Support Vector Machines and Kernel Methods

Chih-Jen Lin
Department of Computer Science
National Taiwan University

Tutorial at ACML, November 8, 2010

Outline

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

Data Classification

- Given training data in different classes (labels known)

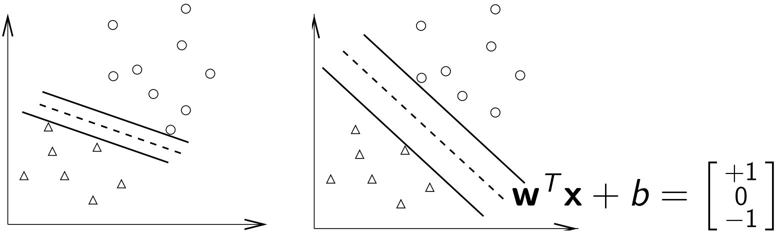
  Predict test data (labels unknown)
- Training and testing

Support Vector Classification

- Training vectors : $\mathbf{x}_i, i = 1, \ldots, l$
- Feature vectors. For example,
  A patient = [height, weight, . . .]^T
- 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: $\mathbf{w}^T\mathbf{x} + b = 0$

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

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

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

Maximal Margin

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

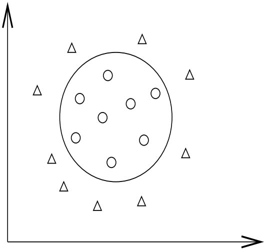
$$2/\|\mathbf{w}\| = 2/\sqrt{\mathbf{w}^T\mathbf{w}}$$

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

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

Data May Not Be Linearly Separable

- An example:



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

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

- Standard SVM (Boser et al., 1992; 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: $\mathbf{x} \in R^3, \phi(\mathbf{x}) \in R^{10}$

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

Finding the Decision Function

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

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

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

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

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

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

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

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

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

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

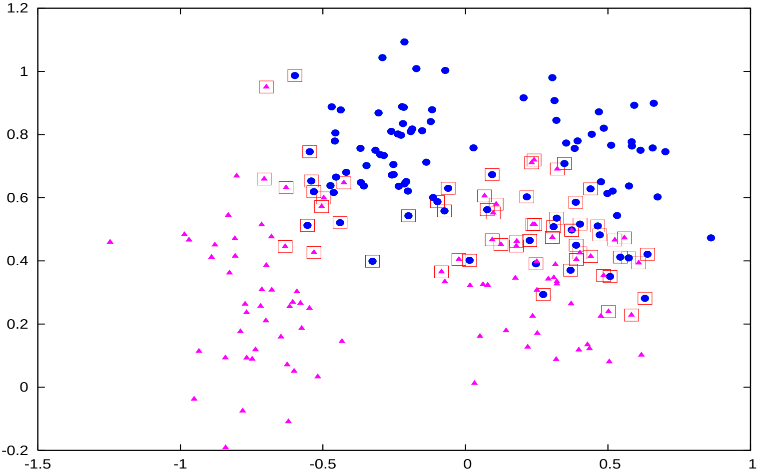
- Decision function

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

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

Support Vectors: More Important Data

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



Outline

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

Large Dense Quadratic Programming

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

- $Q_{ij} \neq 0$, 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, $N = \{1, \ldots, l\} \backslash B$ fixed
- Sub-problem at the kth iteration:

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

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

Avoid Memory Problems

- The new objective function

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

- 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 α

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

  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`

Outline

- Basic concepts: SVM and kernels
- Training SVM
- Practical use of SVM
- 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
1 1:2.17e+01 2:1.24e+02 3:1.53e-01  4:1.52e+02
1 1:9.13e+01 2:2.93e+02 3:1.42e-01  4:1.60e+02
1 1:5.53e+01 2:1.79e+02 3:1.65e-01  4:1.11e+02
1 1:2.95e+01 2:1.91e+02 3:9.90e-02  4:1.03e+02
```

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

Sparse format: zero values not stored

Poor Results from Direct Training/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
- Linearly scale the first to $[0, 1]$ by:

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

  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 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

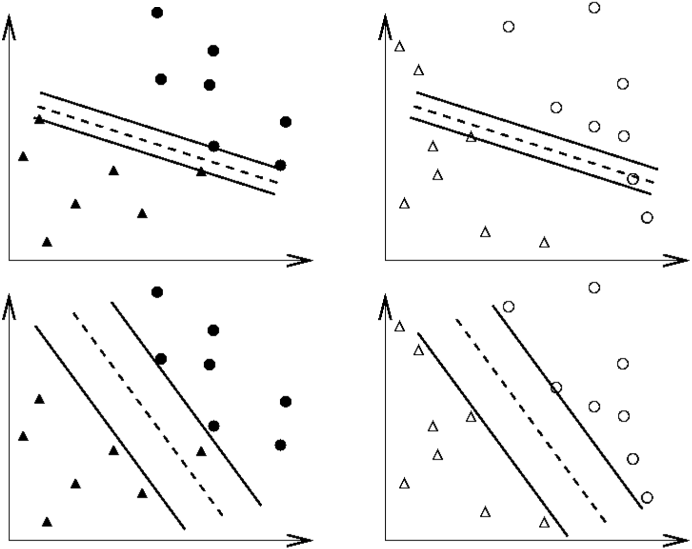
- In theory

  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

- 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?

Performance Evaluation

- Available data $\Rightarrow$ 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

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

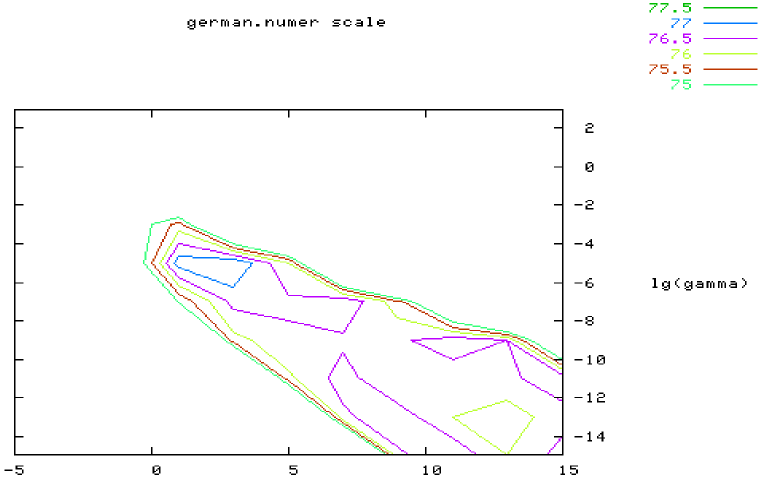
  More parameters than RBF

A Simple Procedure

1. Conduct simple scaling on the data
2. Consider RBF kernel $K(\mathbf{x}, \mathbf{y}) = e^{-\gamma \|\mathbf{x}-\mathbf{y}\|^2}$
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

Outline

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

$$Q_{\text{SV, SV}}$$

- 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

Multicore		Shared-memory	
1	80	1	100
2	48	2	57
4	32	4	36
8	27	8	28

  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).

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

Approximately Training SVM (Cont'd)

- Sophisticated techniques may not be always useful
- Sometimes slower than sub-sampling
- covtype: 500k training and 80k testing
  rcv1: 550k training and 14k testing

covtype		rcv1	
Training size	Accuracy	Training size	Accuracy
50k	92.5%	50k	97.2%
100k	95.3%	100k	97.4%
500k	98.2%	550k	97.8%

Approximately Training SVM (Cont'd)

- Sophisticated techniques may not be always useful
- Sometimes slower than sub-sampling
- covtype: 500k training and 80k testing
  rcv1: 550k training and 14k testing

covtype		rcv1	
Training size	Accuracy	Training size	Accuracy
50k	92.5%	50k	97.2%
100k	95.3%	100k	97.4%
500k	98.2%	550k	97.8%

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
- Research directions: large-scale training
- Research directions: linear SVM
- Research directions: others
- Conclusions

Linear SVM

- Data not mapped to another space

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

- In theory, RBF kernel with certain parameters ⇒ as good as linear (Keerthi and Lin, 2003):

  Test accuracy of linear ≤ Test accuracy of RBF
- But can be an approximation to nonlinear

  Recently linear SVM an important research topic

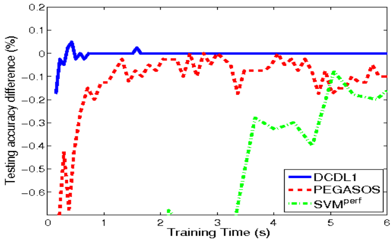
Linear SVM for Large Document Sets

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

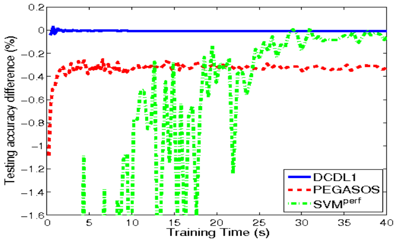
A Comparison: LIBSVM and LIBLINEAR

- 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`

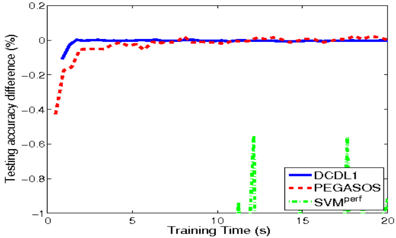
Testing Accuracy versus Training Time



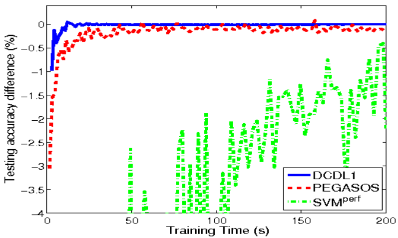
news20



yahoo-japan



rcv1



yahoo-korea

Why Training Linear SVM Is Faster?

- In optimization, each iteration often needs

$$\nabla_i f(\boldsymbol{\alpha}) = (Q\boldsymbol{\alpha})_i - 1$$

- Nonlinear SVM

$$\nabla_i f(\boldsymbol{\alpha}) = \sum_{j=1}^{l} y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)\alpha_j - 1$$

  cost: $O(nl)$; n: # features, l: # data
- Linear: use

$$\mathbf{w} \equiv \sum_{j=1}^{l} y_j \alpha_j \mathbf{x}_j \text{ and } \nabla_i f(\boldsymbol{\alpha}) = y_i \mathbf{w}^T \mathbf{x}_i - 1$$

- Only $O(n)$ cost if $\mathbf{w}$ is maintained

Extension: Training Explicit Form of Nonlinear Mappings

Linear-SVM method to train $\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_l)$

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

Low-degree Polynomial Mappings

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

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

- When degree is small, train the explicit form of $\phi(\mathbf{x})$

Testing Accuracy and Training Time

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

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

Discussion: Directly Train $\phi(\mathbf{x}_i), \forall i$

- See details in our work (Chang et al., 2010)
- A related development: Sonnenburg and Franc (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

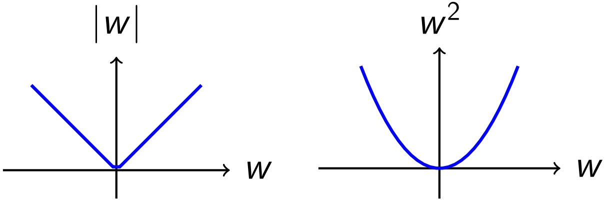
- Training often by stochastic gradient descent methods

  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

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., $\mathbf{x}_1$ ranks higher than $\mathbf{x}_2$
- RankSVM (Joachims, 2002): add constraint

$$\mathbf{w}^T\mathbf{x}_i \geq \mathbf{w}^T\mathbf{x}_j + \xi_{ij} \text{ if } \mathbf{x}_i \text{ ranks better than } \mathbf{x}_j$$

- 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
- 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

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

A. Bordes, L. Bottou, and P. Gallinari. SGD-QN: Careful quasi-Newton stochastic gradient descent. *Journal of Machine Learning Research*, 10:1737–1754, 2009.

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

B. Catanzaro, N. Sundaram, and K. Keutzer. Fast support vector machine training and classification on graphics processors. In *Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML)*, 2008.

E. Chang, K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, and H. Cui. Parallelizing support vector machines on distributed computers. In *NIPS 21*, 2007.

Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, and C.-J. Lin. Training and testing low-degree polynomial data mappings via linear SVM. *Journal of Machine Learning Research*, 11:1471–1490, 2010. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/lowpoly_journal.pdf`.

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

References II

S. Fine and K. Scheinberg. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.

C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML)*, 2008. URL http://www.csie.ntu.edu.tw/~cjlin/papers/cddual.pdf.

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

T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

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

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

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

References III

J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10:771–801, 2009.

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

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

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

A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: primal estimated sub-gradient solver for SVM. In *Proceedings of the Twenty Fourth International Conference on Machine Learning (ICML)*, 2007.

S. Sonnenburg and V. Franc. COFFIN : A computational framework for linear SVMs. In *Proceedings of the Twenty Seventh International Conference on Machine Learning (ICML)*, 2010.

I. Steinwart. Sparseness of support vector machines. *Journal of Machine Learning Research*, 4: 1071–1105, 2003.

References IV

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

I. Tsang, J. Kwok, and P. Cheung. Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In T. Leen, T. Dietterich, and V. Tresp, editors, *Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

H.-F. Yu, C.-J. Hsieh, K.-W. Chang, and C.-J. Lin. Large linear classification when data cannot fit in memory. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/kdd_disk_decomposition.pdf`.

G.-X. Yuan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin. A comparison of optimization methods and software for large-scale l1-regularized linear classification. 2010. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/l1.pdf`. To appear in Machine Learning Research.

L. Zanni, T. Serafini, and G. Zanghirati. Parallel software for training large scale support vector machines on multiprocessor systems. *Journal of Machine Learning Research*, 7: 1467–1492, 2006.

Z. A. Zhu, W. Chen, G. Wang, C. Zhu, and Z. Chen. P-packSVM: Parallel primal gradient descent kernel SVM. In *Proceedings of the 2009 edition of the IEEE International Conference on Data Mining*, 2009.