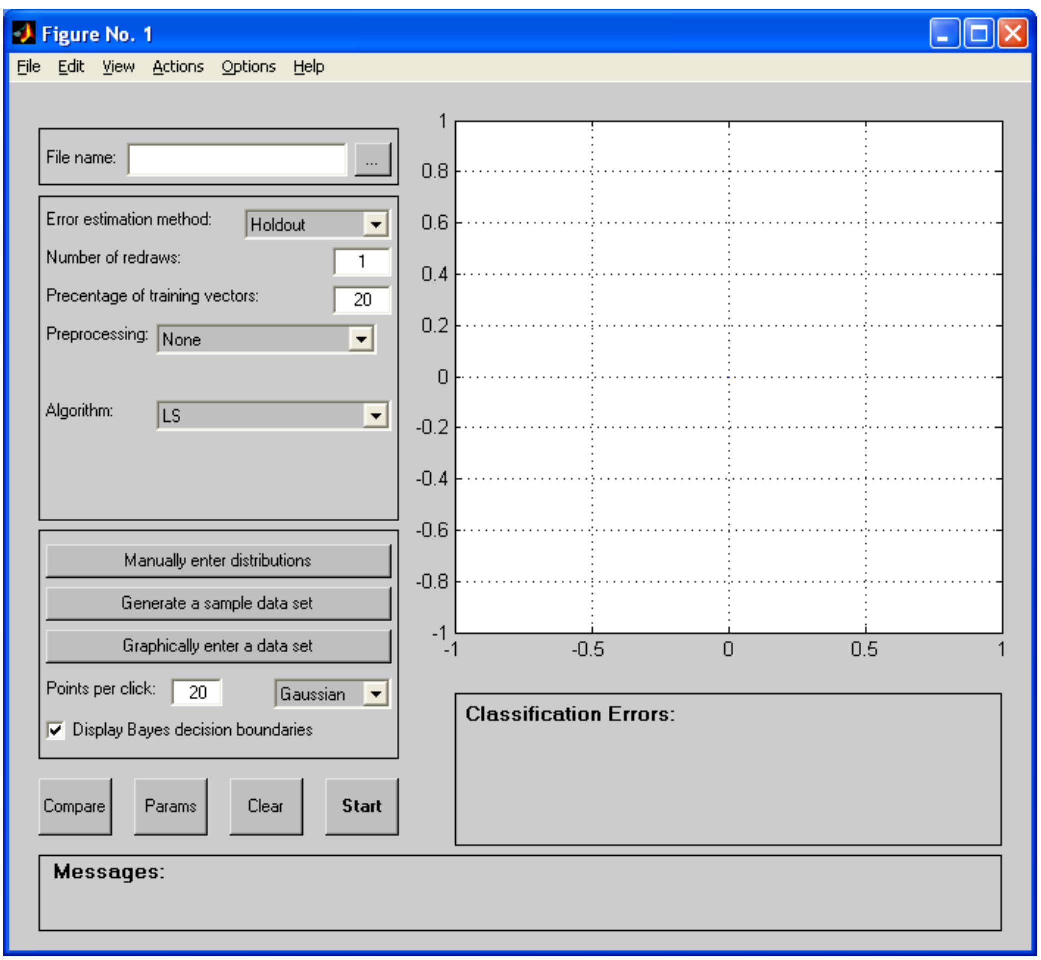
Introduction to the DHS ToolBox

Nawei Chen and Dorothea Blostein, 2004

These notes describe how to use the classification toolbox that accompanies *Pattern Classification*, second edition, by Duda, Hart, and Stork. The toolbox is written in MATLAB, but no knowledge of MATLAB is needed for the graphical user interface. The toolbox and manual can be purchased at www.wileyeurope.com/WileyCDA/WileyTitle/productCd-0471429775.html, but this is *not* required for CISC859. You are asked to read these notes for an illustration of how this toolbox can be used to compare Bayes classifier to Least Squares, 3-Nearest Neighbor, and Decision Tree. Similar comparisons can be carried out using other environments such as Weka or R.

Use these steps to start the user interface for the classification toolbox.

1. Download the zip file Classification_toolbox.zip
2. Unzip the zip file into a new directory
3. Add the path of the new directory to the MATLAB search path by typing in the MATLAB command window: addpath <directory>
4. Start the GUI by typing "classifier" in the MATLAB command window. This window appears:



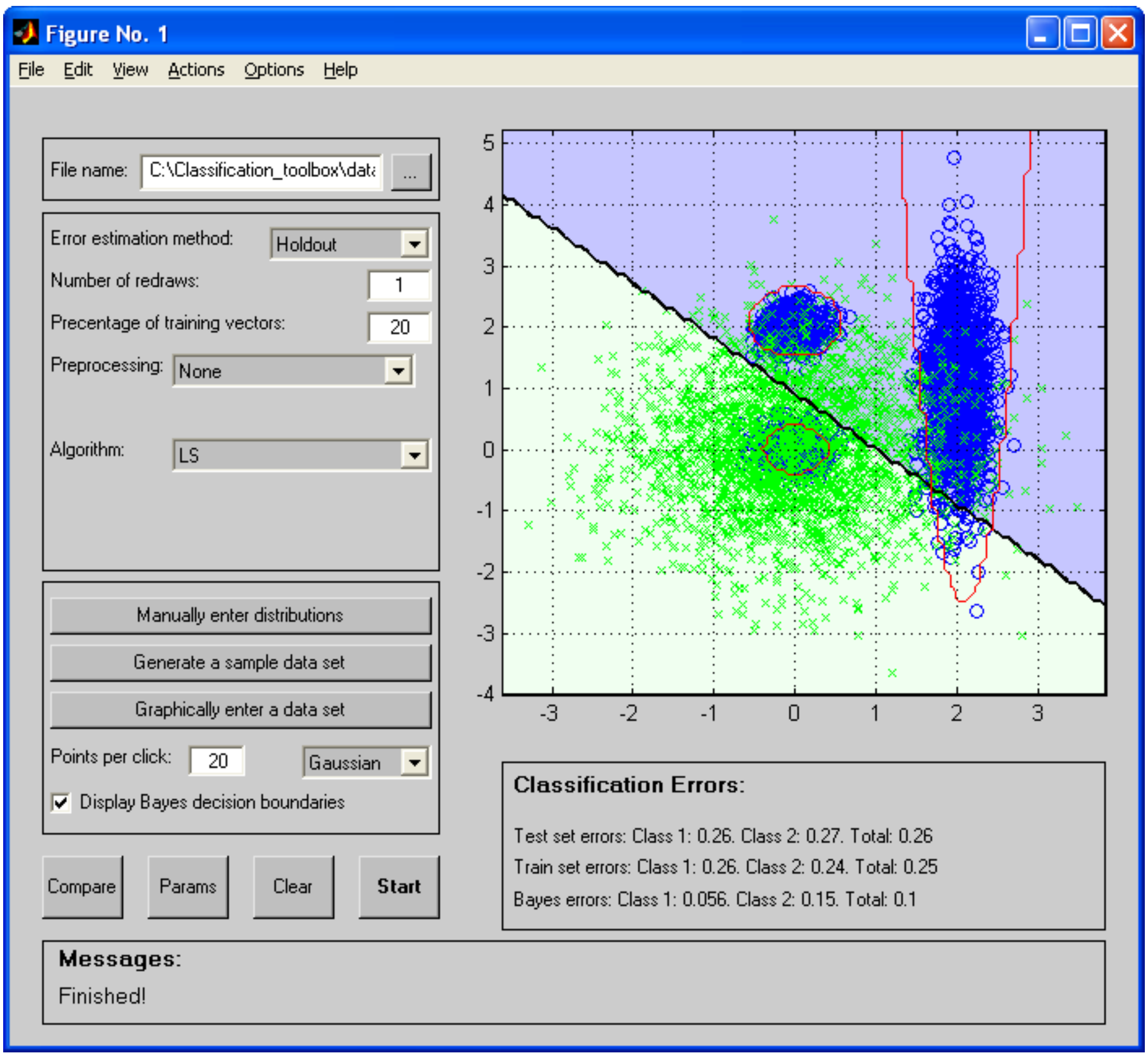
Examples of toolbox use are shown next.

Example 1. Access the “clouds” data set in the **datasets** sub-directory of the toolbox. This data consists of two classes (green and blue), with two features measured for each sample. Each blue circle indicates the location (in feature space) of a sample that is labeled as belonging to the blue class. Similarly, each green x indicates the location (in feature space) of a sample that belongs to the green class. As you can see, the green class consists of one “cloud”, and the blue class consists of two “clouds”, as well as a third small blue cloud that is located in the middle of the green cloud.

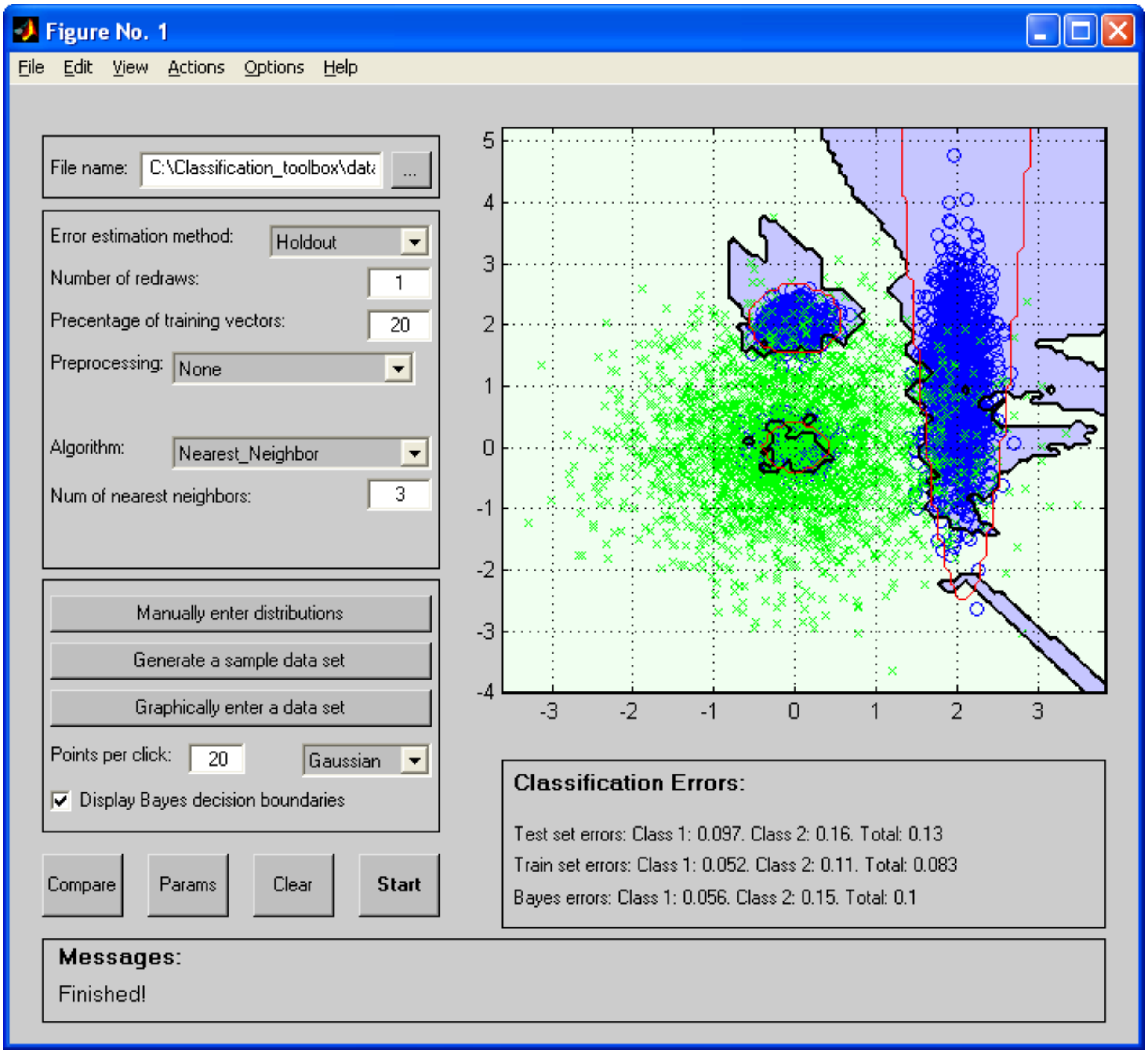
The black line shows one classification method (LS - Least Squares): classify everything under the black line as “green” and everything above the black line as “blue”.

The three red lines show another classification method (Bayes’ Classifier): lassify everything inside the three red-edged regions as “blue” and everything outside as green. (The green and blue data were generated using Gaussian probability densities. Since these densities are known, the Bayes’ classifier can be computed exactly; there is no need to estimate the probability densities from the training data.)

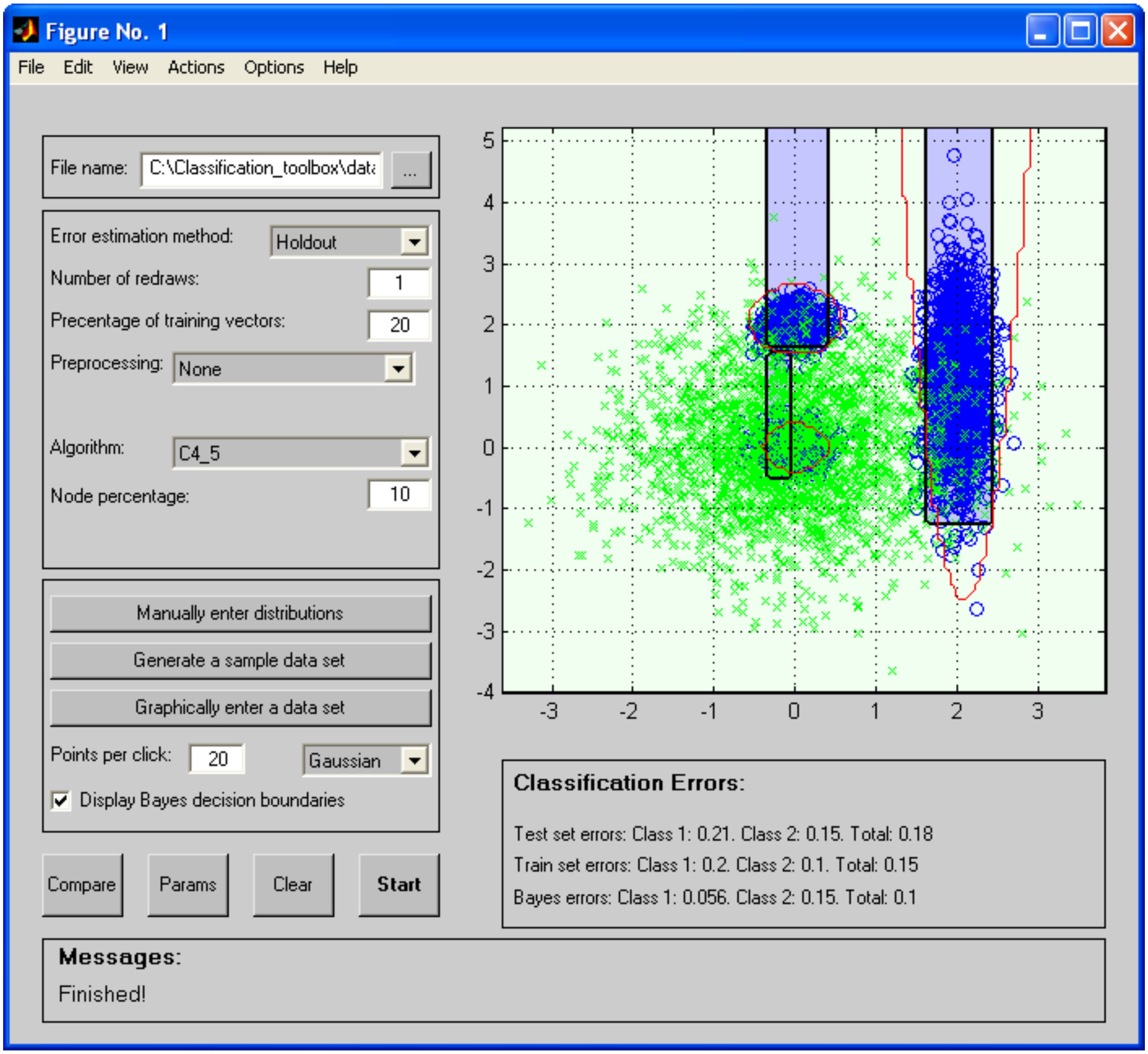
As selected in the boxes to the left, we use an LS classifier, with 20% of the data used to train the classifier, and 80% used to test the classifier. The LS classifier has errors on 26% of the test data, and the Bayes’ classifier has errors on10% of the test data.



Another example using the clouds data set. This time, a "3 nearest neighbour" classifier (black decision boundary) is compared to the Bayes' classifier (red decision boundary). As in the previous example, we use 20% of the data for training and 80% for testing. The 3 nearest neighbour classifier gest errors on 13% of the test set, compared to 10% error by the Bayes' classifier.



A third example using the clouds dataset. This time, a decision tree classifier is compared to the Bayes' classifier. The C4_5 algorithm is used to create the decision tree from the trraining data, using a node percentage of 10. The decision boundary for the decision tree is shown in black, and the decision boundary for the Bayes' classifier is shown in red. The decision tree gets 18% error on the test data, compared to 10% for the Bayes' classifier.

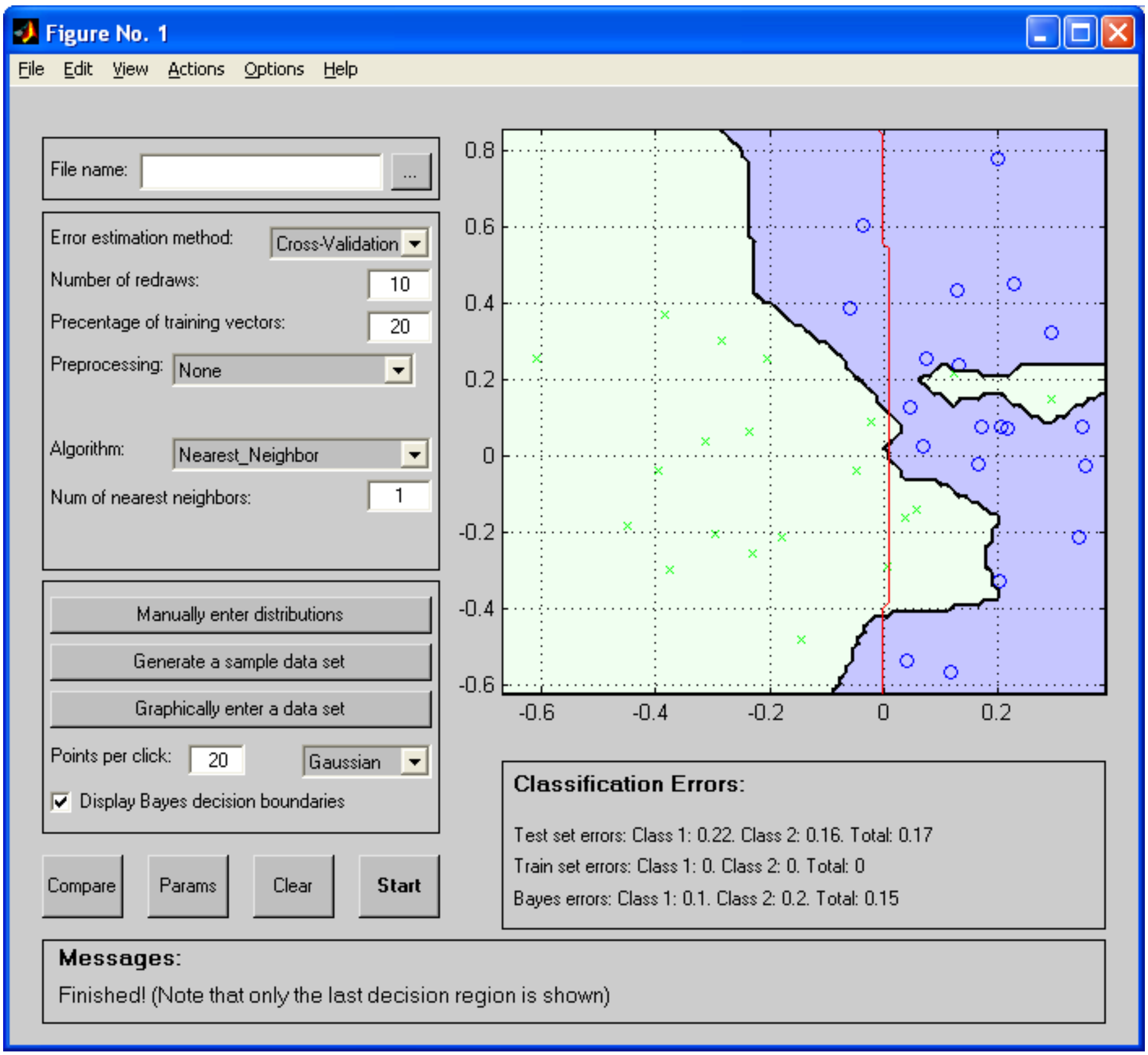


Creating your own data.

As shown in the screen shot below, click "Graphically enter a data set" to enter your own data. By default each mouse click adds 20 points in a Gaussian distribution, centered around the spot the user clicked.

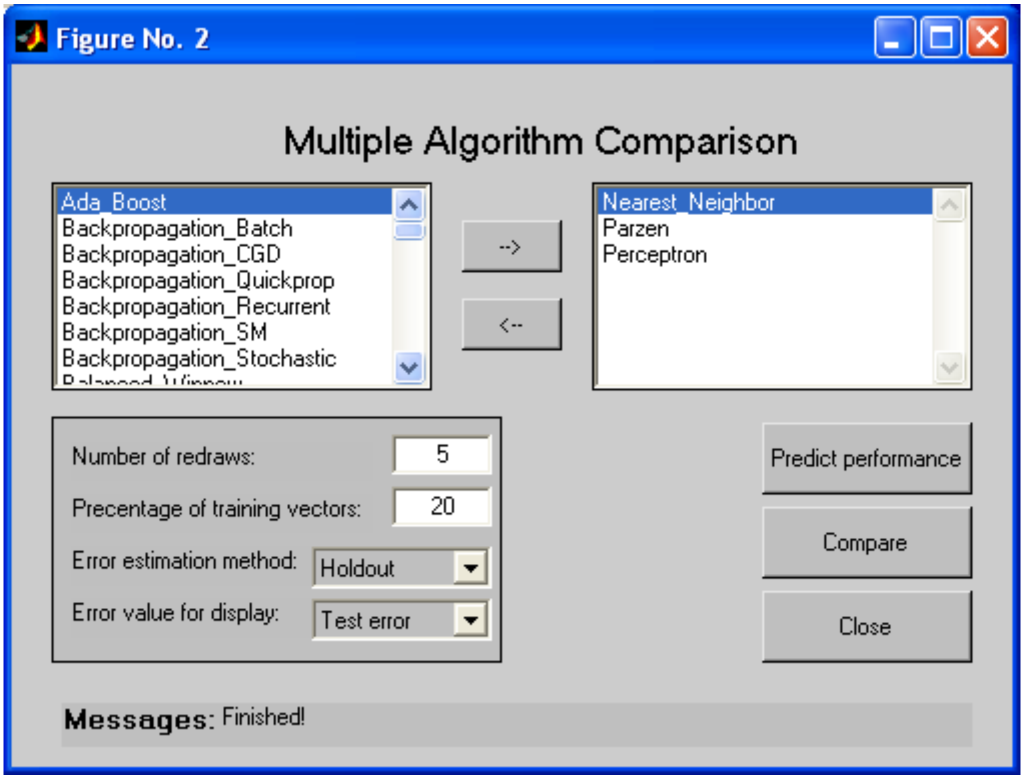
In this example the nearest neighbor classifier (black decision boundary) is compared to the Bayes' classifier (red decition boundary). Cross-validation error estimation is used, with 10 redraws. In this example, the nearest neighbor classifier has an error rate of 17%, and the Bayes' classifier has an error of 15%.

An alternate way of generating sample data is to click "manually enter distributions", select the distribution, and then click "Generate a sample data set".



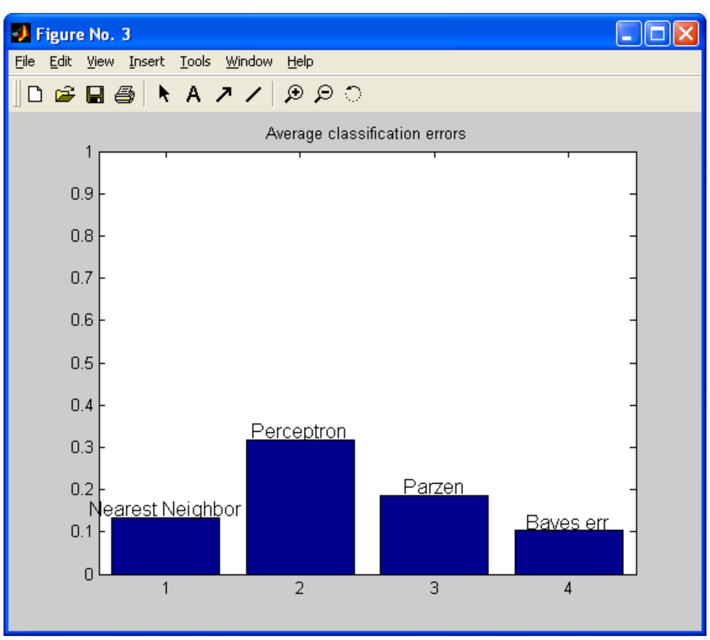
Compare performance of sevearl classifiers.

Start by loading a data set from the classifier window. Then launch the Multiple Algorithm Comparison window by clicking the "Compare" button in the classifier window.



The above window shows that we want to compare three algorithms: nearest neighbor, parzen, and perceptron.

This is the result: the performance of the three classifiers (nearest neighbor, parzen, perceptron) as well as Bayes' classifier, on the clouds data set.



Using the text-based interface

Here is an example of using the text-based interface. The graphical interface suffices for most purposes, so you probably will not have to use the text-based interface. However, you can look at the list of available algorithms below: algorithms for classification, for preprocessing, and for feature selection.

##load data set

```
>> load datasets/clouds
>> whos
  Name                              Size                        Bytes  Class

  distribution_parameters       1x2                          1464  struct array
  patterns                      2x5000                      80000  double array
  targets                       1x5000                      40000  double array

Grand total is 15076 elements using 121464 bytes
```

Data sets are stored as two variables in Matlab, patterns and targets.

```
## Choose test methods, training data and test data
%Make a draw according to the error method chosen
>> L = length(targets);
percent=20;
[test_indices, train_indices] = make_a_draw(floor(percent/100*L), L);
train_patterns = patterns(:, train_indices);
train_targets  = targets (:, train_indices);
test_patterns  = patterns(:, test_indices);
test_targets   = targets (:, test_indices);

## Choose a classifier. Find out parameters using help <classifier name>
>> help Nearest_Neighbor

  Classify using the Nearest neighbor algorithm
  Inputs:
       train_patterns      - Train patterns
       train_targets - Train targets
    test_patterns        - Test  patterns
       Knn                  - Number of nearest neighbors

  Outputs
       test_targets  - Predicted targets

## Build the classifier and classify the data
>> test_out=Nearest_Neighbor(train_patterns,train_targets,test_patterns,3);

## Estimate the error
>>error=mean(test_targets ~= test_out)

error =

    0.1313
```

Following are the algorithms implemented in the classification toolbox. The *show_algorithms* shows the name, parameters and their default values of all the algorithms implemented in the classification toolbox. It groups into three major categories, classification, clustering and preprocessing.

```
>> show_algorithms('classification',1)
ALGORITHM                  INPUTS                                   DEFAULT
-----------------------------------------------------------------------------
Ada_Boost                  Num iter, type, params:                  [100,'Stumps',[]]
Backpropagation_Batch      Nh, Theta, Convergence rate:             [5, 0.1, 0.1]
Backpropagation_CGD        Nh, Theta:                               [5, 0.1]
Backpropagation_Quickprop  Nh, Theta, Converge rate, mu:            [5, 0.1, 0.1, 2]
Backpropagation_Recurrent  Nh, Theta, Convergence rate:             [5, 0.1, 0.1]
```

```
Backpropagation_SM         Nh, Theta, Alpha, Converge rate:        [5, 0.1, .9, 0.1]
Backpropagation_Stochastic Nh, Theta, Convergence rate:            [5, 0.1, 0.1]
Balanced_Winnow            Num iter, Alpha, Convergence rate:      [1000, 2, 0.1]
Bayesian_Model_Comparison  Maximum number of Gaussians:            [5, 5]
C4_5                       Node percentage:                        1
Cascade_Correlation        Theta, Convergence rate:                [0.1, 0.1]
CART                       Impurity type, Node percentage:         ['Entropy', 1]
Components_with_DF         Number of components:                   10
Components_without_DF      Components:                     [('LS'),('ML'),('Parzen', 1)]
Deterministic_Boltzmann    Ni, Nh, eta, Type, Param:       [10, 10, 0.99, 'LS', []]
Discrete_Bayes             None
EM                         nGaussians [clss0,clss1]:               [1,1]
Genetic_Algorithm         Type,Params,TargetErr,Nchrome,Pco,Pmut:['LS',[],0.1,10,0.5,0.1]
Genetic_Programming        Init fun len, Ngen, Nsol:               [10, 100, 20]
Gibbs                      Division resolution:                    10
Ho_Kashyap                 Decision, Max_iter, Theta, Eta:  ['Basic', 1000, 0.1, 0.01]
ID3                        Number of bins, Node percentage:        [5, 1]
Interactive_Learning       Number of points, Relative weight:      [10, .05]
Local_Polynomial           Num of test points:                     10
LocBoost                   Nb,Nem,Nopt,LwrBnd,Opt,Ltype,Lparam:    [10, 10, 10, 'LS', []]
LMS                        Max_iter, Theta, Converge rate:         [1000, 0.1, 0.01]
LS                         None
Marginalization            #missing feature, #Bins:                [1, 10]
Minimum_Cost               Cost matrix:                            [0, 1; 1, 0]
ML                         None
ML_diag                    None
ML_II                      Maximum number of Gaussians:            [5, 5]
Multivariate_Splines       Spline degree, Number of knots:         [2, 10]
NDDF                       None
Nearest_Neighbor           Num of nearest neighbors:               3
Optimal_Brain_Surgeon      Nh, Convergence criterion:              [10, 0.1]
Parzen                     Normalizing factor for h:               1
Perceptron                 Num of iterations:                      500
Perceptron_Batch           Max iter, Theta, Convergence rate:      [1000, 0.01, 0.01]
Perceptron_BVI             Max iter, Convergence rate:             [1000, 0.01]
Perceptron_FM              Num of iterations, Slack:               [500, 1]
Perceptron_VIM             Max iter, Margin, Converge rate:        [1000, 0.1, 0.01]
Perceptron_Voted           #Prcptrn, Mthd, Mthd_P:                 [7,'Linear',0.5]
PNN                        Gaussian width                          1
Pocket                     Num of iterations:                      500
Projection_Pursuit         Number of components:                   4
RBF_Network                Num of hidden units:                    6
RCE                        Maximum radius:                         1
RDA                        Lambda:                                 0.4
Relaxation_BM              Max iter, Margin, Converge rate:        [1000, 0.1, 0.1]
Relaxation_SSM             Max iter, Margin, Converge rate:        [1000, 0.1, 0.1]
Store_Grabbag              Num of nearest neighbors:               3
Stumps                     None
SVM                    Kernel, Ker param, Solver, Slack:['RBF', 0.05, 'Perceptron', inf]
None                       None

>> show_algorithms('preprocessing',1)
ALGORITHM                 INPUTS                          DEFAULT
-----------------------------------------------------------------
ADDC                      Number of partitions:           4
AGHC                      Number of partitions, Distance: [4, 'min']
BIMSEC                    Num of partitions, Nattempts:   [4, 1]
Competitive_learning      Number of partitions, eta:      [4, .01]
Deterministic_Annealing   Num partitions, Cooling rate:   [4, .95]
Deterministic_SA          Num partitions, Cooling rate:   [4, .95]
DSLVQ                     Number of partitions:           4
FishersLinearDiscriminant None
Fuzzy_k_means             Number of partitions:           4
k_means                   Number of partitions:           4
Kohonen_SOFM              Num units, Window width:        [10, 5]
Leader_Follower           Min Distance, Rate:             [0.1, 0.1]
LVQ1                      Number of partitions:           4
LVQ3                      Number of partitions:           4
Min_Spanning_Tree         Method, Factor:                 ['NN', 2]
NearestNeighborEditing    None
```

```
PCA                       New data dimension:            2
Scaling_transform         None
SOHC                      Num of partitions:             4
Stochastic_SA             Num partitions, Cooling rate:  [4, .95]
Whitening_transform       None
None                      None

>> show_algorithms('feature_selection',1)
ALGORITHM                    INPUTS                                          DEFAULT
------------------------------------------------------------------------------------
Exhaustive_Feature_Selection Out dim, classifier, classifier params       [2,'LS',[]]
Genetic_Culling     %groups, Out dim, classifier, classifier params [0.1,2,'LS',[]]
HDR                          Out dimension                                  2
ICA                          Out dimension, Convergence rate:             [2, 1e-1]
Information_based_selection  Out dimension                                  2
MDS                          Method, Out dimension, Convergence rate    ['ee', 2, 0.1]
NLCA                         Out dimension, Number of hidden units:         [2, 5]
PCA                          Out dimension                                  2
```