A Practical Guide to LIBLINEAR

Appendix N.

In this section, we present a practical guide for LIBLINEAR users. For instructions of using LIBLINEAR and additional information, see the `README` file included in the package and the LIBLINEAR FAQ, respectively.

N.1 When to Use Linear (e.g., LIBLINEAR) Rather Then Nonlinear (e.g., LIBSVM)?

Please see the discussion in Appendix C of Hsu et al. (2003).

N.2 Data Preparation (In Particular, Document Data)

LIBLINEAR can be applied to general problems, but it is particularly useful for document data. We discuss how to transform documents to the input format of LIBLINEAR.

A data instance of a classification problem is often represented in the following form.

label feature 1, feature 2, ..., feature n

The most popular method to represent a document is the "bag-of-words" model (Harris, 1954), which considers each word as a feature. Assume a document contains the following two sentences

This is a simple sentence.
This is another sentence.

We can transfer this document to the following Euclidean vector:

a	an	$\cdots$	another	$\cdots$	is	$\cdots$	this	$\cdots$
1	0	$\cdots$	1	$\cdots$	2	$\cdots$	2	$\cdots$

The feature "is" has a value 2 because "is" appears twice in the document. This type of data often has the following two properties.

1. The number of features is large.
2. Each instance is sparse because most feature values are zero.

Let us consider a 20-class data set news20 at libsvmtools.[5] The first instance is

```
1 197:2 321:3 399:1 561:1 575:1 587:1 917:1 1563:1 1628:3 1893:1 3246:1 4754:2
6053:1 6820:1 6959:1 7571:1 7854:2 8407:1 11271:1 12886:1 13580:1 13588:1
13601:2 13916:1 14413:1 14641:1 15950:1 20506:1 20507:1
```

Each (index:value) pair indicates a word's "term frequency" (TF). In practice, additional steps such as removing stop words or stemming (i.e., using only root words) may be applied, although we do not discuss details here.

Existing methods to generate document feature vectors include

5. `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html#news20`

1. TF: term frequency.
2. TF-IDF (term frequency, inverse document frequency); see Salton and Yang (1973) and any information retrieval book.
3. binary: each feature indicates if a word appears in the document or not.

For example, the binary-feature vector of the news20 instance is

```
1 197:1 321:3 399:1 561:1 575:1 587:1 917:1 1563:1 1628:3 1893:1 3246:1 4754:1
6053:1 6820:1 6959:1 7571:1 7854:1 8407:1 11271:1 12886:1 13580:1 13588:1
13601:1 13916:1 14413:1 14641:1 15950:1 20506:1 20507:1
```

Our past experiences indicate that binary and TF-IDF are generally useful if normalization has been properly applied (see the next section).

N.3 Normalization

In classification, large values in data may cause the following problems.

1. Features in larger numeric ranges may dominate those in smaller ranges.
2. Optimization methods for training may take longer time.

The typical remedy is to scale data feature-wisely. This is suggested in the popular SVM guide by Hsu et al. (2003).[6] However, for document data, often a simple instance-wise normalization is enough. Each instance becomes a unit vector; see the following normalized news20 instance.

```
1 197:0.185695 321:0.185695 399:0.185695 561:0.185695 575:0.185695 587:0.185695
917:0.185695 1563:0.185695 1628:0.185695 1893:0.185695 3246:0.185695 4754:0.185695
6053:0.185695 6820:0.185695 6959:0.185695 7571:0.185695 7854:0.185695 8407:0.185695
11271:0.185695 12886:0.185695 13580:0.185695 13588:0.185695 13601:0.185695
13916:0.185695 14413:0.185695 14641:0.185695 15950:0.185695 20506:0.185695 20507:0.185695
```

There are 29 non-zero features, so each one has the value $1/\sqrt{29}$. We run LIBLINEAR to find five-fold cross-validation (CV) accuracy on the news20 data.[7] For the original data using TF features, we have

```
$ time ./train -v 5 news20
Cross Validation Accuracy = 80.4644%
57.527s
```

During the computation, the following warning message appears many times to indicate certain difficulties.

```
WARNING: reaching max number of iterations
```

6. For sparse data, scaling each feature to $[0, 1]$ is more suitable than $[-1, 1]$ because the latter makes the problem has many non-zero elements.
7. Experiments in this guide were run on a 64-bit machine with Intel Xeon 2.40GHz CPU (E5530), 8MB cache, and 72GB main memory.

Table 1: Current solvers in LIBLINEAR for L2-regularized problems.

	Dual-based solvers (coordinate descent methods)	Primal-based solvers (Newton-type methods)
Property	Extremely fast in some situations, but may be very slow in some others	Moderately fast in most situations
When to use it	1. Large sparse data (e.g., documents) under suitable scaling and C is not too large 2. Data with # instances $\ll$ # features	Others

If data is transformed to binary and then normalized, we have

```
$ time ./train -v 5 news20.scale
Cross Validation Accuracy = 84.7254%
6.140s
```

Clearly, the running time for the normalized data is much shorter and no warning message appears.

Another example is in Lewis et al. (2004), which normalizes each log-transformed TF-IDF feature vector to have unit length.

N.4 Selection of Solvers

Appendix A lists many LIBLINEAR's solvers. Users may wonder how to choose them. Fortunately, these solvers usually give similar performances. Some in fact generate the same model because they differ in only solving the primal or dual optimization problems. For example, `-s 1` solves dual L2-regularized L2-loss SVM, while `-s 2` solves primal.

```
$ ./train -s 1 -e 0.0001 news20.scale
(skipped)
Objective value = -576.922572
nSV = 3540
$./train -s 2 -e 0.0001 news20.scale
(skipped)
iter  7 act 1.489e-02 pre 1.489e-02 delta 1.133e+01 f 5.769e+02 |g| 3.798e-01 CG  10
```

Each run trains 20 two-class models, so we only show objective values of the last one. Clearly, the dual objective value `-576.922572` coincides with the primal value `5.769e+02`. We use the option `-e` to impose a smaller stopping tolerance, so optimization problems are solved more accurately.

While LIBLINEAR's solvers give similar performances, their training time may be different. For current solvers for L2-regularized problems, a rough guideline is in Table 1. We recommend users

1. Try the default dual-based solver first.

2. If it is slow, check primal-based solvers.

For L1-regularized problems, the choice is limited because currently we only offer primal-based solvers.

To choose between using L1 and L2 regularization, we recommend trying L2 first unless users need a sparse model. In most cases, L1 regularization does not give higher accuracy but may be slightly slower in training; see a comparison in Section D of the supplemenary materials of Yuan et al. (2010).

N.5 Parameter Selection

For linear classification, the only parameter is C in Eq. (1). We summarize some properties below.

1. Solvers in LIBLINEAR is *not very sensitive* to C. Once C is larger than certain value, the obtained models have similar performances. A theoretical proof of this kind of results is in Theorem 3 of Keerthi and Lin (2003).

2. Using a larger C value usually causes longer training time. Users should avoid trying such values.

We conduct the following experiments to illustrate these properties. To begin, we show in Figure 2 the relationship between C and CV accuracy. Clearly, CV rates are similar after C is large enough. For running time, we compare the results using $C = 1$ and $C = 100$.

```
$ time ./train -c 1 news20.scale
2.528s
$ time ./train -c 100 news20.scale
28.589s
```

We can see that a larger C leads to longer training time. Here a dual-based coordinate descent method is used. For primal-based solvers using Newton methods, the running time is usually less affected by the change of C.

```
$ time ./train -c 1 -s 2 news20.scale
8.596s
$ time ./train -c 100 -s 2 news20.scale
11.088s
```

While users can try a few C values by themselves, LIBLINEAR (after version 2.0) provides an easy solution to find a suitable parameter. If primal-based classification solvers are used (`-s 0` or `-s 2`), the `-C` option efficiently conducts cross validation several times and finds the best parameter automatically.

```
$ ./train -C -s 2 news20.scale
log2c= -15.00   rate=69.0806
log2c= -14.00   rate=69.3505
```

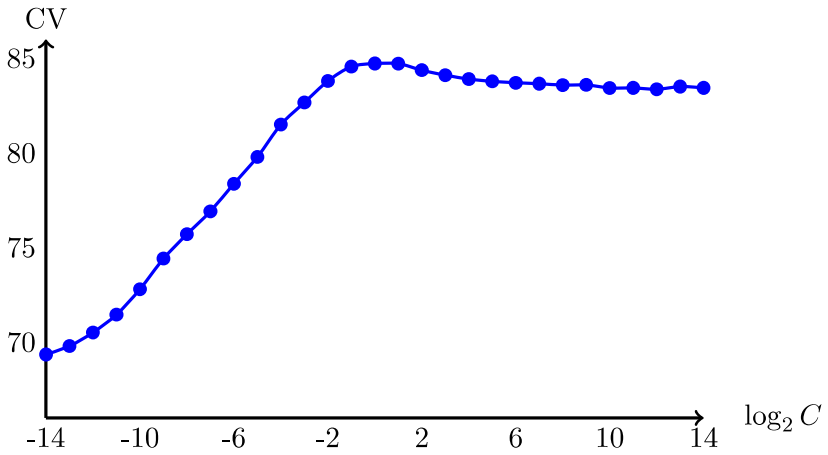

Figure 2: CV accuracy versus $\log_2 C$.

```
(skipped)
log2c=   9.00   rate=83.7151
log2c=  10.00   rate=83.6837
warning: maximum C reached.
Best C = 1.000000  CV accuracy = 84.7192%
```

Users do not need to specify the range of C to explore because LIBLINEAR finds a reasonable range automatically. Note that users can still use the `-c` option to specify the smallest C value of the search range. For example,

```
$ ./train -C -s 2 -c 0.5 -e 0.0001 news20.scale
log2c=  -1.00   rate=84.5623
log2c=   0.00   rate=84.7254
(skipped)
log2c=   9.00   rate=83.6272
log2c=  10.00   rate=83.621
warning: maximum C reached.
Best C = 1.000000  CV accuracy = 84.7254%
```

This option is useful when users want to rerun the parameter selection procedure from a specified C under a different setting, such as a stricter stopping tolerance `-e 0.0001` in the above example.

Note that after finding the best C, users must apply the same solver to train a model for future prediction. Switching from a primal-based solver to a corresponding dual-based solver (e.g., from `-s 2` to `-s 1`) is fine because they produce the same model.

Some solvers such as `-s 5` or `-s 6` are not covered by the `-C` option.[8] We can use a parameter selection tool `grid.py` in LIBSVM to find the best C value. For example,

```
$ ./grid.py -log2c -14,14,1 -log2g null -svmtrain ./train -s 5 news20.scale
```

8. Note that we have explained that for some dual-based solvers you can use `-C` on their corresponding primal-based solvers for parameter selection.

checks the CV rates of $C \in \{2^{-14}, 2^{-13}, \ldots, 2^{14}\}$. Note that `grid.py` should be used only if `-C` is not available for the desired solver. The `-C` option is much faster than `grid.py` on a single computer.

References

Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. Coordinate descent method for large-scale L2-loss linear SVM. *Journal of Machine Learning Research*, 9:1369–1398, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/cdl2.pdf`.

Bo-Yu Chu, Chia-Hua Ho, Cheng-Hao Tsai, Chieh-Yen Lin, and Chih-Jen Lin. Warm start for parameter selection of linear classifiers. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2015.

Koby Crammer and Yoram Singer. On the learnability and design of output codes for multiclass problems. In *Computational Learning Theory*, pages 35–46, 2000.

Jerome H. Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

Zellig S. Harris. Distributional structure. *Word*, 10:146–162, 1954.

Chia-Hua Ho and Chih-Jen Lin. Large-scale linear support vector regression. *Journal of Machine Learning Research*, 13:3323–3348, 2012. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/linear-svr.pdf`.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML)*, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/cddual.pdf`.

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2003. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.

Tzu-Kuo Huang, Ruby C. Weng, and Chih-Jen Lin. Generalized Bradley-Terry models and multi-class probability estimates. *Journal of Machine Learning Research*, 7:85–115, 2006. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/generalBT.pdf`.

S. Sathiya Keerthi and Chih-Jen Lin. Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation*, 15(7):1667–1689, 2003.

S. Sathiya Keerthi, Sellamanickam Sundararajan, Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. A sequential dual method for large scale multi-class linear SVMs. In *Proceedings of the Forteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 408–416, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/sdm_kdd.pdf`.

Ching-Pei Lee and Chih-Jen Lin. A study on L2-loss (squared hinge-loss) multi-class SVM. *Neural Computation*, 25(5):1302–1323, 2013. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/l2mcsvm/l2mcsvm.pdf`.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5: 361–397, 2004.

Chih-Jen Lin, Ruby C. Weng, and S. Sathiya Keerthi. Trust region Newton method for large-scale logistic regression. *Journal of Machine Learning Research*, 9:627–650, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/logistic.pdf`.

Gerard Salton and Chung-Shu Yang. On the specification of term values in automatic indexing. *Journal of Documentation*, 29:351–372, 1973.

Paul Tseng and Sangwoon Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117:387–423, 2009.

Hsiang-Fu Yu, Fang-Lan Huang, and Chih-Jen Lin. Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning*, 85(1-2):41–75, October 2011. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/maxent_dual.pdf`.

Guo-Xun Yuan, Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *Journal of Machine Learning Research*, 11:3183–3234, 2010. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/l1.pdf`.

Guo-Xun Yuan, Chia-Hua Ho, and Chih-Jen Lin. An improved GLMNET for l1-regularized logistic regression. *Journal of Machine Learning Research*, 13:1999–2030, 2012. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/l1_glmnet/long-glmnet.pdf`.