Journal of Machine Learning Research 9 (2008) 1871-1874 Submitted 5/08; Published 8/08

LIBLINEAR: A Library for Large Linear Classification

Rong-En Fan B90098@CSIE.NTU.EDU.TW
Kai-Wei Chang B92084@CSIE.NTU.EDU.TW
Cho-Jui Hsieh B92085@CSIE.NTU.EDU.TW
Xiang-Rui Wang R95073@CSIE.NTU.EDU.TW
Chih-Jen Lin CJLIN@CSIE.NTU.EDU.TW
Department of Computer Science
National Taiwan University
Taipei 106, Taiwan

Editor: Soeren Sonnenburg

Abstract

LIBLINEAR is an open source library for large-scale linear classification. It supports logistic regression and linear support vector machines. We provide easy-to-use command-line tools and library calls for users and developers. Comprehensive documents are available for both beginners and advanced users. Experiments demonstrate that LIBLINEAR is very efficient on large sparse data sets.

Keywords: large-scale linear classification, logistic regression, support vector machines, open source, machine learning

1. Introduction

Solving large-scale classification problems is crucial in many applications such as text classification. Linear classification has become one of the most promising learning techniques for large sparse data with a huge number of instances and features. We develop LIBLINEAR as an easy-to-use tool to deal with such data. It supports L2-regularized logistic regression (LR), L2-loss and L1-loss linear support vector machines (SVMs) (Boser et al., 1992). It inherits many features of the popular SVM library LIBSVM (Chang and Lin, 2001) such as simple usage, rich documentation, and open source license (the BSD license[1]). LIBLINEAR is very efficient for training large-scale problems. For example, it takes only several *seconds* to train a text classification problem from the Reuters Corpus Volume 1 (rcv1) that has more than 600,000 examples. For the same task, a general SVM solver such as LIBSVM would take several hours. Moreover, LIBLINEAR is competitive with or even faster than state of the art linear classifiers such as Pegasos (Shalev-Shwartz et al., 2007) and SVM$^{\text{perf}}$ (Joachims, 2006). The software is available at `http://www.csie.ntu.edu.tw/~cjlin/liblinear`.

This article is organized as follows. In Sections 2 and 3, we discuss the design and implementation of LIBLINEAR. We show the performance comparisons in Section 4. Closing remarks are in Section 5.

1. The New BSD license approved by the Open Source Initiative.

2. Large Linear Classification (Binary and Multi-class)

LIBLINEAR supports two popular binary linear classifiers: LR and linear SVM. Given a set of instance-label pairs $(\mathbf{x}_i, y_i), i = 1, \ldots, l,\ \mathbf{x}_i \in R^n,\ y_i \in \{-1, +1\}$, both methods solve the following unconstrained optimization problem with different loss functions $\xi(\mathbf{w}; \mathbf{x}_i, y_i)$:

$$\min_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum\nolimits_{i=1}^{l} \xi(\mathbf{w}; \mathbf{x}_i, y_i), \tag{1}$$

where $C > 0$ is a penalty parameter. For SVM, the two common loss functions are $\max(1 - y_i\mathbf{w}^T\mathbf{x}_i, 0)$ and $\max(1 - y_i\mathbf{w}^T\mathbf{x}_i, 0)^2$. The former is referred to as L1-SVM, while the latter is L2-SVM. For LR, the loss function is $\log(1 + e^{-y_i\mathbf{w}^T\mathbf{x}_i})$, which is derived from a probabilistic model. In some cases, the discriminant function of the classifier includes a bias term, b. LIBLINEAR handles this term by augmenting the vector $\mathbf{w}$ and each instance $\mathbf{x}_i$ with an additional dimension: $\mathbf{w}^T \leftarrow [\mathbf{w}^T, b], \mathbf{x}_i^T \leftarrow [\mathbf{x}_i^T, B]$, where B is a constant specified by the user. The approach for L1-SVM and L2-SVM is a coordinate descent method (Hsieh et al., 2008). For LR and also L2-SVM, LIBLINEAR implements a trust region Newton method (Lin et al., 2008). The Appendix of our SVM guide.[2] discusses when to use which method. In the testing phase, we predict a data point $\mathbf{x}$ as positive if $\mathbf{w}^T\mathbf{x} > 0$, and negative otherwise. For multi-class problems, we implement the one-vs-the-rest strategy and a method by Crammer and Singer. Details are in Keerthi et al. (2008).

3. The Software Package

The LIBLINEAR package includes a library and command-line tools for the learning task. The design is highly inspired by the LIBSVM package. They share similar usage as well as application program interfaces (APIs), so users/developers can easily use both packages. However, their models after training are quite different (in particular, LIBLINEAR stores $\mathbf{w}$ in the model, but LIBSVM does not.). Because of such differences, we decide not to combine these two packages together. In this section, we show various aspects of LIBLINEAR.

3.1 Practical Usage

To illustrate the training and testing procedure, we take the data set news20,[3] which has more than one million features. We use the default classifier L2-SVM.

```
$ train news20.binary.tr
[output skipped]
$ predict news20.binary.t news20.binary.tr.model prediction
Accuracy = 96.575% (3863/4000)
```

The whole procedure (training and testing) takes less than 15 seconds on a modern computer. The training time without including disk I/O is less than one second. Beyond this simple way of running LIBLINEAR, several parameters are available for advanced use. For example, one may specify a parameter to obtain probability outputs for logistic regression. Details can be found in the `README` file.

2. The guide can be found at `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.
3. This is the news20.binary set from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets`. We use a 80/20 split for training and testing.

3.2 Documentation

The LIBLINEAR package comes with plenty of documentation. The README file describes the installation process, command-line usage, and the library calls. Users can read the "Quick Start" section, and begin within a few minutes. For developers who use LIBLINEAR in their software, the API document is in the "Library Usage" section. All the interface functions and related data structures are explained in detail. Programs `train.c` and `predict.c` are good examples of using LIBLINEAR APIs. If the README file does not give the information users want, they can check the online FAQ page.[4] In addition to software documentation, theoretical properties of the algorithms and comparisons to other methods are in Lin et al. (2008) and Hsieh et al. (2008). The authors are also willing to answer any further questions.

3.3 Design

The main design principle is to keep the whole package as simple as possible while making the source codes easy to read and maintain. Files in LIBLINEAR can be separated into source files, pre-built binaries, documentation, and language bindings. All source codes follow the C/C++ standard, and there is no dependency on external libraries. Therefore, LIBLINEAR can run on almost every platform. We provide a simple `Makefile` to compile the package from source codes. For Windows users, we include pre-built binaries.

Library calls are implemented in the file `linear.cpp`. The `train()` function trains a classifier on the given data and the `predict()` function predicts a given instance. To handle multi-class problems via the one-vs-the-rest strategy, `train()` conducts several binary classifications, each of which is by calling the `train_one()` function. `train_one()` then invokes the solver of users' choice. Implementations follow the algorithm descriptions in Lin et al. (2008) and Hsieh et al. (2008). As LIBLINEAR is written in a modular way, a new solver can be easily plugged in. This makes LIBLINEAR not only a machine learning tool but also an experimental platform.

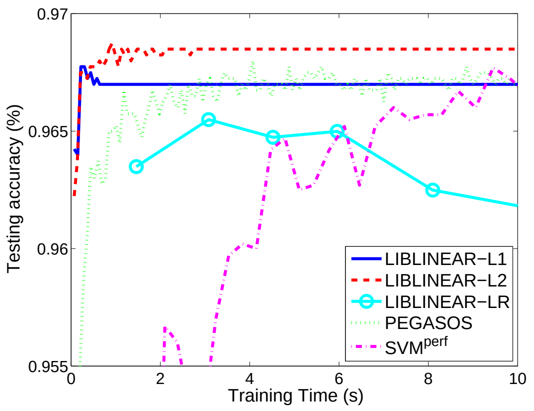
Making extensions of LIBLINEAR to languages other than C/C++ is easy. Following the same setting of the LIBSVM MATLAB/Octave interface, we have a MATLAB/Octave extension available within the package. Many tools designed for LIBSVM can be reused with small modifications. Some examples are the parameter selection tool and the data format checking tool.

4. Comparison

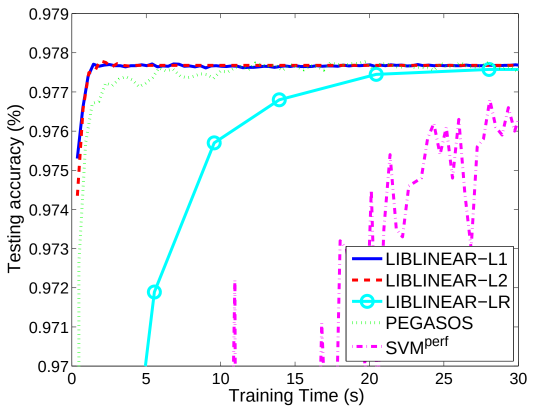
Due to space limitation, we skip here the full details, which are in Lin et al. (2008) and Hsieh et al. (2008). We only demonstrate that LIBLINEAR quickly reaches the testing accuracy corresponding to the optimal solution of (1). We conduct five-fold cross validation to select the best parameter C for each learning method (L1-SVM, L2-SVM, LR); then we train on the whole training set and predict the testing set. Figure 1 shows the comparison between LIBLINEAR and two state of the art L1-SVM solvers: Pegasos (Shalev-Shwartz et al., 2007) and SVM$^{\text{perf}}$ (Joachims, 2006). Clearly, LIBLINEAR is efficient.

To make the comparison reproducible, codes used for experiments in Lin et al. (2008) and Hsieh et al. (2008) are available at the LIBLINEAR web page.

4. FAQ can be found at `http://www.csie.ntu.edu.tw/~cjlin/liblinear/FAQ.html`.



(a) news20, l: 19,996, n: 1,355,191, #nz: 9,097,916



(b) rcv1, l: 677,399, n: 47,236, #nz: 156,436,656

Figure 1: Testing accuracy versus training time (in seconds). Data statistics are listed after the data set name. l: number of instances, n: number of features, #nz: number of nonzero feature values. We split each set to 4/5 training and 1/5 testing.

5. Conclusions

LIBLINEAR is a simple and easy-to-use open source package for large linear classification. Experiments and analysis in Lin et al. (2008), Hsieh et al. (2008) and Keerthi et al. (2008) conclude that solvers in LIBLINEAR perform well in practice and have good theoretical properties. LIBLINEAR is still being improved by new research results and suggestions from users. The ultimate goal is to make easy learning with huge data possible.

References

B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *COLT*, 1992.

C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *ICML*, 2008.

T. Joachims. Training linear SVMs in linear time. In *ACM KDD*, 2006.

S. S. Keerthi, S. Sundararajan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin. A sequential dual method for large scale multi-class linear SVMs. In *ACM KDD*, 2008.

C.-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region Newton method for large-scale logistic regression. *JMLR*, 9:627–650, 2008.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: primal estimated sub-gradient solver for SVM. In *ICML*, 2007.