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

Table of Contents
=================

Introduction
============

This tool provides a simple interface to LIBLINEAR, a library for large-scale regularized linear classification (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 Unix systems, we recommend using GNU g++ as your compiler and type 'make' to build 'train.mexglx' and 'predict.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
```

On Windows systems, pre-built 'train.mexw64' and 'predict.mexw64' are included in this package (in ..\windows), so no need to conduct installation unless you run 32 bit windows. 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.

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++ 2005 in C:\Program Files\Microsoft Visual Studio 8
[0] None
Compiler: 1
Please verify your choices:
Compiler: Microsoft Visual C/C++ 2005
Location: C:\Program Files\Microsoft Visual Studio 8
```

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

        matlab> make

For 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
        -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
        -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.

Result of Prediction
====================

The function 'predict' has three outputs. The first one,
predicted_label, is a vector of predicted labels.
The second output is a scalar meaning accuracy.
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' for training and testing:

```
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>.