Support Vector Machines

Chih-Jen Lin
Department of Computer Science
National Taiwan University

Talk at Machine Learning Summer School 2006, Taipei

Outline

- Basic concepts
- SVM primal/dual problems
- Training linear and nonlinear SVMs
- Parameter/kernel selection and practical issues
- Multi-class classification
- Discussion and conclusions

Outline

- Basic concepts
- SVM primal/dual problems
- Training linear and nonlinear SVMs
- Parameter/kernel selection and practical issues
- Multi-class classification
- Discussion and conclusions

Why SVM and Kernel Methods

- SVM: in many cases competitive with existing classification methods

  Relatively easy to use
- Kernel techniques: many extensions

  Regression, density estimation, kernel PCA, etc.

Support Vector Classification

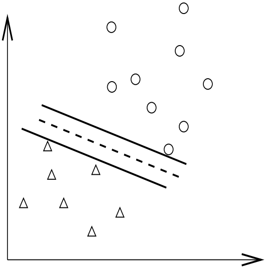
- Training vectors : $\mathbf{x}_i, i = 1, \ldots, l$
- Feature vectors. For example,

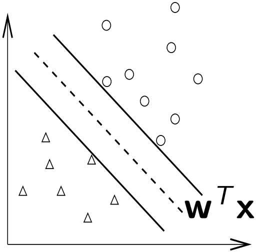
  A patient = [height, weight, ...]
- Consider a simple case with two classes:

  Define an indicator 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





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

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

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

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

Maximal Margin

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

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

$\frac{2}{\|w\|}$

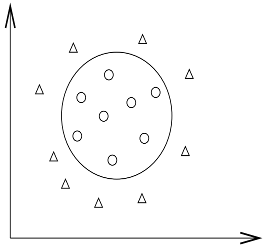
$\frac{1}{\|w\|}$

- A quadratic programming problem [Boser et al., 1992]

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

Data May Not Be Linearly Separable

- An example:



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

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

- Standard SVM [Cortes and Vapnik, 1995]

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

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

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

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

Finding the Decision Function

- $\mathbf{w}$: maybe infinite variables
- The dual problem

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

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

- At optimum

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

- A finite problem: #variables = #training data

Kernel Tricks

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

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

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

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

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

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

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

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

More about Kernels

- How do we know kernels help to separate data?
- In R^l, any l independent vectors

  $\Rightarrow$ linearly separable

$$\begin{bmatrix} (\mathbf{x}^1)^T \\ \vdots \\ (\mathbf{x}^l)^T \end{bmatrix} \mathbf{w} = \begin{bmatrix} +\mathbf{e} \\ -\mathbf{e} \end{bmatrix}$$

- If K positive definite $\Rightarrow$ data linearly separable

  $K = LL^T$.

  Transforming training points to independent vectors in R^l

- So what kind of kernel should I use?
- What kind of functions are valid kernels?
- How to decide kernel parameters?
- Will be discussed later

Decision function

- At optimum

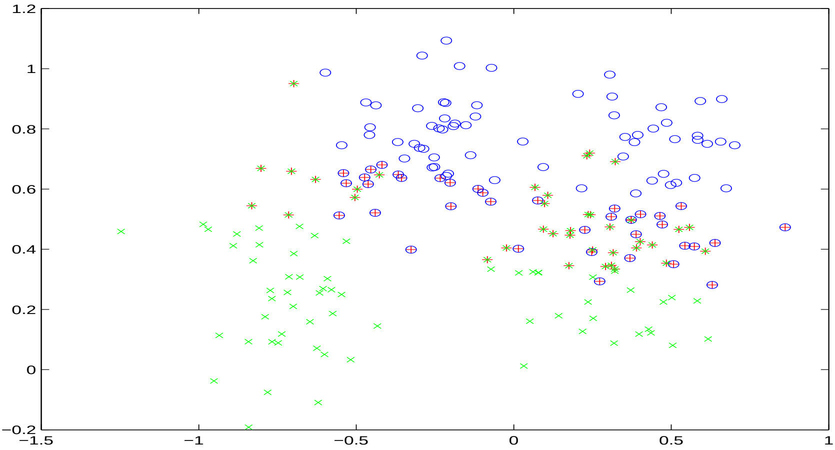
$$\boxed{\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \phi(\mathbf{x}_i)}$$

- Decision function

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

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

Support Vectors: More Important Data



- So we have roughly shown basic ideas of SVM
- A 3-D demonstration
  www.csie.ntu.edu.tw/˜cjlin/libsvmtools/svmtoy3d
- Further references, for example, [Cristianini and Shawe-Taylor, 2000, Schölkopf and Smola, 2002]
- Also see discussion on kernel machines blackboard
  www.kernel-machines.org/phpbb/

Outline

- Basic concepts
- SVM primal/dual problems
- Training linear and nonlinear SVMs
- Parameter/kernel selection and practical issues
- Multi-class classification
- Discussion and conclusions

Deriving the Dual

- Consider the problem without ξ_i

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

- Its dual

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

Lagrangian Dual

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

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 (be careful about this)

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

- 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

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

- 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

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

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

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

More about Dual Problems

- 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

  Convex programming; Constraint qualification
- We have them

  SVM primal is convex; Linear constraints

- Our problems may be infinite dimensional
- Can still use Lagrangian duality

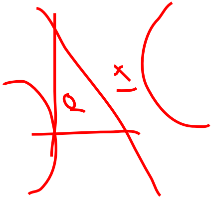
  See a rigorous discussion in [Lin, 2001]

Outline

- Basic concepts
- SVM primal/dual problems
- Training linear and nonlinear SVMs
- Parameter/kernel selection and practical issues
- Multi-class classification
- Discussion and conclusions

Training Nonlinear SVMs

- If using kernels, we solve the dual



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

- Large dense quadratic programming
- $Q_{ij} \neq 0$, 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:
- Traditional methods: Newton, Quasi Newton cannot be directly applied

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 each iteration:

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

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

  Trade time for space

Does it Really Work?

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

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

  Prediction not affected much
- In some situations, $\#$ support vectors $\ll$ $\#$ training points
  Initial $\boldsymbol{\alpha}^1 = 0$, some elements never used
- Machine learning knowledge affects optimization

- An example of training 50,000 instances using LIBSVM

  ```
  $ ./svm-train -m 200 -c 16 -g 4 22features
  optimization finished, #iter = 24981
  Total nSV = 3370
  time    5m1.456s
  ```

- On a Pentium M 1.4 GHz Laptop
- Calculating Q may have taken more than 5 minutes
- $\#\text{SVs} = 3{,}370 \ll 50{,}000$

  A good case where some remain at zero all the time

Issues of Decomposition Methods

- Working set size/selection
- Asymptotic convergence
- Finite termination & stopping conditions
- Convergence rate
- Numerical issues

Optimization researchers are now also interested in these issues

If interested in them, check my talk to optimization researchers in Rome last year:

`http://www.csie.ntu.edu.tw/~cjlin/talks/rome.pdf`

Caching and Shrinking

- Speed up decomposition methods
- Caching [Joachims, 1998]

  Store recently used kernel columns in computer memory
- 100K Cache

  ```
  $ time ./libsvm-2.81/svm-train -m 0.01 a4a
  11.463s
  ```
- 40M Cache

  ```
  $ 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]

  So caching and shrinking are useful

Caching: Issues

- A simple way:
  Store recently used columns
- What if in working set selection,
  deliberately select some indices in cache
- Goal: minimize the total number of columns calculated
- Difficult to connect algorithm and this goal

SVM doesn't Scale Up

Yes, if you use kernels

- Training millions of data is time consuming
- But other nonlinear methods face the same problem
  e.g., kernel logistic regression

Two possibilities

1. Linear SVMs: in some situations, can solve much larger problems
2. Approximation

Training Linear SVMs

- Linear kernel:

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

- At optimum:

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

- Remaining variables: $\mathbf{w}, b$

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

- #variables = #features + 1
- If #features small, easier to solve

- Traditional optimization methods can be applied
- Training time similar to methods such as logistic regression
- What if #features and #instances both large?
  Very challenging
- Some language/document problems are of this type

Decomposition Methods for Linear SVMs

- Could we still solve the dual by decomposition methods?
- Even if #features small
  Slow convergence when C is large

```
$bsvm-train -b 500 -c 500 -t 0 australian_scale
optimization finished, #iter = 260092
obj = -99310.588975, rho = 0.000000
```

- $K_{ij} = \mathbf{x}_i^T \mathbf{x}_j$, rank $\leq$ #features
  positive semi-definite only
- Still a research topic in understanding this

Decomposition Methods for Linear SVMs

- But no need to use large C
- C large enough, $\mathbf{w}$ the same [Keerthi and Lin, 2003]

  decision function the same
- Remember

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \in R^n, \quad b \in R^1$$

$$|\# \text{ of } 0 < \alpha_i < C| \leq n + 1$$

- As C changes, optimal $\boldsymbol{\alpha}$ share many elements at 0 and C

Decomposition Methods for Linear SVMs (Cont'd)

- Warm start very effective [Kao et al., 2004]

  Starting from small C, faster convergence
- Using $C = 1, 2, 4, 8, \ldots$

```
$bsvm-train -c 500 -t 0 australian_scale
optimization finished, #iter = 10087
```

- So decomposition methods can still handle large linear SVMs

Approximations

- #instances large and using nonlinear kernels
  Difficult to solve the dual
- Subsampling
  Simple and often effective
- From this many more advanced techniques
- E.g., stratified subsampling

Approximations (Cont'd)

- Incremental way: (e.g., [Syed et al., 1999])
  Data $\Rightarrow$ 10 parts
  train 1st part $\Rightarrow$ SVs, train SVs + 2nd part, ...
- Select good points first: KNN or heuristics
  e.g., [Bakır et al., 2005]
- Hierarchical settings (e.g., [Yu et al., 2003])
  Clustering training data to several groups
  SVM models built for each group

Approximations (Cont'd)

- Using only a subset to construct **w**

$$\mathbf{w} = \sum_{i \in B} \alpha_i y_i \phi(\mathbf{x}_i).$$

- Put this into the primal

$$\begin{aligned} \min_{\boldsymbol{\alpha}_B, b, \boldsymbol{\xi}} \quad & \frac{1}{2} \boldsymbol{\alpha}_B^T Q_{BB} \boldsymbol{\alpha}_B + C \sum_{i=1}^{l} \xi_i \\ \text{subject to} \quad & Q_{:,B} \boldsymbol{\alpha}_B + b\mathbf{y} \geq \mathbf{e} - \boldsymbol{\xi} \end{aligned}$$

- Without considering ξ_i, $\#\text{variables} = |B| + 1$

Approximations (Cont'd)

- Selecting B:
  random [Lee and Mangasarian, 2001],
  incremental [Keerthi et al., 2006],
  and many other ways

Approximations (Cont'd)

- All these approaches

  some simple but some sophisticated
- In machine learning, very often

  balance between simplification and performance

Outline

- Basic concepts
- SVM primal/dual problems
- Training linear and nonlinear SVMs
- Parameter/kernel selection and practical issues
- Multi-class classification
- Discussion and conclusions

Let's Try a Practical Example

A problem from astroparticle physics

```
1 1:2.6173e+01 2:5.88670e+01 3:-1.89469e-01 4:1.25122e+02
1 1:5.7073e+01 2:2.21404e+02 3:8.60795e-02  4:1.22911e+02
1 1:1.7259e+01 2:1.73436e+02 3:-1.29805e-01 4:1.25031e+02
1 1:2.1779e+01 2:1.24953e+02 3:1.53885e-01  4:1.52715e+02
1 1:9.1339e+01 2:2.93569e+02 3:1.42391e-01  4:1.60540e+02
1 1:5.5375e+01 2:1.79222e+02 3:1.65495e-01  4:1.11227e+02
1 1:2.9562e+01 2:1.91357e+02 3:9.90143e-02  4:1.03407e+02
```

Training and testing sets available: 3,089 and 4,000

The Story Behind this Data Set

- User:

  `I am using libsvm in a astroparticle physics application .. First, let me congratulate you to a really easy to use and nice package. Unfortunately, it gives me astonishingly bad results...`
- OK. Please send us your data
- I am able to get 97% test accuracy. Is that good enough for you ?
- User:

  `You earned a copy of my PhD thesis`

Training and Testing

Training

```
$./svm-train train.1
optimization finished, #iter = 6131
nSV = 3053, nBSV = 724
Total nSV = 3053
```

Testing

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

nSV and nBSV: number of SVs and bounded SVs $(\alpha_i = C)$.

Why this Fails

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

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

- Most kernel elements:

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

- Some features in rather large ranges

Data Scaling

- Without scaling

  Attributes in greater numeric ranges may dominate

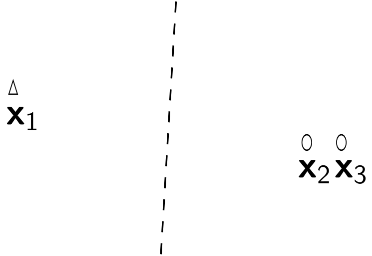
- Example:

	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 almost vertical



- Strongly depends on the first attribute; but second may be also important
- Linearly scale the first to $[0, 1]$ by:

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

- Scaling generally helps, but not always

- Other ways for scaling
- Needed for k Nearest Neighbor, Neural networks as well

  unless the method is scale-invariant

Data Scaling: Same Factors

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.scale
$./svm-scale -r range1 test.1 > test.1.scale
```

After Data Scaling

Train scaled data and then prediction

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

Training accuracy now is

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

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)
```

- 100% training accuracy but

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

- Very bad test accuracy
- Overfitting happens

Overfitting

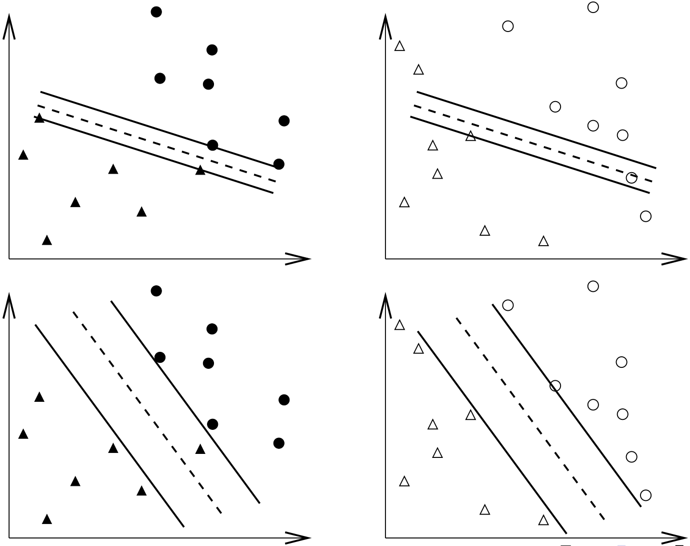
- In theory

  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



Parameter Selection

- Is important
- Now parameters are
  C, kernel parameters
- Example:

$$\gamma \text{ of } e^{-\gamma\|\mathbf{x}_i-\mathbf{x}_j\|^2}$$
$$a, b, d \text{ of } (\mathbf{x}_i^T\mathbf{x}_j/a+b)^d$$

- How to select them?
  So performance better?

Parameter Selection (Cont'd)

- Also how to select kernels?

  e.g., RBF or polynomial
- Moreover, how to select methods?

  e.g., SVM or decision trees?

Performance Evaluation

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

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

  Different α: different machines

- The expected test error (generalized error)

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

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

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

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

- Training errors not important; only test errors count
- $\frac{1}{2}|y_i - f(\mathbf{x}_i, \alpha)|$: loss, choose $0 \leq \eta \leq 1$, with probability at least $1 - \eta$:

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

- A good classification method:

  minimize both terms at the same time

- But $R_{emp}(\alpha) \rightarrow 0$; another term $\rightarrow$ large
- SVM:

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

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

- $\sum_{i=1}^{l}\xi_i$ related to training error
- $\mathbf{w}^T\mathbf{w}/2$ relate to another term: called regularization term
- C: balance between the two

Performance Evaluation (Cont'd)

- In practice
  Available data $\Rightarrow$ training and validation
- Train the training
- Test the validation
- k-fold cross validation:
  Data randomly separated to k groups
  Each time $k-1$ as training and one as testing

- Using CV on training + validation
- Predict testing with the best parameters from CV

CV and Test Accuracy

- If we select parameters so that CV is the highest, Does CV represent future test accuracy ?

  Slightly different
- If we have enough parameters, we can achieve 100% CV as well

  e.g., more parameters than # of training data
- Available data with class labels

  $\Rightarrow$ training, validation, testing

Selecting Kernels

- RBF, polynomial, or others?
  or even combinations
- Two situations:
  Too many kernels complicates the selection
  Design kernels suitable for target applications

Selecting Kernels (Cont'd)

Contradicting but practically ok

- We have few general kernels
  RBF, polynomial, etc. somewhat related
  Beginners' don't have many choices
- On the other hand
  researchers design many special ones
  e.g., string kernels

Selecting Kernels (Cont'd)

- For beginners, use RBF first
- Linear kernel: special case of RBF

  Performance of linear the same as RBF under certain parameters [Keerthi and Lin, 2003]
- Polynomial: numerical difficulties

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

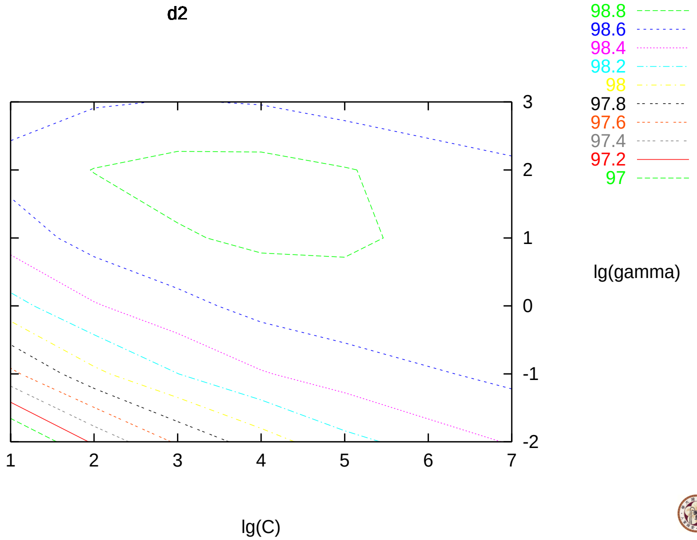
  More parameters than RBF

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 parameter C and γ
4. Use the best C and γ to train the whole training set
5. Test

For beginners only, you can do a lot more

Contour of Parameter Selection



- 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

Efficient Parameter Selection

- CV on grid points may be time consuming
  OK if one or two parameters
- But if more than two?
  E.g., feature scaling:

$$K(\mathbf{x}, \mathbf{y}) = e^{-\sum_{i=1}^{n} \gamma_i (x_i - y_i)^2}$$

  Some features more important
- Still a challenging research issue

- Remember given parameters C and γ, we solve SVM to obtain optimal $\mathbf{w}$ or $\boldsymbol{\alpha}$
- Model a function of parameters

$$\min_{C,\gamma_1,\ldots,\gamma_n} f(\boldsymbol{\alpha}(C,\gamma_1,\ldots,\gamma_n),C,\gamma_1,\ldots,\gamma_n)$$

  But usually non-convex
- The function

  from Bayesian frameworks (e.g., [Chu et al., 2003])

  or

  smoothing CV bound

$$\text{CV}(C,\gamma_1,\ldots,\gamma_n) \leq f(\boldsymbol{\alpha}(C,\gamma_1,\ldots,\gamma_n),C,\gamma_1,\ldots,\gamma_n)$$

- The minimization:

  Gradient-type methods

  or

  global optimization (e.g., genetic algorithms)
- The difficulty:

  Certainly more efforts than one single γ

  But performance may be just similar?

Kernel Combination

- How about using

$$t_1 K_1 + t_2 K_2 + \cdots + t_r K_r,$$

where

$$t_1 + \cdots + t_r = 1$$

as the kernel
- Related to parameter selection

$$t_1 e^{-\gamma_1 \|\mathbf{x}-\mathbf{y}\|} + \cdots + t_r e^{-\gamma_r \|\mathbf{x}-\mathbf{y}\|}$$

If γ_1 good $\Rightarrow$ t_1 close to 1, others close to 0

- [Lanckriet et al., 2004] form a convex

$$f(\boldsymbol{\alpha}(t_1, \ldots, t_r), t_1, \ldots, t_r)$$

  when C is fixed
- Semi-definite programming problem
- But computational cost is also high
- Need more empirical studies

Design Kernels

- Still a research issue

  e.g., in bioinformatics and vision, many new kernels
- But, should be careful if the function is a valid one

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

- For example, any two strings s_1, s_2 we can define edit distance

$$e^{-\gamma \text{edit}(s_1,s_2)}$$

  It's not a valid kernel [Cortes et al., 2003]

Mercer condition

- What kind of K_{ij} can be represented as $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$?
- $K(\mathbf{x},\mathbf{y}) = \phi(\mathbf{x})^T\phi(\mathbf{y})$ if and only if $\forall g$ s.t.

$$\begin{gathered}\int g(\mathbf{x})^2 d\mathbf{x} \text{ finite} \\ \Rightarrow \int K(\mathbf{x},\mathbf{y})g(\mathbf{x})g(\mathbf{y})d\mathbf{x}d\mathbf{y} \geq 0\end{gathered}$$

  A condition developed early last century
- However, still not easy to check

Outline

- Basic concepts
- SVM primal/dual problems
- Training linear and nonlinear SVMs
- Parameter/kernel selection and practical issues
- Multi-class classification
- Discussion and conclusions

Multi-class Classification

- k classes
- One-against-the rest: Train k binary SVMs:

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

- k decision functions

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

$$\vdots$$

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

- Prediction:

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

- Reason: If the 1st class, then we should have

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

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

- 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

More Complicated Forms

- For example,
  [Vapnik, 1998, Weston and Watkins, 1999]:

$$
\begin{aligned}
\min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\sum_{m=1}^{k}\mathbf{w}_m^T\mathbf{w}_m + C\sum_{i=1}^{l}\sum_{m\neq y_i}\xi_i^m \\
& \mathbf{w}_{y_i}^T\phi(\mathbf{x}_i) + b_{y_i} \geq \mathbf{w}_m^T\phi(\mathbf{x}_i) + b_m + 2 - \xi_i^m, \\
& \xi_i^m \geq 0, i = 1,\ldots,l,\ m \in \{1,\ldots,k\}\backslash y_i.
\end{aligned}
$$

  y_i: class of $\mathbf{x}_i$
- kl constraints
- Dual: kl variables; very large

- There are many other methods
- A comparison in [Hsu and Lin, 2002]
- Accuracy similar for many problems
  But 1-against-1 fastest for training

Why 1vs1 Faster in Training

- 1 vs. 1

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

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

  polynomial of the size with degree d
- Their complexities

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

Outline

- Basic concepts
- SVM primal/dual problems
- Training linear and nonlinear SVMs
- Parameter/kernel selection and practical issues
- Multi-class classification
- Discussion and conclusions

Future Directions

I mentioned quite a few. Here are others.

- Better ways to handle unbalanced data
  i.e., some classes few data, some classes a lot
- Multi-label classification
  An instance associated with ≥ 2 labels
  e.g., a document in both politics, sports
- Structural data sets
  An instance may not be a vector
  e.g., a tree from a sentence

Conclusions

- Dealing with data is interesting
  especially if you get good accuracy
- Some basic understandings are essential when applying classification methods
- SVM is a rather mature topic
  but still quite a few interesting research issues

References I

Bakır, G. H., Bottou, L., and Weston, J. (2005).
Breaking svm complexity with cross-training.
In Saul, L. K., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*, pages 81–88. MIT Press, Cambridge, MA.

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

Chu, W., Keerthi, S., and Ong, C. (2003).
Bayesian trigonometric support vector classifier.
Neural Computation, 15(9):2227–2254.

Cortes, C., Haffner, P., and Mohri, M. (2003).
Positive definite rational kernels.
In *Proceedings of the 16th Annual Conference on Learning Theory*, pages 41–56.

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

References II

Cristianini, N. and Shawe-Taylor, J. (2000).
An Introduction to Support Vector Machines.
Cambridge University Press, Cambridge, UK.

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

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

Kao, W.-C., Chung, K.-M., Sun, C.-L., and Lin, C.-J. (2004).
Decomposition methods for linear support vector machines.
Neural Computation, 16(8):1689–1704.

Keerthi, S. S., Chapelle, O., and DeCoste, D. (2006).
Building support vector machines with reduced classifier complexity.
Journal of Machine Learning Research, 7:1493–1515.

References III

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

Lanckriet, G., Cristianini, N., Bartlett, P., El Ghaoui, L., and Jordan, M. (2004).
Learning the Kernel Matrix with Semidefinite Programming.
Journal of Machine Learning Research, 5:27–72.

Lee, Y.-J. and Mangasarian, O. L. (2001).
RSVM: Reduced support vector machines.
In *Proceedings of the First SIAM International Conference on Data Mining*.

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

Lin, C.-J. (2002).
A formal analysis of stopping criteria of decomposition methods for support vector machines.
IEEE Transactions on Neural Networks, 13(5):1045–1052.

References IV

Osuna, E., Freund, R., and Girosi, F. (1997).
Training support vector machines: An application to face detection.
In *Proceedings of CVPR'97*, pages 130–136, New York, NY. IEEE.

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

Schölkopf, B. and Smola, A. J. (2002).
Learning with kernels.
MIT Press.

Syed, N. A., Liu, H., and Sung, K. K. (1999).
Incremental learning with support vector machines.
In *Workshop on Support Vector Machines, IJCAI99*.

Vapnik, V. (1998).
Statistical Learning Theory.
Wiley, New York, NY.

References V

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

Yu, H., Yang, J., and Han, J. (2003).
Classifying large data sets using svms with hierarchical clusters.
In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 306–315, New York, NY, USA. ACM Press.