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

--- MATLAB/OCTAVE interface of LIBSVM ---

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

Table of Contents

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

- Introduction

- Installation

- Usage

- Returned Model Structure

- Examples

- Other Utilities

- Additional Information

Introduction

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

This tool provides a simple interface to LIBSVM, a library for support vector

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

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

Installation

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

On Unix systems, we recommend using GNU g++ as your

compiler and type 'make' to build 'svmtrain.mexglx' and 'svmpredict.mexglx'.

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

**Windows:**

On Windows systems, pre-built 'svmtrain.mexw32' and 'svmpredict.mexw32' are

included in this package, so no need to conduct installation. If you

have modified the sources and would like to re-build the package, type

'mex -setup' in MATLAB to choose a compiler for mex first. Then type

'make' to start the installation.

Starting from MATLAB 7.1 (R14SP3), the default MEX file extension is changed

from .dll to .mexw32 or .mexw64 (depends on 32-bit or 64-bit Windows). If your

MATLAB is older than 7.1, you have to build these files yourself.

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

Under 64-bit Windows, Visual Studio 2005 user will need "X64 Compiler and Tools".

The package won't be installed by default, but you can find it in customized

installation options.

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

following page:

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

Usage

=====

matlab> model = svmtrain(training\_label\_vector, training\_instance\_matrix [, 'libsvm\_options']);

-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 can be dense or sparse (type must be double).

-libsvm\_options:

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

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

-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 can be dense or sparse. (type must be double)

-model:

The output of svmtrain.

-libsvm\_options:

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

Returned Model Structure

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

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

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

totalSV, rho, Label, ProbA, ProbB, nSV, sv\_coef, SVs]:

-Parameters: parameters

-nr\_class: number of classes; = 2 for regression/one-class svm

-totalSV: total #SV

-rho: -b of the decision function(s) wx+b

-Label: label of each class; empty for regression/one-class SVM

-ProbA: pairwise probability information; empty if -b 0 or in one-class SVM

-ProbB: pairwise probability information; empty if -b 0 or in one-class SVM

-nSV: number of SVs for each class; empty for regression/one-class SVM

-sv\_coef: coefficients for SVs in decision functions

-SVs: support vectors

If you do not use the option '-b 1', ProbA and ProbB are empty

matrices. 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.

More details about this model can be found in LIBSVM FAQ

(http://www.csie.ntu.edu.tw/~cjlin/libsvm/faq.html) and LIBSVM

implementation document

(http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf).

Result of Prediction

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

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

predictd\_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,

for decision values, each row includes results of predicting

k(k-1/2) binary-class SVMs. 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.

Examples

========

Train and test on the provided data heart\_scale:

matlab> load heart\_scale.mat

matlab> model = svmtrain(heart\_scale\_label, heart\_scale\_inst, '-c 1 -g 0.07');

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

For probability estimates, you need '-b 1' for training and testing:

matlab> load heart\_scale.mat

matlab> model = svmtrain(heart\_scale\_label, heart\_scale\_inst, '-c 1 -g 0.07 -b 1');

matlab> load heart\_scale.mat

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

To use precomputed kernel, you must include sample serial number as

the first column of the training and testing data (assume your kernel

matrix is K, # of instances is n):

matlab> K1 = [(1:n)', K]; % include sample serial number as first column

matlab> model = svmtrain(label\_vector, K1, '-t 4');

matlab> [predict\_label, accuracy, dec\_values] = svmpredict(label\_vector, K1, model); % test the training data

Take linear kernel for example, the following precomputed kernel example

gives exactly same training error as LIBSVM built-in linear kernel

matlab> load heart\_scale.mat

matlab> n = size(heart\_scale\_inst,1);

matlab> K = heart\_scale\_inst\*heart\_scale\_inst';

matlab> K1 = [(1:n)', K];

matlab> model = svmtrain(heart\_scale\_label, K1, '-t 4');

matlab> [predict\_label, accuracy, dec\_values] = svmpredict(heart\_scale\_label, K1, model);

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

details of precomputed kernels, please read the section ``Precomputed

Kernels'' in the README of the LIBSVM package.

Other Utilities

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

A matlab function read\_sparse reads files in LIBSVM format:

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

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

of svmtrain or svmpredict. This code is derived from svm-train.c in

LIBSVM by Rong-En Fan from National Taiwan University.

Additional Information

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

This interface was initially written by Jun-Cheng Chen, Kuan-Jen Peng,

Chih-Yuan Yang and Chih-Huai Cheng from Department of Computer

Science, National Taiwan University. The current version was prepared

by Rong-En Fan and Ting-Fan Wu. If you find this tool useful, please

cite LIBSVM as follows

Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for

support vector machines, 2001. Software available at

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

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

or check the FAQ page:

http://www.csie.ntu.edu.tw/~cjlin/libsvm/faq.html#/Q9:\_MATLAB\_interface