Classification Toolbox

For use with MATLAB®

David G. Stork

Elad Yom-Tov

User's Guide

Introduction

The classification toolbox is a list of functions for supervised and unsupervised classification algorithms. These algorithms help design categorization methods for experimental as well as synthetic data, using either the basic Matlab interface or a convenient graphic user interfaces. The graphic user interface is designed for two-dimensional, two-class problems, but the algorithms can be used on higher dimensional data, and most can be used to classify multi-class data. In addition, higher dimentional data can be reduced to two dimensions using one of several feature selection algorithms.

This user guide explains how to use the toolbox, for the novice as well as the experienced user. Since most of the functions can be operated from within the graphic user interface, only basic knowledge of Matlab is required. No theoretic background about classification is given in this manual, and the user is referred to [1] for such a background.

The toolbox is intended to accompany a book to be published during 2003, and users are encouraged to send comments and suggestions to elad@ieee.org.

History and acknowledgements

This toolbox started as a course assignment in Dr. Ron Meir's graduate course, Pattern Recognition. The foundation for the toolbox, as well as most of the basic algorithms, were coded by Elad Yom-Tov and Hilit Serby. This resulted in a toolbox with (about) 15 algorithms.

A year later, Igor Makienko and Victor Yosef coded the Voted Perceptron algorithm. Some time later Elad Yom-Tov continued coding, making the toolbox in it's present form with over 100 different algorithms.

The toolbox is now a pre-publication version of a work tentatively titled Computer Manual in MATLAB to Accompany Pattern Classification by David G. Stork and Elad Yom-Tov to be published during 2003.

Copyrights etc.

The toolbox is available from this address:
http://tiger.technion.ac.il/~eladyt/Classification_toolbox.html

Correspondence: Elad Yom-Tov
Faculty of Electrical Engineering
Technion – Israel Institute of Technology
Haifa 32000
Israel
elad@ieee.org

This is a pre-publication version of a work tentatively titled Computer Manual in MATLAB to Accompany Pattern Classification by David G. Stork and Elad Yom-Tov to be privately distributed to students enrolled in academic courses during 2002. NOT FOR GENERAL DISTRIBUTION.

By using the Classification toolbox you agree to the following licensing terms:

I will, of course, thank you for any bugs you report to me (elad@ieee.org), but I will not be held responsible for any problems or difficulties you encounter because of them.

Table of contents

Changes in version 2.0

Version 2.0, first uploaded in the middle of March 2003, contains several important improvements:

1. The algorithms can process data in high dimensions. Note that this change caused a difference in the input and output format of the algorithms.
2. Most can work with multiclass data. If there are more than two classes the returned test targets will not be collapsed to integer values.
3. Data can be easily classified using the Matlab text-based interface.
4. Data format has changed. Dataset should now contain a matrix called "patterns" and a vector called "targets". The structure of parametric distributions also has a new format.
5. During clustering the Voronoi regions can be shown in addition to the cluster centers.
6. Parameteric structures can contain both Gaussian and Uniform distributions.
7. All sample datasets (Including the sample datasets in [1]) are now placed in a separate directory.
8. There are now upwards of 50 classification algorithms, 20 preprocessing algorithms, and 7 feature selection algorithms.

Installation

The classification toolbox was tested under Matlab version 5.2 and higher. You will need a basic installation of Matlab of these versions on your computer in order to run the toolbox.

After downloading the classification toolbox zip file, please follow these steps in order to install it:

1. Create a new directory (preferably under the Matlab directory).
2. Unzip the contents of the toolbox zip file into this directory.
3. Include the path of the new directory in the Matlab search path by typing (in the Matlab command window): `addpath` <directory>

An important note for Matlab 5.2 users:

For an unknown reason, there is a bug with the main GUI file using this version of Matlab. Please do the following in order to avoid it:

1. Open the classifier.m file.
2. Remove line 23 which reads: `'FileName', 'D:\Users\elad\HW\Classification_toolbox\classifier.m', ...`
3. Remove the text "`'ToolBar','none'`" from line 32.
4. Save classifier.m

Single algorithms: The graphic user interface

The easiest way to operate the classification toolbox is using the graphic user interface (GUI). Start the GUI by typing `classifier` in the Matlab command window. This will bring up a screen similar to the one shown in figure 1.

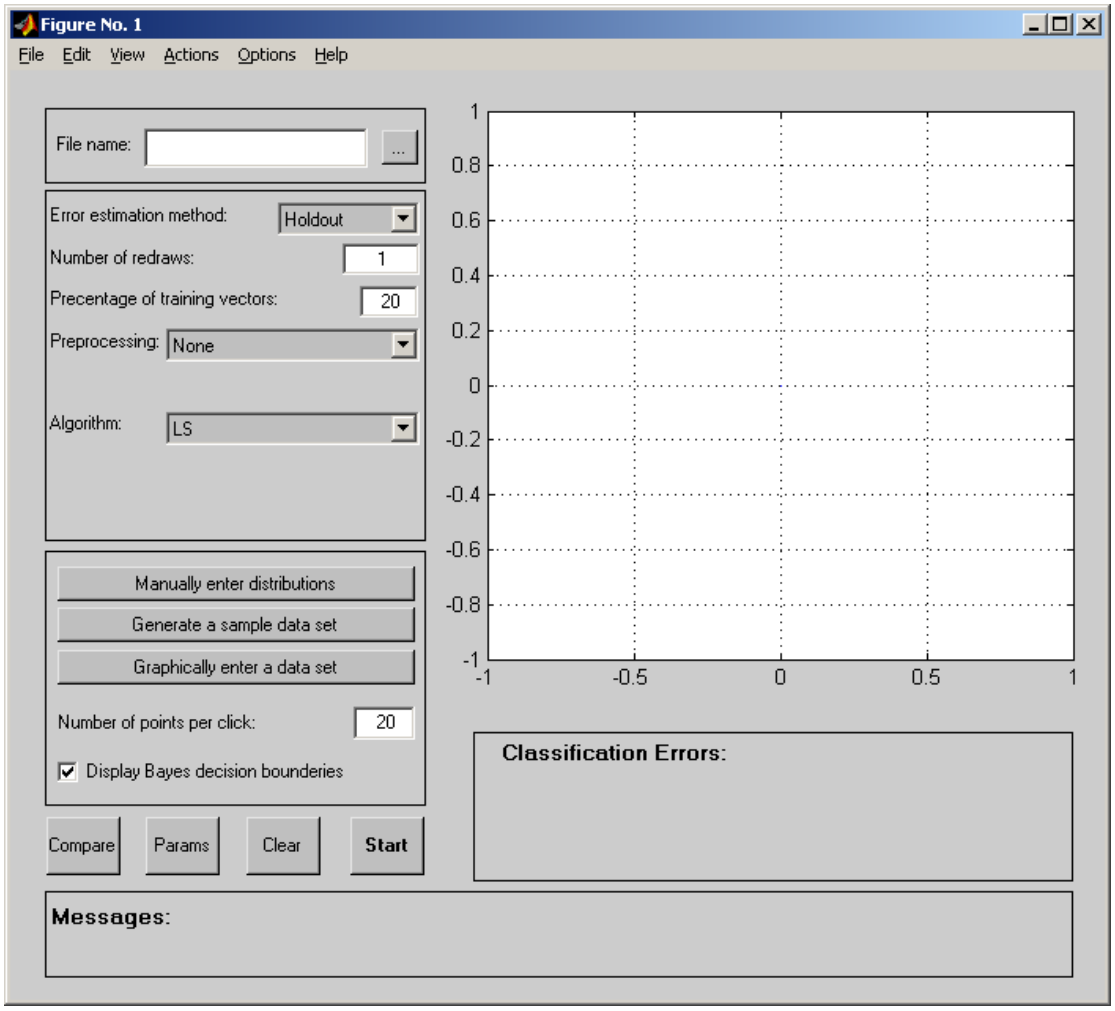

Figure 1: The classification GUI

A simple example

Before we explain the various options of the toolbox, an example will help explain the basic features of the toolbox.

1. Change the Matlab current directory to that of the classification toolbox.
2. Type `clouds` in the File name box (Top left corner of the GUI). This will load a data set called "clouds" into the workspace, and plot it on the graph area.
3. Press the `start` button (lower left in the GUI), and after a few seconds a black line showing a decision border will be drawn. Additionally, the test and train set error will be computed and shown under the graph.

What you have done in these two simple steps is load a distribution file into the workspace, and classify it using a linear Least Squares classifier, trained using 20% of the data, and tested on the remaining 80% (Holdout method). These are the default parameters for classification. In the following sections we will show how to change these parameters as needed.

Sections of the graphic user interface

The GUI called by typing `classifier` includes a command menu as well as the following sections:

1. Input file area.
2. Classification and preprocessing parameters area.
3. Parameteric distributions area.
4. Command buttons.
5. System messages box.
6. Error percentages box.
7. Graphic area.

Menu

Using the file menu, the following commands may be issued:

1. Open: This menu option assists in loading files with a distribution data (i.e., features and targets) and/or parameters of a distribution file (i.e., the parameters of Gaussian distributions). The structure of the file is explained in the files section below.
2. Save: This menu option assists in saving a distribution file or the parameters of a distribution.
3. Clear: This menu option opens a submenu which assists in clearing the bounds of a recently constructed classifier or clearing the entire workspace, including features, targets, classifiers etc.
4. Permute features: Mixes the features, so that cross-validation of sets that are ordered according to the targets will work.
5. Exit: Closes the GUI.

Using the edit menu, the following commands may be issued:

1. Print: Sends the current decision boundaries and distributions to a printer.
2. Copy: Copies the current decision boundaries and distributions.

Using the view menu, the following commands may be issued:

1. Zoom: Toggles zoom on the figure.
2. Grid: Toggles the grid on the figure.

Using the actions menu, the following commands may be issued:

1. Start classification: Start the classification process, using the parameters shown on screen.
2. Show distribution parameters: Show the means and standard deviations of the features, as well as the Chernoff and Bhattacharyya bound assuming that the data is Gaussian. These are shown in a separate GUI screen which will automatically open.
3. Find best parameters for the chosen classifier: This will open a screen for help in selecting the best parameters for the chosen classification algorithm, as described in [9].
4. Compare algorithms: Start the comparison GUI (See below).
5. Manually enter distributions: Pressing this button will ask for the number of parametric structures (Gaussians or Uniform distributions) for each class, and the relative probability of each class. Then, a screen in which the parameters of these structures may be entered is loaded.
6. Generate dataset: Once the parameters of the distribution are loaded, pressing this button will generate a sample data set of the necessary size.
7. Graphically enter a dataset: Allows entering of a data set via clicking of the screen or dragging of boxes on it. The box below shows how many points will be generated by each click of the mouse.

Note that except for option 3, all the options are available through buttons on the GUI.

Using the options menu, the following commands may be issued:

1. Perform preprocessing separately for each class: Checking this option will carry out the preprocessing for each class of the data separately, if possible (It is impossible in preprocessing such as LVQ3, PCA etc.).
2. Show centers of partition during training: Checking this option will show the location of partition centers for iterative preprocessing algorithms (for example: Deterministic annealing).

3. Show training points: Checking this option will color training points in different colors from the test points.
4. Shade decision region: This will cause the GUI to both show the decision boundry and color the regions in different colors.

Using the help menu, the following commands may be issued:

1. Help of the preprocessing algorithm: Displays the help of the currently chosen preprocessing algorithm.
2. Help of the classification algorithm: Displays the help of the currently chosen classification algorithm.
3. About: Try and see…

Input file area

This area of the GUI enables you to load a distribution and/or a the parameters of a distribution. Entering the name of the file in the box (and typing enter) will load the file, if it is in a directry included in the Matlab search path (or in the local directory). The file structure is explained in the file structure chapter below.

Pressing the three dots on the right of the name box will bring up a file dialog box for locating a distribution file.

Classification and preprocessing parameters area

This area is the principal part of the GUI. In this section, the classification parameters are entered. The possible parameters are divided into three parts:

Error estimation method

This section describes the selection of the test and training points. The following options are available:

1. Holdout (the default option): This will use the percentage of points shown in the box below for training, retaining all other points for testing
2. Cross-validation: This will divide the data into parts, the number of which is determined in the "number of redraws box". Then, a the classifier will be trained using one part and tested on the other parts of the data, and repeated for all the parts.
3. Resubstitution: This will train and test the data on all the points in the data set.

If holdout or resubstitution is used, and more than one redraw specified, the classifier will be trained a number of times, and the reported error rate will be the average error.

Preprocessing

This part lets the user enter a preprocessing method (if such a method is necessary; otherwise, enter 'None' in this box). If any additional parameters are needed (such as the number of partitions) an input box specifying the required parameter will open below the preprocessing box. For example, entering k_means preprocessing method will open a box calling for the number of clusters to be entered. An important point to note is that the GUI requires at least two features of each class for the classification process. Therefore, if less than two features are generated by the preprocessing algorithm, and classification is required, an error will be generated.

Using the appropriate option in the Options menu, it is possible to perform the preprocessing for each class separately.

The list of currently supported algorithms follows. These algorithms are described in [1], unless otherwise noted:

1. Agglomerative clustering (ADDC). See [2] for details.
2. Agglomerative hierarchical clustering algorithm (AGHC).
3. Basic iterative MSE clustering (BIMSEC).
4. Basic leader-follower clustering.
5. Competitive learning.
6. Deterministic annealing (DA) [3]: Two implementations exist in the toolbox, one (Stochastic_DA) is similar to that proposed in [1].
7. Distinction sensitive learning vector quantization (DSLVQ) [4].
8. K-means.
9. Kohonen self-organizing feature maps (Kohonen_SOFM).
10. Fuzzy k-means.
11. Fishers' linear discriminant.
12. LVQ1
13. LVQ3
14. Minimum spanning tree.
15. Principal component analysis (PCA), also known as Karhunen-Louve transform (KLA).
16. Stepwise optimal hierarchical clustering (SOHC).
17. Stochstic simulated annealing (SA).

Additional algorithms may be added by following the directions in the appendix.

If the "Perform preprocessing separately for each class" option in the options menu is checked, preprocessing will be performed separately for each class, if such preprocessing is possible.

If the "Show centers of partition during training" option is checked, the centers of the partition will be shown during iterations of iterative algorithms. This is useful when better understanding of how an algorithm works is needed.

When algorithms which generate representative features are used, the resulting features are shown as red crosses and circles (depending on their label) in the plot area. The label is determined by assigning each of the features to the nearest cluster

center, and a majority vote on all the features assigned to each cluster. Such algorithms will also generate a Voronoi diagram of the partitioning, which is shown on a separate graph.

Classification algorithms

This part lets the user enter a classification algorithm. If only preprocessing is needed (to check various unsupervised learning methods), enter 'None' in this box. In this case, error percentages are irrelevant.

The classification toolbox is designed for two-class two-dimensional problems. However, most algorithms can be operated (not using the GUI) with higher dimensional problems. Some algorithms can be used on multi-class problems as well. Higher dimentional data can be reduced to two dimensions using one of several feature selection algorithms. See the "Feature selection" chapter for more on this feature.

The list of currently supported algorithms follows. These algorithms are described in [1], unless otherwise noted:

1. Ada Boost.
2. Backpropagation neural network with batch training (Backpropagation_Batch).
3. Backpropagation neural network with conjugate gradient descent (Backpropagation_CGD).
4. Backpropagation neural network with Quickprop learning (Backpropagation_Quickprop).
5. Backpropagation neural network with stochastic training (Backpropagation_Stochastic).
6. Backpropagation neural network with stochastic training and momentum (Backpropagation_SM).
7. Balanced Winnow.

8. Batch perceptron (Perceptron_Batch).
9. Batch relaxation with margin (Relaxation_BM).
10. Batch variable-increment perceptron (Perceptron_BVI).
11. Bayesian Model Comparison
12. C4.5.
13. Cascade-Correlation type neural network (Cascade_correlation).
14. Classification and regression trees (CART).
15. Component classifiers with descriminant functions (Components_with_DF).
16. Component classifiers without descriminant functions (Components_without_DF).
17. Deterministic Boltzmann classifier.
18. Discrete Bayes classifier.
19. Expectation-maximization (EM).
20. Genetic algorithm (Basic GA).
21. Genetic programming.
22. Gibbs algorithm.
23. ID3.
24. Interactive Learning (Learning with queries).
25. Linear Least squares (LS).
26. Least-mean squares (LMS).
27. Local polynomial fitting.
28. Local boosting [6].
29. LVQ1.
30. LVQ3.
31. Marginalization
32. Maximum likelihood (ML).
33. Maximum likelihood using diagonal covariance matrices (ML_diag).
34. Maximum likelihood model comparison (ML_II).
35. Minimum cost classification.
36. Multivariate adaptive regression splines.
37. Nearest Neighbors.
38. Nearest Neighbor Editing.
39. Normal density discriminant function (NDDF).

40. Optimal brain surgeon.
41. Parzen windows.
42. Perceptron.
43. Pocket.
44. Probabilistic neural network.
45. Projection pursuit regression.
46. Quickprop (Backpropagation_Quickprop).
47. Recurrent neural network (Backpropagation_Recurrent).
48. Radial basis function network (RBF_Network).
49. Reduced coulomb energy algorithm (RCE).
50. Regularized descriminant analysis (RDA): Also known as Friedman shrinkage [5].
51. Single-step relaxation with margin (Relaxation_SSM).
52. Store-Grabbag.
53. Stumps.
54. Support-vector machines (SVM).
55. Variable-increment perceptron with margin (Perceptron_VIM).
56. Voted perceptron.

Additional algorithms may be added by following the directions in the appendix.

If any additional parameters are needed an input box specifying the requiered parameter will open below the preprocessing box. For example, entering the Parzen windows classification method will open a box calling for the normalizing factor h to be entered. Since support-vector machine classifiers are controled by many parameters, a separate parameter screen will open for this classifier.

Graphic area

This area of the GUI displays the features on the screen, togeather with the decision region and the Bayes decision region (if the parameters of the distribution are entered).

The features of the first class are shown as blue circles, and the features of the second class as green crosses. The decision region is shown in black, and the Bayes decision region in red. If a preprocessing algorithm which generates representative features is used, the resulting features are shown as red crosses and circles (depending on their label).

Parameteric distributions area

This area of the screen is useful for entering mixtures of Gaussian distributions. Three buttons may be used for these functions:

1. Manually enter distributions: Pressing this button will ask for the number of parametric structures (Gaussian or Uniform) for each class, and the relative probability of each class. Then, a screen in which the parameters of these structures may be entered is loaded.
2. Generate sample data set: Once the parameters of the distribution are loaded, pressing this button will generate a sample data set of the necessary size.
3. Graphically enter a data set: Allows entering of a data set via clicking of the screen or dragging of boxes on it. The box below shows how many points will be generated by each click of the mouse.

Checking the "Display Bayes decision bounderies" will show the optimal decision region on the graph, once a classifier is trained. This boundery is shown in red. The boundery is displayed only if the distributions are obtained parameterically.

Command buttons

Four command buttons help operate the GUI. These buttons are:

1. Start: Start the classification process, using the parameters shown on screen.
2. Clear: Remove decision bounderies and error rates.

3. Params: Show the means and standard deviations of the features, as well as the Chernoff and Bhattacharyya bound assuming that the data is Gaussian. These are shown in a separate GUI screen which will automatically open.
4. Compare: Starts the multiple algorithm comparison GUI, described below.

System messages box

This box shows system messages, such as the completion of the classification process, missing variables etc. Some critical errors will appear on the Matlab work area togeather with a warning tone, instead of in the message box.

Error percentages box

This box displays the test and train set errors, togeather with the Bayes error (if the parameters of the distribution are entered) after a classifier is trained. The errors are claculated for each class separately.

Feature selection

As descriped in the previous sections, the GUI is meant for use with two dimensional data. If a higher dimensional dataset is loaded, feature selection will have to be performed before this data can be used with the toolbox. Thus, if an attempt is made to open a file containing a high-dimention data, a new GUI will open and request the user to select a feature selection method.

Currently available feature selection methods (described in [1] unless otherwise noted) are:

1. Culling genetic algorithm [7].
2. Hierarchical dimensionality reduction (HDR).
3. Independent component analysis (ICA).
4. Multidimensional scaling (MDS).
5. Non-linear principle component analysis (NLPCA).
6. Principle component analysis (PCA).
7. The Koller algorithm [8].

After selection of a feature selection method is made, this algorithm will be run to select two representative features, and these two features will be used in the main classifier GUI.

A comprehensive example

Suppose we want to test the maximum likelihood algorithm on features reduced using the k-means algorithms on the "clouds" data set. We will use 5-fold cross-validation, 20 reduced features, and display the Bayes decision region. To do all this, the following steps are needed:

1. Load the clouds data set by typing `clouds` in the file name box.
2. Change error estimation method to `cross-validation`, and choose 5 redraws in the box below.
3. Choose `k_means` preprocessing and change the number of partitions to 20 (the default is 4).
4. Change the classification algorithm to `ML`.
5. Press start.

After a few seconds, a figure similar to that shown in figure 2 will appear. On this figure, the features are shown (in blue and green). The reduced features are shown as red crosses. The last decision region (of the 5 redraws) is shown as a black line, and the Bayes decision region as a red line. Error rate values of 25%, compared to 13% for the Bayes error rate are reported.

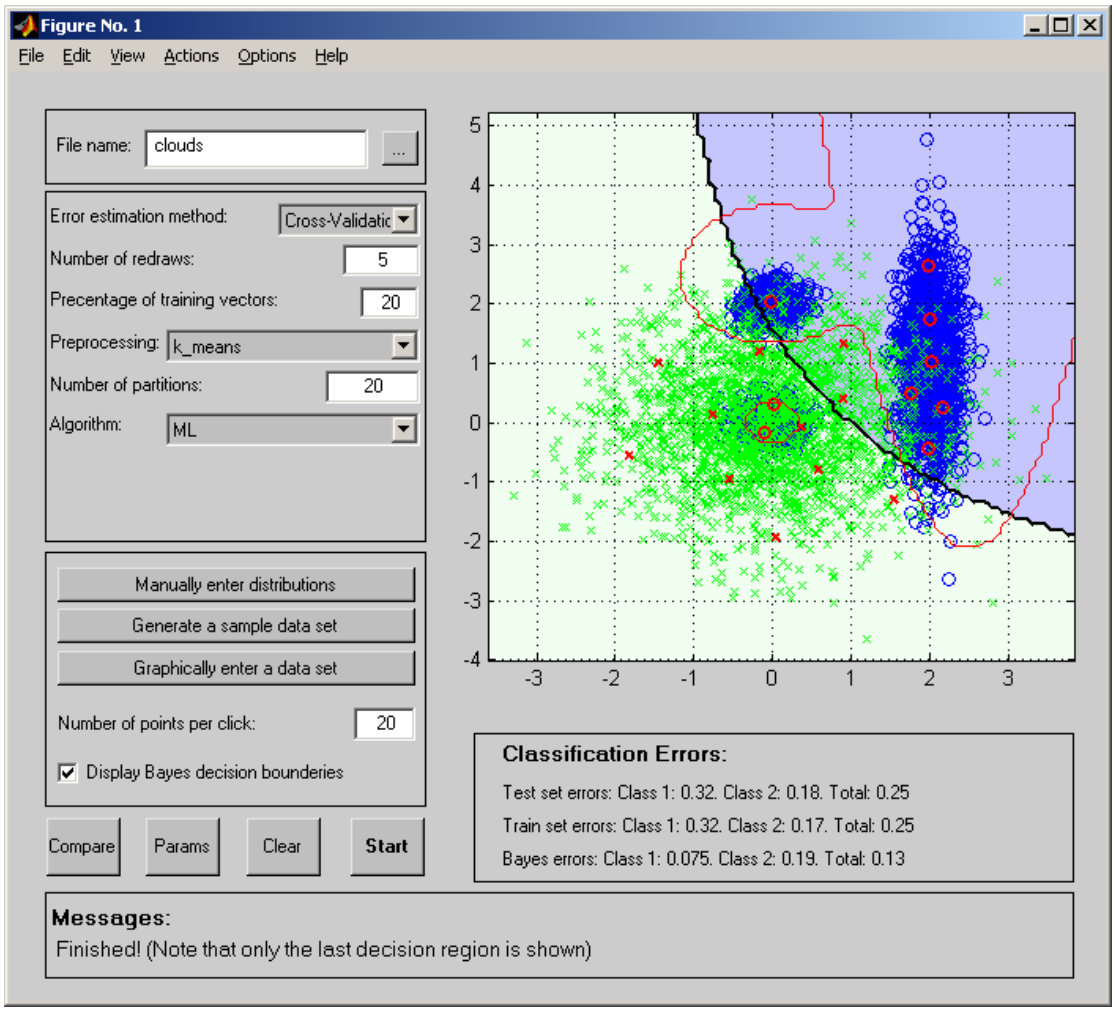

Figure 2: Results of the comprehensive example

Comparing the performance of several algorithms

The `classifier` GUI is useful when a specific algorithm is tested. However, if several algorithms need to be compared, a second GUI should be used. This GUI is loaded by pressing "Compare" on the classifier GUI, or by typing `multialgorithms` on the Matlab command window.

This GUI, shown in Figure 3, allows the selection of several algorithms for comparison. The algorithms are selected by marking them and moving them to the box on the right using the appropriate arrow. Then, the number of processing redraws needs to be entered, together with the error estimation method. To start the GUI, press the start button. The result is a bar graph (shown on a new figure), which displays the error percentage for each algorithm, togeather with the Bayes error (if the parameters of the distribution were entered).

In addition, using learning curves the performance of several algorithms can be predicted by pressing the appropriate button.

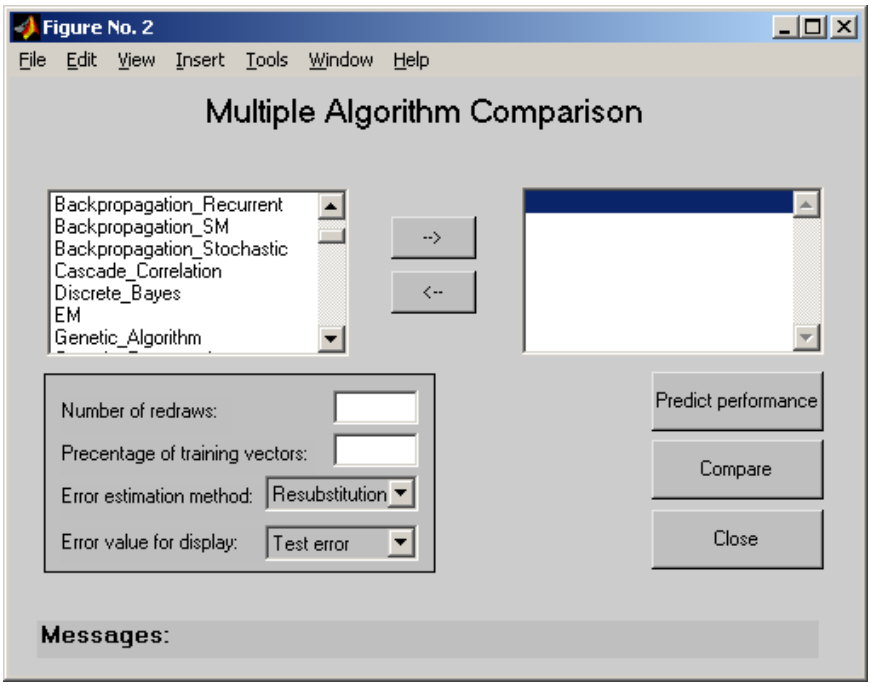

Figure 3: GUI for comparing multiple algorithms

An example

We now return to the same example given for the single algorithm testing. Suppose we want to test the maximum likelihood algorithm on features reduced using several algorithms on the "clouds" data set. We will use 5-fold cross-validation. To do all this, the following steps are needed:

1. Load the clouds data set by typing `clouds` in the file name box of the `classifier` GUI.
2. Press `compare` on the `classifier` GUI.
3. Change error estimation method to `cross-validation`, and choose 5 redraws in the box below.
4. Choose the algorithms you want to use for classification.
5. Press start.

The result is a screen similar to that shown in figure 4, where the average error rates of the 5-redraws (of the test sets) are reported, togeather with the Bayes error.

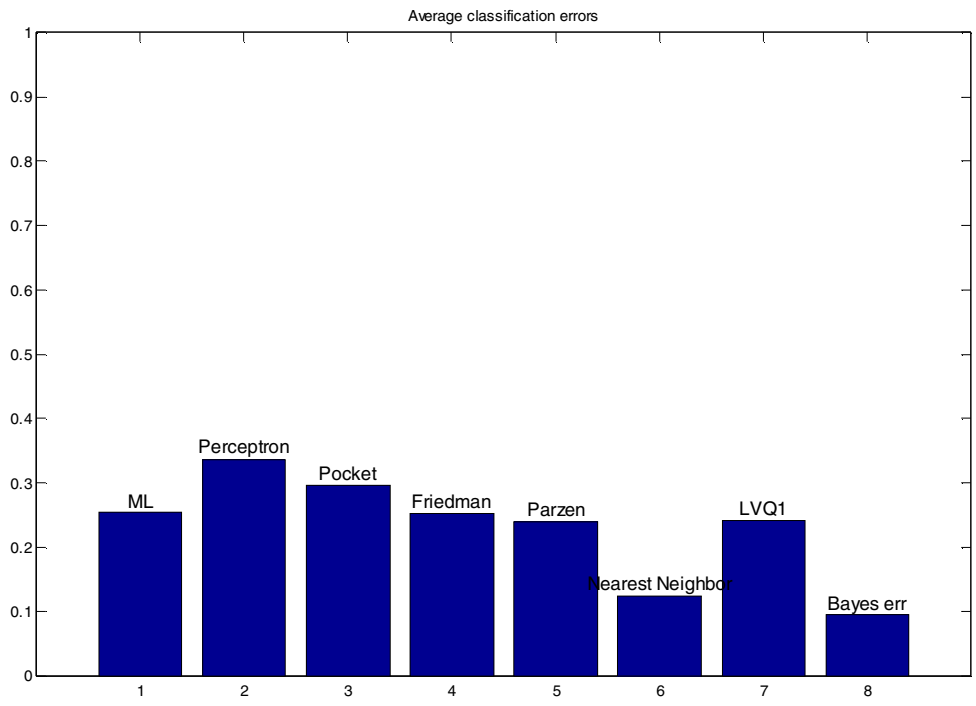

Figure 4: Results of the example

File structures

One type of file is used by the classification toolbox. This files can contains distribution features and/or the parameters of distributions. This chapter describes the structure of the file.

Distributions are stored as two varibles:

1. patterns: This variable holds the examples matrix. It is of size DxN, where N is the number of examples, and D is the number of dimensions of the data. Usually D=2. If D is larger than 2, a feature selection GUI will open and request the user to select a feature selection method in order to reduce the data to two dimensions.
2. targets: This vector holds the label of the features. It is of size 1xN, where N is the number of examples. The targets should be either 0 or 1.

The variables should be named “patterns” and “targets” in order for the GUI to read the distribution correctly.

If the distribution is a mixture of Gaussians, and their parameters are known, these parameters can be stored in a structure for computation of the Bayes error. This structure should be named *distribution_parameters*. The fields of this structure are:

Mu	Nx2	Means of the structures in each class.
sigma	Nx2x2	Covariance matrices of the structures if they are Guassian, or width of the Uniform distribution.
w	Nx1	The relative weight of each of the structures. The sum of each of these vectors should be 1.
P	Scalar	The probability of the class.

Note: N is the number of structures in a class, which can be different for each of the two classes.

Classifying using the text-based interface

Data can be classified using the normal text-based interface. For example, suppose we want to build an AdaBoost classifier using half of the data in the clouds dataset, and test it's error on the other half of the dataset. The stages for this are:

1. Load the dataset by typing: `load clouds`
2. Build an AdaBoost classifier (with a least squares (LS) weak learner) on half the data and classify the other half using the classifier:
```
test_targets=Ada_Boost(patterns(:,1:2500),targets(1:2
500),patterns(:,2501:end),'[100,''LS'',[]]');
```
3. Estimate the error:
```
error=mean(targets(2501:end) ~= test_targets)
```

The error should be in the order of 24.5%.

List of functions

This chapter details the list of functions found in the toolbox. This list is also found in the contents file of the toolbox.

GUI start commands	
Classifier	Start the classification GUI
enter_distributions	Starts the parameter input screen (used by classifier)
multialgorithms	Start the algorithm comparison screen
Preprocessing methods	
ADDC	Agglomerative clustering method
AGHC	Agglomerative hierarchical clustering method
BIMSEC	Basic iterative MSE clustering
Competitive_learning	Competitive learning algorithm
Deterministic_annealing	
Deterministic_SA	Deterministic stochastic annealing (Another implementation)
DSLVQ	Distinction sensitive linear vector quantization
Fisher linear discriminant	
Fuzzy_k_means	
fuzzy_k_means	
k_means	
Kohonen_SOFM	Kohonen self-organizing feature maps
Leader_follower	Basic leader-follower clustering
LVQ1	Linear vector quantization with one neighbor
LVQ3	Linear vector quantization 3 algorithm
Min_spanning_tree	Minimum spanning tree methods
PCA	Principal component analysis
SOHC	Stepwise optimal hierarchical clustering
Stochastic_SA	Stochastic simulated annealing
Parametric classification algorithms	
Balanced_Winnow	Balanced Winnow algorithm
Bayesian_Model_Comparison	Build a Gaussian model using Bayesian model comparison
EM	Expectation maximization algorithm
Gibbs	Gibbs algorithm
LMS	Least-means square algorithm
LS	Least squares algorithm
Marginalization	Classify with missing features

ML	Maximum likelihood algorithm
ML_diag	Maximum likelihood with diagonal covariance matrices
ML_II	Maximum likelihood model comparison
NDDF	Normal density discriminant function classifier
p_single	Used by EM algorithm
Perceptron	Single perceptron algorithm
Perceptron_Batch	Batch perceptron
Perceptron_BVI	Batch variable-increment perceptron
Perceptron_FM	Perceptron which improves on the example farthest from the margin
Perceptron_VIM	Variable-increment perceptron with margin
Pocket	Pocket algorithm
RDA	Regularized descriminant analysis
Relaxation_BM	Batch relaxation with margin
Relaxation_SSM	Single-step relaxation with margin
Stumps	A line in one of the dimensions that minimizes the error
Non-parametric classification algorithms	
Ada_Boost	Ada boost algorithm.
Backpropagation_Batch	Neural network trained with a batch backpropagation algorithm
Backpropagation_CGD	Neural network trained with a conjugate gradient descent algorithm
Backpropagation_Quickprop	Neural network trained with a batch quickprop backpropagation algorithm
Backpropagation_Recurrent	A recurrent neural network
Backpropagation_SM	Neural network trained with a stochastic backpropagation with momentum algorithm
Backpropagation_Stochastic	Neural network trained with a stochastic backpropagation algorithm
C4_5	The C4.5 algorithm
Cascade_Correlation	A cascade-correlation neural network
CART	Classification and regression trees
Components_with_DF	Component classifiers with descriminant functions
Components_without_DF	Component classifiers without descriminant functions
Deterministic_Boltzmann	Deterministic Boltzmann learning
Discrete_Bayes	Bayes classifier for discrete features
Genetic_algorithm	Basic genetic algorithm
Genetic_Programming	Genetic programming algorithm

ID3	Quinlan's ID3 algorithm
Local_Polynomial	Local polynomial fitting
LocBoost	Local boosting
LocBoostFunctions	Used by LocBoost
loglikelihood	Used by local_polynomial
Marginalization	Marginal distribution
Minimum_Cost	Minimum cost classification
Multivariate_Splines	Multivariate adaptive regression splines
Nearest_Neighbor	Nearest neighbor algorithm
NearestNeighborEditing	Nearest Neighbor Editing algorithm
Optimal_Brain_Surgeon	Prune a NN using the optimal brain surgeon algorithm
Parzen	Parzen windows algorithm
PNN	Probabilistic neural network
Projection_Pursuit	Projection pursuit regression for classification
RBF_Network	Radial basis function network
RCE	Reduced coulomb energy algorithm
Store_Grabbag	An improvement on the nearest neighbor algorithm
SVM	Support-vector machine algorithm
Voted_Perceptron	Voted perceptron algorithm
Feature selection	
Genetic_Culling	Culling genetic algorithm for feature selection
Koller	Koller algorithm for feature selection
ICA	Independent component analysis
Infomat	Calculate the mutual information matrix (Used by Koller)
HDR	Hierarchical dimensionality reduction
MDS	Multidimentional scaling
NLPCA	Non linear PCA
PCA	Principal component analysis
Error estimation	
Bhattacharyya	Compute the Bhattacharyya bound
calculate_error	Calculates the classification error given a decision surface
Chernoff	Compute the Chernoff bound
classification_error	Used by claculate_error
Classify_features	Classify new features given a decision surface
decision_region	Builds a decision region for multi-Gaussian distributions
Discriminability	Compute the discriminability for Gaussian distributions
Predict_performance	Predict performance using learning curves

GUI housekeeping functions	
calculate_region	Finds the data scatter region
classifier_commands	Classifier screen housekeeping commands
Click_points	Make a distribution by clicking on the screen
Enter_distribution_commands	Enter distributions housekeeping commands
Feature_selection	Feature selection GUI
Feature_selection_commans	Commands file for the feature selection GUI
find_classes	Find which classes exist in a data set
FindParamters	A GUI for selecting the best parameters for a classifier
FindParametersFunctions	Commands for the parameter selection GUI
GaussianParameters	Shows the estimated Gaussian parameters of the distribution (Caleed by classifier.m)
generate_data_set	Generate a data set given Gaussian parameters
High_histogram	Calculate histogram of N dimensional data
load_file	Load data files
make_a_draw	Randomly find indices from a data set
Multialgorithms_commands	Multialgorithms screen commands
plot_process	Plot partition centers during the algorithm execution
plot_scatter	Make a scatter plot of a data set
process_params	Read a parameter vector
Read_algorithms	Reads an algorithm file into a data structure
start_classify	Main function used by classifier
voronoi_regions	Plot Voronoi regions

References

[1] Duda RO, Hart PE, Stork DG, "Pattern Classification", John Wiley & Sons (2000).

[2] Guedalia ID, London M, Werman M, "An on-line agglomerative clustering method for nonstationary data". Neural computation, 11:521-540 (1999).

[3] Rose K, "Deterministic annealing for clustering, compression, classification, regression, and related optimization problems". Proceedings of the IEEE, 86(1): 2210-2239 (1998).

[4] Pregenzer M, Flotzinger D, Pfurtscheler G, "Distinction sensitive learning vector quantization: A new noise-insensitive classification method". Proceedings of 4th International Conference on Artificial Neural Networks. Cambridge, UK (1995).

[5] Friedman S, "Regularized discriminant analysis", Journal of the American statistical association, 84:165-175 (1989).

[6] Meir R, El-Yaniv R, Ben-David S, "Localized boosting", Proceedings of the Thirteenth Annual Conference on Computational Learning Theory.

[7] E. Yom-Tov, G.F. Inbar, "Selection of Relevant Features for Classification of Movements from Single Movement-Related Potentials Using a Genetic Algorithm". 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2001.

[8] D. Koller, M. Sahami, "Toward optimal feature selection," Machine Learning - Proceedings of the Thirteenth International Conference, pp. 284-292, 1996.

[9] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An introduction to kernel-based learning algorithms," IEEE Transactions on Neural Networks, 12(2):181-201, 2001.

Appendix: Steps for entering new preprocessing and classification algorithms

The classification toolbox is meant to be a development tool for testing solutions to practical problems. As such, it is sometimes useful to add new preprocessing and classification algorithms to the toolbox. This section describes the steps needed to do this.

Adding a new preprocessing algorithm

Preprocessing algorithm m-files can have three different structures, determined by the type of output from the algorithm. The format of these algorithms detemines where in the classification toolbox file they will be added.

All algorithms have four inputs: train patterns, train tagets, a parameter (or vector of parameters) for the algorithm, and a plot flag. The plot flag (either 0,1,2 or 3) determines if the user requested plots of the algorithm during training. In this case, it is recommended to use the `plot_process` function for this purpose.

All algorithms have at least two outputs, namely: Reduced patterns and reduced targets. Some have an additional parameter, which is used in order to reshape the test patterns. The three possible output formats are:

Additional output	Example	Line in `start_classify.m`
Matrix (Reshaping matrix)	PCA	129
Vector (Reshaping vector, reduces feature dimension to 1)	Fisher	138
None (No reshaping of test patterns)	K_means	151

Once a new algorithm has been written and its m-file copied to the toolbox directory, it also has to be added to the file `Preprocessing.txt`. In this file there exists a line for every preprocessing algorithm. The structure of each line is as follows:

<Algorithm name>@<Caption>@<Default Parameters>@<Display field>
Where:
<Algorithm name> is the algorithms' name, as well as the m-file name.
<Caption> is the cpation to be displayed near the paramter entry box.
<Default parameters> are the defaults for this algorithm.
<Display field>: Write 'N' in this field if no parameters are needed, otherwise, any other letter can be written here.

Finally, for your convenience, describe the file in `contents.m`.

Adding a new classification algorithm

Classification algorithm m-files receive training patterns and targets, test patterns and an algorithm specific parameter. The algorithms return the classified test targets.

Once a new classification algorithm has been written and its m-file placed in the classification toolbox directory, it needs to be written into two m-files for including it in the classification toolbox.

First, the new algorithm needs to be included in the `Classification.txt` file. In this file there exists a line for every preprocessing algorithm. The structure of each line is as follows:
<Algorithm name>@<Caption>@<Default Parameters>@<Display field>
Where:
<Algorithm name> is the algorithms' name, as well as the m-file name.
<Caption> is the cpation to be displayed near the paramter entry box.
<Default parameters> are the defaults for this algorithm.
<Display field>: Write 'N' in this field if no parameters are needed, otherwise, 'S' will open a short parameter window and 'L' wiill open a long one whenever this algorithm is selected.

Finally, for your convenience, describe the file in `contents.m`.

Adding a new feature selection algorithm

Feature selection algorithm m-files receive patterns and targets, togeather with algorithm specific parameters. The algorithms return the data reduced to a lower dimension.

Once a new feature selection algorithm has been written and its m-file placed in the classification toolbox directory, it needs to be written into two m-files for including it in the classification toolbox.

First, the new algorithm needs to be included in the `Feature_selection.txt` file. In this file there exists a line for every preprocessing algorithm. The structure of each line is as follows:
<Algorithm name>@<Caption>@<Default Parameters>@<Display field>
Where:
<Algorithm name> is the algorithms' name, as well as the m-file name.
<Caption> is the cpation to be displayed near the paramter entry box.
<Default parameters> are the defaults for this algorithm.
<Display field>: Write 'N' in this field if no parameters are needed, otherwise, any other letter can be written here.

Finally, for your convenience, describe the file in `contents.m`.

Index

M

N

O

P

Q