Support Vector Machines and Kernel Methods

Chih-Jen Lin
Department of Computer Science
National Taiwan University

Talk at International Workshop on Recent Trends in Learning, Computation, and Finance, Pohang, Korea, August 30, 2010
Outline

- Basic concepts: SVM and kernels
- Training SVM
- Practical use of SVM
- Multi-class classification
- Research directions: large-scale training
- Research directions: linear SVM
- Research directions: others
- Conclusions
Outline

- Basic concepts: SVM and kernels
- Training SVM
- Practical use of SVM
- Multi-class classification
- Research directions: large-scale training
- Research directions: linear SVM
- Research directions: others
- Conclusions
Support Vector Classification

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

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

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

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

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

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

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

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

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

\[
\begin{align*}
\min_{w,b} & \quad \frac{1}{2} w^T w \\
\text{subject to} & \quad y_i (w^T x_i + b) \geq 1, \\
& \quad i = 1, \ldots, l.
\end{align*}
\]
Data May Not Be Linearly Separable

- An example:

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

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

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

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

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

- **w**: maybe infinite variables
- **The dual problem**: finite number of variables

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

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

- At optimum

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

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

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

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

Then \(\phi(x_i)^T \phi(x_j) = (1 + x_i^T x_j)^2\).
- Kernel: \(K(x, y) = \phi(x)^T \phi(y)\); common kernels:

\[
e^{-\gamma \|x_i - x_j\|^2}, \text{ (Radial Basis Function)}
\]
\[
(x_i^T x_j / a + b)^d \text{ (Polynomial kernel)}
\]
Can be inner product in infinite dimensional space

Assume $x \in R^1$ and $\gamma > 0$.

$$e^{-\gamma \|x_i - x_j\|^2} = e^{-\gamma (x_i - x_j)^2} = e^{-\gamma x_i^2 + 2\gamma x_i x_j - \gamma x_j^2}$$

$$= e^{-\gamma x_i^2 - \gamma x_j^2} \left(1 + \frac{2\gamma x_i x_j}{1!} + \frac{(2\gamma x_i x_j)^2}{2!} + \frac{(2\gamma x_i x_j)^3}{3!} + \cdots\right)$$

$$= e^{-\gamma x_i^2 - \gamma x_j^2} \left(1 \cdot 1 + \sqrt{\frac{2\gamma}{1!}} x_i \cdot \sqrt{\frac{2\gamma}{1!}} x_j + \sqrt{\frac{(2\gamma)^2}{2!}} x_i^2 \cdot \sqrt{\frac{(2\gamma)^2}{2!}} x_j^2 + \cdots\right) = \phi(x_i)^T \phi(x_j),$$

where

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

- So what kind of kernel should I use?
- What kind of functions are valid kernels?
- How to decide kernel parameters?
- Some of these issues will be discussed later
**Decision function**

- **At optimum**

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

- **Decision function**

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

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

Only $\phi(x_i)$ of $\alpha_i > 0$ used $\Rightarrow$ support vectors
We have roughly shown basic ideas of SVM

- A 3-D demonstration
  
  http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/svmtoy3d

More about dual problems

- We omit detailed derivations here
- Quite a few people think that for any optimization problem
  
  \[ \Rightarrow \text{Lagrangian dual exists} \]
- **Wrong!** We usually need
  
  *Convex* programming; *Constraint qualification*
- We have them for SVM
Outline

- Basic concepts: SVM and kernels
- Training SVM
- Practical use of SVM
- Multi-class classification
- Research directions: large-scale training
- Research directions: linear SVM
- Research directions: others
- Conclusions
Training SVM

Large Dense Quadratic Programming

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - \mathbf{e}^T \alpha$$
subject to \(0 \leq \alpha_i \leq C, i = 1, \ldots, l\)
\(\mathbf{y}^T \alpha = 0\)

- \(Q_{ij} \neq 0, Q\) : 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 \(Q\)
- Traditional optimization 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 $B$, $N = \{1, \ldots, l\} \setminus B$ fixed
- Sub-problem at the $k$th iteration:

$$
\begin{align*}
\min_{\alpha_B} & \quad \frac{1}{2} \begin{bmatrix} \alpha_B^T & (\alpha_k^N)^T \end{bmatrix} \begin{bmatrix} Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN} \end{bmatrix} \begin{bmatrix} \alpha_B \\ \alpha_k^N \end{bmatrix} - \\
& \quad \begin{bmatrix} e_B^T & (e_k^N)^T \end{bmatrix} \begin{bmatrix} \alpha_B \\ \alpha_k^N \end{bmatrix} \\
\text{subject to} & \quad 0 \leq \alpha_t \leq C, \ t \in B, \ y_B^T \alpha_B = -y_N^T \alpha_k^N 
\end{align*}
$$
Avoid Memory Problems

- The new objective function

\[
\frac{1}{2} \alpha_B^T Q_{BB} \alpha_B + (-e_B + Q_{BN} \alpha_N^k)^T \alpha_B + \text{constant}
\]

- Only \( B \) columns of \( Q \) needed (\(|B| \geq 2\))

- Calculated when used

Trade time for space
How Decomposition Methods Perform?

- Convergence not very fast
- But, no need to have very accurate $\alpha$
  
  Prediction not affected much
- In some situations, $\#$ support vectors $\ll \#$ training points
  
  Initial $\alpha^1 = 0$, some instances never used
- An example of training 50,000 instances using LIBSVM

```
$svm-train -c 16 -g 4 -m 400 22features
```
Total nSV = 3370
Time 79.524s

- On a Xeon 2.0G machine
- Calculating the whole $Q$ takes more time
- $\#SVs = 3,370 \ll 50,000$
  A good case where some remain at zero all the time
Issues of Decomposition Methods

Techniques for faster decomposition methods

- store recently used kernel elements
- working set size/selection
- theoretical issues: convergence
- and others (details not discussed here)

Major software:

- LIBSVM
  - http://www.csie.ntu.edu.tw/~cjlin/libsvm

- SVM^light
  - http://svmlight.joachims.org
Caching and Shrinking

- Speed up decomposition methods
- Caching (Joachims, 1998)
  Store recently used kernel columns in computer memory
- 100K Cache
  
  $\text{time} ./\text{svm-train} -m 0.01 -g 0.01 \text{ a6a}$
  13.62s
- 40M Cache
  
  $\text{time} ./\text{svm-train} -m 40 -g 0.01 \text{ a6a}$
  11.40s
Shrinking (Joachims, 1998)
Some elements remain bounded until the end
Heuristically resized to a smaller problem
See -h 1 option in LIBSVM software
After certain iterations, most bounded elements identified and not changed (Lin, 2002)
So caching and shrinking are useful
Outline

- Basic concepts: SVM and kernels
- Training SVM
- **Practical use of SVM**
- Multi-class classification
- Research directions: large-scale training
- Research directions: linear SVM
- Research directions: others
- Conclusions
Let’s Try a Practical Example

A problem from astroparticle physics

1 1:2.61e+01 2:5.88e+01 3:-1.89e-01 4:1.25e+02
1 1:5.70e+01 2:2.21e+02 3:8.60e-02 4:1.22e+02
1 1:1.72e+01 2:1.73e+02 3:-1.29e-01 4:1.25e+02
...
0 1:2.39e+01 2:3.89e+01 3:4.70e-01 4:1.25e+02
0 1:2.23e+01 2:2.26e+01 3:2.11e-01 4:1.01e+02
0 1:1.64e+01 2:3.92e+01 3:-9.91e-02 4:3.24e+01

Training and testing sets available: 3,089 and 4,000
Sparse format: zero values not stored
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
  
  $\texttt{./svm-predict train.1 train.1.model o}$
  
  Accuracy = 99.7734% (3082/3089)

- Most kernel elements:
  
  $$K_{ij} = e^{-\|x_i - x_j\|^2/4} \begin{cases} 
  1 & \text{if } i = j, \\
  0 & \text{if } i \neq j.
  \end{cases}$$

- Some features in rather large ranges
Data Scaling

- Without scaling
  Attributes in greater numeric ranges may dominate
- Linearly scale each feature to $[0, 1]$ by:
  $$\frac{\text{feature value} - \min}{\max - \min},$$

There are other ways
- Scaling generally helps, but not always
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 predict

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

Accuracy = 96.15%

Training accuracy now is

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

Accuracy = 96.439%

Default parameter: $C = 1, \gamma = 0.25$
Different Parameters

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

  $./svm-train -c 20 -g 400 \text{train.1.scale}$
  $./svm-predict \text{train.1.scale} \text{train.1.scale.model}$
  Accuracy = 100% (3089/3089)

- 100% training accuracy but

  $./svm-predict \text{test.1.scale} \text{train.1.scale.model}$
  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

- Need to select suitable parameters
- \( C \) 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?
  So performance better?
Available data ⇒ training and validation
Train the training; test the validation
$k$-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
Selecting Kernels

- RBF, polynomial, or others?
- 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 \( K(x, y) = e^{-\gamma\|x-y\|^2} \)
3. Use cross-validation to find the best parameter \( C \) and \( \gamma \)
4. Use the best \( C \) and \( \gamma \) to train the whole training set
5. Test

For beginners only, you can do a lot more
Contour of Parameter Selection

german.numer scale

lg(C)

lg(gamma)
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.
Outline

- Basic concepts: SVM and kernels
- Training SVM
- Practical use of SVM
- Multi-class classification
- Research directions: large-scale training
- Research directions: linear SVM
- Research directions: others
- Conclusions
Multi-class Classification

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

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

Reason: If \( x \in 1st \) class, then we should have

\[ (w^1)^T \phi(x) + b_1 \geq +1 \]

\[ (w^2)^T \phi(x) + b_2 \leq -1 \]

\[ \vdots \]

\[ (w^k)^T \phi(x) + b_k \leq -1 \]
Multi-class Classification (Cont’d)

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

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

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

Select the one with the largest vote

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

- Solving a **single** optimization problem (Weston and Watkins, 1999; Crammer and Singer, 2002; Lee et al., 2004)

- There are many other methods

- A comparison in Hsu and Lin (2002)

- RBF kernel: accuracy similar for different methods

  But 1-against-1 fastest for training
Outline

- Basic concepts: SVM and kernels
- Training SVM
- Practical use of SVM
- Multi-class classification
- Research directions: large-scale training
- Research directions: linear SVM
- Research directions: others
- Conclusions
SVM doesn’t Scale Up

Yes, if using kernels

- Training millions of data is time consuming
- Cases with many support vectors: quadratic time bottleneck on $Q_{SV, sv}$

- For noisy data: # SVs increases linearly in data size (Steinwart, 2003)

Some solutions

- Parallelization
- Approximation
Parallelization

Multi-core/Shared Memory/GPU

- One line change of LIBSVM

<table>
<thead>
<tr>
<th>Multicore</th>
<th>Shared-memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>80</td>
</tr>
<tr>
<td>2</td>
<td>48</td>
</tr>
<tr>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td>8</td>
<td>27</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>57</td>
</tr>
<tr>
<td>4</td>
<td>36</td>
</tr>
<tr>
<td>8</td>
<td>28</td>
</tr>
</tbody>
</table>

50,000 data (kernel evaluations: 80% time)

- GPU (Catanzaro et al., 2008)

Distributed Environments

- Chang et al. (2007); Zanni et al. (2006); Zhu et al. (2009). All use MPI; reasonably good speed-up
Approximately Training SVM

- Can be done in many aspects
- Data level: sub-sampling
- Optimization level:
  - Approximately solve the quadratic program
- Other non-intuitive but effective ways
  - I will show one today
- Many papers have addressed this issue
Approximately Training SVM (Cont’d)

Subsampling

- Simple and often effective

More advanced techniques

- Incremental training: (e.g., Syed et al., 1999))
  Data $\Rightarrow$ 10 parts
  train 1st part $\Rightarrow$ SVs, train SVs + 2nd part, ... 
- Select and train good points: KNN or heuristics
  For example, Bakır et al. (2005)
Approximately Training SVM (Cont’d)

- **Approximate the kernel**; e.g., Fine and Scheinberg (2001); Williams and Seeger (2001)
- **Use part of the kernel**; e.g., Lee and Mangasarian (2001); Keerthi et al. (2006)
- **Early stopping** of optimization algorithms
  Tsang et al. (2005) and others
- And many more
  Some simple but some sophisticated
Sophisticated techniques may not be always useful

Sometimes slower than sub-sampling

covtype: 500k training and 80k testing
rcv1: 550k training and 14k testing

<table>
<thead>
<tr>
<th>covtype</th>
<th>Training size</th>
<th>Accuracy</th>
<th>rcv1</th>
<th>Training size</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50k</td>
<td>92.5%</td>
<td></td>
<td>50k</td>
<td>97.2%</td>
</tr>
<tr>
<td></td>
<td>100k</td>
<td>95.3%</td>
<td></td>
<td>100k</td>
<td>97.4%</td>
</tr>
<tr>
<td></td>
<td>500k</td>
<td>98.2%</td>
<td></td>
<td>550k</td>
<td>97.8%</td>
</tr>
</tbody>
</table>
Sophisticated techniques may not be always useful
Sometimes slower than sub-sampling
covtype: 500k training and 80k testing
rcv1: 550k training and 14k testing

<table>
<thead>
<tr>
<th></th>
<th>covtype</th>
<th>rcv1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training size</td>
<td>Accuracy</td>
<td>Training size</td>
</tr>
<tr>
<td>50k</td>
<td>92.5%</td>
<td>50k</td>
</tr>
<tr>
<td>100k</td>
<td>95.3%</td>
<td>100k</td>
</tr>
<tr>
<td>500k</td>
<td>98.2%</td>
<td>550k</td>
</tr>
</tbody>
</table>
Discussion: Large-scale Training

- We don’t have many large and **well labeled** sets
- Expensive to obtain true labels
- Specific properties of data should be considered
- We will illustrate this point using linear SVM
- The design of software for very large data sets should be application different
Outline

- Basic concepts: SVM and kernels
- Training SVM
- Practical use of SVM
- Multi-class classification
- Research directions: large-scale training
- Research directions: linear SVM
- Research directions: others
- Conclusions
Linear SVM

- Data **not mapped** to another space

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

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

\( \xi_i \geq 0, \quad i = 1, \ldots, l. \)

- In theory, RBF kernel with certain parameters \( \Rightarrow \) as good as linear (Keerthi and Lin, 2003):
  Test accuracy of linear \( \leq \) Test accuracy of RBF

- But can be an approximation to nonlinear
  Recently linear SVM an important research topic
Linear SVM for Large Document Sets

- Bag of words model (TF-IDF or others)
  - A large number of features
- Accuracy similar with/without mapping vectors
- What if training is much faster?
  - A very effective approximation to nonlinear SVM
A Comparison: LIBSVM and LIBLINEAR

- rcv1: # data: > 600k, # features: > 40k
- Using LIBSVM (linear kernel): > 10 hours
- Using LIBLINEAR (same stopping condition)
  Computation: < 5 seconds; I/O: 60 seconds
- Accuracy similar to nonlinear; more than 100x speedup
- Training millions of data in a few seconds
- See some results in Hsieh et al. (2008) by running LIBLINEAR
  http://www.csie.ntu.edu.tw/~cjlin/liblinear
Research directions: linear SVM

Testing Accuracy versus Training Time

- news20
- yahoo-japan
- rcv1
- yahoo-korea
Why Training Linear SVM Is Faster?

- In optimization, each iteration often needs
  \[ \nabla_i f(\alpha) = (Q\alpha)_i - 1 \]

- Nonlinear SVM
  \[ \nabla_i f(\alpha) = \sum_{j=1}^{l} y_i y_j K(x_i, x_j) \alpha_j - 1 \]
  cost: \(O(nl)\); \(n\): # features, \(l\): # data

- Linear: use
  \[ \mathbf{w} \equiv \sum_{j=1}^{l} y_j \alpha_j \mathbf{x}_j \] and \( \nabla_i f(\alpha) = y_i \mathbf{w}^T \mathbf{x}_i - 1 \)

- Only \(O(n)\) cost if \(\mathbf{w}\) is maintained
Extension: Training Explicit Form of Nonlinear Mappings

Linear-SVM method to train $\phi(x_1), \ldots, \phi(x_l)$

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

Low-degree Polynomial Mappings

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

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

- When degree is small, train the explicit form of $\phi(x)$
## Testing Accuracy and Training Time

<table>
<thead>
<tr>
<th>Data set</th>
<th>Degree-2 Polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training time (s)</td>
</tr>
<tr>
<td></td>
<td>LIBLINEAR</td>
</tr>
<tr>
<td>a9a</td>
<td>1.6</td>
</tr>
<tr>
<td>real-sim</td>
<td>59.8</td>
</tr>
<tr>
<td>ijcnn1</td>
<td>10.7</td>
</tr>
<tr>
<td>MNIST38</td>
<td>8.6</td>
</tr>
<tr>
<td>covtype</td>
<td>5,211.9</td>
</tr>
<tr>
<td>webspam</td>
<td>3,228.1</td>
</tr>
</tbody>
</table>

Training $\phi(x_i)$ by linear: faster than kernel, but sometimes competitive accuracy.
Discussion: Directly Train $\phi(x_i)$, $\forall i$

- See details in our work (Chang et al., 2010)
- Useful for certain applications
Linear Classification: Data Larger than Memory

- Existing methods cannot easily handle this situation
- See our recent KDD work (Yu et al., 2010)
  KDD 2010 best paper award
- Training several million data (or more) on your laptop
Linear Classification: Online Learning

For extremely large data, cannot keep all data

- After using new data to update the model; may not need them any more

**Online** learning instead of **offline** learning

- Training often by stochastic gradient descent methods
  - They use only a subset of data at each step
- Now an important research topic (e.g., Shalev-Shwartz et al., 2007; Langford et al., 2009; Bordes et al., 2009)
Linear Classification: L1 Regularization

- 1-norm versus 2-norm

\[ \| \mathbf{w} \|_1 = |w_1| + \cdots + |w_n|, \quad \| \mathbf{w} \|_2^2 = w_1^2 + \cdots + w_n^2 \]

- 2-norm: all \( w_i \) are non-zeros; 1-norm: some \( w_i \) may be zeros; useful for feature selection
- Recently a hot topic; see our survey (Yuan et al., 2010)
Outline

- Basic concepts: SVM and kernels
- Training SVM
- Practical use of SVM
- Multi-class classification
- Research directions: large-scale training
- Research directions: linear SVM
- Research directions: others
- Conclusions
Multiple Kernel Learning (MKL)

- How about using

\[ t_1 K_1 + t_2 K_2 + \cdots + t_r K_r, \text{ where } t_1 + \cdots + t_r = 1 \]

as the kernel

- Related to parameter/kernel selection
  - If \( K_1 \) better \( \Rightarrow t_1 \) close to 1, others close to 0

- Earlier development (Lanckriet et al., 2004): high computational cost

- Many subsequent works (e.g., Rakotomamonjy et al., 2008).

- Still ongoing; so far MKL has not been a practical tool yet
Ranking

• Labels become ranking information
e.g., $x_1$ ranks higher than $x_2$

• RankSVM (Joachims, 2002): add constraint

$$w^T x_i \geq w^T x_j + \xi_{ij} \text{ if } x_i \text{ ranks better than } x_j$$

• Many subsequent works

• However, whether SVM is the most suitable method for ranking is an issue
Other Directions

- Semi-supervised learning
  Use information from unlabeled data
- Active learning
  Needs cost to obtain labels of data
- Cost sensitive learning
  For unbalanced data
- Structured Learning
  Data instance not an Euclidean vector
  Maybe a parse tree of a sentence
- Feature selection
Outline

- Basic concepts: SVM and kernels
- Training SVM
- Practical use of SVM
- Multi-class classification
- Research directions: large-scale training
- Research directions: linear SVM
- Research directions: others
- Conclusions
Discussion and Conclusions

- SVM and kernel methods are rather mature areas
- But still quite a few interesting research issues
- Many are extensions of standard classification (e.g., semi-supervised learning)
- It is possible to identify more extensions through real applications
References I


