--------------------------------------------

--- MATLAB/OCTAVE interface of LIBLINEAR ---

--------------------------------------------

Table of Contents

=================

- Introduction

- Installation

- Usage

- Returned Model Structure

- Other Utilities

- Examples

- Additional Information

Introduction

============

This tool provides a simple interface to LIBLINEAR, a library for

large-scale regularized linear classification and regression

(http://www.csie.ntu.edu.tw/~cjlin/liblinear). It is very easy to use

as the usage and the way of specifying parameters are the same as that

of LIBLINEAR.

Installation

============

On Windows systems, pre-built binary files are already in the

directory '..\windows', so no need to conduct installation. Now we

provide binary files only for 64bit MATLAB on Windows. If you would

like to re-build the package, please rely on the following steps.

We recommend using make.m on both MATLAB and OCTAVE. Just type 'make'

to build 'libsvmread.mex', 'libsvmwrite.mex', 'train.mex', and

'predict.mex'.

On MATLAB or Octave:

>> make

If make.m does not work on MATLAB (especially for Windows), try 'mex

-setup' to choose a suitable compiler for mex. Make sure your compiler

is accessible and workable. Then type 'make' to start the

installation.

Example:

matlab>> mex -setup

(ps: MATLAB will show the following messages to setup default compiler.)

Please choose your compiler for building external interface (MEX) files:

Would you like mex to locate installed compilers [y]/n? y

Select a compiler:

[1] Microsoft Visual C/C++ version 7.1 in C:\Program Files\Microsoft Visual Studio

[0] None

Compiler: 1

Please verify your choices:

Compiler: Microsoft Visual C/C++ 7.1

Location: C:\Program Files\Microsoft Visual Studio

Are these correct?([y]/n): y

matlab>> make

On Unix systems, if neither make.m nor 'mex -setup' works, please use

Makefile and type 'make' in a command window. Note that we assume

your MATLAB is installed in '/usr/local/matlab'. If not, please change

MATLABDIR in Makefile.

Example:

linux> make

To use octave, type 'make octave':

Example:

linux> make octave

For a list of supported/compatible compilers for MATLAB, please check

the following page:

http://www.mathworks.com/support/compilers/current\_release/

Usage

=====

matlab> model = train(training\_label\_vector, training\_instance\_matrix [,'liblinear\_options', 'col']);

-training\_label\_vector:

An m by 1 vector of training labels. (type must be double)

-training\_instance\_matrix:

An m by n matrix of m training instances with n features.

It must be a sparse matrix. (type must be double)

-liblinear\_options:

A string of training options in the same format as that of LIBLINEAR.

-col:

if 'col' is set, each column of training\_instance\_matrix is a data instance. Otherwise each row is a data instance.

matlab> [predicted\_label, accuracy, decision\_values/prob\_estimates] = predict(testing\_label\_vector, testing\_instance\_matrix, model [, 'liblinear\_options', 'col']);

-testing\_label\_vector:

An m by 1 vector of prediction labels. If labels of test

data are unknown, simply use any random values. (type must be double)

-testing\_instance\_matrix:

An m by n matrix of m testing instances with n features.

It must be a sparse matrix. (type must be double)

-model:

The output of train.

-liblinear\_options:

A string of testing options in the same format as that of LIBLINEAR.

-col:

if 'col' is set, each column of testing\_instance\_matrix is a data instance. Otherwise each row is a data instance.

Returned Model Structure

========================

The 'train' function returns a model which can be used for future

prediction. It is a structure and is organized as [Parameters, nr\_class,

nr\_feature, bias, Label, w]:

-Parameters: Parameters

-nr\_class: number of classes; = 2 for regression

-nr\_feature: number of features in training data (without including the bias term)

-bias: If >= 0, we assume one additional feature is added to the end

of each data instance.

-Label: label of each class; empty for regression

-w: a nr\_w-by-n matrix for the weights, where n is nr\_feature

or nr\_feature+1 depending on the existence of the bias term.

nr\_w is 1 if nr\_class=2 and -s is not 4 (i.e., not

multi-class svm by Crammer and Singer). It is

nr\_class otherwise.

If the '-v' option is specified, cross validation is conducted and the

returned model is just a scalar: cross-validation accuracy for

classification and mean-squared error for regression.

Result of Prediction

====================

The function 'predict' has three outputs. The first one,

predicted\_label, is a vector of predicted labels. The second output,

accuracy, is a vector including accuracy (for classification), mean

squared error, and squared correlation coefficient (for regression).

The third is a matrix containing decision values or probability

estimates (if '-b 1' is specified). If k is the number of classes

and k' is the number of classifiers (k'=1 if k=2, otherwise k'=k), for decision values,

each row includes results of k' binary linear classifiers. For probabilities,

each row contains k values indicating the probability that the testing instance is in

each class. Note that the order of classes here is the same as 'Label'

field in the model structure.

Other Utilities

===============

A matlab function libsvmread reads files in LIBSVM format:

[label\_vector, instance\_matrix] = libsvmread('data.txt');

Two outputs are labels and instances, which can then be used as inputs

of svmtrain or svmpredict.

A matlab function libsvmwrite writes Matlab matrix to a file in LIBSVM format:

libsvmwrite('data.txt', label\_vector, instance\_matrix]

The instance\_matrix must be a sparse matrix. (type must be double)

For windows, `libsvmread.mexw64' and `libsvmwrite.mexw64' are ready in

the directory `..\windows'.

These codes are prepared by Rong-En Fan and Kai-Wei Chang from National

Taiwan University.

Examples

========

Train and test on the provided data heart\_scale:

matlab> [heart\_scale\_label, heart\_scale\_inst] = libsvmread('../heart\_scale');

matlab> model = train(heart\_scale\_label, heart\_scale\_inst, '-c 1');

matlab> [predict\_label, accuracy, dec\_values] = predict(heart\_scale\_label, heart\_scale\_inst, model); % test the training data

Note that for testing, you can put anything in the testing\_label\_vector.

For probability estimates, you need '-b 1' only in the testing phase:

matlab> [predict\_label, accuracy, prob\_estimates] = predict(heart\_scale\_label, heart\_scale\_inst, model, '-b 1');

Additional Information

======================

Please cite LIBLINEAR as follows

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin.

LIBLINEAR: A Library for Large Linear Classification, Journal of

Machine Learning Research 9(2008), 1871-1874.Software available at

http://www.csie.ntu.edu.tw/~cjlin/liblinear

For any question, please contact Chih-Jen Lin <cjlin@csie.ntu.edu.tw>.