Machine Learning Software: Design and Practical Use

Chih-Jen Lin

National Taiwan University

eBay Research Labs

Talk at Machine Learning Summer School, Santa Cruz, July 16, 2012

Machine Learning Software

- Most machine learning works focus on developing algorithms
- Researchers didn't pay much attention to software
- Recently, some think software is important. For example, "The need for open source software in machine learning" by Sonnenburg et al. (2007)
- One reasons is for replicating and evaluating research results
- However, a good software package is beyond that

Machine Learning Software (Cont'd)

- In this talk, I will share our experiences in developing LIBSVM and LIBLINEAR.
- LIBSVM (Chang and Lin, 2011):

  One of the most popular SVM packages; cited almost $10,000$ times on Google Scholar
- LIBLINEAR (Fan et al., 2008):

  A library for large linear classification; popular in Internet companies for document classification and NLP applications

Machine Learning Software (Cont'd)

- This talk will contain two parts:
- First, we discuss practical use of SVM as an example to see users' needs
- Second, we discuss design considerations for a good machine learning package.

  We didn't know or expect most of them in the beginning!
- The talk is biased toward SVM and logistic regression, but materials are useful for other machine learning methods.

Outline

Outline

Outline

Support Vector Classification

- Training data $(\mathbf{x}_i, y_i), i = 1, \ldots, l, \mathbf{x}_i \in R^n, y_i = \pm 1$
- Maximizing the margin (Boser et al., 1992; Cortes and Vapnik, 1995)

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

- High dimensional (maybe infinite) feature space

$$\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots).$$

- $\mathbf{w}$: maybe infinite variables

Support Vector Classification (Cont'd)

- The dual problem (finite # variables)

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

  where $Q_{ij} = y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ and $\mathbf{e} = [1, \ldots, 1]^T$

- At optimum

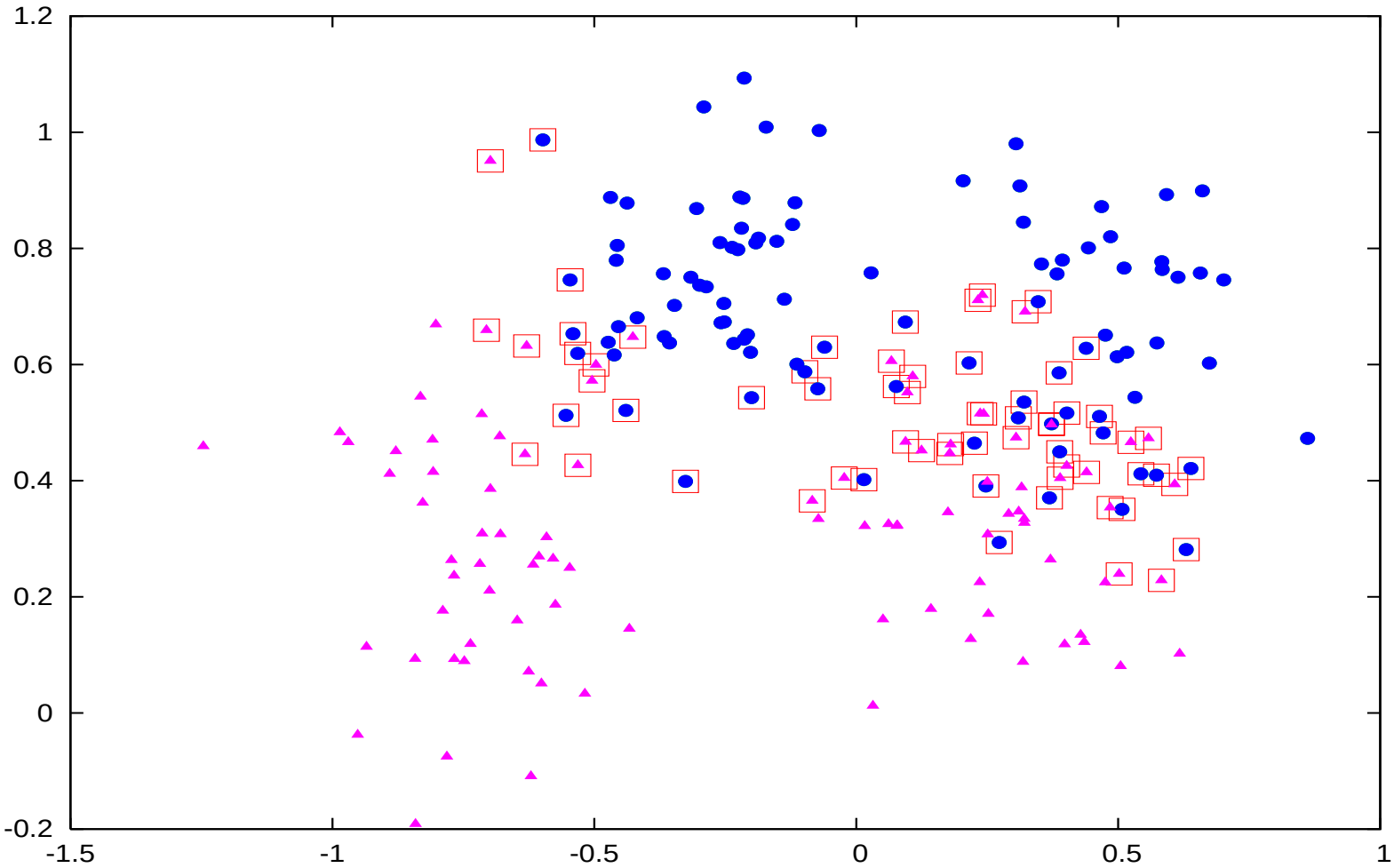
$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \phi(\mathbf{x}_i)$$

- Kernel: $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$; closed form

  Example: RBF kernel: $e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}$

Support Vector Classification (Cont'd)

Only $\mathbf{x}_i$ of $\alpha_i > 0$ used $\Rightarrow$ support vectors



Support Vector Classification (Cont'd)

- For most users, what they hope is
  1. Prepare training and testing sets
  2. Run a package and get good results
- But things are not that simple
- Let's start with a practical example from a user

Outline

Let's Try a Practical Example

A problem from astroparticle physics

```
1 1:2.61e+01 2:5.88e+01 3:-1.89e-01 4:1.25e+02
1 1:5.70e+01 2:2.21e+02 3:8.60e-02  4:1.22e+02
1 1:1.72e+01 2:1.73e+02 3:-1.29e-01 4:1.25e+02
...
0 1:2.39e+01 2:3.89e+01 3:4.70e-01 4:1.25e+02
0 1:2.23e+01 2:2.26e+01 3:2.11e-01 4:1.01e+02
0 1:1.64e+01 2:3.92e+01 3:-9.91e-02 4:3.24e+01
```

Training and testing sets available: 3,089 and 4,000
Sparse format: zero values not stored
Data available at LIBSVM Data Sets

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

The Story Behind this Data Set (Cont'd)

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

But isn't it machine learning people's responsibility to make their methods easily give reasonable results?

The Story Behind this Data Set (Cont'd)

- We will discuss how we eventually came up with a setting for beginners
- It can quickly give them some reasonable results

Training and Testing

Training

```
$./svm-train svmguide1
optimization finished, #iter = 6131
nSV = 3053, nBSV = 724
Total nSV = 3053
```

Testing

```
$./svm-predict svmguide1.t svmguide1.model out
Accuracy = 66.925% (2677/4000)
```

nSV and nBSV: number of SVs and bounded SVs ($\alpha_i = C$).

Why this Fails

- After training, nearly 100% support vectors
- Training and testing accuracy different

```
$./svm-predict svmguide1 svmguide1.model out
Accuracy = 99.7734% (3082/3089)
```

- Most kernel elements:

$$K_{ij} = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/4} \begin{cases} = 1 & \text{if } i = j, \\ \to 0 & \text{if } i \neq j. \end{cases}$$

- Some features in rather large ranges

Data Scaling

- Without scaling

  Attributes in greater numeric ranges may dominate
- Linearly scale each feature to $[0, 1]$ by:

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

  There are other scaling methods
- Scaling generally helps, but not always

Data Scaling: Same Factors

A common mistake

```
$./svm-scale -l -1 -u 1 svmguide1 > svmguide1.s
$./svm-scale -l -1 -u 1 svmguide1.t > svmguide1
```

`-l -1 -u 1`: scaling to $[-1, 1]$

Same factor on training and testing

```
$./svm-scale -s range1 svmguide1 > svmguide1.sc
$./svm-scale -r range1 svmguide1.t > svmguide1.
```

Later we will give a real example

After Data Scaling

Train scaled data and then predict

```
$./svm-train svmguide1.scale
$./svm-predict svmguide1.t.scale svmguide1.scale
  svmguide1.t.predict
Accuracy = 96.15%
```

Training accuracy is now similar

```
$./svm-predict svmguide1.scale svmguide1.scale.m
Accuracy = 96.439%
```

Default parameter: $C = 1, \gamma = 0.25$

Outline

Different Parameters

- If we use $C = 20, \gamma = 400$

```
$./svm-train -c 20 -g 400 svmguide1.scale
$./svm-predict svmguide1.scale svmguide1.sca
Accuracy = 100% (3089/3089)
```

- 100% training accuracy but

```
$./svm-predict svmguide1.t.scale svmguide1.s
Accuracy = 82.7% (3308/4000)
```

- Very bad test accuracy
- Overfitting happens

Overfitting

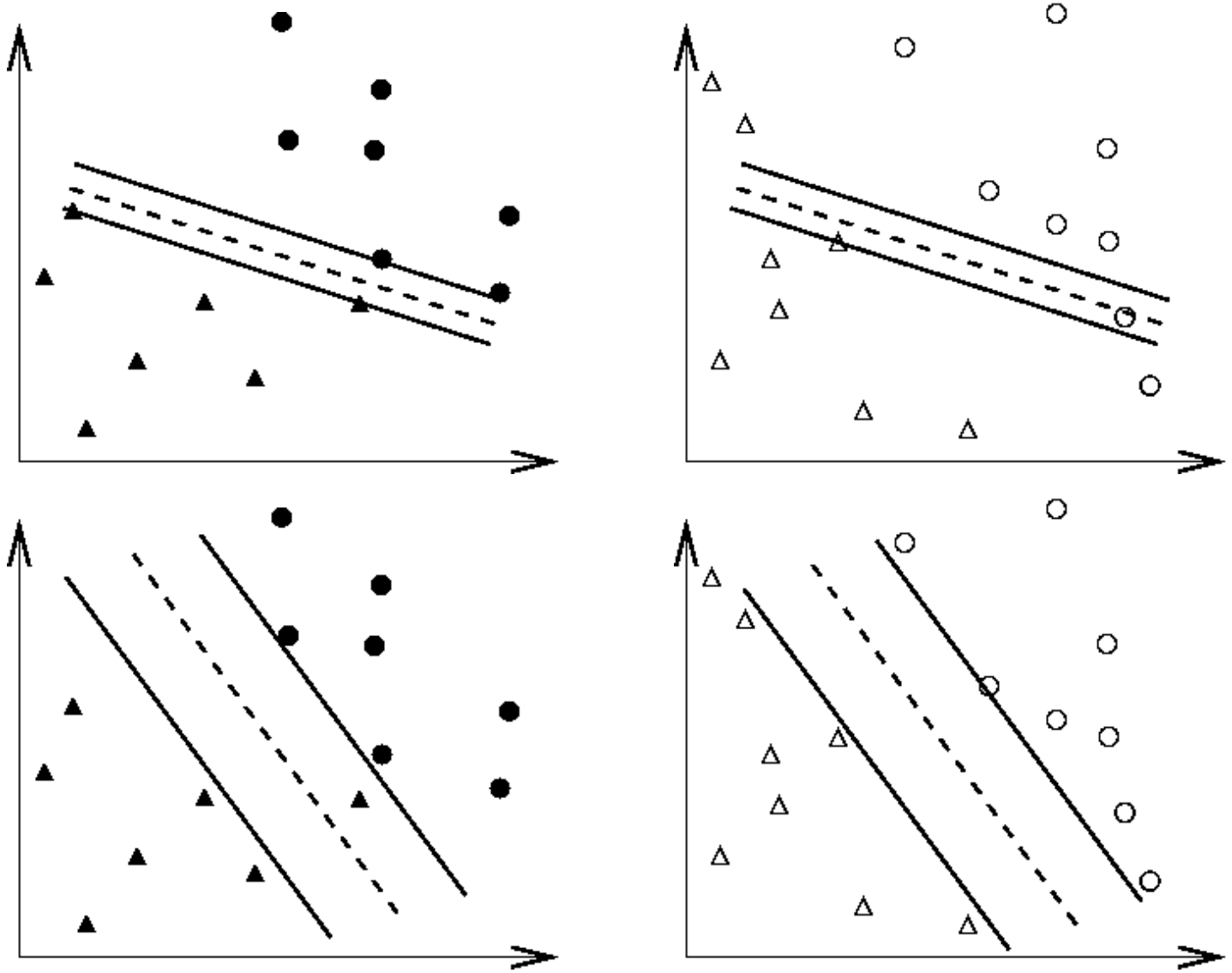
- In theory

  You can easily achieve 100% training accuracy
- This is useless
- When training and predicting a data, we should

  Avoid underfitting: small training error

  Avoid overfitting: small testing error

● and ▲: training; ○ and △: testing



Parameter Selection

- Need to select suitable parameters
- C and kernel parameters
- Example:

$$\gamma \text{ of } e^{-\gamma\|\mathbf{x}_i-\mathbf{x}_j\|^2}$$

$$a, b, d \text{ of } (\mathbf{x}_i^T\mathbf{x}_j/a + b)^d$$

- How to select them so performance is better?

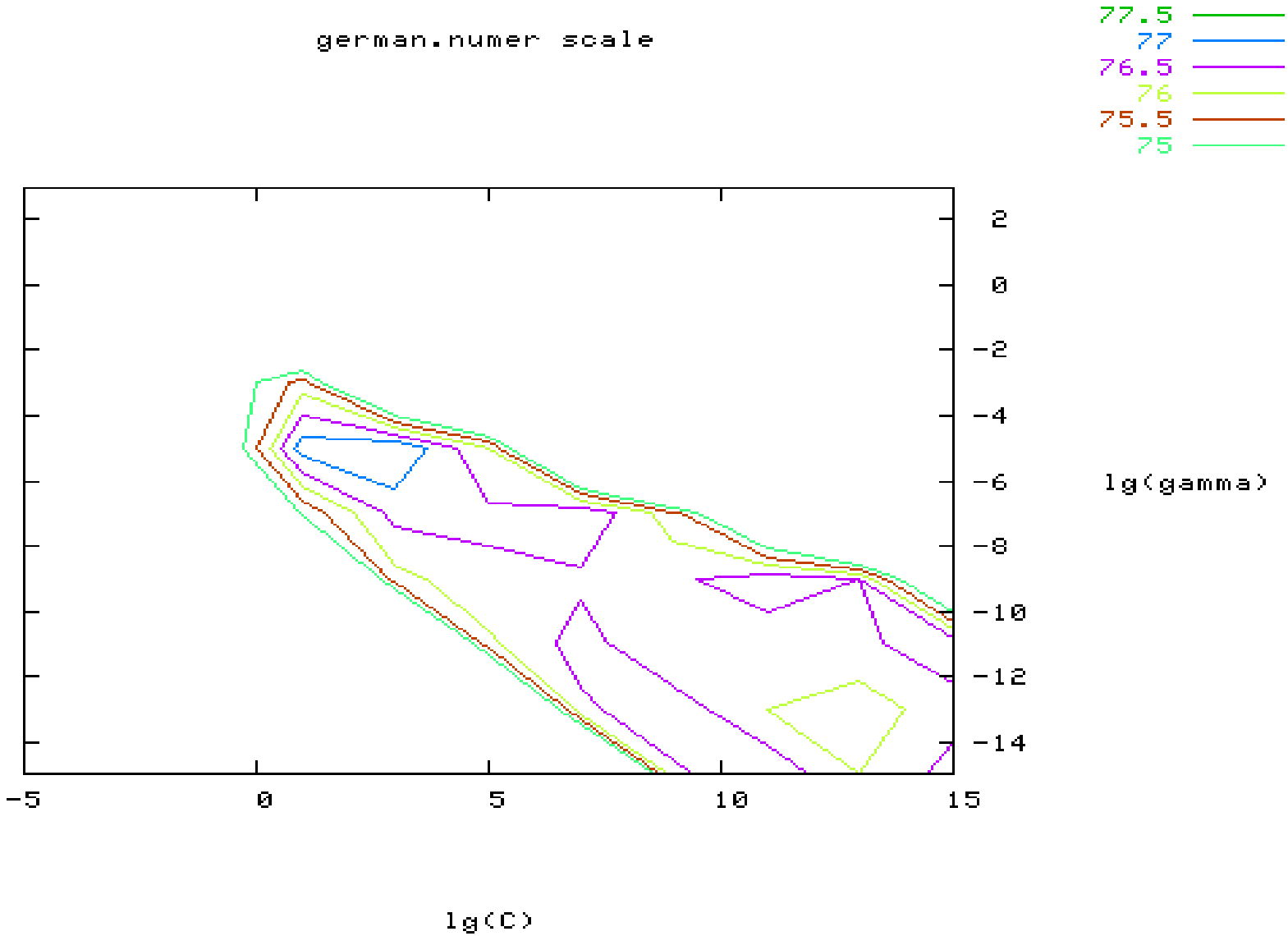
Performance Evaluation

- Available data $\Rightarrow$ training and validation
- Train the training; test the validation
- k-fold cross validation (CV):

  Data randomly separated to k groups

  Each time $k-1$ as training and one as testing
- Select parameters/kernels with best CV result

Contour of CV Accuracy



- The good region of parameters is quite large
- SVM is sensitive to parameters, but not that sensitive
- Sometimes default parameters work

  but it's good to select them if time is allowed

Example of Parameter Selection

```
$./svm-train svmguide3
$./svm-predict svmguide3.t svmguide3.model o
```

→ Accuracy = 2.43902%

```
$./svm-scale -s range3 svmguide3 > svmguide3.sca
$./svm-scale -r range3 svmguide3.t > svmguide3.t
$./svm-train svmguide3.scale
$./svm-predict svmguide3.t.scale svmguide3.scale
```

→ Accuracy = 12.1951%

Low accuracy even after data scaling

Example of Parameter Selection (Cont'd)

Conduct parameter selection by a simple grid search

```
$ python grid.py svmguide3.scale
...
128.0 0.125 84.8753
```

(Best C=128.0, γ=0.125 with five-fold cross-validation rate=84.8753%)

Train and predict using the obtained parameters

```
$ ./svm-train -c 128 -g 0.125 svmguide3.scale
$ ./svm-predict svmguide3.t.scale svmguide3.scal
```

$\rightarrow$ Accuracy = 87.8049%

Selecting Kernels

- RBF, polynomial, or others?
- For beginners, use RBF first
- Linear kernel: special case of RBF

  Performance of linear the same as RBF under certain parameters (Keerthi and Lin, 2003)
- Polynomial kernel:

$$(\mathbf{x}_i^T \mathbf{x}_j / a + b)^d$$

  Numerical difficulties: $(< 1)^d \to 0, (> 1)^d \to \infty$

  More parameters than RBF

A Simple Procedure for Beginners

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

In LIBSVM, we have a python script `easy.py` implementing this procedure. It has been very useful for beginners

More details can be seen in Hsu et al. (2003).

A Real Example of Wrong Scaling

Separately scale each feature of training and testing data to $[0, 1]$

```
$ ../svm-scale -l 0 svmguide4 > svmguide4.scale
$ ../svm-scale -l 0 svmguide4.t > svmguide4.t.s
$ python easy.py svmguide4.scale svmguide4.t.sca
Accuracy = 69.2308% (216/312) (classification)
```

The accuracy is low even after parameter selection

```
$../svm-scale -l 0 -s range4 svmguide4 > svmgui
$ ../svm-scale -r range4 svmguide4.t > svmguide4
$ python easy.py svmguide4.scale svmguide4.t.sca
Accuracy = 89.4231% (279/312) (classification)
```

A Real Example of Wrong Scaling (Cont'd)

With the correct setting, the 10 features in testing data `svmguide4.t.scale` have the following maximal values:

0.7402, 0.4421, 0.6291, 0.8583, 0.5385, 0.7407, 0.3982, 1.0000, 0.8218, 0.9874

Scaling the testing set to $[0, 1]$ generated an erroneous set.

Outline

Outline

Users' Machine Learning Knowledge

- When we started developing LIBSVM, we didn't know who our users are or whether we will get any
- Very soon we found that for a classification package like LIBSVM, many users have zero machine learning knowledge
- It is unbelievable that many asked what the difference between training and testing is

Users' Machine Learning Knowledge (Cont'd)

- A sample mail

```
From:
To: cjlin@csie.ntu.edu.tw
Subject: Doubt regarding SVM
Date: Sun, 18 Jun 2006 10:04:01 +0530 (IST)

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

- Sometimes we cannot do much for such users.

Users' Machine Learning Knowledge (Cont'd)

- Fortunately, more people have taken machine learning courses (or attend MLSS)
- On the other hand, because users are not machine learning researchers, some automatic or semi-automatic settings are helpful
- This leads to the simple procedure discussed above.
- Also, your target users affect your design.

  For example, we assume LIBLINEAR users are more experienced.

We are Our Own Users

- You may ask why we care non-machine learning users so much
- The reason is that we were among them before
- My background is in optimization. When we started working on SVM, we tried some UCI sets.
- We failed to obtain similar accuracy values in papers
- Through a painful process we learned that scaling may be needed

We are Our Own Users (Cont'd)

- Machine learning researchers sometimes failed to see the difficulties of general users.
- As users of our own software, we constantly think about difficulties others may face

Users are Our Teachers

- While we criticize users' lack of machine learning knowledge, they help to 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 data are in k classes

Users are Our Teachers (Cont'd)

- One-against-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-against-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 well cited.
- Currently LIBSVM supports one-vs-one approach

Users are Our Teachers (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.

Outline

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 software 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

  Example: Some need χ^2 kernel, but some have no idea what it is

One or Many Options (Cont'd)

- For LIBSVM, we basically took the "one option" approach

  We are very careful in adding things to LIBSVM
- However, users do have different needs. For example, some need precision/recall rather than accuracy
- We end up with developing another web site "LIBSVM Tools" to serve users' special needs

One or Many Options (Cont'd)

- Sample code in LIBSVM tools

  - Cross Validation with Different Criteria (AUC, F-score, etc.)

  - ROC Curve for Binary SVM

  - LIBSVM for string data
- Not sure if this is the best way, but seems ok so far
- Another advantage is we can maintain high quality for the core package. Things in LIBSVM Tools are less well maintained.

Simplicity versus Better Performance

- This issue is related to "one or many options" discussed before
- Example: Before, our cross validation (CV) procedure is not stratified

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

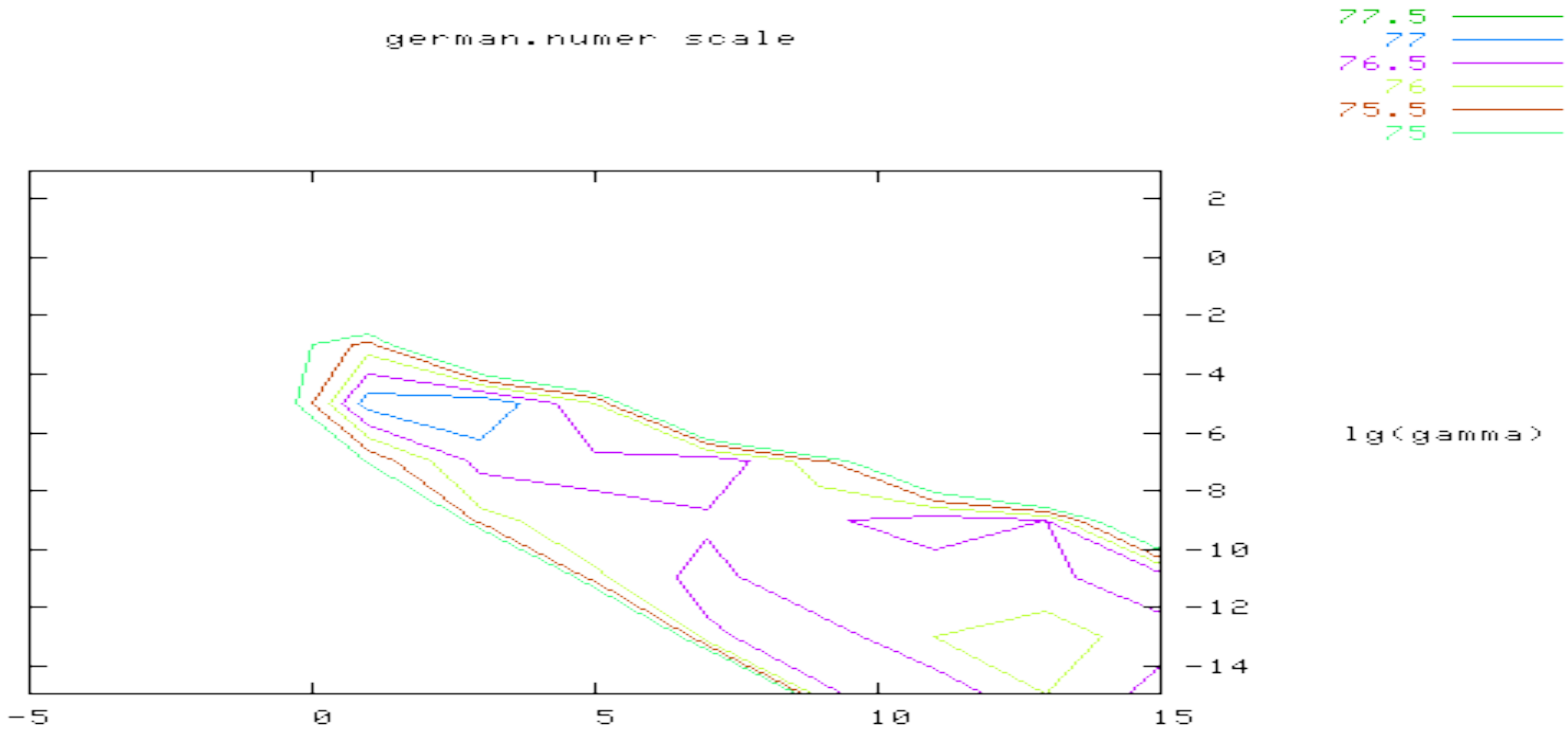
  - We now support stratified CV, but code becomes more complicated
- In general, we avoid changes for just marginal improvements

Simplicity versus Better Performance (Cont'd)

- A recent 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

Simplicity versus Better Performance (Cont'd)

Example: LIBSVM uses a grid search to find two parameters C and γ. We may think this is simple and naive



Simplicity versus Better Performance (Cont'd)

- Indeed, we studied loo bound in detail:

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

and solved

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

- Results not very stable because $f(C, \gamma)$ is only an approximation. Implementation is quite complicated.
- For only two parameters, a simple grid search may be a suitable choice

Numerical Stability

- Many classification methods (e.g., SVM, neural networks) solve optimization problems
- Core of the implementation is essentially a numerical method
- Numerical analysts have a high standard on their code, but we machine learning people do not
- This situation is expected:
  If we put efforts on implementing method A and one day method B gives higher accuracy $\Rightarrow$ Efforts are wasted

Numerical Stability (Cont'd)

- However, quality of the numerical programs is important for a machine learning package
- We will give an example.
- In LIBSVM's probability outputs and LIBLINEAR's logistic regression, we calculate

$$-\sum_{i=1}^{l}\Big(t_i \log(p_i) + (1-t_i)\log(1-p_i)\Big),$$

where

$$p_i \equiv \frac{1}{1+\exp(Af_i + B)}$$

Numerical Stability (Cont'd)

- log and exp could easily cause an overflow.

$$\text{If } Af_i + B \text{ is large} \Rightarrow \exp(Af_i + B) \to \infty.$$

- Then

$$p_i \approx 0 \Rightarrow \log(p_i) \quad \to \quad -\infty$$

- When p_i is close to one.

$$1 - p_i = 1 - \frac{1}{1 + \exp(Af_i + B)}$$

is a catastrophic cancellation

Numerical Stability (Cont'd)

- Catastrophic cancellation (Goldberg, 1991): when subtracting two nearby numbers that are already results of floating-point operations, the relative error can be large so most digits are meaningless.
- If

$$f_i = 1, \text{ and } (A, B) = (-64, 0),$$

in a simple C++ program with double precision,

$$1 - p_i \text{ returns zero}$$

but

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

Numerical Stability (Cont'd)

- Catastrophic cancellation can usually be resolved by reformulation:

$$-\Big(t_i \log p_i + (1 - t_i) \log(1 - p_i)\Big) \tag{1}$$

$$= (t_i - 1)(Af_i + B) + \log\Big(1 + \exp(Af_i + B)\Big) \tag{2}$$

$$= t_i(Af_i + B) + \log\Big(1 + \exp(-Af_i - B)\Big) \tag{3}$$

- To handle the overflow issue, we implement (1) with the following rule:

If $Af_i + B \geq 0$ then use (3); Else use (2).

Legacy Issues

- The compatibility between earlier and later versions is an issue
- Such legacy issues restrict developers to conduct certain changes.
- We face a similar situation. For example, we chose "one-vs-one" as the multi-class strategy. This decision affects subsequent buildups.
- Multi-class probability outputs must follow the one-vs-one structure. For classes i and j, we obtain

$$P(\mathbf{x} \text{ in class } i \mid \mathbf{x} \text{ in class } i \text{ or } j),$$

Legacy Issues (Cont'd)

- Then we need to couple all $\binom{k}{2}$ results (k: the number of classes) and obtain

$$P(\mathbf{x} \text{ in class } i), i = 1, \ldots, k.$$

- If we further develop multi-label methods, we are restricted to extend from one-versus-one multi-class strategy
- If we considered other multi-class methods, methods for multi-class probabilities and multi-label classification would be different.

Legacy Issues (Cont'd)

- In LIBSVM, we understand this legacy issue in the beginning
- Example: we did not make the trained model a public structure
- Typically a user write the following C code to train and test

```
#include <svm.h>
...
model = svm_train(...);
...
predict_label = svm_predict(model,x);
```

- `svm.h` includes all public functions and structures

Legacy Issues (Cont'd)

- We decided not to put model structure in `svm.h`
  Instead we put it in `svm.cpp`
- User can call

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

  but cannot do

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

- We provide functions so users can obtain some model information

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

Legacy Issues (Cont'd)

- Reason: if one day we replace one-versus-one method with another one, users are transparent to the change
- But some users (mainly machine learning researchers) complained

  They need to access details of the $\binom{k}{2}$ models
- We insisted on not changing it for a long time.
- Recently, because software becomes mature and the chance of switching to another multi-class strategy is small, we make the structure public.

Documentation and Support

- Any software needs good documents and support
- I cannot count how many mails my students and I replied. Maybe 20,000 or more.
- How to write good documents is an interesting issue

  Users may not understand what you wrote
- Here is an example: some users asked if LIBSVM supported multi-class classification
- I thought it's well documented in `README`

Documentation and Support (Cont'd)

- Finally I realized that they didn't read the whole README.
- And they didn't see "multi-class" in the usage

```
Usage: svm-train [options] training_set_file
options:
-s svm_type : set type of SVM (default 0)
    0 -- C-SVC
    1 -- nu-SVC
    2 -- one-class SVM
    3 -- epsilon-SVR
    4 -- nu-SVR
...
```

Documentation and Support (Cont'd)

- In the next version we will change the usage to

```
-s svm_type : set type of SVM (default 0)
    0 -- C-SVC              (multi-class classific
    1 -- nu-SVC             (multi-class classific
    2 -- one-class SVM
    3 -- epsilon-SVR        (regression)
    4 -- nu-SVR             (regression)
```

- I am going to see how many asked "why LIBSVM doesn't support two-class SVM"

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

  Example: to check an implementation trick in a proposed algorithm, need to run with/without the trick

Software versus Experiment Code (Cont'd)

- Software: for general users

  One or a few reasonable settings with a suitable interface are enough
- Many are now willing to release their experimental code

  Basically you clean up the code after finishing a paper
- But working on and maintaining high-quality software take much more work

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
- In my group, we release experiment code for every paper $\Rightarrow$ for replicability

  And carefully select and modify some results to our software $\Rightarrow$ (hopefully) for reproducibility

Software versus Experiment Code (Cont'd)

- The community now lacks incentives for researchers to work on high quality software
- JMLR recently started "open source software" section (4-page description of the software)
- This is a positive direction
- How to properly evaluate such papers is an issue
- Some software are very specific on a small problem, but some are more general

Research versus Software Development

Shouldn't software be developed by companies?

Two issues

1. Business models of machine learning software
2. Research problems in developing software

Research versus Software Development (Cont'd)

Business model

- It is unclear to me what a good model should be
- Machine learning software are basically "research" software
- For example, LIBSVM and LIBLINEAR are used by Weka and Rapidminer through some interface functions
- These data mining packages are open sourced and their business is mainly on consulting
- Should we on the machine learning side use a similar way?

Research versus Software Development (Cont'd)

Research issues

- A good machine learning package involves more than the core machine learning algorithms
- There are many other research issues
  - Numerical stability
  - Solving optimization problems
  - Parameter tuning
  - Serious comparisons
- These issues need researchers rather than engineers
- Currently we lack a system to encourage machine learning researchers to study these issues.

Conclusions

- From my experience, developing machine learning software is very interesting.
- We have learned a lot from users in different application areas.
- We should encourage more researchers to develop high quality machine learning software

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

References I

B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

C. Cortes and V. Vapnik. Support-vector network. *Machine Learning*, 20:273–297, 1995.

C. Drummond. Replicability is not reproducibility: Nor is it good science. In *Proceedings of the Evaluation Methods for Machine Learning Workshop at the 26th ICML*, 2009.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/liblinear.pdf`.

D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991.

C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

C.-W. Hsu, C.-C. Chang, and C.-J. 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`.

References II

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

S. Sonnenburg, M. Braun, C. Ong, S. Bengio, L. Bottou, G. Holmes, Y. LeCun, K. Müller, F. Pereira, C. Rasmussen, et al. The need for open source software in machine learning. *Journal of Machine Learning Research*, 8:2443–2466, 2007.