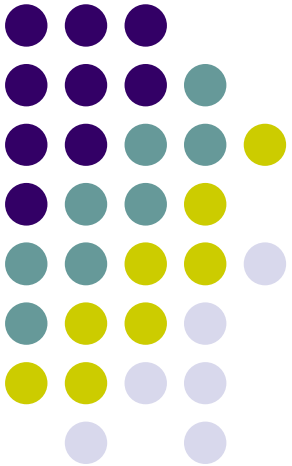
Examples to Matlab Toolboxes

Nigar ŞEN KÖKTAŞ

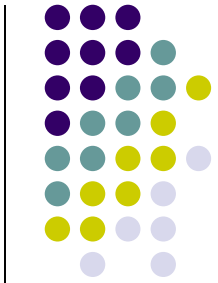
07.02.2008



Outline

- Key features of toolboxes
- Examples
  - NN toolbox
    - Functions
    - GUI functions
    - An example
  - Statistical toolbox
    - Functions
    - A classification example

Key features

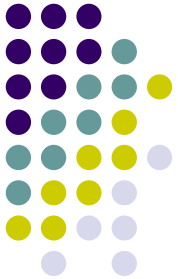


- Many of the toolbox functions are MATLAB M-files
- They implement the specialized algorithms
- To view and edit code you can type
  - edit function_name
- You can extend the capabilities of toolboxes by writing your own M-files, or by using the toolbox in combination with other toolboxes
- Examples: Neural Network Toolbox, Signal Processing Toolbox and Statistics toolbox

Neural Network Toolbox

- Analysis Functions
- Distance Functions
- Graphical Interface Functions
- Layer Initialization Functions
- Learning Functions
- Line Search Functions
- Net Input Functions
- Network Initialization Function
- Network Use Functions
- New Networks Functions
- Performance Functions
- Plotting Functions
- Processing Functions
- Simulink Support Function
- Topology Functions
- Training Functions
- Transfer Functions
- Utility Functions
- Vector Functions
- Weight and Bias Initialization Functions
- Weight Functions
- Transfer Function Graphs

Functions cont.



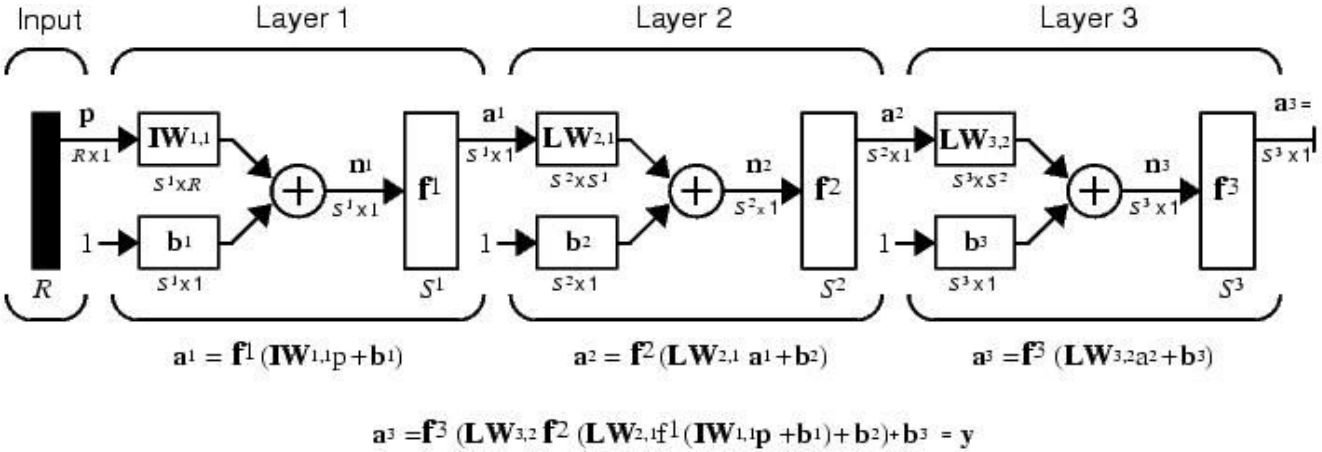
Layer Initialization Functions

initnw	Nguyen-Widrow layer initialization function
initwb	By-weight-and-bias layer initialization function

Learning Function

learncon	Conscience bias learning function
learngd	Gradient descent weight/bias learning function
learngdm	Gradient descent with momentum weight/bias learning function
learnh	Hebb weight learning function
learnhd	Hebb with decay weight learning rule
learnis	Instar weight learning function
learnk	Kohonen weight learning function
learnlv1	LVQ1 weight learning function
learnlv2	LVQ2 weight learning function
learnos	Outstar weight learning function
learnp	Perceptron weight and bias learning function
learnpn	Normalized perceptron weight and bias learning function
learnsom	Self-organizing map weight learning function
learnwh	Widrow-Hoff weight and bias learning rule

Network Architecture

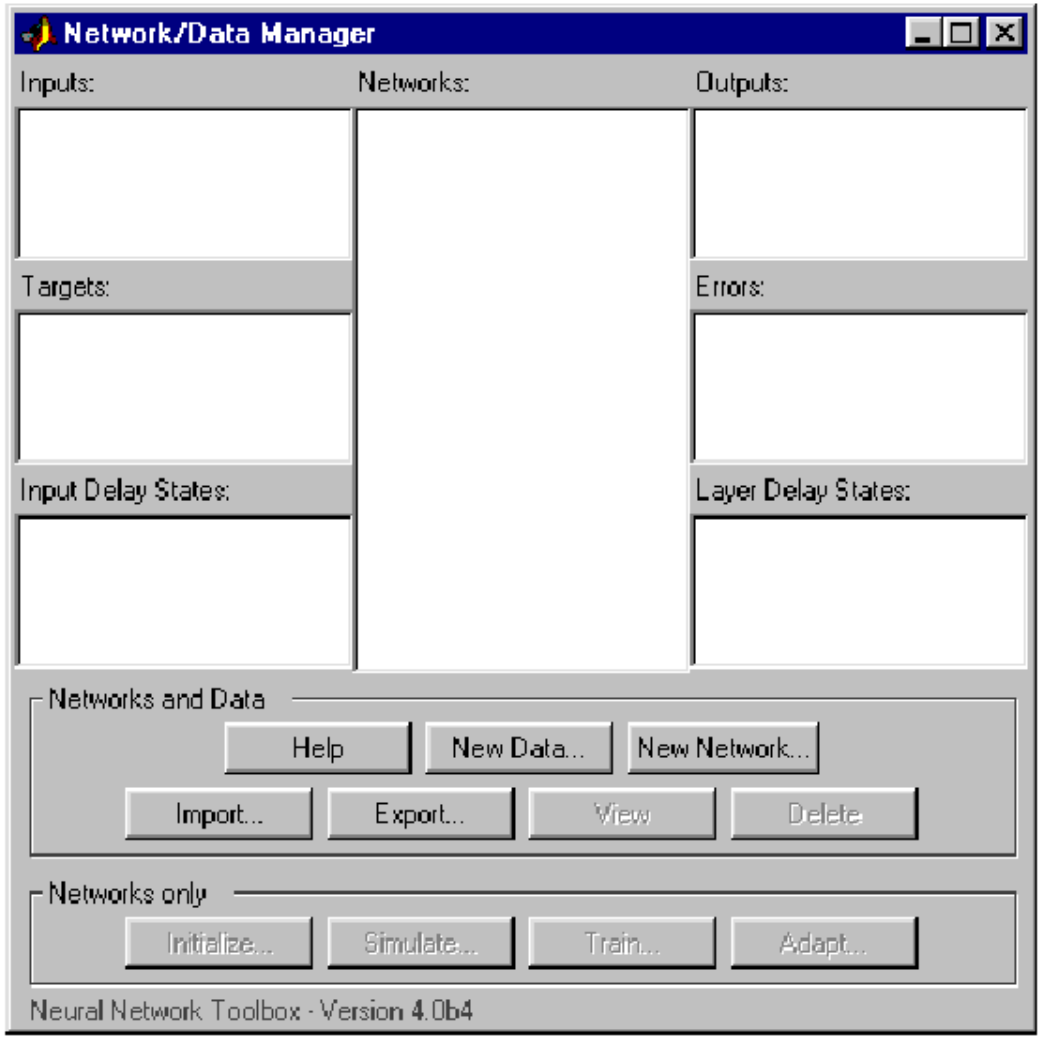


- Here the input vector **p** is represented by the solid dark vertical bar at the left.
- A *layer* includes the combination of the weights, the multiplication and summing operation (here realized as a vector product **Wp**), the bias *b*, and the transfer function *f*.
- Each time this abbreviated network notation is used, the sizes of the matrices are shown just below their matrix variable names.
- This notation will allow you to understand the architectures and follow the matrix mathematics associated with them.

Example

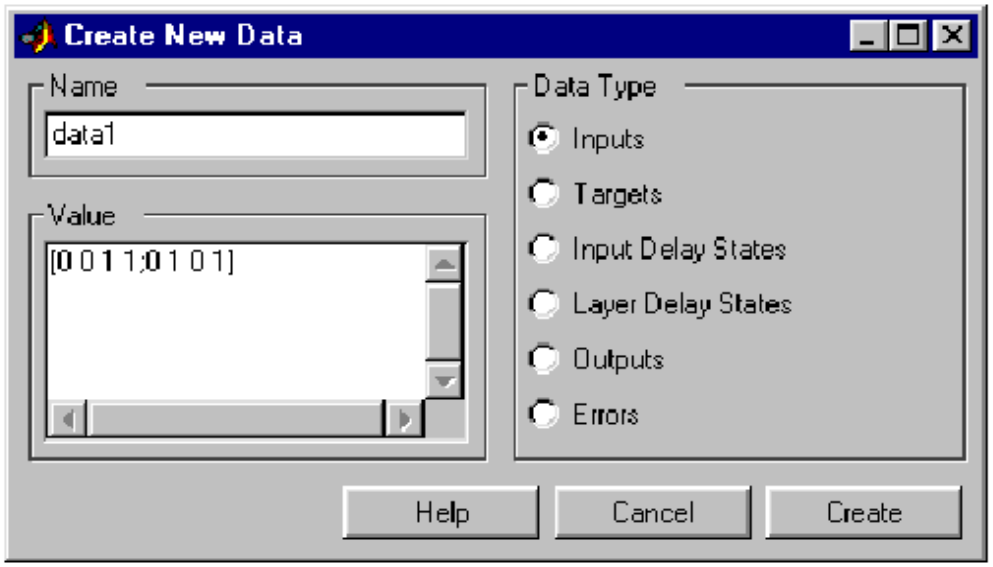
- `net = newp([-2 2;-2 +2],1);`
- `net.trainParam.epochs = 1;`
- The input vectors and targets are
  - `p = [[2;2] [1;-2] [-2;2] [-1;1]] t =[0 1 0 1]`
- Now train the network with
  - `net = train(net,p,t);`
- The new weights and bias are
  - w = -3 -1 b = 0
- Finally, simulate the trained network for each of the inputs.
  - `a = sim(net,p)`
  - `a = 0 0 1 1`
- The outputs do not yet equal the targets, so you need to train the network for more than one pass. Try four epochs. This run gives the following results:
  - `TRAINC, Epoch 0/20`
  - `TRAINC, Epoch 3/20`
  - `TRAINC, Performance goal met.`
- The final weights and bias are
  - `w = -2 -3 b = 1`
- The simulated output and errors for the various inputs are
  - `a = 0 1.00 0 1.00`
  - `error = [a(1)-t(1) a(2)-t(2) a(3)-t(3) a(4)-t(4)]`
  - `error = 0 0 0 0`

Graphical user interface



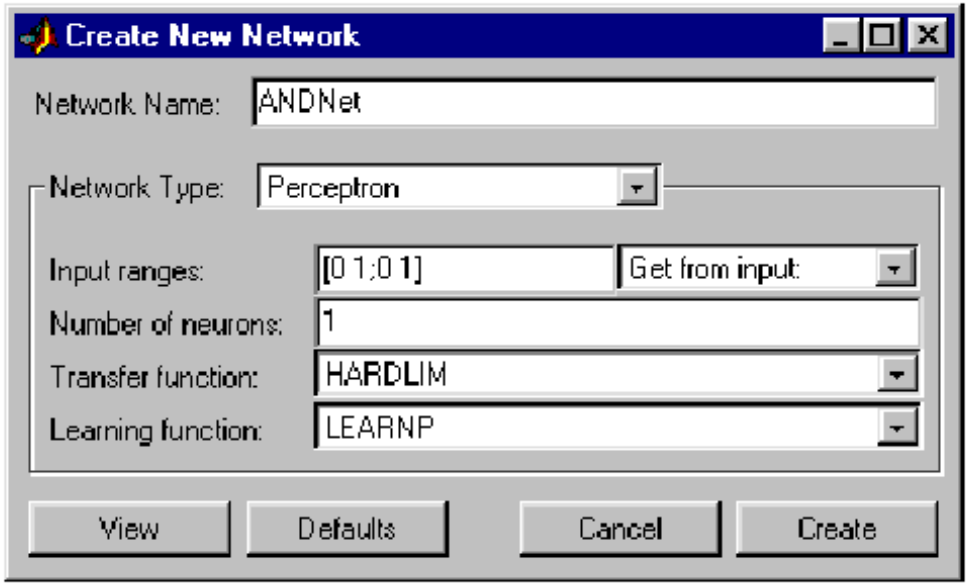
Create data

- Click **Create and then click OK** to create an input p. The Network/Data Manager window appears, and p shows as an input.
- Next create a network target. This time enter the variable name t, specify the value [0 0 0 1], and click **Target** under **Data Type**. Again click **Create and OK.** You will see in the resulting Network/Data Manager window that you now have t as a target as well as the previous p as an input.



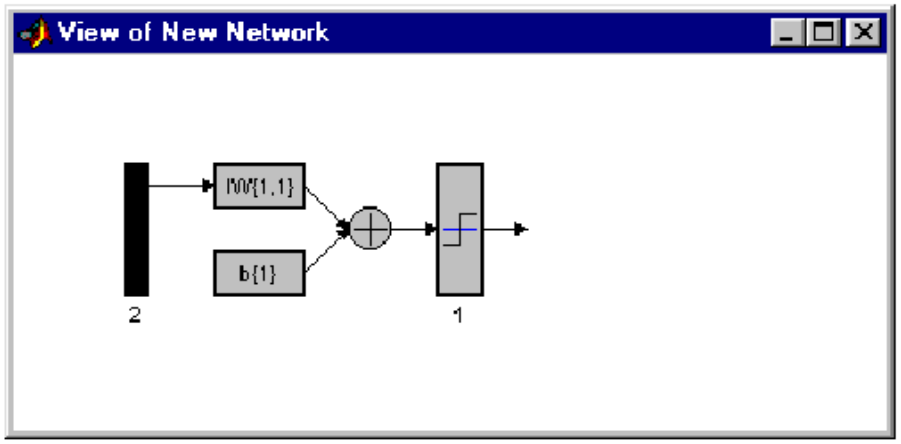
Create Network

- Now create a new network and call it ANDNet. Select the **Network** tab. Enter ANDNet under **Name**. Set the **Network Type** to Perceptron, and other parameters as shown in figure



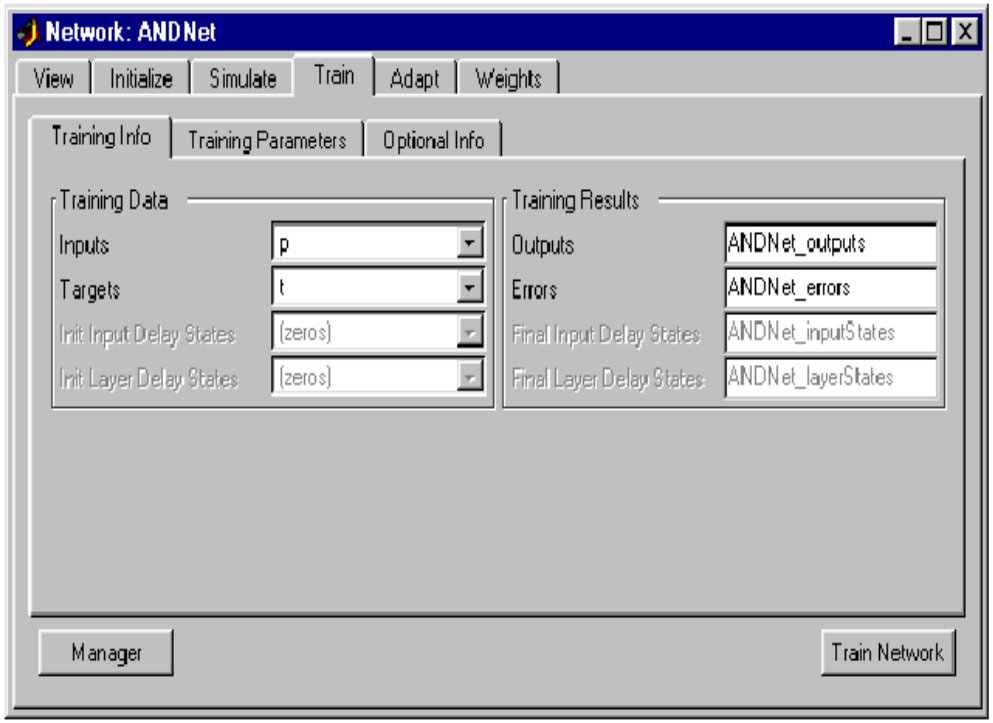
View network

- Next you can look at the network by clicking **View**.
- This picture shows that you are about to create a network with a two units of inputs, a hardlim transfer function, and a single output. This is the perceptron network that you want.

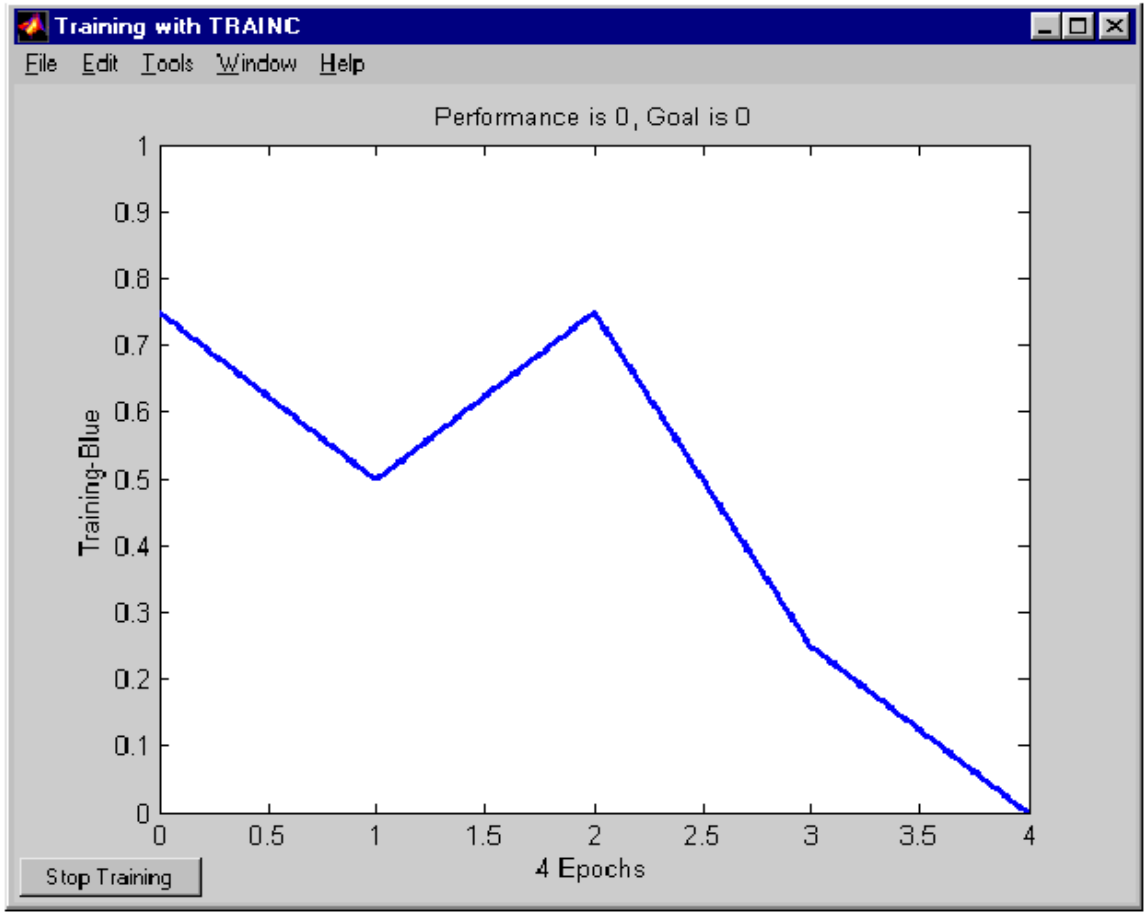


Train perceptron

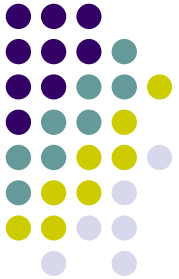
- To train the network, click ANDNet to highlight it. Then click **Open**.
- This leads to a new window, labeled **Network: ANDNet**.
- You can check on the initialization by clicking the **Initialize** tab.
- Now click the **Train** tab. Specify the inputs and output by clicking the **Training Info** tab and selecting **p** from the list of inputs and **t** from the list of targets.



Training result

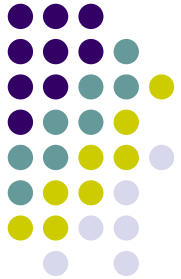


Statistics Toolbox



- Data organization and management
- Descriptive statistics
- Statistical plotting and data visualization
- Probability distributions
- Analysis of variance (ANOVA)
- Linear and nonlinear modeling
- Multivariate statistics
- Design of Experiments (DOE)
- Hypothesis testing
- Statistical Process Control (SPC)

Functions



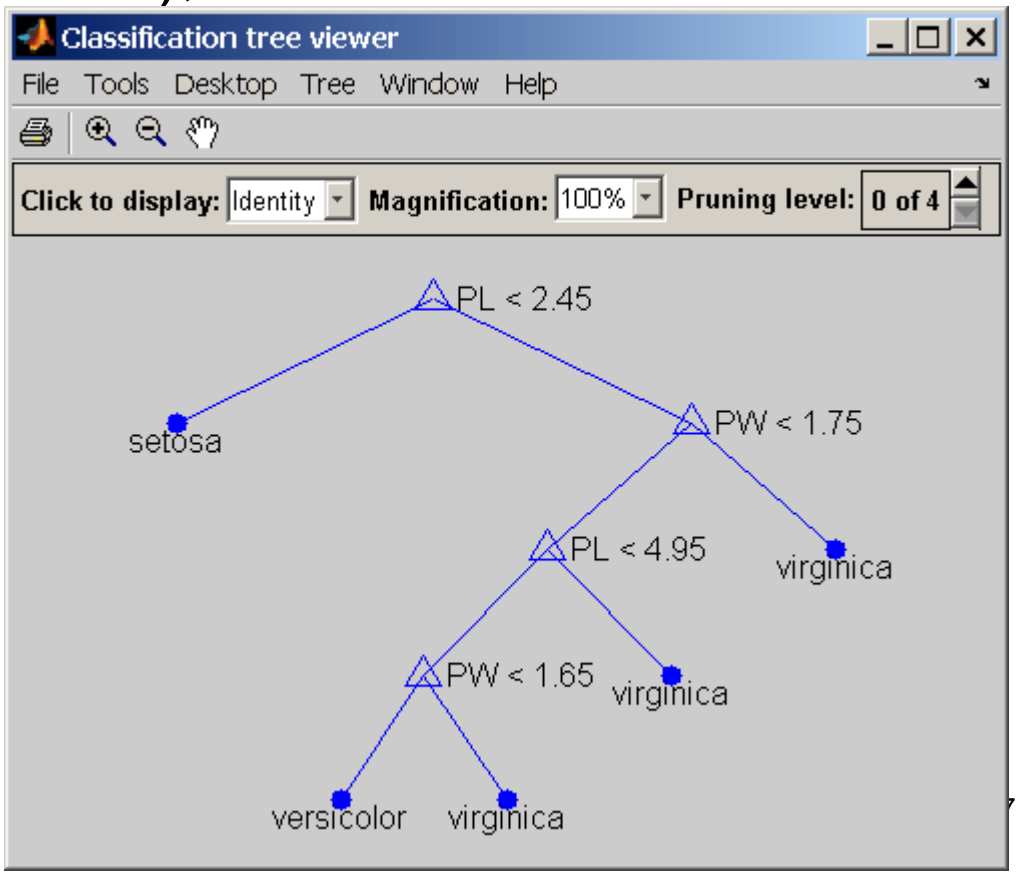
File I/O	Data input/output with external files
Organizing Data	Preparing for statistical processing
Descriptive Statistics	Summaries of data
Statistical Visualization	Plotting data patterns and trends
Probability Distributions	Describing distributions of data
Hypothesis Tests	Inferential statistics
Analysis of Variance	Explaining sample variance
Regression Analysis	Continuous data modeling
Multivariate Methods	Dimension reduction techniques
Cluster Analysis	Identifying data clusters
Classification	Categorical data modeling
Markov Models	Stochastic data modeling
Design of Experiments	Active data collection
Statistical Process Control	Monitoring industrial processes
Graphical User Interfaces	Interactive tools
Utility Functions	General purpose

Classification functions

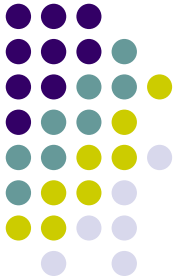
- t = treefit(X,y)
  t = treefit(X,y,*param1*,*val1*,*param2*,*val2*,...)
- t = treefit(X,y) creates a decision tree t for predicting response y as a function of predictors X.
- X is an n-by-m matrix of predictor values.
- y is either a vector of n response values (for regression), or a character array or cell array of strings containing n class names (for classification).
- Either way, t is a binary tree where each non-terminal node is split based on the values of a column of X.

Example

- load fisheriris;
- t = treefit(meas,species);
- treedisp(t,'names', {'SL' 'SW' 'PL' 'PW'});



Example cont.

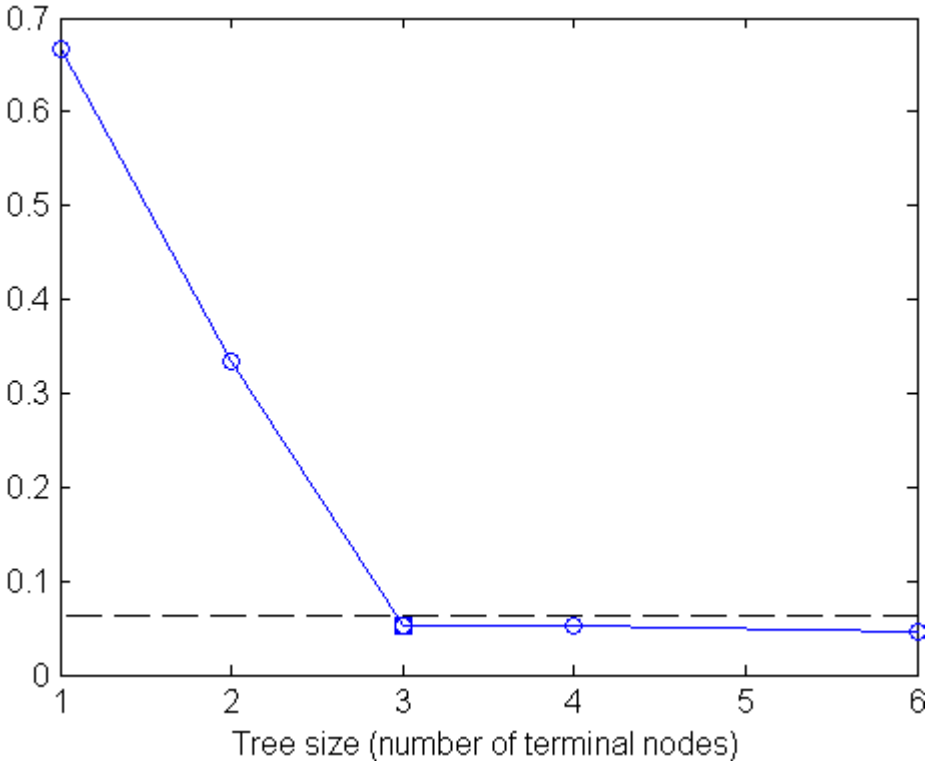


- *Start with a large tree.*
- load fisheriris;
- t = **treefit**(meas,species','splitmin',5);

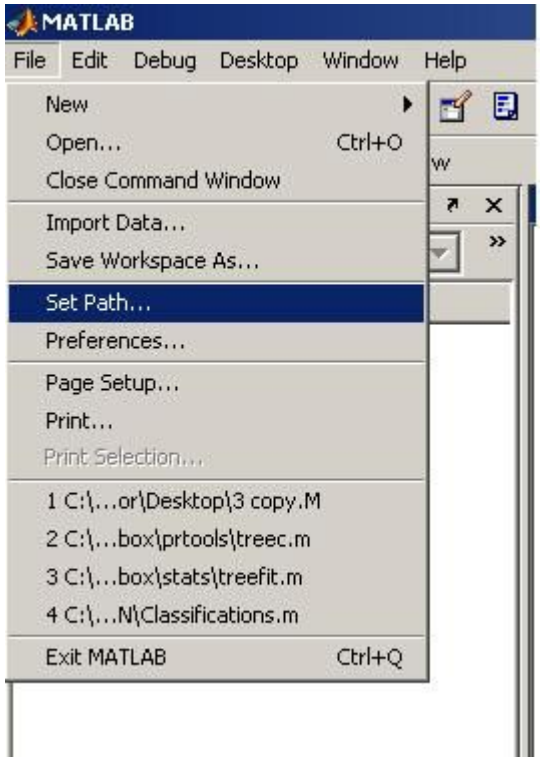
- *Find the minimum-cost tree.*
- [c,s,n,best] = **treetest**(t,'cross',meas,species);
- tmin = **treeprune**(t,'level',best);

- *Plot smallest tree within 1 std of minimum cost tree.*
- [mincost,minloc] = min(c);
- plot(n,c,'b-o',n,c+s,'r:',n(best+1),c(best+1),'bs', n,(mincost+s(minloc))*ones(size(n)),'k—'); xlabel('Tree size (number of terminal nodes)')

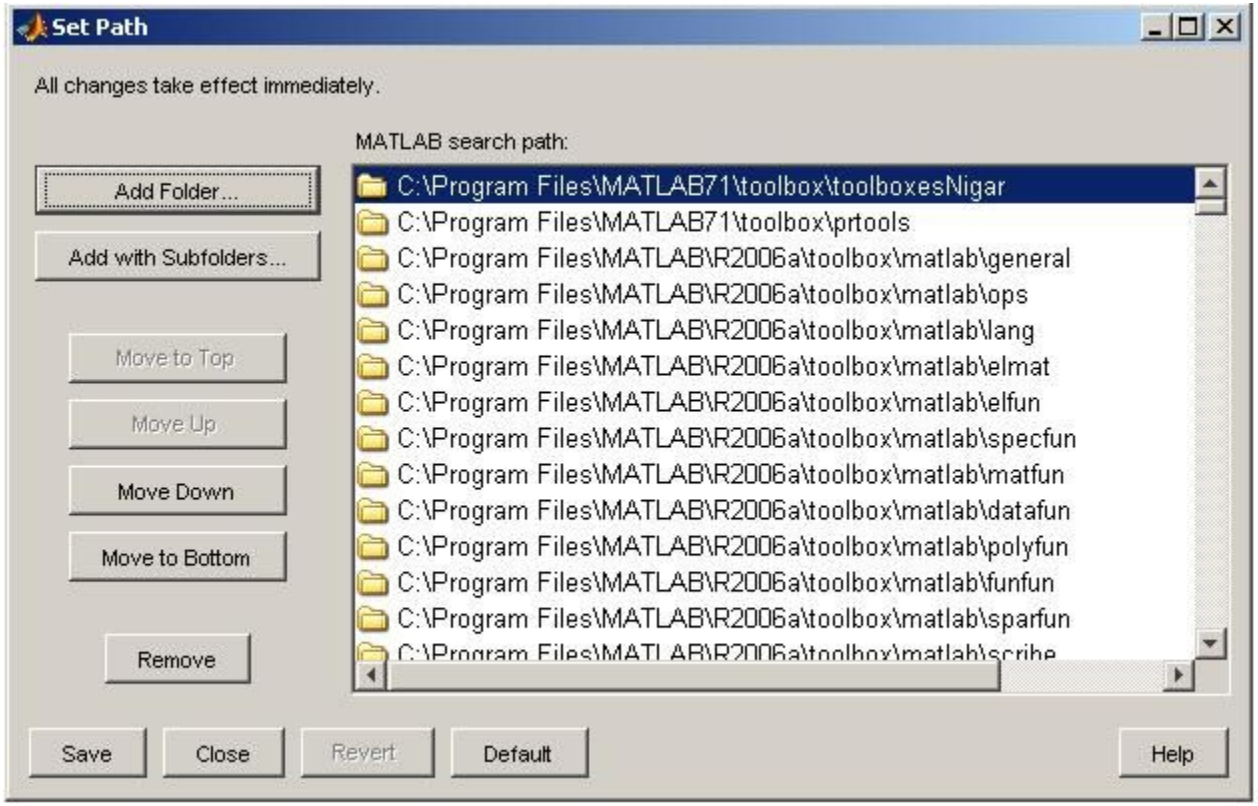
Result of plot function



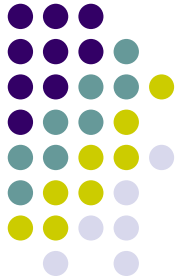
Adding a new toolbox



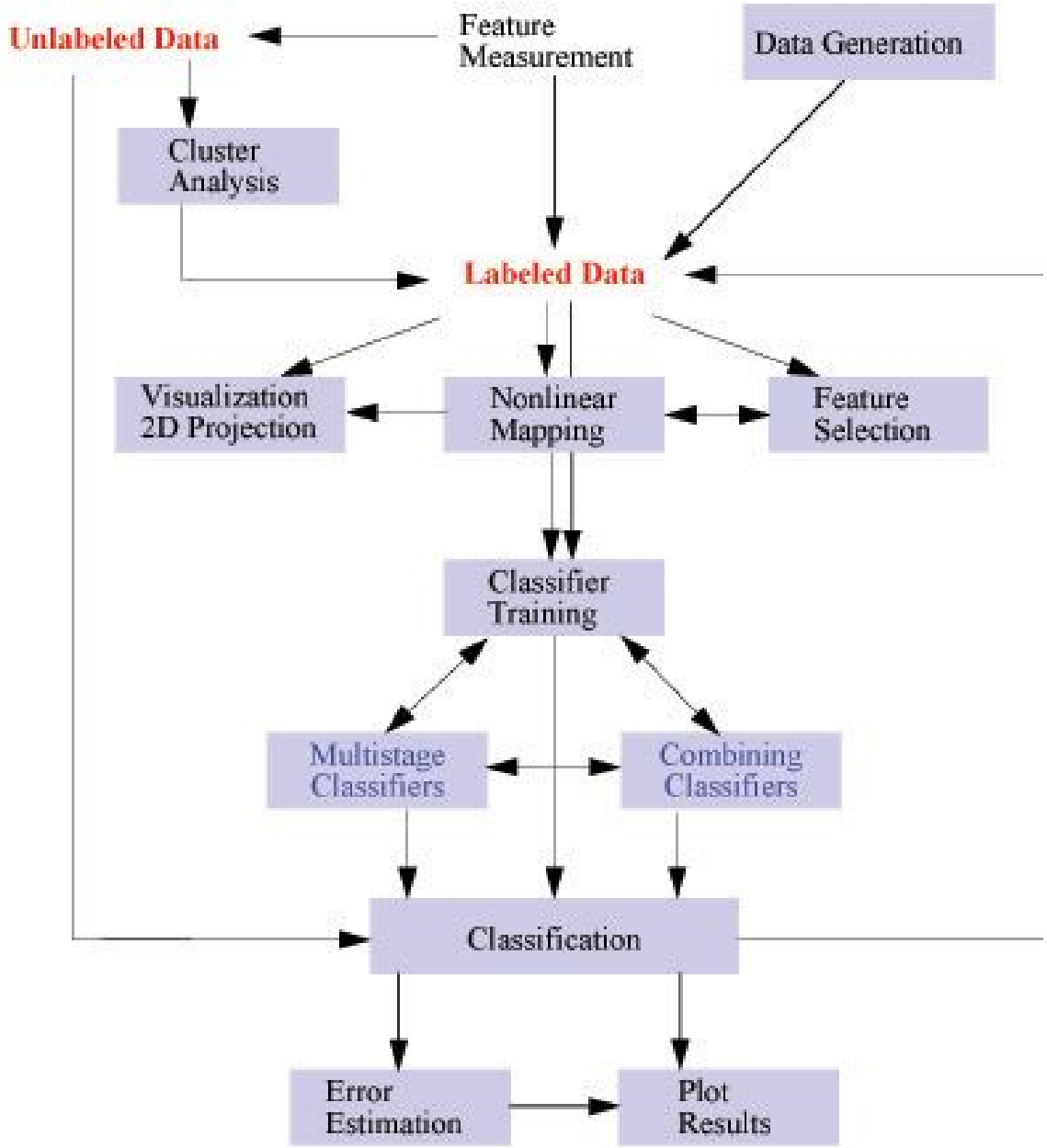
Cont.


Set Path
All changes take effect immediately.
MATLAB search path:
Add Folder...
Add with Subfolders...
Move to Top
Move Up
Move Down
Move to Bottom
Remove
C:\Program Files\MATLAB71\toolbox\toolboxesNigar
C:\Program Files\MATLAB71\toolbox\prtools
C:\Program Files\MATLAB\R2006a\toolbox\matlab\general
C:\Program Files\MATLAB\R2006a\toolbox\matlab\ops
C:\Program Files\MATLAB\R2006a\toolbox\matlab\lang
C:\Program Files\MATLAB\R2006a\toolbox\matlab\elmat
C:\Program Files\MATLAB\R2006a\toolbox\matlab\elfun
C:\Program Files\MATLAB\R2006a\toolbox\matlab\specfun
C:\Program Files\MATLAB\R2006a\toolbox\matlab\matfun
C:\Program Files\MATLAB\R2006a\toolbox\matlab\datafun
C:\Program Files\MATLAB\R2006a\toolbox\matlab\polyfun
C:\Program Files\MATLAB\R2006a\toolbox\matlab\funfun
C:\Program Files\MATLAB\R2006a\toolbox\matlab\sparfun
Save
Close
Revert
Default
Help

PRTools toolbox



- PRtoolsPRTools supplies about 200 user routines for traditional statistical pattern recognition tasks.
- It includes procedures for data generation, training classifiers, combining classifiers, features selection, linear and non-linear feature extraction, density estimation, cluster analysis, evaluation and visualization.
- It is intended to aid students and researchers in designing and evaluating new algorithms and in building prototypes.
- PRTools can be freely used for academic research.
- www.prtools.org


Unlabeled Data
Feature Measurement
Data Generation
Cluster Analysis
Labeled Data
Visualization 2D Projection
Nonlinear Mapping
Feature Selection
Classifier Training
Multistage Classifiers
Combining Classifiers
Classification
Error Estimation
Plot Results

Thank you for listening