Support Vector Machines for Data Classification and Regression

Chih-Jen Lin

Department of Computer Science
National Taiwan University

Talk at Academia Sinica, April 30 and May 7, 2004

Outline

- Support vector classification
- Two practical example
- Support vector regression
- 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

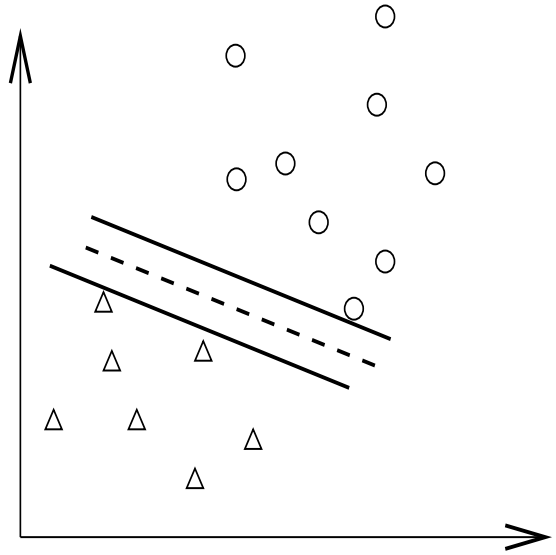
- Existing methods:
  Nearest neighbor, Neural networks, decision trees.
- SVM: a new one
- In my opinion, after careful data pre-processing
  Appropriately use NN or SVM $\Rightarrow$ similar accuracy
- But, users may not use them properly
- The chance of SVM
  - Easier for users to appropriately use it
  - The ambition: replacing NN on some applications

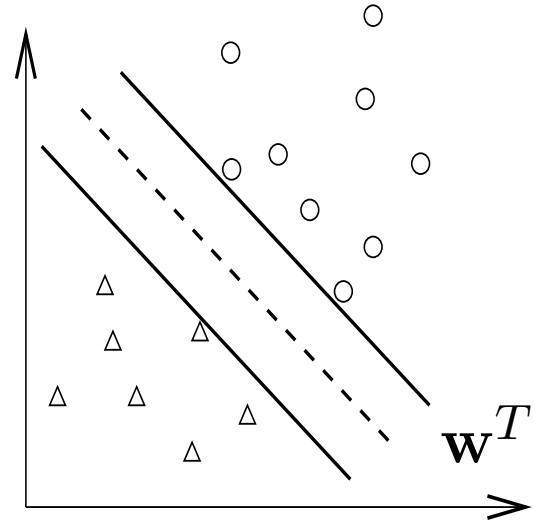
Support Vector Classification

- 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 $\mathbf{w}^T\mathbf{x} + b = \pm 1$

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

- 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

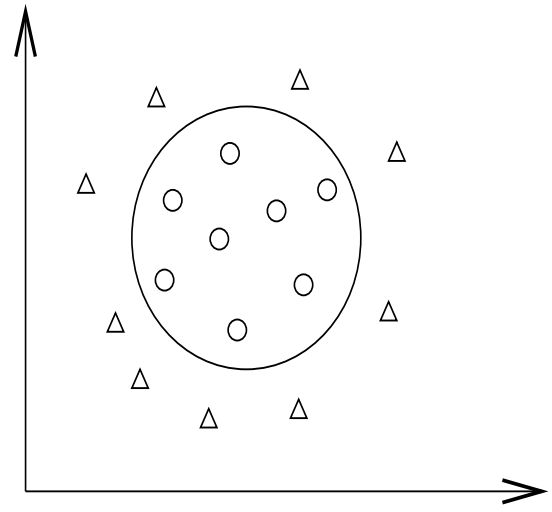
- Earlier we tried to find a linear separating hyperplane
  Data may not be linear separable
- Non-separable case: allow training errors

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

$$y_i((\mathbf{w}^T\mathbf{x}_i) + b) \geq 1 - \xi_i,$$
$$\xi_i \geq 0,\ i = 1, \ldots, l$$

- $\xi_i > 1$, $\mathbf{x}_i$ 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: $\mathbf{x}_i \in R^3, \phi(\mathbf{x}_i) \in R^{10}$

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

  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

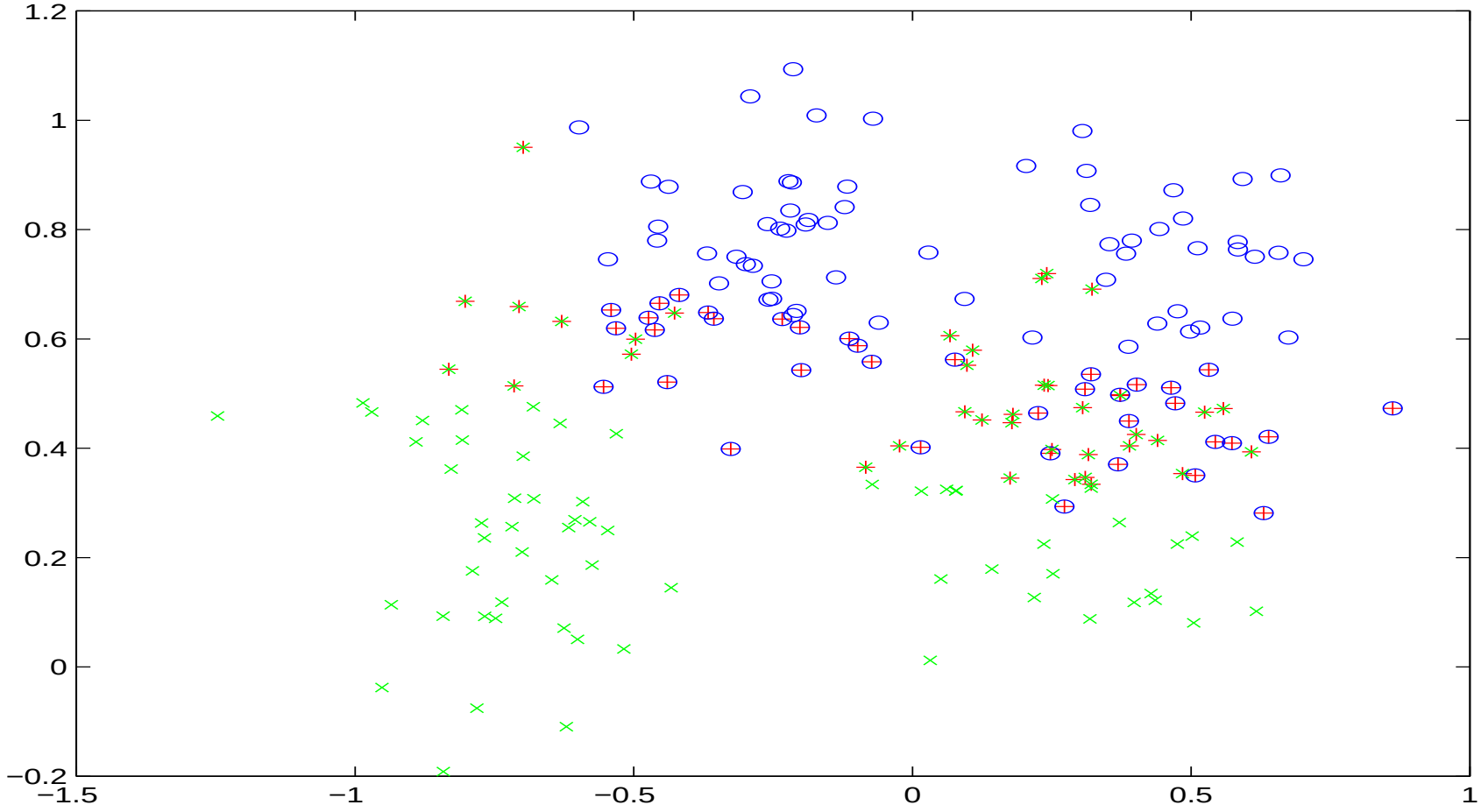
$$\begin{aligned} & \mathbf{w}^T \phi(\mathbf{x}) + b \\ = & \sum_{i=1}^{l} \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + b \\ = & \sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \end{aligned}$$

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



Let Us Try An 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 outp
```

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:

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

Data Scaling

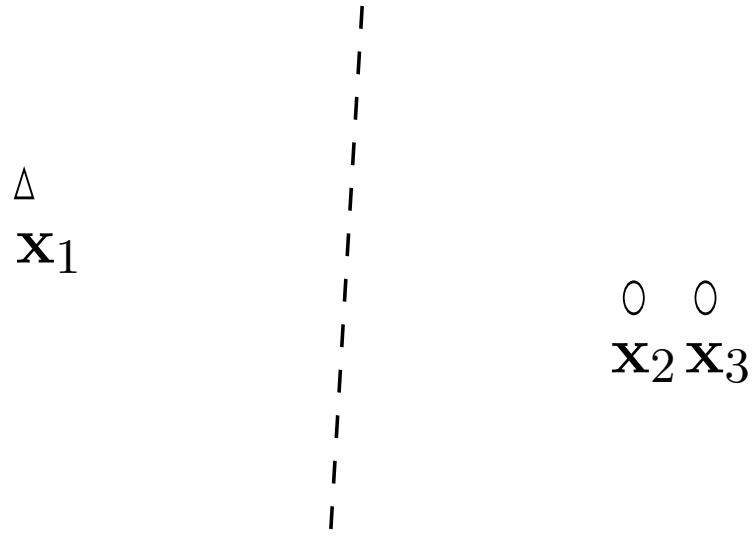
- Without scaling
  Attributes in greater numeric ranges may dominate
- Example:

	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

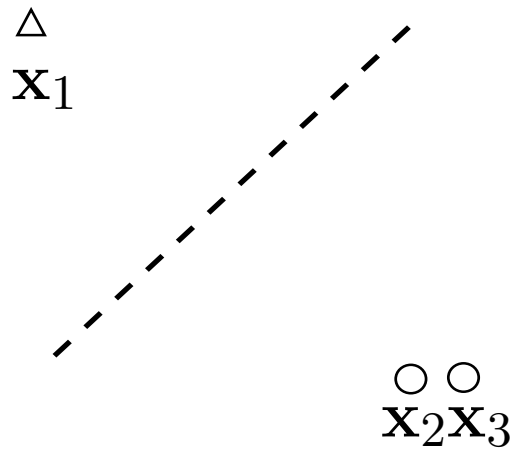


- Decision strongly depends on the first attribute
- What if the second is more important

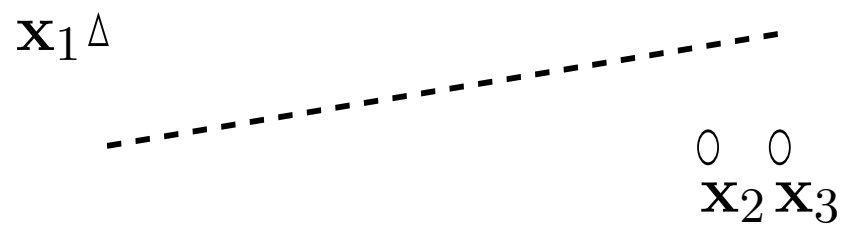
- Linearly scale the first to $[0, 1]$ by:

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

- 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.mo
  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.mo
  test.1.predict
 → Accuracy = 96.15%
```

- Training accuracy now is

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

- 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

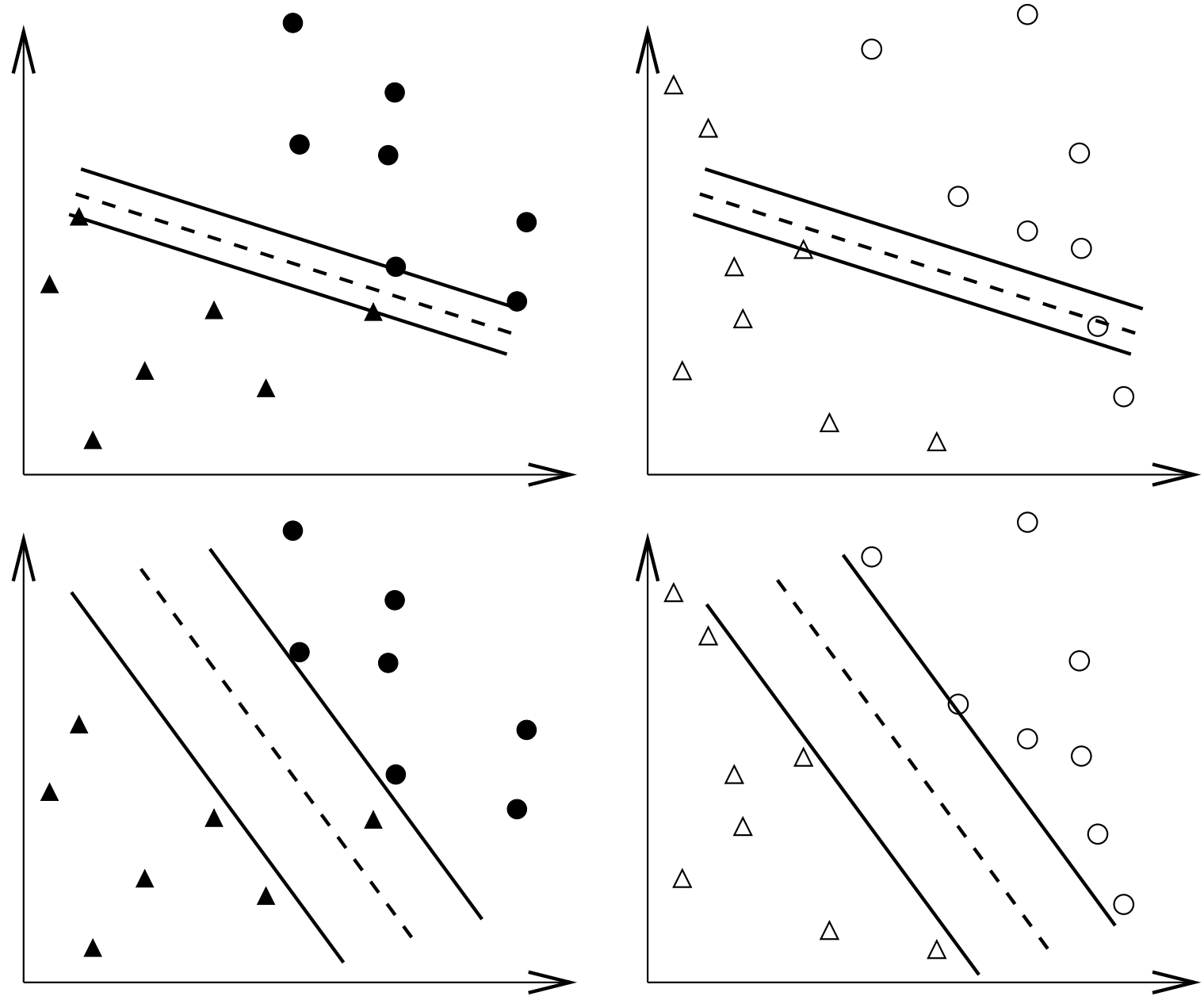
```
$./svm-predict test.1.scale train.1.scale.mo
Accuracy = 82.7% (3308/4000) (classification
```

- 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

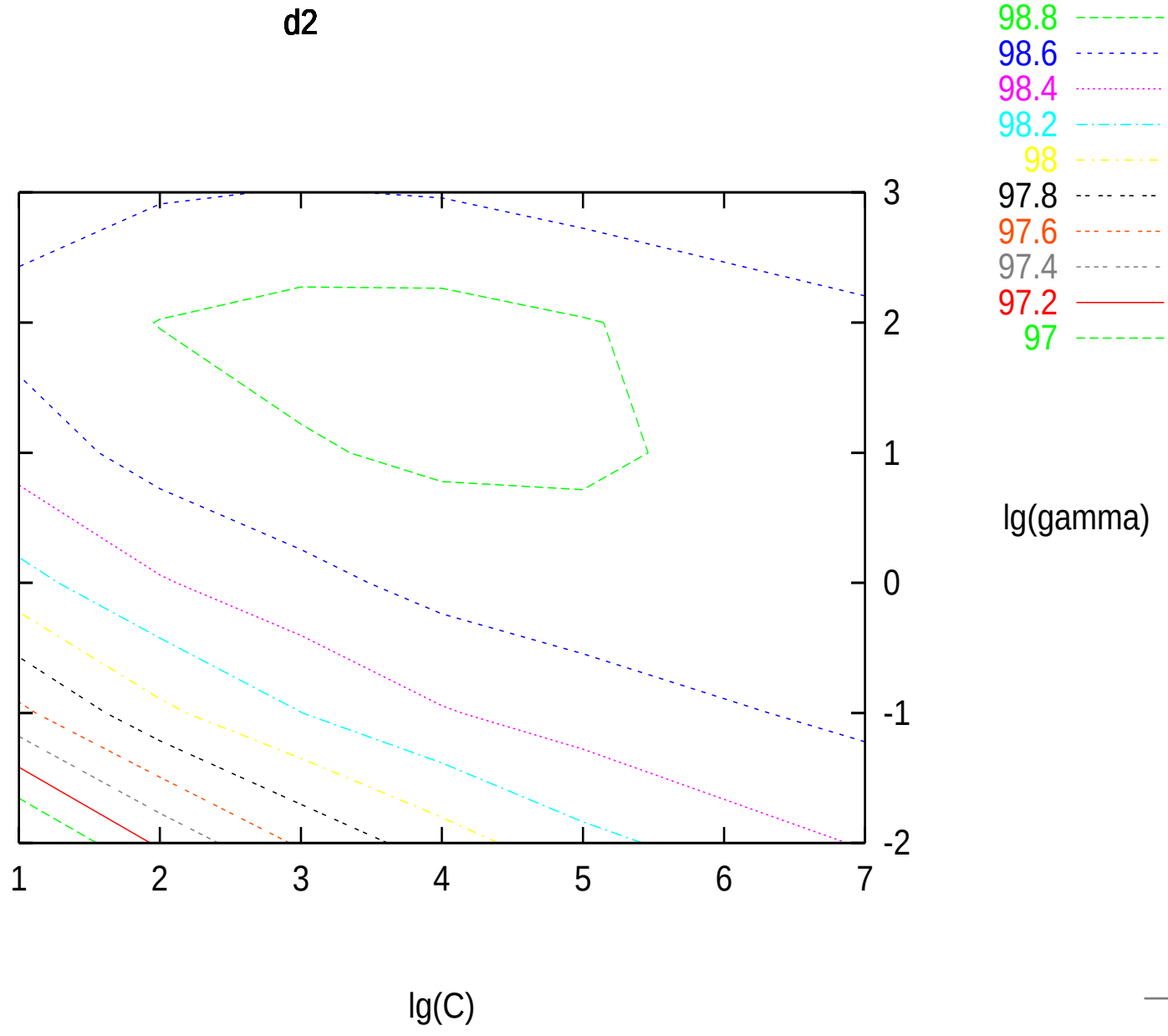
- A simple centralized control

  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

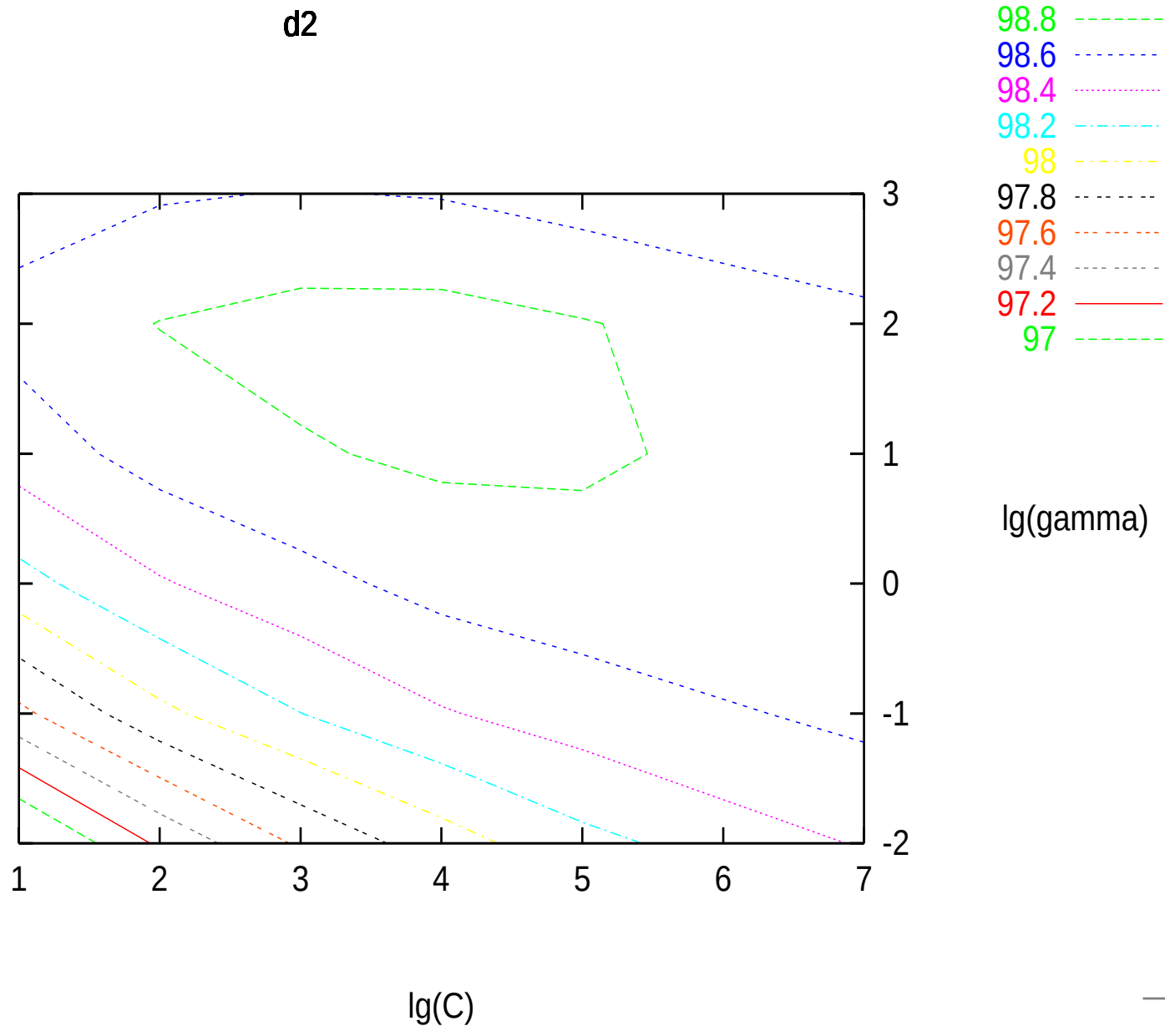
  Including $x_4(k-5), \ldots, x_4(k+4)$ for the kth data
- Each training data: 22 attributes

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.

SVM for Vehicle Classification

Machine Learning Is Sometimes An Art

- But not a science
- For complicated problems, there is no real systematic procedure
- Some tricks + domain knowledge can largely help

An Example: Vehicle Classification

- Vehicle classification in distributed sensor networks
  - `http://www.ece.wisc.edu/~sensit` and `http://mmsp-2.caenn.wisc.edu/events.zip`
  - Prepared by Duarte and Hu in University of Wisconsin
- Three classes of data:

  two vehicles and noise
- Each instance: acoustic and seismic features
  - # features of each part: 50 and 50
  - # data: 98528

- Distribution of data:

#class 1	#class 2	#class 3
1	1	2

How Data Are Generated

- Wireless distributed sensor networks (WDSN)

  Several sensors in a field
- Event extraction

  Only information when the vehicle is close enough to the sensor
- Then a time series

  FFT-based features
- Noise: high-energy factors such as wind and radio chatter.

Sample instances: Acoustic Data

2 1:-1.8893190e-02 2:-7.2501253e-03 3:-9.3349372e-03 4:8.2397278e-02 5:1.0000000e+00 6:2.8431799e-02 7:-3.9595759e-03 8:-2.2467102e-02 9:-2.7549071e-03 10:-2.2973921e-02 11:-2.4513591e-02 12:-2.7172349e-02 13:-2.2274419e-02 14:-1.8458129e-02 15:-2.6647322e-02 16:-2.6252666e-02 17:-2.2212002e-02 18:-2.5001779e-02 19:-2.6927617e-02 20:-2.7374419e-02 21:-2.7112618e-02 22:-2.4502704e-02 23:-2.5475226e-02 24:-2.5618921e-02 25:-2.6852989e-02 26:-2.5735666e-02 27:-2.7456095e-02 28:-2.7803905e-02 29:-2.6621734e-02 30:-2.4935499e-02 31:-2.7729578e-02 32:-2.6718499e-02 33:-1.9738297e-02 34:-2.2609663e-02 35:-2.3814977e-02 36:-2.6252692e-02 37:-2.4909885e-02 38:-2.5807719e-02 39:-2.4148006e-02 40:-2.5490619e-02 41:-2.7913212e-02 42:-2.7597027e-02 43:-2.5268295e-02 44:-2.7936994e-02 45:-2.7851349e-02 46:-2.7829329e-02 47:-2.7685600e-02 48:-2.5771240e-02 49:-2.5038023e-02 50:-2.4134665e-02

Results from the Authors

- Paper available from `http://www.ece.wisc.edu/~sensit/publications/`
- Three-fold CV Accuracy

Method	Acoustic	Seismic
k-nearest neighbor	69.36%	56.24%
Maximal likelihood	68.95%	62.81%
SVM	69.48%	63.79%

- We think more investigation may improve the accuracy
- So I decided to let students do a project on this

- A report presented in my statistical learning theory course
- By C.-C. Chou, S.-T. Wang, R.-E. Fan, C.-W. Lin, and C.-C. Lin
- Accuracy improved to 87%

Authors' Approach

- Data split to three folds
- Two as training and one as validation
- Average of three validation accuracy reported
- Polynomial kernel used

$$(1 + \mathbf{x}_i^T \mathbf{x}_j)^T$$

$C = 1$

- No parameter selection

My Students' Approach

- Cross-validation is a biased estimate
- Too many parameters: CV accuracy overfitted

  Practically ok for two/three parameters
- We do a more formal way

  98528 $\Rightarrow$ 4/5 training and validation, 1/5 testing

Kernel/Parameter Selection

- RBF kernel

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

- Parameter selection very important

  C and γ

  Fewer than polynomial kernel
- Huge training time
- 10% of the 4/5 training data for cross-validation

- Issue: best (C, γ) for 10% may not be the best for the whole
- In theory C should be decreased a bit

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

- Need further investigation

Results

- Test accuracy ($\log_2 C, \log_2 \gamma$)

Acoustic	Seismic
75.01 (7,-2)	72.03 (18,-10)

- Not very good
- Try to combine two features
- New accuracy 83.70 (9,-6)
- This case:

  Combining features seems to provide more information

Data Scaling

- Earlier we mention the importance of data scaling
- How about this data set ?
- Each attribute in a suitable range ?
- First 4 attributes of training/validation:

```
X1                  X2                  X3                  X4
Min.:-0.5988 Min.:-0.5194 Min.:-0.4806 Min.:-0.51
Mean: 0.1319 Mean: 0.2481 Mean: 0.1512 Mean: 0.18
Max.: 1.0000 Max.: 1.0000 Max.: 1.0000 Max.: 1.00
```

- Other features similar

Data Scaling (Cont.)

- From the authors' original matlab code: $\mathbf{x} \in R^n$:

$$x_i \leftarrow \frac{x_i}{\max_j(|x_j|)}$$

- Instance-wise scaling
- Earlier: feature-wise scaling
- First 4 features scaled to $[-1, 1]$

```
X1                  X2                  X3                  X4
Min.:-1.0000 Min.:-1.0000 Min.:-0.9999 Min.:-1.00
Mean:-0.8570 Mean:-0.7580 Mean:-0.8389 Mean:-0.83
Max.: 0.9703 Max.: 1.0000 Max.: 0.8968 Max.: 1.00
```

- Other features similar
- $\max$ of X1 < 1 as

  we scale all and the above: only 4/5
- Very different distributions
- How attributes scaled to $[-1, 1]$:

$$\frac{x_i - \min}{\max - \min} \times 2 - 1$$

- In original data, most x_i close to $\min$
- After instance-wise scaling, may not be that close to the new $\min$
- Instance-wise scaling may not be that appropriate

After Scaling

- New results

Acoustic	Seismic	Combined
79.71 (6,-2)	76.68 (6,-2)	87.18 (5,-3)

- Compare to earlier results

Acoustic	Seismic	Combined
75.01 (7,-2)	72.03 (18,-10)	83.70 (9,-6)

- New results consistently better
- Feature-wise scaling seems more appropriate
- Six data sets available at

`www.csie.ntu.edu.tw/~cjlin/libsvmtools/t/vehicl`

Issues not Investigated Yet

- If most values close to $\min$ of the features
- are these values outliers or useful information ?
- Is 86% enough for practical use ?

  Originally
  - Assault Amphibian Vehicle (AAV)
  - Main Battle Tank (M1)
  - High Mobility Multipurpose Wheeled Vehicle (HMMWV)
  - Dragon Wagon (DW)

- So five-class problem
- Now we have only AAV, DW, and noise
- # of SVs is an issue
  - Now around 20,000 SVs
  - Can they be stored in a sensor ?
- Further improvement
  - Feature selection
  - How about other methods

Lesson from This Experiment

- No systematic way for a machine learning task
- However, some simple techniques/analysis help
- Better understanding on ML methods also helps
- Of course you need good luck

SVM Optimization Problems

SVM Primal and Dual

- Standard SVM

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

- $\mathbf{w}$: huge vector variable

  Possibly infinite variables

- Practically we solve a different but strongly related problem

- Dual problem

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

$$\text{subject to} \quad 0 \le \alpha_i \le C, \qquad i = 1,\ldots,l,$$

$$\sum_{i=1}^{l} y_i\alpha_i = 0.$$

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

  using special ϕ
- $\boldsymbol{\alpha}$: l variables; finite
- Original SVM: called primal

Primal Dual Relationship

- At optimum

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

$$\frac{1}{2}\bar{\mathbf{w}}^T\bar{\mathbf{w}} + C\sum_{i=1}^{l}\bar{\xi}_i = \mathbf{e}^T\bar{\boldsymbol{\alpha}} - \frac{1}{2}\bar{\boldsymbol{\alpha}}^T Q\bar{\boldsymbol{\alpha}}. \tag{2}$$

  where $\mathbf{e} = [1, \ldots, 1]^T$.
- Primal objective value = - Dual objective value

  LIBSVM solves dual $\Rightarrow$ negative objective value
- How does this dual come from ?

Derivation of the Dual

- We follow the description in [Bazaraa et al., 1993]
- Consider a simpler problem

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

- Its dual

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) - \sum_{i=1}^{l}\alpha_i \\ \text{subject to} \quad & 0 \leq \alpha_i, \qquad i = 1,\ldots,l, \\ & \sum_{i=1}^{l} y_i\alpha_i = 0. \end{aligned}$$

Lagrangian Dual

- Defined as

$$\max_{\boldsymbol{\alpha} \geq 0} (\min_{\mathbf{w}, b} L(\mathbf{w}, b, \boldsymbol{\alpha})), \tag{3}$$

  where

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i \left(y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b) - 1 \right). \tag{4}$$

- Minimize with respect to the *primal variables* $\mathbf{w}$ and b
- Maximize with respect to the *dual variables* α_i.
- There are different dual problems

  Lagrangian dual is one

- Assume $(\bar{\mathbf{w}}, \bar{b})$ optimal for the primal with optimal objective value $\gamma = \frac{1}{2}\|\bar{\mathbf{w}}\|^2$.
- No $(\mathbf{w}, b)$ satisfies

$$\frac{1}{2}\|\mathbf{w}\|^2 < \gamma \text{ and } y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1, \quad i = 1, \ldots, l. \tag{5}$$

- There is $\bar{\boldsymbol{\alpha}} \geq 0$ such that for all $\mathbf{w}, b$

$$\frac{1}{2}\|\mathbf{w}\|^2 - \gamma - \sum_{i=1}^{l} \bar{\alpha}_i \left(y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) - 1 \right) \geq 0. \tag{6}$$

- Quite intuitive, detailed proof omitted

- Thus

$$\max_{\boldsymbol{\alpha}\geq 0} \min_{\mathbf{w},b} L(\mathbf{w}, b, \boldsymbol{\alpha}) \geq \gamma. \tag{7}$$

i.e., for any $\boldsymbol{\alpha}$,

$$\min_{\mathbf{w},b} L(\mathbf{w}, b, \boldsymbol{\alpha}) \leq L(\bar{\mathbf{w}}, \bar{b}, \boldsymbol{\alpha}),$$

so

$$\max_{\boldsymbol{\alpha}\geq 0} \min_{\mathbf{w},b} L(\mathbf{w}, b, \boldsymbol{\alpha}) \leq \max_{\boldsymbol{\alpha}\geq 0} L(\bar{\mathbf{w}}, \bar{b}, \boldsymbol{\alpha}) = \frac{1}{2}\|\bar{\mathbf{w}}\|^2 = \gamma. \tag{8}$$

- "=" holds

Strong duality: primal and dual the same optimal objective value.

- With $\bar{\alpha}_i \geq 0$ and $y_i(\bar{\mathbf{w}}^T \phi(\mathbf{x}_i) + \bar{b}) - 1 \geq 0$,

$$\bar{\alpha}_i [y_i(\bar{\mathbf{w}}^T \phi(\mathbf{x}_i) + \bar{b}) - 1] = 0, \quad i = 1, \ldots, l,$$

Complementarity condition.

- Simplify the dual, when $\boldsymbol{\alpha}$ is fixed,

$$\min_{\mathbf{w},b} L(\mathbf{w}, b, \boldsymbol{\alpha})$$

$$= \begin{cases} -\infty & \text{if } \sum_{i=1}^{l} \alpha_i y_i \neq 0 \\ \min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{l} \alpha_i [y_i(\mathbf{w}^T \phi(\mathbf{x}_i) - 1] & \text{if } \sum_{i=1}^{l} \alpha_i y_i = 0 \end{cases}$$

(9)

- If $\sum_{i=1}^{l} \alpha_i y_i \neq 0$,
  decrease $-b \sum_{i=1}^{l} \alpha_i y_i$ in $L(\mathbf{w}, b, \boldsymbol{\alpha})$ to $-\infty$
- If $\sum_{i=1}^{l} \alpha_i y_i = 0$,
  Optimum of $\frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{l} \alpha_i [y_i(\mathbf{w}^T \phi(\mathbf{x}_i) - 1]$
  happens when

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0.$$

- Thus,

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \phi(\mathbf{x}_i). \tag{10}$$

- More details

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = \begin{bmatrix} \frac{\partial}{\partial w_1} L(\mathbf{w}, b, \boldsymbol{\alpha}) \\ \vdots \\ \frac{\partial}{\partial w_n} L(\mathbf{w}, b, \boldsymbol{\alpha}) \end{bmatrix}$$

Assume $\mathbf{w} \in R^n$

- $L(\mathbf{w}, b, \boldsymbol{\alpha})$ rewritten as

$$\frac{1}{2} \sum_{j=1}^{n} w_j^2 - \sum_{i=1}^{l} \alpha_i [y_i (\sum_{j=1}^{n} w_j \phi(\mathbf{x}_i)_j - 1]$$

- So

$$\frac{\partial}{\partial w_j} L(\mathbf{w}, b, \boldsymbol{\alpha}) = w_j - \sum_{i=1}^{l} \alpha_i y_i \phi(\mathbf{x}_i)_j = 0$$

- Note that

$$\begin{aligned} \mathbf{w}^T \mathbf{w} &= \left(\sum_{i=1}^{l} \alpha_i y_i \phi(\mathbf{x}_i) \right)^T \left(\sum_{j=1}^{l} \alpha_j y_j \phi(\mathbf{x}_j) \right) \\ &= \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \end{aligned}$$

- The dual is

$$\max_{\boldsymbol{\alpha}\geq 0} \begin{cases} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) & \text{if } \sum_{i=1}^{l} \alpha_i y_i = 0 \\ -\infty & \text{if } \sum_{i=1}^{l} \alpha_i y_i \neq 0 \end{cases}$$

- $-\infty$ definitely not maximum of the dual

  Dual optimal solution not happen when $\sum_{i=1}^{l} \alpha_i y_i \neq 0$.

- Dual simplified to

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

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

- Karush-Kuhn-Tucker (KKT) optimality conditions of the primal:

$$\bar{\alpha}_i[y_i(\bar{\mathbf{w}}^T\phi(\mathbf{x}_i)+\bar{b})-1]=0, \quad i=1,\ldots,l,$$

$$\sum_{i=1}^{l}\alpha_i y_i = 0, \alpha_i \geq 0, \forall i,$$

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

- $(\mathbf{w}, b)$ optimal $\Leftrightarrow$ if feasible and there is $\boldsymbol{\alpha}$ which satisfies KKT
- The derivation with additional variables ξ_i

  Similar

An 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

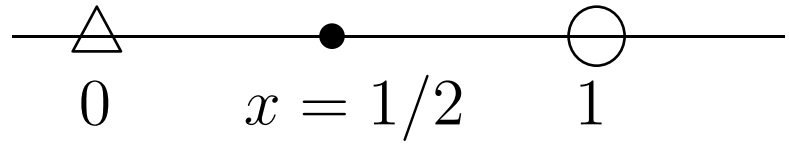
$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}w^2 \\ \text{subject to} \quad & w \cdot 1 + b \geq 1, \qquad (11) \\ & -1(w \cdot 0 + b) \geq 1. \qquad (12) \end{aligned}$$

- $-b \geq 1$ 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

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

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

- 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

Dual Problems for Other Formulas

- So we think that for any optimization problem Lagrangian dual exists
- This is wrong
- Remember we calculate

$$\min \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{l} \alpha_i[y_i(\mathbf{w}^T\phi(\mathbf{x}_i) - 1]$$

by

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0.$$

- Note that

$$f'(x) = 0 \Leftrightarrow x \text{ minimum}$$

  is wrong

- Example

$$f(x) = x^3, x = 0 \text{ not minimum}$$

- This function must satisfy certain conditions
- Some papers wrongly derived the dual of their new formulations without checking conditions

- $[2,2]^T$ satisfies constraints $0 \leq \alpha_1$ and $0 \leq \alpha_2$

  It is optimal
- Primal-dual relation

$$\begin{aligned} w &= y_1\alpha_1 x_1 + y_2\alpha_2 x_2 \\ &= 1 \cdot 2 \cdot 1 + (-1) \cdot 2 \cdot 0 \\ &= 2 \end{aligned}$$

- The same as solving the primal

Multi-class Classification

- k classes
- One-against-all: Train k binary SVMs:

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

- k decision functions

$$(\mathbf{w}^1)^T\phi(\mathbf{x}) + b_1$$

$$\vdots$$

$$(\mathbf{w}^k)^T\phi(\mathbf{x}) + b_k$$

- Select the index with the largest $(\mathbf{w}^j)^T \phi(\mathbf{x}) + b_j$

Multi-class Classification (Cont.)

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

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

  Select the one with the largest vote
- This is the method used by LIBSVM
- Try a 4-class problem

  6 binary SVMs

```
$libsvm-2.5/svm-train bsvm-2.05/vehicle.scale
optimization finished, #iter = 173
obj = -142.552559, rho = 0.748453
nSV = 194, nBSV = 183
optimization finished, #iter = 330
obj = -149.912202, rho = -0.786410
nSV = 227, nBSV = 217
optimization finished, #iter = 169
obj = -139.655613, rho = 0.998277
nSV = 186, nBSV = 177
optimization finished, #iter = 268
obj = -185.161735, rho = -0.674739
nSV = 253, nBSV = 244
optimization finished, #iter = 477
obj = -378.264371, rho = 0.177314
nSV = 405, nBSV = 394
optimization finished, #iter = 337
obj = -186.182860, rho = 1.104943
nSV = 261, nBSV = 247
Total nSV = 739
```

- There are many other methods

  A comparison in [Hsu and Lin, 2002]
- For a software

  We select one which is generally good but not always the best
- Finally I chose 1 vs. 1

  Similar accuracy to others

  Shortest training

  A bit longer on testing than 1 vs. all

Why Shorter Training Time

- 1 vs. 1

  $k(k-1)/2$ problems, each $2l/k$ data on average
- 1 vs. all

  k problems, each l data
- If solving the optimization problem:

  polynomial of the size with degree d
- Their complexities

$$\frac{k(k-1)}{2} O\left(\left(\frac{2l}{k}\right)^d\right) \text{ vs. } kO(l^d)$$

SVM Regression

Outline

- Support vector regression (SVR)
- Practical examples
- Discussion

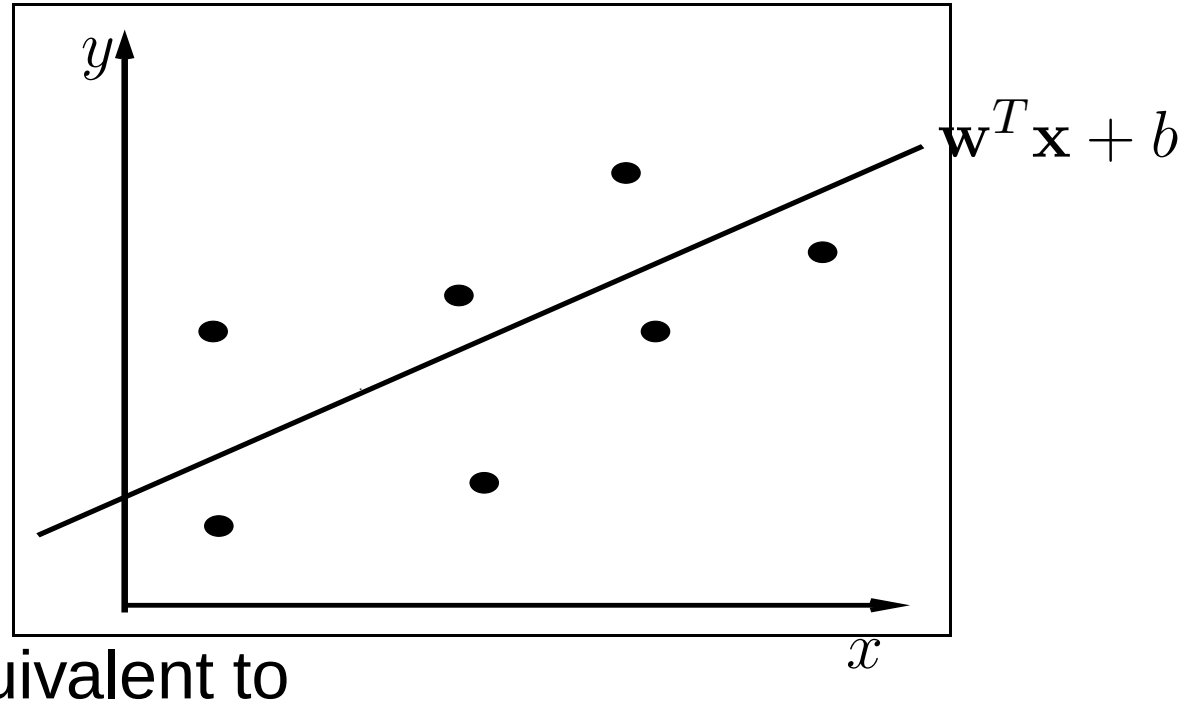
Support Vector Regression (SVR)

- Support vector machines: a new method for data classification and prediction
- Given training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$
- Regression: find a function so that

$$f(\mathbf{x}_i) \approx y_i$$

- Least square regression:

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



- This is equivalent to

$$\min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*} \quad \sum_{i=1}^{l} \xi_i^2 + (\xi_i^*)^2$$

$$\text{subject to} \quad -\xi_i^* \le y_i - (\mathbf{w}^T\mathbf{x}_i + b) \le \xi_i,$$

$$\xi_i \ge 0, \xi_i^* \ge 0, i = 1, \ldots, l.$$

- A quadratic programming problem
- L1-norm regression

$$\min_{\mathbf{w},b} \sum_{i=1}^{l} |y_i - (\mathbf{w}^T \mathbf{x}_i + b)|$$

or

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

- A linear programming problem
- This is equivalent to

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

- C: a constant

- Linear support vector regression

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

- ϵ-insensitive loss function

- A tube

$$-\epsilon \leq y - (\mathbf{w}^T\mathbf{x} + b) \leq \epsilon$$

  Data in the tube considered no error

  Most training data in the tube

- $\frac{1}{2}\mathbf{w}^T\mathbf{w}$: regularization, $\mathbf{w}^T\mathbf{x} + b$ more smooth

  Similar to the classification case

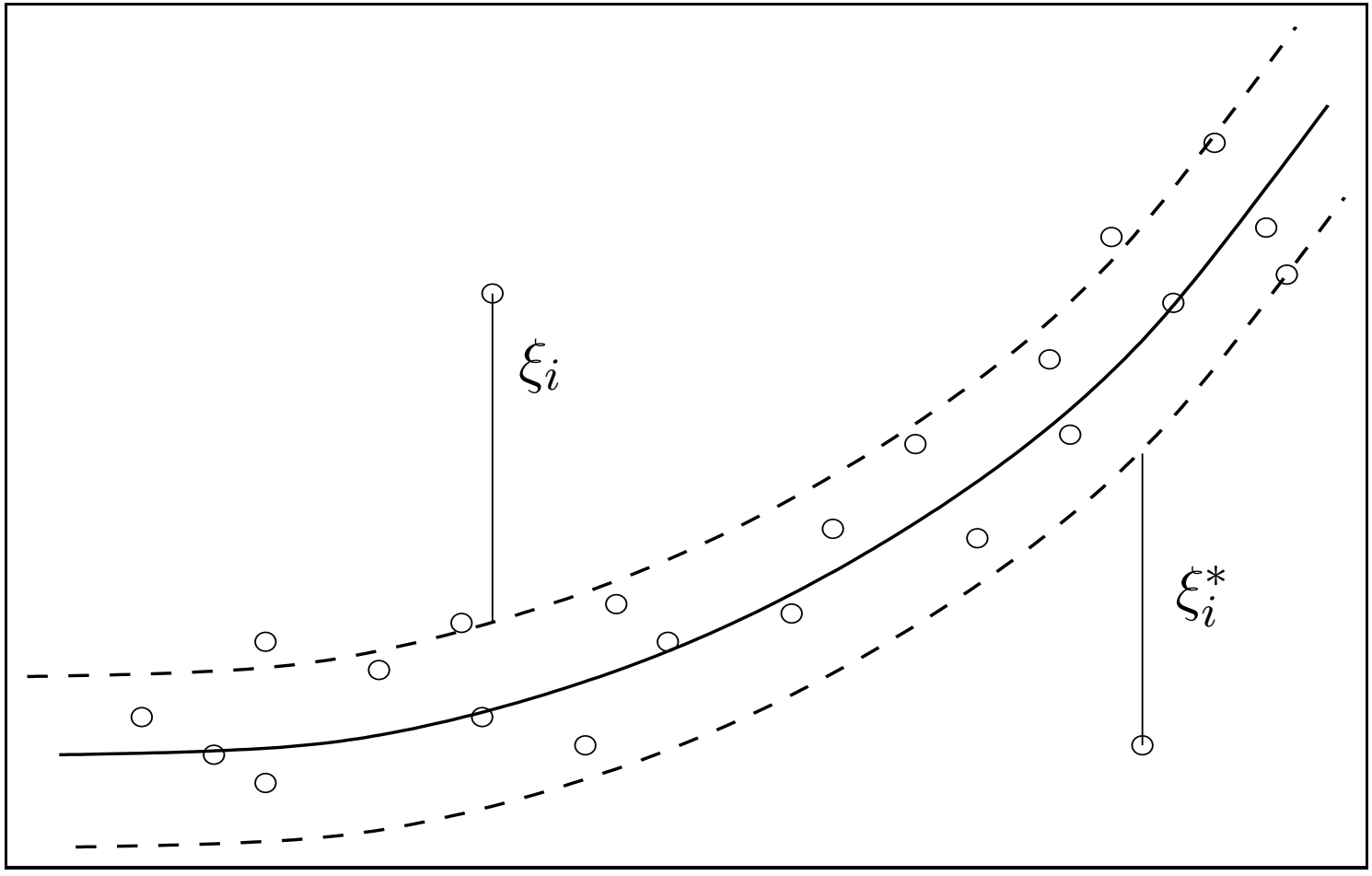
- General support vector regression:

  Data mapped to a higher space by $\phi(\mathbf{x})$

- The new approximation function

$$\mathbf{w}^T\phi(\mathbf{x}) + b$$

$$\mathbf{w}^T\phi(\mathbf{x}) + b = \begin{bmatrix} \epsilon \\ 0 \\ -\epsilon \end{bmatrix}$$



- Regression in a higher dimensional space

- Standard SVR

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

$$-\epsilon - \xi_i^* \le y_i - (\mathbf{w}^T\phi(\mathbf{x}_i) + b) \le \epsilon + \xi_i,$$

$$\xi_i, \xi_i^* \ge 0, i = 1, \ldots, l.$$

- Data in high dimensional spaces

  Possible $\mathbf{w}^T\phi(\mathbf{x}_i) + b = y_i, i = 1, \ldots, l$

  $\Rightarrow$ overfitting
- This is like in theory 100% training accuracy in classification

- Good regression methods: balance between overfitting and underfitting

  $\min \frac{1}{2}\mathbf{w}^T\mathbf{w}$: avoid overfitting

  $\min \sum_{i=1}^{l}(\xi_i + \xi_i^*)$: avoid underfitting

Support vector regression using LIBSVM

- Using the option `-s 3`

```
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
```

- Check a regression data

```
$head -n 5 svrprob/trans/abalone.scale.shuffle.
13 1:-1 2:0.310811 3:0.193277 4:-0.707965 5:-0.
12 1:-1 2:0.797297 3:0.764706 4:-0.637168 5:0.8
9 1:-1 2:-0.121622 3:-0.159664 4:-0.769912 5:-0
8 1:-1 2:0.189189 3:0.142857 4:-0.761062 5:-0.2
19 1:1 2:0.202703 3:0.12605 4:-0.752212 5:-0.42
```

- Additional parameter ϵ

```
$svm-train -s 3 -c 64 -g 0.25 -p 0.5 svrprob
.....................................*...*
optimization finished, #iter = 40152
nu = 0.710465
obj = -54155.806305, rho = -11.610859
nSV = 592, nBSV = 547
```

- Test

```
$svm-predict svrprob/trans/abalone.scale.shu
Accuracy = 0% (0/200) (classification)
Mean squared error = 5.00931 (regression)
Squared correlation coefficient = 0.58387 (r
```

- Test data

```
$head -n 5 svrprob/trans/abalone.scale.shuff
8 1:1 2:0.0945946 3:0.0420168 4:-0.823009 5:
20 2:0.756757 3:0.747899 4:-0.690265 5:0.662
7 1:1 2:0.175676 3:0.142857 4:-0.769912 5:-0
8 1:-1 2:0.189189 3:0.12605 4:-0.752212 5:-0
13 1:-1 2:0.608108 3:0.529412 4:-0.681416 5:
```

- Check

```
$head -n 5 o
7.62474
14.7385
8.40741
7.13157
9.74578
```

SVR Performance Evaluation

- Not accuracy any more
- MSE (Mean Square Error):

$$\sum_{i=1}^{l}(y_i - \hat{y}_i)^2$$

- Squared correlation coefficient (also called r^2)

$$\frac{\sum_{i=1}^{l}(y_i - \bar{y})^2(\hat{y}_i - \bar{\hat{y}})^2}{\sum_{i=1}^{l}(y_i - \bar{y})^2 \sum_{i=1}^{l}(\hat{y}_i - \bar{\hat{y}})^2}$$

- $\bar{y}$: mean of $y_i, i = 1, \ldots, l$

  $0 \le r^2 \le 1$: close to 1 $\Rightarrow$ better

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