# ECE 662 Spring 2008
# Homework 2

xxxxxxxxxxxx
Purdue University
xxxxxx@purdue.edu

xxxxxxxxxxxx
Purdue University
xxxxxxx@purdue.edu

April 17, 2008

## 1 Problem 2 and 3: Methodology

For both problem 2 and problem 3, we use a common dataset and a common toolkit to run our experiments.In this section, we briefly describe the toolkit and the dataset used by us for the experiments. We then describe our attribute selection procedure in detail.

### 1.1 Toolkit

We use Weka 3 - a data mining software written in Java [7]. "Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes" [4]. Weka's main graphical user interface, the Explorer, gives access to all its facilities using menu selection and form filling. It is illustrated in Figure 1. We use the Explorer interface for this homework to experiment with the various machine learning algorithms.

### 1.2 Dataset

We use a dataset available at UCI Machine Learning dataset repository [5]. The dataset is called "Pen-Based Recognition of Handwritten Digits" [2]. It is a digit database created by collecting 250 samples from 44 writers. The dataset attributes are the digits represented as constant length feature vectors. The samples written by 30 writers are used for training, cross-validation and writer dependent testing,

and the digits written by the other 14 are used for writer independent testing. The original dataset contains 10 classes. We use only 3 of these classes (the first 3) for the purpose of better visualization. Following are the salient characteristics of this dataset:

```
1. Number of Instances
    Training    2339
    Testing     1091

2. Number of Attributes
    16 input + 1 class attribute

3. For Each Attribute:
    All input attributes are integers in the range 0..100.
    The last attribute is the class code 0,1,2.

4. Class Distribution
    Class: No of examples in training set
    0:     780
    1:     779
    2:     780
    Class: No of examples in testing set
    0:     363
    1:     364
    2:     364
```

### 1.3 Attribute Selection

We have 16 attributes in our dataset. We perform our experiments using all the attributes in the dataser. For the purpose of better visualization, we present a plot matrix of all pairs of attributes(see Figure 2). The classes are represented with different colors. If we are asked to select the best 2 attributes to classify our dataset based on the plot matrix, which attributes would we choose? As we can see from the plot matrix, attributes $A15$ and $A16$ seem to give a "good" linear separation between the classes. On the other hand, the plot matrix for attribute $A10$ and $A13$ seem to suggest that a "good" separation between the classes will not be possible if we use only $A10$ and $A13$ for the classification. We confirm our intuition by performing experiments on these two pairs of attributes.

# 2 Problem 2 and 3: Experiments

## 2.1 Support Vector Classification

Weka implements John Platt's sequential minimal optimization algorithm for training a support vector classifier [6]. This implementation globally replaces all missing values and transforms nominal attributes into binary ones. It also normalizes all attributes by default. Table 1 gives the classification accuracy (CA) for various kernels using all 16 attributes in the dataset. The complexity parameter $C$ that control the error term in the SVM optimization problem (see equation 1 below) is set to 1 for all the kernels.

$$min\frac{1}{2}||w||^2 + C\sum_i \epsilon_i \tag{1}$$

such that

$$c_i(w.x_i - b) \geq 1 - \epsilon_i, 1 \leq i \leq n.$$

| **Kernel** | **Kernel Function** | **CA**(%) |
|---|---|---|
| Linear | $K(x,y) =< x,y >$ | 96.4 |
| Quadratic | $K(x,y) = (< x,y >)^2$ | 97.6 |
| Radial Basis Function (RBF)($\gamma = 1$) | $K(x,y) = \exp^{-(\gamma * < x-y, x-y >^2)}$ | 98.6 |

Table 1: SVM results for different kernels using all 16 attributes

We see that the RBF kernel performs the best on our dataset with $98.6\%$ accuracy although the linear and quadratic kernels are not bad either with $96.4\%$ and $97.6\%$ accuracy respectively. This is probably because of the fact that the number of attributes (dimensions) is already large enough $(= 16)$ such that mapping the data to higher dimensions using the quadratic or the RBF kernel does yield significantly better results. We confirm this intuition by performing experiments on a lower dimensional dataset. Table 2 presents the results for the attribute pairs $(A15, A16)$ and $(A10, A13)$ identified in Section 1.3.

| **Kernel** | $A10, A13$ **CA**$\%$ | $A15, A16$ **CA**$\%$) |
|---|---|---|
| Linear | 60.9 | 83 |
| RBF($C = 1, \gamma = 1$) | 69.1 | 83.2 |

Table 2: SVM CA for different kernels using attributes $A15, A16$ and $A10, A13$

The results in Table 2 present some interesting observations. The scatter plot 2 for the attribute pair $A15, A16$ shows good linear separation between the classes. This is confirmed as the performance of SVM with linear kernel is almost as good as that of the RBF kernel for $A15, A16$. Moreover, we observe that the

classification accuracy for $A10, A13$ using the RBF kernel is much higher at $69.1\%$ than the linear kernel. This highlights the usefulness of the SVM classifier for obtaining non-linear separation between classes.

There are two parameters while using RBF kernels: $C$ and $\gamma$. It is not known beforehand which $C$ and $\gamma$ are the best for a given problem; consequently some kind of model selection (parameter search) must be done [3]. Unfortunately, Weka (unlike libsvm [1]) does not provide any built-in methods for doing an automatic search for these parameters. We present some results on tuning the $C$ parameter while keeping $\gamma$ fixed at $= 1$. Figure 3 shows the CA for increasing values of $C$ for all $16$ attributes. Note that the parameter $C$ controls the weight of the error term in 1. We observe that increasing value of $C$ till $C = 20$ increases the classification accuracy till $99.2$, and remains constant after that.

## 2.2 Neural Network Classification

A neural network in Weka uses backpropagation to classify instances. The nodes in this network are all sigmoid (except for when the class is numeric in which case the the output nodes become un-thresholded linear units). The results for different network configurations are presented in Figure 4. We note that the best classification accuracy of $97.6\%$ is obtained with the most simple network configuration of 1 hidden layer with 2 nodes per hidden layer.

Table 3 gives the classification accuracy for $4$ different configuration of the neural network for the attribute pairs $(A15, A16)$ and $(A10, A13)$. Network configuration of 1 hidden layer and 3 nodes for the hidden layer gives the best classification accuracy for $(A10, A13)$ attribute pair, while the CA for $(A15, A16)$ attribute pair is approx. same for all configurations.

| **Network Configuration** | $A10, A13$ **CA(%)** | $A15, A16$ **CA(%))** |
|---|---|---|
| 1 hidden layer, 2 nodes | 66.6 | 82.8 |
| 1 hidden layer, 3 nodes | 71.6 | 82.4 |
| 2 hidden layers, 2 nodes/layer | 65.9 | 82 |
| 2 hidden layers, 3 nodes/layer | 70.66 | 83.3 |

Table 3: SVM CA for different kernels using attributes $A15, A16$ and $A10, A13$

## 2.3 K-nearest neighbors Classification

Figure 5 present the results for $k$-nearest neighbor classification with different values of $k$ for both euclidean and $l_{max}$ distance measures. Weka does not have

built-in support for the $l_{max}$ measure, so we implement that and integrate it with Weka's GUI (as shown in Figure 6). There are two observations from this experiment. First, the euclidean distance measure perform slightly better than the $l_{max}$ distance measure for higher values of $k$. For lower values of $k$ ($k \leq 2$), $l_{max}$ gives slightly better results. Second, the best classification accuracy is obtained for $k = 2$ at $98.4$ for the $l_max$ measure and the CA decreases consistently after that as $k$ increases. Weka has a built-in mechanism for determining the best value of $k$ using a cross-validation procedure on the training dataset. The best values of $k$ found out by Weka's cross-validation procedure is $k = 3$ which results in an $98.2\%$ classification accuracy.

Next, we use Weka's cross validation procedure to find out the best $k$ for dataset containing the attribute pair $(A10, A13)$ and $(A15, A16)$. Best $k$ for $(A10, A13)$ is found to be $k = 9$ resulting in an $72.13\%$ classification accuracy. For the attribute pair (A15,A16), the best $k$ comes out to be $k = 20$ with a classification accuracy of $82.5\%$.

## 2.4 Parzen window classification

Weka does not contain an implementation of the Parzen window classifier. So we implement that and integrate it with Weka's GUI (as shown in Figure 7). The implementation uses a hypercube kernel for the volume parameterized by the side length $h$. The experimental results for different values of $h$ for this shown in Figure 8. The maximum classification accuracy of $94.2\%$ is achieved at $h = 40$. This potentially makes parzen window the worst performing classifier among all classifiers tested in this homework. Note that the kernel used by us is the hypercube kernel which is not the most recommended kernel of all, but it is the easiest to implement. Better results may be possible by using other kernel such as gaussian.

## 2.5 All Classifiers

In this section, we compare the classification accuracy of all classifiers discussed in this homework. Table 4 lists the best classification accuracy of each classifier considering all attributes in the dataset.

| **SVM** ($C = 10, \gamma = 1$) | **ANN** (1 hidden layer with 2 nodes) | **k-nn** ($k = 2$) | **parzen window** ($h = 40$) |
|---|---|---|---|
| 99.2 | 97.6 | 98.2 | 94.2 |

Table 4: Classification Accuracy for all classifiers using all attributes in the dataset

Not surprisingly, SVM gives the best results among all classifiers. $k$-nn comes close second with $k = 2$ and $98.2\%$ classification accuracy which explains the popularity of the $k$-nearest neighbor approach. Not only it gives results that are comparable to the best possible results available using other classifiers, it is easy to implement, needs tuning for only a single parameter $k$ (that can tuned using cross-validation) and is gives fast results if a $kd$-tree (or similar) technique is used to store the distance information among the instances.

# 3 Problem 1

In this problem, we wish to compare the classification ability of the linear separating hyperplanes obtained by optimizing two different cost functions. The separating hyperplanes are to be used for two-class classification problem. One cost function simultaneously maximizes the between-class variance and mini-mizes the within-class variance for the two classes. This is the Fisher's Linear Discriminant (FLD) cost function. The other cost function just maximizes the between-class variance. For this problem also, we have used a pendigits dataset introduced in Section 1.2. Since our problem is related to linear discriminant functions, we want a 2-class dataset that is approximately linearly separable. We investigated the dataset and found that for classes $1$ and $4$, the features $1$ and $8$ are approximately linearly separable in 2D and the features $1$, $8$ and $15$ are approximately linearly separable in 3D (Figure 9(a) and 9(b)). Classes $1$ and $4$ have $780$ and $719$ training instances and $363$ and $336$ testing instances respectively.

First we considered the 2D case. We calculated the overall scatter matrix as

$$S_W = \sum_{x \in C_1} (x - m_1)(x - m_1)^T + \sum_{x \in C_2} (x - m_2)(x - m_2)^T$$

and the projection direction as $w_{opt} = S_W^{-1}(m_1 - m_2)$. $m_1$ and $m_2$ are the means of training data from the two classes. Optimizing the FLD cost function gives us the optimal weight vector but in order to derive the discriminant function $(w^T x + w_0)$, we also need the bias term $w_0$. For this, we posed the FLD problem as a linear least squares problem (see [] Section 4.1.5) and obtained an expression for the bias of the form $w_0 = -w_{opt}^T m$ where $m$ is the mean of the total training dataset, given by

$$m = \frac{1}{N} \sum_{n=1}^{N} x_n$$

### 3.1 Results and Discussion

Figure 10(a) shows the $w_opt$ (black line) and the separating hyperplane (magenta line) that are obtained by optimizing the FLD cost function. From the figure, it is apparent that $w_opt$ is infact a good choice for maximizing the between-class variance and minimizing the within-class scatter for the two classes. When we do not include the within-class scatter in the cost function, we are effectively setting $S_W$ to identity matrix and in this case, $w_opt$ is simply $(m_1 - m_2)$ i.e. the optimal weight vector is along the line joining the two class means. For this case, the weight vector $w_opt$ and the separating hyperplane are shown in Figure 10(b). We refer to these as difference-of-mean weight vector and difference-of-mean hyperplane. From the figure, we can tell that in general these are not the best choices for the optimal weight vector and the separating hyperplane. Interestingly enough, when we used these hyperplanes to classify the test data from the two classes, the results were a little counter intuitive. The difference-of-mean hyperplane performed slightly better than the FLD hyperplane. The numerical results are summarized in Table 5.

| **FLD** | **Difference of Means** |
|---|---|
| 95.85 | 96.85 |

Table 5: Classification accuracy comparison in 2D

We had anticipated that FLD classifier will perform better than the difference-of-means classifier. The unexpected result may be due to the particular distributions of the two classes. We can definitely construct two class distributions where the FLD classifier performs better than the difference-of-means classifier. We demonstrate such a case later in our report. For right now, we proceed with discussing the performance comparison between the two classifiers on 3D dataset.

For the 3D case, we obtain $w_opt$ and $w_0$ for both types of classifiers, similarly to the 2D case. Figures 11(a) and 11(b) show the weight vector $w_opt$ and the separating hyperplanes for the two classifiers. In 3D also, the difference-of-means classifier performs better than the FLD classifier as is apparent from Table 6.

| **FLD** | **Difference of Means** |
|---|---|
| 96.57 | 98.14 |

Table 6: Classification accuracy comparison in 3D

A little earlier in our report, we had commented that in general we expect FLD classifier to perform superior to the difference-of-means classifier. The reason is that FLD cost function simultaneously optimizes two different criteria (maximize

between-class variance and minimize within-class variance) which is more logical. But there may be some data distributions for which the latter performs better. This is precisely the case with our selected real dataset. On the other hand, we can show that there exist data distributions for which FLD classifier will perform better. One such example is shown in Figure 12(a) and 12(b), where we have shown the FLD classifier and difference-of-mean classifier results for 2 class Gaussian distributions. The means of the Gaussian distributions are horizontally displaced from each other and their axes of maximum variance are parallel to each other and inclined at an angle with the horizontal. From the numbers in Table 7, we observe that FLD gives perfect classification ($100\%$ accuracy) for this dataset while difference-of-mean classifier yields $95.8\%$ accuracy.

| FLD | Difference of Means |
|---|---|
| 100 | 95.8 |

Table 7: Classification accuracy comparison in 2D for a mixture of gaussians

# References


[1] Libsvm – a library for support vector machines. http://www.csie.ntu.edu.tw/ cjlin/libsvm/.

[2] Pen-based recognition of handwritten digits data set. http://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits.

[3] A practical guide to support vector classification. http://www.csie.ntu.edu.tw/ cjlin/papers/guide/guide.pdf.

[4] Weka - data mining with open source machine learning software in java. http://www.cs.waikato.ac.nz/ml/weka/.

[5] A. Asuncion and D. Newman. Uci: Machine learning repository, 2007.

[6] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. pages 185–208, 1999.

[7] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2nd Edition, 2005.

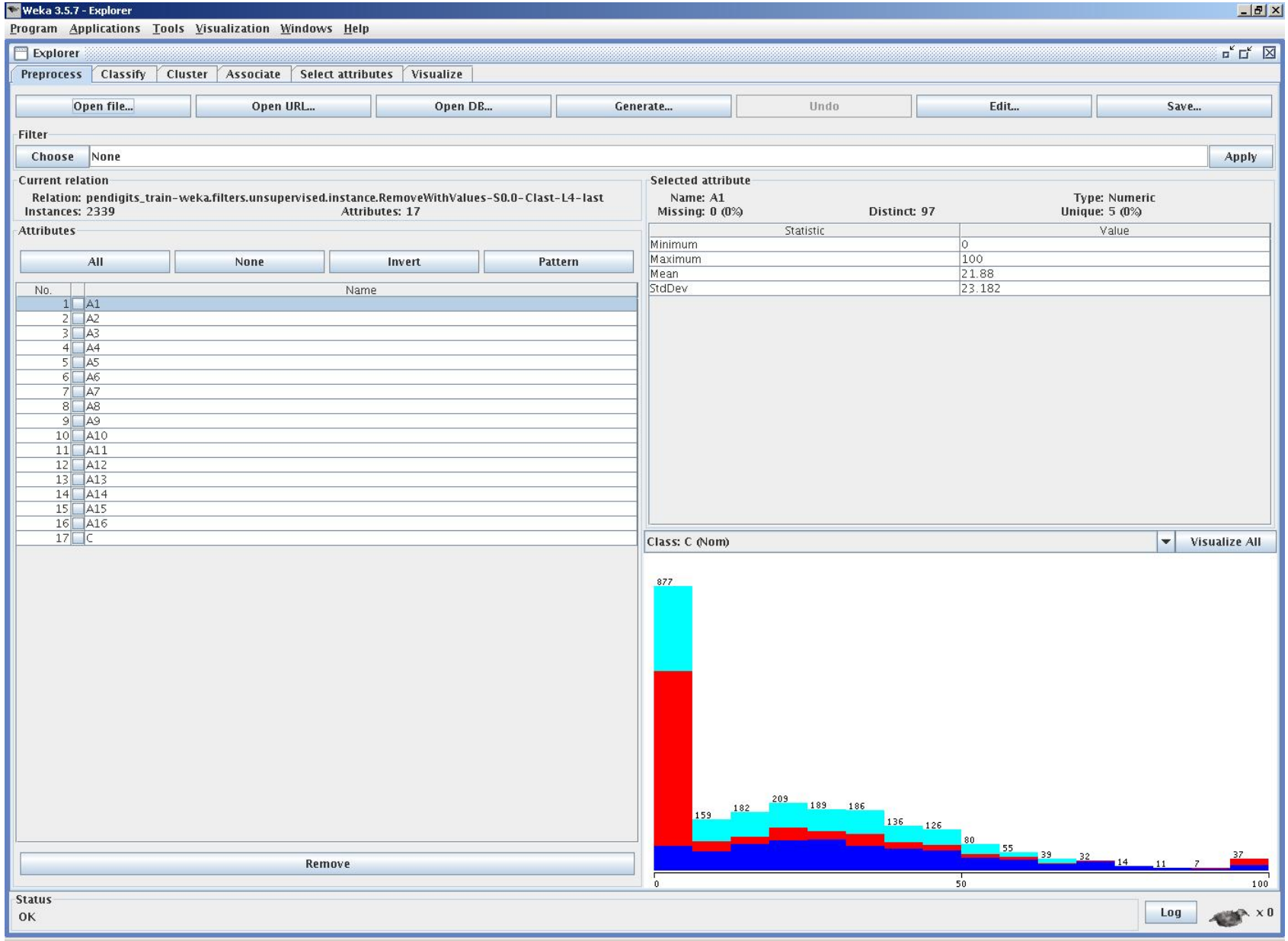



Figure 1: Weka - Explorer Interface

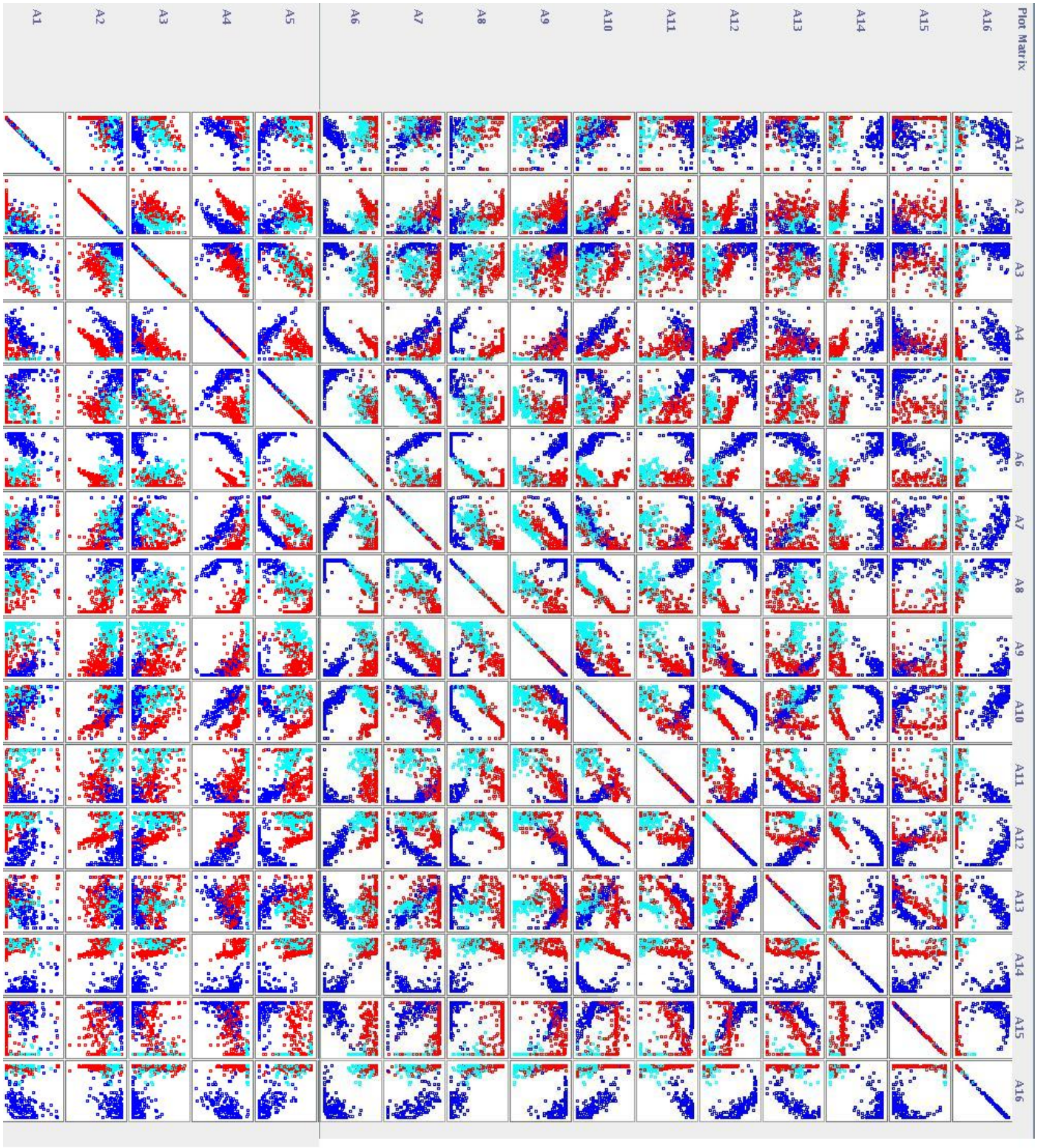





Figure 2: Plot matrix for the dataset

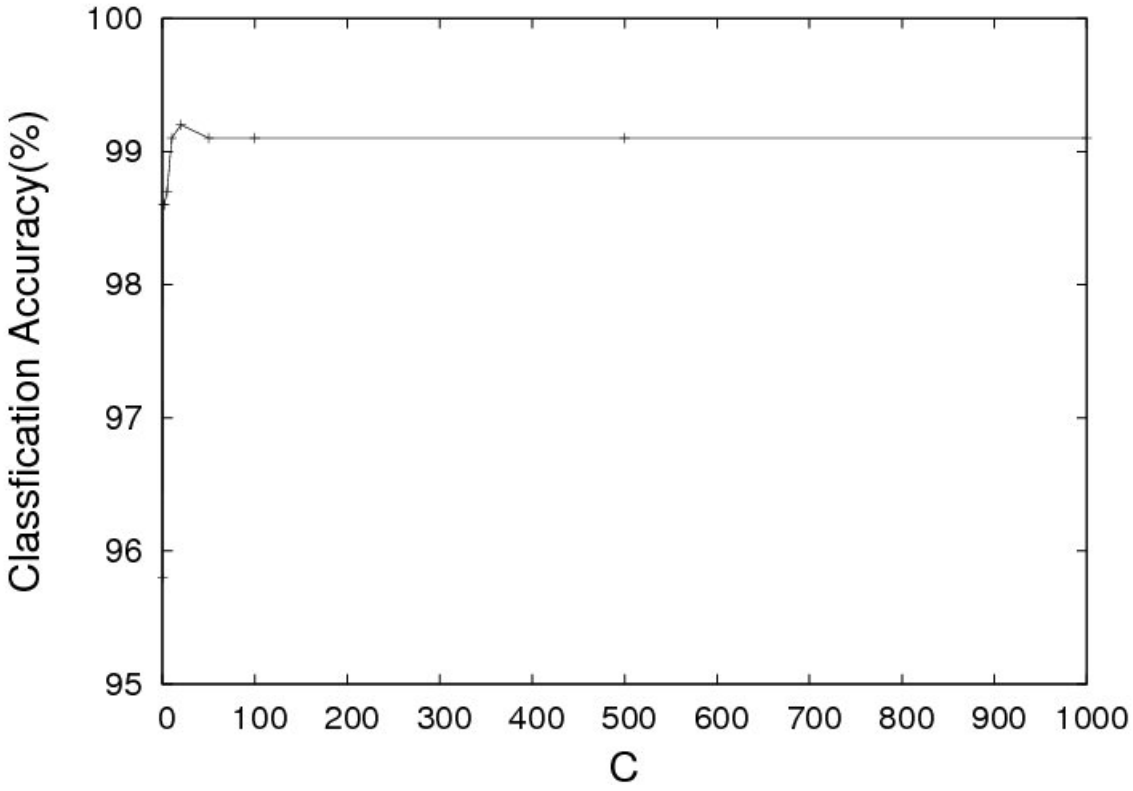



Figure 3: Classification accuracy for SVM for increasing values of $C$

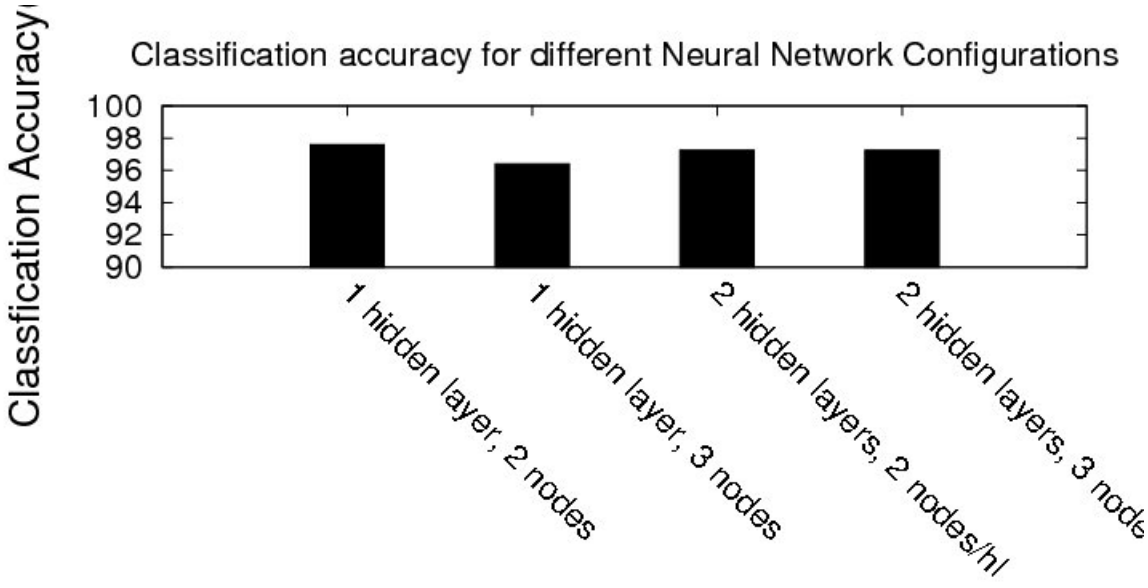



Figure 4: Classification accuracy for ANN with different network configurations

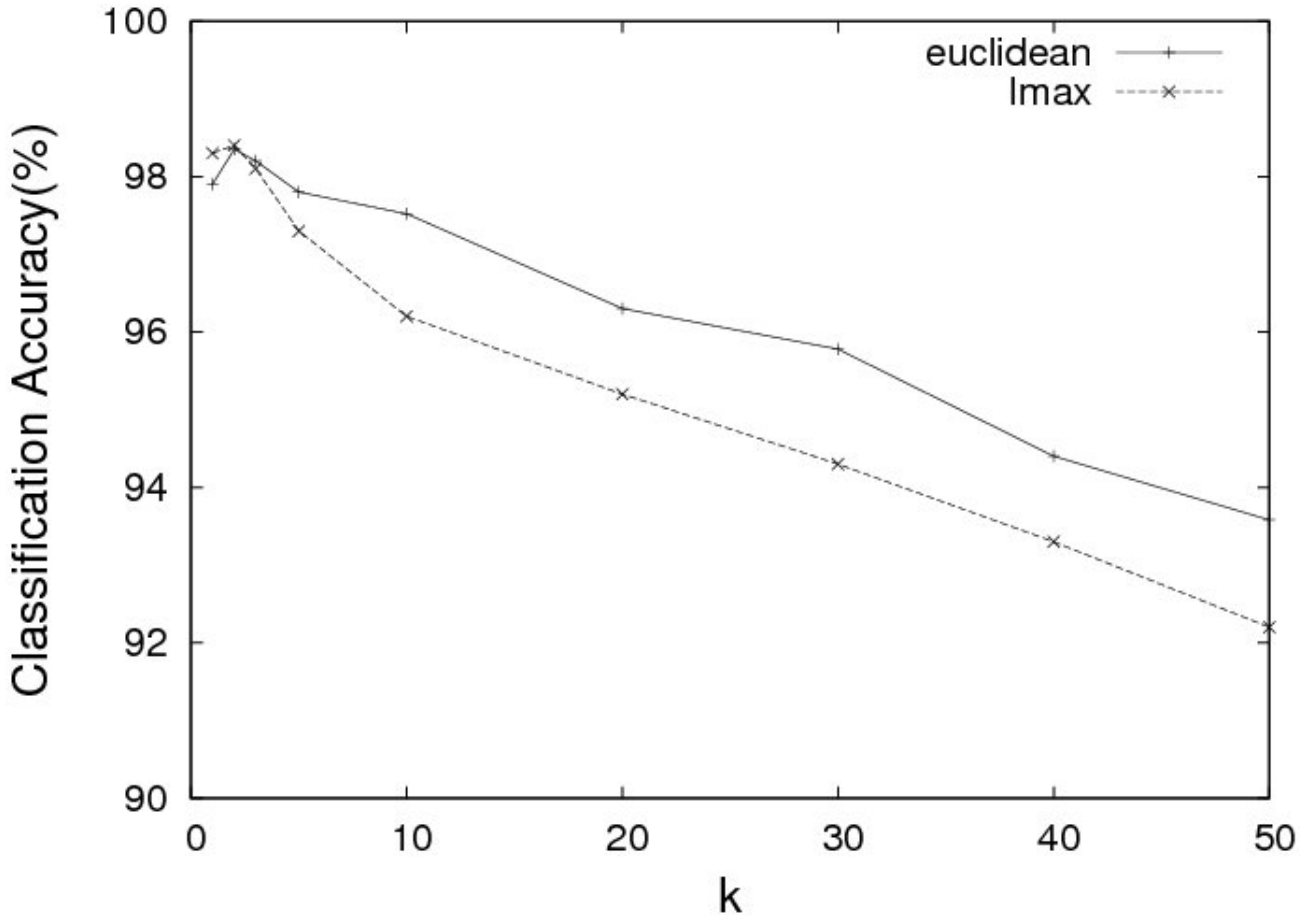



Figure 5: Classification accuracy for $k$-nearest neighbors

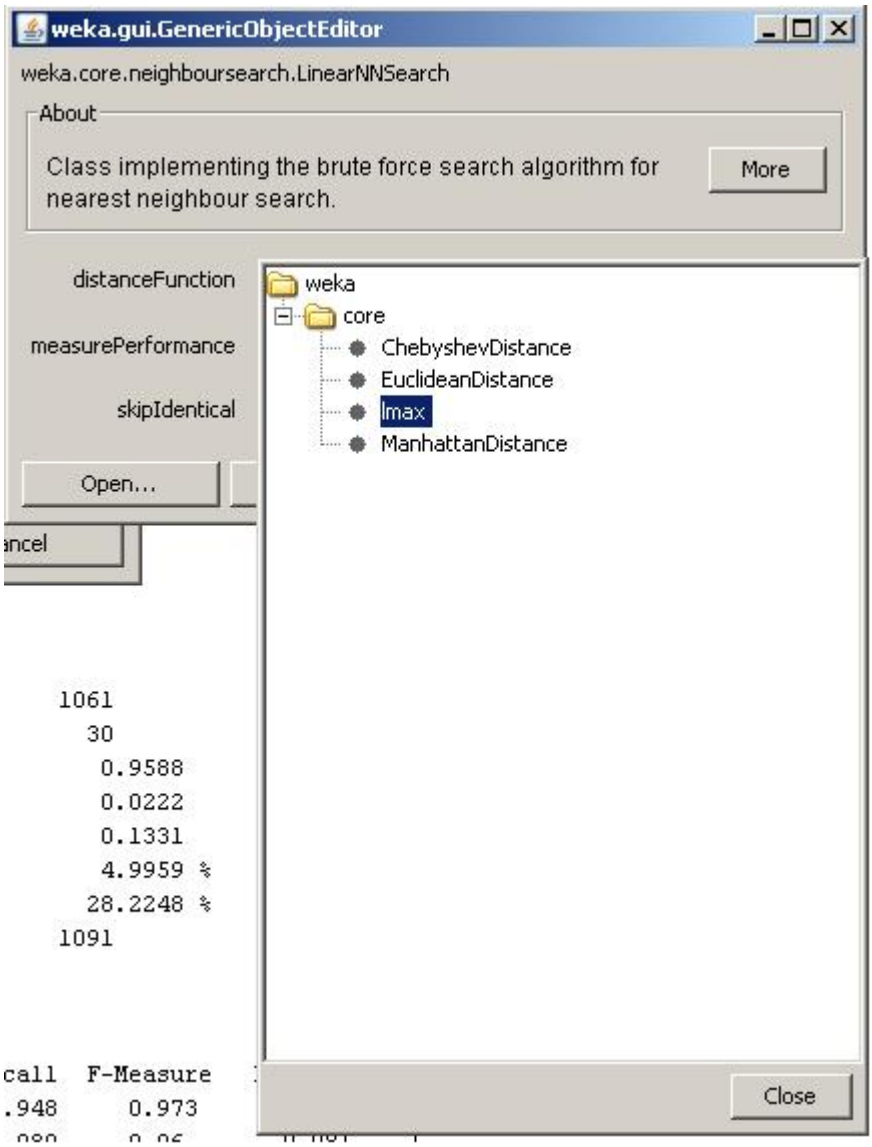



Figure 6: $l_{max}$ implemented in Weka's GUI

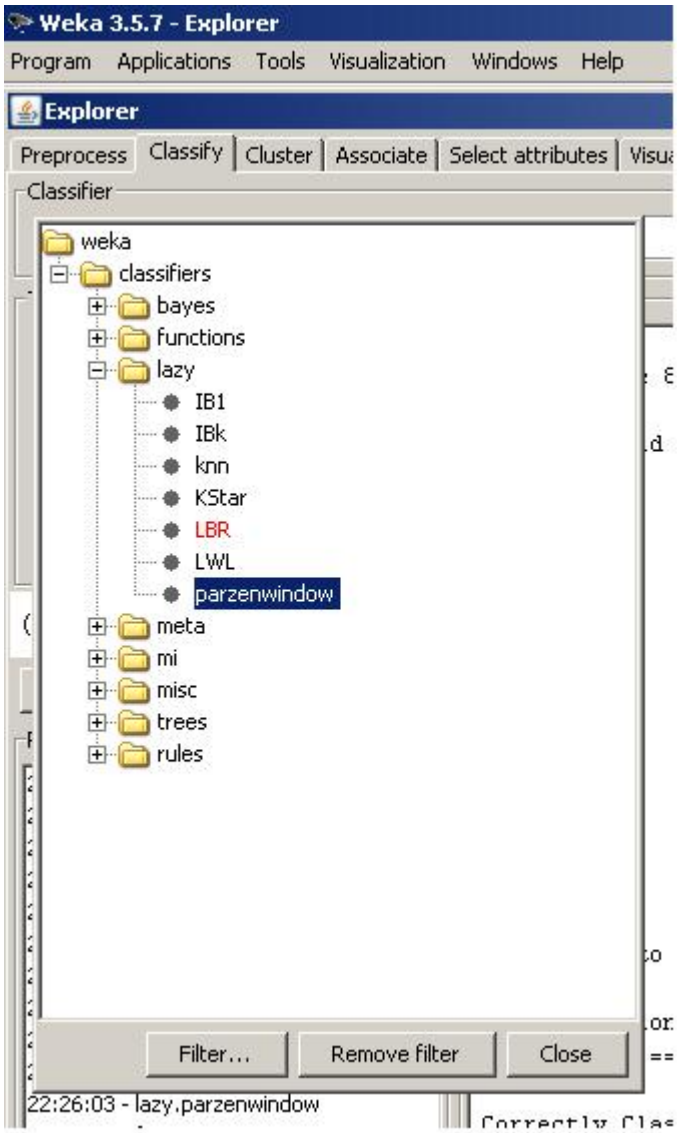



Figure 7: Parzen Window classifier implemented in Weka's GUI

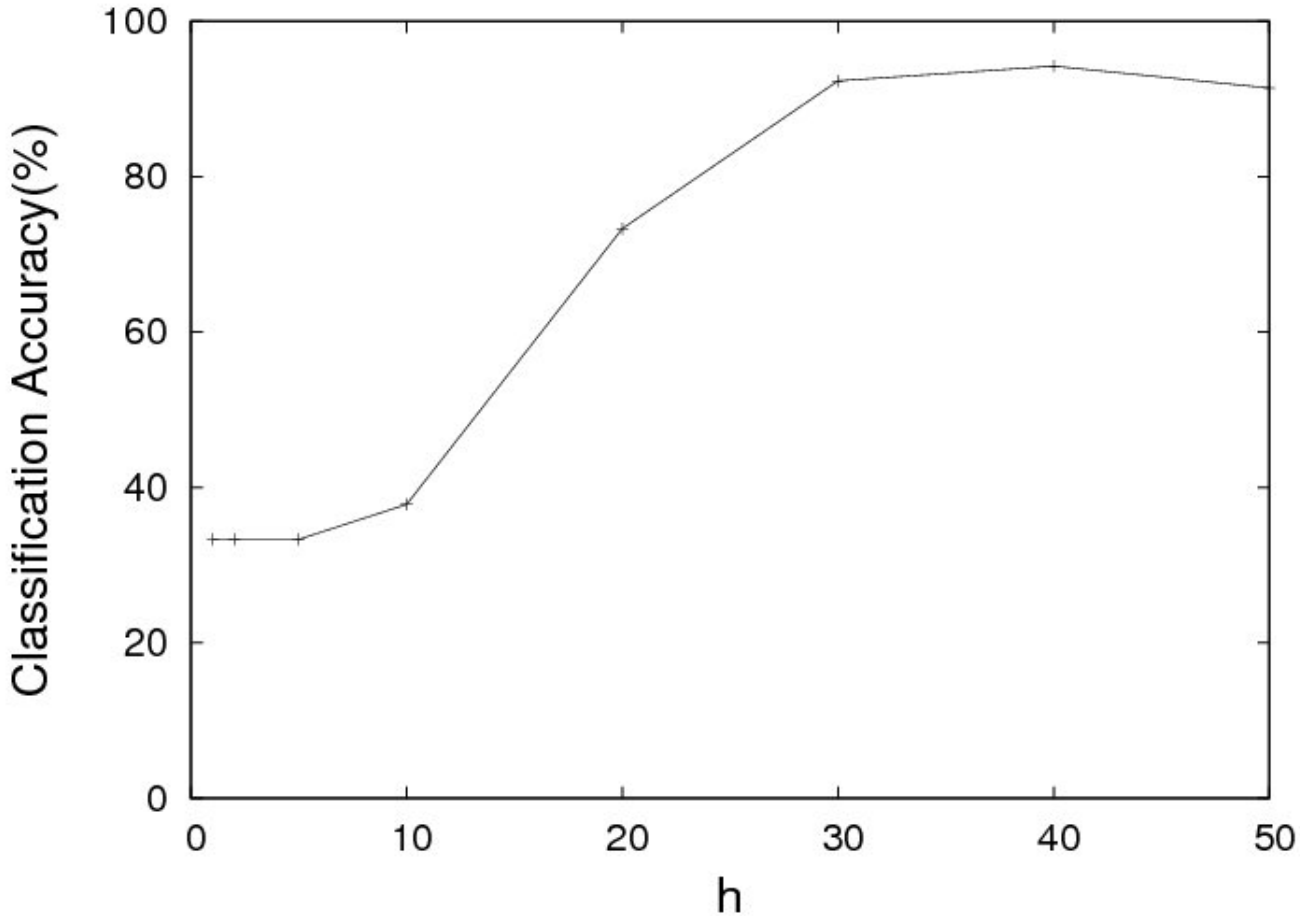
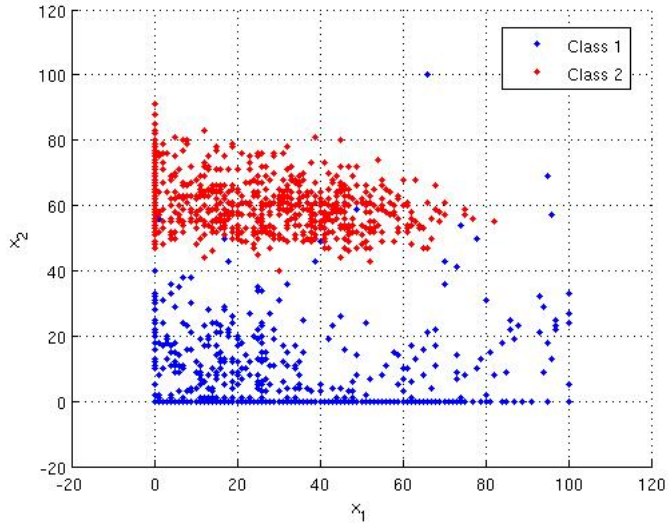
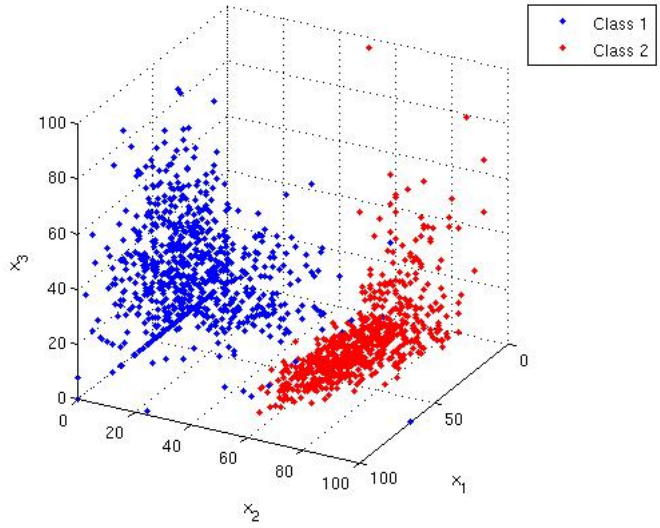



Figure 8: Classification accuracy for parzen window classifier

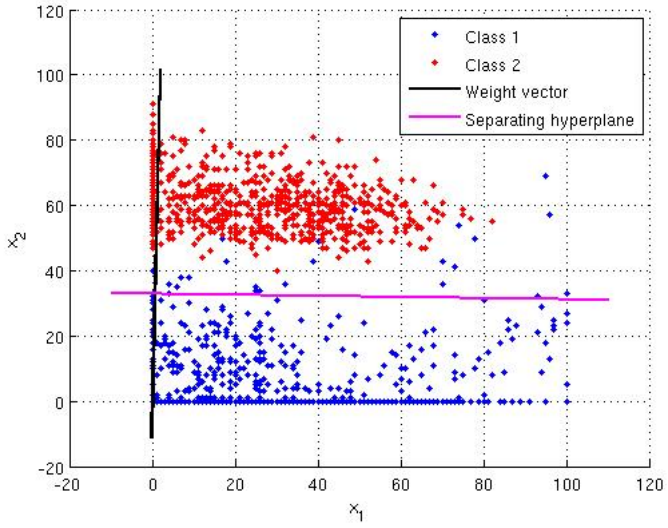



(a)

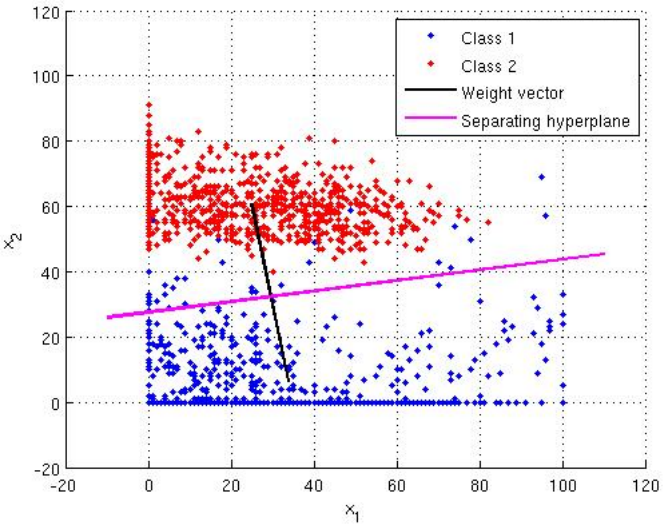



(b)

Figure 9: (a) Scatter plot for attributes 1 and 8 (b) 3D plot for attributes 1 and 8 and 15

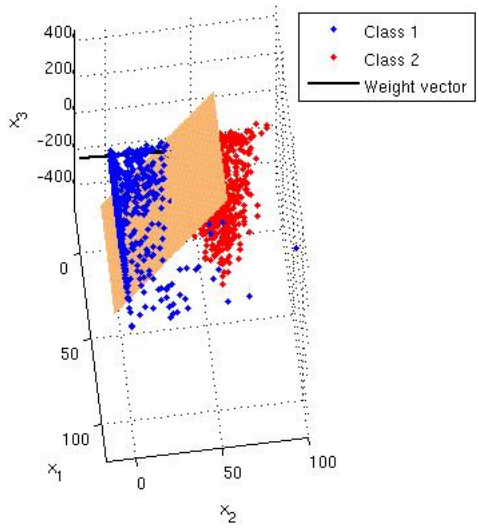



(a)

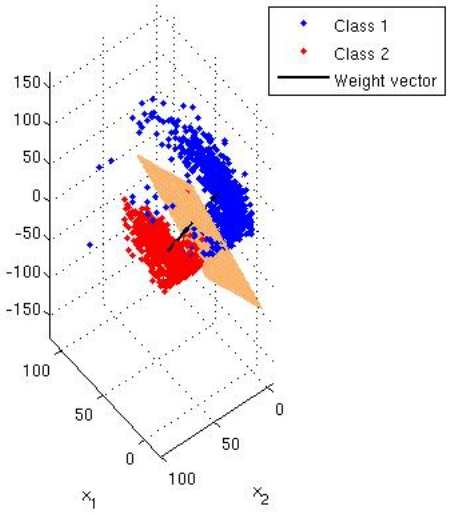



(b)

Figure 10: (a) FLD hyperplane in 2D (b) Difference of means hyperplane in 2D

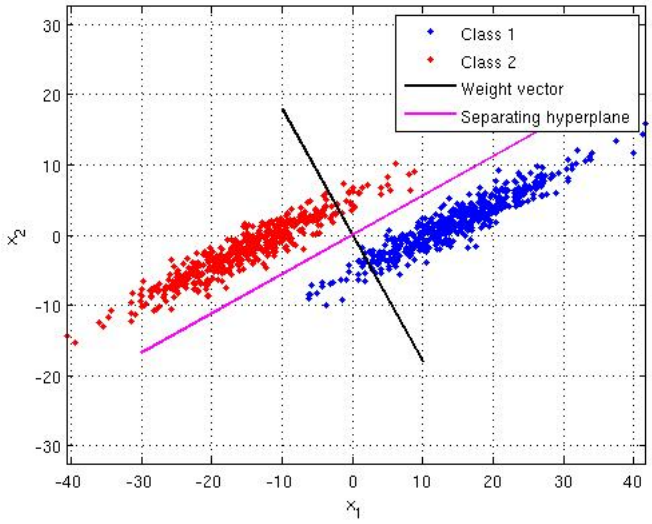



(a)

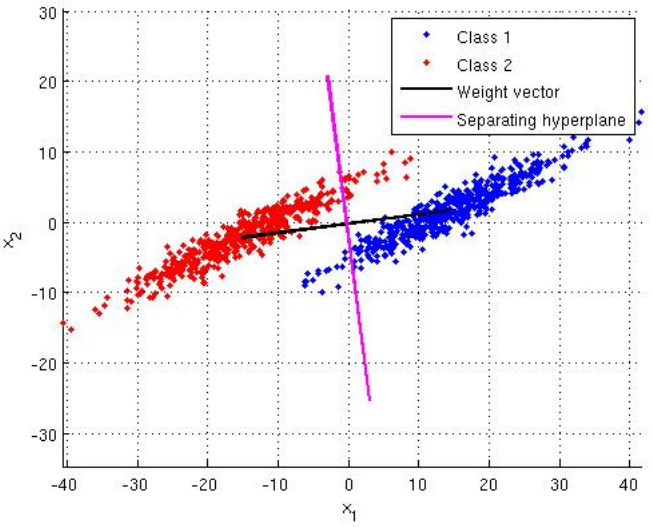



(b)

Figure 11: (a) FLD hyperplane in 3D (b) Difference of means hyperplane in 3D




(a)




(b)

Figure 12: (a) FLD hyperplane in 2D for a mixture of gaussians(b) Difference of means hyperplane in 2D for a mixture of gaussians