from either class close enough to contribute to the convolutional density estimate.

Section 7 reports the evaluation of more useful (as will be shown) methods of setting the window-width than this brute force search for the value of h which maximizes the leave-one-out estimate of the number of correctly classified samples.

5.3.1 Two-Class Problem

Table 7 presents the kernel-based classification performances in the nine-dimensional feature space for the two-class problem.

Class	(Undecided)	1	2
1	5	119	58
2	7	64	125

Table 8: Confusion matrix for kernel-based classifier with $h = 0.3$

The confusion matrix for the best ($h = .3$) kernel-based classifier in nine-dimensional feature space for the two-class problem is shown in Table 8.

5.3.2 Fourteen-Class Problem

Table 9 presents the kernel-based classification performances for the nine-dimensional feature space for the fourteen-class problem.

The best kernel-based classifier ($h = .2$) for the problem had the confusion matrix shown in Table 10.

5.4 Analysis of Classifiers on the Development Set

A number conclusions can be drawn from the above. First, we address the two-class problem then the fourteen-class problem.

5.4.1 Two-Class Problem

For this problem, feature subsets of size from 3 to 5 appear best, in terms of classification performance, given the limited number of training samples available. Second, the features do not do well in discriminating the two classes. It appears that the nine features used in this study are not designed to separate tracked from non-tracked targets without the use of additional *a priori* information.

For this problem, the "best" feature subsets are relatively unstable, though both Feature(5) and Feature(6) appear quite often in the best subsets.

It is interesting to compare the best results for the kernel-based classifier on all nine features to those of the Gaussian and KNN classifiers. For h=0.3, the kernel-based classifier yields 244 correct classifications with 12 undecided patterns. The

Value of h	Correct Classifications	Undecided
0.1	84	50
0.2	89	15
0.3	85	11
0.4	87	6
0.5	80	3
0.6	76	3
0.7	78	3
0.8	79	2
0.9	76	1
1.0	79	1
1.1	75	1
1.2	72	1
1.3	65	1
1.4	58	1
1.5	55	1
1.6	58	1
1.7	56	1
1.8	55	0
1.9	56	0
2.0	52	0

Table 9: Kernel results for the fourteen-class problem

Class	(U)	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	2	5	1	3	0	1	0	2	1	1	0	0	1	0	2
2	2	1	10	2	2	0	0	3	1	2	1	3	0	3	0
3	0	3	2	2	3	0	2	0	0	9	1	0	1	4	0
4	1	0	1	1	7	1	0	1	0	1	1	1	0	8	1
5	4	2	0	1	1	1	1	1	0	0	1	2	3	1	3
6	0	0	0	2	0	1	3	2	0	0	2	2	0	0	2
7	0	2	0	1	1	2	4	10	5	8	1	4	2	2	4
8	0	1	1	0	0	0	0	0	4	0	2	0	1	0	0
9	1	1	4	5	2	0	2	5	0	10	1	2	0	3	1
10	1	0	0	0	0	2	2	2	2	2	11	3	3	1	5
11	0	0	0	0	1	1	2	1	2	2	2	6	1	1	4
12	2	1	0	2	1	0	0	3	2	2	4	1	6	1	6
13	1	1	0	1	10	1	1	0	0	2	1	1	2	4	1
14	1	3	1	0	2	2	1	5	0	2	3	2	5	0	10

Table 10: Confusion matrix for the kernel-based classifier for the fourteen-class problem with $h = 0.2$

Feature	KNN		Kernel		
Subset	K	Number Correct	h	Number Correct	Undecided
7	7	232	0.8	228	0
5,9	7	245	0.1	197	9
5,6,9	1	253	0.2	214	4
3,5,7,9	1	246	0.1	225	14

Table 11: Comparison of KNN and kernel results

best KNN classifier with $K = 8$ had only 215 correct classifications. The best Gaussian classifier had 205 correct classifications.

We also compared the kernel-based classifier's performance on the best feature subsets found using the KNN classifier. A few of the best KNN feature subsets, along with their performance in terms of the number of correctly classified patterns, are shown in Table 11.

Thus, on the *best* KNN subsets, the kernel-based classifier was not able to achieve the same level of performance as the KNN classifier.

Similarly, we compare the error rates for the best Gaussian classifier subsets to the error rates for these same subsets under the kernel-based classifier.

Feature	Gaussian	Kernel	
Subset	Number Correct	h	Number Correct
2	202	1.0	188
5	202	0.6	231
1,6	218	0.5	221
2,5,6	228	0.5	254
2,4,5,6	234	0.2	246

Table 12: Comparison of gaussian and kernel results

A few of the Gaussian classifier's best feature subsets, along with their performance in terms of the number of patterns correctly classified are shown in Table 12.

Note that the best performance we have achieved on any subsets of features has been for subset 2, 5, and 6 with the kernel-based classifier and h=0.5.

As an additional aid in understanding the structure of the features in this data, we computed the best (in terms of most correctly classified samples) kernel-based classifier in separating the two classes when using only *one* of the original nine features. This was done by searching over h parameter values for each of the nine original features in isolation. Shown in Figures 9 through 17 plots are the density functions fit to each of the two classes. Each figure shows one of the original features and each of the two class density functions. Since we are using a maximum probability criteria to classify the samples, given the density functions, decision regions are easily determined from the plots by looking for regions where one density function is greater than the other.

In these figures, little separation between the classes can be seen. Also, the ability of the kernel-based classifier to range from Gaussian classifier-like decision regions, e.g. Feature 9, to KNN-type decision regions, e.g. Feature 5, is shown.

5.4.2 Fourteen-Class Problem

For the fourteen-class problem, results similar to the two-class problem are seen. In this case, Feature 4 does not appear to contain much class separability information when evaluated by KNN classifier, but the Gaussian classifier adds it to its best feature subset at subset size of three. Likewise the KNN classifier includes

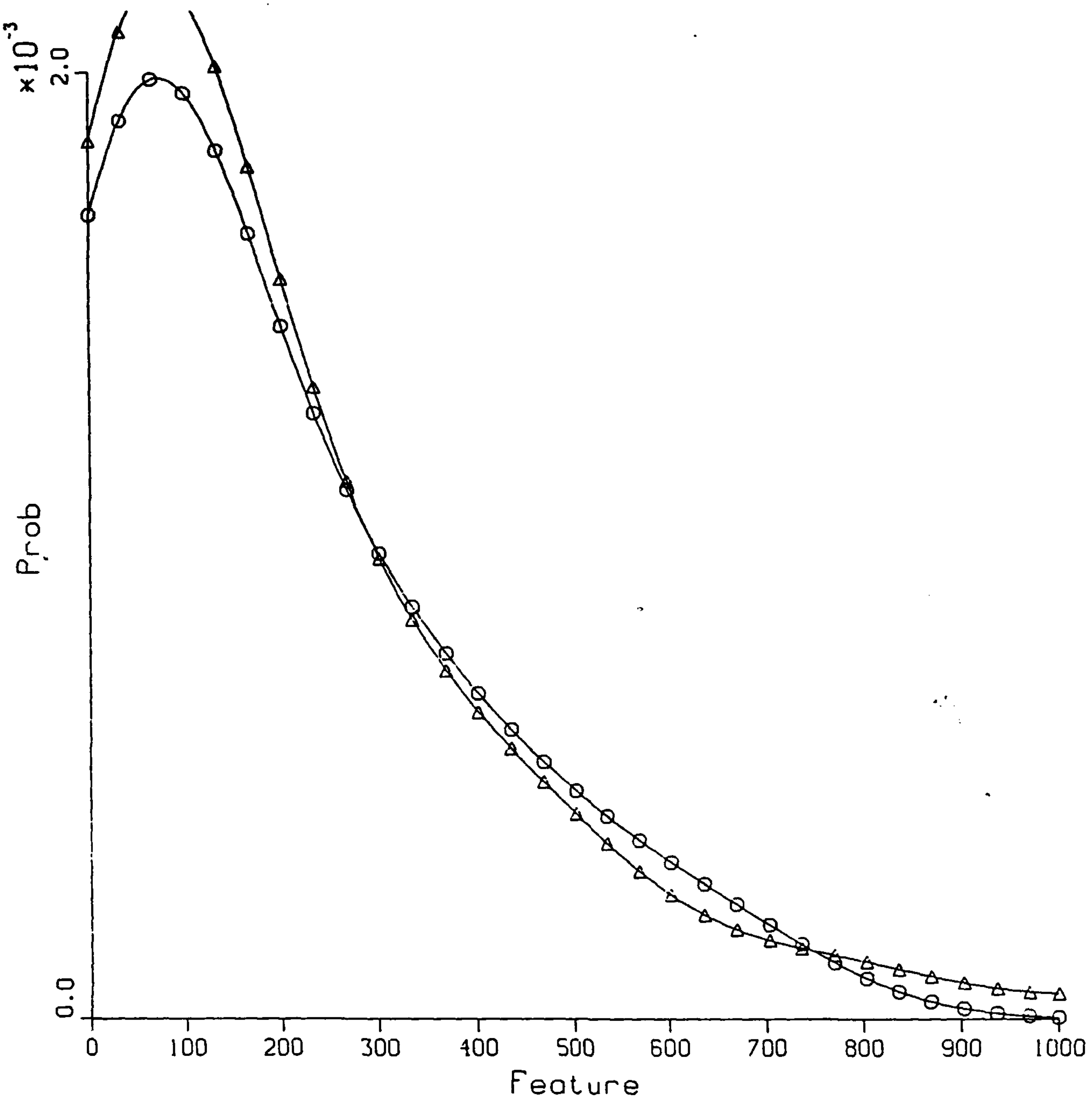


Figure 9: Best kernel-based density functions for Feature 1

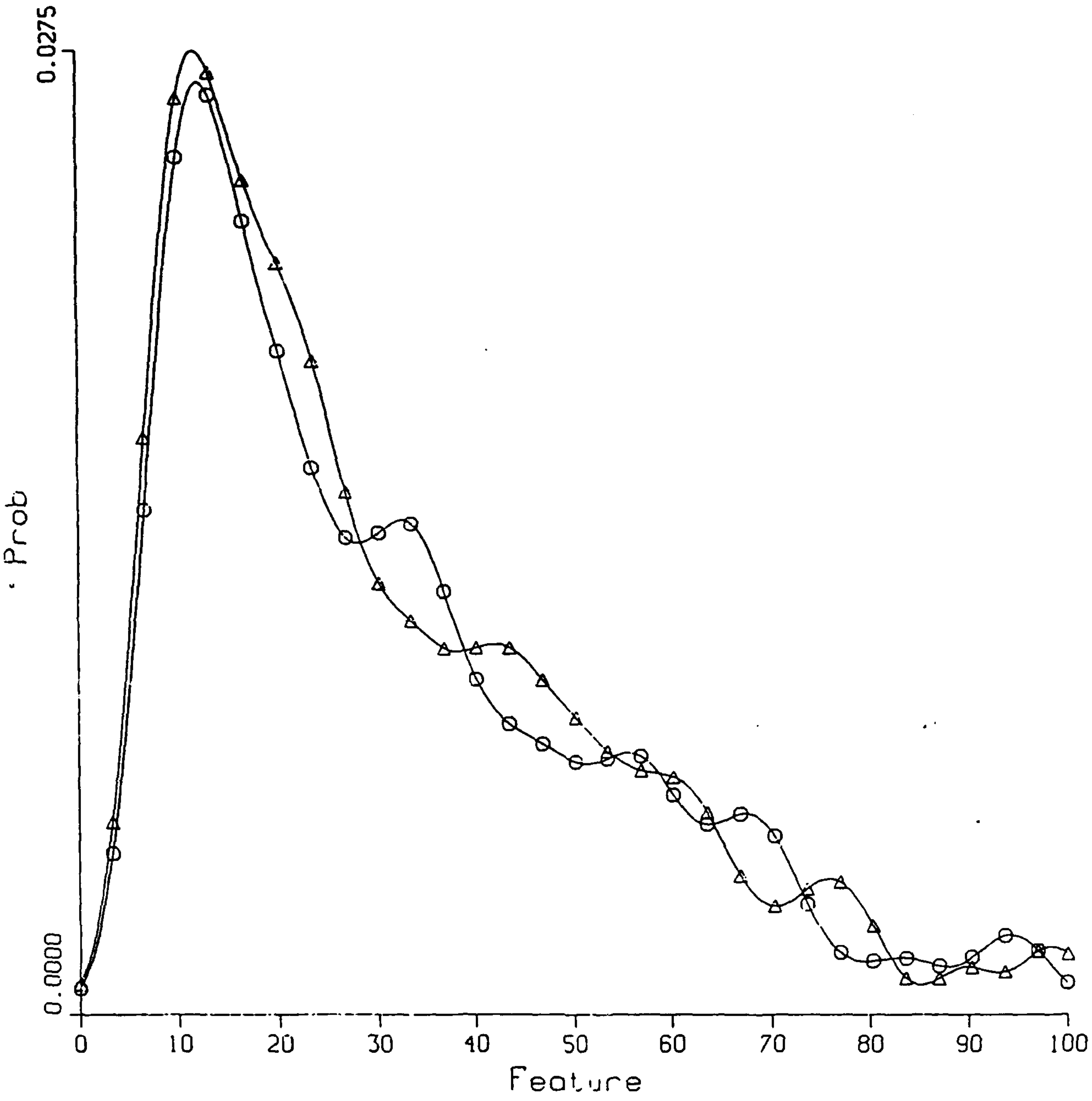


Figure 10: Best kernel-based density functions for Feature 2

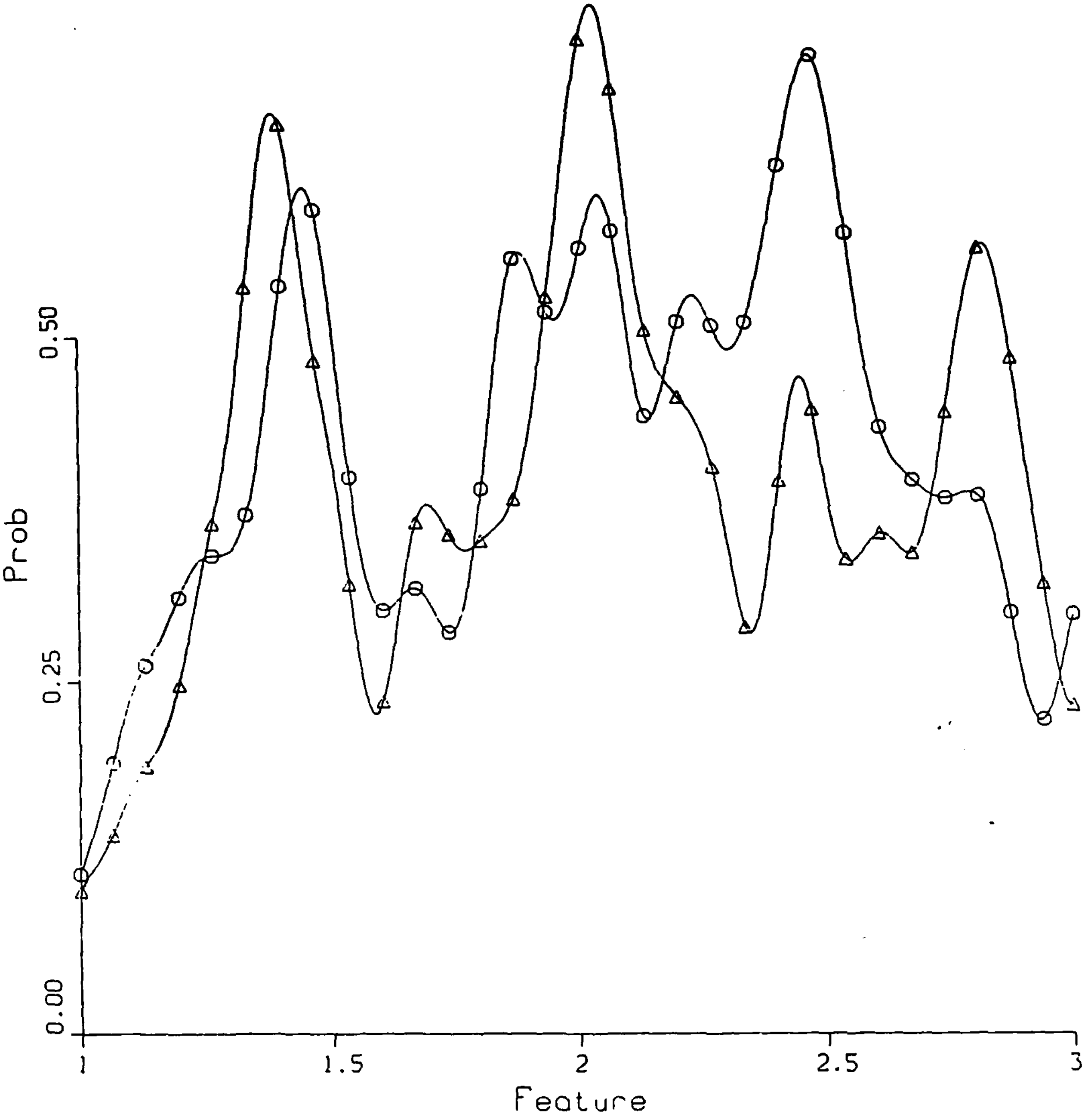


Figure 11: Best kernel-based density functions for Feature 3

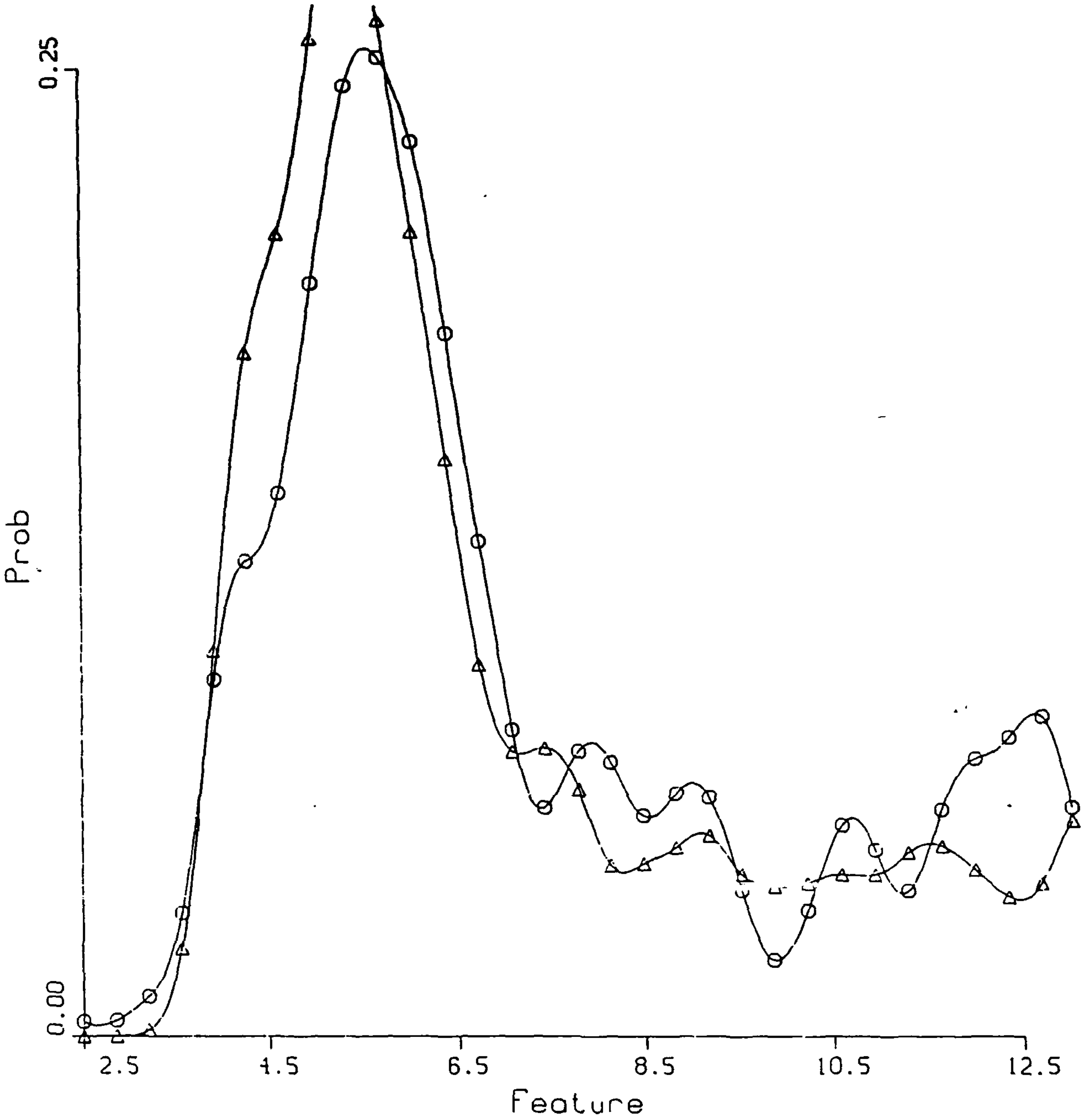


Figure 12: Best kernel-based density functions for Feature 4

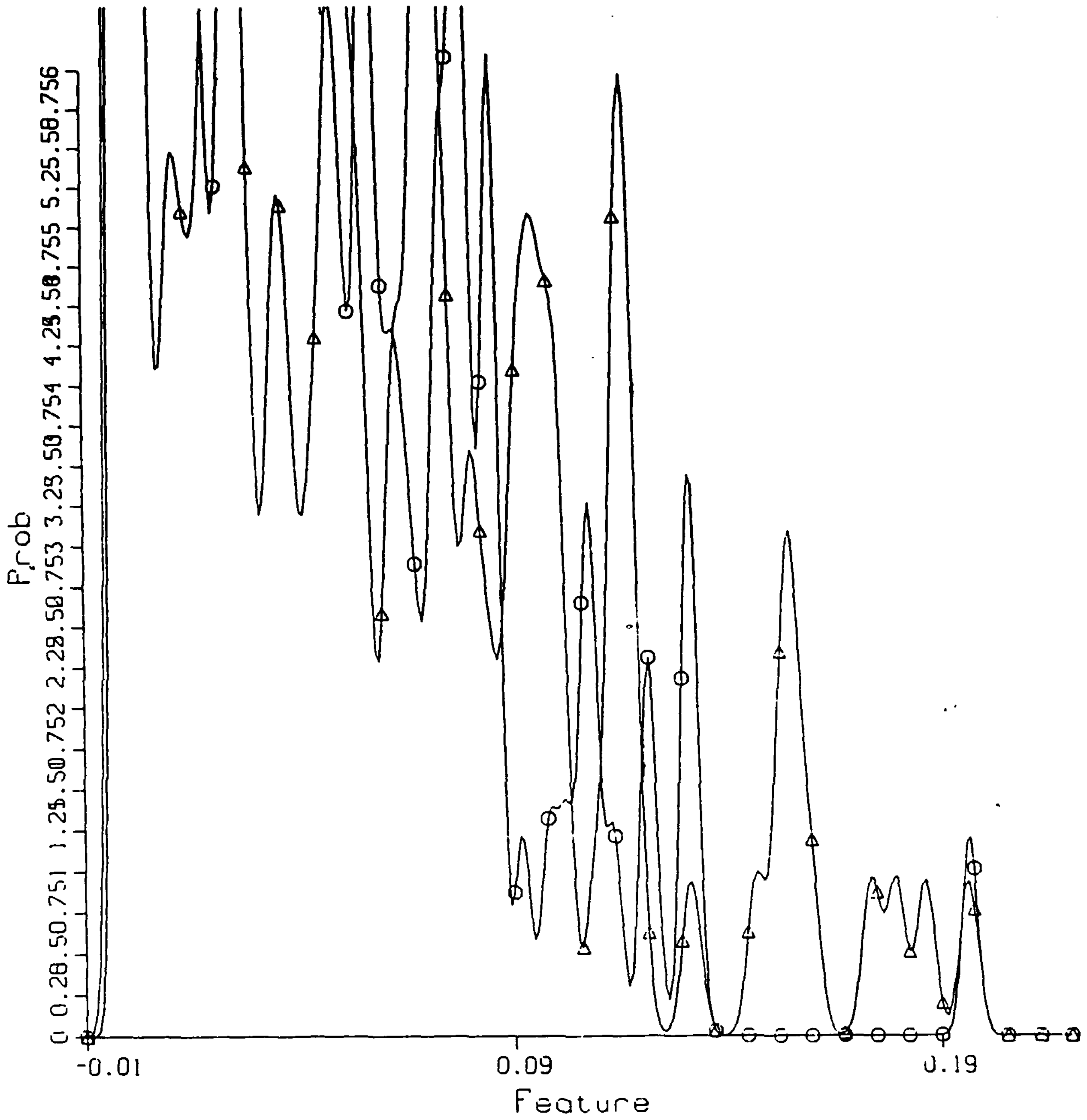


Figure 13: Best kernel-based density functions for Feature 5

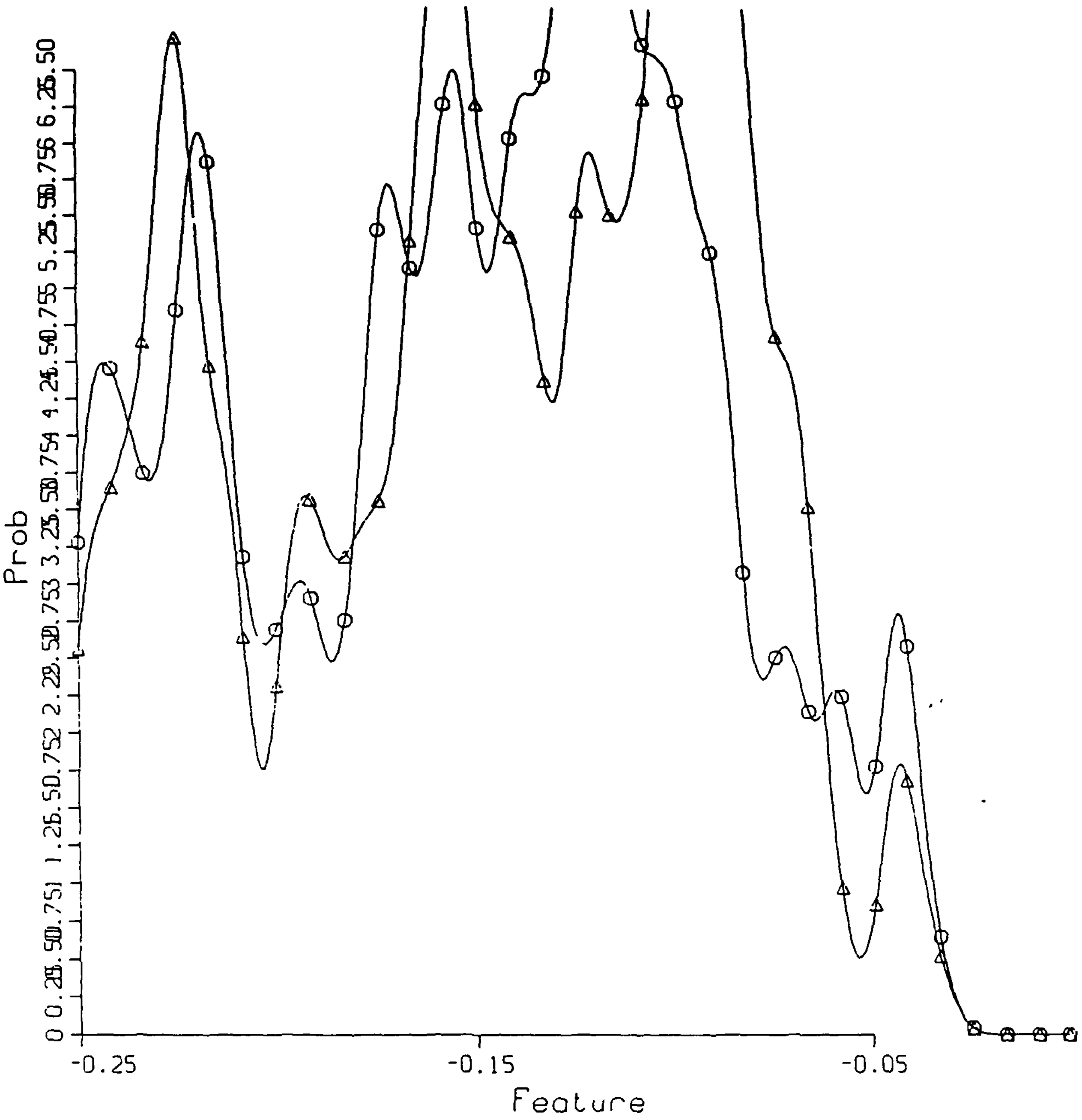


Figure 14: Best kernel-based density functions for Feature 6

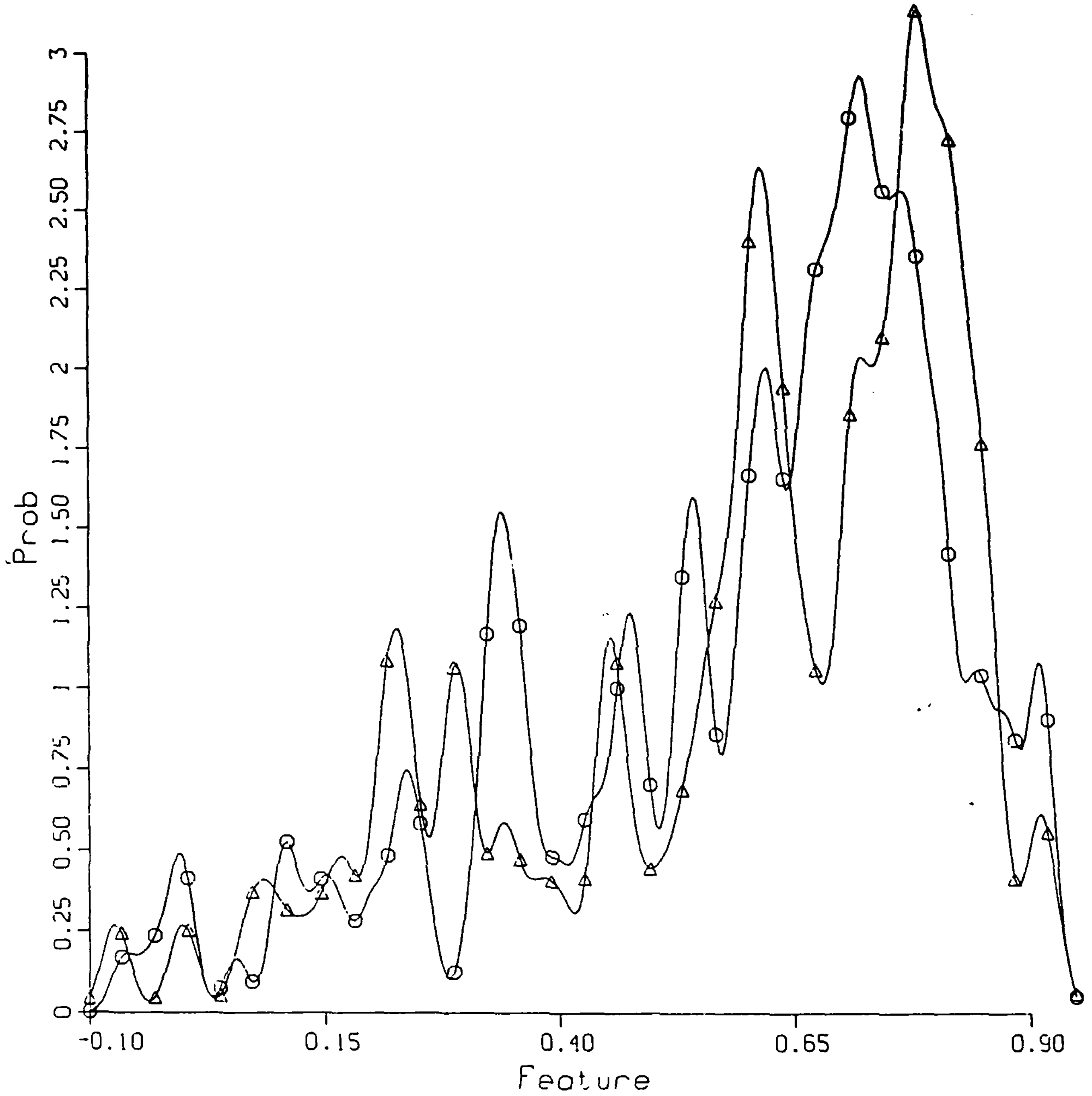


Figure 15: Best kernel-based density functions for Feature 7

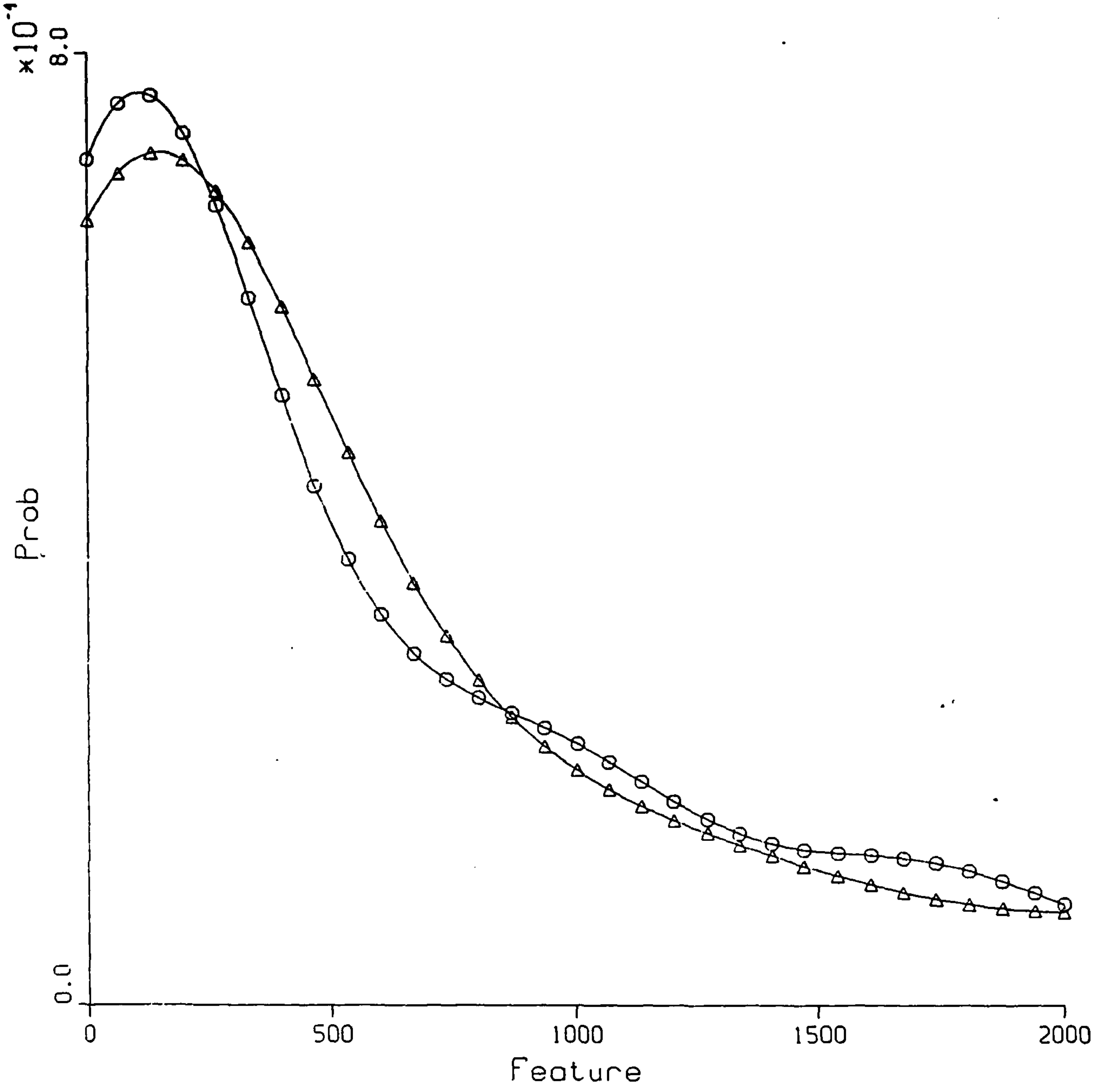


Figure 16: Best kernel-based density functions for Feature 8

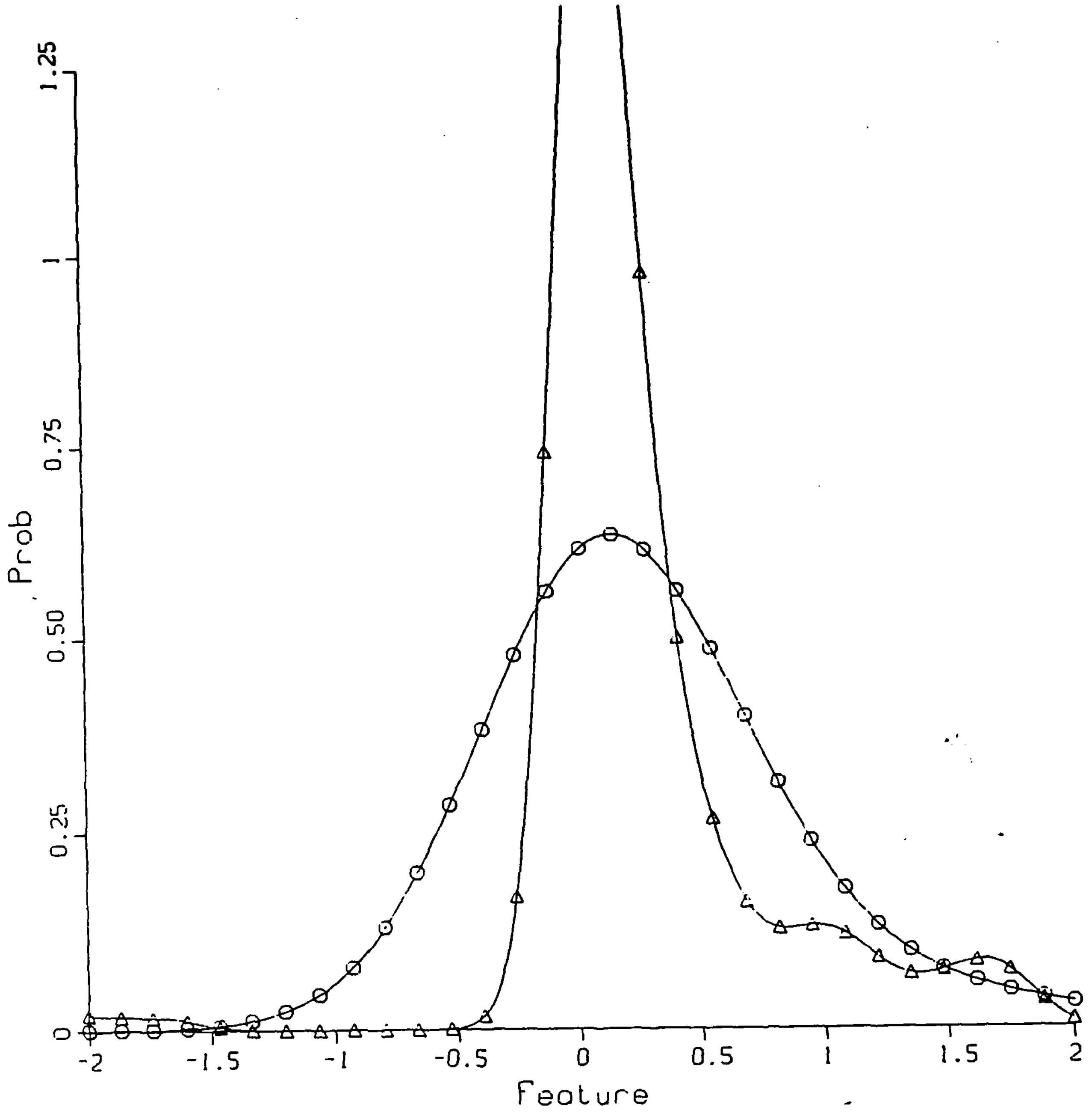


Figure 17: Best kernel-based density functions for Feature 9

Feature 1 in all subsets and while the Gaussian method likes it as the best single feature, it chooses not to include it in the smaller size subsets with multiple features.

It is interesting that the best performance occurs with the Gaussian classifier with eight features. This is fairly unusual.

Note that for the kernel-based classifier, with $h = 0.2$, we had 89 correctly classified patterns with 15 undecided. The KNN classifier on all the nine features had a performance of 84 correct classifications with $K = 3$, while the Gaussian classifier could not be run on all nine features due to a non-singular matrix for one of the classes.

6 Evaluation – Characterization Set

This section details the results of applying the classifiers trained on the development set to the characterization set. Here we use the actual class information present in the characterization set and examine the actual error rates obtained by using the trained classifiers to the predicted error rates when using the leave-one-out method of error estimation on the training set.

The characterization set pattern matrix consisting of nine features for each of the 373 test target objects was classified using the classifiers built for the development set. The results of these comparisons follow. For each, we report a table of the leave-one-out error estimate for the development (training) set and the error rate found for the characterization (test) set. Each table presents the best (in terms of the minimum error rate) classifier type and its parameters for the given problem.

6.1 Two-Class Problem

The first set of classifiers was built for the problem of determining two classes in the data (tracked versus non-tracked) when using all nine features. Table 13 shows, for each of our three types of classifiers, the best error rate estimated by the leave-one-out technique for the development set and compares this rate to that over the characterization set.

The next set of classifiers were trained for the two-class problem using the three features 2, 5, and 6 – these yielded the best feature subset of size 3 for the Gaussian classifier. Table 14 shows, for each of our three types of classifiers, the

Classifier	Parameters	Error Rates		
		Training	Testing	Percent Change
Kernel	$h = 0.3$	35.5	53.4	+17.4
KNN	$K = 8$	42.6	48.8	+06.2
Gaussian	N/A	37.3	38.3	+01.0

Table 13: Error rates for two-class problem using all nine features

Classifier	Parameters	Error Rates		
		Training	Testing	Percent Change
Kernel	$h = 0.5$	32.8	44.0	+11.2
KNN	$K = 6$	47.9	49.3	+01.4
Gaussian	N/A	39.7	42.1	+02.4

Table 14: Error rates for two-class problem using features 2, 5, and 6

best error rate estimated by the leave-one-out technique for the development set and compares this rate to that over the characterization set.

Another set of classifier was trained on the feature subset of size three consisting of features 5, 6, 9; this subset had the best performance for any subset of size 3 for the KNN classifier. Table 15 shows, for each of our three types of classifiers, the best error rate estimated by the leave-one-out technique for the development set and compares this rate to that over the characterization set.

Classifier	Parameters	Error Rates		
		Training	Testing	Percent Change
Kernel	$h = 0.2$	43.4	45.3	+01.9
KNN	$K = 1$	33.6	41.3	+07.7
Gaussian	N/A	48.1	52.3	+04.2

Table 15: Error rates for two-class problem using features 5, 6, and 9

Classifier	Parameters	Error Rates		
		Training	Testing	Percent Change
Kernel	$h = 0.2$	76.5	90.1	+13.6
KNN	$K = 3$	77.8	86.9	+09.1
Gaussian	Not Run	–	–	–

Table 16: Error rates for fourteen-class problem using all features

6.2 Fourteen-Class Problem

The final set of classifiers were trained on all nine features for the fourteen-class problem. Table 16 shows, for each of our three types of classifiers, the best error rate estimated by the leave-one-out technique for the development set and compares this rate to that over the characterization set.

6.3 Analysis

A number of conclusions can be drawn from the above data.

- First, as expected, the error rates for each of the classifiers increase when the testing data is used to estimate the error rate.

- Among the three classifiers, the kernel-based classifier appears the most susceptible to training-set overfitting while the Gaussian classifier appears the least susceptible.

- To further quantify this overfitting, for the kernel classifier, plots were produced of the values of h versus classifier performance for both the training and the test sets.

- Figure 18 shows such a plot for the two-class data with all nine features. The curve with filled-in circles shows the training set results and the curve with hollow triangles shows the training set results. Note the overfitting of the data for $h < 0.5$. A surprising aspect shown in Figure 18 is the higher performance of the trained classifier on the testing data when $1.0 < h < 4.0$. Figure 19 shows the same data plotted out further in values of h showing that the performance predicted by the leave-one-out classifier matches the actual pertormance seen for the testing data when $h > 4.0$.

- Figure 20 show the same curves for the two-class problem using only features 2, 5, and 6. Similar results are seen.
- Figure 21 shows the same curves as above for all nine features in the fourteen-class problem. Here more "normal" behavior is shown: the test set performance is always below the training set performance. Again, data-set overfitting can be seen for $h < 0.25$.

7 Window-Width Optimization Study

The window-width optimization study is now described.

7.1 Development Set

For the development set, we evaluated the window-width optimization techniques on the two-class problem using all nine of the original features.

Figure 22 shows a graph of the results of the number of correctly classified samples for the development set using the kernel-based classifier versus the h window-width parameter. The best result occurs at $h = 0.3$ with 244 correctly classified samples.

Figure 23 shows the graph of the Bhattacharya metric over the development set for the same values of h as the previous figure. The maximum value of the metric is achieved at $h = 0.9$. This yields a classification performance of 217 correctly classified samples, in other words a percentage of 89 of the best achievable classification performance.

Figure 24 shows the graph of the overlap metric over the development set for the same values of h as the previous graphs. The minimum value of the metric is achieved at $h = 1.1$ which yields a classification performance of 217 correctly classified samples.

Figures 23 and 24 show, respectively, that the Bhattacharya metric and the overlap metric perform reasonably on this data set. They achieve their optimal value of h close to the best value of h found by the leave-one-out error estimator. As functions of h, they have a unique global optimal and are well-behaved.

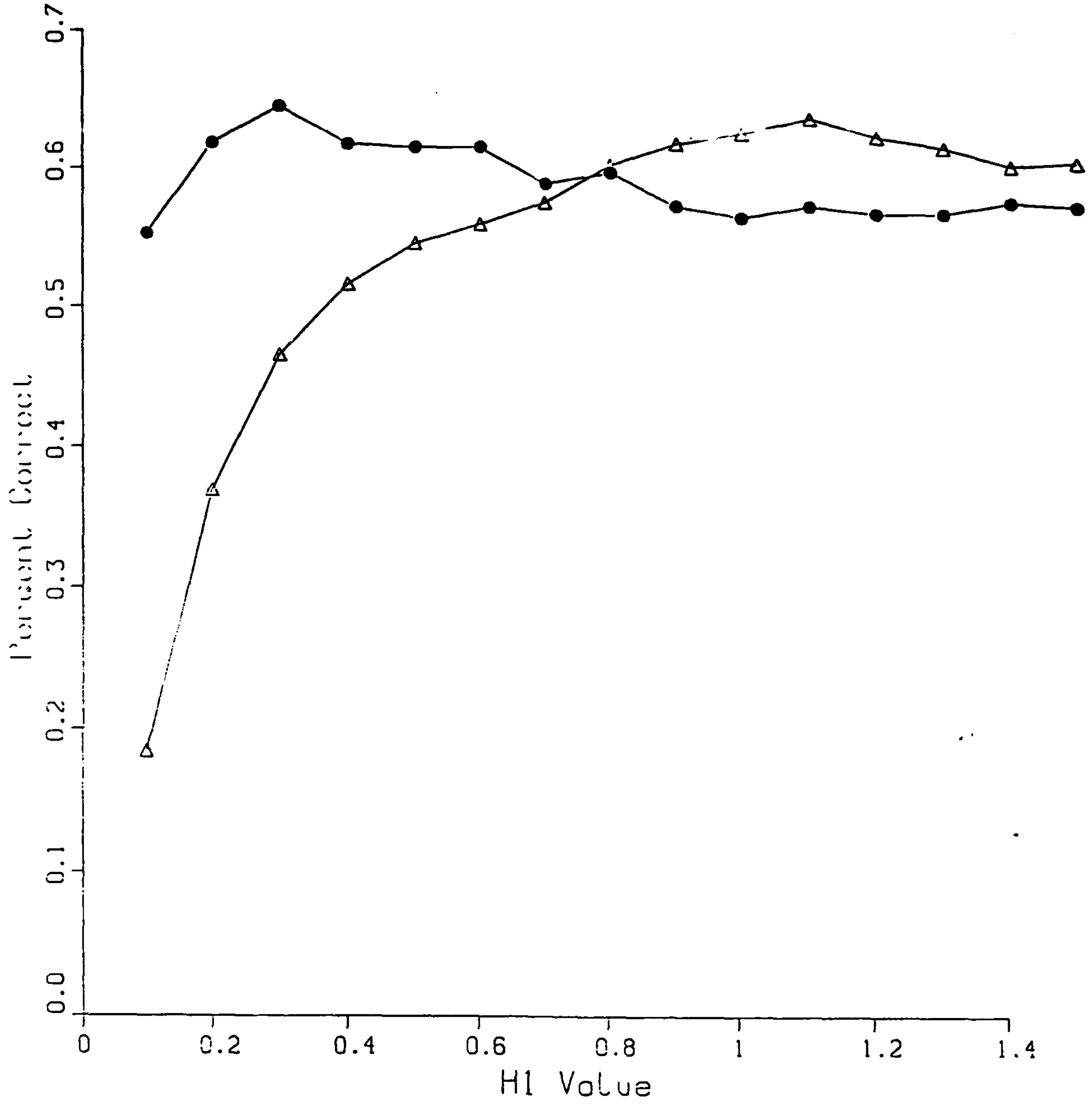


Figure 18: Percent Correct Classification versus *h* for both the development and characterization sets over the two-class, nine-feature problem

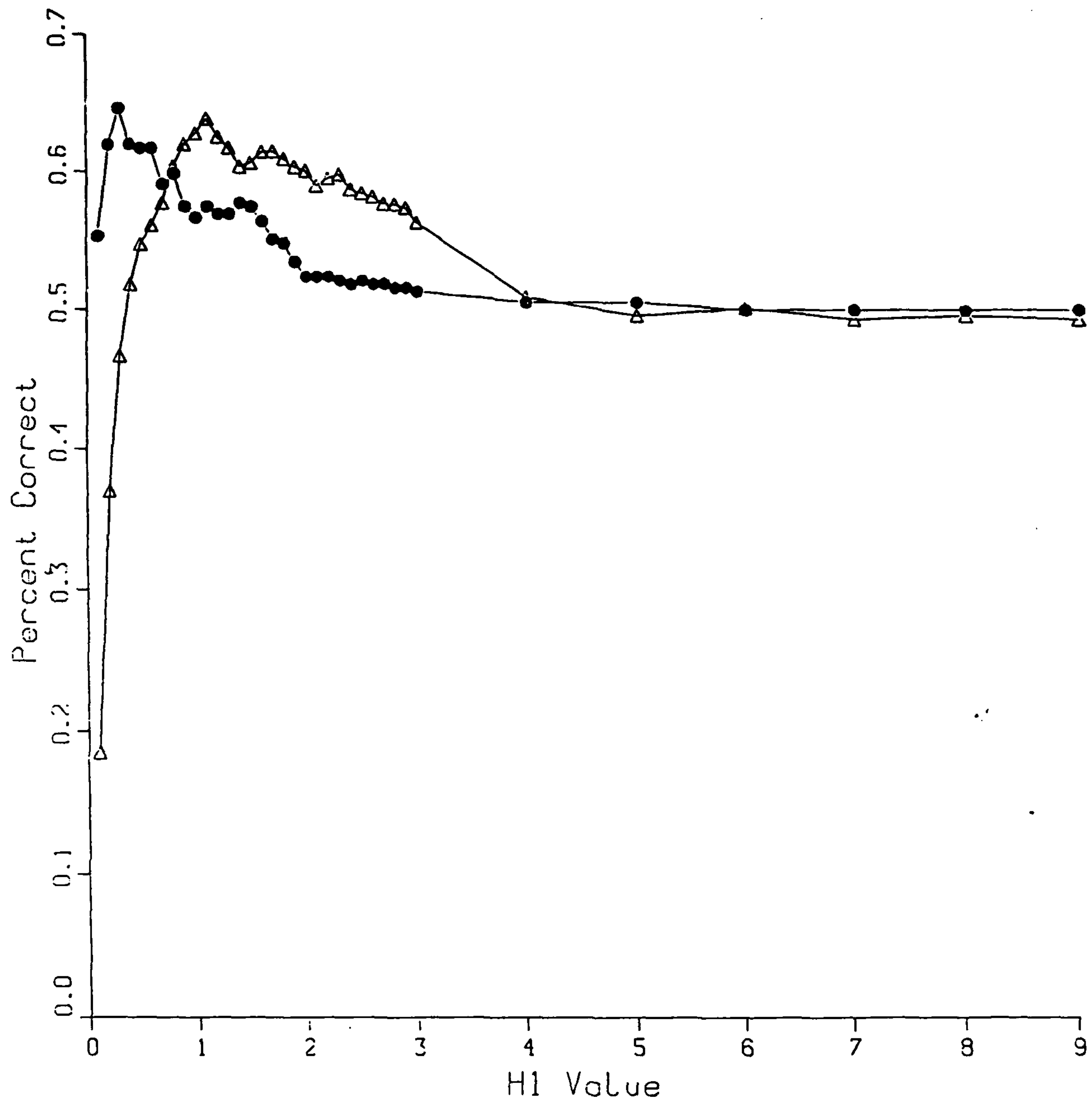


Figure 19: Percent Correct Classification versus h for both the development and characterization sets over the two-class, nine-feature problem – Larger values of h

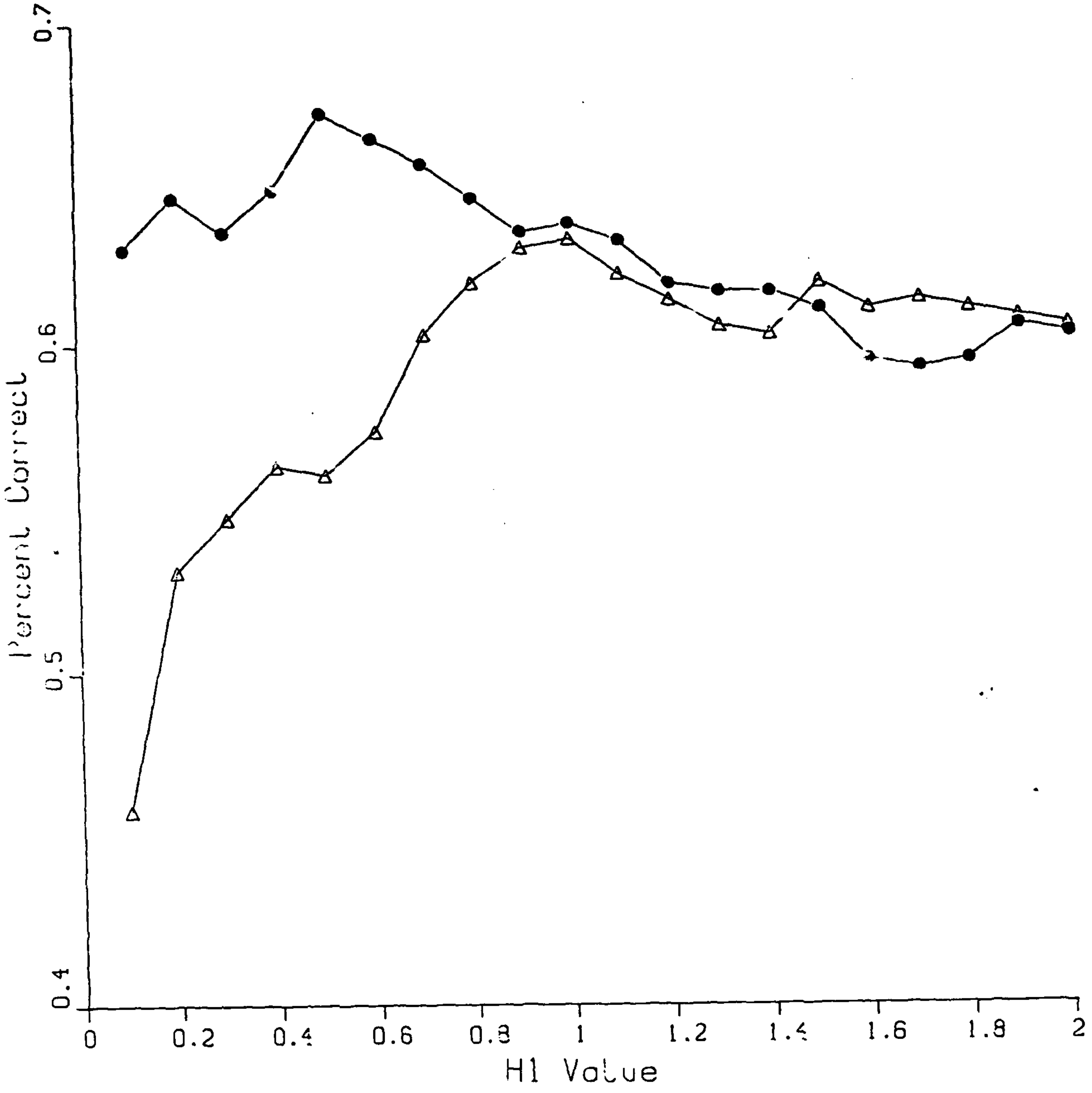


Figure 20: Percent Correct Classification versus h for both the development and characterization sets over the two-class problem with features 2, 5, and 6

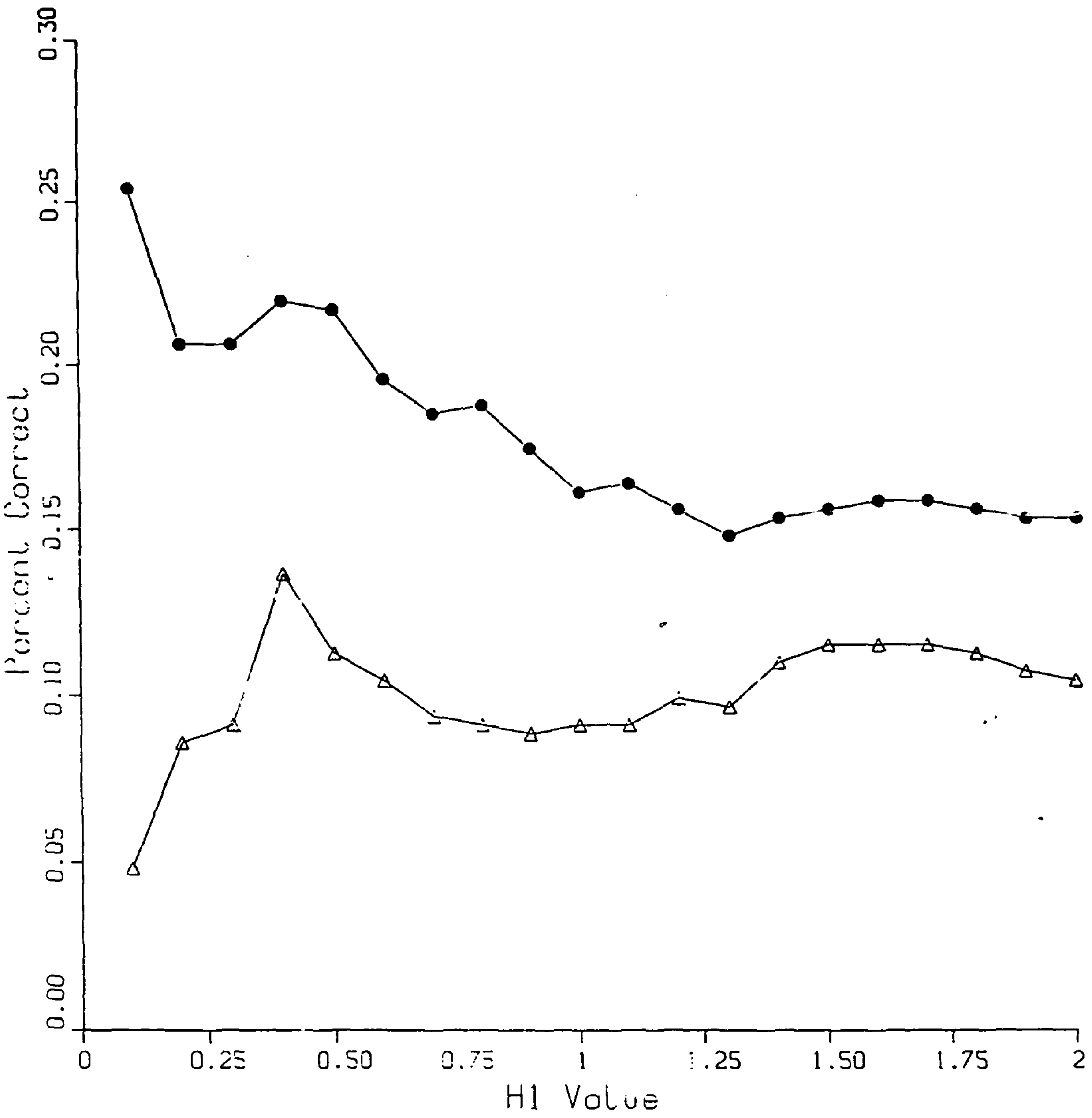


Figure 21: Percent Correct Classification versus h for both the development and characterization sets over the fourteen-class, nine-feature problem

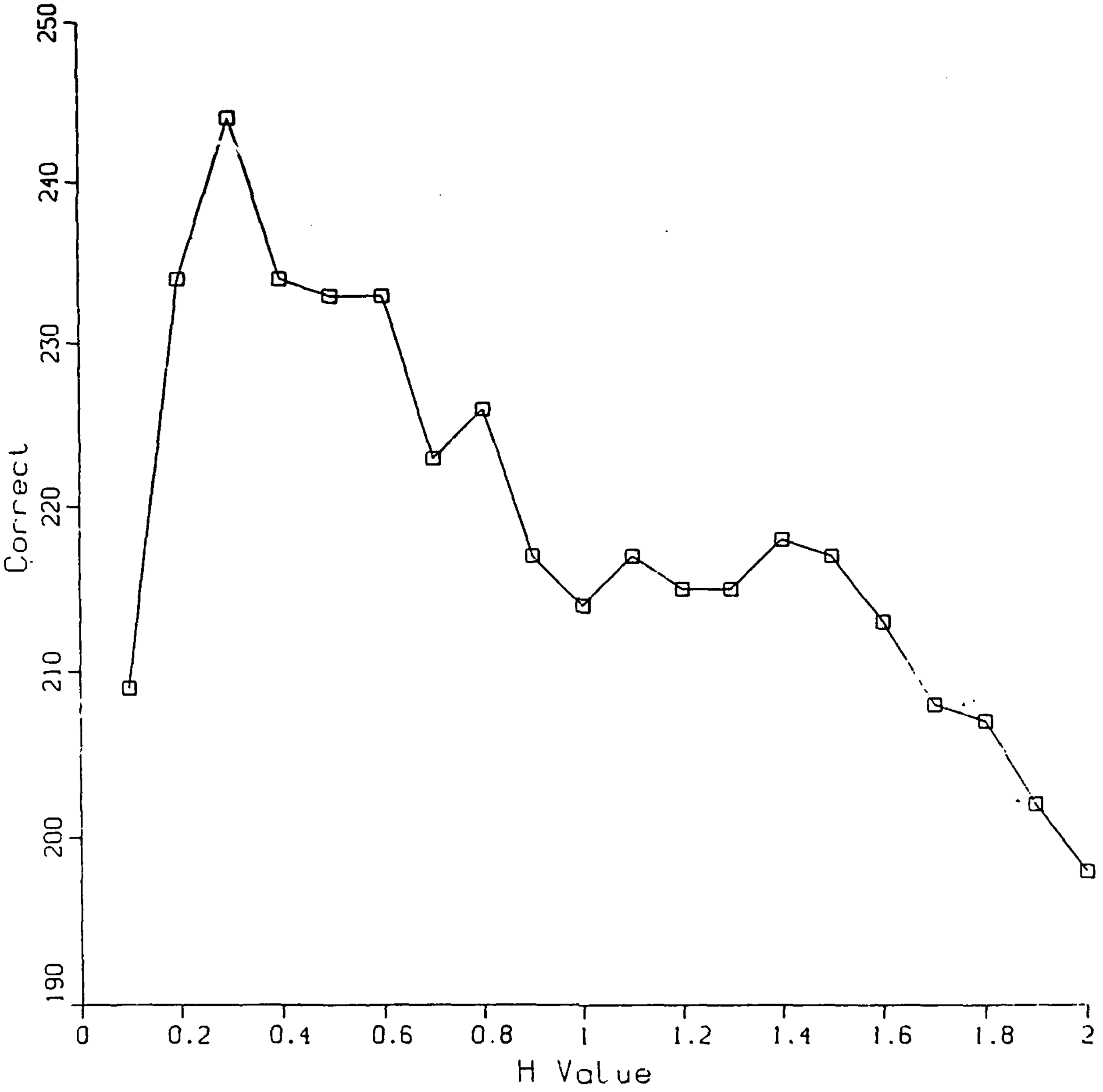


Figure 22: Number of Correctly Classified Samples versus the window-width parameter for the kernel-classifier over the development set

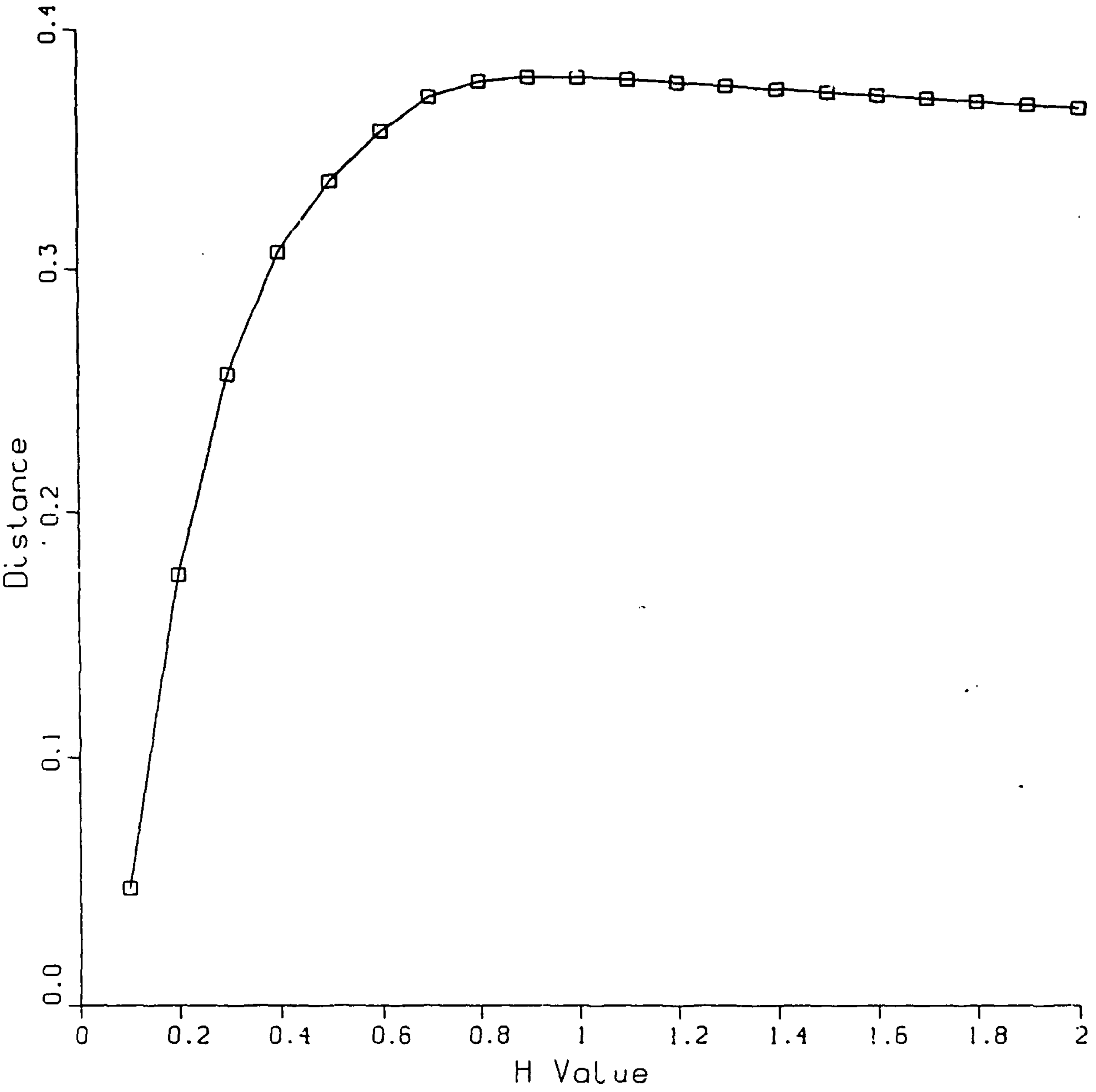


Figure 23: Bhattacharya distance versus *h* for the development set data

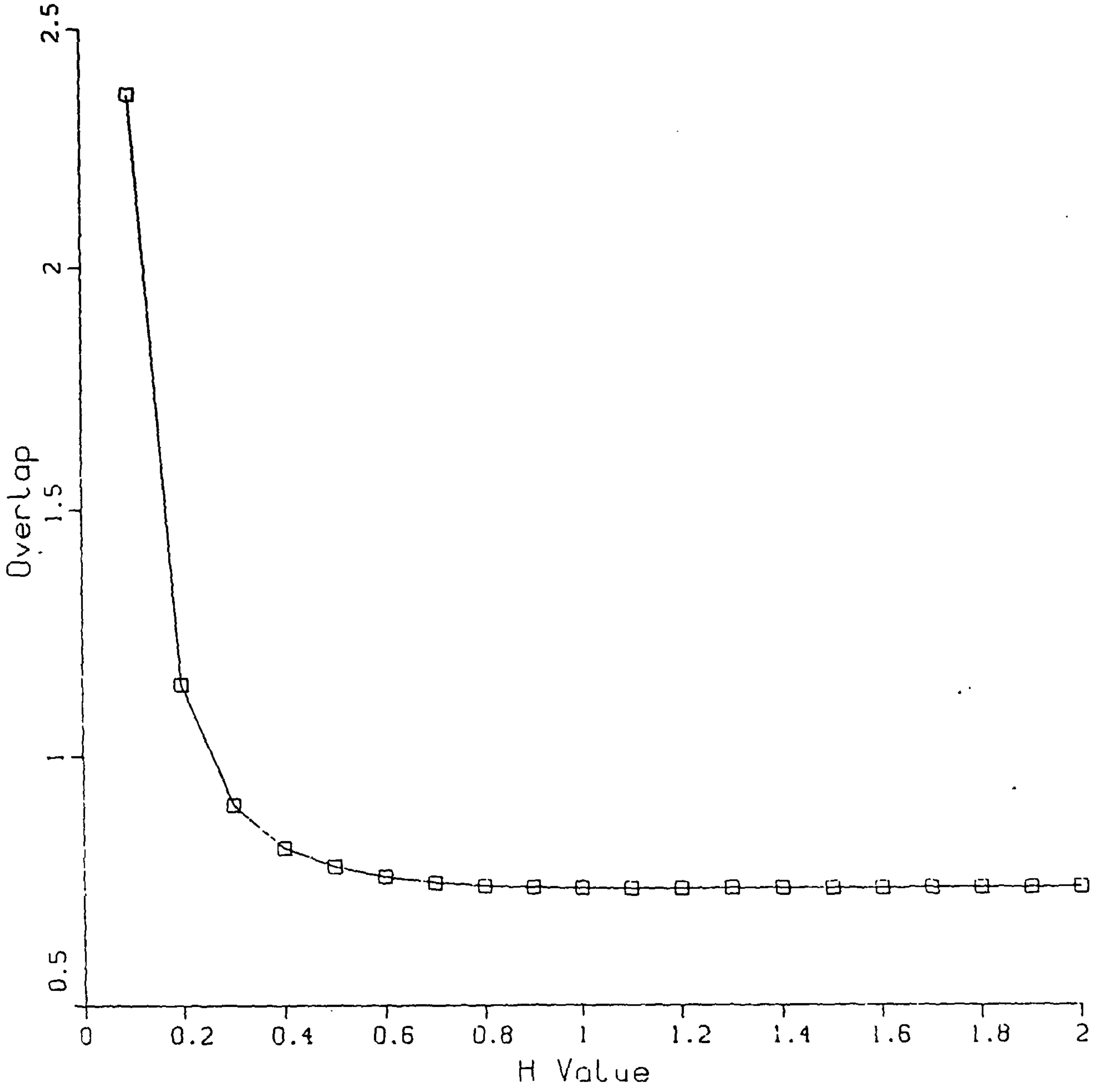


Figure 24: Overlap metr'c versus h for the development set data

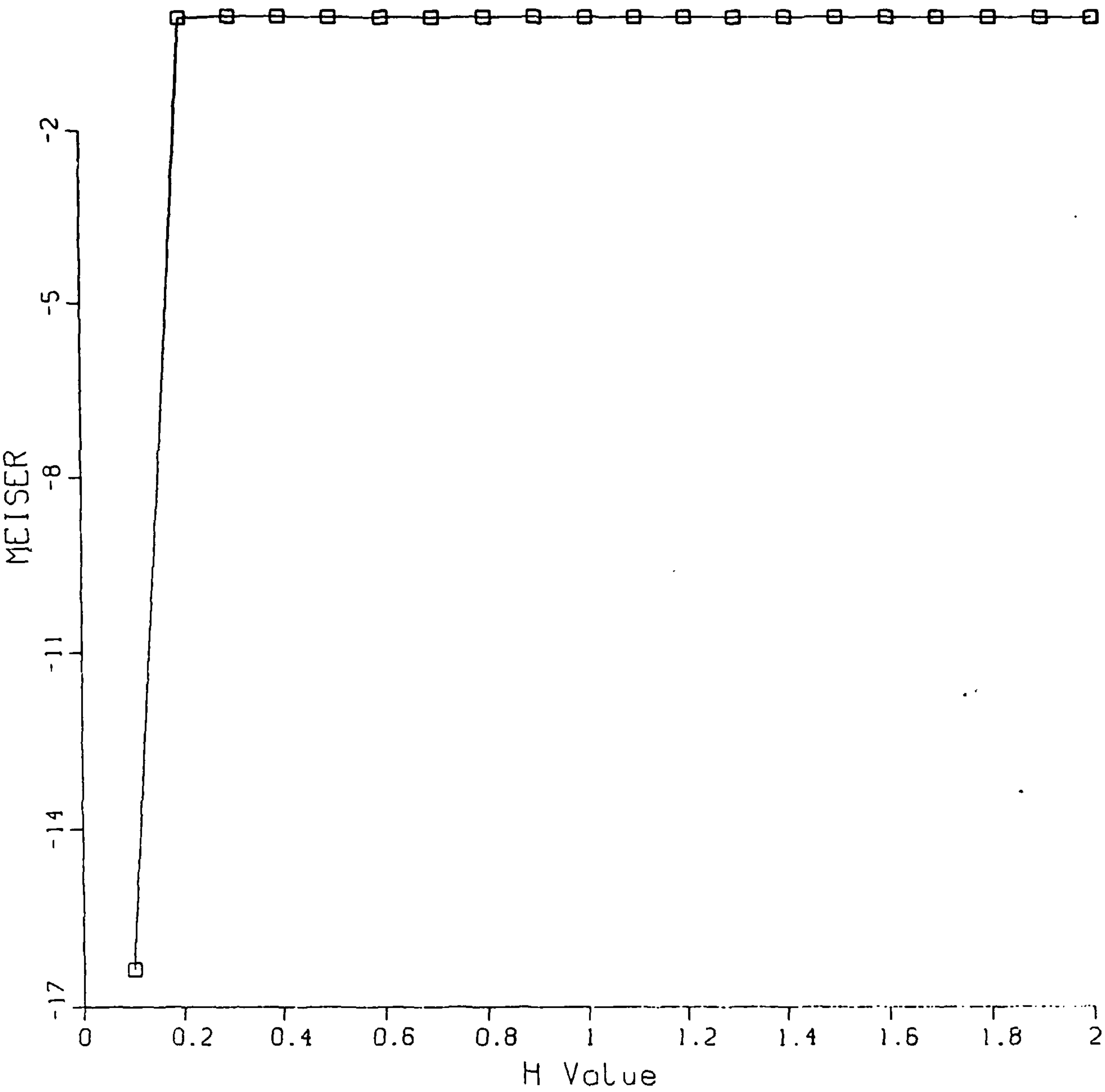


Figure 25: MEISER criterion versus h for the development set data

Figure 25 shows a different behavior. This graph plots the values of the MEISER criterion over over reasonable values of h. In this figure, it is easy to see that the MEISER criterion is not achieving its minimum in a sensible range of h. Additional runs have confirmed that, for this data, the MEISER criterion goes to $-\infty$ as $h \rightarrow 0$. While the MEISER criterion has performed sensibly on other data, here the leave-one-out term dominates the convolution term, sending the result to $-\infty$. Thus some care must be taken in depending on the MEISER results.

7.2 Characterization Set

To get the true value of the window-width metrics, we now look at the results of setting h using the value predicted by these metrics and then testing this classifier on the characterization set. We do this for the two-class problem using all nine features and using the feature subset with features 2, 5, and 6.

For the two-class problem with all nine features, the leave-one-out classifier had the best performance with $h = 0.3$. As we have shown in a previous section, this setting resulted in overfitting the classifier to the development set. Examination of Figures 18 along with Figures 24 and 23 shows that using the setting of h predicted by either the overlap metric or the Bhattacharya metric results in better results on the characterization set.

The Bhattacharya window-width optimization method predicted an h value of 0.9, while the overlap window-width technique predicted a value of 1.1. The error rate estimated by the leave-one-out technique for $h = 1.1$ was 42.6, while the actual error rate on the characterization set was 36.2 for this value of h. This produced a −6.4 percent change. Similarly, at $h = 0.9$, the leave-one-out technique predicted an error rate of 42.6 while the actual error rate on the characterization set was 38.1, resulting in a −4.5 percent change. Thus, the use of these window-width optimization metrics does tend to avoid overfitting of the training data.

For this same classification problem, but using the feature subset with features 2, 5, and 6, Figure 20 shows the development set (filled-in circles) and the characterization set (hollow triangles) percent correct classifications as a function of h for the kernel classifier. Note that values of h below 0.9 result in overfitting the classifier to the development data. However, the best leave-one-out performance on the development set is at $h = 0.5$, indicating that the leave-one-out selection of h tends to overfit the data.

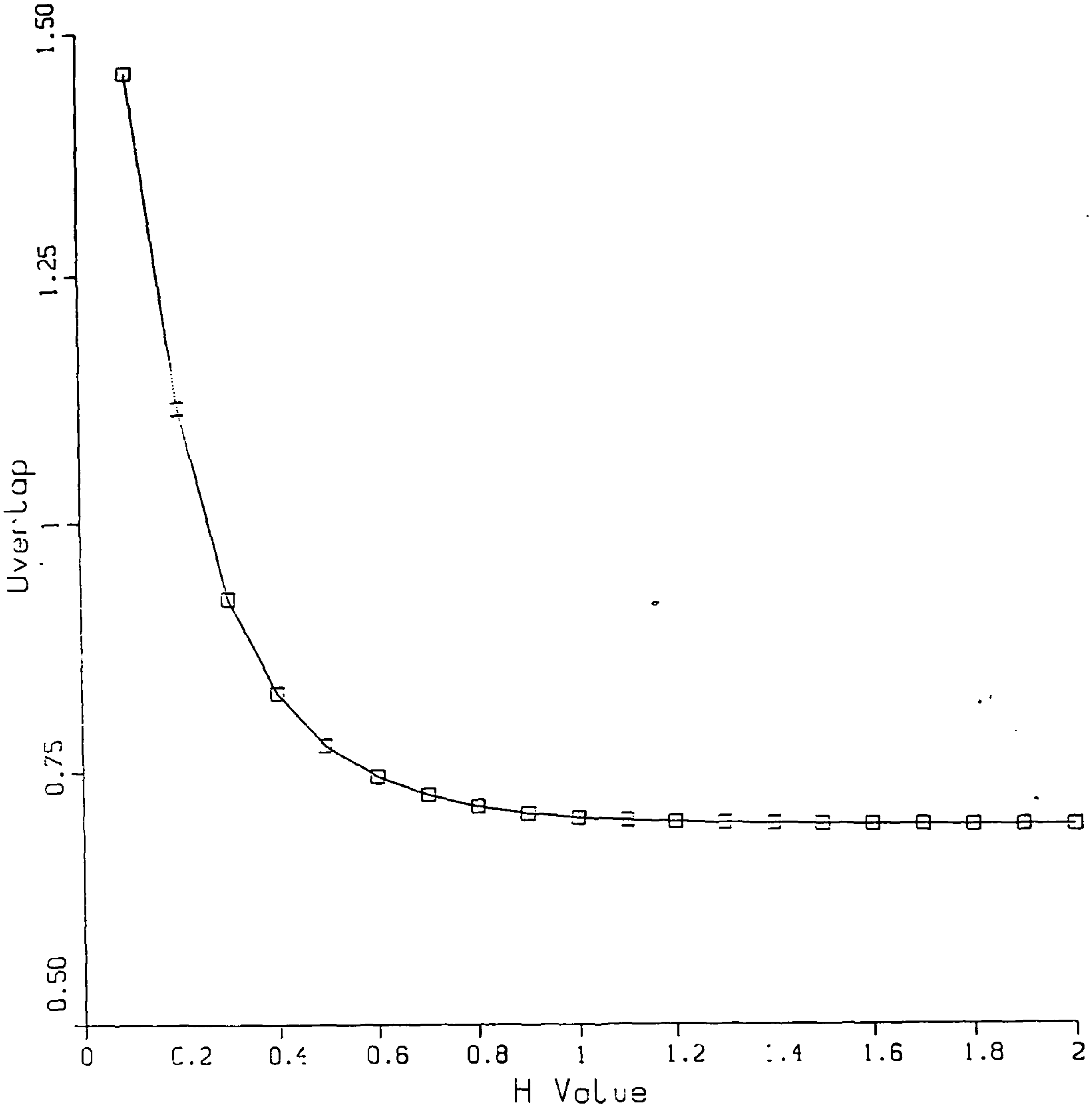


Figure 26: Overlap Metric Values versus *h* for the two-class problem with features 2, 5, and 6

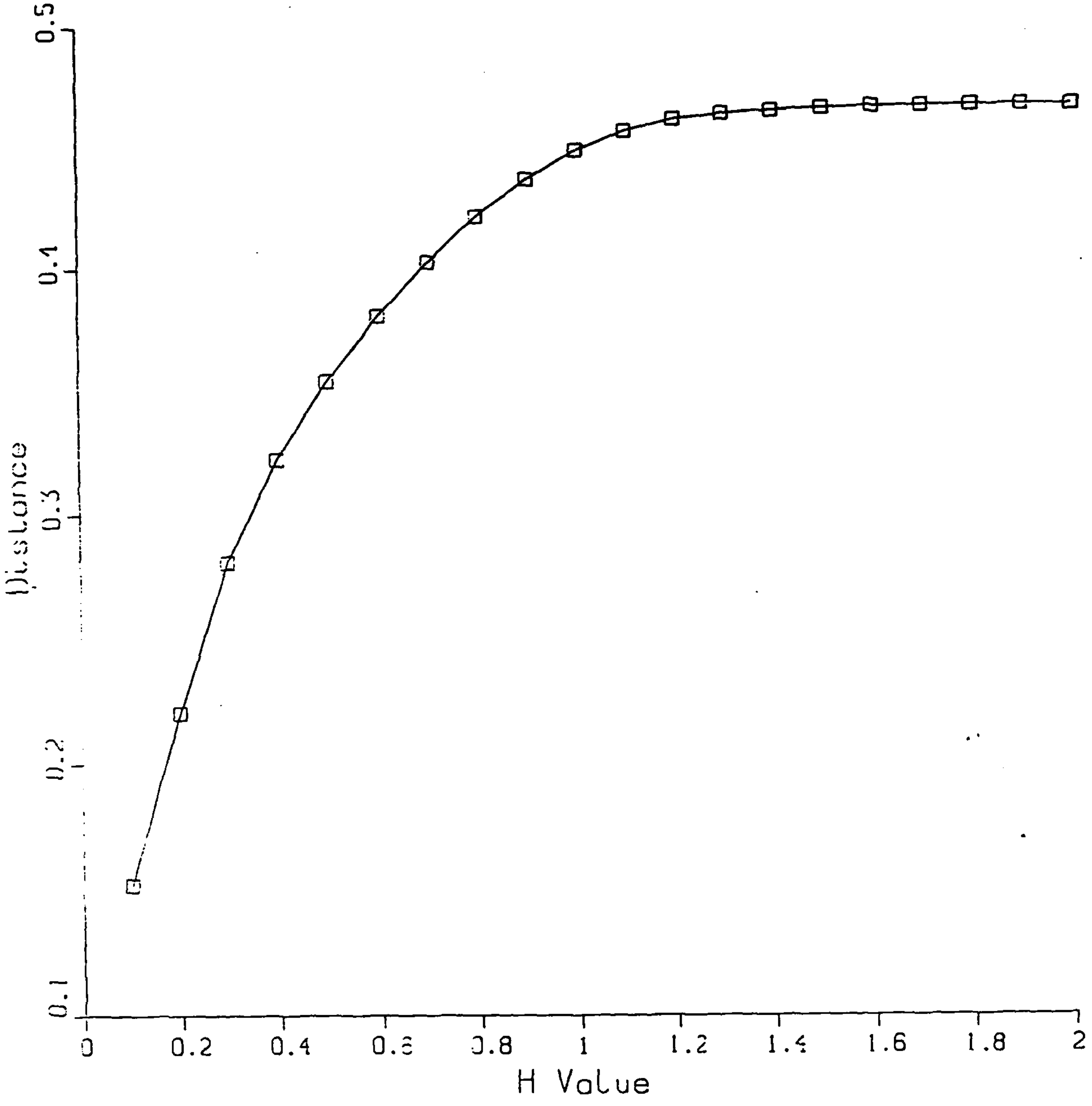


Figure 27: Bhattacharya metric values versus *h* for the two-class problem with features 2, 5, and 6

Classifier	Parameters	Error Rates		
		Training	Testing	Percent Change
Kernel	$h = 1.1$	42.6	36.2	-06.4
KNN	$K = 8$	42.6	48.8	+06.2
Gaussian	N/A	37.3	38.3	+01.0

Table 17: Error rates for two-class problem using all nine features

Classifier	Parameters	Error Rates		
		Training	Testing	Percent Change
Kernel	$h = 1.1$	36.8	37.8	+01.0
KNN	$K = 6$	47.9	49.3	+01.4
Gaussian	N/A	39.7	42.1	+02.4

Table 18: Error rates for two-class problem using features 2, 5, and 6

Figures 26 and 27 show, for the same data, h versus the overlap metric and the Bhattacharya metric, respectively. While neither achieves their minimum (maximum) over the graphed range of h, it is clear that both suggest that values of h below 1.0 will result in data set overfitting. Again, these two window-width optimization methods are useful for determining a good value of h. Though not plotted, the MEISER criterion on this data was again not useful as it suggested choosing as small an h as possible.

We can now repeat Tables 13 and 14 but with the values of h selected by the overlap window-width estimation technique. These repeated tables are shown in Tables 17 and 18, respectively. As can be seen, the kernel-based classifier now avoids overfitting the training data and gives the best performance among the classifiers on the characterization set of data.

8 Confidence Measure Experiment

In this experiment, the best classifier found for the development set for the two-class problem (the kernel-based classifier with $h_1 = 0.5$ and feature subset consisting of features 2, 5, and 6) was used to study our proposed confidence measures.

Figure 28 shows the results of the experiment. Each pattern in the development set is plotted using its non-linear confidence estimate (x-axis) and its bootstrap confidence estimate (y-axis). The plotted symbol of a pattern is a filled-in circle if it was correctly classified (by the kernel-based classifier mentioned above) and a hollow triangle symbol if the pattern was incorrectly classified. Any pattern which was "undecided", i.e. had equal probabilities for the two classes, is shown in the graph as an asterisk.

Figure 28 shows a slight trend in the relationship between confidence measure values and classifier performance. Correctly classified patterns do tend to group around the +1 and −1 values of the bootstrap confidence measure. However, unclassified patterns appear randomly spread in confidence measure values.

To further quantify this trend, we performed a Mann-Whitney non-parametric test on the confidence measure values. Let G be a randomly chosen correctly classified sample and let E be a randomly chosen incorrectly classified sample. Let $nls(G)$ and $nls(E)$ be the non-linear scaling confidence measures for G and E, respectively, and let $cboot(.)$ be the corresponding bootstrap confidence measure. For the non-linear scaling confidence measure, the small (close to zero) values seem to be more highly probable for correctly classified samples so the null hypotheses is set to be $P[nls(G) \geq nls(E)] \leq 1/2$ with alternative hypothesis $P[nls(G) < nls(E)] > 1/2$. For the bootstrap confidence measure, values close to 1 and −1 appear to be more likely for correctly classified samples. The null hypothesis for the bootstrap confidence measure is then $P[abs(cboot(G)) \leq abs(cboot(E))] \geq 1/2$ with alternative hypothesis $P[abs(cboot(G)) < abs(cboot(E))] < 1/2$.

The Mann-Whitney statistic for the non-linear scaling confidence measure is 13,109 which, for this size data set, implies that small values of the non-linear scaling measure are more likely for correctly classified samples than for incorrectly classified samples, but only with a confidence of approximately 0.95.

The Mann-Whitney statistic for the bootstrap confidence measure is 17,505 which, for the size of this data set, implies that absolute values of the statistic are very likely (with confidence > 0.99) to be higher (nearer one) for correctly classified samples than for incorrectly classified samples.

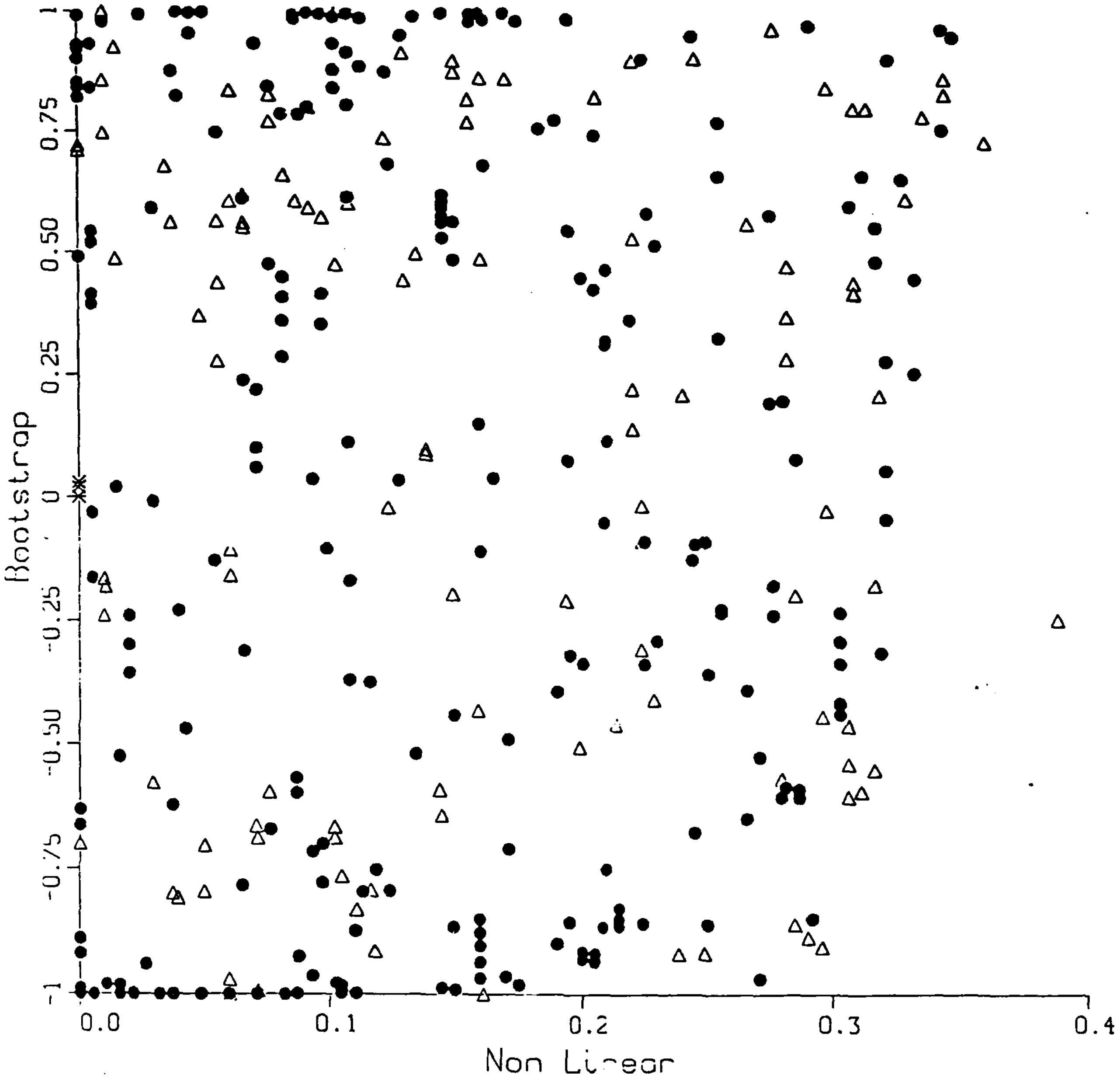


Figure 28: Scatter Plot of Non-Linear Scaling Confidence Measure versus Bootstrap Confidence Measure for the development set data

Thus, while neither confidence measure perfectly predicts the goodness of classification, both do capture important confidence information. It remains to be seen how to use this information correctly, however.

9 Conclusions

9.1 Summary

This study has evaluated three types of classifiers on a particular set of ATR data.

The ATR data consisted of two sets of images: one used to train the classifiers and one used to test the classifiers. For each target object in the ATR data, nine simple shaped-based features were computed and this full nine-dimensional feature space (along with a few sub-spaces of smaller dimension) was used as the basis for classifier evaluation.

The three classifiers chosen for evaluation were: a kernel-based classifier, the K-Nearest Neighbor (KNN) classifier, and a simple parametric Gaussian classifier. The error rate of each classifier was reported using the leave-one-out approach to error estimation over the training set and this error rate was compared to the actual error rate obtained on the test set.

In addition, two further evaluations were performed. The first involved the selection of the window-width parameter needed in kernel-based density estimation. As well as the baseline leave-one-out method of setting this parameter, three additional selection procedures were evaluated: the MEISER technique, a Bhattacharya technique, and an density overlap technique. The second study involved the definition and evaluation of confidence measures for estimated densities. Two measures were evaluated: a bootstrap measure and a non-linear scaling measure.

9.2 Features

We can state a very definite conclusion about the feature set we have chosen to study: *it is not very useful in determining classes in this ATR data.* Much evidence is available to support this conclusion, including the poor performance of all the classifiers over the perfect silhouettes used, as well as the simple feature analysis performed.

Since the binary mask images used in this study were "perfect" segmentations, it is clear that other—different or modified—features must be employed to achieve reliable classification performance. Any future evaluations of this ATR data should undertake the task of searching for other features, as well as looking more closely at the current feature set after it has been normalized by target metric information, such as target distance.

9.3 Classifiers

The conclusions one can draw from reviewing the classifier performance results of this study are not as sharp as those drawn for the feature set. One problem is the relatively poor performance of all the classifiers; the *classifiability* of the data with this feature set is low and this places a strong upper bound on the performance of any possible classifier.

Tables 17 and 18 present the heavily distilled results of our evaluation. These two tables show that the kernel-based classifier performs better on the test set than either of the other two classifiers.

The Gaussian classifier showed remarkably good performance on this data. Its advantages are that it has no *a priori* parameters and that it is quick and cheap (computationally) to invoke. For these reasons, we recommend its use for initial determination of data set classifiability.

The KNN classifier is non-parametric and is between the Gaussian classifier and the kernel-based classifier in computational complexity. Given the results of Tables 17 and 18 (and the fact that the kernel-based classifier is able to approximate the KNN classifier's decision regions when K is small by setting h small—see Figure 15), there is little reason to recommend use of the KNN classifier.

Among the three classifiers evaluated, the kernel-based classifier is the most complex in software coding and lengthy in execution time. The results of Table 17 and 18 show that it does perform the best, however.

The KNN classifier has one *a priori* parameter of importance, the value of K. In our evaluation, we have chosen the best value of K by a brute-force search for the value which minimizes the leave-one-out error estiamte over the training set. The kernel-based classifier also has one *a priori* parameter, the value of h, the window-width. One advantage of dealing with the kernel-based classifier is the existance of techniques for choosing this window-width—techniques which are more reliable than using the value which minimizes the leave-one-out error on the training set.

9.4 Window-Width Selection

From our window-width selection evaluation, it is clear that both the Bhattacharya and the overlap techniques should be applied for determining an appropriate window-width. The kernel-based classifier was shown to be robust when using either of these window-width selectors. If the window-width is set by using the usual technique based on minimizing the leave-one-out error over the training set, we have shown that the resultant classifier can overfit the training data.

The MEISER technique, though theoretically interesting, fails to provide useful window-width selection on this data. While it has proven useful on artificial data sets, it is clear that more careful studies are needed on distributions similar to the feature distribution of this ATR data to determine how and why the MEISER technique fails.

9.5 Confidence Estimators

The confidence estimators evaluated in this study appear interesting, as they do yield some additional information—though not predicative power—about classifier performance. Their evaluation is difficult, as they were initially proposed to *combine* the output of a few subservient classifiers at a higher level. However, the Mann-Whitney test performed shows precise evidence for the conclusion that the bootstrap technique shows more promise than the non-linear scaling technique.