Support Vector Machines for Data Classification

Chih-Jen Lin

Department of Computer Science
National Taiwan University

Outline

- Support vector classification
- Practical use of SVM
- Support vector regression
- An Example
- Discussion and conclusions

Data Classification

- Given training data in different classes (labels known)

  Predict test data (labels unknown)
- Examples
  - Handwritten digits recognition
  - Spam filtering
  - Text classification
  - Prediction of signal peptide in human secretory proteins
- Training and testing

- Methods:
  - Nearest Neighbor
  - Neural Networks
  - Decision Tree
- Support vector machines: a new method
- Becoming more and more popular

Why Support Vector Machines

- Existing methods:

  Nearest neighbor, Neural networks, decision trees.
- SVM: a new one
- In my opinion, after careful data pre-processing

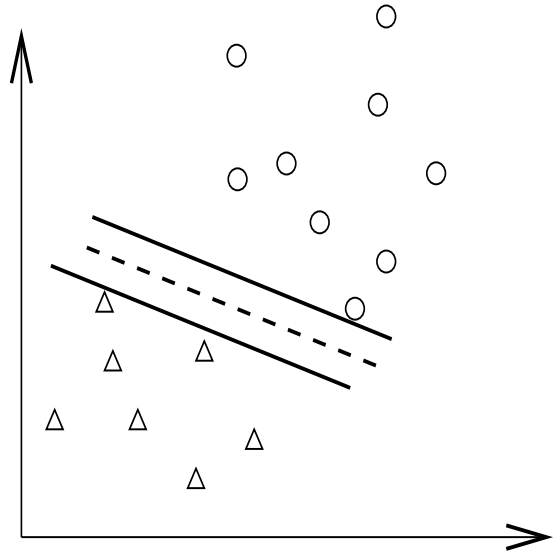
  Appropriately use NN or SVM $\Rightarrow$ similar accuracy
- But, users may not use them properly
- The chance of SVM
  - Easier for users to appropriately use it
  - The ambition: replacing NN on some applications

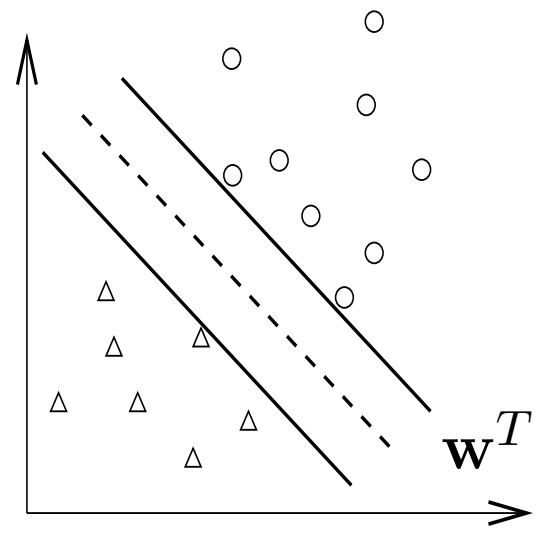
Support Vector Classification

- Training vectors : $\mathbf{x}_i, i = 1, \ldots, l$
- Consider a simple case with two classes:
  Define a vector $\mathbf{y}$

$$y_i = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ in class 1} \\ -1 & \text{if } \mathbf{x}_i \text{ in class 2,} \end{cases}$$

- A hyperplane which separates all data





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

- A separating hyperplane: $\mathbf{w}^T\mathbf{x} + b = 0$

$$(\mathbf{w}^T\mathbf{x}_i) + b > 0 \quad \text{if } y_i = 1$$
$$(\mathbf{w}^T\mathbf{x}_i) + b < 0 \quad \text{if } y_i = -1$$

- Decision function $f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$, $\mathbf{x}$: test data

  Variables: $\mathbf{w}$ and b : Need to know coefficients of a plane

  Many possible choices of $\mathbf{w}$ and b

- Select $\mathbf{w}, b$ with the maximal margin.
  Maximal distance between $\mathbf{w}^T\mathbf{x} + b = \pm 1$

$$\begin{aligned} (\mathbf{w}^T\mathbf{x}_i) + b \geq 1 \quad & \text{if } y_i = 1 \\ (\mathbf{w}^T\mathbf{x}_i) + b \leq -1 \quad & \text{if } y_i = -1 \end{aligned}$$

- Distance between $\mathbf{w}^T\mathbf{x} + b = 1$ and -1:

$$2/\|\mathbf{w}\| = 2/\sqrt{\mathbf{w}^T\mathbf{w}}$$

- $\max 2/\|\mathbf{w}\| \equiv \min \mathbf{w}^T\mathbf{w}/2$

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & y_i((\mathbf{w}^T\mathbf{x}_i) + b) \geq 1, \\ & i = 1, \ldots, l. \end{aligned}$$

Higher Dimensional Feature Spaces

- Earlier we tried to find a linear separating hyperplane

  Data may not be linear separable
- Non-separable case: allow training errors

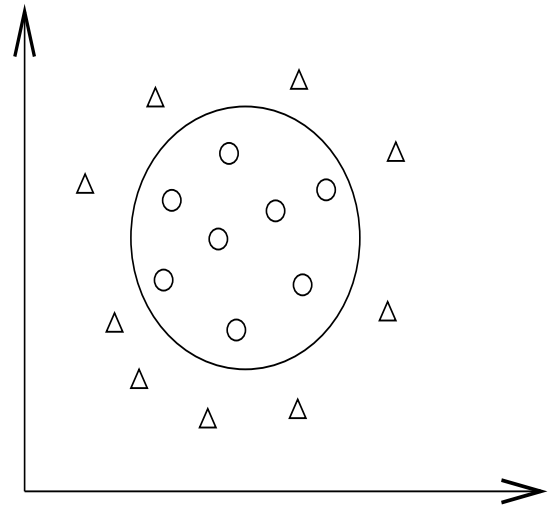
$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i$$

$$y_i((\mathbf{w}^T\mathbf{x}_i) + b) \geq 1 - \xi_i,$$

$$\xi_i \geq 0,\ i = 1, \ldots, l$$

- $\xi_i > 1$, $\mathbf{x}_i$ not on the correct side of the separating plane
- C: large penalty parameter, most ξ_i are zero

- Nonlinear case: linear separable in other spaces ?



- Higher dimensional (maybe infinite) feature space

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

- Example: $\mathbf{x} \in R^3, \phi(\mathbf{x}) \in R^{10}$

$$\begin{aligned}\phi(\mathbf{x}) \quad = \quad & (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, x_1^2, \\ & x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3)\end{aligned}$$

- A standard problem (Cortes and Vapnik, 1995):

$$\begin{aligned}\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \ i = 1, \ldots, l.\end{aligned}$$

Finding the Decision Function

- $\mathbf{w}$: a vector in a high dimensional space $\Rightarrow$ maybe infinite variables
- The dual problem

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

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

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

- Primal and dual: optimization theory. Not trivial.

  Infinite dimensional programming.
- A finite problem:

  #variables = #training data
- $Q_{ij} = y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ needs a closed form

  Efficient calculation of high dimensional inner products

  Kernel trick, $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

- Example: $\mathbf{x}_i \in R^3, \phi(\mathbf{x}_i) \in R^{10}$

$$\begin{aligned}\phi(\mathbf{x}_i) &= (1, \sqrt{2}(x_i)_1, \sqrt{2}(x_i)_2, \sqrt{2}(x_i)_3, (x_i)_1^2, \\ &\quad (x_i)_2^2, (x_i)_3^2, \sqrt{2}(x_i)_1(x_i)_2, \sqrt{2}(x_i)_1(x_i)_3, \sqrt{2}(x_i)_2(x_i)_3)\end{aligned}$$

  Then $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$.

- Popular methods: $K(\mathbf{x}_i, \mathbf{x}_j) =$

$$e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}, \text{ (Radial Basis Function)}$$

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

Kernel Tricks

- Kernel: $K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y})$
- No need to explicitly know $\phi(\mathbf{x})$
- Common kernels $K(\mathbf{x}_i, \mathbf{x}_j) =$

$$e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}, \text{ (Radial Basis Function)}$$

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

- They can be inner product in infinite dimensional space
- Assume $x \in R^1$ and $\gamma > 0$.

$$
\begin{aligned}
& e^{-\gamma\|x_i - x_j\|^2} = e^{-\gamma(x_i - x_j)^2} = e^{-\gamma x_i^2 + 2\gamma x_i x_j - \gamma x_j^2} \\
= \quad & e^{-\gamma x_i^2 - \gamma x_j^2}\left(1 + \frac{2\gamma x_i x_j}{1!} + \frac{(2\gamma x_i x_j)^2}{2!} + \frac{(2\gamma x_i x_j)^3}{3!} + \cdots\right) \\
= \quad & e^{-\gamma x_i^2 - \gamma x_j^2}\left(1 \cdot 1 + \sqrt{\frac{2\gamma}{1!}} x_i \cdot \sqrt{\frac{2\gamma}{1!}} x_j + \sqrt{\frac{(2\gamma)^2}{2!}} x_i^2 \cdot \sqrt{\frac{(2\gamma)^2}{2!}} x_j^2\right. \\
& \qquad \left. + \sqrt{\frac{(2\gamma)^3}{3!}} x_i^3 \cdot \sqrt{\frac{(2\gamma)^3}{3!}} x_j^3 + \cdots\right) \\
= \quad & \phi(x_i)^T \phi(x_j),
\end{aligned}
$$

where

$$
\phi(x) = e^{-\gamma x^2}\left[1, \sqrt{\frac{2\gamma}{1!}} x, \sqrt{\frac{(2\gamma)^2}{2!}} x^2, \sqrt{\frac{(2\gamma)^3}{3!}} x^3, \cdots\right]^T.
$$

Decision function

- $\mathbf{w}$: maybe an infinite vector
- At optimum

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

- Decision function

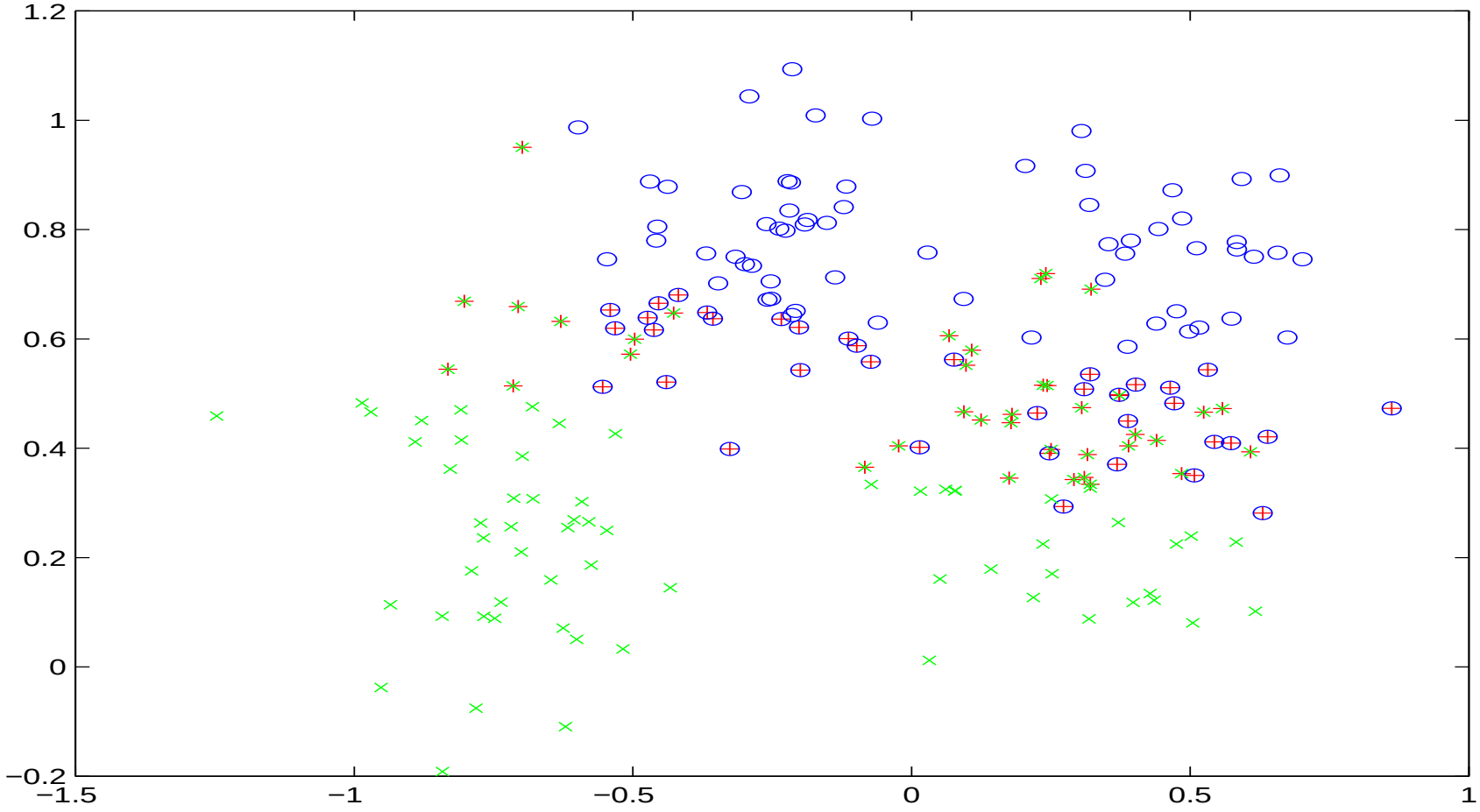
$$\begin{aligned} & \mathbf{w}^T \phi(\mathbf{x}) + b \\ = & \sum_{i=1}^{l} \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + b \\ = & \sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \end{aligned}$$

No need to have $\mathbf{w}$

- > 0: 1st class, < 0: 2nd class
- Only $\phi(\mathbf{x}_i)$ of $\alpha_i > 0$ used

$\alpha_i > 0 \Rightarrow$ support vectors

Support Vectors: More Important Data



A Toy Example

- Two training data in R^1:

- What is the separating hyperplane ?

Primal Problem

- $\mathbf{x}_1 = 0, \mathbf{x}_2 = 1$ with $\mathbf{y} = [-1, 1]^T$.
- Primal problem

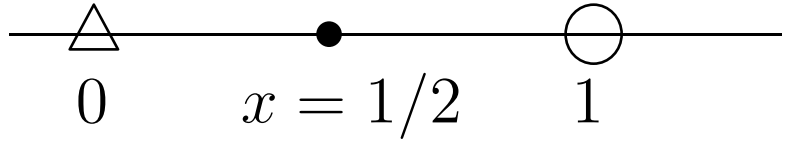
$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}w^2 \\ \text{subject to} \quad & w \cdot 1 + b \geq 1, \qquad (1) \\ & -1(w \cdot 0 + b) \geq 1. \qquad (2) \end{aligned}$$

- $-b \geq 1$ and $w \geq 1 - b \geq 2$.
- For any (w, b) satisfying two inequality constraints

$$w \geq 2$$

- We are minimizing $\frac{1}{2}w^2$

  The smallest possibility is $w = 2$.
- $(w, b) = (2, -1)$ is the optimal solution.
- The separating hyperplane $2x - 1 = 0$
  In the middle of the two training data:



Dual Problem

- Formula derived before

$$\min_{\boldsymbol{\alpha} \in R^l} \quad \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j - \sum_{i=1}^{l} \alpha_i$$

$$\text{subject to} \quad \alpha_i \geq 0, i = 1, \ldots, l, \text{ and } \sum_{i=1}^{l} \alpha_i y_i = 0.$$

- Get the objective function

$$\mathbf{x}_1^T \mathbf{x}_1 = 0, \mathbf{x}_1^T \mathbf{x}_2 = 0$$

$$\mathbf{x}_2^T \mathbf{x}_1 = 0, \mathbf{x}_2^T \mathbf{x}_2 = 1$$

- Objective function

$$\begin{aligned} & \frac{1}{2}\alpha_1^2 - (\alpha_1 + \alpha_2) \\ = \quad & \frac{1}{2}\begin{bmatrix}\alpha_1 & \alpha_2\end{bmatrix}\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\alpha_1\\ \alpha_2\end{bmatrix} - \begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}\alpha_1\\ \alpha_2\end{bmatrix}. \end{aligned}$$

- Constraints

$$\alpha_1 - \alpha_2 = 0, 0 \leq \alpha_1, 0 \leq \alpha_2.$$

- $\alpha_2 = \alpha_1$ to the objective function,

$$\frac{1}{2}\alpha_1^2 - 2\alpha_2$$

- Smallest value at $\alpha_1 = 2$.

  α_2 as well
- If smallest value < 0

  clipped to 0

Let Us Try A Practical Example

- A problem from astroparticle physics

```
1.0 1:2.617300e+01 2:5.886700e+01 3:-1.894697e-01 4:1.251225e+02
1.0 1:5.707397e+01 2:2.214040e+02 3:8.607959e-02 4:1.229114e+02
1.0 1:1.725900e+01 2:1.734360e+02 3:-1.298053e-01 4:1.250318e+02
1.0 1:2.177940e+01 2:1.249531e+02 3:1.538853e-01 4:1.527150e+02
1.0 1:9.133997e+01 2:2.935699e+02 3:1.423918e-01 4:1.605402e+02
1.0 1:5.537500e+01 2:1.792220e+02 3:1.654953e-01 4:1.112273e+02
1.0 1:2.956200e+01 2:1.913570e+02 3:9.901439e-02 4:1.034076e+02
```

- Training and testing sets available: 3,089 and 4,000
- Data format is an issue

SVM software: LIBSVM

- `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
- Now one of the most used SVM software
- Installation
- On Unix:

  Download zip file and make
- On Windows:
  - Download zip file and make
  - `c:nmake -f Makefile.win`
  - Windows binaries included in the package

Usage of LIBSVM

- Training

```
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
-t kernel_type : set type of kernel function
```

- Testing

```
Usage: svm-predict test_file model_file outpu
```

Training and Testing

- Training

```
$./svm-train train.1
......*
optimization finished, #iter = 6131
nu = 0.606144
obj = -1061.528899, rho = -0.495258
nSV = 3053, nBSV = 724
Total nSV = 3053
```

- Testing

```
$./svm-predict test.1 train.1.model
  test.1.predict
Accuracy = 66.925% (2677/4000)
```

What does this Output Mean

- obj: the optimal objective value of the dual SVM
- rho: $-b$ in the decision function
- nSV and nBSV: number of support vectors and bounded support vectors

  (i.e., $\alpha_i = C$).
- nu-svm is a somewhat equivalent form of C-SVM where C is replaced by ν.

Why this Fails

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

```
$./svm-predict train.1 train.1.model o
Accuracy = 99.7734% (3082/3089)
```

- Most kernel elements:

$$K_{ij} \begin{cases} = 1 & \text{if } i = j, \\ \to 0 & \text{if } i \neq j. \end{cases}$$

Data Scaling

- Without scaling

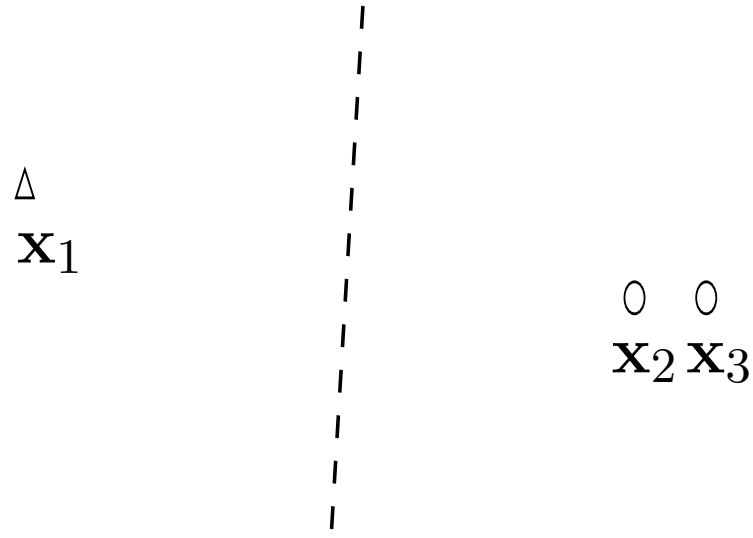
  Attributes in greater numeric ranges may dominate
- Example:

	height	sex
$\mathbf{x}_1$	150	F
$\mathbf{x}_2$	180	M
$\mathbf{x}_3$	185	M

and

$$y_1 = 0, y_2 = 1, y_3 = 1.$$

- The separating hyperplane

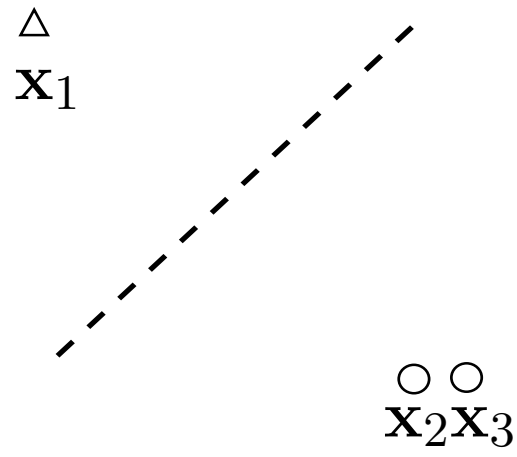


- Decision strongly depends on the first attribute
- What if the second is more important

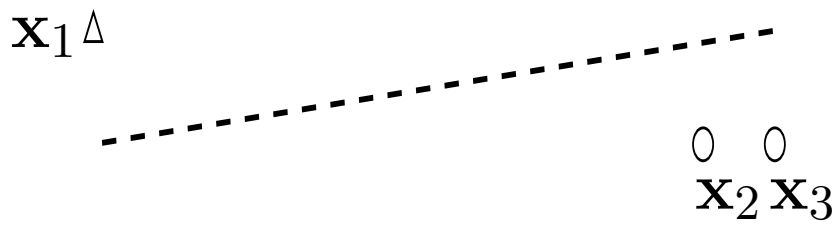
- Linearly scale the first to $[0, 1]$ by:

$$\frac{\text{1st attribute} - 150}{185 - 150},$$

- New points and separating hyperplane



- Transformed to the original space,



- The second attribute plays a role

After Data Scaling

- A common mistake

```
$./svm-scale -l -1 -u 1 train.1 > train.1.scale
$./svm-scale -l -1 -u 1 test.1 > test.1.scale
```

- Same factor on training and testing

```
$./svm-scale -s range1 train.1 > train.1.sca
$./svm-scale -r range1 test.1 > test.1.scale
$./svm-train train.1.scale
$./svm-predict test.1.scale train.1.scale.mod
  test.1.predict
 → Accuracy = 96.15%
```

- We store the scaling factor used in training and apply them for testing set

More on Training

- Train scaled data and then prediction

```
$./svm-train train.1.scale
$./svm-predict test.1.scale train.1.scale.mod
  test.1.predict
 → Accuracy = 96.15%
```

- Training accuracy now is

```
$./svm-predict train.1.scale train.1.scale.mo
Accuracy = 96.439% (2979/3089) (classificatio
```

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

Different Parameters

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

```
$./svm-train -c 20 -g 400 train.1.scale
$./svm-predict train.1.scale train.1.scale.mo
Accuracy = 100% (3089/3089) (classification)
```

- 100% training accuracy but

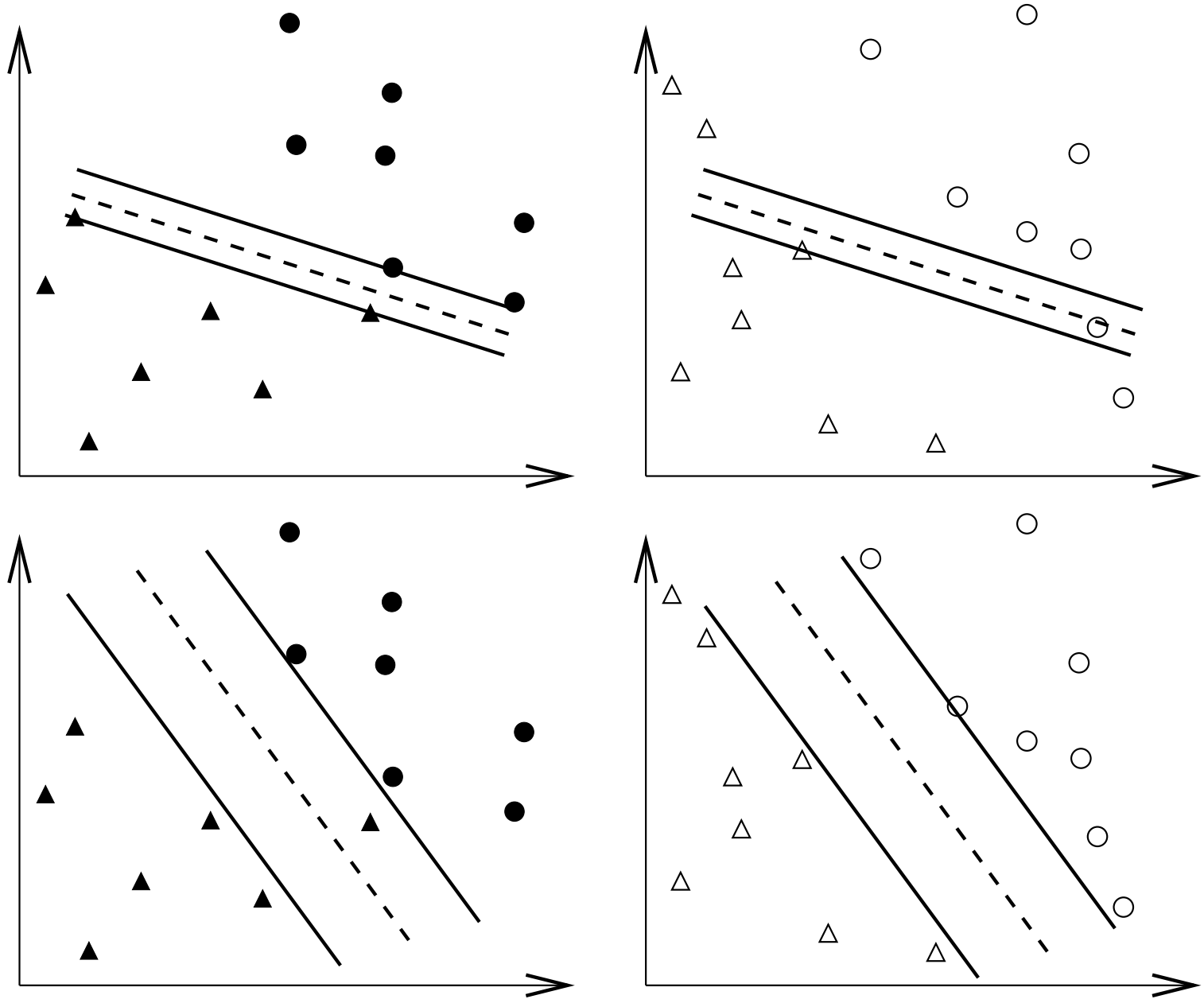
```
$./svm-predict test.1.scale train.1.scale.mod
Accuracy = 82.7% (3308/4000) (classification)
```

- Very bad test accuracy
- Overfitting happens

Overfitting and Underfitting

- When training and predicting a data, we should
  - Avoid underfitting: small training error
  - Avoid overfitting: small testing error

● and ▲: training; ○ and △: testing



Overfitting

- In theory

  You can easily achieve 100% training accuracy
- This is useless
- Surprisingly

  Many application papers did this

Parameter Selection

- Is very important
- Now parameters are

  C, 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 better ?

Performance Evaluation

- Training errors not important; only test errors count
- l training data, $\mathbf{x}_i \in R^n, y_i \in \{+1, -1\}, i = 1, \ldots, l$, a learning machine:

$$x \rightarrow f(\mathbf{x}, \alpha), f(\mathbf{x}, \alpha) = 1 \text{ or } -1.$$

Different α: different machines

- The expected test error (generalized error)

$$R(\alpha) = \int \frac{1}{2}|y - f(\mathbf{x}, \alpha)| dP(\mathbf{x}, y)$$

y: class of $\mathbf{x}$ (i.e. 1 or -1)

- $P(\mathbf{x}, y)$ unknown, empirical risk (training error):

$$R_{emp}(\alpha) = \frac{1}{2l} \sum_{i=1}^{l} |y_i - f(\mathbf{x}_i, \alpha)|$$

- $\frac{1}{2}|y_i - f(\mathbf{x}_i, \alpha)|$: loss, choose $0 \leq \eta \leq 1$, with probability at least $1 - \eta$:

$$R(\alpha) \leq R_{emp}(\alpha) + \text{ another term}$$

  - A good pattern recognition method: minimize both terms at the same time
  - $R_{emp}(\alpha) \to 0$
    another term $\to$ large

Performance Evaluation (Cont.)

- In practice

  Available data $\Rightarrow$ training and validation
- Train the training
- Test the validation
- k-fold cross validation:
  - Data randomly separated to k groups.
  - Each time $k-1$ as training and one as testing

CV and Test Accuracy

- If we select parameters so that CV is the highest,
  - Does CV represent future test accuracy ?
  - Slightly different
- If we have enough parameters, we can achieve 100% CV as well
  - e.g. more parameters than # of training data
  - But test accuracy may be different
- So
  - Available data with class labels
  - $\Rightarrow$ training, validation, testing

- Using CV on training + validation
- Predict testing with the best parameters from CV

A Simple 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 γ
4. Use the best C and γ to train the whole training set
5. Test

- Best C and γ by training $k-1$ and the whole ?

  In theory, a minor difference

  No problem in practice

Parameter Selection Procedure in LIBSVM

- grid search + CV

```
$./grid.py train.1 train.1.scale
[local] -1 -7 85.1408  (best c=0.5, g=0.0078125, rate=85.1408)
[local] 5 -7 95.4354  (best c=32.0, g=0.0078125, rate=95.4354)
.
.
.
```

- grid.py: a python script in the python directory of LIBSVM

Easy parallelization on a cluster

```
$./grid.py train.1 train.1.scale
[linux1] -1 -7 85.1408  (best c=0.5, g=0.0078125, rate=85.1408)
[linux7] 5 -7 95.4354  (best c=32.0, g=0.0078125, rate=95.4354)
.
.
.
```

Parallel Parameter Selection

- Specify machine names in grid.py

```
telnet_workers = []
ssh_workers = ['linux1','linux1','linux2',
'linux3']
nr_local_worker = 1
```

  linux1: more powerful or two CPUs

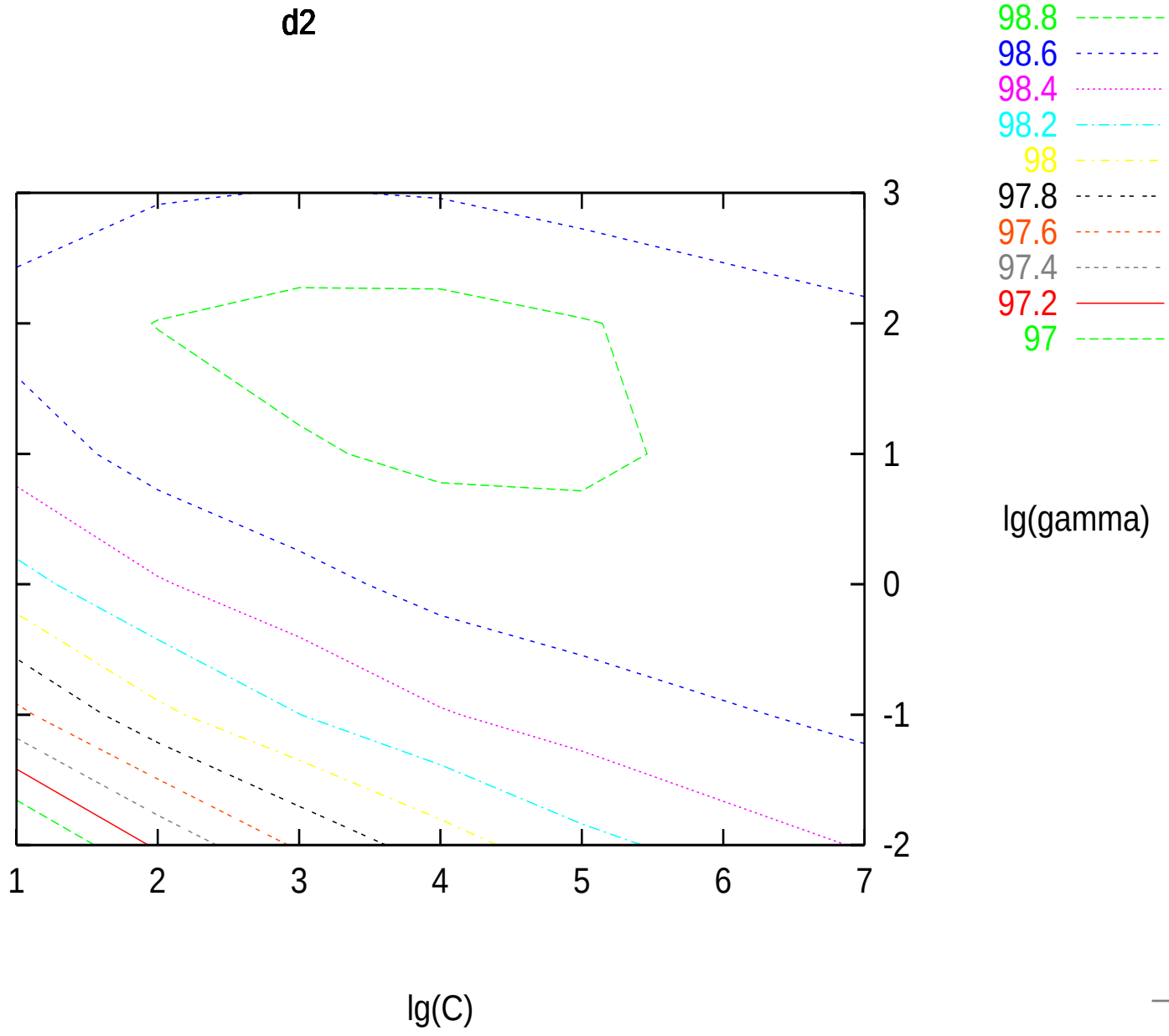
- A simple centralized control

  Load balancing not a problem

- We can use other tools

  Too simple so not consider them

Contour of Parameter Selection



Simple script in LIBSVM

- easy.py: a script for dummies

```
$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)
```

Example: Engine Misfire Detection

Problem Description

- First problem of IJCNN Challenge 2001, data from Ford
- Given time series length $T = 50,000$
- The kth data

$$x_1(k), x_2(k), x_3(k), x_4(k), x_5(k), y(k)$$

- $y(k) = \pm 1$: output, affected only by $x_1(k), \ldots, x_4(k)$
- $x_5(k) = 1$, kth data considered for evaluating accuracy
- 50,000 training data, 100,000 testing data (in two sets)

- Past and future information may affect $y(k)$
- $x_1(k)$: periodically nine 0s, one 1, nine 0s, one 1, and so on.
- Example:

```
0.000000 -0.999991 0.169769 0.000000 1.000000
0.000000 -0.659538 0.169769 0.000292 1.000000
0.000000 -0.660738 0.169128 -0.020372 1.00000
1.000000 -0.660307 0.169128 0.007305 1.000000
0.000000 -0.660159 0.169525 0.002519 1.000000
0.000000 -0.659091 0.169525 0.018198 1.000000
0.000000 -0.660532 0.169525 -0.024526 1.00000
0.000000 -0.659798 0.169525 0.012458 1.000000
```

- $x_4(k)$ more important

Background: Engine Misfire Detection

- How engine works

  Air-fuel mixture injected to cylinder

  intact, compression, combustion, exhaustion
- Engine misfire: a substantial fraction of a cylinder's air-fuel mixture fails to ignite
- Frequent misfires: pollutants and costly replacement
- On-board detection:

  Engine crankshaft rational dynamics with a position sensor
- Training data: from some expensive experimental environment

Encoding Schemes

- For SVM: each data is a vector
- $x_1(k)$: periodically nine 0s, one 1, nine 0s, one 1, ...
  - 10 binary attributes
    $x_1(k-5), \ldots, x_1(k+4)$ for the kth data
  - $x_1(k)$: an integer in 1 to 10
  - Which one is better
  - We think 10 binaries better for SVM
- $x_4(k)$ more important

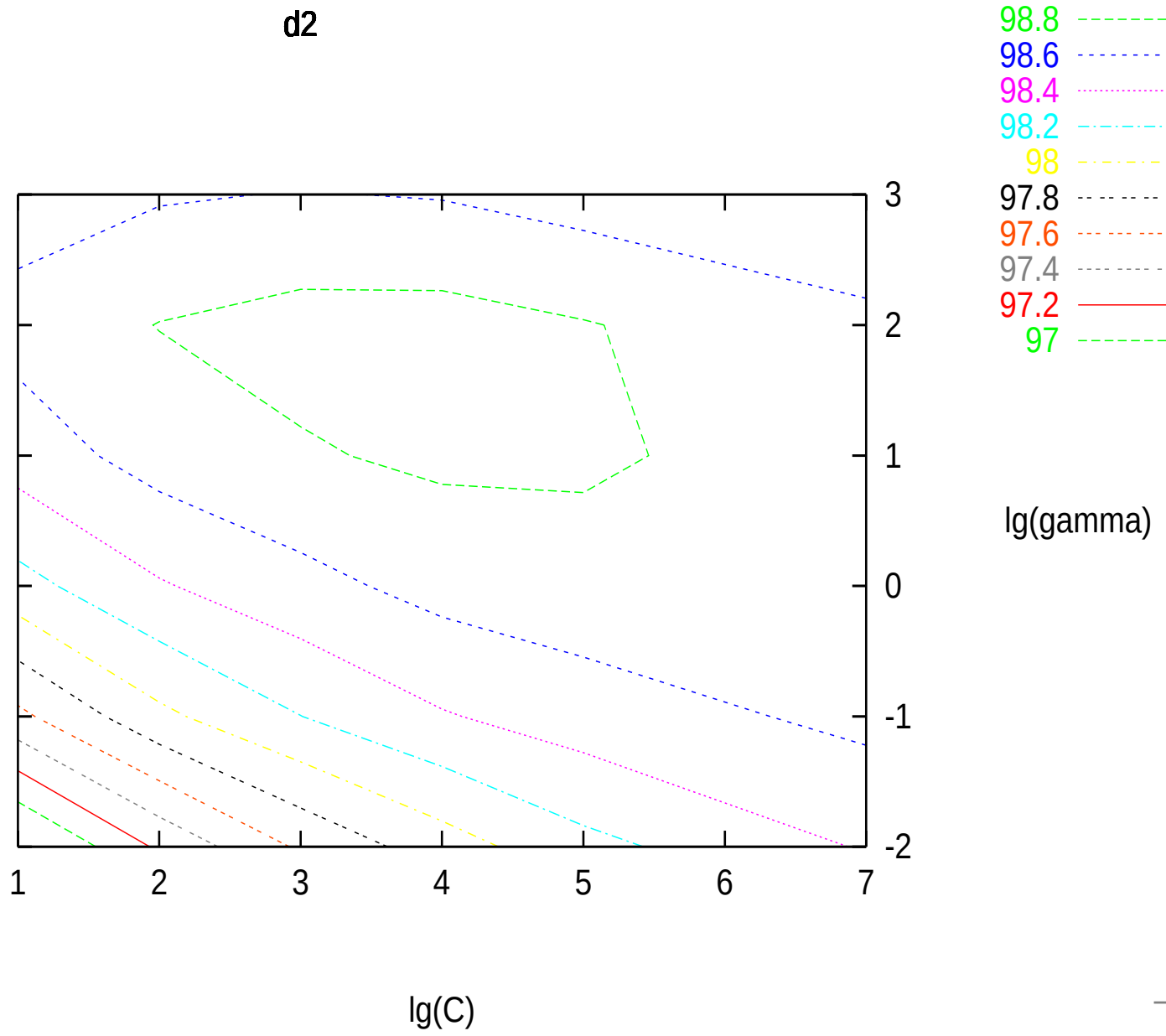
  Including $x_4(k-5), \ldots, x_4(k+4)$ for the kth data
- Each training data: 22 attributes

Training SVM

- Selecting parameters; generating a good model for prediction
- RBF kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}$
- Two parameters: γ and C
- Five-fold cross validation on 50,000 data

  Data randomly separated to five groups.

  Each time four as training and one as testing
- Use $C = 2^4, \gamma = 2^2$ and train 50,000 data for the final model


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
1
2
3
4
5
6
7
lg(gamma)
lg(C)

- Test set 1: 656 errors, Test set 2: 637 errors
- About 3000 support vectors of 50,000 training data

  A good case for SVM
- This is just the outline. There are other details.
- It is essential to do model selection.

Conclusions

- Dealing with data is interesting

  especially if you get good accuracy
- Some basic understandings are essential when applying methods

  e.g. the importance of validation
- No method is the best for all data

  Deep understanding of one or two methods very helpful