

# Lessons from Developing Machine Learning Algorithms and Systems

Chih-Jen Lin  
Department of Computer Science  
National Taiwan University



Talk at TAAI 2017, December 2017

# Outline

- 1 Introduction
- 2 Some lessons learned from developing machine learning packages
- 3 Challenges in designing future machine learning systems
- 4 Conclusions



# Outline

- 1 Introduction
- 2 Some lessons learned from developing machine learning packages
- 3 Challenges in designing future machine learning systems
- 4 Conclusions



# Introduction

- My past research has been on **machine learning algorithms and software**
- In this talk, I will share our experiences in developing two packages
  - LIBSVM: 2000–now  
A library for support vector machines
  - LIBLINEAR: 2007–now  
A library for large linear classification
- To begin, let me shamelessly describe how these packages have been widely used



# Introduction (Cont'd)

- LIBSVM: now probably the most widely used SVM package  
Its implementation document (Chang and Lin, 2011), maintained since 2001 and published in 2011, has been cited more than **34,000** times (Google Scholar, 9/2017)
- Citeseer (a CS citation indexing system) showed that it's among the **10 most cited work in computer science history**



# Introduction (Cont'd)

- For LIBLINEAR, it's popularly used in Internet companies

The official paper (Fan et al., 2008) has been cited more than 5,400 times (Google Scholar, 9/2017)

- Citeseer shows it's **second highest cited** CS paper published in 2008



# Introduction (Cont'd)

- So these past efforts were reasonably successful
- In the rest of this talk I will humbly share some lessons and thoughts from developing these packages



# How I Got Started?

- My Ph.D. study was in numerical optimization, a small area not belonging to any big field
- 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 I Got Started? (Cont'd)

- While redoing experiments in some published works, surprisingly my students and I had **difficulties to reproduce** 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 I 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 I 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 has been 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  $(y_i, \mathbf{x}_i), i = 1, \dots, l$ ,  
 $\mathbf{x}_i \in R^n, y_i = \pm 1$
- Standard SVM (Boser et al., 1992) solves

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

- Loss term: we hope

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



# From LIBSVM to LIBLINEAR

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

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

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

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

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**  
⇒ every English word corresponds to a feature
- After some experiments I realized that if each instance already has so many features  
⇒ then a further mapping may not improve the performance much
- Without mappings, we have **linear classification**:

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



# From LIBSVM to LIBLINEAR (Cont'd)

Comparison between linear and kernel

Data set	Linear		RBF Kernel	
	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

Data set	Linear		RBF Kernel	
	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

Data set	Linear		RBF Kernel	
	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

- 1 Introduction
- 2 Some lessons learned from developing machine learning packages
- 3 Challenges in designing future machine learning systems
- 4 Conclusions



# Most Users aren't ML Experts

- Nowadays people talk about “democratizing AI” or “democratizing machine learning”
- The reasons that **not everybody is an ML expert**
- Our development of easy-to-use SVM procedures is one of the early attempts to democratize ML



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

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
- **They may inappropriately use most methods they tried**

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



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

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,

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

then  $\gamma$  (kernel parameter) and  $C$  (regularization parameter) must be decided

- Sometimes we need to properly select parameters  
⇒ 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  $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2}$
- 3 Use cross-validation to find the **best parameter**  $C$  and  $\gamma$
- 4 Use the best  $C$  and  $\gamma$  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 more than 5,600 times (Google Scholar, 9/17)
- 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, ..., 9) recognition?



# Users are Our Teachers (Cont'd)

- At that time I was new to ML. I didn't know how 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.

More than 7,000 citations on Google Scholar (9/17)





# Users are Our Teachers (Cont'd)

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

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

- 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 1,600 citations on Google Scholar (11/17)
- Lesson: **users help to identify what are useful and what are not.**



# ML and Other Areas

- ML requires techniques from other areas (e.g., optimization, numerical analysis, etc.)
- The interplay between different areas is an interesting issue.
- For example, if you studied hard on optimization techniques for method A but later method B is better  $\Rightarrow$  Efforts are wasted
- But for good machine learning packages, these things are essential
- We deeply **study optimization methods by taking machine learning properties into account**



# ML and Other Areas (Cont'd)

- We **carefully handle numerical computation** in the package
- Example: In LIBSVM's probability outputs, we need to calculate

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

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



# ML and 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**



# ML and Other Areas (Cont'd)

- Lesson: applying techniques of one area to another can sometimes lead to breakthroughs or better quality work
- But what if the ML method you target at turns out to be not useful?  
Everyday new ML methods are being proposed
- That's true. But we don't know beforehand. We simply need to think deeply and take risks. Further, failure is an option



# Outline

- 1 Introduction
- 2 Some lessons learned from developing machine learning packages
- 3 Challenges in designing future machine learning systems
- 4 Conclusions



# 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**

- Constant maintenance is essential for software
- One thing I like in developing software is that students learned to be **responsible for their work**



# Software versus Experiment Code (Cont'd)

- However, we are researchers rather than software engineers
- How to balance between these maintenance works and the creative research is a challenge
- The community now lacks incentives for researchers to work on high quality software



# 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 invest on machine learning**
- Interface (API, GUI, and others) becomes much more important



# Machine Learning in Industry (Cont'd)

- The whole ML process may be by drawing a flowchart. Then things are run on the cloud.
- It makes machine learning **easier for non-experts**
- However, ML packages become huge and complicated
- They can only be either company projects (e.g., Google TensorFlow, Amazon MXNet, Microsoft Cognitive Toolkit, etc.) or community projects (e.g., scikit-learn, Spark MLlib, etc.)



# 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 cop with such a situation is something we need to think about



# Outline

- 1 Introduction
- 2 Some lessons learned from developing machine learning packages
- 3 Challenges in designing future machine learning systems
- 4 Conclusions**



# 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:)) .”



# 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





# Acknowledgments

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

