Experiences and Lessons in Developing Machine Learning Software

Chih-Jen Lin
Department of Computer Science
National Taiwan University

Talk at CIKM, November 4, 2014

Machine Learning Software

- Generating useful machine learning software for practical industry use is difficult and challenging
- In this talk, I will share our experiences in developing LIBSVM and LIBLINEAR.
- LIBSVM (Chang and Lin, 2011):

  Probably the most popular SVM package; cited more than 19,000 times on Google Scholar
- LIBLINEAR (Fan et al., 2008):

  A library for large linear classification; widely used in Internet companies (e.g., Google, Yahoo!, eBay)

Outline

1. How users apply machine learning methods
2. An example: support vector machines
3. Considerations in developing machine learning software
4. Discussion and conclusions

Outline

Most Users aren't Machine Learning Experts

- In developing LIBSVM, we found that many users have zero machine learning knowledge
- It is unbelievable that many asked what the difference between training and testing is

Most Users aren't Machine Learning Experts (Cont'd)

- A sample mail

```
From:
To: cjlin@csie.ntu.edu.tw
Subject: Doubt regarding SVM

Dear Sir,
        sir what is the difference between
testing data and training data?
```

- Sometimes we cannot do much for such users

Most Users aren't Machine Learning Experts (Cont'd)

- Fortunately, more people have taken machine learning courses

  Also, companies hire people with machine learning knowledge
- However, these engineers are still not machine learning experts

How Users Apply Machine Learning Methods?

For most users, what they hope is

- Prepare training and testing 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

How Users Apply Machine Learning Methods? (Cont'd)

In my opinion

- Machine learning 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
- If such settings are not enough, users many need to consult with machine learning experts.

I will illustrate the first point by a procedure we developed for SVM

Outline

Support Vector Classification

- Training data $(y_i, \boldsymbol{x}_i), i = 1, \ldots, l, \boldsymbol{x}_i \in R^n, y_i = \pm 1$
- Most users know that SVM takes the following formulation (Boser et al., 1992; Cortes and Vapnik, 1995)

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

- $\phi(\boldsymbol{x})$: high dimensional. Kernel tricks are used:

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

Let's Try a Practical Example

- A problem from a user in 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

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

Direct Training and Testing

- 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)

Data Scaling

- For SVM, features shouldn't be in too large numeric ranges
- Also we need to avoid that some features dominate
- A simple solution is to scale each feature to $[0, 1]$

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

  There are other scaling methods
- 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

Parameter Selection

- For the earlier example, we use

$$C = 1, \quad \gamma = 1/4,$$

where γ is the parameter Gaussian (RBF) kernel

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

- Sometimes we need to properly select parameters
- For another set from a user

Direct training and test

$$\text{Test accuracy} = 2.44\%$$

After proper data scaling

$$\text{Test accuracy} = 12.20\%$$

Parameter Selection (Cont'd)

- Use parameter from cross validation on a grid of (C, γ) values

$$\text{Test accuracy} = 87.80\%$$

- For SVM and other machine learning methods, parameter selection is sometimes needed

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

A Simple Procedure for Beginners

After helping many users, we came up with the following procedure

1. Conduct simple scaling on the data
2. Consider RBF kernel $K(\boldsymbol{x}, \mathbf{y}) = e^{-\gamma\|\boldsymbol{x}-\mathbf{y}\|^2}$
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

A Simple Procedure for Beginners (Cont'd)

- We proposed this procedure in an "SVM guide" (Hsu et al., 2003) and implemented it in LIBSVM
- From research viewpoints, this procedure is not novel. We never thought about submiting our guide somewhere
- But this procedure has been tremendously useful.

  Now almost the standard thing to do for SVM beginners

Lessons in Designing a Machine Learning Algorithm

- The method doesn't need to be the best all the time
- A method that is
  1. reasonably good in accuracy,
  2. general enough to cover some types of data, and
  3. not very sensitive to parameters

  is often the most convenient for users
- Random Forests (Breiman, 2001) is such an example

Outline

Which Functions to be Included?

- The answer is simple: listen to users
- While we criticize users' lack of machine learning knowledge, they point out many useful directions
- Example: LIBSVM supported only binary classification in the beginning. From many users' requests, we knew the importance of multi-class classification
- There are many possible approaches for multi-class SVM. Assume k classes

Which Function to be Included? (Cont'd)

- - One-versus-the rest: Train k binary SVMs:

$$\begin{array}{lll} \text{1st class} & \text{vs.} & (2, \cdots, k)\text{th class} \\ \text{2nd class} & \text{vs.} & (1, 3, \ldots, k)\text{th class} \\ & \vdots & \end{array}$$

  - One-versus-one: train $k(k-1)/2$ binary SVMs

  $(1,2), (1,3), \ldots, (1,k), (2,3), (2,4), \ldots, (k-1,k)$

- We finished a study in Hsu and Lin (2002), which is now highly cited.
- Currently LIBSVM supports one-vs-one approach

Which Function to be Added? (Cont'd)

- LIBSVM is among the first SVM software to handle multi-class data.

  This helps to attract many users.
- Users help to identify what are useful and what are not.

One or Many Options

- Sometimes we received the following requests

  1. In addition to "one-vs-one," could you include other multi-class approaches such as "one-vs-the rest?"

  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

One or Many Options (Cont'd)

- We are very careful in adding things to LIBSVM
- A new feature is included only if enough requests have been made
- We put specialized extensions in a different place "LIBSVM Tools"

Simplicity versus Better Performance

- This issue is related to "one or many options" discussed earlier
- Example: Before, our cross validation (CV) procedure is 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)

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

Numerical Stability

- Many classification methods (e.g., SVM, neural networks) involve numerical methods (e.g., solving an optimization problem)
- Numerical analysts have a high standard on their code, but machine learning people do not
- This situation is expected:
  If we carefully implement method A but later method B gives higher accuracy $\Rightarrow$ Efforts are wasted
- But for good machine learning packages, quality of numerical implementations is essential

Numerical Stability (Cont'd)

- 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 Δ 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.

Numerical Stability (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
- This example shows that some techniques can be applied to improve numerical stability

Legacy Issues

- The compatibility between earlier and later versions restricts developers to conduct certain changes.
- We can avoid legacy issues by some programming techniques
- Example: we chose "one-vs-one" as the multi-class strategy in LIBSVM.
- What if one day we would like to use a different multi-class method?

Legacy Issues (Cont'd)

- Earlier in LIBSVM, we did not make the trained model a public structure. We employed encapsulation in object-oriented programming
- User can call

```
model = svm_train(...);
```

  but cannot directly access a model's contents

```
int y1 = model.label[1];
```

- We provide functions to get model information

```
svm_get_nr_class(model);
svm_get_labels(model, ...);
```

- Then users are transparent to the internal change on multi-class methods

Outline

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

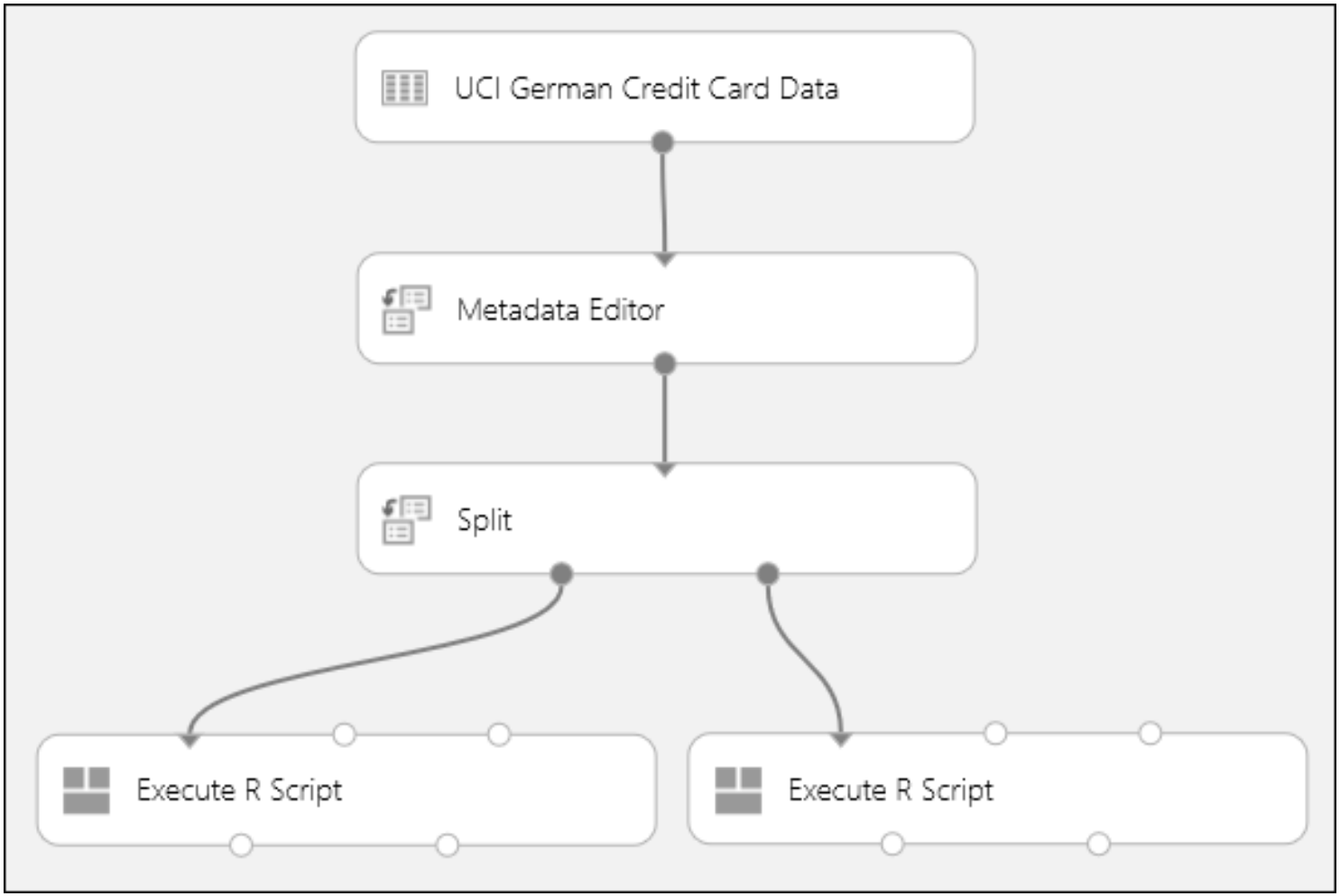
  Reproducibility: ensure that things work in general
- The community now lacks incentives for researchers to work on high quality software

Automatic Machine Learning

- I mentioned a kind of automatic procedure for SVM
- Others in machine learning community now consider this as an important direction
- Recently, at ICML 2014 there is an AutoML workshop
- People discuss the automatic selection of methods, hyper-parameter, features, and others

Machine Learning without Programming

From Microsoft Azure Machine Learning page:



Machine Learning without Programming (Cont'd)

- The whole process is by generating a flowchart. Things are run on the cloud
- Several companies are developing such cloud- and web-based machine learning service
- It makes machine learning easier for non-experts
- However, many design issues remain to be solved
- For example, we mentioned the importance of data scaling/normalization. Should it be default or not?

Conclusions

- From my experience, developing machine learning software is very interesting
- We should encourage people in the community to develop high quality machine learning software
- Making machine learning easy to use for non-expert is an important direction

Acknowledgments

- All users have greatly helped us to make improvements

  Without them we cannot get this far
- We also thank all our past group members