Some Thoughts on Machine Learning Software Design

Chih-Jen Lin

Department of Computer Science

National Taiwan University

Talk at University of Southampton, February 6, 2004

About This Talk

- Machine learning software design involves with interesting research issues
- Also other issues

  Implementation

  Users
- Would like to share my past experience on the software LIBSVM for discussion
- Many issues are controversial

- Focus on software of one method

  e.g. SVM software

- Integrated ML environments: even more complicated issues
  - A bit different from data mining software
  - Examples:

    Spider

    `http://www.kyb.tuebingen.mpg.de/bs/people/spider/main.html`

    PyML

    `http://cmgm.stanford.edu/~asab/pyml/pyml.html`

  So issues such as data types etc. will not be discussed

Good Machine Learning Software

- Must use good methods

  Has been the focus of machine learning research

Issues less discussed

- Should include tools needed by users

  e.g. a simple scaling code

- Should be simple and complete

  e.g. multi-class classification

- Should be numerically stable

  Efficiency may not be the only concern

Tools for Users: e.g. Scaling

- I started working on SVM in 1999

  Saw SVM papers presenting excellent accuracy

  Decided to try by myself

- Statlog data set (`http://www.liacc.up.pt/ML/statlog/`)

  heart data

```
70.0 1.0 4.0 130.0 322.0 0.0 2.0 109.0 0.0 2.4 2.0 3.0 3.0 2
67.0 0.0 3.0 115.0 564.0 0.0 2.0 160.0 0.0 1.6 2.0 0.0 7.0 1
57.0 1.0 2.0 124.0 261.0 0.0 0.0 141.0 0.0 0.3 1.0 0.0 7.0 2
64.0 1.0 4.0 128.0 263.0 0.0 0.0 105.0 1.0 0.2 2.0 1.0 7.0 1
```

  100% SVs

  Bad accuracy

- No idea what happened

  In few papers one simple sentence mentions "normalization" or "scaling" to [-1,1]

- Then I also realized

  SVM dual

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ \text{subject to} \quad & 0 \le \alpha_i \le C, i = 1, \ldots, l, \\ & y^T\alpha = 0, \end{aligned}$$

  RBF kernel

$$K(x_i, x_j) = \phi(x_i)^T\phi(x_j) = e^{-\gamma\|x_i - x_j\|^2}$$

- If $Q \to I$, and $C \geq 2l_1/l$.

$$\alpha_i \to \begin{cases} 2l_2/l & \text{if } y_i = 1, \\ 2l_1/l & \text{if } y_i = -1 \end{cases}$$

All are SVs

- Lesson:

  ML researchers know the importance of scaling

  Most users do not know

  Such simple tools should be provided

- David Meyer (author of R interface to LIBSVM) had exactly the same experience

  He decided to scale data by default

Simple and Complete:

e.g. Multi-class classification

- Many methods when proposed:

  Only two-class case considered

- OK for a paper

  Standard extension to multi-class

- But if no one implements it

  The proposed method can never be useful

- I did not realize this before

  LIBSVM released in April 2000: 2-class only

  By the summer: many requests for multi-class

Multi-class Implementation: One or Many

- So I was forced to implement it
- Many options:

  1 vs. the rest, 1 vs. 1 (pairwise), error correcting codes,

  All k-class together as one optimization formula
- Include one or many ?

Two Types of Numerical Software

1. Include all options and let users choose
2. Provide only one which is generally good

- The argument never ends

  Also depends on different situations

- For SVM software, I prefer the 2nd
  - Historical reason: I was from a numerical optimization group supporting the 2nd
  - A black box type implementation may be useful

    Many have no ability to choose from different options

- Need a serious comparison to find a "generally good" one
- Finally I chose 1 vs. 1 [Hsu and Lin, 2002]

  Similar accuracy to others

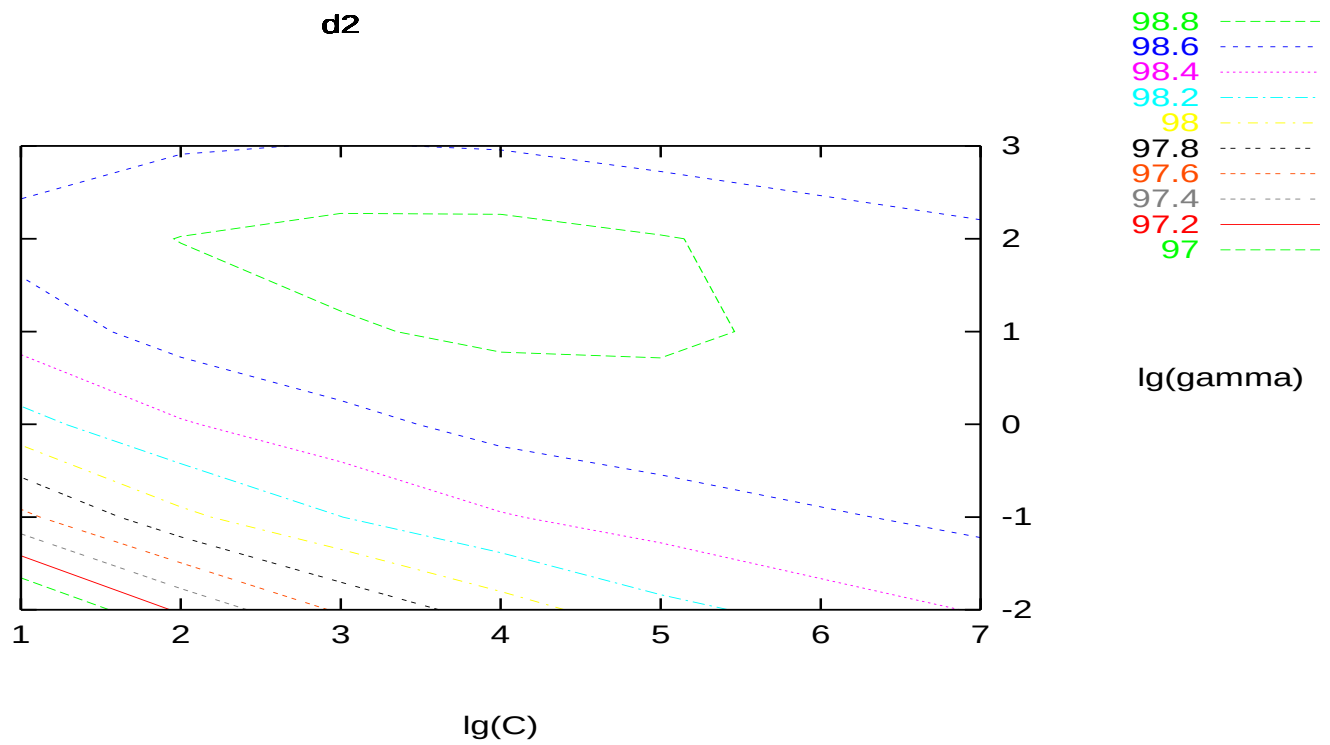
  Shortest training

  A bit longer on testing than 1 vs. the rest
- In scientific computing

  Numerical comparison: seriously conducted and considered part of the research
- We should emphasize more on such issues

More on Completeness: Parameter Selection

- SVM: a bit sensitive to parameters



- I spent a lot of time on loo bound

$$\text{leave-one-out error} \le f(C, \gamma)$$

so

$$\min_{C,\gamma} f(C, \gamma)$$

- Not stable, so for two parameters, now we recommend CV+grid search
- But, unlike 1vs1 for multi-class, this is still far from settled

- OK if no need for feature selection
- Feature selection considered

  $\Rightarrow$ # parameters may be > 2

  CV+grid not work

  Loo bound or Bayesian evidence more suitable ?
- In other words, we may have

	CV+grid	loo/Bayesian
parameter selection	$>$	
feature selection	$<$	

Comparing Two Methods

- If

	Method 1	Method 2
Two-class classification	so so	excellent
Multi-class	easy	complicated
Probability output	easy	easy
Parameter selection	easy	difficult
Feature selection	easy	difficult
regression	easy	easy

- Which should we use ?

- When comparing two methods

  All aspects should be considered

- SVM

  Not particularly good

  Each item: by several research papers

- Any method: one paper provides all and results reasonably good ?

Random Forest Is One

- 500 trees

  Each: full tree using m_{try} random features

- Prediction: by voting

- Multi-class: by tree
- Probability output: proportion of 500
- Parameter selection: m_{try} the only parameter

  Moreover, not sensitive
- Feature selection:

  Out-of-bag validation of each tree

  $\Rightarrow$ feature importance
- All these are discussed in Breiman's paper

- Performance: My experience and [Meyer et al. 2003]

  Competitive with (or only a bit worse than) SVM

- Though some said:

  Comparing random forest with SVM not fair

  $\Rightarrow$ random forest, random nearest neighbor, random SVM

- RF: simple and complete

- My goal for SVM: as simple and complete software

Numerical Stability

- Many classification methods (e.g., SVM, neural networks) solve optimization problems
- Part of their implementations:

  Essentially numerical software
- Numerical analysts: high standard on their code

  We do not
- Reasonable:

  Efforts on implementing method A

  One day method B: higher accuracy

  Efforts wasted
- Really a dilemma

Example: SMO and Linear Kernel

- Selecting working set $\{i, j\}$, solve

$$\begin{aligned}
\min_{\alpha_i,\alpha_j} \quad & \frac{1}{2}\begin{bmatrix}\alpha_i & \alpha_j\end{bmatrix}\begin{bmatrix}Q_{ii} & Q_{ij}\\ Q_{ji} & Q_{jj}\end{bmatrix}\begin{bmatrix}\alpha_i\\ \alpha_j\end{bmatrix} \\
& +(Q_{i,N}\alpha_N^k - 1)\alpha_i + (Q_{j,N}\alpha_N^k - 1)\alpha_j \\
\text{subject to} \quad & y_i\alpha_i + y_j\alpha_j = -y_N^T\alpha_N^k, \\
& 0 \le \alpha_i, \alpha_j \le C,
\end{aligned}$$

- If $y_i = y_j$, substituting $\alpha_i = -\alpha_j - \cdots$

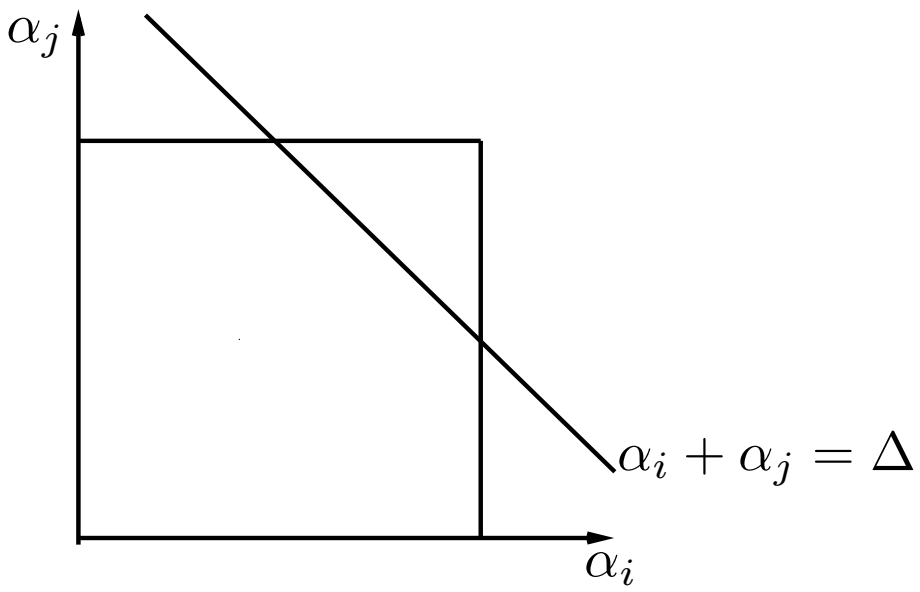
  One-variable minimization:

$$\alpha_j^{new} = \alpha_j + \frac{G_i - G_j}{Q_{ii} + Q_{jj} - 2Q_{ij}} \tag{1}$$

where

$$G_i \equiv (Q\alpha)_i - 1 \text{ and } G_j \equiv (Q\alpha)_j - 1.$$

Clipping it back to $[0, C]$



- Linear kernel: matrix may be PSD but not PD

  $Q_{ii} + Q_{jj} - 2Q_{ij} = 0$

  Division by zero

- Some may say

  Check if $Q_{ii} + Q_{jj} - 2Q_{ij} = 0$, if so,

  add a small threshold

- Remember floating point "==" not recommended in general

- Indeed, no need to worry about this

- As long as $-G_i - G_j \neq 0$, (1) goes to ∞ or $-\infty$,
  defined under IEEE 754/854 floating-point standard
- Comparing C and INF: valid IEEE operations
- Correctly clipped to 0 or C
- 0/0 not defined
- $-G_i - G_j > \epsilon$ always holds
  $\Rightarrow$ the stopping criteria
- 0/0 never happens

- What if $Q_{ii} + Q_{jj} - 2Q_{ij} < 0$ due to numerical error

  Or rounded to zero ?

- Under IEEE: +0, -0

  -0 causes wrong direction

- Use

$$\frac{G_i - G_j}{\max(0, Q_{ii} + Q_{jj} - 2Q_{ij})}$$

- Proper $\max(-0, 0)$ gives 0

  `java.lang.math`: max:

  If one argument is positive zero and the other negative zero, the result is positive zero.

- Goldberg, ACM Computing Surveys, 1991

  What every computer scientist should know about floating-point arithmetic

Example: SMO and tanh Kernel

- Whether it should be used or not is another issue

  Let's assume it is there

- Kernel matrix: non-PSD

  $Q_{ii} + Q_{jj} + 2Q_{ij} < 0$

- Objective value $\nearrow$ but not $\searrow$

  Not converge to a local minimum

- Infinite loop using LIBSVM

  At one point: I have to warn users this in LIBSVM FAQ

- Later we developed a simple strategy for all non-PSD kernels and proved convergence [Lin and Lin 2003]
- But someone said

$$\frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i = \frac{1}{2}\alpha^T Q\alpha + C\sum_{i=1}^{l} l((Q\alpha)_i - 1)$$

  Non-convex; change it to

$$\frac{1}{2}\alpha^T\alpha + C\sum_{i=1}^{l} l((Q\alpha)_i - 1)$$

- tanh still used, but convex

- Accuracy may be similar (sparsity another issue)
- He/she is right; but I cannot force users not to use SVM+tanh

  Such issues may still need to be investigated
- Different points of view :

  One is from designing methods

  One is from designing software

There are Many Such Issues

- For example

  How to check support vectors ? $\alpha_i > 0, < C$

  A place where floating point "==" may be used

- Not only numerical analysis techniques

  SVM: optimization issues

- Implementation of ML software

  Can be a quite interdisciplinary issue

Conclusions

- ML software: many interesting research issues

  Some are traditional ML considerations

  Some are not

- It is rewarding to see users benefit from such research efforts