Support Vector Classification and Regression

Chih-Jen Lin
Department of Computer Science
National Taiwan University

Short course at ITRI, 2016

Outline

Outline

About this Course

- 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

About this Course (Cont'd)

- 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)

Outline

Support Vector Classification

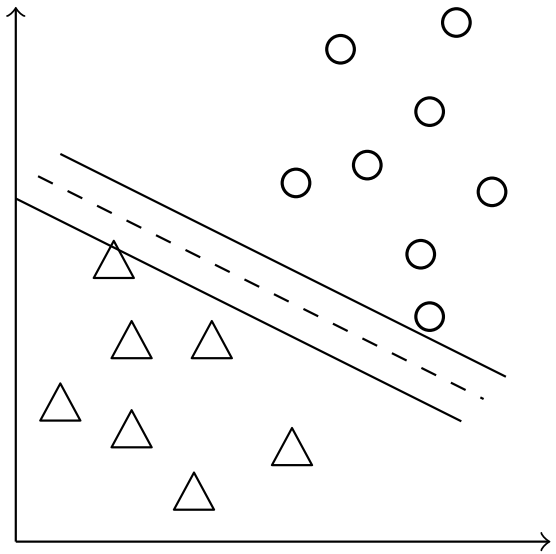
- Training vectors : $\boldsymbol{x}_i, i = 1, \ldots, l$
- Feature vectors. For example,

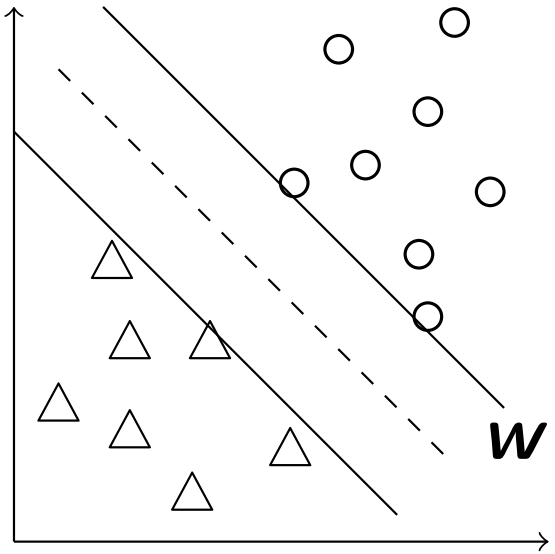
  A patient $=$ [height, weight, $\ldots$]^T
- Consider a simple case with two classes:

  Define an indicator vector $\mathbf{y} \in R^l$

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

- A hyperplane which separates all data





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

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

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

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

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

Maximal Margin

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

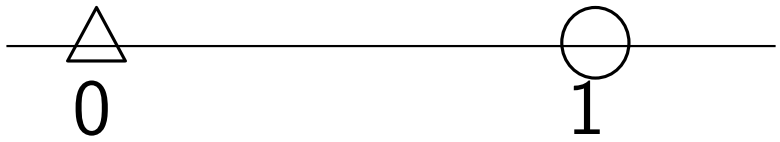
$$2/\|\boldsymbol{w}\| = 2/\sqrt{\boldsymbol{w}^T\boldsymbol{w}}$$

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

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

Example

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



  What is the separating hyperplane ?
- Now two data are $\boldsymbol{x}_1 = 1, \boldsymbol{x}_2 = 0$ with

$$\mathbf{y} = [+1, -1]^T$$

Example (Cont'd)

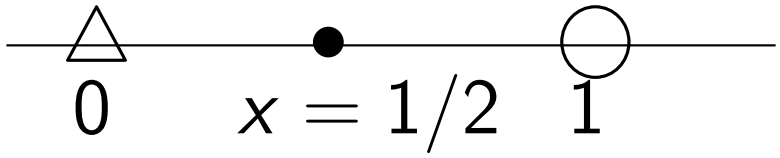
- Now $\boldsymbol{w} \in R^1$. The optimization problem is

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

- From (2), $-b \geq 1$.
- Putting this into (1), $w \geq 2$.
- That is, for any (w, b) satisfying (1) and (2), $w \geq 2$.

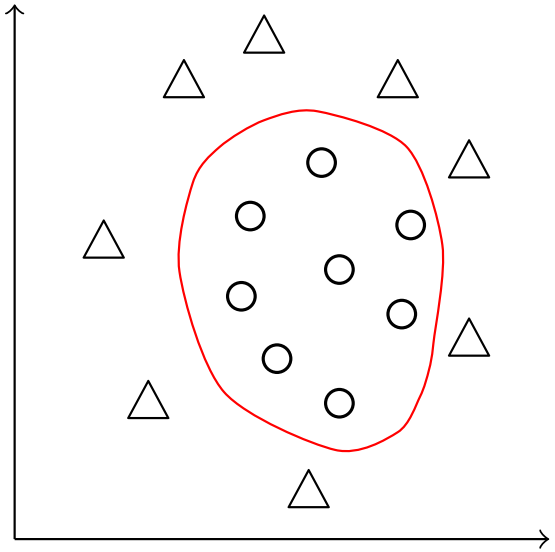
Example (Cont'd)

- 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:



Data May Not Be Linearly Separable

- An example:



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

$$\phi(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \ldots]^T.$$

- Standard SVM (Boser et al., 1992; Cortes and Vapnik, 1995)

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

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

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

Finding the Decision Function

- $\boldsymbol{w}$: maybe infinite variables
- The dual problem: finite number of variables

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

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

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

- A finite problem: #variables = #training data

Kernel Tricks

- $Q_{ij} = y_i y_j \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$ needs a closed form
- Example: $\boldsymbol{x}_i \in R^3, \phi(\boldsymbol{x}_i) \in R^{10}$

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

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

- Kernel: $K(\boldsymbol{x}, \mathbf{y}) = \phi(\boldsymbol{x})^T \phi(\mathbf{y})$; common kernels:

$$e^{-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}, \text{ (Radial Basis Function or Gaussian kernel)}$$
$$(\boldsymbol{x}_i^T \boldsymbol{x}_j / a + b)^d \text{ (Polynomial kernel)}$$

Can be inner product in infinite dimensional space

Assume $x \in R^1$ 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

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

Decision function

- At optimum

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

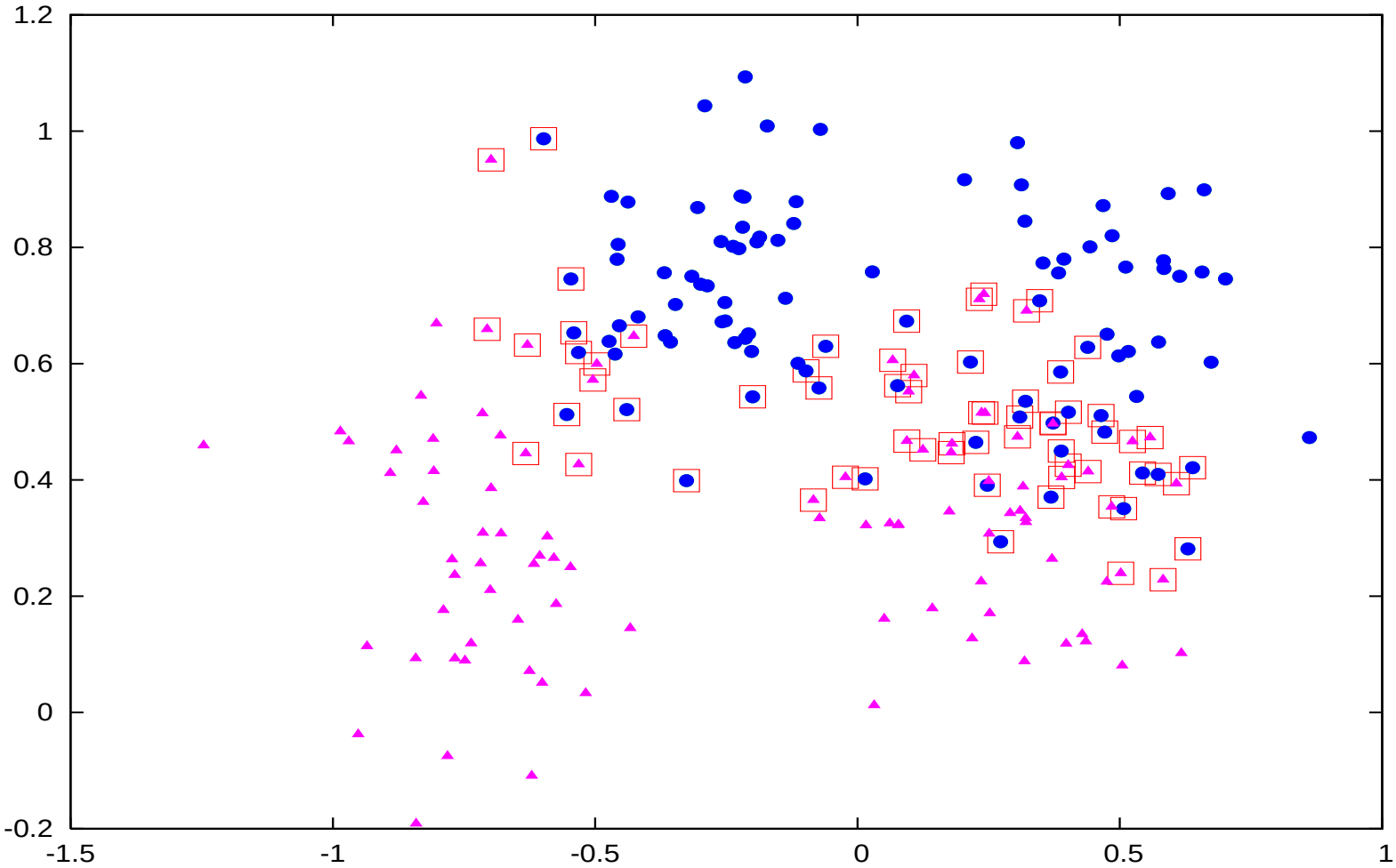
- Decision function

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

- Only $\phi(\boldsymbol{x}_i)$ of $\alpha_i > 0$ used $\Rightarrow$ support vectors

Support Vectors: More Important Data

Only $\phi(\boldsymbol{x}_i)$ 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 ξ_i

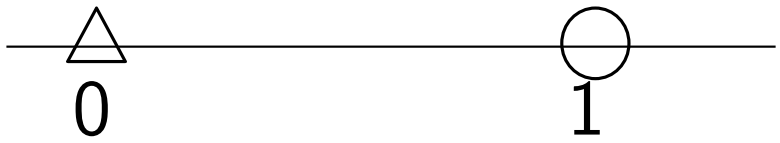
$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} \\ \text{subject to} \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1, i = 1, \ldots, l. \end{aligned}$$

Dual problem is

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

Example: Primal-dual Relationship (Cont'd)

- Consider the earlier example:



- Now two data are $\boldsymbol{x}_1 = 1, \boldsymbol{x}_2 = 0$ with

$$\mathbf{y} = [+1, -1]^T$$

- The solution is $(w, b) = (2, -1)$

Example: Primal-dual Relationship (Cont'd)

- The dual objective function

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

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

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

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

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

- 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

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) < 1 \Rightarrow \alpha_i > 0 \Rightarrow \text{ support vector},$$

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) = 1 \Rightarrow \alpha_i \geq 0 \Rightarrow \text{ maybe SV}$$

and

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) > 1 \Rightarrow \alpha_i = 0 \Rightarrow \text{ not SV}$$

Outline

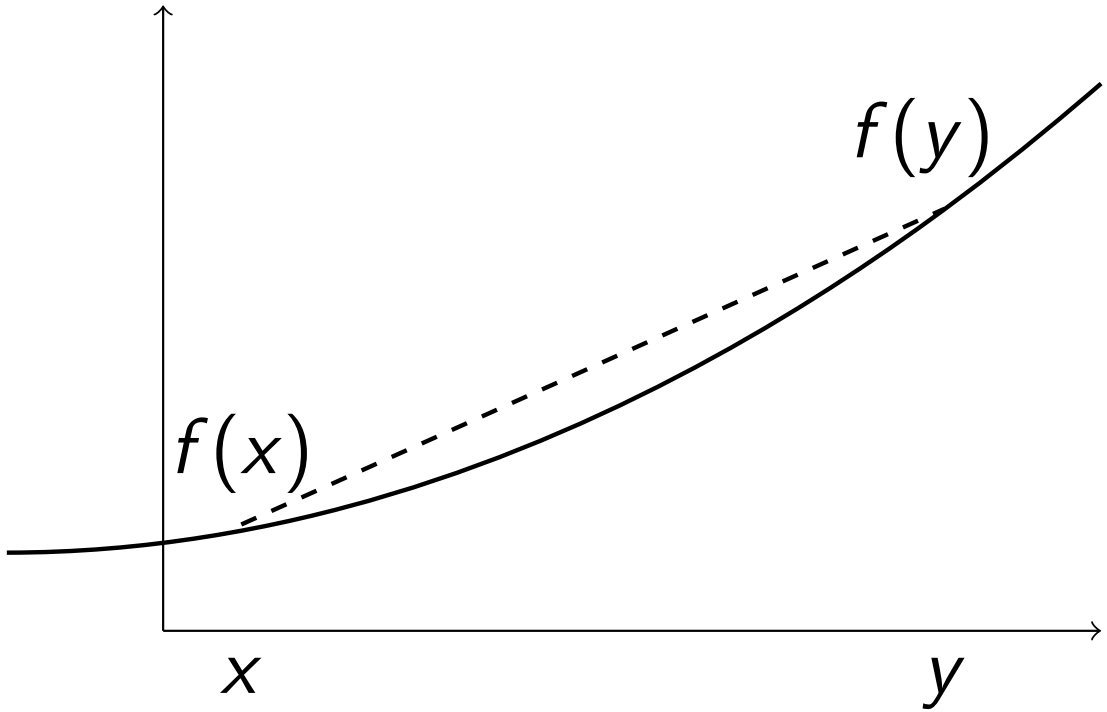
Convex Optimization I

- Convex problems are a 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



Convex Optimization III

- A convex optimization problem takes the following form

$$\begin{aligned} \min \quad & f_0(\boldsymbol{w}) \\ \text{subject to} \quad & f_i(\boldsymbol{w}) \leq 0, i = 1, \ldots, m, \\ & h_i(\boldsymbol{w}) = 0, i = 1, \ldots, p, \end{aligned} \tag{3}$$

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

$$h_i(\boldsymbol{w}) = \mathbf{a}^T \boldsymbol{w} + b$$

Convex Optimization IV

- A nice property of convex optimization problems is that

$$\inf_{\boldsymbol{w}} \{ f_0(\boldsymbol{w}) \mid \boldsymbol{w} \text{ satisfies constraints} \}$$

  is unique
- Optimal objective value is unique, but optimal $\boldsymbol{w}$ 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 ξ_i

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

- Its dual is

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

where

$$Q_{ij} = y_i y_j \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$$

Lagrangian Dual I

- Lagrangian dual

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

where

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

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

  $\Rightarrow$ 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 $\boldsymbol{\alpha}$ is fixed,

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

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

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

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

in $L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ to $-\infty$

- If $\sum_{i=1}^{l} \alpha_i y_i = 0$, optimum of the strictly convex function

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

  happens when

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

- Thus,

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

- Note that

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

- The dual is

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

- Lagrangian dual: $\max_{\boldsymbol{\alpha}\geq 0}\left(\min_{\boldsymbol{w},b} L(\boldsymbol{w}, b, \boldsymbol{\alpha})\right)$
- $-\infty$ 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

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

- Our problems may be infinite dimensional (i.e., $\boldsymbol{w} \in R^\infty$)
- We can still use Lagrangian duality

  See a rigorous discussion in Lin (2001)

Primal versus Dual I

- Recall the dual problem is

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

and at optimum

$$\boldsymbol{w} = \sum_{i=1}^{l} \alpha_i y_i \phi(\boldsymbol{x}_i) \tag{4}$$

Primal versus Dual II

- What if we put (4) into primal

$$\begin{aligned} \min_{\boldsymbol{\alpha},\boldsymbol{\xi}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & (Q\boldsymbol{\alpha} + b\mathbf{y})_i \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned} \tag{5}$$

- Note that

$$\begin{aligned} y_i \boldsymbol{w}^T \phi(\boldsymbol{x}_i) &= y_i \sum_{j=1}^{l} \alpha_j y_j \phi(\boldsymbol{x}_j)^T \phi(\boldsymbol{x}_i) \\ &= \sum_{j=1}^{l} Q_{ij}\alpha_j = (Q\boldsymbol{\alpha})_i \end{aligned}$$

Primal versus Dual III

- If Q is positive definite, we can prove that the optimal α of (5) is the same as that of the dual
- So dual is not the only choice to obtain the model

Large Dense Quadratic Programming

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

- $Q_{ij} \neq 0$, 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

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, $N = \{1, \ldots, l\} \backslash B$ fixed
- Sub-problem at the kth iteration:

$$\min_{\boldsymbol{\alpha}_B} \quad \frac{1}{2} \begin{bmatrix} \boldsymbol{\alpha}_B^T & (\boldsymbol{\alpha}_N^k)^T \end{bmatrix} \begin{bmatrix} Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_B \\ \boldsymbol{\alpha}_N^k \end{bmatrix} - \begin{bmatrix} \boldsymbol{e}_B^T & (\boldsymbol{e}_N^k)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_B \\ \boldsymbol{\alpha}_N^k \end{bmatrix}$$

$$\text{subject to} \quad 0 \le \alpha_t \le C, t \in B, \ \mathbf{y}_B^T \boldsymbol{\alpha}_B = -\mathbf{y}_N^T \boldsymbol{\alpha}_N^k$$

Avoid Memory Problems

- The new objective function

$$\frac{1}{2}\begin{bmatrix}\boldsymbol{\alpha}_B^T & (\boldsymbol{\alpha}_N^k)^T\end{bmatrix}\begin{bmatrix}Q_{BB}\boldsymbol{\alpha}_B + Q_{BN}\boldsymbol{\alpha}_N^k \\ Q_{NB}\boldsymbol{\alpha}_B + Q_{NN}\boldsymbol{\alpha}_N^k\end{bmatrix}$$
$$-\boldsymbol{e}_B^T\boldsymbol{\alpha}_B + \text{ constant}$$
$$= \frac{1}{2}\boldsymbol{\alpha}_B^T Q_{BB}\boldsymbol{\alpha}_B + (-\boldsymbol{e}_B + Q_{BN}\boldsymbol{\alpha}_N^k)^T\boldsymbol{\alpha}_B + \text{ constant}$$

- 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 α

  decision function:

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

  Prediction may still be correct with a rough α
- Further, in some situations,

$$\#\text{ support vectors} \ll \#\text{ training points}$$

  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

```
$svm-train -c 16 -g 4 -m 400 22features
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

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

- A large C means to overfit training data
- This does not give good test accuracy. More details about overfitting will be discussed later

Outline

Equivalent Optimization Problem

- Recall SVM optimization problem is

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

- It is equivalent to

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\max(0, 1 - y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b))$$

- The reformulation is useful to derive SVM from a different viewpoint

Equivalent Optimization Problem (Cont'd)

- That is, at optimum,

$$\xi_i = \max(0, 1 - y_i(\boldsymbol{w}^T \phi(\boldsymbol{x}_i) + b))$$

- Reason: from constraints

$$\xi_i \geq 1 - y_i(\boldsymbol{w}^T \phi(\boldsymbol{x}_i) + b) \text{ and } \xi_i \geq 0$$

but we also want to minimize ξ_i

Linear and Kernel I

- Linear classifier

$$\text{sgn}(\boldsymbol{w}^T\boldsymbol{x} + b)$$

- Kernel classifier

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

- Linear is a special case of kernel
- An important difference is that for linear we can store $\boldsymbol{w}$

Linear and Kernel II

- For kernel, $\boldsymbol{w}$ 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

$$\text{sgn}(\boldsymbol{w}^T\boldsymbol{x} + b)$$

- Sometimes the bias term b is omitted

$$\text{sgn}(\boldsymbol{w}^T\boldsymbol{x})$$

- This is fine if the number of features is not too small

Minimizing Training Errors

- For classification naturally we aim to minimize the training error

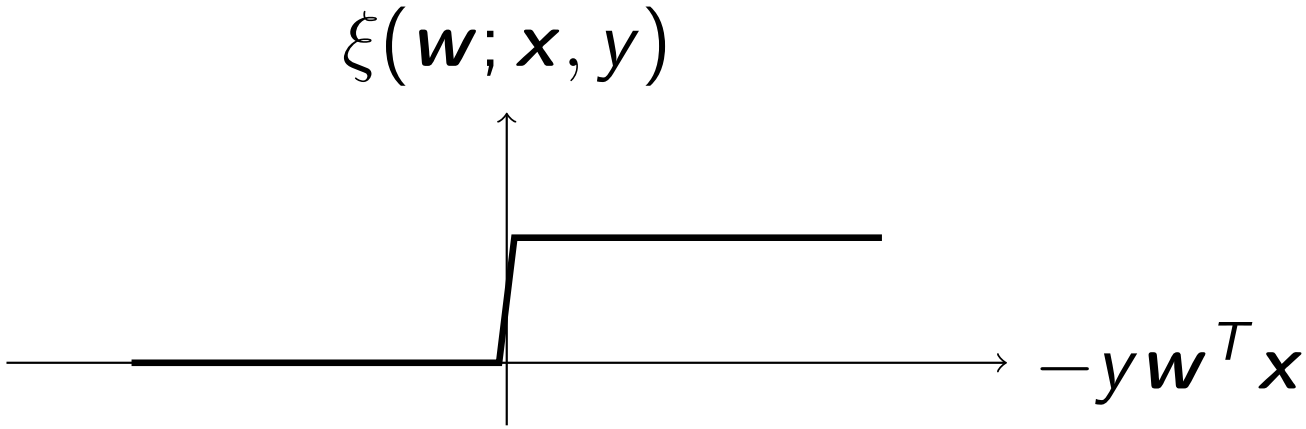
$$\min_{\boldsymbol{w}} \quad \text{(training errors)}$$

- To characterize the training error, we need a loss function $\xi(\boldsymbol{w}; \boldsymbol{x}, y)$ for each instance $(\boldsymbol{x}, y)$
- Ideally we should use 0–1 training loss:

$$\xi(\boldsymbol{w}; \boldsymbol{x}, y) = \begin{cases} 1 & \text{if } y\boldsymbol{w}^T\boldsymbol{x} < 0, \\ 0 & \text{otherwise} \end{cases}$$

Minimizing Training Errors (Cont'd)

- However, this function is discontinuous. The optimization problem becomes difficult



- We need continuous approximations

Common Loss Functions

- Hinge loss (l1 loss)

$$\xi_{\text{L1}}(\boldsymbol{w}; \boldsymbol{x}, y) \equiv \max(0, 1 - y\boldsymbol{w}^T\boldsymbol{x}) \tag{6}$$

- Squared hinge loss (l2 loss)

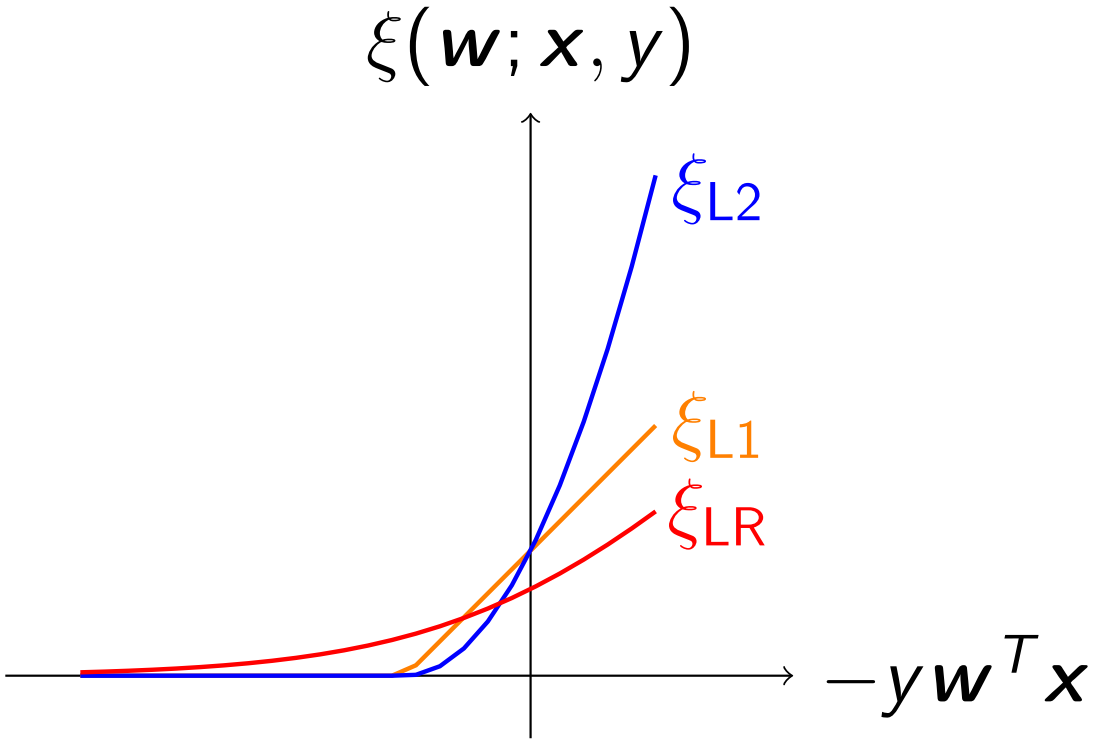
$$\xi_{\text{L2}}(\boldsymbol{w}; \boldsymbol{x}, y) \equiv \max(0, 1 - y\boldsymbol{w}^T\boldsymbol{x})^2 \tag{7}$$

- Logistic loss

$$\xi_{\text{LR}}(\boldsymbol{w}; \boldsymbol{x}, y) \equiv \log(1 + e^{-y\boldsymbol{w}^T\boldsymbol{x}}) \tag{8}$$

- SVM: (6)-(7). Logistic regression (LR): (8)

Common Loss Functions (Cont'd)



- 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

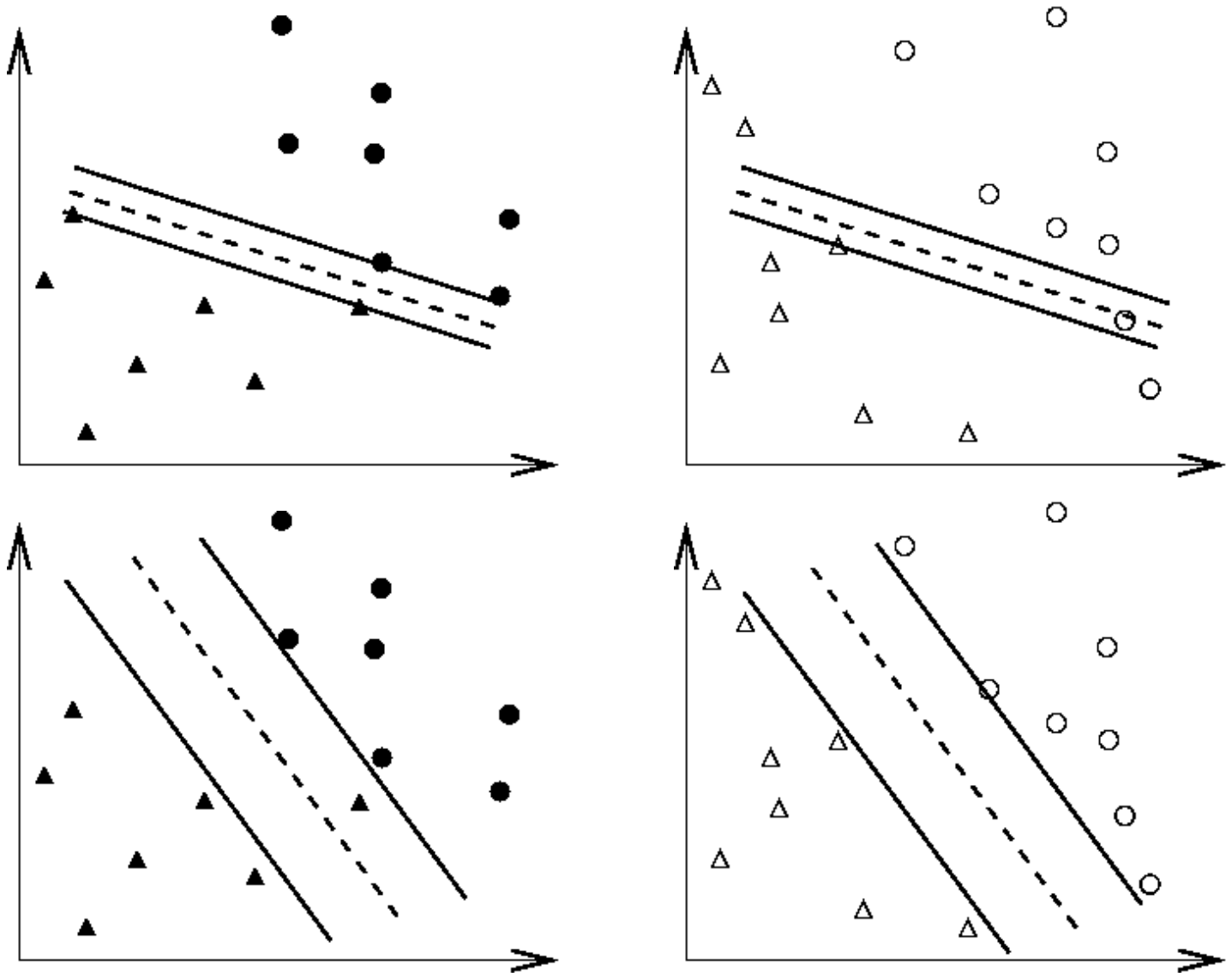
- See the illustration in the next slide
- For classification,

  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 $\boldsymbol{w}$ vector so that it fits the data
- To avoid overfitting we need a way to make $\boldsymbol{w}$'s values less extreme.
- One idea is to make $\boldsymbol{w}$ values closer to zero
- We can add, for example,

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

to the objective function

General Form of Linear Classification I

- Training data $\{y_i, \boldsymbol{x}_i\}, \boldsymbol{x}_i \in R^n, i = 1, \ldots, l, y_i = \pm 1$
- l: # of data, n: # of features

$$\min_{\boldsymbol{w}} f(\boldsymbol{w}), \quad f(\boldsymbol{w}) \equiv \frac{\boldsymbol{w}^T \boldsymbol{w}}{2} + C \sum_{i=1}^{l} \xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i) \tag{9}$$

- $\boldsymbol{w}^T \boldsymbol{w}/2$: regularization term
- $\xi(\boldsymbol{w}; \boldsymbol{x}, y)$: loss function
- C: regularization parameter

General Form of Linear Classification II

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

$$\min_{\boldsymbol{w}} f(\boldsymbol{w}), \quad f(\boldsymbol{w}) \equiv \frac{\boldsymbol{w}^T \boldsymbol{w}}{2} + C \sum_{i=1}^{l} \xi(\boldsymbol{w}; \phi(\boldsymbol{x}_i), y_i)$$

SVM and Logistic Regression I

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

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\max(0, 1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i)$$

  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 $\Rightarrow$ kernel SVM
  - when people say logistic regression $\Rightarrow$ linear logistic regression
- Indeed we can do kernel logistic regression

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\log(1+e^{-y_i\boldsymbol{w}^T\phi(\boldsymbol{x}_i)})$$

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 $(y, \boldsymbol{x})$, assume the probability model is

$$p(y|\boldsymbol{x}) = \frac{1}{1 + e^{-y\boldsymbol{w}^T\boldsymbol{x}}}.$$

- Note that

$$\begin{aligned} & p(1|\boldsymbol{x}) + p(-1|\boldsymbol{x}) \\ = & \frac{1}{1 + e^{-\boldsymbol{w}^T\boldsymbol{x}}} + \frac{1}{1 + e^{\boldsymbol{w}^T\boldsymbol{x}}} \\ = & \frac{e^{\boldsymbol{w}^T\boldsymbol{x}}}{1 + e^{\boldsymbol{w}^T\boldsymbol{x}}} + \frac{1}{1 + e^{\boldsymbol{w}^T\boldsymbol{x}}} \\ = & 1 \end{aligned}$$

Logistic Regression (Cont'd)

- Idea of this model

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

- Assume training instances are

$$(y_i, \boldsymbol{x}_i), i = 1, \ldots, l$$

Logistic Regression (Cont'd)

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

$$\max_{\boldsymbol{w}} \quad \prod_{i=1}^{l} p\left(y_i|\boldsymbol{x}_i\right). \tag{10}$$

- Negative log-likelihood

$$-\log \prod_{i=1}^{l} p\left(y_i|\boldsymbol{x}_i\right) = -\sum_{i=1}^{l} \log p\left(y_i|\boldsymbol{x}_i\right)$$

$$= \sum_{i=1}^{l} \log\left(1 + e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i}\right)$$

Logistic Regression (Cont'd)

- Logistic regression

$$\min_{\boldsymbol{w}} \quad \sum_{i=1}^{l} \log\left(1 + e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i}\right).$$

- Regularized logistic regression

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l} \log\left(1 + e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i}\right). \quad (11)$$

C: regularization parameter decided by users

Loss Functions: Differentiability

However,

ξ_{L1}: not differentiable
ξ_{L2}: differentiable but not twice differentiable
ξ_{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

$$\|\boldsymbol{w}\|_1 \text{ and } \boldsymbol{w}^T\boldsymbol{w}/2$$

- $\boldsymbol{w}^T\boldsymbol{w}/2$: smooth, easier to optimize
- $\|\boldsymbol{w}\|_1$: non-differentiable

  sparse solution; possibly many zero elements
- Possible advantages of L1 regularization:

  Feature selection

  Less storage for $\boldsymbol{w}$

Linear and Kernel Classification

Methods such as SVM and logistic regression can used in two ways

- Kernel methods: data mapped to a higher dimensional space

$$\boldsymbol{x} \Rightarrow \phi(\boldsymbol{x})$$

  $\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$ easily calculated; little control on $\phi(\cdot)$
- Feature engineering + linear classification:

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

We refer to them as kernel and linear classifiers

Linear and Kernel Classification

- Let's check the prediction cost

$$\boldsymbol{w}^T\boldsymbol{x} + b \quad \text{versus} \quad \sum_{i=1}^{l} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b$$

- If $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ takes $O(n)$, then

$$O(n) \quad \text{versus} \quad O(nl)$$

- Linear is much cheaper
- A similar difference occurs for training

Linear and Kernel Classification (Cont'd)

- 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

$$\begin{aligned} \text{test accuracy:} &\quad \text{kernel} \geq \text{linear} \\ \text{cost:} &\quad \text{kernel} \gg \text{linear} \end{aligned}$$

- Speed is the reason to use linear

Linear and Kernel Classification (Cont'd)

- 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)

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

Size reasonably large: e.g., yahoo-japan: 140k instances and 830k features

Comparison Between Linear and Kernel (Training Time & Testing Accuracy)

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

Size reasonably large: e.g., yahoo-japan: 140k instances and 830k features

Comparison Between Linear and Kernel (Training Time & Testing Accuracy)

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

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 $\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_l)$

- Kernel not used
- Applicable only if dimension of $\phi(\boldsymbol{x})$ not too large

Low-degree Polynomial Mappings

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^T \boldsymbol{x}_j + 1)^2 = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

$$\phi(\boldsymbol{x}) = [1, \sqrt{2}x_1, \ldots, \sqrt{2}x_n, x_1^2, \ldots, x_n^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_{n-1}x_n]^T$$

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 $\phi(\boldsymbol{x})$

Testing Accuracy and Training Time

Data set	Degree-2 Polynomial			Accuracy diff.	
	Training time (s)		Accuracy		
	LIBLINEAR	LIBSVM		Linear	RBF
a9a	1.6	89.8	85.06	0.07	0.02
real-sim	59.8	1,220.5	98.00	0.49	0.10
ijcnn1	10.7	64.2	97.84	5.63	−0.85
MNIST38	8.6	18.4	99.29	2.47	−0.40
covtype	5,211.9	NA	80.09	3.74	−15.98
webspam	3,228.1	NA	98.44	5.29	−0.76

Training $\phi(\boldsymbol{x}_i)$ by linear: faster than kernel, but sometimes competitive accuracy

Example: Dependency Parsing I

- This is an NLP Application

	Kernel		Linear	
	RBF	Poly-2	Linear	Poly-2
Training time	3h34m53s	3h21m51s	3m36s	3m43s
Parsing speed	0.7x	1x	1652x	103x
UAS	89.92	91.67	89.11	91.71
LAS	88.55	90.60	88.07	90.71

- 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 $\phi(\boldsymbol{x}_i), \forall i$
- We consider the following low-degree polynomial mapping:

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

Handing High Dimensionality of $\phi(\boldsymbol{x})$

A multi-class problem with sparse data

n	Dim. of $\phi(\boldsymbol{x})$	l	$\bar{n}$	$\boldsymbol{w}$'s # nonzeros
46,155	1,065,165,090	204,582	13.3	1,438,456

- $\bar{n}$: average # nonzeros per instance
- Dimensionality of $\boldsymbol{w}$ is very high, but $\boldsymbol{w}$ is sparse
  Some training feature columns of $x_i x_j$ are entirely zero
- Hashing techniques are used to handle sparse $\boldsymbol{w}$

Example: Classifier in a Small Device

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

Classifiers	Test accuracy	Model Size
Decision Tree	77.77	76.02KB
AdaBoost (10 trees)	78.84	1,500.54KB
SVM (RBF kernel)	85.33	1,287.15KB

- 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

SVM method	Test accuracy	Model Size
RBF kernel	85.33	1,287.15KB
Polynomial kernel	84.79	2.28KB
Linear kernel	78.51	0.24KB

Outline

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:

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

- k decision functions

$$(\boldsymbol{w}^1)^T \phi(\boldsymbol{x}) + b_1$$
$$\vdots$$
$$(\boldsymbol{w}^k)^T \phi(\boldsymbol{x}) + b_k$$

- Prediction:

$$\arg\max_j \ (\boldsymbol{w}^j)^T \phi(\boldsymbol{x}) + b_j$$

- Reason: If $\boldsymbol{x} \in$ 1st class, then we should have

$$\begin{aligned}
(\boldsymbol{w}^1)^T \phi(\boldsymbol{x}) + b_1 &\geq +1 \\
(\boldsymbol{w}^2)^T \phi(\boldsymbol{x}) + b_2 &\leq -1 \\
&\vdots \\
(\boldsymbol{w}^k)^T \phi(\boldsymbol{x}) + b_k &\leq -1
\end{aligned}$$

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

$y_i = 1$	$y_i = -1$	Decision functions
class 1	class 2	$f^{12}(\boldsymbol{x}) = (\boldsymbol{w}^{12})^T\boldsymbol{x} + b^{12}$
class 1	class 3	$f^{13}(\boldsymbol{x}) = (\boldsymbol{w}^{13})^T\boldsymbol{x} + b^{13}$
class 1	class 4	$f^{14}(\boldsymbol{x}) = (\boldsymbol{w}^{14})^T\boldsymbol{x} + b^{14}$
class 2	class 3	$f^{23}(\boldsymbol{x}) = (\boldsymbol{w}^{23})^T\boldsymbol{x} + b^{23}$
class 2	class 4	$f^{24}(\boldsymbol{x}) = (\boldsymbol{w}^{24})^T\boldsymbol{x} + b^{24}$
class 3	class 4	$f^{34}(\boldsymbol{x}) = (\boldsymbol{w}^{34})^T\boldsymbol{x} + b^{34}$

- For a testing data, predicting all binary SVMs

Classes		winner
1	2	1
1	3	1
1	4	1
2	3	2
2	4	4
3	4	3

- Select the one with the largest vote

class	1	2	3	4
# votes	3	1	1	1

- May use decision values as well

Solving a Single Problem

- An approach by Crammer and Singer (2002)

$$\min_{\boldsymbol{w}_1,\dots,\boldsymbol{w}_k} \quad \frac{1}{2}\sum_{m=1}^{k}\|\boldsymbol{w}_m\|_2^2 + C\sum_{i=1}^{l}\xi(\{\boldsymbol{w}_m\}_{m=1}^{k};\boldsymbol{x}_i,y_i),$$

where

$$\xi(\{\boldsymbol{w}_m\}_{m=1}^{k};\boldsymbol{x},y) \equiv \max_{m\neq y}\max(0,1-(\boldsymbol{w}_y-\boldsymbol{w}_m)^T\boldsymbol{x}).$$

- We hope the decision value of $\boldsymbol{x}_i$ by the model $\boldsymbol{w}_{y_i}$ is larger than others
- Prediction: same as one-against-the rest

$$\arg\max_j \ (\boldsymbol{w}_j)^T\boldsymbol{x}$$

Discussion

- 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

- Maximum Entropy: a generalization of logistic regression for multi-class problems
- It is widely applied by NLP applications.
- Conditional probability of label y given data $\boldsymbol{x}$.

$$P(y|\boldsymbol{x}) \equiv \frac{\exp(\boldsymbol{w}_y^T \boldsymbol{x})}{\sum_{m=1}^{k} \exp(\boldsymbol{w}_m^T \boldsymbol{x})},$$

Maximum Entropy (Cont'd)

- We then minimizes regularized negative log-likelihood.

$$\min_{\boldsymbol{w}_1,\dots,\boldsymbol{w}_m} \quad \frac{1}{2}\sum_{m=1}^{k}\|\boldsymbol{w}_k\|^2 + C\sum_{i=1}^{l}\xi(\{\boldsymbol{w}_m\}_{m=1}^{k};\boldsymbol{x}_i,y_i),$$

where

$$\xi(\{\boldsymbol{w}_m\}_{m=1}^{k};\boldsymbol{x},y) \equiv -\log P(y|\boldsymbol{x}).$$

Maximum Entropy (Cont'd)

- Is this loss function reasonable?
- If

$$\boldsymbol{w}_{y_i}^T \boldsymbol{x}_i \gg \boldsymbol{w}_m^T \boldsymbol{x}_i, \forall m \neq y_i,$$

  then

$$\xi(\{\boldsymbol{w}_m\}_{m=1}^k; \boldsymbol{x}_i, y_i) \approx 0$$

  That is, no loss
- In contrast, if

$$\boldsymbol{w}_{y_i}^T \boldsymbol{x}_i \ll \boldsymbol{w}_m^T \boldsymbol{x}_i, m \neq y_i,$$

  then $P(y_i|\boldsymbol{x}_i) \ll 1$ and the loss is large.

Features as Functions

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

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

- The earlier probability model is a special case by

$$\boldsymbol{f}(\boldsymbol{x}, y) = \begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \boldsymbol{x} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} \left.\begin{matrix} \\ \\ \\ \\ \\ \\ \\ \end{matrix}\right\} y-1 \in \mathbf{R}^{nk} \text{ and } \boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}_1 \\ \vdots \\ \boldsymbol{w}_k \end{bmatrix}.$$

Outline

Least Square Regression I

- Given training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_l, y_l)$
- Now

$$y_i \in R$$

  is the target value
- Regression: find a function so that

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

Least Square Regression II

- Least square regression:

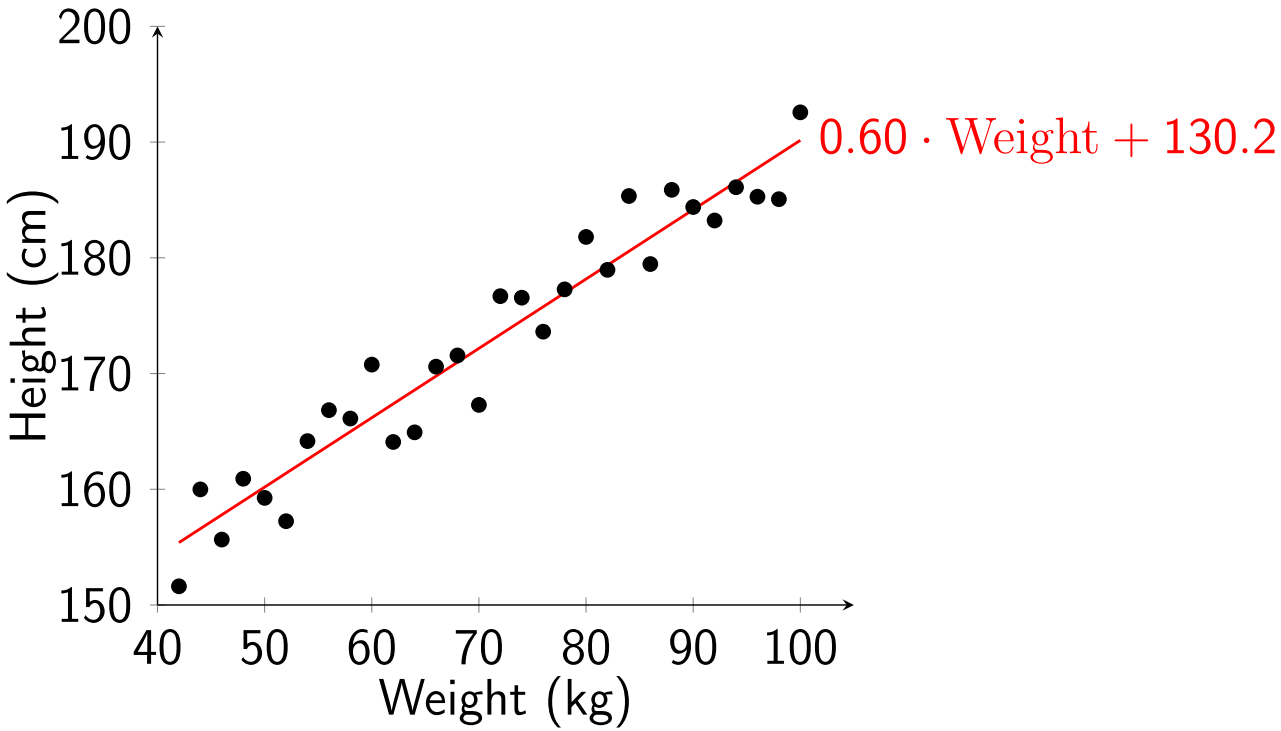
$$\min_{\boldsymbol{w},b} \sum_{i=1}^{l} (y_i - (\boldsymbol{w}^T \boldsymbol{x}_i + b))^2$$

- That is, we model $f(\boldsymbol{x})$ by

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$$

- An example

Least Square Regression III



note: picture is from

http://tex.stackexchange.com/questions/119179/how-to-add-a-regression-line-to-randomly-generated-p

Least Square Regression IV

- This is equivalent to

$$\min_{\boldsymbol{w},b} \quad \sum_{i=1}^{l} \xi(\boldsymbol{w}, b; \boldsymbol{x}_i, y_i)$$

where

$$\xi(\boldsymbol{w}, b; \boldsymbol{x}_i, y_i) = (y_i - (\boldsymbol{w}^T \boldsymbol{x}_i + b))^2$$

Regularized Least Square

- $\xi(\boldsymbol{w}, b; \boldsymbol{x}_i, y_i)$ is a kind of loss function
- We can add regularization.

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

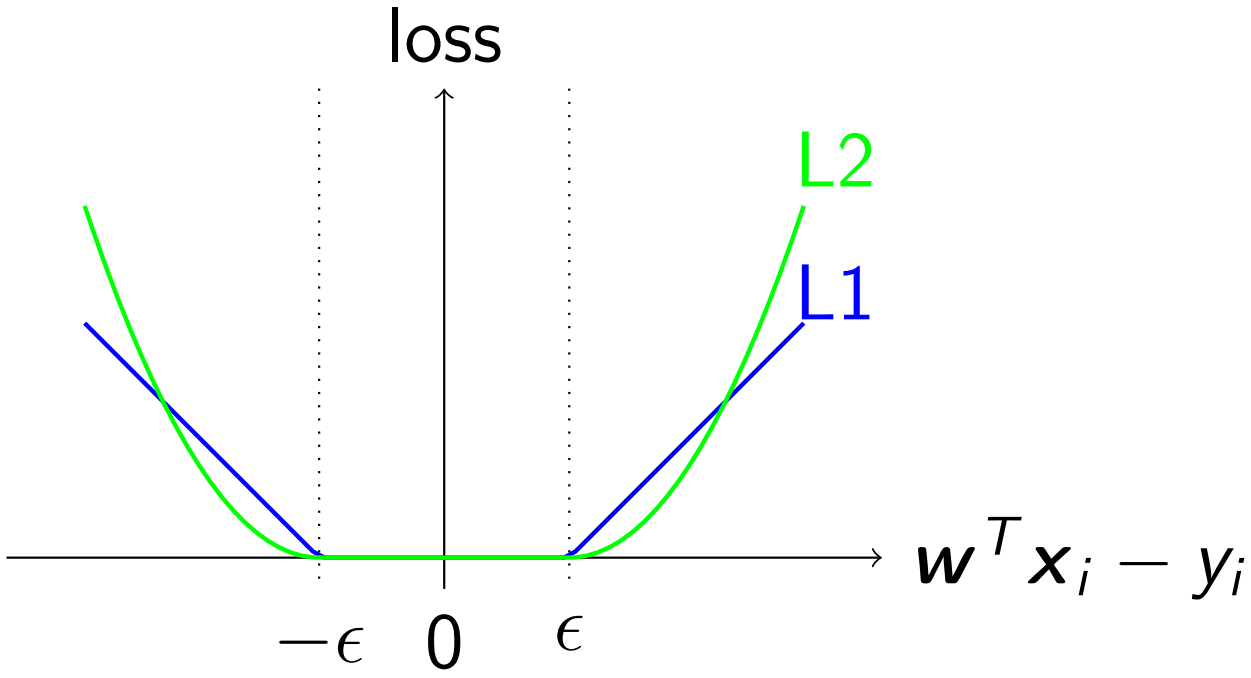
- C is still the regularization parameter
- Other loss functions?

Support Vector Regression I

- ϵ-insensitive loss function (b omitted)

$$\max(|\boldsymbol{w}^T \boldsymbol{x}_i - y_i| - \epsilon, 0)$$
$$\max(|\boldsymbol{w}^T \boldsymbol{x}_i - y_i| - \epsilon, 0)^2$$

Support Vector Regression II



- ϵ: errors small enough are treated as no error
- This make the model more robust (less overfitting the data)

Support Vector Regression III

- One more parameter (ϵ) to decide
- An equivalent form of the optimization problem

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

- This form is similar to the SVM formulation derived earlier

Support Vector Regression IV

- The dual problem is

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

$$\text{subject to} \quad \boldsymbol{e}^T(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0,$$

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

where $Q_{ij} = K(\boldsymbol{x}_i,\boldsymbol{x}_j) \equiv \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$.

Support Vector Regression V

After solving the dual problem,

$$\boldsymbol{w} = \sum_{i=1}^{l}(-\alpha_i + \alpha_i^*)\phi(\boldsymbol{x}_i)$$

and the approximate function is

$$\sum_{i=1}^{l}(-\alpha_i + \alpha_i^*)K(\boldsymbol{x}_i, \boldsymbol{x}) + b.$$

Discussion

- SVR and least-square regression are very related
- Why people more commonly use l2 (least-square) rather than l1 losses?
- Easier because of differentiability

Outline

One-class SVM I

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

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

One-class SVM II

- Instead of the parameter C is SVM, here the parameter is ν.

$$\boldsymbol{w}^T \phi(\boldsymbol{x}_i) \geq \rho - \xi_i$$

means that we hope most data satisfy

$$\boldsymbol{w}^T \phi(\boldsymbol{x}_i) \geq \rho.$$

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

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \le \alpha_i \le 1/(\nu l), i = 1, \ldots, l, \\ & \boldsymbol{e}^T \boldsymbol{\alpha} = 1, \end{aligned}$$

  where $Q_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$.

- The decision function is

$$\operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) - \rho\right).$$

One-class SVM IV

- 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

$$\boldsymbol{e}^T \boldsymbol{\alpha} \leq 1/\nu < 1$$

violates the linear constraint.

- Clearly, a larger ν means we don't need to push ξ_i to zero $\Rightarrow$ more data are considered as outliers

Support Vector Data Description (SVDD) I

- SVDD is another technique to identify outliers (Tax and Duin, 2004)

$$\begin{aligned} \min_{R,\mathbf{a},\boldsymbol{\xi}} \quad & R^2 + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & \|\phi(\boldsymbol{x}_i) - \mathbf{a}\|^2 \le R^2 + \xi_i, i = 1,\ldots,l, \\ & \xi_i \ge 0, i = 1,\ldots,l, \end{aligned}$$

Support Vector Data Description (SVDD) II

- We obtain a hyperspherical model characterized by the center $\mathbf{a}$ and the radius R.
- A test instance $\boldsymbol{x}$ is detected as an outlier if

$$\|\phi(\boldsymbol{x}) - \mathbf{a}\|^2 > R^2.$$

Support Vector Data Description (SVDD) III

- The dual problem

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - \sum_{i=1}^{l} \alpha_i Q_{i,i} \\ \text{subject to} \quad & \boldsymbol{e}^T \boldsymbol{\alpha} = 1, \\ & 0 \le \alpha_i \le C, i = 1, \ldots, l, \end{aligned} \tag{13}$$

- This dual problem is very close to that of one-class SVM

Support Vector Data Description (SVDD) IV

- Consider a scaled version of one-class SVM dual

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \le \alpha_i \le 1, \qquad i = 1, \dots, l, \\ & \boldsymbol{e}^T \boldsymbol{\alpha} = \nu l. \end{aligned}$$

- If Gaussian kernel is used,

$$Q_{i,i} = e^{-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_i\|^2} = 1$$

and the two dual problems are equivalent

Discussion

- For unsupervised settings, evaluation is very difficult
- Usually the evaluation is by a subjective way

Outline

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

$$K_{ij} = e^{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/4} \begin{cases} = 1 & \text{if } i = j, \\ \to 0 & \text{if } i \neq j. \end{cases}$$

  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

$$\phi(\boldsymbol{x}_1) = [1, 0, \ldots, 0]^T$$
$$\vdots$$
$$\phi(\boldsymbol{x}_l) = [0, \ldots, 0, 1]^T$$

  then

$$K = I$$

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

Overfitting

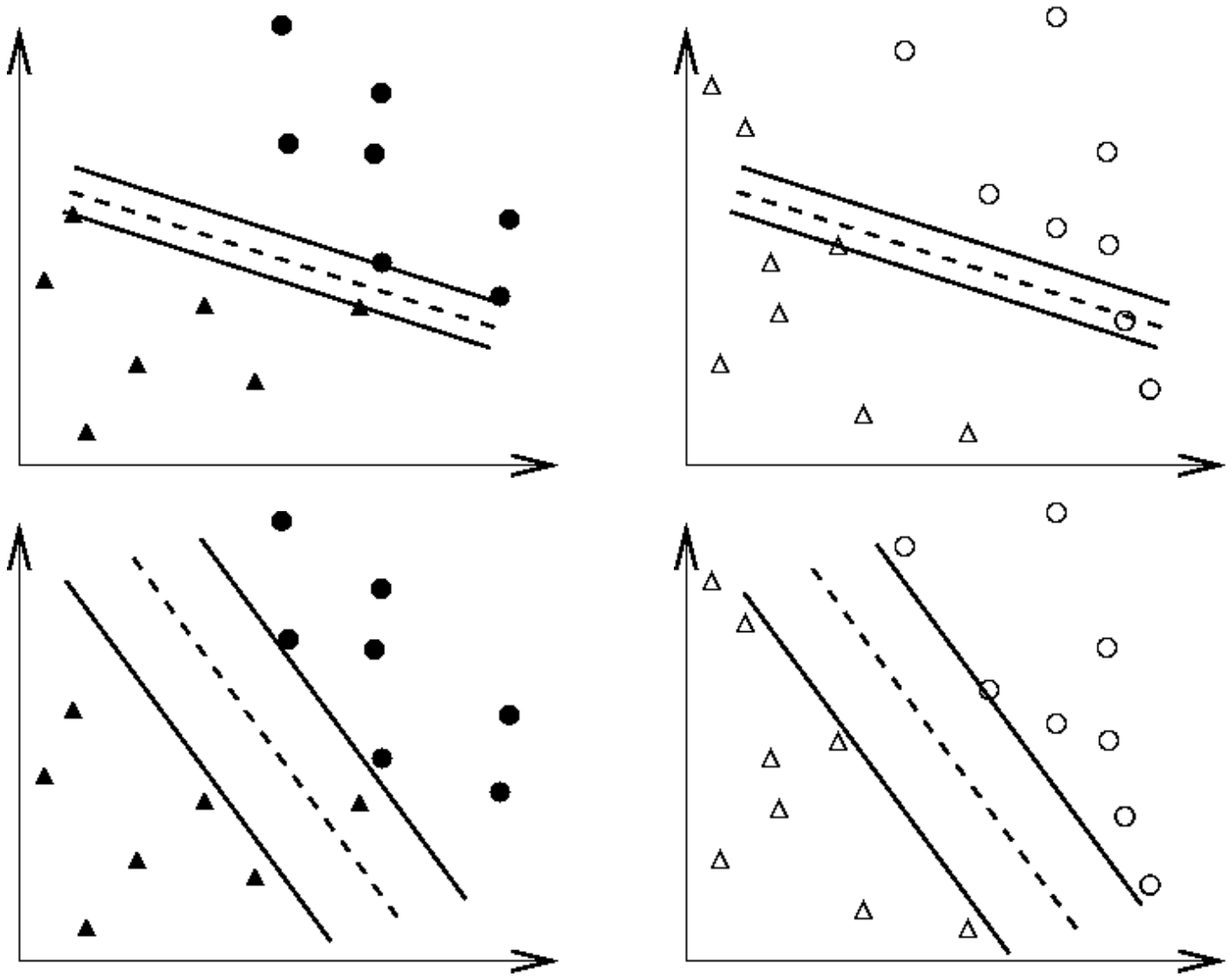
- See the illustration in the next slide
- In theory

  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:

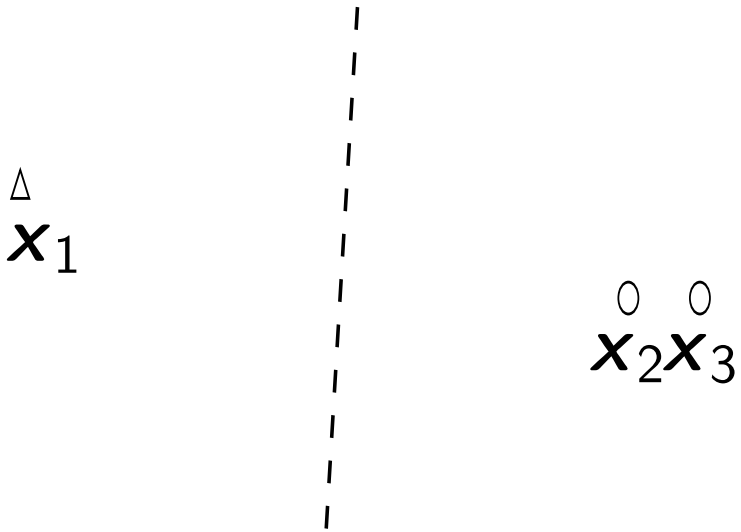
	height	gender
$\boldsymbol{x}_1$	150	F
$\boldsymbol{x}_2$	180	M
$\boldsymbol{x}_3$	185	M

and

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

Data Scaling (Cont'd)

- The separating hyperplane almost vertical



- Strongly depends on the first attribute; but second may be also important

Data Scaling (Cont'd)

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

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

We need to use same factors on training and testing

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

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.m
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.sca
Accuracy = 100% (3089/3089)
```

- 100% training accuracy but

```
$./svm-predict svmguide1.t.scale svmguide1.s
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:

$$\gamma \text{ of } e^{-\gamma\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2}$$

$$a, b, d \text{ of } (\boldsymbol{x}_i^T\boldsymbol{x}_j/a+b)^d$$

- How to select them so performance is better?

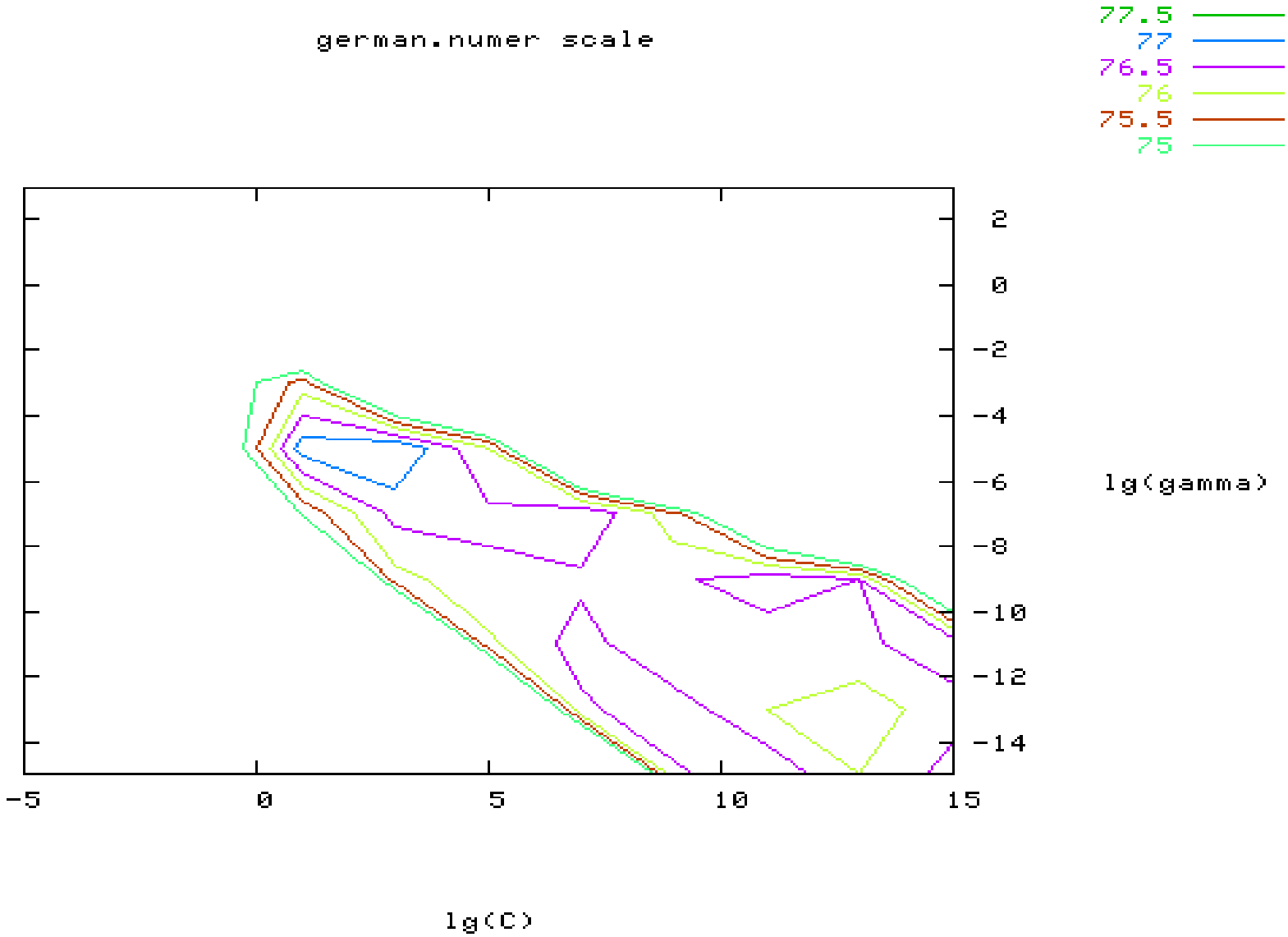
Performance Evaluation

- Available data $\Rightarrow$ 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.sca
$./svm-scale -r range3 svmguide3.t > svmguide3.t
$./svm-train svmguide3.scale
$./svm-predict svmguide3.t.scale svmguide3.scale
```

→ Accuracy = 12.1951%

Example of Parameter Selection (Cont'd)

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

```
$ python grid.py svmguide3.scale
...
128.0 0.125 84.8753
```

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

Train and predict using the obtained parameters

```
$ ./svm-train -c 128 -g 0.125 svmguide3.scale
$ ./svm-predict svmguide3.t.scale svmguide3.scal
```

$\rightarrow$ 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:

$$(\boldsymbol{x}_i^T \boldsymbol{x}_j / a + b)^d$$

  Numerical difficulties: $(< 1)^d \to 0, (> 1)^d \to \infty$

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

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.s
$ python easy.py svmguide4.scale svmguide4.t.sca
Accuracy = 69.2308% (216/312) (classification)
```

The accuracy is low even after parameter selection

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

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

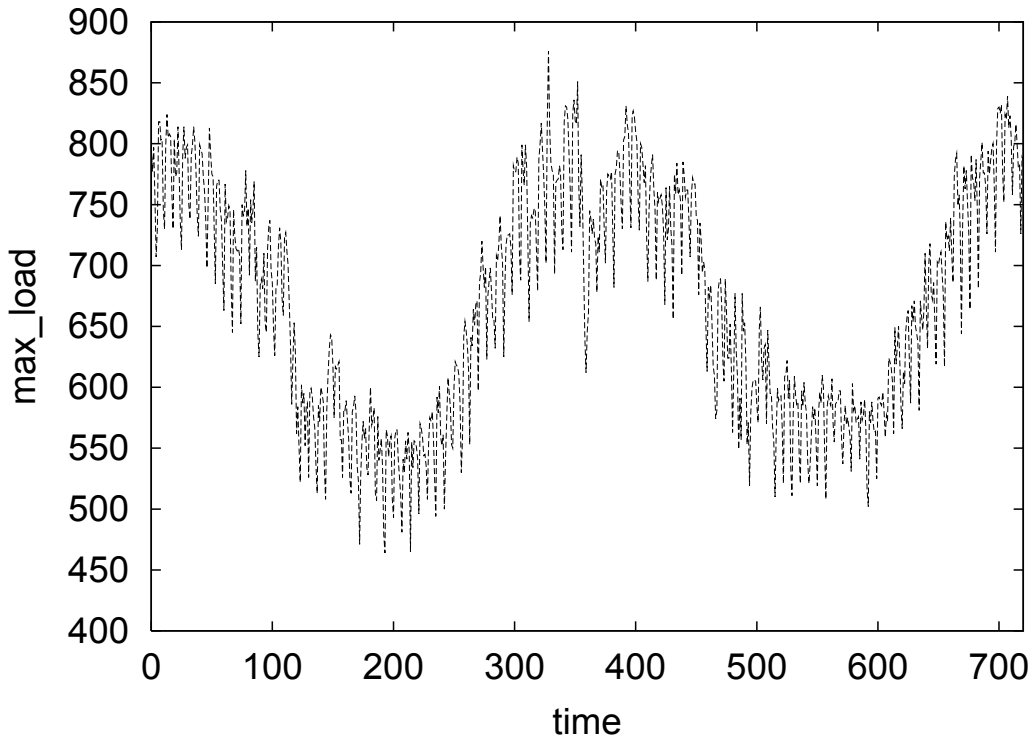
- Given $(\cdots, y_{t-\Delta}, \cdots, y_{t-1}, y_t, \cdots, y_l)$ as training series
- Generate training data:
  $(y_{t-\Delta}, \cdots, y_{t-1})$ as attributes (features) of $\boldsymbol{x}_i$
  y_t as the target value $\boldsymbol{x}_i$
- One-step ahead prediction
- Prediction:
  Starting from the last segment

$$(y_{l-\Delta+1}, \ldots, y_l) \rightarrow \hat{y}_{l+1}$$

  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



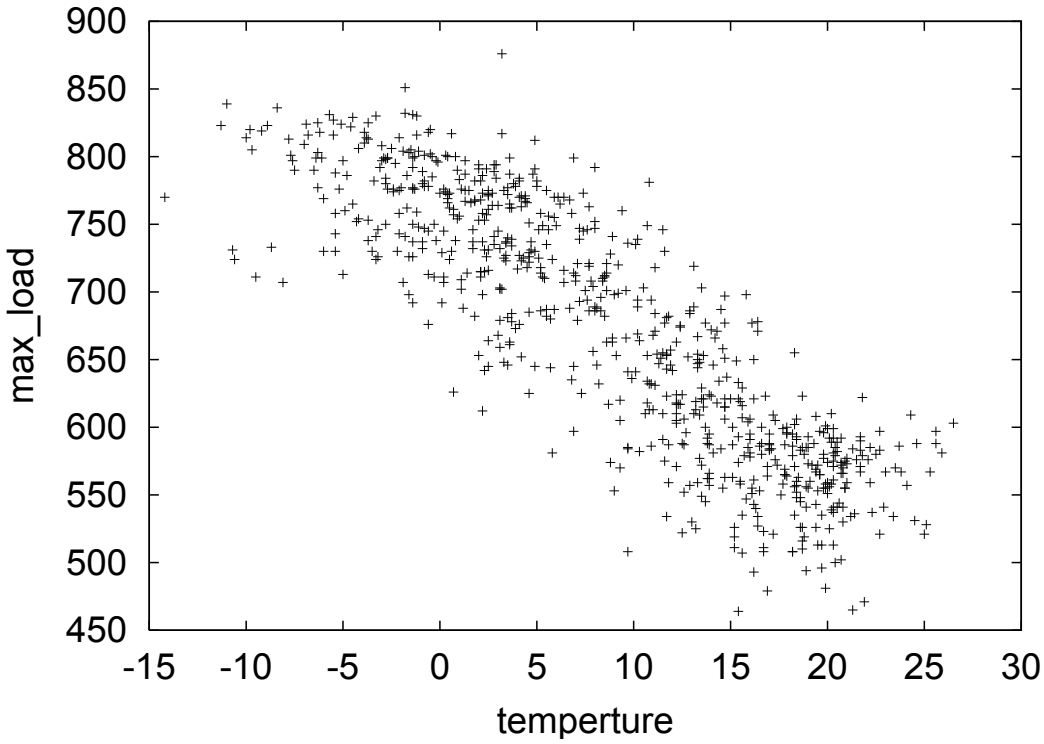
Data Analyses II

- 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

Data Analyses III

- Most early work on short-term prediction:
  Temperature available



Data Analyses IV

- 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 a segment (a vector) for predicting the next value

  Encoding: contents of a segment
- The simplest:

  Each segment: load of the previous Δ 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

$$\begin{pmatrix} y_{t-\Delta} \\ T_{t-\Delta} \end{pmatrix}, \cdots, \begin{pmatrix} y_{t-1} \\ T_{t-1} \end{pmatrix} \xrightarrow{\text{SVM1}} y_t$$

$$T_{t-\Delta}, \cdots, T_{t-1} \xrightarrow{\text{SVM2}} T_t$$

Model Selection I

- Parameters and features
  - Δ: for both approaches
  - Local model: # of similar segments
  - SVR:
    1. C: cost of error
    2. ϵ: width of the ϵ-insensitive loss
    3. mapping function ϕ
- 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

Summer Data I

- 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)

$$\frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{|y_i|}$$

- Error propagation: larger error later

  Unfair to earlier prediction if MSE is used

Evaluation of Time Series Prediction II

- There are other criteria

Results: Local Model I

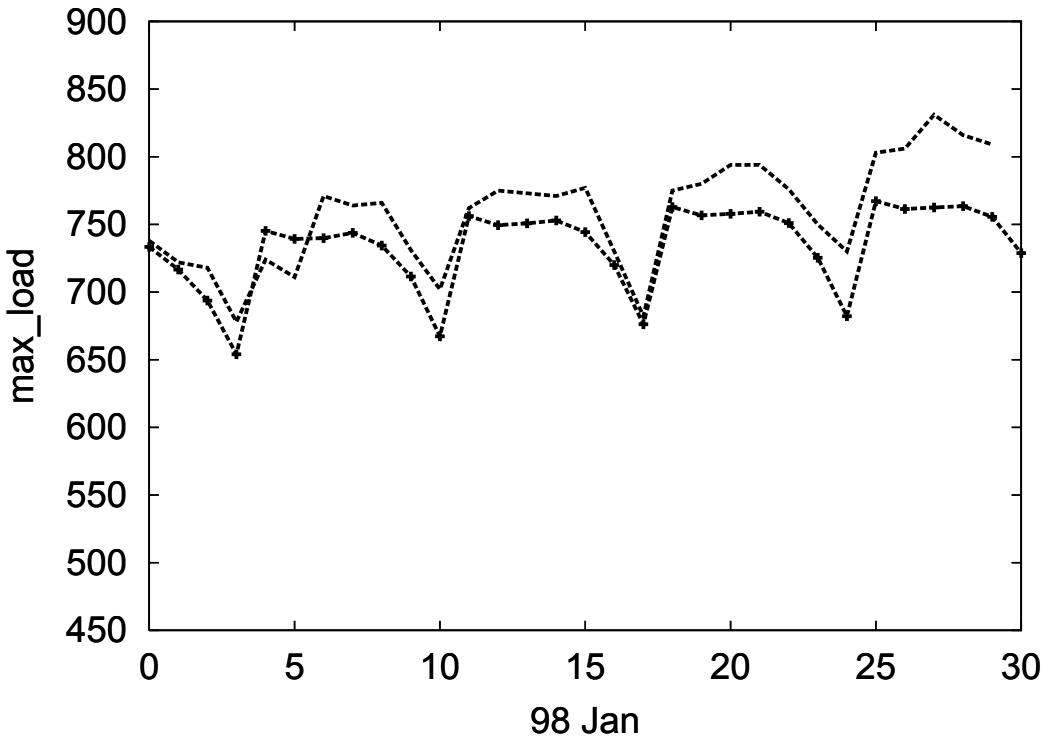
- Validation on different number of segments
- Results in the competition: slightly worse than SVR

Results: SVR I

- Two SVRs: very difficult to predict temperature
- If in one day, temperature suddenly drops or increases

  $\Rightarrow$ 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:

Results: SVR II



- The load of each week is similar
- However, the model manages to find the trend

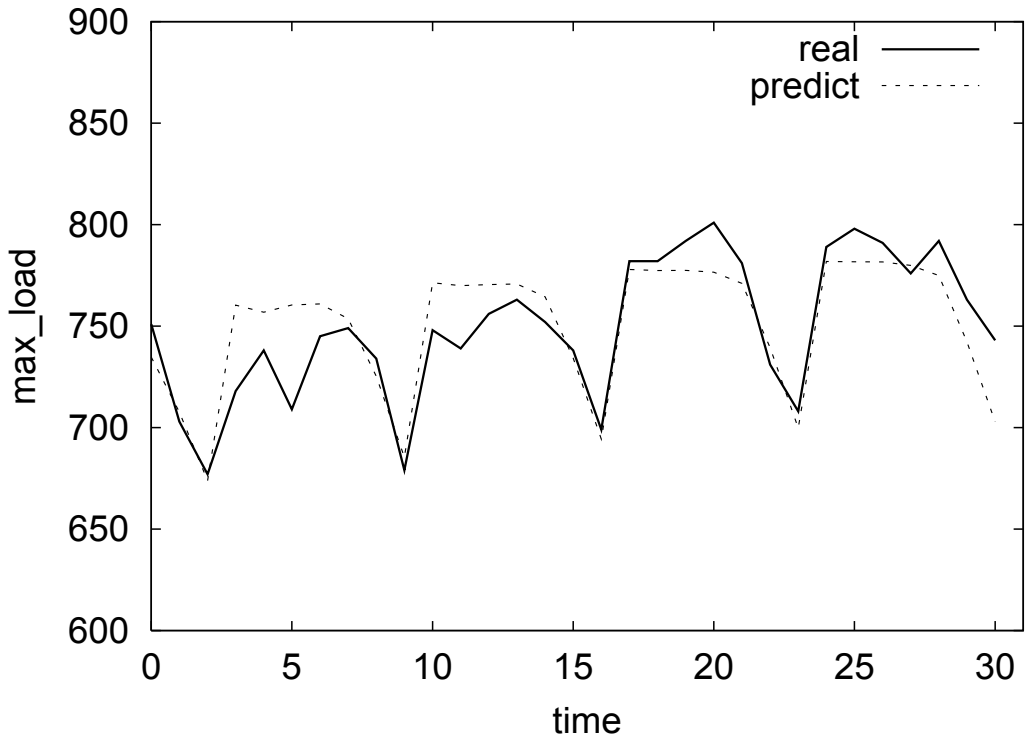
Results: SVR III

- 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 $\max_i(\text{error}_i)$ 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

Outline

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

B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

C. Cortes and V. Vapnik. Support-vector network. *Machine Learning*, 20:273–297, 1995.

K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, (2–3):201–233, 2002.

C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2003. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.

T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 169–184, Cambridge, MA, 1998. MIT Press.

S. S. Keerthi and C.-J. Lin. Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation*, 15(7):1667–1689, 2003.

References II

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines. *Journal of the American Statistical Association*, 99(465):67–81, 2004.

C.-J. Lin. Formulations of support vector machines: a note from an optimization point of view. *Neural Computation*, 13(2):307–317, 2001.

E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 130–136, 1997.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, 1998. MIT Press.

B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

D. M. J. Tax and R. P. W. Duin. Support vector data description. *Machine Learning*, 54(1): 45–66, 2004.

J. Weston and C. Watkins. Multi-class support vector machines. In M. Verleysen, editor, *Proceedings of ESANN99*, pages 219–224, Brussels, 1999. D. Facto Press.

M.-C. Yu, T. Yu, S.-C. Wang, C.-J. Lin, and E. Y. Chang. Big data small footprint: The design of a low-power classifier for detecting transportation modes. *Proceedings of the VLDB Endowment*, 7:1429–1440, 2014.