# Optimization, Support Vector Machines, and Machine Learning

**Chih-Jen Lin**
Department of Computer Science
National Taiwan University

Talk at DIS, University of Rome and IASI, CNR, September, 2005

# Outline

- Introduction to machine learning and support vector machines (SVM)
- SVM and optimization theory
- SVM and numerical optimization
- Practical use of SVM
- Talk slides available at

  http://www.csie.ntu.edu.tw/˜cjlin/talks/rome.pdf
- This talk intends to give optimization researchers an overview of SVM research

# What Is Machine Learning?

- Extract knowledge from data
- Classification, clustering, and others

  We focus only on classification here
- Many new optimization issues

# Data Classification

- Given training data in different classes (labels known)

  Predict test data (labels unknown)
- Examples
  - Handwritten digits recognition
  - Spam filtering
- Training and testing

- Methods:
  - Nearest Neighbor
  - Neural Networks
  - Decision Tree
- Support vector machines: another popular method

  Main topic of this talk
- Machine learning, applied statistics, pattern recognition

  Very similar, but slightly different focuses
- As it's more applied, machine learning is a bigger research area than optimization

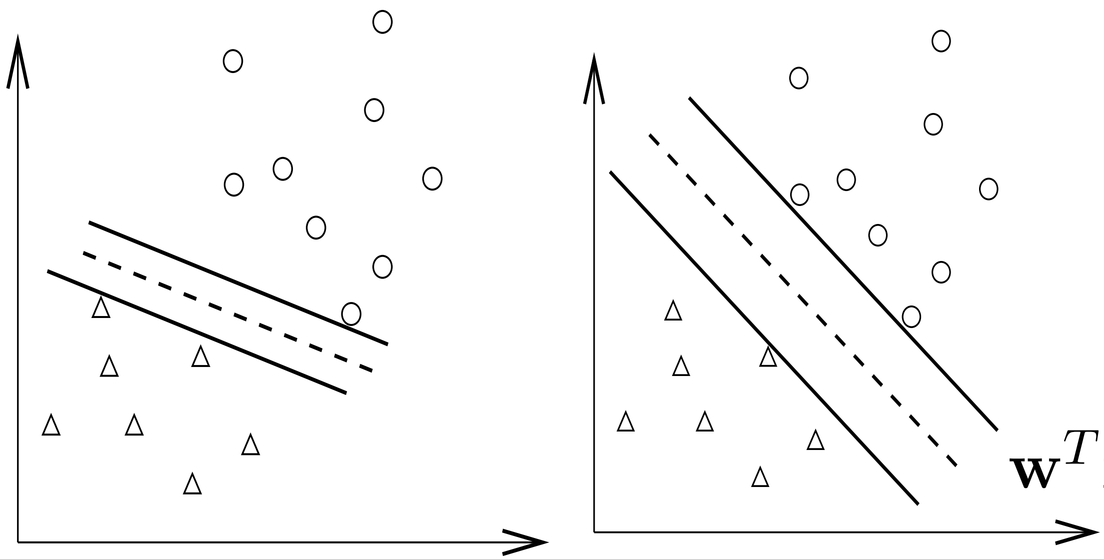
# Support Vector Classification

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

  Define a vector $\mathbf{y}$

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

- A hyperplane which separates all data



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

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

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

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

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

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

- Select $\mathbf{w}, b$ with the maximal margin.

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

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

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

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

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

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

  A nonlinear programming problem

- A 3-D demonstration

  http://www.csie.ntu.edu.tw/˜cjlin/libsvmtools/svmtoy3d

- Notations very different from optimization

  Well, this is unavoidable

# Higher Dimensional Feature Spaces

- Earlier we tried to find a linear separating hyperplane

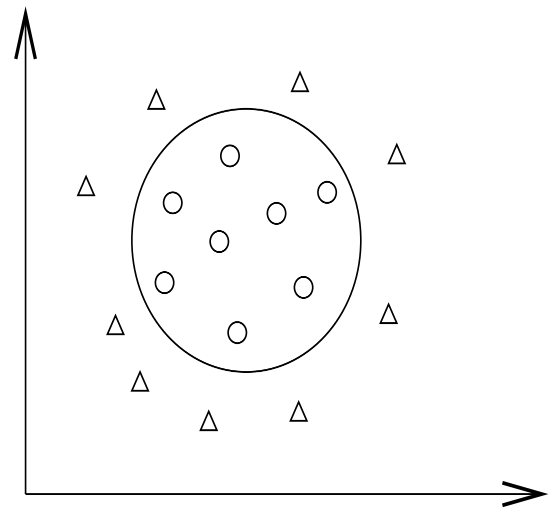
  Data may not be linearly separable

- Non-separable case: allow training errors

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

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

- Nonlinear case: linearly separable in other spaces ?



- Higher dimensional ( maybe infinite ) feature space

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

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

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

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

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

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

# Finding the Decision Function

- $\mathbf{w}$: a vector in a high dimensional space

  $\Rightarrow$ maybe infinite variables
- The dual problem

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

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

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

- SVM problem: primal

- Primal and dual: Discussed later
- A finite problem:

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

  Efficient calculation of high dimensional inner products
- Example: $\mathbf{x}_i \in R^3, \phi(\mathbf{x}_i) \in R^{10}$

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

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

# Kernel Tricks

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

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

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

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

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

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

where

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

# Decision function

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

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

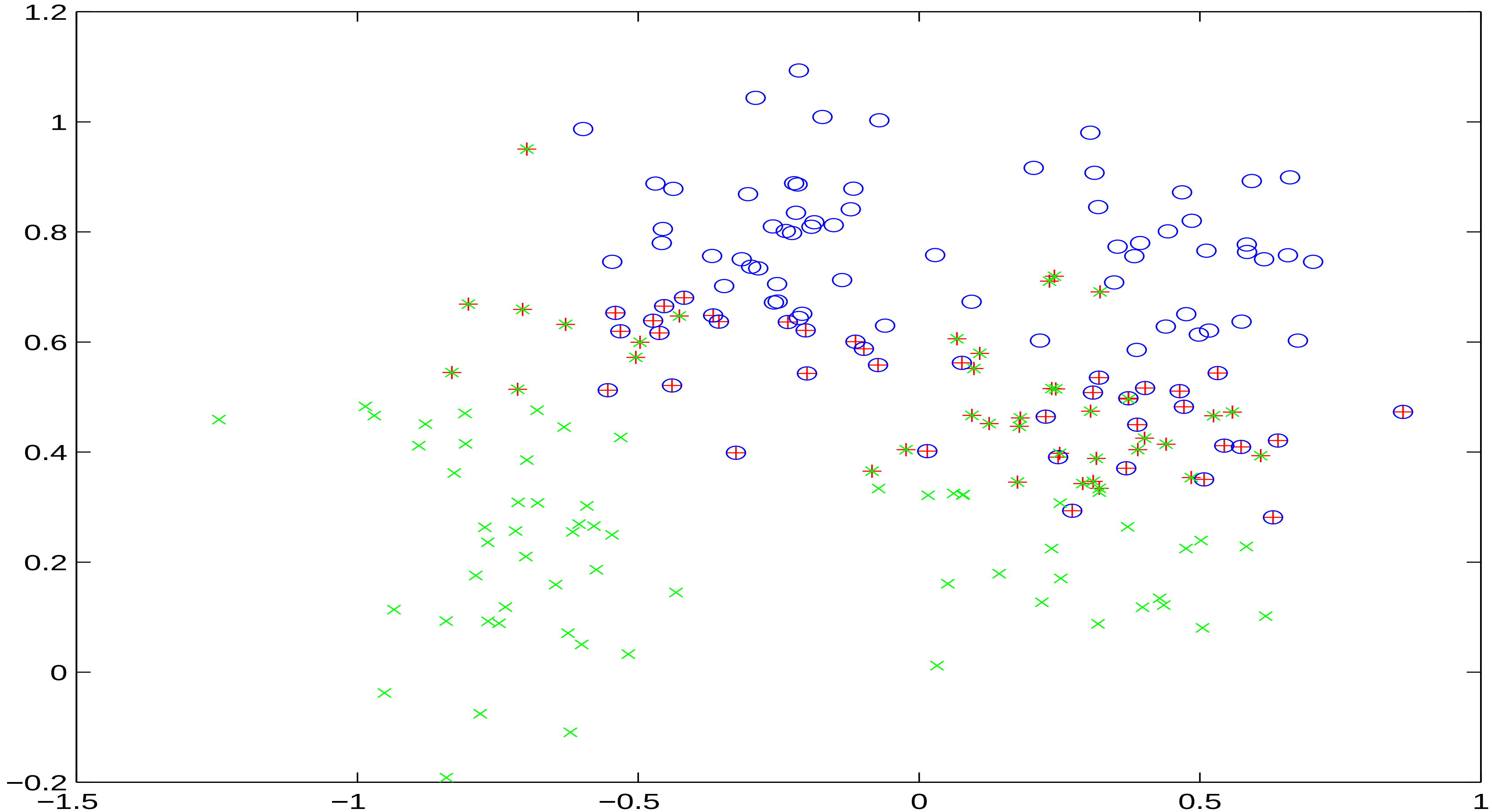
- Decision function

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

- No need to have $\mathbf{w}$
- $> 0$: 1st class, $< 0$: 2nd class
- Only $\phi(\mathbf{x}_i)$ of $\alpha_i > 0$ used

$\alpha_i > 0 \Rightarrow$ support vectors

# Support Vectors: More Important Data



# Is Kernel Really Useful?

- Training data mapped to be linearly independent

  $\Rightarrow$ separable
- Except this, we know little in high dimensional spaces
- Selection is another issue

  On the one hand, very few general kernels

  On the other hand, people try to design kernels specific to applications
- Overall this may be the weakest point of SVM

# SVM and Optimization

- Dual problem is essential for SVM

  There are other optimization issues in SVM

- But, things are not that simple

  If SVM isn't good, useless to study its optimization issues

# Optimization in ML Research

- Everyday there are new classification methods
- Most are related to optimization problems
- Most will never be popular
- Things optimization people focused (e.g., convergence rate) may not be that important for ML people

  More examples later

- In machine learning

  The use of optimization techniques sometimes not rigorous

- Usually an optimization algorithm
  1. Strictly decreasing
  2. Convergence to a stationary point
  3. Convergence rate

- In some ML papers, 1 even does not hold
- Some wrongly think 1 and 2 the same

# Status of SVM

- Existing methods:

  Nearest neighbor, Neural networks, decision trees.
- SVM: similar status (competitive but may not be better)
- In my opinion, after careful data pre-processing

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

- So SVM has survived as a ML method
- There are needs to seriously study its optimization issues

# SVM and Optimization Theory

# A Primal-Dual Example

- Let us have an example before deriving the dual
  To check the primal dual relationship:

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

- Two training data in $R^1$:


- What is the separating hyperplane ?

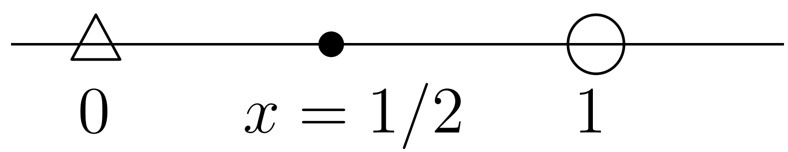
# Primal Problem

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

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

- $-b \geq 1$ and $w \geq 1 - b \geq 2$.
- We are minimizing $\frac{1}{2}w^2$

  The smallest is $w = 2$.
- $(w, b) = (2, -1)$ optimal solution.
- The separating hyperplane $2x - 1 = 0$



# Dual Problem

- Formula without penalty parameter $C$

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

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

- Get the objective function

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

- Objective function

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

- Constraints

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

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

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

- Smallest value at $\alpha_1 = 2$.

  $\alpha_2 = 2$ as well
- $[2, 2]^T$ satisfies $0 \leq \alpha_1$ and $0 \leq \alpha_2$

  Optimal
- Primal-dual relation

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

# SVM Primal and Dual

- Standard SVM (Primal)

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

- $\mathbf{w}$: huge (maybe infinite) vector variable
- Practically we solve dual, a different but related problem

- Dual problem

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

- $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ available
- $\boldsymbol{\alpha}$: $l$ variables; finite

# Primal Dual Relationship

- At optimum

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

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

  where $\mathbf{e} = [1, \ldots, 1]^T$.

- Primal objective value $= -$Dual objective value
- How does this dual come from ?

# Derivation of the Dual

- Consider a simpler problem

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

- Its dual

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

# Lagrangian Dual

- Defined as

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

where

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

- Strong duality

$$\min \text{ Primal} = \max_{\boldsymbol{\alpha} \geq 0} (\min_{\mathbf{w}, b} L(\mathbf{w}, b, \boldsymbol{\alpha}))$$

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

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

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

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

  decrease $-b\sum_{i=1}^{l} \alpha_i y_i$ in $L(\mathbf{w}, b, \boldsymbol{\alpha})$ to $-\infty$

- If $\sum_{i=1}^{l} \alpha_i y_i = 0$, optimum of the strictly convex $\frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{l} \alpha_i [y_i(\mathbf{w}^T\phi(\mathbf{x}_i) - 1]$ happens when

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

- Assume $\mathbf{w} \in R^n$. $L(\mathbf{w}, b, \boldsymbol{\alpha})$ rewritten as

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

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

- Thus,

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

- Note that

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

- The dual is

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

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

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

- Dual simplified to

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

subject to $\quad \alpha_i \geq 0, i = 1, \ldots, l,$ and $\mathbf{y}^T\boldsymbol{\alpha} = 0$.

# Karush-Kuhn-Tucker (KKT) conditions

- The KKT condition of the dual:

$$Q\boldsymbol{\alpha} - \mathbf{e} = -b\mathbf{y} + \boldsymbol{\lambda}$$

$$\alpha_i \lambda_i = 0$$

$$\lambda_i \geq 0$$

- The KKT condition of the primal:

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

$$\alpha_i (y_i \mathbf{w}^T \mathbf{x}_i + b y_i - 1) = 0$$

$$\mathbf{y}^T \boldsymbol{\alpha} = 0, \alpha_i \geq 0$$

- Let $\lambda_i = y_i((\mathbf{w}^T \mathbf{x}_i) + b) - 1$,

$$
\begin{aligned}
& (Q\boldsymbol{\alpha} - \mathbf{e} + b\mathbf{y})_i \\
= \;& \sum_{j=1}^{l} y_i y_j \alpha_j \mathbf{x}_i^T \mathbf{x}_j - 1 + b y_i \\
= \;& y_i \mathbf{w}^T \mathbf{x}_i - 1 + y_i b \\
= \;& y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1
\end{aligned}
$$

- The KKT of the primal the same as the KKT of the dual

# More about Dual Problems

- $\mathbf{w}$ may be infinite

  Seriously speaking, infinite programming (Lin, 2001a)
- In machine learning, quite a few think that for any optimization problem

  Lagrangian dual exists
- This is wrong
- Lagrangian duality usually needs
  - Convex programming problems
  - Constraint qualification

- We have them
  - SVM primal is convex
  - Constraints linear
- Why ML people sometimes make such mistakes

  They focus on developing new methods

  It is difficult to show a counter example

# SVM and Numerical Optimization

# Large Dense Quadratic Programming

- $\min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \mathbf{e}^T\boldsymbol{\alpha}$, subject to $\mathbf{y}^T\boldsymbol{\alpha} = 0, 0 \leq \alpha_i \leq C$
- $Q_{ij} \neq 0$, $Q$ : an $l$ by $l$ fully dense matrix
- 30,000 training points: 30,000 variables:

  $(30,000^2 \times 8/2)$ bytes = 3GB RAM to store $Q$: still difficult
- Traditional methods:

  Newton, Quasi Newton cannot be directly applied
- Current methods:
  - Decomposition methods (e.g., (Osuna et al., 1997; Joachims, 1998; Platt, 1998))
  - Nearest point of two convex hulls (e.g., (Keerthi et al., 1999))

# Decomposition Methods

- Working on a few variable each time
- Similar to coordinate-wise minimization
- Working set $B$, $N = \{1, \ldots, l\} \backslash B$ fixed

  Size of $B$ usually $<= 100$
- Sub-problem in each iteration:

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

$$\begin{bmatrix} \mathbf{e}_B^T & (\mathbf{e}_N^k)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_B \\ \boldsymbol{\alpha}_N^k \end{bmatrix}$$

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

# Avoid Memory Problems

- The new objective function

$$\frac{1}{2}\boldsymbol{\alpha}_B^T Q_{BB}\boldsymbol{\alpha}_B + (-\mathbf{e}_B + Q_{BN}\boldsymbol{\alpha}_N^k)^T\boldsymbol{\alpha}_B + \text{ constant}$$

- $B$ columns of $Q$ needed
- Calculated when used

# Decomposition Method: the Algorithm

1. Find initial feasible $\boldsymbol{\alpha}^1$

   Set $k = 1$.

2. If $\boldsymbol{\alpha}^k$ stationary, stop. Otherwise, find working set $B$.

   Define $N \equiv \{1, \ldots, l\} \backslash B$

3. Solve sub-problem of $\boldsymbol{\alpha}_B$:

$$
\begin{aligned}
\min_{\boldsymbol{\alpha}_B} \quad & \frac{1}{2} \boldsymbol{\alpha}_B^T Q_{BB} \boldsymbol{\alpha}_B + (-\mathbf{e}_B + Q_{BN} \boldsymbol{\alpha}_N^k)^T \boldsymbol{\alpha}_B \\
\text{subject to} \quad & 0 \le \alpha_t \le C, t \in B \\
& \mathbf{y}_B^T \boldsymbol{\alpha}_B^T = -\mathbf{y}_N^T \boldsymbol{\alpha}_N^k,
\end{aligned}
$$

4. $\boldsymbol{\alpha}_N^{k+1} \equiv \boldsymbol{\alpha}_N^k$. Set $k \leftarrow k + 1$; goto Step 2.

# Does it Really Work?

- Compared to Newton, Quasi-Newton

  Slow convergence
- However, no need to have very accurate $\boldsymbol{\alpha}$

$$\text{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right)$$

  Prediction not affected much
- In some situations, # support vectors $\ll$ # training points

  Initial $\boldsymbol{\alpha}^1 = 0$, some elements never used
- An example where ML knowledge affect optimization

# Working Set Selection

- Very important

  Better selection $\Rightarrow$ fewer iterations

- But

  Better selection $\Rightarrow$ higher cost per iteration

- Two issues:

  1. Size

     $|B| \nearrow$, # iterations $\searrow$

     $|B| \searrow$, # iterations $\nearrow$

  2. Selecting elements

# Size of the Working Set

- Keeping all nonzero $\alpha_i$ in the working set

  If all SVs included $\Rightarrow$ optimum

  Few iterations (i.e., few sub-problems)

  Size varies

  May still have memory problems
- Existing software

  Small and fixed size

  Memory problems solved

  Though sometimes slower

# Sequential Minimal Optimization (SMO)

- Consider $|B| = 2$ (Platt, 1998)

  $|B| \geq 2$ because of the linear constraint

  Extreme of decomposition methods
- Sub-problem analytically solved; no need to use optimization software

$$
\begin{aligned}
\min_{\alpha_i,\alpha_j} \quad & \frac{1}{2}\begin{bmatrix}\alpha_i & \alpha_j\end{bmatrix}\begin{bmatrix}Q_{ii} & Q_{ij}\\ Q_{ij} & Q_{jj}\end{bmatrix}\begin{bmatrix}\alpha_i\\ \alpha_j\end{bmatrix} + (Q_{BN}\boldsymbol{\alpha}_N^k - \mathbf{e}_B)^T\begin{bmatrix}\alpha_i\\ \alpha_j\end{bmatrix}\\
\text{s.t.} \quad & 0 \leq \alpha_i, \alpha_j \leq C,\\
& y_i\alpha_i + y_j\alpha_j = -\mathbf{y}_N^T\boldsymbol{\alpha}_N^k,
\end{aligned}
$$

- Optimization people may not think this a big advantage

  Machine learning people do: they like simple code
- A minor advantage in optimization

  No need to have inner and outer stopping conditions
- $B = \{i, j\}$
- Too slow convergence?

  With other tricks, $|B| = 2$ fine in practice

# Selection by KKT violation

- $\min_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}),$ subject to $\mathbf{y}^T\boldsymbol{\alpha} = 0, 0 \leq \alpha_i \leq C$
- $\boldsymbol{\alpha}$ stationary if and only if

$$\nabla f(\boldsymbol{\alpha}) + b\mathbf{y} = \boldsymbol{\lambda} - \boldsymbol{\mu},$$
$$\lambda_i\alpha_i = 0, \mu_i(C - \alpha_i) = 0, \lambda_i \geq 0, \mu_i \geq 0, i = 1, \ldots, l,$$

- $\nabla f(\boldsymbol{\alpha}) \equiv Q\boldsymbol{\alpha} - \mathbf{e}$
- Rewritten as

$$\begin{aligned} \nabla f(\boldsymbol{\alpha})_i + by_i &\geq 0 \qquad \text{if } \alpha_i < C, \\ \nabla f(\boldsymbol{\alpha})_i + by_i &\leq 0 \qquad \text{if } \alpha_i > 0. \end{aligned}$$

- Note $y_i = \pm 1$
- KKT further rewritten as

$$
\begin{aligned}
\nabla f(\boldsymbol{\alpha})_i + b \geq 0 \qquad & \text{if } \alpha_i < C, y_i = 1 \\
\nabla f(\boldsymbol{\alpha})_i - b \geq 0 \qquad & \text{if } \alpha_i < C, y_i = -1 \\
\nabla f(\boldsymbol{\alpha})_i + b \leq 0 \qquad & \text{if } \alpha_i > 0, y_i = 1, \\
\nabla f(\boldsymbol{\alpha})_i - b \leq 0 \qquad & \text{if } \alpha_i > 0, y_i = -1
\end{aligned}
$$

- A condition on the range of $b$:

$$
\begin{aligned}
& \max\{-y_t \nabla f(\boldsymbol{\alpha})_t \mid \alpha_t < C, y_t = 1 \text{ or } \alpha_t > 0, y_t = -1\} \\
\leq \; & b \\
\leq \; & \min\{-y_t \nabla f(\boldsymbol{\alpha})_t \mid \alpha_t < C, y_t = -1 \text{ or } \alpha_t > 0, y_t = 1\}
\end{aligned}
$$

- Define

$$I_{\text{up}}(\boldsymbol{\alpha}) \equiv \{t \mid \alpha_t < C, y_t = 1 \text{ or } \alpha_t > 0, y_t = -1\}, \text{ and}$$
$$I_{\text{low}}(\boldsymbol{\alpha}) \equiv \{t \mid \alpha_t < C, y_t = -1 \text{ or } \alpha_t > 0, y_t = 1\}.$$

- $\boldsymbol{\alpha}$ stationary if and only if feasible and

$$\max_{i \in I_{\text{up}}(\boldsymbol{\alpha})} -y_i \nabla f(\boldsymbol{\alpha})_i \leq \min_{i \in I_{\text{low}}(\boldsymbol{\alpha})} -y_i \nabla f(\boldsymbol{\alpha})_i.$$

# Violating Pair

- KKT equivalent to

$$-y_t \nabla f(\boldsymbol{\alpha})_t$$

$$t \in I_{\text{up}}(\boldsymbol{\alpha}) \qquad t \in I_{\text{low}}(\boldsymbol{\alpha})$$

- Violating pair (Keerthi et al., 2001)

$$i \in I_{\text{up}}(\boldsymbol{\alpha}), j \in I_{\text{low}}(\boldsymbol{\alpha}), \text{ and } -y_i \nabla f(\boldsymbol{\alpha})_i > -y_j \nabla f(\boldsymbol{\alpha})_j.$$

- Strict decrease if and only if $B$ has at least one violating pair.

  However, having violating pair not enough for convergence.

# Maximal Violating Pair

- If $|B| = 2$, naturally indices most violate the KKT condition:

$$i \in \arg \max_{t \in I_{\text{up}}(\boldsymbol{\alpha}^k)} -y_t \nabla f(\boldsymbol{\alpha}^k)_t,$$

$$j \in \arg \min_{t \in I_{\text{low}}(\boldsymbol{\alpha}^k)} -y_t \nabla f(\boldsymbol{\alpha}^k)_t,$$

- Can be extended to $|B| > 2$

# Calculating Gradient

- To find violating pairs, gradient maintained throughout all iterations
- Memory problem occur as $\nabla f(\boldsymbol{\alpha}) = Q\boldsymbol{\alpha} - \mathbf{e}$ involves $Q$
- Solved by following tricks
  1. $\boldsymbol{\alpha}^1 = \mathbf{0}$ implies $\nabla f(\boldsymbol{\alpha}^1) = Q \cdot \mathbf{0} - \mathbf{e} = -\mathbf{e}$
     Initial gradient easily obtained
  2. Update $\nabla f(\boldsymbol{\alpha})$ using only $Q_{BB}$ and $Q_{BN}$:

$$
\begin{aligned}
\nabla f(\boldsymbol{\alpha}^{k+1}) &= \nabla f(\boldsymbol{\alpha}^k) + Q(\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^k) \\
&= \nabla f(\boldsymbol{\alpha}^k) + Q_{:,B}(\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^k)_B
\end{aligned}
$$

- Only $|B|$ columns needed per iteration

# Selection by Gradient Information

- Maximal violating pair same as using gradient information

$$\{i,j\} = \arg\min_{B:|B|=2} \text{Sub}(B),$$

where

$$
\begin{aligned}
\text{Sub}(B) \equiv \min_{\mathbf{d}_B} \quad & \nabla f(\boldsymbol{\alpha}^k)_B^T \mathbf{d}_B && \text{(1a)} \\
\text{subject to} \quad & \mathbf{y}_B^T \mathbf{d}_B = 0, \\
& d_t \geq 0, \ \text{if } \alpha_t^k = 0, t \in B, && \text{(1b)} \\
& d_t \leq 0, \ \text{if } \alpha_t^k = C, t \in B, && \text{(1c)} \\
& -1 \leq d_t \leq 1, t \in B. && \text{(1d)}
\end{aligned}
$$

- First considered in (Joachims, 1998)
- Let $\mathbf{d} \equiv [\mathbf{d}_B; \mathbf{0}_N]$, (1a) comes from minimizing

$$
\begin{aligned}
f(\boldsymbol{\alpha}^k + \mathbf{d}) &\approx f(\boldsymbol{\alpha}^k) + \nabla f(\boldsymbol{\alpha}^k)^T \mathbf{d} \\
&= f(\boldsymbol{\alpha}^k) + \nabla f(\boldsymbol{\alpha}^k)_B^T \mathbf{d}_B.
\end{aligned}
$$

First order approximation

- $0 \le \alpha_t \le C$ leads to (1b) and (1c).
- $-1 \le d_t \le 1, t \in B$ avoid $-\infty$ objective value

- Rough explanation connecting to maximal violating pair

$$
\begin{aligned}
& \nabla f(\boldsymbol{\alpha}^k)_i d_i + \nabla f(\boldsymbol{\alpha}^k)_j d_j \\
= \quad & y_i \nabla f(\boldsymbol{\alpha}^k)_i \cdot y_i d_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j \cdot y_j d_j \\
= \quad & (y_i \nabla f(\boldsymbol{\alpha}^k)_i - y_j \nabla f(\boldsymbol{\alpha}^k)_j) \cdot (y_i d_i)
\end{aligned}
$$

- We used $y_i d_i + y_j d_j = 0$
- Find $\{i, j\}$ so that

$$
y_i \nabla f(\boldsymbol{\alpha}^k)_i - y_j \nabla f(\boldsymbol{\alpha}^k)_j \text{ the smallest}, \; y_i d_i = 1, \; y_j d_j = -1
$$

- $y_i d_i = 1$ corresponds to $i \in I_{\text{up}}(\boldsymbol{\alpha}^k)$:

$$
I_{\text{up}}(\boldsymbol{\alpha}) \equiv \{t \mid \alpha_t < C, y_t = 1 \text{ or } \alpha_t > 0, y_t = -1\}
$$

# Convergence: Maximal Violating Pair

- Special case of (Lin, 2001c)
- Let $\bar{\alpha}$ limit of any convergent subsequence $\{\boldsymbol{\alpha}^k\}, k \in \mathcal{K}$.
- If not stationary, $\exists$ a violating pair

$$\bar{i} \in I_{\text{up}}(\boldsymbol{\alpha}), \bar{j} \in I_{\text{low}}(\boldsymbol{\alpha}), \text{ and } -y_{\bar{i}}\nabla f(\bar{\boldsymbol{\alpha}})_{\bar{i}} + y_{\bar{j}}\nabla f(\bar{\boldsymbol{\alpha}})_{\bar{j}} > 0$$

- If $i \in I_{\text{up}}(\bar{\boldsymbol{\alpha}})$, then $i \in I_{\text{up}}(\boldsymbol{\alpha}^k), \forall k \in \mathcal{K}$ large enough

  If $i \in I_{\text{low}}(\bar{\boldsymbol{\alpha}})$, then $i \in I_{\text{low}}(\boldsymbol{\alpha}^k), \forall k \in \mathcal{K}$ large enough
- So

  $\{\bar{i}, \bar{j}\}$ a violating pair at $k \in \mathcal{K}$

- From $k$ to $k+1$:

  $B^k = \{i, j\}$

  $$i \notin I_{\text{up}}(\boldsymbol{\alpha}^{k+1}) \text{ or } j \notin I_{\text{low}}(\boldsymbol{\alpha}^{k+1})$$

  because of optimality of sub-problem

- If we can show

  $$\{\boldsymbol{\alpha}^k\}_{k \in \mathcal{K}} \to \bar{\boldsymbol{\alpha}} \Rightarrow \{\boldsymbol{\alpha}^{k+1}\}_{k \in \mathcal{K}} \to \bar{\boldsymbol{\alpha}},$$

  then $\{\bar{i}, \bar{j}\}$ should not be selected at $k, k+1, \ldots, k+r$

- A procedure showing in finite iterations, it is selected

  Contradiction

# Key of the Proof

- Essentially we proved

In finite iterations, $B = \{\bar{i}, \bar{j}\}$ selected

to have a contradiction

- Can be used to design working sets (Lucidi et al., 2005):

$\exists N > 0$ such that for all $k$, any violating

pair of $\boldsymbol{\alpha}^k$ selected at least once in iterations $k$ to $k+N$

- A cyclic selection

$$\{1,2\},\{1,3\},\ldots,\{1,l\},\{2,3\},\ldots,\{l-1,l\}$$

# Beyond Maximal Violating Pair

- Better working sets?

  Difficult: # iterations $\searrow$ but cost per iteration $\nearrow$

  May not imply shorter training time

- A selection by second order information (Fan et al., 2005)

  As $f$ is a quadratic,

$$
\begin{aligned}
f(\boldsymbol{\alpha}^k + \mathbf{d}) &= f(\boldsymbol{\alpha}^k) + \nabla f(\boldsymbol{\alpha}^k)^T \mathbf{d} + \frac{1}{2}\mathbf{d}^T \nabla^2 f(\boldsymbol{\alpha}^k)\mathbf{d} \\
&= f(\boldsymbol{\alpha}^k) + \nabla f(\boldsymbol{\alpha}^k)_B^T \mathbf{d}_B + \frac{1}{2}\mathbf{d}_B^T \nabla^2 f(\boldsymbol{\alpha}^k)_{BB}\mathbf{d}_B
\end{aligned}
$$

# Selection by Quadratic Information

- Using second order information

$$\min_{B:|B|=2} \text{Sub}(B),$$

$$\begin{aligned}
\text{Sub}(B) \equiv \min_{\mathbf{d}_B} \quad & \frac{1}{2}\mathbf{d}_B^T \nabla^2 f(\boldsymbol{\alpha}^k)_{BB} \mathbf{d}_B + \nabla f(\boldsymbol{\alpha}^k)_B^T \mathbf{d}_B \\
\text{subject to} \quad & \mathbf{y}_B^T \mathbf{d}_B = 0, \\
& d_t \geq 0, \text{ if } \alpha_t^k = 0, t \in B, \\
& d_t \leq 0, \text{ if } \alpha_t^k = C, t \in B.
\end{aligned}$$

- $-1 \leq d_t \leq 1, t \in B$ not needed if $\nabla^2 f(\boldsymbol{\alpha}^k)_{BB} = Q_{BB}$ PD
- Too expensive to check $\binom{l}{2}$ sets

- A heuristic

1. Select

$$i \in \arg\max_t \{-y_t \nabla f(\boldsymbol{\alpha}^k)_t \mid t \in I_{\text{up}}(\boldsymbol{\alpha}^k)\}.$$

2. Select

$$\begin{aligned} j \in \arg\min_t \{\textsf{Sub}(\{i,t\}) \mid t \in I_{\text{low}}(\boldsymbol{\alpha}^k), \\ -y_t \nabla f(\boldsymbol{\alpha}^k)_t < -y_i \nabla f(\boldsymbol{\alpha}^k)_i\}. \end{aligned}$$

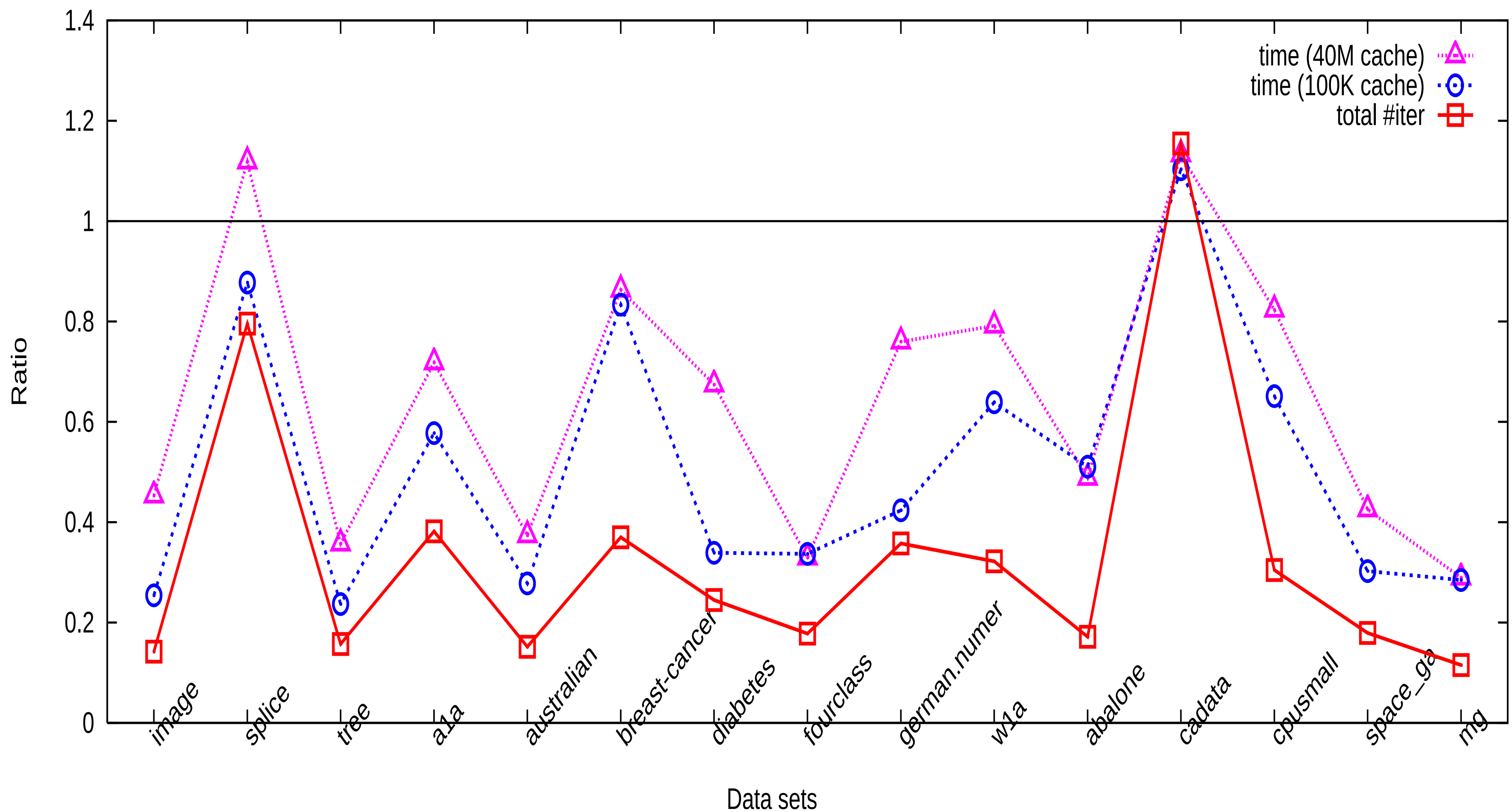
3. Return $B = \{i, j\}$.

- The same $i$ as the maximal violating pair

  Check only $O(l)$ possible $B$'s to decide $j$

# Comparison of Two Selections

- Iteration and time ratio between using quadratic information and maximal violating pair



# Comparing SVM Software/Methods

- In optimization, straightforward to compare two methods
- Now the comparison under one set of parameters may not be enough
- Unclear yet the most suitable way of doing comparisons
- In ours, we check two
  1. Time/total iterations for several parameter sets used in parameter selection
  2. Time/iterations for final parameter set
- Simulate how we use SVM in practice

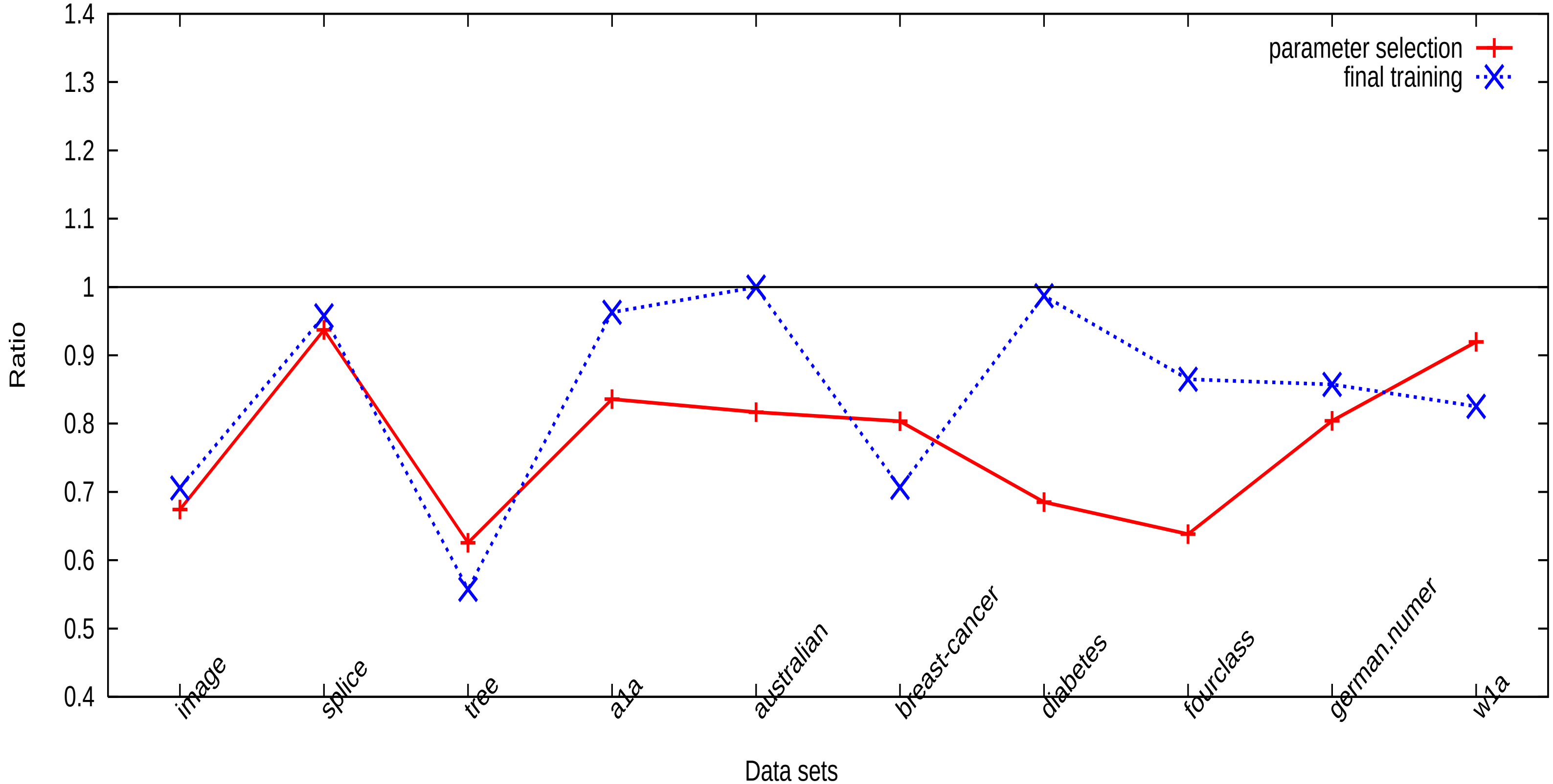
# Issues about the Quadratic Selection

- Asymptotic convergence holds
- Faster convergence than maximal violating pair

  Better approximation per iteration

  But lacks global explanation yet
- What if we check all $\binom{l}{2}$ sets

  Iteration ratio between checking all and checking $O(l)$:




- Fewer iterations, but ratio (0.7 to 0.8) not enough to justify the higher cost per iteration

# Caching and Shrinking

- Speed up decomposition methods
- Caching (Joachims, 1998)

  Store recently used Hessian columns in computer memory
- Example

```
$ time ./libsvm-2.81/svm-train -m 0.01 a4a
11.463s

$ time ./libsvm-2.81/svm-train -m 40 a4a
7.817s
```

- Shrinking (Joachims, 1998)

  Some bounded elements remain until the end

  Heuristically resized to a smaller problem
- After certain iterations, most bounded elements identified and not changed (Lin, 2002)

# Stopping Condition

- In optimization software such conditions are important

  However, don't be surprised if you see no stopping conditions in an optimization code of ML software

  Sometimes time/iteration limits more suitable

- From KKT condition

$$\max_{i \in I_{\mathrm{up}}(\boldsymbol{\alpha})} -y_i \nabla f(\boldsymbol{\alpha})_i \leq \min_{i \in I_{\mathrm{low}}(\boldsymbol{\alpha})} -y_i \nabla f(\boldsymbol{\alpha})_i + \epsilon \tag{2}$$

  a natural stopping condition

# Better Stopping Condition

- Now in out software $\epsilon = 10^{-3}$

  Past experience: ok but sometimes too strict

  At one point we almost changed to $10^{-1}$

- Large $C \Rightarrow$ large $\nabla f(\boldsymbol{\alpha})$ components

  Too strict $\Rightarrow$ many iterations

  Need a relative condition

- A very important issue not fully addressed yet

# Example of Slow Convergence

- Using $C = 1$

```
$./libsvm-2.81/svm-train -c 1 australian_sca
optimization finished, #iter = 508
obj = -201.642538, rho = 0.044312
```

- Using $C = 5000$

```
$./libsvm-2.81/svm-train -c 5000 australian_s
optimization finished, #iter = 35241
obj = -242509.157367, rho = -7.186733
```

- Optimization researchers may rush to solve difficult cases

  That's what I did in the beginning

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

# Finite Termination

- Given $\epsilon$, finite termination under (2) (Keerthi and Gilbert, 2002; Lin, 2002)

  Not implied from asymptotic convergence as

  $$\min_{i \in I_{\text{low}}(\boldsymbol{\alpha})} -y_i \nabla f(\boldsymbol{\alpha})_i - \max_{i \in I_{\text{up}}(\boldsymbol{\alpha})} -y_i \nabla f(\boldsymbol{\alpha})_i$$

  not a continuous function of $\boldsymbol{\alpha}$
- We worry

  $$\alpha_i^k \rightarrow 0 \text{ and } i \in I_{\text{up}}(\boldsymbol{\alpha}^k) \cap I_{\text{low}}(\boldsymbol{\alpha}^k)$$

  causes the program never ends
- Important from optimization point of view

- ML people do not care this much

  Many think finite termination same as asymptotic convergence

- We are careful on such issues in our software
- A good SVM software should

  1. be a rigorous numerical optimization code
  2. serve the need of users in ML and other areas

- Both are equally important

# Issues Not Discussed Here

- $Q$ not PSD
- Solving sub-problems

  Analytic form for SMO (two-variable problem)
- Linear convergence (Lin, 2001b)

$$f(\boldsymbol{\alpha}^{k+1}) - f(\bar{\boldsymbol{\alpha}}) \leq c(f(\boldsymbol{\alpha}^{k}) - f(\bar{\boldsymbol{\alpha}}))$$

  Best worst case analysis

# Practical Use of SVM

# Let Us Try An Example

- A problem from astroparticle physics

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

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

# SVM software: LIBSVM

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

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

# Usage of LIBSVM

- Training

```
Usage: svm-train [options] training_set_file
options:
-s svm_type : set type of SVM (default 0)
        0 -- C-SVC
        1 -- nu-SVC
        2 -- one-class SVM
        3 -- epsilon-SVR
        4 -- nu-SVR
-t kernel_type : set type of kernel function
```

- Testing

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

# Training and Testing

- Training

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

- Testing

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

# What does this Output Mean

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

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

# Why this Fails

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

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

- RBF kernel used

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

Then

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

- $K \rightarrow I$

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

- Optimal solution

$$2 > \boldsymbol{\alpha} = \mathbf{e} - \frac{\mathbf{y}^T\mathbf{e}}{l}\mathbf{y} > 0$$

- $\alpha_i > 0$

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

Zero training error

# Data Scaling

- Without scaling

  Attributes in greater numeric ranges may dominate

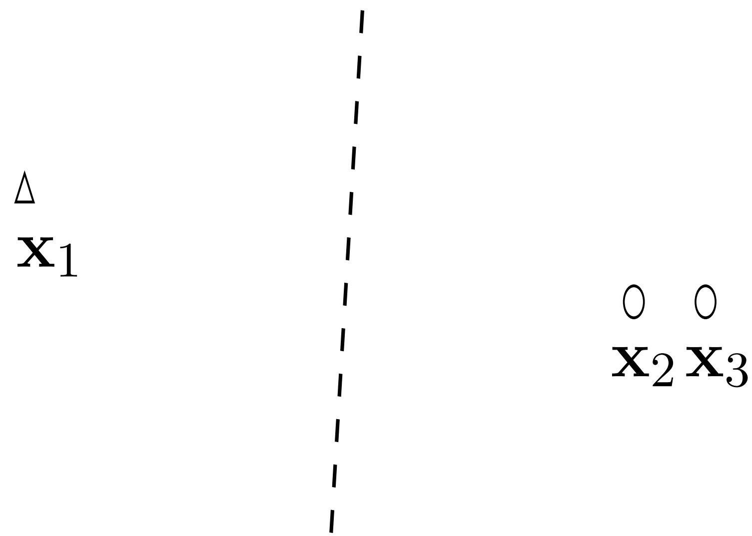
- Example:

| | height | gender |
|---|---|---|
| $\mathbf{x}_1$ | 150 | F |
| $\mathbf{x}_2$ | 180 | M |
| $\mathbf{x}_3$ | 185 | M |

and

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

- The separating hyperplane

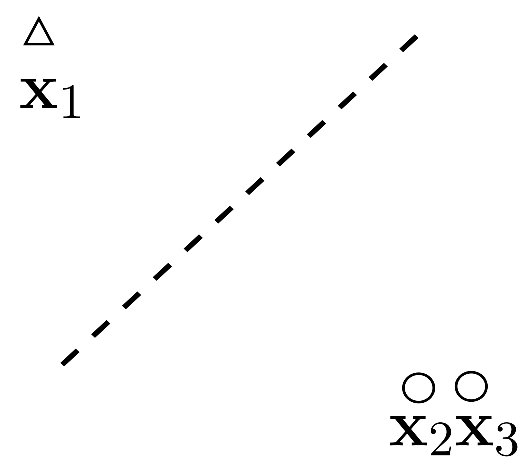



- Decision strongly depends on the first attribute
- What if the second is more important

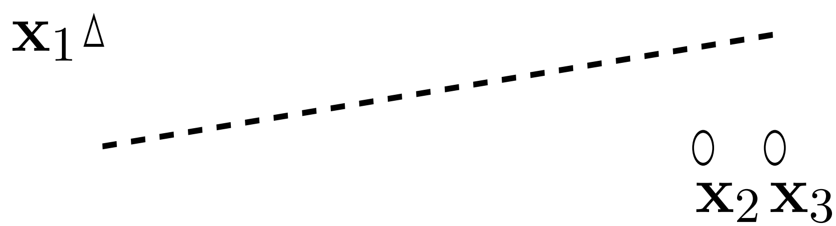
- Linearly scale the first to $[0,1]$ by:

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

- New points and separating hyperplane



- Transformed to the original space,




- The second attribute plays a role
- Scaling generally helps, but not always

# More about Data Scaling

- A common mistake

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

- Same factor on training and testing

```
$./svm-scale -s range1 train.1 > train.1.sca
$./svm-scale -r range1 test.1 > test.1.scale
$./svm-train train.1.scale
$./svm-predict test.1.scale train.1.scale.mo
  test.1.predict
```

$\rightarrow$ Accuracy = 96.15%

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

# More on Training

- Train scaled data and then prediction

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

- Training accuracy now is

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

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

# Different Parameters

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

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

- 100% training accuracy but

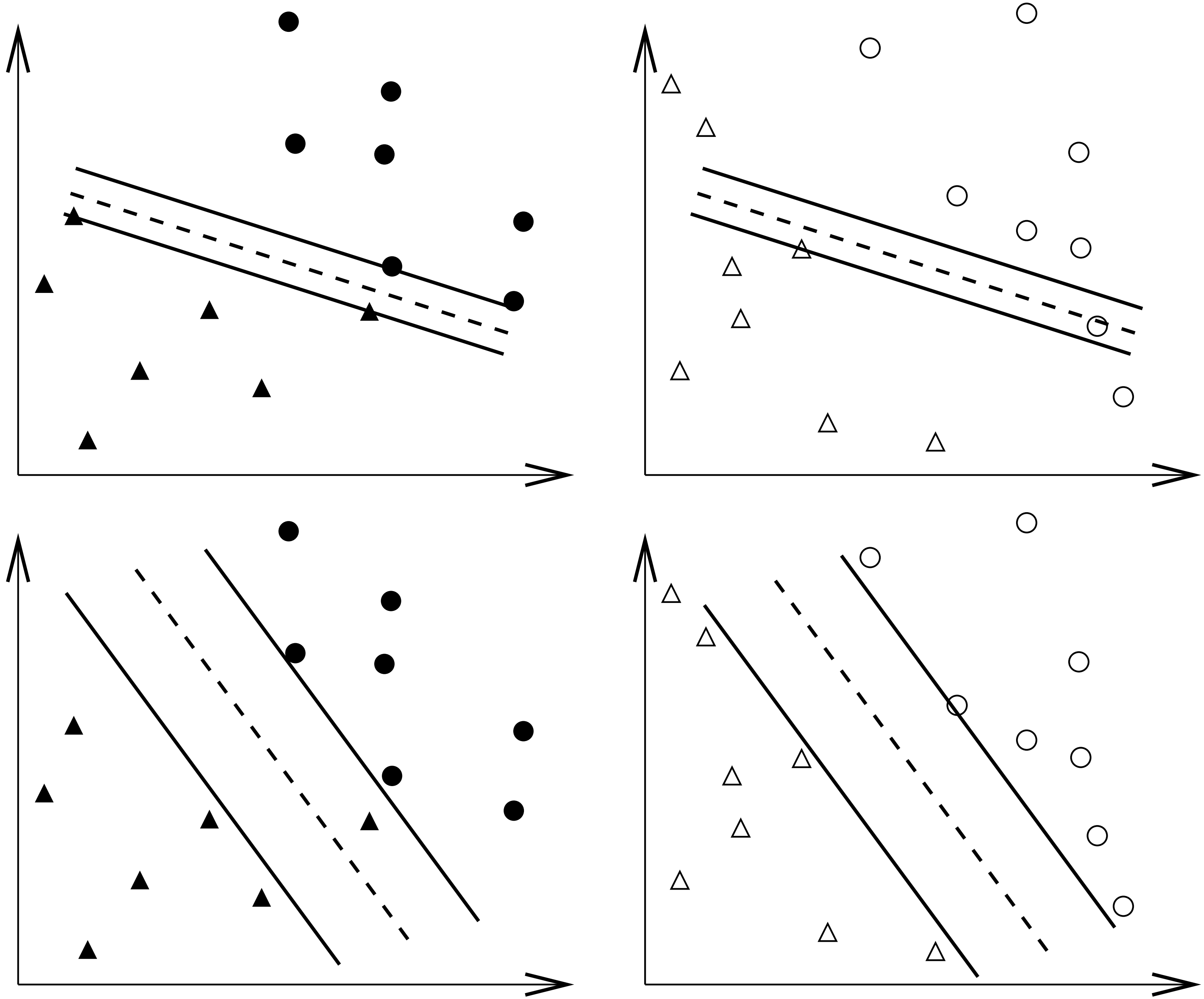
```
$./svm-predict test.1.scale train.1.scale.mo
Accuracy = 82.7% (3308/4000) (classification
```

- Very bad test accuracy
- Overfitting happens

# Overfitting and Underfitting

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

# ● and ▲: training; ○ and △: testing



# Overfitting

- In theory

  You can easily achieve 100% training accuracy

- This is useless
- Surprisingly

  Many application papers did this

# Parameter Selection

- Sometimes important
- Now parameters are

  $C$ and kernel parameters
- Example:

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

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

- How to select them ?

# Performance Evaluation

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

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

Different $\alpha$: different machines

- The expected test error (generalized error)

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

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

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

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

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

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

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

# Performance Evaluation (Cont.)

- In practice

  Available data $\Rightarrow$ training, validation, and (testing)
- Train + validation $\Rightarrow$ model
- $k$-fold cross validation:
  - Data randomly separated to $k$ groups.
  - Each time $k-1$ as training and one as testing
  - Select parameters with highest CV
  - Another optimization problem

# A Simple Procedure

1. Conduct simple scaling on the data
2. Consider RBF kernel $K(\mathbf{x}, \mathbf{y}) = e^{-\gamma \|\mathbf{x}-\mathbf{y}\|^2}$
3. Use cross-validation to find the best parameters $C$ and $\gamma$
4. Use the best $C$ and $\gamma$ to train the whole training set
5. Test

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

  In theory, a minor difference

  No problem in practice
- Just a rough guideline. E.g., scaling hurts sometimes

# Why trying RBF Kernel First

- Linear kernel: special case of RBF (Keerthi and Lin, 2003)

  Leave-one-out cross-validation accuracy of linear the same as RBF under certain parameters

  Related to optimization as well

- Polynomial: numerical difficulties

  $(< 1)^d \rightarrow 0, (> 1)^d \rightarrow \infty$

  More parameters than RBF

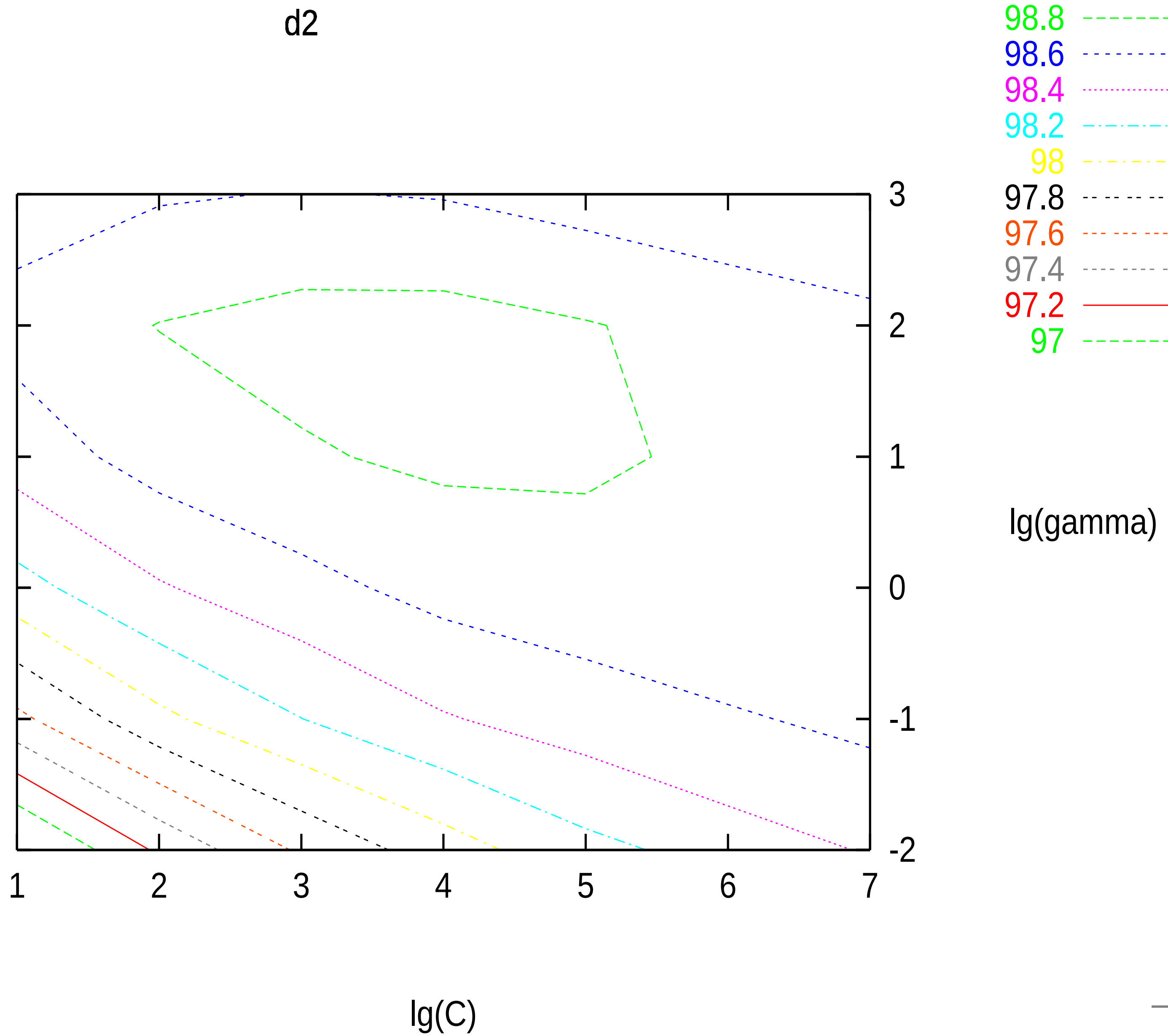
# Parameter Selection in LIBSVM

- grid search + CV

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

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

# Contour of Parameter Selection



# Simple script in LIBSVM

- easy.py: a script for dummies

```
$python easy.py train.1 test.1
Scaling training data...
Cross validation...
Best c=2.0, g=2.0
Training...
Scaling testing data...
Testing...
Accuracy = 96.875% (3875/4000)
```

# Example: Engine Misfire Detection

# Problem Description

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

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

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

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

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

- $x_4(k)$ more important

# Background: Engine Misfire Detection

- How engine works

  Air-fuel mixture injected to cylinder

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

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

# Encoding Schemes

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

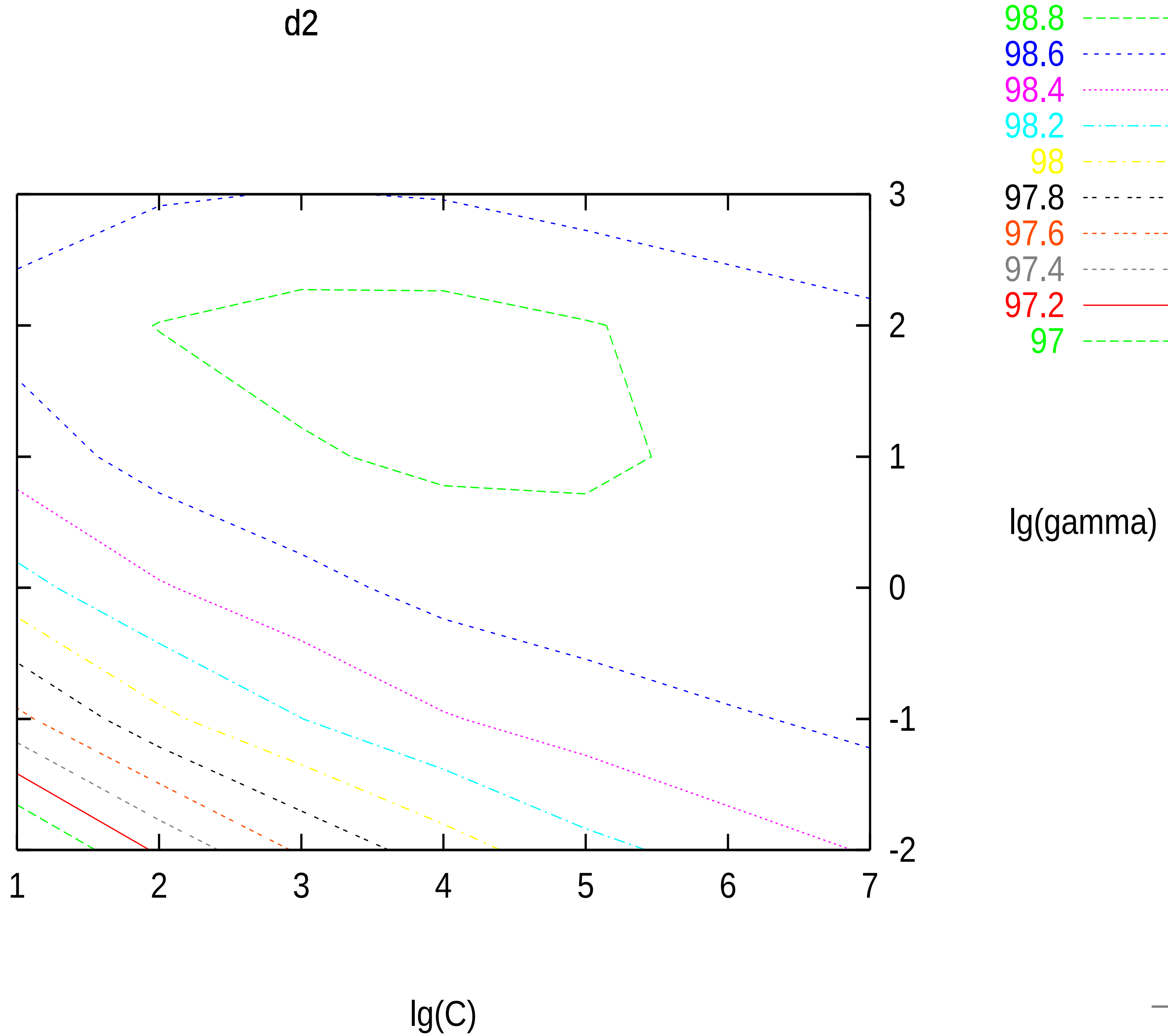
  Including $x_4(k-5), \ldots, x_4(k+4)$ for the $k$th data
- Each training data: 22 attributes

# Training SVM

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

  Data randomly separated to five groups.

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


d2
98.8
98.6
98.4
98.2
98
97.8
97.6
97.4
97.2
97
3
2
1
0
-1
-2
lg(gamma)
1
2
3
4
5
6
7
lg(C)

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

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

# Conclusions

- SVM optimization issues are challenging

  Quite extensively studied

  But better results still possible

- Why working on machine learning?

  It is less mature than optimization

  More new issues

- Many other optimization issues from machine learning

  Need to study things useful for ML tasks

- While we complain ML people's lack of optimization knowledge, we must admit this fact first

  ML people focus on developing methods, so pay less attention to optimization details

- Only if we widely apply solid optimization techniques to machine learning

  then the contribution of optimization in ML can be recognized