Adaptive Local Hyperplane (ALH) Classifier Toolbox

April 2008

Introduction

The toolbox is used for solving the classification, i.e. pattern recognition, tasks by the adaptive local hyperplane algorithm as introduced in the reference paper below. Note that the files can be run in MATLAB's version 7 and higher. (Sorry, it doesn't work for vs. 5 and 6).

Copyrights

Tao Yang tyan028@gmail.com
Vojislav Kecman v.kecman@auckland.ac.nz
Faculty of Engineering, The University of Auckland, Auckland, New Zealand

Note

The software is for an academic use only and it is not to be used for any commercial purposes.

Reference

Yang T., Kecman V., Adaptive local hyperplane classification, Neurocomputing, in press, 2008.

Routines

scale - Scale the dataset into zero mean and unit variance
scale2 - Scale both the training and test sets
alhloo - Leave-one-out cross-validation (LOOCV) for ALH classifier
alhcv - Cross-validation for ALH classifier
alhval - Testing ALH classifier on a validation set
alhweight – Calculation of features weights for a given parameter T
alhclass - Classify a set of unlabelled input patterns given a trained ALH classifier

How to use ALH routines

(1) Data preprocessing phase. You can scale your data or you can skip a scaling. If you want to scale the data you proceed as follows:
Run scale.m to scale the whole dataset into zero mean and unit variance data,
or run scale2.m to scale the training and test sets separately.

(2) Parameters tuning phase:
Finding the best K, T and lambda by running either **alhloo** or **alhcv** when you have training dataset only,
or finding the best K, T and lambda by running **alhval** when you have both training and validation datasets.
Note that the intermediate results in alhloo, alhcv or alhval will be displayed, so the time comparison between ALH and other algorithms should not be used by these routines.

(3) Weighting calculation phase:
Run **alhweight** with the best T found in (2) to calculate features' weights.
(4) Testing phase:
Run **alhclass** to classify unlabelled test data with results from (2) and (3).

Details

scale **Scale the dataset into zero mean and unit variance**

Usage: Xs = scale(X)

scale2 **Scale two datasets simultaneously**

Usage: [X1, X2] = scale2(X1, X2)

X1 is scaled to zero mean and unit variance, and X2 is scaled according to the mean and variance of X1

alhloo **Leave-one-out cross-validation** (LOOCV) for ALH classifier

Usage: [] = alhloo(input, label, K, T, lambda)

Inputs:
input - data inputs (size: n*d)
label - data labels (size: n*1)
K - the candidate values for the number of nearest neighbors
T - the candidate values for the temperature used in feature weighting
lambda - the candidate values for the regularization parameter

Outputs:
If K, T and lambda are all scalars, only the corresponding LOOCV accuracy will be displayed;
Otherwise, the best K, T, lambda and the corresponding LOOCV accuracy will be displayed, and the plot for the LOOCV accuracy against K and T (with best lambda) will also be displayed.

Example:
load iris; X = scale(X);
alhloo(X,Y,4:2:10,[0.9 1:2],0)

alhcv — Cross-validation for ALH classifier

Usage: [] = alhcv(input, label, K, T, lambda, nfold, seed_value)

Inputs:

input - data inputs (size: n*d)

label - data labels (size: n*1)

K - the candidate values for the number of nearest neighbors

T - the candidate values for the temperature used in feature weighting

lambda - the candidate values for the regularization parameter

nfold - the fold size of cross-validation

seed_value - the seed value used for randomly shuffling the whole dataset before doing cross-validation. If seed_value == 0, then no shuffling will be used. (default: 2007)

Outputs:

If K, T and lambda are all scalars, only the corresponding CV accuracy will be displayed;

Otherwise, the best K, T, lambda and the corresponding CV accuracy will be displayed, and the plot for the CV accuracy against K and T (with best lambda) will also be displayed.

Example:

```
load iris; X = scale(X);
alhcv(X,Y,4:2:10,[0.9 1:2],0,10)
```

alhval — Testing ALH classifier on a validation set

Usage: [] = alhval(train_input, train_label, val_input, val_label, K, T, lambda)

Inputs:

train_input - the training inputs (size: n1*d)

train_label - the training labels (size: n1*1)

val_input - the validation inputs (size: n2*d)

val_label - the validation labels (size: n2*1)

K - the candidate values for the number of nearest neighbors

T - the candidate values for the temperature used in feature weighting

lambda - the candidate values for the regularization parameter

Outputs:

If K, T and lambda are all scalars, only the corresponding validation set accuracy will be displayed;

Otherwise, the best K, T, lambda and the corresponding validation set accuracy will be displayed, and the plot for the validation set accuracy against K and T (with best lambda) will also be displayed.

Example:

```
[X1, X2] = scale2(X1, X2);
alhval(X1,Y1,X2,Y2,4:2:10,1:3,0.1)
```

alhweight **Calculation of features weights for a given parameter T**

Finding the feature weights by the ratio of between-group to within-group sum of squares

Usage: weight = alhweight(input, label, T)

Inputs:

- input - data inputs (size: n*d)
- label - data labels (size: n*1)
- T - the temperature used in feature weighting (T >= 0)

Output:

- weight - the resulting feature weights (size: d*1)

alhclass **Classify a set of unlabelled input patterns given a trained ALH classifier**

Usage: test_label = alhclass(test_input, train_input, train_label, weight, K, lambda)

Inputs:

- test_input - the test inputs (size: m*d)
- train_input - the training inputs (size: n*d)
- train_label - the training labels (size: n*1)
- weight - the feature weights computed by alhweight routine (size: d*1)
- K - the number of nearest neighbors
- lambda - the regularization parameter

Output:

- test_label - the test labels predicted by ALH classifier (size: m*1)