Support Vector Machines for Data Classification and Regression

Chih-Jen Lin
Department of Computer Science
National Taiwan University

Talk at Academia Sinica, April 1, 2005
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: $x_i, i = 1, \ldots, l$

- Consider a simple case with two classes:
Define a vector $y$

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

- A hyperplane which separates all data
A separating hyperplane: $w^T x + b = 0$

$(w^T x_i) + b > 0$ if $y_i = 1$

$(w^T x_i) + b < 0$ if $y_i = -1$
Decision function $f(x) = \text{sign}(w^T x + b)$, $x$: test data

Variables: $w$ and $b$: Need to know coefficients of a plane

Many possible choices of $w$ and $b$

Select $w, b$ with the maximal margin.

Maximal distance between $w^T x + b = \pm 1$

\[
\begin{align*}
(w^T x_i) + b &\geq 1 \quad \text{if } y_i = 1 \\
(w^T x_i) + b &\leq -1 \quad \text{if } y_i = -1
\end{align*}
\]
Distance between $w^T x + b = 1$ and $-1$:

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

$$\max 2/\|w\| \equiv \min w^T w / 2$$

$$\min_{w, b} \frac{1}{2} w^T w$$

subject to $y_i ((w^T x_i) + b) \geq 1,$

$i = 1, \ldots, l.$
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.

\[
\begin{align*}
\min_{w, b, \xi} & \quad \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\
y_i((w^T x_i) + b) & \geq 1 - \xi_i, \\
\xi_i & \geq 0, \ i = 1, \ldots, l
\end{align*}
\]

- \(\xi_i > 1\), \(x_i\) not on the correct side of the separating plane.
- \(C\): large penalty parameter, most \(\xi_i\) are zero.
Nonlinear case: linear separable in other spaces?

Higher dimensional (maybe infinite) feature space

\[ \phi(x) = (\phi_1(x), \phi_2(x), \ldots). \]
Example: $x \in R^3, \phi(x) \in R^{10}$

\[
\phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, x_1^2, x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3)
\]

A standard problem [Cortes and Vapnik, 1995]:

\[
\begin{align*}
\min_{w, b, \xi} & \quad \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\
\text{subject to} & \quad y_i (w^T \phi(x_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, l.
\end{align*}
\]
Finding the Decision Function

- \( w \): a vector in a high dimensional space \( \Rightarrow \) maybe infinite variables

- The dual problem

\[
\min_{\alpha} \quad \frac{1}{2} \alpha^T Q \alpha - \mathbf{e}^T \alpha \\
\text{subject to} \quad 0 \leq \alpha_i \leq C, i = 1, \ldots, l \\
y^T \alpha = 0,
\]

where \( Q_{ij} = y_i y_j \phi(x_i)^T \phi(x_j) \) and \( \mathbf{e} = [1, \ldots, 1]^T \)

\[
w = \sum_{i=1}^{l} \alpha_i y_i \phi(x_i)
\]

A finite problem:
#variables = #training data

\[ Q_{ij} = y_i y_j \phi(x_i)^T \phi(x_j) \] needs a closed form

Efficient calculation of high dimensional inner products

Kernel trick, \( K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \)
Example: \( x_i \in \mathbb{R}^3, \phi(x_i) \in \mathbb{R}^{10} \)

\[
\phi(x_i) = (1, \sqrt{2}(x_i)_1, \sqrt{2}(x_i)_2, \sqrt{2}(x_i)_3, (x_i)_1^2, (x_i)_2^2, (x_i)_3^2, \sqrt{2}(x_i)_1(x_i)_2, \sqrt{2}(x_i)_1(x_i)_3, \sqrt{2}(x_i)_2(x_i)_3),
\]

Then \( \phi(x_i)^T \phi(x_j) = (1 + x_i^T x_j)^2 \).

Popular methods: \( K(x_i, x_j) = \)

\[
e^{-\gamma \|x_i - x_j\|^2}, \text{ (Radial Basis Function)}
\]

\[
(x_i^T x_j / a + b)^d \text{ (Polynomial kernel)}
\]
**Kernel Tricks**

- Kernel: \( K(x, y) = \phi(x)^T \phi(y) \)
- No need to explicitly know \( \phi(x) \)
- Common kernels \( K(x_i, x_j) = \)
  
  \[ e^{-\gamma \|x_i - x_j\|^2}, \text{ (Radial Basis Function)} \]
  
  \[ (x_i^T x_j / a + b)^d \text{ (Polynomial kernel)} \]

- They can be inner product in infinite dimensional space
- Assume \( x \in \mathbb{R}^1 \) and \( \gamma > 0 \).
\[ e^{-\gamma \|x_i - x_j\|^2} = e^{-\gamma (x_i - x_j)^2} = e^{-\gamma x_i^2 + 2\gamma x_i x_j - \gamma x_j^2} \]
\[ = e^{-\gamma x_i^2 - \gamma x_j^2} \left( 1 + \frac{2\gamma x_i x_j}{1!} + \frac{(2\gamma x_i x_j)^2}{2!} + \frac{(2\gamma x_i x_j)^3}{3!} + \cdots \right) \]
\[ = e^{-\gamma x_i^2 - \gamma x_j^2} \left( 1 \cdot 1 + \sqrt{\frac{2\gamma}{1!}} x_i \cdot \sqrt{\frac{2\gamma}{1!}} x_j + \sqrt{\frac{(2\gamma)^2}{2!}} x_i^2 \cdot \sqrt{\frac{(2\gamma)^2}{2!}} x_j^2 \right. \]
\[ \left. + \sqrt{\frac{(2\gamma)^3}{3!}} x_i^3 \cdot \sqrt{\frac{(2\gamma)^3}{3!}} x_j^3 + \cdots \right) \]
\[ = \phi(x_i)^T \phi(x_j), \]

where

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

- \( w \): maybe an infinite vector
- At optimum

\[
\begin{align*}
  w &= \sum_{i=1}^{l} \alpha_i y_i \phi(x_i) \\

\end{align*}
\]

Decision function

\[
\begin{align*}
  w^T \phi(x) + b &= \sum_{i=1}^{l} \alpha_i y_i \phi(x_i)^T \phi(x) + b \\
  &= \sum_{i=1}^{l} \alpha_i y_i K(x_i, x) + b
\end{align*}
\]

No need to have \( w \)
> 0: 1st class, < 0: 2nd class

Only $\phi(x_i)$ of $\alpha_i > 0$ used

$\alpha_i > 0 \Rightarrow$ support vectors
Support Vectors: More Important Data
A Toy Example

- Two training data in \( \mathbb{R}^1 \):

\[
\begin{array}{c}
\triangle & \bigcirc \\
0 & 1
\end{array}
\]

- What is the separating hyperplane?
Primal Problem

- $x_1 = 0, x_2 = 1$ with $y = [-1, 1]^T$.
- Primal problem

$$\min_{w,b} \quad \frac{1}{2}w^2$$

subject to

$$w \cdot 1 + b \geq 1,$$

$$-1(w \cdot 0 + b) \geq 1.$$
- $b \geq 1$ 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

\[
\min_{\alpha \in \mathbb{R}^l} \quad \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) - \sum_{i=1}^{l} \alpha_i
\]

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

- Get the objective function

\[
x_1^T x_1 = 0, \quad x_1^T x_2 = 0
\]
\[
x_2^T x_1 = 0, \quad x_2^T x_2 = 1
\]
Objective function

\[
\frac{1}{2} \alpha_1^2 - (\alpha_1 + \alpha_2)
\]

\[
= \frac{1}{2} \begin{bmatrix} \alpha_1 & \alpha_2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} - \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}.
\]

Constraints

\[
\alpha_1 - \alpha_2 = 0, \ 0 \leq \alpha_1, \ 0 \leq \alpha_2.
\]
\[
\alpha_2 = \alpha_1 \text{ to the objective function,}
\]

\[
\frac{1}{2} \alpha_1^2 - 2\alpha_2
\]

- Smallest value at \( \alpha_1 = 2 \).
- \( \alpha_2 \) as well
- If smallest value < 0
  clipped to 0
Let Us Try A Practical Example

A problem from astroparticle physics

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](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

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 (default 0)

Testing

Usage: svm-predict test_file model_file output_file
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 $\nu$. 
Why this Fails

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

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

Accuracy = 99.7734% (3082/3089)

- Most kernel elements:

\[
K_{ij} \begin{cases} 
1 & \text{if } i = j, \\
0 & \text{if } i \neq j.
\end{cases}
\]
Without scaling
Attributes in greater numeric ranges may dominate

Example:

<table>
<thead>
<tr>
<th>height</th>
<th>sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>x₁</td>
<td>150</td>
</tr>
<tr>
<td>x₂</td>
<td>180</td>
</tr>
<tr>
<td>x₃</td>
<td>185</td>
</tr>
</tbody>
</table>

and

\[ y₁ = 0, y₂ = 1, y₃ = 1. \]
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:

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

New points and separating hyperplane

\[
\begin{align*}
\Delta \\
x_1
\end{align*}
\]
Transformed to the original space,

\[ x_1 \Delta \]

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

$./svm-scale -s rangel train.1 > train.1.scale$
$./svm-scale -r rangel test.1 > test.1.scale$
$./svm-train train.1.scale$
$./svm-predict test.1.scale train.1.scale.model$
  test.1.predict
→ Accuracy = 96.15%

We store the scaling factor used in training and apply them for testing set
More on Training

- Train scaled data and then prediction

  $./svm-train train.1.scale$
  $./svm-predict test.1.scale train.1.scale.model$
  test.1.predict
  $\rightarrow$ Accuracy = 96.15%

- Training accuracy now is

  $./svm-predict train.1.scale train.1.scale.model$
  Accuracy = 96.439% (2979/3089) (classification)

- 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.model
Accuracy = 100% (3089/3089) (classification)
```

100% training accuracy but

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

Very bad test accuracy

Overfitting happens
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:

$$\gamma \text{ of } e^{-\gamma \|x_i - x_j\|^2}$$

$$a, b, d \text{ of } (x_i^T x_j / a + b)^d$$

- How to select them?
- So performance better?
Performance Evaluation

- Training errors not important; only test errors count

- $l$ training data, $x_i \in \mathbb{R}^n$, $y_i \in \{+1, -1\}$, $i = 1, \ldots, l$, a learning machine:

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

  Different $\alpha$: different machines

  - The expected test error (generalized error)

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

  $y$: class of $x$ (i.e. 1 or -1)
\[ P(x, y) \text{ unknown, empirical risk (training error)}: \]

\[ R_{\text{emp}}(\alpha) = \frac{1}{2l} \sum_{i=1}^{l} |y_i - f(x_i, \alpha)| \]

\[ \frac{1}{2} |y_i - f(x_i, \alpha)| : \text{loss, choose } 0 \leq \eta \leq 1, \text{ with probability at least } 1 - \eta: \]

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

- A good pattern recognition method: minimize both terms at the same time

\[ R_{\text{emp}}(\alpha) \rightarrow 0 \]

- another term \( \rightarrow \text{large} \)
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
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

⇒ 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 $\gamma$
4. Use the best $C$ and $\gamma$ to train the whole training set
5. Test

Best $C$ and $\gamma$ by training $k-1$ and the whole?
In theory, a minor difference

No problem in practice
Parameter Selection Procedure in LIBSVM

- grid search + CV

```bash
./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

```python
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

\[ d^2 \]

\[ \text{lg(C)} \]

\[ \text{lg(\gamma)} \]
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 $k$th 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 $x_1(k), \ldots, x_4(k)$
- $x_5(k) = 1$, $k$th 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:

\[
\begin{array}{cccccc}
0.000000 & -0.999991 & 0.169769 & 0.000000 & 1.000000 \\
0.000000 & -0.659538 & 0.169769 & 0.000292 & 1.000000 \\
0.000000 & -0.660738 & 0.169128 & -0.020372 & 1.000000 \\
1.000000 & -0.660307 & 0.169128 & 0.007305 & 1.000000 \\
0.000000 & -0.660159 & 0.169525 & 0.002519 & 1.000000 \\
0.000000 & -0.659091 & 0.169525 & 0.018198 & 1.000000 \\
0.000000 & -0.660532 & 0.169525 & -0.024526 & 1.000000 \\
0.000000 & -0.659798 & 0.169525 & 0.012458 & 1.000000 \\
\end{array}
\]

$x_4(k)$ 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 \( k \)th 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 \( k \)th data

Each training data: 22 attributes
Training SVM

- Selecting parameters; generating a good model for prediction
- RBF kernel \( K(x_i, x_j) = \phi(x_i)^T \phi(x_j) = e^{-\gamma \|x_i - x_j\|^2} \)
- Two parameters: \( \gamma \) 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
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.
Dual Problems for Other Formulas

So we think that for any optimization problem Lagrangian dual exists

This is wrong

Remember we calculate

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

by

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

\[ f'(x) = 0 \iff x \text{ minimum} \]

is wrong

Example

\[ f(x) = x^3, \quad x = 0 \text{ not minimum} \]

This function must satisfy certain conditions

Some papers wrongly derived the dual of their new formulations without checking conditions
[2, 2]^T satisfies constraints $0 \leq \alpha_1 \text{ and } 0 \leq \alpha_2$

It is optimal

Primal-dual relation

$$w = y_1 \alpha_1 x_1 + y_2 \alpha_2 x_2$$

$$= 1 \cdot 2 \cdot 1 + (-1) \cdot 2 \cdot 0$$

$$= 2$$

The same as solving the primal
Multi-class Classification

- \( k \) classes
- One-against-all: Train \( k \) binary SVMs:

1st class vs. \((2 - k)\)th class
2nd class vs. \((1, 3 - k)\)th class

\[ \begin{align*}
(w^1)^T \phi(x) & + b_1 \\
\vdots \\
(w^k)^T \phi(x) & + b_k
\end{align*} \]

- \( k \) decision functions
Select the index with the largest \((w^j)^T \phi(x) + b_j\)
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
$\text{libsvm-2.5/svm-train bsvm-2.05/vehicle.scale}$

optimization finished, \#\text{iter} = 173
\n\text{obj} = -142.552559, \ \text{rho} = 0.748453
\n\text{nSV} = 194, \ \text{nBSV} = 183

optimization finished, \#\text{iter} = 330
\n\text{obj} = -149.912202, \ \text{rho} = -0.786410
\n\text{nSV} = 227, \ \text{nBSV} = 217

optimization finished, \#\text{iter} = 169
\n\text{obj} = -139.655613, \ \text{rho} = 0.998277
\n\text{nSV} = 186, \ \text{nBSV} = 177

optimization finished, \#\text{iter} = 268
\n\text{obj} = -185.161735, \ \text{rho} = -0.674739
\n\text{nSV} = 253, \ \text{nBSV} = 244

optimization finished, \#\text{iter} = 477
\n\text{obj} = -378.264371, \ \text{rho} = 0.177314
\n\text{nSV} = 405, \ \text{nBSV} = 394

optimization finished, \#\text{iter} = 337
\n\text{obj} = -186.182860, \ \text{rho} = 1.104943
\n\text{nSV} = 261, \ \text{nBSV} = 247

Total \text{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
\( \frac{k(k - 1)}{2} \) problems, each \( \frac{2l}{k} \) 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

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