Development of Open-Source Machine Learning Packages

Chih-Jen Lin

National Taiwan Univ.

MBZUAI

April 2023

Outline

1. Why open-source machine learning software?
2. Some lessons learned from developing machine learning packages
3. Discussion and conclusions

Outline

Introduction

- My past research has been on machine learning algorithms and software
- My group developed several open-source packages in the past
- Important ones include
  - LIBSVM: 2000–now
    A library for support vector machines
  - LIBLINEAR: 2007–now
    A library for large linear classification
- They were reasonably successful

Introduction (Cont'd)

- LIBSVM: probably the most widely used SVM package

  Its official paper (Chang and Lin, 2011) has been cited more than 53,000 times (Google Scholar, 2/2023)
- In the hay day of SVM (i.e., before deep learning), this is the software everybody used
- For LIBLINEAR, it's popularly used in Internet companies

  The official paper (Fan et al., 2008) has been cited more than 9,400 times (Google Scholar, 2/2023)

Introduction (Cont'd)

- In the rest of this talk I will share some lessons and thoughts on developing ML software
- The above-mentioned packages are early ML developments
- The landscape has dramatically changed due to new things: deep learning, popularity of Python, Github development environment, etc
- However, many principles remain the same
- The discussion is biased towards small-scale software done in a group. However, most results discussed here still apply to large industry-scale projects (e.g., PyTorch)

Software versus Experiment Code

- Many researchers now release experiment code used for their papers

  Reason: experiments can be reproduced
- This is important, but experiment code is different from software
- Experiment code often includes messy scripts for various settings in the paper – useful for reviewers
- Software: for general users

  One or a few reasonable settings with a suitable interface are enough

Software versus Experiment Code (Cont'd)

- Reproducibility different from replicability (Drummond, 2009)

  Replicability: make sure things work on the sets used in the paper

  Reproducibility: ensure that things work in general
- To illustrate that the purpose of a software package is to serve users, I will share my own experiences by discussing how some my past packages got started

How LIBSVM Got Started?

- My Ph.D. study was in numerical optimization instead of machine learning
- After joining a CS department, I found students were not interested in optimization theory
- I happened to find some machine learning papers that solve optimization problems
- So I thought maybe we can redo some experiments

How LIBSVM Got Started? (Cont'd)

- While redoing experiments in some published works, surprisingly my students and I had difficulties to replicate some results
- This doesn't mean that these researchers gave fake results
- The reason is that as experts in that area, they may not clearly say some subtle steps
- But of course I didn't know because I was new to the area

How LIBSVM Got Started? (Cont'd)

- For example, assume you have the following data

height	gender
180	1
150	0

  The first feature is in a large range, so some normalization or scaling is needed
- After realizing that others may face similar problems, we felt that software including these subtle steps should be useful

How LIBSVM Got Started? (Cont'd)

- Lesson: doing something useful to the community should always be our goal as a researcher
- Nowadays we are constantly under the pressure to publish papers or get grants, but we should remember that as a researcher, our real job is to solve problems
- I will further illustrate this point by explaining how we started LIBLINEAR

From LIBSVM to LIBLINEAR

- In 2006, LIBSVM was popularly used by researchers and engineers
- In that year I went to Yahoo! Research for a 6-month visit
- Engineers there told me that SVM couldn't be applied for their web documents due to lengthy training time
- To explain why that's the case, let's write down the SVM formulation

From LIBSVM to LIBLINEAR (Cont'd)

- Given training data $(y_i, \boldsymbol{x}_i), i = 1, \ldots, l$, $\boldsymbol{x}_i \in R^n, y_i = \pm 1$
- Standard SVM (Boser et al., 1992) solves

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\underbrace{\sum_{i=1}^{l} \max(1 - y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b), 0)}_{\text{sum of losses}}$$

- Loss term: we hope

$$y_i \text{ and } \boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b \text{ have the same sign}$$

From LIBSVM to LIBLINEAR

- $\boldsymbol{x}$ is mapped to a higher (maybe infinite) dimensional space for better separability

$$\boldsymbol{x} \rightarrow \phi(\boldsymbol{x})$$

- However, the high dimensionality causes difficulties
- People use kernel trick (Cortes and Vapnik, 1995) so that

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) \equiv \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

  can be easily calculated
- All operations then rely on kernels (details omitted)
- Unfortunately, the training/prediction cost is still very high

From LIBSVM to LIBLINEAR (Cont'd)

- At Yahoo! I found that these document sets have a large number of features
- The reason is that they use the bag-of-words model

  $\Rightarrow$ every English word corresponds to a feature
- After some experiments I realized that if each instance already has many features

  $\Rightarrow$ then a further mapping may not improve the performance much
- Without mappings, we have linear classification:

$$\phi(\boldsymbol{x}) = \boldsymbol{x}$$

From LIBSVM to LIBLINEAR (Cont'd)

Comparison between linear and kernel

	Linear		RBF Kernel	
Data set	Time	Accuracy	Time	Accuracy
MNIST38	0.1	96.82	38.1	99.70
ijcnn1	1.6	91.81	26.8	98.69
covtype	1.4	76.37	46,695.8	96.11
news20	1.1	96.95	383.2	96.90
real-sim	0.3	97.44	938.3	97.82
yahoo-japan	3.1	92.63	20,955.2	93.31
webspam	25.7	93.35	15,681.8	99.26

Size reasonably large: e.g., yahoo-japan: 140k instances and 830k features

From LIBSVM to LIBLINEAR (Cont'd)

Comparison between linear and kernel

	Linear		RBF Kernel	
Data set	Time	Accuracy	Time	Accuracy
MNIST38	0.1	96.82	38.1	99.70
ijcnn1	1.6	91.81	26.8	98.69
covtype	1.4	76.37	46,695.8	96.11
news20	1.1	96.95	383.2	96.90
real-sim	0.3	97.44	938.3	97.82
yahoo-japan	3.1	92.63	20,955.2	93.31
webspam	25.7	93.35	15,681.8	99.26

Size reasonably large: e.g., yahoo-japan: 140k instances and 830k features

From LIBSVM to LIBLINEAR (Cont'd)

Comparison between linear and kernel

	Linear		RBF Kernel	
Data set	Time	Accuracy	Time	Accuracy
MNIST38	0.1	96.82	38.1	99.70
ijcnn1	1.6	91.81	26.8	98.69
covtype	1.4	76.37	46,695.8	96.11
news20	1.1	96.95	383.2	96.90
real-sim	0.3	97.44	938.3	97.82
yahoo-japan	3.1	92.63	20,955.2	93.31
webspam	25.7	93.35	15,681.8	99.26

Size reasonably large: e.g., yahoo-japan: 140k instances and 830k features

From LIBSVM to LIBLINEAR (Cont'd)

- For linear rather than kernel, we are able to develop much more efficient optimization algorithms
- After the Yahoo! visit, I spent the next 10 years on this research topic (large-scale linear classification)
- This is a good example that I identify the industry needs and bring directions back to school for deep studies

Outline

Most Users aren't ML Experts

For most users, what they hope is

- Prepare training and test sets
- Run a package and get good results

What we have seen over the years is that

- Users expect good results right after using a method
- If method A doesn't work, they switch to B. Even worse, they may not know that results are unsatisfactory
- They may inappropriately use most methods they tried

Most Users aren't ML Experts (Cont'd)

For example, consider how people are using BERT (Devlin et al., 2019) for document classification

- They expect good results after directly running it with a fixed number of epochs
- Some may not know that BERT takes up to only 512 tokens of each document
- Thus if the document is long, only limited information is used for training

Shouldn't the software have some mechanisms to address this issue?

Most Users aren't ML Experts (Cont'd)

From our experiences, ML packages should provide some simple and automatic/semi-automatic settings for users

These settings may not be the best, but easily give users some reasonable results

I will illustrate this point by a procedure we developed for SVM

Easy and Automatic Procedure

- Let's consider a practical example from astroparticle physics

```
1   2.61e+01   5.88e+01   -1.89e-01   1.25e+02
1   5.70e+01   2.21e+02   8.60e-02    1.22e+02
1   1.72e+01   1.73e+02   -1.29e-01   1.25e+02
...
0   2.39e+01   3.89e+01   4.70e-01    1.25e+02
0   2.23e+01   2.26e+01   2.11e-01    1.01e+02
0   1.64e+01   3.92e+01   -9.91e-02   3.24e+01
```

- Training set: 3,089 instances

  Test set: 4,000 instances

Easy and Automatic Procedure (Cont'd)

The story behind this data set

- User:

  I am using libsvm in a astroparticle physics application .. First, let me congratulate you to a really easy to use and nice package. Unfortunately, it gives me astonishingly bad results...
- OK. Please send us your data
- I am able to get 97% test accuracy. Is that good enough for you ?
- User:

  You earned a copy of my PhD thesis

Easy and Automatic Procedure (Cont'd)

- For this data set, direct training and testing yields

  66.925% test accuracy

- But training accuracy close to 100%
- Overfitting occurs because some features are in large numeric ranges (details not explained here)

Easy and Automatic Procedure (Cont'd)

- A simple solution is to scale each feature to $[0, 1]$

$$\frac{\text{feature value} - \min}{\max - \min}$$

- For this problem, after scaling, test accuracy is increased to 96.15%
- Scaling is a simple and useful step; but many users didn't know it

Easy and Automatic Procedure (Cont'd)

- For SVM and other machine learning methods, users must decide certain parameters
- If SVM with Gaussian (RBF) kernel is used,

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}$$

  then γ (kernel parameter) and C (regularization parameter) must be decided
- Sometimes we need to properly select parameters

  $\Rightarrow$ but users may not be aware of this step

Easy and Automatic Procedure (Cont'd)

After helping many users, we came up with a simple procedure

1. Conduct simple scaling on the data
2. Consider RBF kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}$
3. Use cross-validation to find the best parameter C and γ
4. Use the best C and γ to train the whole training set
5. Test

Easy and Automatic Procedure (Cont'd)

- We proposed this procedure in an "SVM guide" (Hsu et al., 2003) and implemented it in LIBSVM
- This procedure has been tremendously useful.

  Now almost the standard thing to do for SVM beginners
- The guide (never published) was cited about 8,800 times (Google Scholar, 2/2023)
- Lesson: an easy and automatic setting is very important for users

Users are Our Teachers

- While doing software is sometimes not considered as research, users help to point out many useful directions
- Example: LIBSVM supported only two-class classification in the beginning
- In standard SVM,

$$\text{label } y = +1 \text{ or } -1,$$

but what if we would like to do digit $(0, \ldots, 9)$ recognition?

Users are Our Teachers (Cont'd)

- When we started LIBSVM, I was quite new to ML. I didn't know that many real-world problems are multi-class
- From many users' requests, we realize the importance of multi-class classification
- LIBSVM is among the first SVM software to handle multi-class data
- We finished a study on multi-class SVM (Hsu and Lin, 2002) that eventually becomes very highly cited

  Around 10,000 citations on Google Scholar (2/2023)

Users are Our Teachers (Cont'd)

- Another example is probability outputs
- SVM does not directly give

$$P(y = 1|\boldsymbol{x}) \text{ and } P(y = -1|\boldsymbol{x}) = 1 - P(y = 1|\boldsymbol{x})$$

- A popular approach to generate SVM probability outputs is by Platt (2000)
- But it's for two-class situations only

Users are Our Teachers (Cont'd)

- Many users asked about multi-class probability outputs
- Thus we did a study in Wu et al. (2004), which has also been highly cited

  Around 2,400 citations on Google Scholar (2/2023)
- Lesson: users help to identify what are useful and what are not

One or Many Options

- Sometimes we received the following requests

  1. Besides the multi-class approach supported, could you include other approaches such as ...?

  2. Could you extend LIBSVM to support other kernels such as χ^2 kernel?

- Two extremes in designing a package

  1. One option: reasonably good for most cases

  2. Many options: users try options to get best results

One or Many Options (Cont'd)

- From a research viewpoint, we should include everything, so users can play with them
- But

$$\text{more options} \Rightarrow \text{more powerful}$$
$$\Rightarrow \text{more complicated}$$

- Some users have no abilities to choose between options
- For LIBSVM, we took the "one option" approach but made it easily extensible
- Which way to take depends on the purpose of the software and the target users

Simplicity versus Better Performance

- We must consider the tradeoff between simplicity and better performance/quality
- Example: earlier in LIBSVM, our cross validation (CV) procedure was not stratified

  - Results less stable because data of each class not evenly distributed to folds

  - We now support stratified CV, but code becomes more complicated
- In general, we avoid changes for just marginal improvements

Simplicity versus Better Performance (Cont'd)

- From an earlier Google research blog "Lessons learned developing a practical large scale machine learning system" by Simon Tong:

  "It is perhaps less academically interesting to design an algorithm that is slightly worse in accuracy, but that has greater ease of use and system reliability. However, in our experience, it is very valuable in practice"
- That is, a complicated method with a slightly higher accuracy may not be useful in practice

Techniques from Other Areas

- ML requires techniques from other areas (e.g., optimization, numerical analysis, etc.)
- For example, deep learning cannot be successful without fast matrix-matrix multiplications and automatic differentiation
- Making best use of them help to improve the quality of the software

ML and Other Areas (Cont'd)

- Example: we carefully handle numerical computation in the package development

  In LIBSVM's probability outputs, we must calculate

$$1 - p_i, \quad \text{where} \quad p_i \equiv \frac{1}{1 + \exp(\Delta)},$$

  where Δ is some value

- If Δ is small, $p_i \approx 1$. Then $1 - p_i$ is a catastrophic cancellation (Goldberg, 1991): when subtracting two nearby numbers, the relative error can be large so most digits are meaningless

Techniques from Other Areas (Cont'd)

- In a simple C++ program with double precision,

$$\Delta = -64 \quad \Rightarrow \quad 1 - \frac{1}{1 + \exp(\Delta)} \text{ returns zero}$$

but

$$\frac{\exp(\Delta)}{1 + \exp(\Delta)} \text{ gives more accurate result}$$

- Catastrophic cancellation may be resolved by reformulation
- Lesson: carefully applying techniques from other areas helps to improve the quality of the ML software

Issues of Documentation

- For a machine learning package, various types of documents are needed
- Document for API functions: explanation of parameters

  For example, AdamW is an API function in PyTorch. Its parameter such as `betas` must be explained
- Quickstart: simple examples to let users easily try something

Issues of Documentation (Cont'd)

- Tutorials: stories about completing a task

  Example: a story of handling a document classification task, including pre-processing, applying a classifier, etc
- Implementation document: technical details of the methods implemented

  Example: details of neural-network architectures or the mathematical formulation of evaluation metrics
- Carefully preparing and organizing documentation is very essential

Outline

Machine Learning in Industry

- In the early days of open-source ML, most developers are researchers at universities
- We propose algorithms as well as design software
- A difference now is that industry now heavily invests in machine learning
- ML packages become huge and complicated

Machine Learning in Industry (Cont'd)

- A concern is that the role of individual researchers and their algorithm development may become less important
- How university researchers cope with such a situation is something we need to think about

The Joy of Doing Software

- As a researcher, knowing that people are using your work is a nice experience
- I received emails like

  "It has been very useful for my research"

  "I am a big fan of your software LIBSVM."

  "I read a lot of your papers and use LIBSVM almost daily. (You have become some sort of super hero for me:)) ."
- Another thing I like in developing software is that students learned to be responsible for their work

Conclusions

- From my experience, developing machine learning software is very interesting and rewarding
- In particular I feel happy when people find my work useful
- I strongly believe that doing something useful to the community should always be our goal as a researcher

For those interested in developing open-source ML software, you are welcome to discuss with me after this short course

Acknowledgments

- All users have greatly helped us to make improvements
- I also thank all my past and current group members