Iterative Denoising Toolbox (IDT) User Manual

Document Information

Bennett Landman, landman@jhu.edu
IDT Package Release 1.0, January 28, 2009
Document Version: 0.2, March 25, 2009

Rationale

1. The IDT provides a toolkit for developing classifiers based on iterative denoising trees. This behaves much like other clustering/classification routines. In fact, IDT combines other clustering and classification routines into a decision tree structure. All functions are passed as function handles so that the user may write their own branch/classification rules or use any of the already implemented functionality. The demonstration code provides one simple illustration of assessing the classification error of these IDT classifiers based on a leave-N-model.
2. Use of IDT in other applications is encouraged and supported. However, the parameters related to constraints on the tree structure would likely need to be adapted. Please see the documentation on `buildIDT.m` and `classifyIDT.m` for the detailed specifications on how to use this functionality.

Requirements

1. Software:
   - Matlab (The Mathworks, Natick, MA) 7.7.0 (R2008b) or compatible.
   - IDT zip file.
2. Hardware
   - Any system compatible with the corresponding version of Matlab.

Installation

1. Unzip the distribution package.
2. To use the package temporarily, change the current Matlab directory to the newly extracted directory containing “idt.m”.
3. To make IDT functionality permanently available, add the newly extracted directory to the Matlab path.

Verify Installation

1. Type “help IDTdemo” at the Matlab command prompt. This should result in the following output:

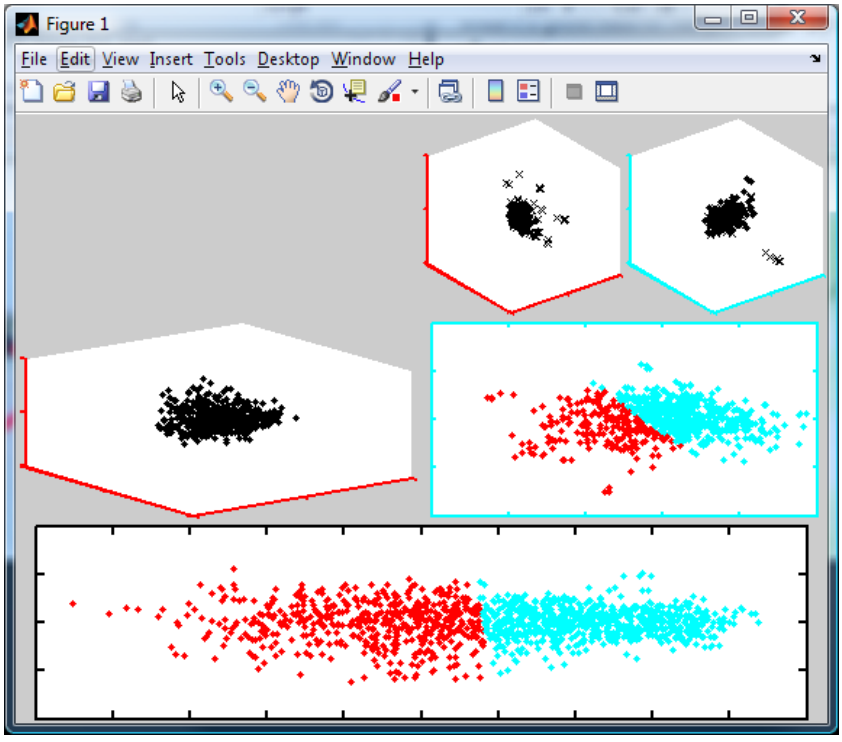
```
>> help IDTdemo
```

```
IDTdemo - run the demo described in idt.m
See idt.m for more information.
*********************************************
Iterative Denoise Toolbox rev 1.0
     Bennett Landman, landman@jhu.edu
  (C) Copyright 2009, Bennett Landman
  Released under Lessor GNU Public License v1.0
     Not for clinical use.
     No warranty expressed or implied.
  Use at your own risk.
*********************************************
1/29/09 bl
```

2. If this does not work, verify that the command is correctly in your working path.
3. To see a demo of the IDT at work, type "IDTdemo". This command takes about 30 seconds to execute on a 2 GHz notebook computer. This should result in the following output and figure:

```
>> IDTdemo

Estimated classification error rate: 3.2%
Note: LO procedure could be repeated to estimate variability of error estimate
The figures shows the IDT from root (bottom) to leaves. For branch
points, the border color shows to which data in the parent node are
shown while the color of the plotted points shows the branch membership.
The first two PCA of the data in the branch point are shown. For leaf
nodes, the the border color shows to which data in the parent node are
shown. The marker symbol shows the label class of the data. The leaf
nodes show the first 3 PCA components of the data in leaf node.
```



4. This demonstration uses a set of feature vectors derived from image data. The dataset is represented by an NxM matrix where there are N objects under observations and M observations made on each object. The lowest level of the figure shows the first two PCA

components (X and Y) of the matrix for each observation (points) colored by the class-label blind branching procedure (k-means clustering). The next row up shows the classification (left) of the left branch using a linear classifier while the right branch shows a further subdivision of the feature space using k-means. The top row shows the final classification of the remaining two leaf nodes. The classification error (as measured by leaving out 500 objects) is indicated in the text above.

Use of the IDT

1. This toolbox provide support for automatically constructing IDT based on arbitrary branching and classification criteria. These criteria are specified by training functions which return functions that act as classifiers for branching (denoising) and for leaf classification (classifiers).

```
% IDT Classification Architecture
  buildIDT -  construct IDT based on training data
  classifyIDT -   use IDT's to classify data with unknown labels

% Denoising (branching classifiers)
  trainDenoisePCA - use k-means to create branches based on PCA of the
                    input data
  trainDenoiseDirect - use k-means to create branches based on the input
                    data
% Leaf Classifiers
  trainClassifierFDA - Fisher's Linear Discriminant
  trainClassifierKnn1 - First Nearest Neighbor
  trainClassifierKnnN - K-th Nearest Neighbor
```

2. These function use a command basis set of data structures:
   - Data - 1x2 cell array, data to be used for training.
     Data in the first cell is used for branching.
     Data in the second cell is used for classification at leaf-nodes.
     The infrastructure is agnostic as to the format within these cells.
     Example classifiers use a NxM matrix where N is the number of subjects and M are the observations for each subject.

   - Labels - Class labels in a matrix for each of the subjects in the training data.
     The infrastructure is agnostic as to the format of Labels.
     Example classifiers use a Nx1 vector where N is the number of subjects

   - Function Handles
     trainDenoise - a matlab function handle of the "IDT denoising type"
     see trainDenoisePCA for the proper input/output parameters

     trainClassifier - a matlab function handle of the "IDT classifiying

see trainClassifierFDA for the proper input/output parameters

- Parameters - a structure object with parameters controling IDT tree growth as well as the denoising and the classifying functions. The parameters structure is directly passed to both members. Please see the documentation embedded in each file for specifics as to the currently supported parameters.

3. An outline of this functionality and the following code used to create the above demonstration is available by typing "idt" at the Matlab command prompt. This results in the following output:

```
>> idt
 IDT: Iterative Denoising Toolbox
% Release 1.0, January 28, 2009
 ********************************************
 Iterative Denoise Toolbox rev 1.0
      Bennett Landman, landman@jhu.edu
   (C) Copyright 2009, Bennett Landman
   Released under Lessor GNU Public License v1.0
      Not for clinical use.
      No warranty expressed or implied.
   Use at your own risk.
 ********************************************
 1/28/09 bl

 This toolbox provide support for automatically constructing IDT based on
 arbitrary branching and classification criteria. These criteria are
 specified by training functions which return functions that act as
 classifiers for branching (denoising) and for leaf classification
 (classifiers)

% IDT Classification Architecture
   buildIDT -  construct IDT based on training data
   classifyIDT -   use IDT's to classify data with unknown labels

% Denoising (branching classifiers)
   trainDenoisePCA - use k-means to create branches based on PCA of the
                     input data
   trainDenoiseDirect - use k-means to create branches based on the input
                     data
% Leaf Classifiers
   trainClassifierFDA - Fisher's Linear Discriminant
   trainClassifierKnn1 - First Nearest Neighbor
   trainClassifierKnnN - K-th Nearest Neighbor

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Demonstration Code
 addpath(genpath(fileparts(which('idt'))),'-END')
 load IDTexampleData;      % Load sample data

 %% Setup parameters
 Parameters = [];
```

```
  Parameters.tree.minPointsPerBranch = 100;
  Parameters.tree.maxTreeDepth = 3;
  Parameters.tree.maxNumBranches = 2;
  Parameters.tree.encodingBasis = 10;
  Parameters.denoise.useBIC = 0;
  Parameters.denoise.numPCAComp = 2;
  Parameters.classify.FDAMinPct = .9;
  Parameters.classify.homogenietyThreshold = 0.9;

  N1=200;           % Just use the first 200 features
  NLO=500;          % Leave out 500 for validation

  %% Randomize the leave-out criteria
  List = randomize(1:length(imgFeatures));
  LeaveOut = List(1:NLO);
  NotLeaveOut = setxor((1:length(imgFeatures)),1);

  %% Setup the training data
  Data{1}=imgFeatures(NotLeaveOut,1:N1);
  Data{2}=imgFeatures(NotLeaveOut,1:N1);
  Labels=trueLabels(NotLeaveOut,:);

  %% Setup the test data
  testData{1}=imgFeatures(LeaveOut,1:N1);
  testData{2}=imgFeatures(LeaveOut,1:N1);
  testLabels = trueLabels(LeaveOut,:);

  %% Build the IDT
  [resultIDT,terminationCriteria]=
buildIDT(Data,Labels,@trainDenoisePCA,@trainClassifierFDA,Parameters);

  %% Use the IDT to classify unlabeled data
  estTestLabels = classifyIDT(testData,resultIDT);

  %% Compute the LO error rate
  LOErr=mean(estTestLabels ~=testLabels);


  figure;plotIDT(Data,Labels,IDT)
  disp(sprintf('Estimated classification error rate: %.1f%%',LOErr*100));
  disp('Note: LO procedure could be repeated to estimate variability of error
estimate');
  disp('The figures shows the IDT from root (bottom) to leaves. For branch');
  disp('points, the border color shows to which data in the parent node are');
  disp('shown while the color of the plotted points shows the branch
membership.');
  disp('The first two PCA of the data in the branch point are shown. For
leaf');
  disp('nodes, the the border color shows to which data in the parent node
are');
  disp('shown. The marker symbol shows the label class of the data. The leaf');
  disp('nodes show the first 3 PCA components of the data in leaf node.');
```

Notes on Data Sources

1. The IDT assumes that there are two, not necessarily distinct, feature sets. The first data set determines the features that are used in the branching process (training does not use labels). The second data set is used for classification at leaf nodes (training uses labels). The paired data are stored as 2x1 cell arrays. The first array is the branching data, while the second array is the classification data. Each dataset is stored as an NxM matrix for N objects and M features per object.
2. See the idtdemo.m code for examples based on the data in IDTexampleData.mat.