Support Vector Classification and Regression

Chih-Jen Lin
Department of Computer Science
National Taiwan University

Short course at ITRI, 2016
Outline

1. Introduction
2. SVM and kernel methods
3. Dual problem and solving optimization problems
4. Regularization and linear versus kernel
5. Multi-class classification
6. Support vector regression
7. SVM for clustering
8. Practical use of support vector classification
9. A practical example of SVR
Outline

1. Introduction
2. SVM and kernel methods
3. Dual problem and solving optimization problems
4. Regularization and linear versus kernel
5. Multi-class classification
6. Support vector regression
7. SVM for clustering
8. Practical use of support vector classification
9. A practical example of SVR
10. Discussion and conclusions
Last year I gave a four-day short course on “introduction of data mining”
In that course, SVM was discussed
This year I received a request to specifically talk about SVM
So I assume that some of you would like to learn more details of SVM
Therefore, this short course will be more technical than last year.

More mathematics will be involved.

We will have breaks at 9:50, 10:50, 13:50, and 14:50.

Course slides:

www.csie.ntu.edu.tw/~cjlin/talks/itri.pdf

I may still update slides (e.g., if we find errors in our lectures).
Support Vector Classification

- **Training vectors**: \( x_i, i = 1, \ldots, l \)
- Feature vectors. For example, a patient = \([\text{height}, \text{weight}, \ldots]\)^T
- Consider a simple case with **two classes**: Define an **indicator vector** \( y \in \mathbb{R}^l \)

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

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

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

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

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

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

- Distance between $\mathbf{w}^T \mathbf{x} + b = 1$ and $-1$:
  
  $$2/\|\mathbf{w}\| = 2/\sqrt{\mathbf{w}^T \mathbf{w}}$$

- A quadratic programming problem (Boser et al., 1992)

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

- Given two training data in $R^1$ as in the following figure:

  $\Delta$ 0 1

  What is the separating hyperplane?

- Now two data are $x_1 = 1, x_2 = 0$ with

  $y = [+1, -1]^T$
Example (Cont’d)

Now \( w \in \mathbb{R}^1 \). The optimization problem is

\[
\begin{align*}
\min_{w, b} & \quad \frac{1}{2} w^2 \\
\text{subject to} & \quad w \cdot 1 + b \geq 1, \quad (1) \\
& \quad -1(w \cdot 0 + b) \geq 1. \quad (2)
\end{align*}
\]

• From (2), \(-b \geq 1\).
• Putting this into (1), \( w \geq 2 \).
• That is, for any \((w, b)\) satisfying (1) and (2), \( w \geq 2 \).
Example (Cont’d)

- We are minimizing $\frac{1}{2}w^2$, so the smallest possibility is $w = 2$.
- Thus, $(w, b) = (2, -1)$ is the optimal solution.
- The separating hyperplane is $2x - 1 = 0$, in the middle of the two training data:

```
\[\begin{array}{cccc}
\triangle & \bullet & \circ \\
0 & x = 1/2 & 1 \\
\end{array}\]
```
Data May Not Be Linearly Separable

- An example:

- Allow training errors
- Higher dimensional (maybe infinite) feature space

\[ \phi(x) = [\phi_1(x), \phi_2(x), \ldots]^T. \]
Standard SVM (Boser et al., 1992; Cortes and Vapnik, 1995)

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

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

\[
\phi(\mathbf{x}) = [1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, x_1^2, \\
x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3]^T
\]
Finding the Decision Function

- \( w \): maybe infinite variables
- The dual problem: finite number of variables

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

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

- At optimum

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

- A finite problem: \#variables = \#training data
**Kernel Tricks**

- \( Q_{ij} = y_i y_j \phi(x_i)^T \phi(x_j) \) needs a **closed form**
- Example: \( x_i \in \mathbb{R}^3, \phi(x_i) \in \mathbb{R}^{10} \)

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

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

- **Kernel**: \( K(x, y) = \phi(x)^T \phi(y) \); common kernels:

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

\[
(x_i^T x_j / a + b)^d \text{ (Polynomial kernel)}
\]
Can be inner product in **infinite** dimensional space

Assume $\mathbf{x} \in \mathbb{R}^1$ and $\gamma > 0$.

\[
e^{-\gamma\|\mathbf{x}_i-\mathbf{x}_j\|^2} = e^{-\gamma (\mathbf{x}_i-\mathbf{x}_j)^2} = e^{-\gamma \mathbf{x}_i^2 + 2\gamma \mathbf{x}_i \mathbf{x}_j - \gamma \mathbf{x}_j^2}
\]

\[
e^{-\gamma \mathbf{x}_i^2 - \gamma \mathbf{x}_j^2} (1 + \frac{2\gamma \mathbf{x}_i \mathbf{x}_j}{1!} + \frac{(2\gamma \mathbf{x}_i \mathbf{x}_j)^2}{2!} + \frac{(2\gamma \mathbf{x}_i \mathbf{x}_j)^3}{3!} + \cdots)
\]

\[
e^{-\gamma \mathbf{x}_i^2 - \gamma \mathbf{x}_j^2} (1 \cdot 1 + \sqrt{\frac{2\gamma}{1!}} \mathbf{x}_i \cdot \sqrt{\frac{2\gamma}{1!}} \mathbf{x}_j + \sqrt{\frac{(2\gamma)^2}{2!}} \mathbf{x}_i^2 \cdot \sqrt{\frac{(2\gamma)^2}{2!}} \mathbf{x}_j^2
\]

\[
+ \sqrt{\frac{(2\gamma)^3}{3!}} \mathbf{x}_i^3 \cdot \sqrt{\frac{(2\gamma)^3}{3!}} \mathbf{x}_j^3 + \cdots) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j),
\]

where

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

At optimum

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

Decision function

\[ w^T \phi(x) + b \]

\[ = \sum_{i=1}^{l} \alpha_i y_i \phi(x_i)^T \phi(x) + b \]

\[ = \sum_{i=1}^{l} \alpha_i y_i K(x_i, x) + b \]

Only \( \phi(x_i) \) of \( \alpha_i > 0 \) used ⇒ support vectors
Support Vectors: More Important Data

Only $\phi(x_i)$ of $\alpha_i > 0$ used $\Rightarrow$ support vectors
See more examples via SVM Toy available at libsvm web page
(http://www.csie.ntu.edu.tw/~cjlin/libsvm/)
Example: Primal-dual Relationship

If separable, primal problem does not have $\xi_i$

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

Dual problem is

$$\begin{align*}
\min_{\alpha} & \quad \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^l \alpha_i \\
\text{subject to} & \quad 0 \leq \alpha_i, \quad i = 1, \ldots, l, \\
& \quad \sum_{i=1}^l y_i \alpha_i = 0.
\end{align*}$$
Consider the earlier example:

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

Now two data are \( x_1 = 1, x_2 = 0 \) with \( y = [+1, -1]^T \)

The solution is \((w, b) = (2, -1)\)
Example: Primal-dual Relationship (Cont’d)

- The dual objective function

\[
\frac{1}{2} \begin{bmatrix} \alpha_1 & \alpha_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} - \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}
= \frac{1}{2} \alpha_1^2 - (\alpha_1 + \alpha_2)
\]

- In optimization, **objective function** means the function to be optimized
- **Constraints** are

\[
\alpha_1 - \alpha_2 = 0, \ 0 \leq \alpha_1, \ 0 \leq \alpha_2.
\]
Example: Primal-dual Relationship (Cont’d)

- Substituting $\alpha_2 = \alpha_1$ into the objective function,

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

has the smallest value at $\alpha_1 = 2$.

- Because $[2, 2]^T$ satisfies constraints

$$0 \leq \alpha_1 \text{ and } 0 \leq \alpha_2,$$

it is optimal.
Example: Primal-dual Relationship (Cont’d)

- Using the primal-dual relation

\[ w = y_1 \alpha_1 x_1 + y_2 \alpha_2 x_2 \]
\[ = 1 \cdot 2 \cdot 1 + (-1) \cdot 2 \cdot 0 \]
\[ = 2 \]

- This is the same as that by solving the primal problem.
More about Support vectors

- We know

\[ \alpha_i > 0 \Rightarrow \text{support vector} \]

- We have

\[ y_i (w^T x_i + b) < 1 \Rightarrow \alpha_i > 0 \Rightarrow \text{support vector}, \]

\[ y_i (w^T x_i + b) = 1 \Rightarrow \alpha_i \geq 0 \Rightarrow \text{maybe SV} \]

and

\[ y_i (w^T x_i + b) > 1 \Rightarrow \alpha_i = 0 \Rightarrow \text{not SV} \]
Dual problem and solving optimization problems

Outline

1. Introduction
2. SVM and kernel methods
3. Dual problem and solving optimization problems
4. Regularization and linear versus kernel
5. Multi-class classification
6. Support vector regression
7. SVM for clustering
8. Practical use of support vector classification
9. A practical example of SVR
10. Discussion and conclusions
Convex Optimization I

- Convex problems are an important class of optimization problems that possess nice properties.
- A function is convex if \( \forall x, y \)

\[
f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y), \forall \theta \in [0, 1]
\]

- That is, the line segment between any two points is not lower than the function value.
Convex Optimization II

- $f(x)$
- $f(y)$

Diagram showing the functions $f(x)$ and $f(y)$ in a Cartesian coordinate system.
A convex optimization problem takes the following form:

\[
\begin{align*}
\min & \quad f_0(w) \\
\text{subject to} & \quad f_i(w) \leq 0, \ i = 1, \ldots, m, \\
& \quad h_i(w) = 0, \ i = 1, \ldots, p,
\end{align*}
\]  

where \(f_0, \ldots, f_m\) are convex functions and \(h_1, \ldots, h_p\) are affine (i.e., a linear function):

\[
h_i(w) = a^T w + b
\]
Convex Optimization IV

- A nice property of convex optimization problems is that

\[ \inf_{\mathbf{w}} \{ f_0(\mathbf{w}) \mid \mathbf{w} \text{ satisfies constraints} \} \]

is unique

- Optimal objective value is unique, but optimal \( \mathbf{w} \) may be not

- There are other nice properties such as the primal-dual relationship that we will use

- To learn more about convex optimization, you can check the book by Boyd and Vandenberghe (2004)
Deriving the Dual

- For simplification, consider the problem without $\xi_i$

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

subject to

$$y_i(w^T \phi(x_i) + b) \geq 1, \ i = 1, \ldots, l.$$ 

- Its dual is

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha$$

subject to

$$0 \leq \alpha_i, \quad i = 1, \ldots, l,$$

$$y^T \alpha = 0,$$

where

$$Q_{ij} = y_i y_j \phi(x_i)^T \phi(x_j).$$
Lagrangian Dual I

- Lagrangian dual

\[
\max_{\alpha \geq 0} \left( \min_{w, b} L(w, b, \alpha) \right),
\]

where

\[
L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^{l} \alpha_i \left( y_i (w^T \phi(x_i) + b) - 1 \right)
\]
Lagrangian Dual II

- Strong duality

\[
\min \text{ Primal} = \max_{\alpha \geq 0} \left( \min_{w, b} L(w, b, \alpha) \right)
\]

- After SVM is popular, quite a few people think that for any optimization problem
  \[ \Rightarrow \text{Lagrangian dual exists and strong duality holds} \]

- **Wrong!** We usually need
  - The optimization problem is **convex**
  - Certain **constraint qualification** holds (details not discussed)
Lagrangian Dual III

- We have them
  SVM primal is convex and has linear constraints
Simplify the dual. When $\alpha$ is fixed,

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

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

If $\sum_{i=1}^l \alpha_i y_i \neq 0$, we can decrease

$$-b \sum_{i=1}^l \alpha_i y_i$$

in $L(w, b, \alpha)$ to $-\infty$. 

Chih-Jen Lin (National Taiwan Univ.)
If $\sum_{i=1}^{l} \alpha_i y_i = 0$, optimum of the strictly convex function

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

happens when

$$\nabla_w L(w, b, \alpha) = 0.$$  

Thus,

$$w = \sum_{i=1}^{l} \alpha_i y_i \phi(x_i).$$
- Note that

\[ \mathbf{w}^T \mathbf{w} = \left( \sum_{i=1}^{l} \alpha_i y_i \phi(x_i) \right)^T \left( \sum_{j=1}^{l} \alpha_j y_j \phi(x_j) \right) \]

\[ = \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) \]

- The dual is

\[
\max_{\alpha \geq 0} \begin{cases} 
\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) & \text{if } \sum_{i=1}^{l} \alpha_i y_i = 0, \\
-\infty & \text{if } \sum_{i=1}^{l} \alpha_i y_i \neq 0.
\end{cases}
\]
Lagrangian dual: \( \max_{\alpha \geq 0} \left( \min_{w, b} L(w, b, \alpha) \right) \)

\(-\infty\) 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

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

subject to

\[
y^T \alpha = 0,
\]

\[
\alpha_i \geq 0, i = 1, \ldots, l.
\]
Our problems may be infinite dimensional (i.e., \( \mathbf{w} \in R^\infty \))

We can still use Lagrangian duality

See a rigorous discussion in Lin (2001)
Recall the dual problem is

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

and at optimum

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

What if we put (4) into primal

$$\min_{\alpha, \xi} \frac{1}{2} \alpha^T Q \alpha + C \sum_{i=1}^{l} \xi_i$$

subject to

$$(Q \alpha + b y)_i \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

Note that

$$y_i w^T \phi(x_i) = y_i \sum_{j=1}^{l} \alpha_j y_j \phi(x_j)^T \phi(x_i)$$

$$= \sum_{j=1}^{l} Q_{ij} \alpha_j = (Q \alpha)_i$$
Primal versus Dual III

- If $Q$ is positive definite, we can prove that the optimal $\alpha$ of (5) is the same as that of the dual.
- So dual is not the only choice to obtain the model.
Dual problem and solving optimization problems

Large Dense Quadratic Programming

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

- \( Q_{ij} \neq 0 \), \( Q \): an \( l \) by \( l \) fully dense matrix
- 50,000 training points: 50,000 variables:
  \((50,000^2 \times 8/2)\) bytes = 10GB RAM to store \( Q \)
Large Dense Quadratic Programming (Cont’d)

- For quadratic programming problems, traditional optimization methods assume that $Q$ is available in the computer memory.
- They cannot be directly applied here because $Q$ cannot even be stored.
- Currently, decomposition methods (a type of coordinate descent methods) are what used in practice.
Dual problem and solving optimization problems

Decomposition Methods

- Working on some variables each time (e.g., Osuna et al., 1997; Joachims, 1998; Platt, 1998)
- Similar to coordinate-wise minimization
- Working set $B$, $N = \{1, \ldots, l\} \setminus B$ fixed
- Sub-problem at the $k$th iteration:

$$
\min_{\alpha_B} \quad \frac{1}{2} \begin{bmatrix} \alpha_B^T & (\alpha_N^k)^T \end{bmatrix} \begin{bmatrix} Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN} \end{bmatrix} \begin{bmatrix} \alpha_B \\ \alpha_N^k \end{bmatrix} - \\
\begin{bmatrix} e_B^T & (e_N^k)^T \end{bmatrix} \begin{bmatrix} \alpha_B \\ \alpha_N^k \end{bmatrix}
$$

subject to $0 \leq \alpha_t \leq C$, $t \in B$, $y_B^T \alpha_B = -y_N^T \alpha_N^k$
Avoid Memory Problems

The new objective function

\[
\frac{1}{2} \begin{bmatrix} \alpha_T \end{bmatrix}^T \begin{bmatrix} \alpha_N^k \end{bmatrix} \begin{bmatrix} Q_{BB} \alpha_B + Q_{BN} \alpha_N^k \\ Q_{NB} \alpha_B + Q_{NN} \alpha_N^k \end{bmatrix} - e_T^T \alpha_B + \text{constant} \\
\frac{1}{2} \alpha_B^T Q_{BB} \alpha_B + (-e_B + Q_{BN} \alpha_N^k)^T \alpha_B + \text{constant}
\]

Only \(|B|\) columns of \(Q\) are needed

In general \(|B| \leq 10\) is used

Calculated when used: trade time for space

But is such an approach practical?
How Decomposition Methods Perform?

- Convergence not very fast. This is known because of using only first-order information.
- But, no need to have very accurate $\alpha$

Decision function:

$$\text{sgn}(\mathbf{w}^T \phi(\mathbf{x}) + b) = \text{sgn} \left( \sum_{i=1}^{l} \alpha_i K(x_i, \mathbf{x}) + b \right)$$

Prediction may still be correct with a rough $\alpha$

- Further, in some situations,
  - $\#$ support vectors $\ll \#$ training points
  - Initial $\alpha^1 = 0$, some instances never used
How Decomposition Methods Perform? (Cont’d)

- An example of training 50,000 instances using the software LIBSVM
  
  $\texttt{svm-train} -c 16 -g 4 -m 400 22\text{features}$
  
  Total $nSV = 3370$
  
  Time 79.524s

- This was done on a typical desktop

- Calculating the whole $Q$ takes more time

- $\#SVs = 3,370 \ll 50,000$

  A good case where some remain at zero all the time
How Decomposition Methods Perform? (Cont’d)

Because many $\alpha_i = 0$ in the end, we can develop a shrinking techniques

Variables are removed during the optimization procedure. Smaller problems are solved
Machine Learning Properties are Useful in Designing Optimization Algorithms

We have seen that special properties of SVM contribute to the viability of decomposition methods:

- For machine learning applications, no need to accurately solve the optimization problem.
- Because some optimal $\alpha_i = 0$, decomposition methods may not need to update all the variables.
- Also, we can use shrinking techniques to reduce the problem size during decomposition methods.
Differences between Optimization and Machine Learning

- The two topics may have different focuses. We give the following example.
- The decomposition method we just discussed converges more slowly when $C$ is large.
- Using $C = 1$ on a data set:
  - \# iterations: 508
- Using $C = 5,000$:
  - \# iterations: 35,241
Optimization researchers may rush to solve difficult cases of large $C$

It turns out that large $C$ is less used than small $C$

Recall that SVM solves

$$\frac{1}{2}w^Tw + C(\text{sum of training losses})$$

A large $C$ means to overfit training data

This does not give good test accuracy. More details about overfitting will be discussed later
Outline

1. Introduction
2. SVM and kernel methods
3. Dual problem and solving optimization problems
4. **Regulatization and linear versus kernel**
5. Multi-class classification
6. Support vector regression
7. SVM for clustering
8. Practical use of support vector classification
9. A practical example of SVR
10. Discussion and conclusions
Equivalent Optimization Problem

- Recall SVM optimization problem is

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

- It is equivalent to

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

- The reformulation is useful to derive SVM from a different viewpoint
Equivalent Optimization Problem (Cont’d)

- That is, at optimum,

\[ \xi_i = \max(0, 1 - y_i(w^T \phi(x_i) + b)) \]

- Reason: from constraints

\[ \xi_i \geq 1 - y_i(w^T \phi(x_i) + b) \text{ and } \xi_i \geq 0 \]

but we also want to minimize \( \xi_i \)
Linear and Kernel I

- Linear classifier

\[ \text{sgn}(\mathbf{w}^T \mathbf{x} + b) \]

- Kernel classifier

\[ \text{sgn}(\mathbf{w}^T \phi(\mathbf{x}) + b) = \text{sgn} \left( \sum_{i=1}^{l} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b \right) \]

- Linear is a special case of kernel
- An important difference is that for linear we can store \( \mathbf{w} \)
For kernel, $w$ may be infinite dimensional and cannot be stored.

We will show that they are useful in different circumstances.
The Bias Term $b$

- Recall the decision function is
  \[ \text{sgn}(\mathbf{w}^T \mathbf{x} + b) \]

- Sometimes the bias term $b$ is omitted
  \[ \text{sgn}(\mathbf{w}^T \mathbf{x}) \]

- This is fine if the number of features is not too small
For classification naturally we aim to minimize the training error

$$\min_{w} \text{ (training errors)}$$

To characterize the training error, we need a loss function $\xi(w; x, y)$ for each instance $(x, y)$.

Ideally we should use 0–1 training loss:

$$\xi(w; x, y) = \begin{cases} 1 & \text{if } yw^Tx < 0, \\ 0 & \text{otherwise} \end{cases}$$
However, this function is discontinuous. The optimization problem becomes difficult

\[ \xi(w; x, y) \]

We need continuous approximations
Common Loss Functions

- **Hinge loss** (l1 loss)
  \[ \xi_{L1}(w; x, y) \equiv \max(0, 1 - yw^T x) \]  
  (6)

- **Squared hinge loss** (l2 loss)
  \[ \xi_{L2}(w; x, y) \equiv \max(0, 1 - yw^T x)^2 \]  
  (7)

- **Logistic loss**
  \[ \xi_{LR}(w; x, y) \equiv \log(1 + e^{-yw^T x}) \]  
  (8)

- **SVM**: (6)-(7). **Logistic regression** (LR): (8)
Logistic regression is very related to SVM
Their performance (i.e., test accuracy) is usually similar
Common Loss Functions (Cont’d)

- However, minimizing training losses may not give a good model for future prediction
- Overfitting occurs
Overfitting

- See the illustration in the next slide
- For classification,
  You can easily achieve 100% training accuracy
- This is useless
- When training a data set, we should
  Avoid underfitting: small training error
  Avoid overfitting: small testing error
and ▲: training; ○ and △: testing
Regularization

- To minimize the training error we manipulate the $w$ vector so that it fits the data.
- To avoid overfitting we need a way to make $w$’s values less extreme.
- One idea is to make $w$ values closer to zero.
- We can add, for example,

$$
\frac{w^Tw}{2} \quad \text{or} \quad \|w\|_1
$$

to the objective function.
General Form of Linear Classification I

- Training data \( \{y_i, x_i\}, x_i \in \mathbb{R}^n, i = 1, \ldots, l, y_i = \pm 1 \)
- \( l \): \# of data, \( n \): \# of features

\[
\min_w f(w), \quad f(w) \equiv \frac{w^T w}{2} + C \sum_{i=1}^{l} \xi(w; x_i, y_i)
\]

\( w^T w / 2 \): regularization term
\( \xi(w; x, y) \): loss function
\( C \): regularization parameter
General Form of Linear Classification II

- Of course we can map data to a higher dimensional space

\[
\min_w f(w), \quad f(w) \equiv \frac{w^T w}{2} + C \sum_{i=1}^{l} \xi(w; \phi(x_i), y_i)
\]
SVM and Logistic Regression I

- If hinge (l1) loss is used, the optimization problem is

\[
\min_w \frac{1}{2} w^T w + C \sum_{i=1}^{l} \max(0, 1 - y_i w^T x_i)
\]

It is the SVM problem we had earlier (without the bias \( b \))

- Therefore, we have derived SVM from a different viewpoint

- We also see that SVM is very related to logistic regression
SVM and Logistic Regression II

- However, many wrongly think that they are different.
- This is wrong.
- Reason of this misunderstanding: traditionally,
  - when people say SVM ⇒ kernel SVM
  - when people say logistic regression ⇒ linear logistic regression
- Indeed we can do kernel logistic regression

\[
\min_w \frac{1}{2} w^T w + C \sum_{i=1}^{l} \log(1 + e^{-y_i w^T \phi(x_i)})
\]
SVM and Logistic Regression III

- A main difference from SVM is that logistic regression has **probability interpretation**
- We will introduce logistic regression from another viewpoint
Logistic Regression

For a label-feature pair \((y, x)\), assume the probability model is

\[ p(y|x) = \frac{1}{1 + e^{-yw^T x}}. \]

Note that

\[ p(1|x) + p(-1|x) = \frac{1}{1 + e^{-w^T x}} + \frac{1}{1 + e^{w^T x}} = \frac{e^{w^T x}}{1 + e^{w^T x}} + \frac{1}{1 + e^{w^T x}} = 1 \]
Logistic Regression (Cont’d)

- Idea of this model

\[
p(1|x) = \frac{1}{1 + e^{-w^T x}} \begin{cases} 
  \rightarrow 1 & \text{if } w^T x \gg 0, \\
  \rightarrow 0 & \text{if } w^T x \ll 0 
\end{cases}
\]

- Assume training instances are

\[(y_i, x_i), i = 1, \ldots, l\]
Logistic Regression (Cont’d)

- Logistic regression finds $\mathbf{w}$ by maximizing the following likelihood

$$\max_{\mathbf{w}} \prod_{i=1}^{l} p (y_i | x_i).$$  \hspace{1cm} (10)

- Negative log-likelihood

$$- \log \prod_{i=1}^{l} p (y_i | x_i) = - \sum_{i=1}^{l} \log p (y_i | x_i)$$

$$= \sum_{i=1}^{l} \log \left( 1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i} \right)$$
Logistic Regression (Cont’d)

- Logistic regression

\[
\min_w \sum_{i=1}^{l} \log \left( 1 + e^{-y_i w^T x_i} \right).
\]

- Regularized logistic regression

\[
\min_w \frac{1}{2} w^T w + C \sum_{i=1}^{l} \log \left( 1 + e^{-y_i w^T x_i} \right). \quad (11)
\]

C: regularization parameter decided by users
Loss Functions: Differentiability

However,

\( \xi_{L1} \): not differentiable
\( \xi_{L2} \): differentiable but not twice differentiable
\( \xi_{LR} \): twice differentiable

The same optimization method may not be applicable to all these losses
Discussion

We see that the same classification method can be derived from different ways

SVM
- Maximal margin
- Regularization and training losses

LR
- Regularization and training losses
- Maximum likelihood
Regularization

- L1 versus L2
  
  \[ \|w\|_1 \text{ and } w^T w / 2 \]

- \( w^T w / 2 \): smooth, easier to optimize

- \( \|w\|_1 \): non-differentiable
  
  sparse solution; possibly many zero elements

Possible advantages of L1 regularization:

Feature selection

Less storage for \( w \)
Methods such as SVM and logistic regression can be used in two ways:

- **Kernel methods:** data mapped to a higher dimensional space

  \[ x \Rightarrow \phi(x) \]

  \[ \phi(x_i)^T \phi(x_j) \] is easily calculated; little control on \( \phi(\cdot) \)

- **Feature engineering + linear classification:**

  We have \( x \) without mapping. Alternatively, we can say that \( \phi(x) \) is our \( x \); full control on \( x \) or \( \phi(x) \)

We refer to them as **kernel and linear classifiers**
Regulation and linear versus kernel

Linear and Kernel Classification

- Let’s check the prediction cost
  \[ \mathbf{w}^T \mathbf{x} + b \text{ versus } \sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \]

- If \( K(\mathbf{x}_i, \mathbf{x}_j) \) takes \( O(n) \), then \( O(n) \) versus \( O(nl) \)

- Linear is much cheaper
- A similar difference occurs for training
In a sense, linear is a special case of kernel.

Indeed, we can prove that test accuracy of linear is the same as Gaussian (RBF) kernel under certain parameters (Keerthi and Lin, 2003).

Therefore, roughly we have

\[
\text{test accuracy: } \text{kernel} \geq \text{linear} \\
\text{cost: } \text{kernel} \gg \text{linear}
\]

Speed is the reason to use linear.
For some problems, accuracy by linear is as good as nonlinear
But training and testing are much faster
This particularly happens for document classification
Number of features (bag-of-words model) very large
Data very sparse (i.e., few non-zeros)
## Comparison Between Linear and Kernel (Training Time & Testing Accuracy)

<table>
<thead>
<tr>
<th>Data set</th>
<th>Linear</th>
<th></th>
<th>RBF Kernel</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>Accuracy</td>
<td>Time</td>
<td>Accuracy</td>
</tr>
<tr>
<td>MNIST38</td>
<td>0.1</td>
<td>96.82</td>
<td>38.1</td>
<td>99.70</td>
</tr>
<tr>
<td>ijcnn1</td>
<td>1.6</td>
<td>91.81</td>
<td>26.8</td>
<td>98.69</td>
</tr>
<tr>
<td>covtype</td>
<td>1.4</td>
<td>76.37</td>
<td>46,695.8</td>
<td>96.11</td>
</tr>
<tr>
<td>news20</td>
<td>1.1</td>
<td>96.95</td>
<td>383.2</td>
<td>96.90</td>
</tr>
<tr>
<td>real-sim</td>
<td>0.3</td>
<td>97.44</td>
<td>938.3</td>
<td>97.82</td>
</tr>
<tr>
<td>yahoo-japan</td>
<td>3.1</td>
<td>92.63</td>
<td>20,955.2</td>
<td>93.31</td>
</tr>
<tr>
<td>webspam</td>
<td>25.7</td>
<td>93.35</td>
<td>15,681.8</td>
<td>99.26</td>
</tr>
</tbody>
</table>

Size reasonably large: e.g., yahoo-japan: 140k instances and 830k features.
Comparison Between Linear and Kernel (Training Time & Testing Accuracy)

<table>
<thead>
<tr>
<th>Data set</th>
<th>Linear</th>
<th></th>
<th>RBF Kernel</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>Accuracy</td>
<td>Time</td>
<td>Accuracy</td>
</tr>
<tr>
<td>MNIST38</td>
<td>0.1</td>
<td>96.82</td>
<td>38.1</td>
<td>99.70</td>
</tr>
<tr>
<td>ijcnn1</td>
<td>1.6</td>
<td>91.81</td>
<td>26.8</td>
<td>98.69</td>
</tr>
<tr>
<td>covtype</td>
<td>1.4</td>
<td>76.37</td>
<td>46,695.8</td>
<td>96.11</td>
</tr>
<tr>
<td>news20</td>
<td>1.1</td>
<td>96.95</td>
<td>383.2</td>
<td>96.90</td>
</tr>
<tr>
<td>real-sim</td>
<td>0.3</td>
<td>97.44</td>
<td>938.3</td>
<td>97.82</td>
</tr>
<tr>
<td>yahoo-japan</td>
<td>3.1</td>
<td>92.63</td>
<td>20,955.2</td>
<td>93.31</td>
</tr>
<tr>
<td>webspam</td>
<td>25.7</td>
<td>93.35</td>
<td>15,681.8</td>
<td>99.26</td>
</tr>
</tbody>
</table>

Size reasonably large: e.g., yahoo-japan: 140k instances and 830k features
## Comparison Between Linear and Kernel (Training Time & Testing Accuracy)

<table>
<thead>
<tr>
<th>Data set</th>
<th>Linear</th>
<th>RBF Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>Accuracy</td>
</tr>
<tr>
<td>MNIST38</td>
<td>0.1</td>
<td>96.82</td>
</tr>
<tr>
<td>ijcnn1</td>
<td>1.6</td>
<td>91.81</td>
</tr>
<tr>
<td>covtype</td>
<td>1.4</td>
<td>76.37</td>
</tr>
<tr>
<td>news20</td>
<td>1.1</td>
<td>96.95</td>
</tr>
<tr>
<td>real-sim</td>
<td>0.3</td>
<td>97.44</td>
</tr>
<tr>
<td>yahoo-japan</td>
<td>3.1</td>
<td>92.63</td>
</tr>
<tr>
<td>webspam</td>
<td>25.7</td>
<td>93.35</td>
</tr>
</tbody>
</table>

Size reasonably large: e.g., yahoo-japan: 140k instances and 830k features
Extension: Training Explicit Form of Nonlinear Mappings I

Linear-SVM method to train $\phi(x_1), \ldots, \phi(x_l)$
- Kernel not used
- Applicable only if dimension of $\phi(x)$ not too large

Low-degree Polynomial Mappings

\[ K(x_i, x_j) = (x_i^T x_j + 1)^2 = \phi(x_i)^T \phi(x_j) \]

\[ \phi(x) = [1, \sqrt{2}x_1, \ldots, \sqrt{2}x_n, x_1^2, \ldots, x_n^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_{n-1}x_n]^T \]
Extension: Training Explicit Form of Nonlinear Mappings II

- For this mapping, \# features = \( O(n^2) \)
- Recall \( O(n) \) for linear versus \( O(nl) \) for kernel
- Now \( O(n^2) \) versus \( O(nl) \)
- **Sparse data**
  - \( n \rightarrow \bar{n} \), average \# non-zeros for sparse data
  - \( \bar{n} \ll n \Rightarrow O(\bar{n}^2) \) may be much smaller than \( O(l\bar{n}) \)
- When degree is small, train the explicit form of \( \phi(x) \)
### Testing Accuracy and Training Time

<table>
<thead>
<tr>
<th>Data set</th>
<th>Degree-2 Polynomial</th>
<th></th>
<th></th>
<th>Accuracy diff.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training time (s)</td>
<td>Accuracy</td>
<td>Linear</td>
<td>RBF</td>
</tr>
<tr>
<td>LIBLINEAR</td>
<td>LIBSVM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a9a</td>
<td>1.6</td>
<td>89.8</td>
<td>85.06</td>
<td>0.07</td>
</tr>
<tr>
<td>real-sim</td>
<td>59.8</td>
<td>1,220.5</td>
<td>98.00</td>
<td>0.49</td>
</tr>
<tr>
<td>ijcnn1</td>
<td>10.7</td>
<td>64.2</td>
<td>97.84</td>
<td>5.63</td>
</tr>
<tr>
<td>MNIST38</td>
<td>8.6</td>
<td>18.4</td>
<td>99.29</td>
<td>2.47</td>
</tr>
<tr>
<td>covtype</td>
<td>5,211.9</td>
<td>NA</td>
<td>80.09</td>
<td>3.74</td>
</tr>
<tr>
<td>webspam</td>
<td>3,228.1</td>
<td>NA</td>
<td>98.44</td>
<td>5.29</td>
</tr>
</tbody>
</table>

Training $\phi(x_i)$ by linear: faster than kernel, but sometimes competitive accuracy.
**Example: Dependency Parsing**

- This is an NLP Application

<table>
<thead>
<tr>
<th></th>
<th>Kernel</th>
<th></th>
<th>Linear</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RBF</td>
<td>Poly-2</td>
<td>Linear</td>
<td>Poly-2</td>
</tr>
<tr>
<td>Training time</td>
<td>3h34m53s</td>
<td>3h21m51s</td>
<td>3m36s</td>
<td>3m43s</td>
</tr>
<tr>
<td>Parsing speed</td>
<td>0.7x</td>
<td>1x</td>
<td>1652x</td>
<td>103x</td>
</tr>
<tr>
<td>UAS</td>
<td>89.92</td>
<td>91.67</td>
<td>89.11</td>
<td>91.71</td>
</tr>
<tr>
<td>LAS</td>
<td>88.55</td>
<td>90.60</td>
<td>88.07</td>
<td>90.71</td>
</tr>
</tbody>
</table>

- We get faster training/testing, while maintain good accuracy
Example: Dependency Parsing II

We achieve this by training low-degree polynomial-mapped data by linear classification.

That is, linear methods to explicitly train $\phi(x_i), \forall i$

We consider the following low-degree polynomial mapping:

$$\phi(x) = [1, x_1, \ldots, x_n, x_1^2, \ldots, x_n^2, x_1x_2, \ldots, x_{n-1}x_n]^T$$
Handing High Dimensionality of $\phi(x)$

A multi-class problem with sparse data

<table>
<thead>
<tr>
<th>$n$</th>
<th>Dim. of $\phi(x)$</th>
<th>$l$</th>
<th>$\bar{n}$</th>
<th>$w$’s # nonzeros</th>
</tr>
</thead>
<tbody>
<tr>
<td>46,155</td>
<td>1,065,165,090</td>
<td>204,582</td>
<td>13.3</td>
<td>1,438,456</td>
</tr>
</tbody>
</table>

- $\bar{n}$: average $\neq$ nonzeros per instance
- Dimensionality of $w$ is very high, but $w$ is sparse
  - Some training feature columns of $x_i x_j$ are entirely zero
- Hashing techniques are used to handle sparse $w$
Example: Classifier in a Small Device

- In a sensor application (Yu et al., 2014), the classifier can use less than 16KB of RAM

<table>
<thead>
<tr>
<th>Classifiers</th>
<th>Test accuracy</th>
<th>Model Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree</td>
<td>77.77</td>
<td>76.02KB</td>
</tr>
<tr>
<td>AdaBoost (10 trees)</td>
<td>78.84</td>
<td>1,500.54KB</td>
</tr>
<tr>
<td>SVM (RBF kernel)</td>
<td>85.33</td>
<td>1,287.15KB</td>
</tr>
</tbody>
</table>

- Number of features: 5
- We consider a degree-3 polynomial mapping

\[
\text{dimensionality} = \binom{5 + 3}{3} + \text{bias term} = 57.
\]
Example: Classifier in a Small Device

- One-against-one strategy for 5-class classification

\[
\binom{5}{2} \times 57 \times 4\text{bytes} = 2.28\text{KB}
\]

Assume single precision

- Results

<table>
<thead>
<tr>
<th>SVM method</th>
<th>Test accuracy</th>
<th>Model Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBF kernel</td>
<td>85.33</td>
<td>1,287.15KB</td>
</tr>
<tr>
<td>Polynomial kernel</td>
<td>84.79</td>
<td>2.28KB</td>
</tr>
<tr>
<td>Linear kernel</td>
<td>78.51</td>
<td>0.24KB</td>
</tr>
</tbody>
</table>
Outline

1. Introduction
2. SVM and kernel methods
3. Dual problem and solving optimization problems
4. Regularization and linear versus kernel
5. Multi-class classification
6. Support vector regression
7. SVM for clustering
8. Practical use of support vector classification
9. A practical example of SVR
10. Discussion and conclusions
Multi-class Classification

- SVM and logistic regression are methods for **two-class** classification
- We need certain ways to extend them for multi-class problems
- This is **not a problem** for methods such as nearest neighbor or decision trees
Multi-class Classification (Cont’d)

- $k$ classes
- One-against-the rest: Train $k$ binary SVMs:
  - 1st class vs. $(2, \cdots, k)$th class
  - 2nd class vs. $(1, 3, \ldots, k)$th class
  - ...
- $k$ decision functions
  \[
  (w^1)^T \phi(x) + b_1
  
  \vdots
  
  (w^k)^T \phi(x) + b_k
  \]
Multi-class classification

- Prediction:

\[
\arg \max_j (\mathbf{w}^j)^T \phi(\mathbf{x}) + b_j
\]

- Reason: If \( \mathbf{x} \in 1\text{st class} \), then we should have

\[
(\mathbf{w}^1)^T \phi(\mathbf{x}) + b_1 \geq +1
\]
\[
(\mathbf{w}^2)^T \phi(\mathbf{x}) + b_2 \leq -1
\]
\[
\vdots
\]
\[
(\mathbf{w}^k)^T \phi(\mathbf{x}) + b_k \leq -1
\]
Multi-class Classification (Cont’d)

- One-against-one: train \( k(k - 1)/2 \) binary SVMs
  \((1, 2), (1, 3), \ldots, (1, k), (2, 3), (2, 4), \ldots, (k - 1, k)\)
- If 4 classes \(\Rightarrow 6\) binary SVMs

\[
\begin{array}{ccc}
  y_i = 1 & y_i = -1 & \text{Decision functions} \\
  \text{class 1} & \text{class 2} & f^{12}(x) = (w^{12})^T x + b^{12} \\
  \text{class 1} & \text{class 3} & f^{13}(x) = (w^{13})^T x + b^{13} \\
  \text{class 1} & \text{class 4} & f^{14}(x) = (w^{14})^T x + b^{14} \\
  \text{class 2} & \text{class 3} & f^{23}(x) = (w^{23})^T x + b^{23} \\
  \text{class 2} & \text{class 4} & f^{24}(x) = (w^{24})^T x + b^{24} \\
  \text{class 3} & \text{class 4} & f^{34}(x) = (w^{34})^T x + b^{34} \\
\end{array}
\]
For a testing data, predicting all binary SVMs

<table>
<thead>
<tr>
<th>Classes</th>
<th>winner</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2</td>
<td>1</td>
</tr>
<tr>
<td>1 3</td>
<td>1</td>
</tr>
<tr>
<td>1 4</td>
<td>1</td>
</tr>
<tr>
<td>2 3</td>
<td>2</td>
</tr>
<tr>
<td>2 4</td>
<td>4</td>
</tr>
<tr>
<td>3 4</td>
<td>3</td>
</tr>
</tbody>
</table>

Select the one with the largest vote

<table>
<thead>
<tr>
<th>class</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td># votes</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

May use decision values as well
Solving a Single Problem

- An approach by Crammer and Singer (2002)

\[
\min_{w_1, \ldots, w_k} \frac{1}{2} \sum_{m=1}^{k} \|w_m\|_2^2 + C \sum_{i=1}^{l} \xi(\{w_m\}_{m=1}^{k}; x_i, y_i),
\]

where

\[
\xi(\{w_m\}_{m=1}^{k}; x, y) \equiv \max_{m \neq y} \max_m (0, 1 - (w_y - w_m)^T x).
\]

- We hope the decision value of \(x_i\) by the model \(w_{y_i}\) is larger than others
- Prediction: same as one-against-the rest

\[
\arg \max_j (w_j)^T x
\]
Other variants of solving a single optimization problem include Weston and Watkins (1999); Lee et al. (2004).

A comparison in Hsu and Lin (2002).

RBF kernel: accuracy similar for different methods.

But 1-against-1 is the fastest for training.
Maximum Entropy

- It is widely applied by NLP applications.
- Conditional probability of label $y$ given data $x$.

$$P(y|x) \equiv \frac{\exp(w_y^T x)}{\sum_{m=1}^k \exp(w_m^T x)},$$
We then minimizes regularized negative log-likelihood.

\[
\min_{w_1, \ldots, w_m} \frac{1}{2} \sum_{m=1}^{k} \| w_k \|^2 + C \sum_{i=1}^{l} \xi(\{ w_m \}_{m=1}^{k}; x_i, y_i),
\]

where

\[
\xi(\{ w_m \}_{m=1}^{k}; x, y) \equiv -\log P(y|x).
\]
Maximum Entropy (Cont’d)

- Is this loss function reasonable?
- If
  
  \[ \mathbf{w}_{y_i}^T \mathbf{x}_i \gg \mathbf{w}_m^T \mathbf{x}_i, \forall m \neq y_i, \]
  
  then
  
  \[ \xi(\{\mathbf{w}_m\}_{m=1}^k; \mathbf{x}_i, y_i) \approx 0 \]

  That is, no loss

- In contrast, if
  
  \[ \mathbf{w}_{y_i}^T \mathbf{x}_i \ll \mathbf{w}_m^T \mathbf{x}_i, m \neq y_i, \]
  
  then \( P(y_i|\mathbf{x}_i) \ll 1 \) and the loss is large.
Features as Functions

- NLP applications often use a function $f(x, y)$ to generate the feature vector

$$P(y|x) \equiv \frac{\exp(w^T f(x, y))}{\sum_{y'} \exp(w^T f(x, y'))}.$$  \hspace{1cm} (12)

- The earlier probability model is a special case by

$$f(x, y) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ x \\ \vdots \\ 0 \\ 0 \end{bmatrix} \quad y - 1 \quad \in \mathbb{R}^{nk} \quad \text{and} \quad w = \begin{bmatrix} w_1 \\ \vdots \\ w_k \end{bmatrix}.$$
Given training data \((x_1, y_1), \ldots, (x_l, y_l)\)

Now

\[ y_i \in \mathbb{R} \]

is the target value

Regression: find a function so that

\[ f(x_i) \approx y_i \]
Least Square Regression II

- Least square regression:

\[
\min_{w, b} \sum_{i=1}^{l} (y_i - (w^T x_i + b))^2
\]

- That is, we model \( f(x) \) by

\[
f(x) = w^T x + b
\]

- An example
Least Square Regression III

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{plot.png}
\caption{Plot of Height (cm) vs Weight (kg) with regression line $0.60 \cdot \text{Weight} + 130.2$}
\end{figure}

This is equivalent to

$$
\min_{w, b} \sum_{i=1}^{l} \xi(w, b; x_i, y_i)
$$

where

$$
\xi(w, b; x_i, y_i) = (y_i - (w^T x_i + b))^2
$$
Regularized Least Square

- $\xi(w, b; x_i, y_i)$ is a kind of loss function
- We can add regularization.

$$\min_{w, b} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi(w, b; x_i, y_i)$$

- $C$ is still the regularization parameter
- Other loss functions?
Support Vector Regression I

- $\epsilon$-insensitive loss function ($b$ omitted)

\[
\begin{align*}
\max(|w^T x_i - y_i| - \epsilon, 0) \\
\max(|w^T x_i - y_i| - \epsilon, 0)^2
\end{align*}
\]
Support Vector Regression II

\[ w^T x_i - y_i \]

- \( \epsilon \): errors small enough are treated as no error
- This make the model more robust (less overfitting the data)
One more parameter ($\epsilon$) to decide

An equivalent form of the optimization problem

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

subject to

$$\mathbf{w}^T \phi(x_i) + b - y_i \leq \epsilon + \xi_i,$$

$$y_i - \mathbf{w}^T \phi(x_i) - b \leq \epsilon + \xi_i^*,$$

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

This form is similar to the SVM formulation derived earlier.
The dual problem is

$$\min_{\alpha, \alpha^*} \quad \frac{1}{2}(\alpha - \alpha^*)^T Q(\alpha - \alpha^*) + \epsilon \sum_{i=1}^{l} (\alpha_i + \alpha_i^*)$$

$$+ \sum_{i=1}^{l} y_i (\alpha_i - \alpha_i^*)$$

subject to

$$e^T (\alpha - \alpha^*) = 0,$$

$$0 \leq \alpha_i, \alpha_i^* \leq C, \quad i = 1, \ldots, l,$$

where

$$Q_{ij} = K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j).$$
After solving the dual problem,

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

and the approximate function is

\[ \sum_{i=1}^{l} (-\alpha_i + \alpha_i^*) K(x_i, x) + b. \]
SVR and least-square regression are very related.

Why people more commonly use $l_2$ (least-square) rather than $l_1$ losses?

Easier because of differentiability.
One-class SVM I

Separate data to normal ones and outliers (Schölkopf et al., 2001)

\[
\begin{align*}
\min_{\mathbf{w}, \xi, \rho} & \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} - \rho + \frac{1}{\nu l} \sum_{i=1}^{l} \xi_i \\
\text{subject to} & \quad \mathbf{w}^T \phi(\mathbf{x}_i) \geq \rho - \xi_i, \\
& \quad \xi_i \geq 0, i = 1, \ldots, l.
\end{align*}
\]
Instead of the parameter $C$ is SVM, here the parameter is $\nu$.

$$\mathbf{w}^T \phi(x_i) \geq \rho - \xi_i$$

means that we hope most data satisfy

$$\mathbf{w}^T \phi(x_i) \geq \rho.$$  

That is, most data are on one side of the hyperplane.

Those on the wrong side are considered as outliers.
One-class SVM III

- The dual problem is

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha$$

subject to

$$0 \leq \alpha_i \leq 1/(\nu l), \quad i = 1, \ldots, l,$$

$$\mathbf{e}^T \alpha = 1,$$

where $Q_{ij} = K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$.

- The decision function is

$$\text{sgn} \left( \sum_{i=1}^{l} \alpha_i K(x_i, x) - \rho \right).$$
The role of $-\rho$ is similar to the bias term $b$ earlier.

From the dual problem we can see that

$$\nu \in (0, 1]$$

Otherwise, if $\nu > 1$, then

$${\mathbf{e}^T \alpha} \leq \frac{1}{\nu} < 1$$

violates the linear constraint.

Clearly, a larger $\nu$ means we don’t need to push $\xi_i$ to zero $\Rightarrow$ more data are considered as outliers.
SVDD is another technique to identify outliers (Tax and Duin, 2004)

\[
\min_{R,a,\xi} \quad R^2 + C \sum_{i=1}^{l} \xi_i \\
\text{subject to} \quad \|\phi(x_i) - a\|^2 \leq R^2 + \xi_i, \quad i = 1, \ldots, l, \\
\xi_i \geq 0, \quad i = 1, \ldots, l,
\]
We obtain a hyperspherical model characterized by the center \( a \) and the radius \( R \).

A test instance \( x \) is detected as an outlier if

\[
\| \phi(x) - a \|^2 > R^2.
\]
The dual problem

\[
\min_{\alpha} \quad \alpha^T Q \alpha - \sum_{i=1}^{l} \alpha_i Q_{i,i}
\]

subject to

\[
e^T \alpha = 1,
\]

\[
0 \leq \alpha_i \leq C, \ i = 1, \ldots, l,
\]

This dual problem is very close to that of one-class SVM
Support Vector Data Description (SVDD)

- Consider a scaled version of one-class SVM dual

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

- If Gaussian kernel is used,

\[
Q_{i,i} = e^{-\gamma \|x_i - x_i\|^2} = 1
\]

and the two dual problems are equivalent.
Discussion

- For unsupervised settings, evaluation is very difficult
- Usually the evaluation is by a subjective way
Let’s Try a Practical Example

A problem from astroparticle physics

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

Training and testing sets available: 3,089 and 4,000
Data available at LIBSVM Data Sets
Training and Testing

Training the set svmguide1 to obtain svmguide1.model

```
$./svm-train svmguide1
```

Testing the set svmguide1.t

```
$./svm-predict svmguide1.t svmguide1.model out
```

Accuracy = 66.925% (2677/4000)

We see that training and testing accuracy are very different. Training accuracy is almost 100%

```
$./svm-predict svmguide1 svmguide1.model out
```

Accuracy = 99.7734% (3082/3089)
Why this Fails

- Gaussian kernel is used here
- We see that most kernel elements have

\[ K_{ij} = e^{-\frac{\|x_i - x_j\|^2}{4}} \begin{cases} = 1 & \text{if } i = j, \\ \rightarrow 0 & \text{if } i \neq j. \end{cases} \]

because some features in large numeric ranges
- For what kind of data,

\[ K \approx I? \]
Why this Fails (Cont’d)

- If we have training data

\[ \phi(x_1) = [1, 0, \ldots, 0]^T \]

\[ \vdots \]

\[ \phi(x_l) = [0, \ldots, 0, 1]^T \]

then

\[ K = I \]

- Clearly such training data can be correctly separated, but how about testing data?

- So overfitting occurs
Overfitting

- See the illustration in the next slide
- In theory
  You can easily achieve 100% training accuracy
- This is useless
- When training and predicting a data, we should
  Avoid **underfitting**: small training error
  Avoid **overfitting**: small testing error
and ▲: training; ○ and △: testing
Data Scaling

- Without scaling, the above overfitting situation may occur.
- Also, features in greater numeric ranges may dominate.
- Example:

<table>
<thead>
<tr>
<th>height</th>
<th>gender</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

\[ y_1 = 0, y_2 = 1, y_3 = 1. \]
The separating hyperplane almost vertical

\[ x_1 \]

\[ x_2 x_3 \]

Strongly depends on the first attribute; but second may be also important.
A simple solution is to linearly scale each feature to $[0, 1]$ by:

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

where $\max, \min$ are maximal and minimal value of each feature.

There are many other scaling methods.

Scaling generally helps, but not always.
Data Scaling (Cont’d)

- Scaling is needed for methods relying on similarity between instances
  - For example, $k$-nearest neighbor
- It’s not needed to methods such as decision trees which rely on relative positions with an attribute
Data Scaling: Same Factors

A common mistake

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

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

We need to use same factors on training and testing

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

Later we will give a real example
After Data Scaling

Train scaled data and then predict

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

Training accuracy is now similar

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

For this experiment, we use parameters $C = 1, \gamma = 0.25$, but sometimes performances are sensitive to parameters...
Parameters versus Performances

- If we use $C = 20, \gamma = 400$
  
  ```
  ./svm-train -c 20 -g 400 svmguide1.scale
  ./svm-predict svmguide1.scale svmguide1.scale.model
  Accuracy = 100% (3089/3089)
  ```

- 100% training accuracy but
  
  ```
  ./svm-predict svmguide1.t.scale svmguide1.scale.model
  Accuracy = 82.7% (3308/4000)
  ```

- Very bad test accuracy

- Overfitting happens
Parameter Selection

- For SVM, we may need to select suitable parameters.
- They are $C$ and 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 is better?
Performance Evaluation

- Available data ⇒ training and validation
- Train the training; test the validation to estimate the performance
- A common way is $k$-fold cross validation (CV):
  - Data randomly separated to $k$ groups
  - Each time $k - 1$ as training and one as testing
- Select parameters/kernels with best CV result
- There are many other methods to evaluate the performance
Contour of CV Accuracy
The good region of parameters is quite large.

SVM is sensitive to parameters, but not that sensitive.

Sometimes default parameters work, but it’s good to select them if time is allowed.
Example of Parameter Selection

Direct training and test

```
./svm-train svmguide3
./svm-predict svmguide3.t svmguide3.model o
→ Accuracy = 2.43902%
```

After data scaling, accuracy is still low

```
./svm-scale -s range3 svmguide3 > svmguide3.scale
./svm-scale -r range3 svmguide3.t > svmguide3.t.scale
./svm-train svmguide3.scale
./svm-predict svmguide3.t.scale svmguide3.scale.model o
→ Accuracy = 12.1951%
```
Example of Parameter Selection (Cont’d)

Select parameters by trying a grid of \((C, \gamma)\) values

\[
\text{python grid.py svmguide3.scale}
\]

\[
128.0 \ 0.125 \ 84.8753
\]

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

Train and predict using the obtained parameters

\[
\text{./svm-train -c 128 -g 0.125 svmguide3.scale}
\]
\[
\text{./svm-predict svmguide3.t.scale svmguide3.scale.model svmguide3.t.predict}
\]

→ Accuracy = 87.8049%
Selecting Kernels

- RBF, polynomial, or others?
- For beginners, use RBF first
- Linear kernel: special case of RBF
  Accuracy of linear the same as RBF under certain parameters (Keerthi and Lin, 2003)
- Polynomial kernel:
  \[(x_i^T x_j / a + b)^d\]
  Numerical difficulties: \((< 1)^d \rightarrow 0, (> 1)^d \rightarrow \infty\)
  More parameters than RBF
Selecting Kernels (Cont’d)

- Commonly used kernels are Gaussian (RBF), polynomial, and linear
- But in different areas, special kernels have been developed. Examples
  1. $\chi^2$ kernel is popular in computer vision
  2. String kernel is useful in some domains
A Simple Procedure for Beginners

After helping many users, we came up with 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

In LIBSVM, we have a python script `easy.py` implementing this procedure.
A Simple Procedure for Beginners (Cont’d)

- We proposed this procedure in an “SVM guide” (Hsu et al., 2003) and implemented it in LIBSVM.
- From research viewpoints, this procedure is not novel. We never thought about submitting our guide somewhere.
- But this procedure has been tremendously useful. Now almost the standard thing to do for SVM beginners.
A Real Example of Wrong Scaling

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

$ ../svm-scale -l 0 svmguide4 > svmguide4.scale
$ ../svm-scale -l 0 svmguide4.t > svmguide4.t.scale
$ python easy.py svmguide4.scale svmguide4.t.scale

Accuracy = 69.2308% (216/312) (classification)

The accuracy is low even after parameter selection

$ ../svm-scale -l 0 -s range4 svmguide4 > svmguide4.scale
$ ../svm-scale -r range4 svmguide4.t > svmguide4.t.scale
$ python easy.py svmguide4.scale svmguide4.t.scale

Accuracy = 89.4231% (279/312) (classification)
A Real Example of Wrong Scaling (Cont’d)

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

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

Scaling the test set to $[0, 1]$ generated an erroneous set.
More about Cross Validation

- CV can be used for other classification methods
- For example, a common way to select $k$ of $k$ nearest neighbor is by CV
- However, it’s easy that CV is misused
More about Cross Validation (Cont’d)

- CV is a biased estimate
- Think about this. If you have many parameters, you may adjust them to boost your CV accuracy
- In some papers, people compare CV accuracy of different methods
- This is not very appropriate
- It’s better to report independent test accuracy
- Indeed you are allowed to predict the test set only once for reporting the results
More about Cross Validation (Cont’d)

- Sometimes you must be careful in splitting data for CV

- Assume you have 20,000 images of 200 users:
  - User 1: 100 images
  - ...
  - User 200: 100 images

- The standard CV may overestimate the performance because of easier predictions
More about Cross Validation (Cont’d)

- An instance in the validation set may find a close one in the training set.
- A more suitable setting is to split data by meta-level information (i.e., users here).
Outline

1. Introduction
2. SVM and kernel methods
3. Dual problem and solving optimization problems
4. Regularization and linear versus kernel
5. Multi-class classification
6. Support vector regression
7. SVM for clustering
8. Practical use of support vector classification
9. A practical example of SVR
10. Discussion and conclusions
Electricity Load Forecasting

- EUNITE world wide competition 2001
  http://neuron-ai.tuke.sk/competition
- We were given
  - Load per half hour from 1997 to 1998
  - Average daily temperature from 1995 to 1998
  - List of holidays
- Goal:
  - Predict daily maximal load of January 1999
- A time series prediction problem
SVR for Time Series Prediction I

- Given \((\cdots, y_{t-\Delta}, \cdots, y_{t-1}, y_{t}, \cdots, y_{l})\) as training series
- Generate training data:
  \((y_{t-\Delta}, \cdots, y_{t-1})\) as attributes (features) of \(x_i\)
  \(y_t\) as the target value \(x_i\)
- One-step ahead prediction
- Prediction:
  Starting from the last segment
  \((y_{l-\Delta+1}, \cdots, y_{l})\) \(\rightarrow\) \(\hat{y}_{l+1}\)

Repeat by using newly predicted values
Data Analyses I

- Maximal load of each day
- Didn’t know how to use all **half-hour data**
- Not used for later analyses/experiments
Data Analyses II

- Issues largely discussed in earlier works
  - Seasonal periodicity
  - Weekly periodicity
    - Weekday: higher, Weekend: lower
  - Holiday effect
  - All above known for January 1999
  - Weather influence
    - Temperature unknown for January 1999
- Temperature is very very important
  - The main difficulty of this competition
Most early work on short-term prediction: Temperature available
Error propagation of time series prediction is an issue
Methods

- In addition to SVR, another simple and effective method is **local modeling**
- It is like nearest neighbor in classification
- Local modeling:
  - Finding segments closely resemble the segment proceeding the point to be predicted
  - Average of elements after these similar segments of points.
Data Encoding I

- Both methods:
  - Use a segment (a vector) for predicting the next value
  - Encoding: contents of a segment
- The simplest:
  - Each segment: load of the previous $\Delta$ days
  - Used for local model: $\Delta = 7$
- For SVM: more information is incorporated
  - Seven attributes: maximal loads of the past 7 days
Data Encoding II

- Seven binary (i.e. 0 or 1) attributes:
  - target day in which day of a week
  - One binary attribute: target day holiday or not
  - One attribute (optional): temperature of the target day

- Temperature unknown: train two SVMs
  One for load and one for temperature

\[
\begin{pmatrix}
  y_{t-\Delta} \\
  T_{t-\Delta}
\end{pmatrix}, \ldots, \begin{pmatrix}
  y_{t-1} \\
  T_{t-1}
\end{pmatrix} \xrightarrow{\text{SVM1}} y_t \\
T_{t-\Delta}, \ldots, T_{t-1} \xrightarrow{\text{SVM2}} T_t
\]
Model Selection I

- Parameters and features
  - \( \Delta \): for both approaches
  - Local model: \# of similar segments
  - SVR:
    1. \( C \): cost of error
    2. \( \epsilon \): width of the \( \epsilon \)-insensitive loss
    3. mapping function \( \phi \)

- Extremely important for data prediction
- Known data separated to
  - Training, validation, and testing
Model Selection II

- January 1997 or January 1998 as validation
- Model selection is expensive
  - Restrict the search space: via reasonable choices or simply guessing
    - $\Delta = 7$
    - SVR:
      - RBF function $\phi(x_i)^T \phi(x_j) = e^{-\gamma \|x_i - x_j\|^2}$
      - Use default width $\epsilon = 0.5$ of LIBSVM
      - $C = 2^{12}, \gamma = 2^{-4}$: decided by validation
Without summer:
Result for testing January 1998 (or 1997) better
Give up information from April to September
This is an example where domain knowledge is used
However, can we do automatic time series segmentation to see that summer and winter are different?
Evaluation of Time Series Prediction I

- **MSE (Mean Square Error):**
  \[ \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \]

- **MAPE (Mean absolute percentage error):**
  \[ \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \]

- Error propagation: larger error later
  Unfair to earlier prediction if MSE is used
There are other criteria
Results: Local Model I

- Validation on different number of segments
- Results in the competition: slightly worse than SVR
Results: SVR I

- Two SVRs: very difficult to predict temperature
- If in one day, temperature suddenly drops or increases
  \[\Rightarrow\text{Erroneous after that day}\]
- We conclude if temperature is used, the variation is higher
- We decide to give up using the temperature information
- Only one SVM used

Prediction results for January 1998:
The load of each week is similar

However, the model manages to find the trend
A practical example of SVR

Results: SVR III

- Holiday is lower but error larger
- Results after encountering a holiday *more* inaccurate
- Holidays: January 1 and 6
- Treat all 31 days in January 1999 as non-holidays
- Some earlier work consider holidays and non-holidays separately
- We cannot do this because information about holidays is quite limited
- Overall we take a very *conservative* approach
- Forgot to manually lower load of January 6
- Reason why our \( \text{max}_i(\text{error}_i) \) not good
Results: SVR IV

- MAPE: 1.98839%
- MSE: 364.498
Discussion I

- Instead of this conservative approach, can we do better?
- Is there a good way to use temperature information?
- Feature selection is the key for our approach
  Example: removing summer data, treating holidays as non-holidays
- Parameter selection: needed but a large range is ok
  For example, if $C = 2^{12}, \gamma = 2^{-4}$ becomes $C = 2^{12}, \gamma = 2^{-5}$
  $\Rightarrow$ results do not change much
Discussion and conclusions

Outline

1. Introduction
2. SVM and kernel methods
3. Dual problem and solving optimization problems
4. Regularization and linear versus kernel
5. Multi-class classification
6. Support vector regression
7. SVM for clustering
8. Practical use of support vector classification
9. A practical example of SVR
10. Discussion and conclusions
Conclusions

- In this short course, we have introduced details of SVM.
- Linear versus kernel is an important issue. You must decide when to use which.
- No matter how many advanced techniques are developed, simple models like linear SVM or logistic regression will remain to be the first thing to try.
References I


References II


