# A Practical Guide to Support Vector Classification

**Chih-Jen Lin**

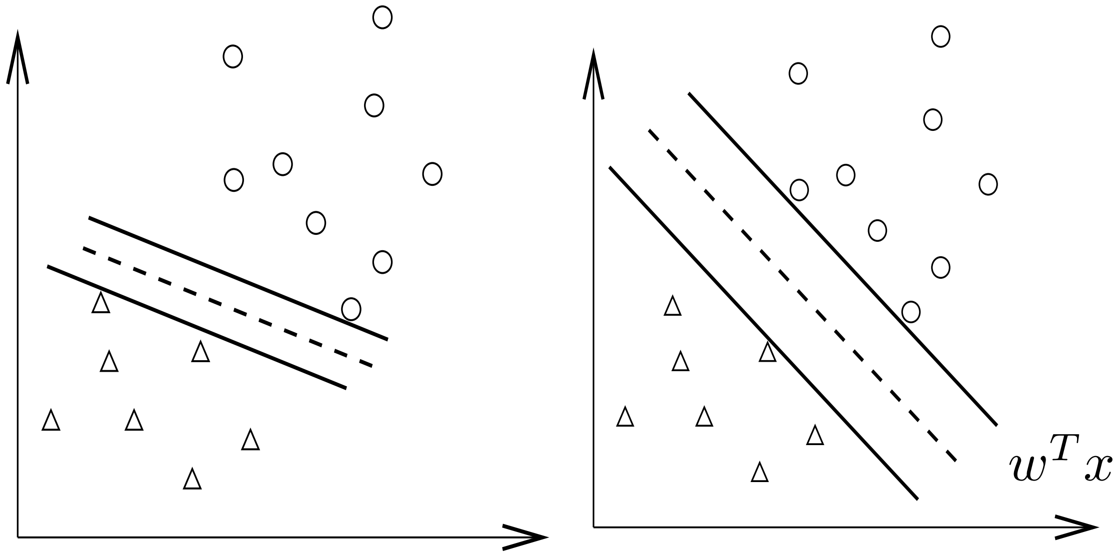
Department of Computer Science
National Taiwan University

Talk at University of Freiburg, July 15, 2003

## Motivation and Outline

- SVM: a hot machine learning issue
- However, many beginners get unsatisfactory accuracy at first
  Some easy but significant steps missed
- This talk
  - Some cookbook approaches based on our experience serving users
  - No guarantee for the best accuracy but usually reasonable accuracy
  - Hope beginners get acceptable results fast and easily.
  - Challenging cases and further extension
    What do we plan to add in LIBSVM

# Basic Concepts of SVM



$$w^T x + b = \begin{bmatrix} +1 \\ 0 \\ -1 \end{bmatrix}$$

$$\min_{\color{red}{w,b,\xi}} \quad \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i$$

$$\text{subject to} \quad y_i(w^T {\color{red}\phi(x_i)} + b) \geq 1 - \xi_i, \;\; \xi_i \geq 0, \; i = 1, \ldots, l.$$

- Kernel: $K(x, y) = \phi(x)^T \phi(y)$

## What Many Beginners are Doing Now

- Transfer data to the format of an SVM software
- May not conduct scaling
- Randomly try few parameters and kernels without validation
- Default parameters are surprisingly important
- If most users doing so, accuracy may not be satisfactory

# Examples

| | training data | testing data | features | classes | Accuracy by users | Accuracy by us |
|---|---|---|---|---|---|---|
| User 1 | 3,089 | 4,000 | 4 | 2 | 75.2% | 96.9% |
| User 2 | 391 | 0 | 20 | 3 | 36% | 85.2% |
| User 3 | 1,243 | 41 | 21 | 2 | 4.88% | 87.8% |

- User 1:

  `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...`

- Answer:

  OK. Send me the data

- Answer:

  I am able to get 97% test accuracy. Is that good enough for you ?

- User 1:

  `You earned a copy of my PhD thesis`

- User 2:

  `I am a developer in a bioinformatics laboratory at ... We would like to use LIBSVM in a project ... But results not good. 36% CV accuracy`

- Answer:

  OK. Send me the data

- Answer:

  I am able to give 83.88% cv accuracy. Is that good enough for you ?

- User 2:

  83.88% accuracy would be excellent...

- User 3:

  `I have problems getting the same result with SVM to compared to neural nets.`

  `Right now I get a correct of 4.88%, which is very bad (neural net 70-90%).`

- Answer

  I play a bit your data. My testing accuracy is 87.8%. Is this good for you ?

- User 3:

  `I found myself described in your talk ;-)`

## We Hope Users At Least Do

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

# Why RBF

- Linear kernel: special case of RBF [Keerthi and Lin 2003]
- Polynomial: numerical difficulties

  $(< 1)^d \rightarrow 0, (> 1)^d \rightarrow \infty$

  More parameters than RBF
- tanh: still a mystery

  May not be positive semi-definite

  In [Lin and Lin 2003], for certain parameters, it behaves like RBF

  Should avoid using tanh in general

## Examples: Using the Proposed Procedure

User 1

- Original sets with default parameters

```
$./svm-train train.1
$./svm-predict test.1 train.1.model test.1.predict
```

 → `Accuracy = 66.925%`

- Scaled sets with default parameters

```
$./svm-scale -s range1 train.1 > train.1.scale
$./svm-scale -r range1 test.1 > test.1.scale
$./svm-train train.1.scale
$./svm-predict test.1.scale train.1.scale.model test.1.predict
```

 → `Accuracy = 96.15%`

- Scaled sets with parameter selection

```
$python grid.py train.1.scale
...
2.0 2.0 96.8922
```

(Best $C$=2.0, $\gamma$=2.0 with five-fold cross-validation rate=96.8922%)

```
$./svm-train -c 2 -g 2 train.1.scale
$./svm-predict test.1.scale train.1.scale.model test.1.predict
```

$\rightarrow$ `Accuracy = 96.875%`

User 2

- Original sets with default parameters

```
$./svm-train -v 5 train.2
```

$\rightarrow$ `Cross Validation Accuracy = 56.5217%`

- Scaled sets with default parameters

```
$./svm-scale train.2 > train.2.scale
$./svm-train -v 5 train.2.scale
```

→ `Cross Validation Accuracy = 78.5166%`

- Scaled sets with parameter selection

  ```
  $python grid.py train.2.scale
  ...
  2.0 0.5 85.1662
  ```

  → `Cross Validation Accuracy = 85.1662%`

  (Best $C$=2.0, $\gamma$=0.5 with five fold cross-validation rate=85.1662%)

User 3

- Original sets with default parameters

  ```
  $./svm-train train.3
  $./svm-predict test.3 train.3.model test.3.predict
  ```

  → `Accuracy = 2.43902%`

- Scaled sets with default parameters

```
$./svm-scale -s range3 train.3 > train.3.scale
$./svm-scale -r range3 test.3 > test.3.scale
$./svm-train train.3.scale
$./svm-predict test.3.scale train.3.scale.model test.3.predict
```

$\rightarrow$ `Accuracy = 12.1951%`

- Scaled sets with parameter selection

```
$python grid.py train.3.scale
...
128.0 0.125 84.8753
```

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

```
$./svm-train -c 128 -g 0.125 train.3.scale
$./svm-predict test.3.scale train.3.scale.model test.3.predict
```

$\rightarrow$ `Accuracy = 87.8049%`

## Scaling

- Important for Neural Networks (Part 2 of NN FAQ)

  Most reasons apply here

- Attributes in greater numeric ranges may dominate

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

- Simple linearly scaling each attribute to $[-1,+1]$ or $[0,1]$.
- The same scaling factor for testing

# Model Selection

- In fact, two-parameter search: $C$ and $\gamma$
- We recommend a simple grid search using cross-validation

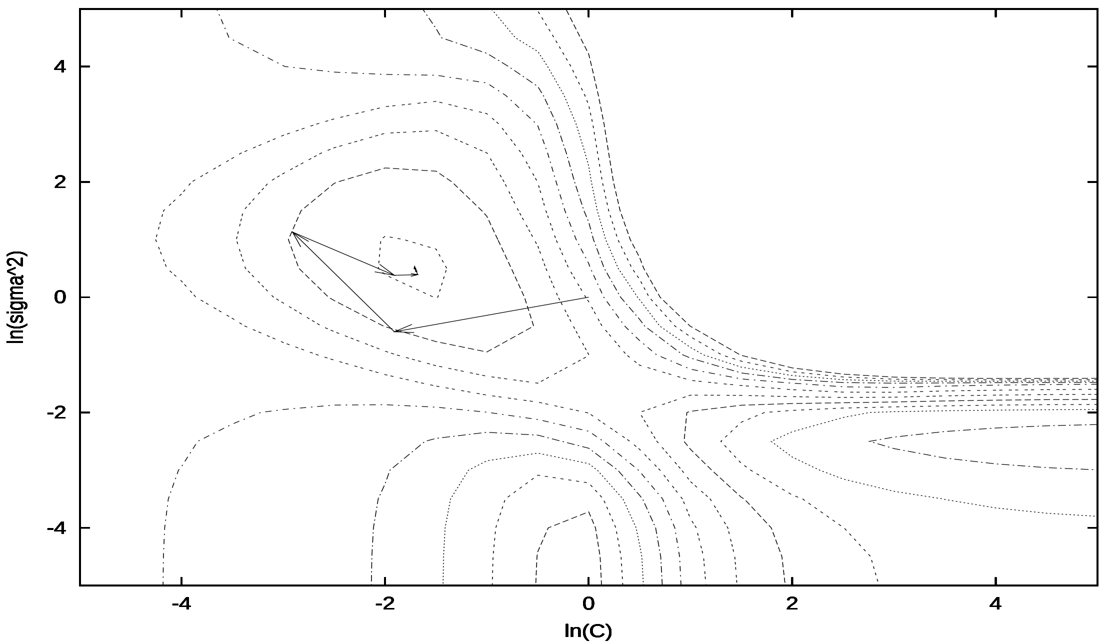
  E.g. 5-fold CV on $C = 2^{-5}, 2^{-3}, \ldots, 2^{15}$, $\gamma = 2^{-15}, 2^{-13}, \ldots, 2^{3}$
- Why not more efficient methods

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

  so

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

  – A path may be found

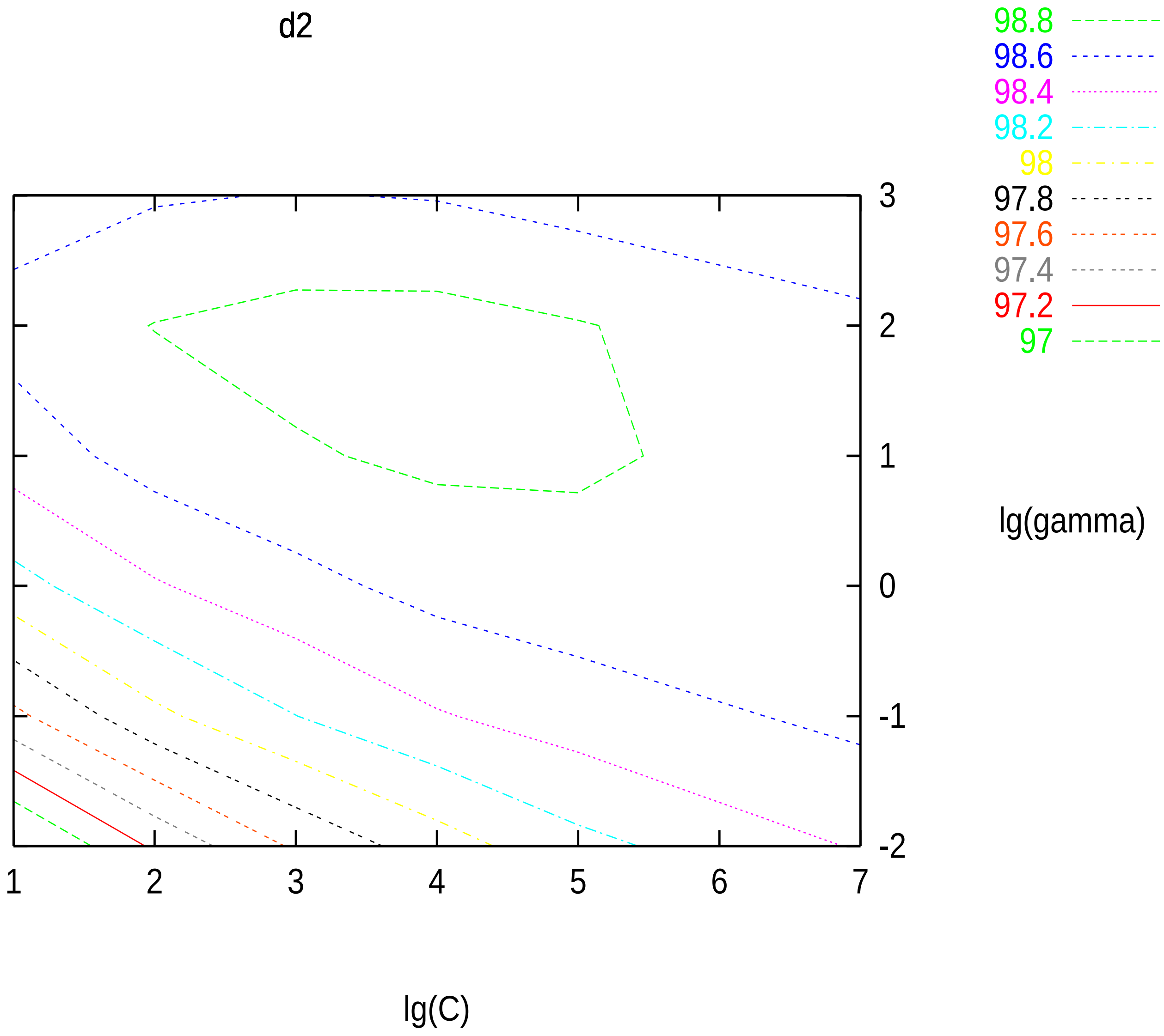

4
2
0
-2
-4
ln(sigma^2)
-4
-2
0
2
4
ln(C)

- Reasons for not using bounds (if two parameters)
  - Implementation more complicated
  - Psychologically, not feel safe
  - In practice: IJCNN competition:

    97.09% and 97.83% using Radius Margin bounds for L1 and L2-SVM

    98.59% using 25-point grid

    2668, 1990, and 1293 testing errors
  - Bounds are useful if more than two parameters


d2
98.8
98.6
98.4
98.2
98
97.8
97.6
97.4
97.2
97
3
2
1
0
-1
-2
lg(gamma)
1
2
3
4
5
6
7
lg(C)

- We propose that users do
  - Start from a loose grid
  - Identify good regions and use a finer grid
- The grid search tool in libsvm
- Easy parallelization

  Every problem is independent

  loo bounds: 20 steps $\Rightarrow$ about $10 \times 10$ grids with five computers

  Automatic load balancing

## Example: Automatic Script

- User 1

```
$python easy.py train.1 test.1
Scaling training data...
Cross validation...
Best c=2.0, g=2.0
Training...
Scaling testing data...
Testing...
Accuracy = 96.875% (3875/4000) (classification)
```

- User 3

```
$python easy.py train.3 test.3
Scaling training data...
Cross validation...
```

```
Best c=128.0, g=0.125
Training...
Scaling testing data...
Testing...
Accuracy = 87.8049% (36/41) (classification)
```

## Challenges

- Is the procedure good enough ?

  Good for some median-sized data sets

- Difficult problems: this procedure not enough
  - Too much training time
  - Low accuracy
- Extension of the procedure ?
- What are we going to include in LIBSVM?

## Feature Selection

- Too many (non-zero) features

  Examples here: 4, 20, 21 features $\ll$ #data

- RBF kernel

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

  Irrelevant attributes cause problems

- How about

$$K(x, y) = e^{-\sum_{i=1}^{n} \gamma_i (x_i - y_i)^2}$$

  Difficult to choose $\gamma_i$

  Possible approaches (e.g. [Chapelle et al. 2002]):

$$\text{leave-one-out error} \leq f(C, \gamma_1, \ldots, \gamma_n)$$

  A non-convex problem. Difficult and unstable.

- Feature selection before training SVM

  SVM can help feature selection as well

  E.g. linear SVM

$$f(x) = w^T x + b$$

  Choose indices with large $|w_i|$ [Guyon et al. 2002]

- Overall, a very difficult issue

  Not sure if a simple and systematic procedure available ?

# Probability Estimates

- SVM outputs decision values only
- Probability estimates for two-class SVM:
  - Platt's sigmoid approximation
  - Isotonic regression
  - SVM density estimation ?

  We are conducting a serious evaluation
- Multi-class probability estimate

  Related to multi-class classification

Currently LIBSVM uses 1vs1 (after an evaluation in [Hsu and Lin, 2002])

10 classes: 45 SVMs, 0vs1, 0vs2, . . . , 8vs9

Given $r_{ij} \approx P(y = i \mid y = i \text{ or } j)$, estimate $P(y = i)$

An issue for all binary classification methods

New and stable methods proposed in [Wu et al., 2003]

- All these are about ready

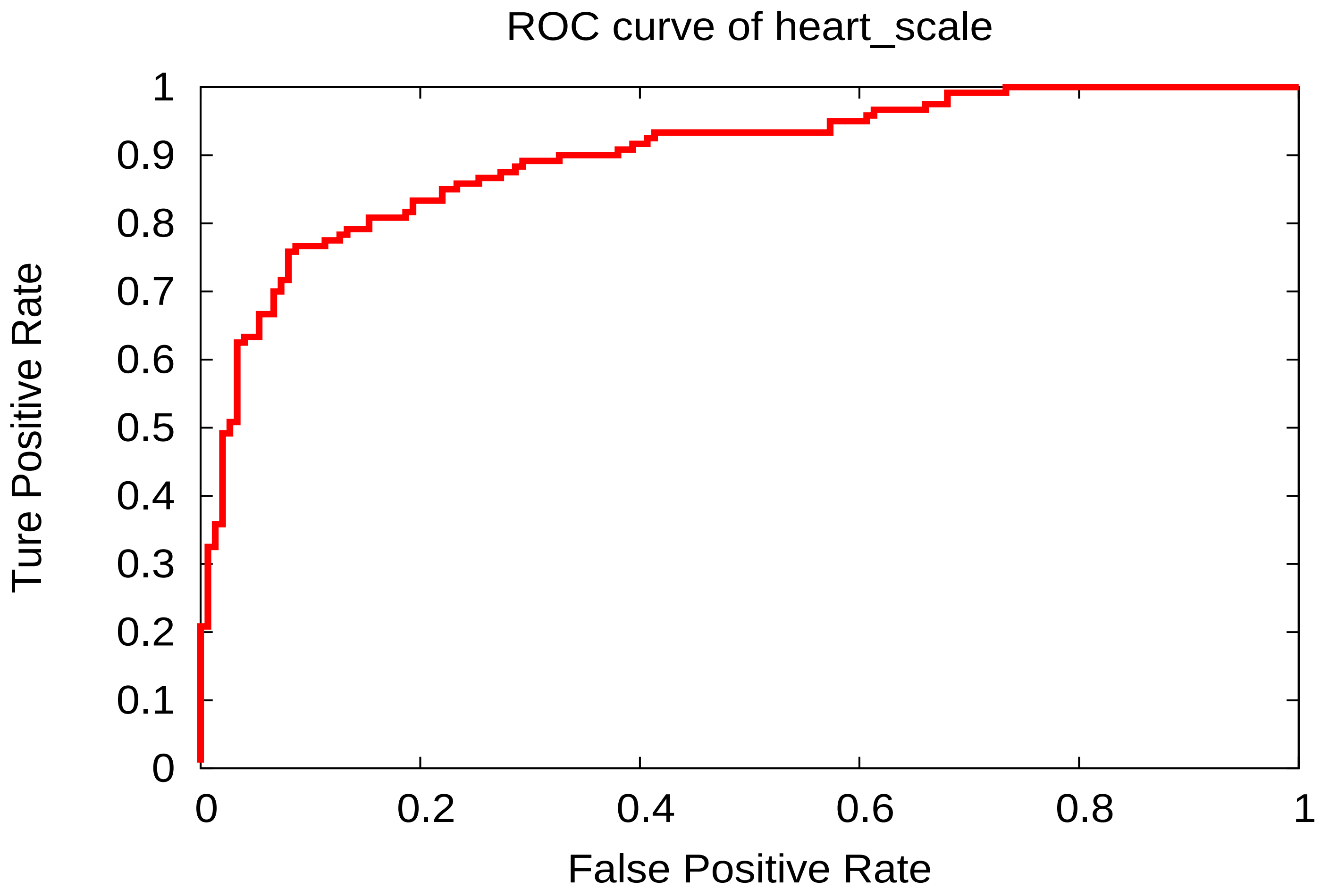
  The main addition to next version of LIBSVM

## Unbalanced Data

- Many information retrieval users ask about ROC curve and adjusting precision/recall

  Not accuracy any more

- Three ways to generate ROC curves
  - Adjust $b$ of
    $$f(x) = w^T x + b$$
  - Unbalanced cost function
    $$\min_{w,b,\xi} \frac{1}{2} w^T w + C_+ \sum_{i:y_i=1} \xi_i + C_- \sum_{i:y_i=-1} \xi_i$$
  - Rank by probability output + cross validation (now available)
  - Which one is more useful ?




- Goal: an integrated tool so users can easily adjust cost matrices or the relation of TP, TN, FT, FN

# Conclusions

- Still a long way to serve all users' needs but we are trying
- We hope more users can benefit from this research and eventually SVM can be an easy-to-use classification method
- Slides based on

  Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin

  A Practical Guide to Support Vector Classification `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`
- LIBSVM available at

  `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
- We thank all users for their comments