Training Large-scale Linear Classifiers

Chih-Jen Lin
Department of Computer Science
National Taiwan University
http://www.csie.ntu.edu.tw/~cjlin

Talk at Hong Kong University of Science and Technology

February 5, 2009

Outline

- Linear versus Nonlinear Classification
- Review of SVM Training
- Large-scale Linear SVM
- Discussion and Conclusions

Outline

- Linear versus Nonlinear Classification
- Review of SVM Training
- Large-scale Linear SVM
- Discussion and Conclusions

Kernel Methods and SVM

- Kernel methods became very popular in the past decade

  In particular, support vector machines (SVM)
- But slow in training large data due to nonlinear mapping (enlarge the # features)
- Example: $\mathbf{x} = [x_1, x_2, x_3]^T \in R^3$

$$\begin{aligned}\phi(\mathbf{x}) &= [1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, x_1^2, \\ &\quad x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3]^T \in R^{10}\end{aligned}$$

- If data are very large $\Rightarrow$ often need approximation

  e.g., sub-sampling and many other ways

Linear Classification

- Certain problems: # features large
- Often similar accuracy with/without nonlinear mappings
- Linear classification: no mapping

  Stay in the original input space
- We can efficiently train very large data
- Document classification is of this type

  Very important for Internet companies

An Example

- rcv1: # data: > 600k, # features: > 40k
- Using LIBSVM (linear kernel)

  > 10 hours
- Using LIBLINEAR

  Computation: < 5 seconds; I/O: 60 seconds
- Same stopping condition in solving SVM optimization problems
- Will show how this is achieved and discuss if there are any concerns

Outline

- Linear versus Nonlinear Classification
- Review of SVM Training
- Large-scale Linear SVM
- Discussion and Conclusions

Support Vector Classification

- Training data $(\mathbf{x}_i, y_i), i = 1, \ldots, l, \mathbf{x}_i \in R^n, y_i = \pm 1$

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

- C: regularization parameter
- High dimensional (maybe infinite) feature space

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

- We omit the bias term b
- $\mathbf{w}$: may have infinite variables

Support Vector Classification (Cont'd)

- The dual problem (finite # variables)

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

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

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

- Kernel: $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

Large Dense Quadratic Programming

- $Q_{ij} \neq 0$, Q : an l by l fully dense matrix

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

- 50,000 training points: 50,000 variables:
  $(50,000^2 \times 8/2)$ bytes $=$ 10GB RAM to store Q
- Traditional methods:
  Newton, Quasi Newton cannot be directly applied
- Now most use decomposition methods
  [Osuna et al., 1997, Joachims, 1998, Platt, 1998]

Decomposition Methods

- We consider a one-variable version
  Similar to coordinate descent methods
- Select the ith component for update:

$$\begin{aligned} \min_{d} \quad & \frac{1}{2}(\boldsymbol{\alpha} + d\mathbf{e}_i)^T Q(\boldsymbol{\alpha} + d\mathbf{e}_i) - \mathbf{e}^T(\boldsymbol{\alpha} + d\mathbf{e}_i) \\ \text{subject to} \quad & 0 \le \alpha_i + d \le C \end{aligned}$$

  where

$$\mathbf{e}_i \equiv [\underbrace{0 \ldots 0}_{i-1}\ 1\ 0 \ldots 0]^T$$

- $\boldsymbol{\alpha}$: current solution; the ith component is changed

Avoid Memory Problems

- The new objective function

$$\frac{1}{2} Q_{ii} d^2 + (Q\boldsymbol{\alpha} - \mathbf{e})_i d + \text{ constant}$$

- To get $(Q\boldsymbol{\alpha} - \mathbf{e})_i$, only Q's ith row is needed

$$(Q\boldsymbol{\alpha} - \mathbf{e})_i = \sum_{j=1}^{l} Q_{ij}\alpha_j - 1$$

- Calculated when needed. Trade time for space
- Used by popular software (e.g., SVM^{light}, LIBSVM)
  They update 10 and 2 variables at a time

Decomposition Methods: Algorithm

- Optimal d:

$$-\frac{(Q\boldsymbol{\alpha}-\mathbf{e})_i}{Q_{ii}} = -\frac{\sum_{j=1}^{l} Q_{ij}\alpha_j - 1}{Q_{ii}}$$

- Consider lower/upper bounds: $[0, C]$
- Algorithm:

  While $\boldsymbol{\alpha}$ is not optimal

  1. Select the ith element for update
  2. $\alpha_i \leftarrow \min\left(\max\left(\alpha_i - \frac{\sum_{j=1}^{l} Q_{ij}\alpha_j - 1}{Q_{ii}}, 0\right), C\right)$

Select an Element for Update

Many ways

- Sequential (easiest)
- Permuting $1, \ldots, l$ every l steps
- Random
- Existing software check gradient information

$$\nabla_1 f(\boldsymbol{\alpha}), \ldots, \nabla_l f(\boldsymbol{\alpha})$$

But is $\nabla f(\boldsymbol{\alpha})$ available?

Select an Element for Update (Cont'd)

- We can easily maintain gradient

$$\nabla f(\boldsymbol{\alpha}) = Q\boldsymbol{\alpha} - \mathbf{e}$$
$$\nabla_s f(\boldsymbol{\alpha}) = (Q\boldsymbol{\alpha})_s - 1 = \sum_{j=1}^{l} Q_{sj}\alpha_j - 1$$

- Initial $\boldsymbol{\alpha} = \mathbf{0}$

$$\nabla f(\mathbf{0}) = -\mathbf{e}$$

- α_i updated to $\bar{\alpha}_i$

$$\nabla_s f(\boldsymbol{\alpha}) \leftarrow \nabla_s f(\boldsymbol{\alpha}) + Q_{si}(\bar{\alpha}_i - \alpha_i), \quad \forall s$$

- $O(l)$ if Q_{si} $\forall s$ (ith column) are available

Select an Element for Update (Cont'd)

- No matter maintaining $\nabla f(\boldsymbol{\alpha})$ or not

  Q's ith row (column) always needed

$$\bar{\alpha}_i \leftarrow \min\left(\max\left(\alpha_i - \frac{\sum_{j=1}^{l} Q_{ij}\alpha_j - 1}{Q_{ii}}, 0\right), C\right)$$

  Q is symmetric

- Using $\nabla f(\boldsymbol{\alpha})$ to select i: faster convergence

  i.e., fewer iterations

Decomposition Methods: Using Gradient

The new procedure

- $\boldsymbol{\alpha} = \mathbf{0}, \nabla f(\boldsymbol{\alpha}) = -\mathbf{e}$
- While $\boldsymbol{\alpha}$ is not optimal
    1. Select the ith element using $\nabla f(\boldsymbol{\alpha})$
    2. $\bar{\alpha}_i \leftarrow \min\left(\max\left(\alpha_i - \frac{\sum_{j=1}^{l} Q_{ij}\alpha_j - 1}{Q_{ii}}, 0\right), C\right)$
    3. $\nabla_s f(\boldsymbol{\alpha}) \leftarrow \nabla_s f(\boldsymbol{\alpha}) + Q_{si}(\bar{\alpha}_i - \alpha_i), \forall s$

Cost per iteration

- $O(ln)$, l: # instances, n: # features
- Assume each $Q_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$ takes $O(n)$

Outline

- Linear versus Nonlinear Classification
- Review of SVM Training
- Large-scale Linear SVM
- Discussion and Conclusions

Linear SVM for Large Document Sets

Document classification

- Bag of words model (TF-IDF or others)
  A large # of features
- Testing accuracy: linear/nonlinear SVMs similar
  nonlinear SVM: we mean SVM via kernels

Recently an active research topic

- SVM^{perf} [Joachims, 2006]
- Pegasos [Shalev-Shwartz et al., 2007]
- LIBLINEAR [Lin et al., 2007, Hsieh et al., 2008]
- and others

Linear SVM

- Primal without the bias term b

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

- Dual

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad f(\boldsymbol{\alpha}) = \; & \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \mathbf{e}^T\boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \le \alpha_i \le C, \forall i \end{aligned}$$

- $Q_{ij} = y_i y_j \mathbf{x}_i^T \mathbf{x}_j$

Revisit Decomposition Methods

- While $\boldsymbol{\alpha}$ is not optimal
  1. Select the ith element for update
  2. $\alpha_i \leftarrow \min\left(\max\left(\alpha_i - \frac{\sum_{j=1}^{l} Q_{ij}\alpha_j - 1}{Q_{ii}}, 0\right), C\right)$
- $O(ln)$ per iteration; n: # features, l: # data
- For linear SVM, define

$$\mathbf{w} \equiv \sum_{j=1}^{l} y_j \alpha_j \mathbf{x}_j \in R^n$$

- $O(n)$ per iteration

$$\sum_{j=1}^{l} Q_{ij}\alpha_j - 1 = \sum_{j=1}^{l} y_i y_j \mathbf{x}_i^T \mathbf{x}_j \alpha_j - 1 = y_i \mathbf{w}^T \mathbf{x}_i - 1$$

- All we need is to maintain $\mathbf{w}$. If

$$\bar{\alpha}_i \leftarrow \alpha_i$$

then $O(n)$ for

$$\mathbf{w} \leftarrow \mathbf{w} + (\bar{\alpha}_i - \alpha_i) y_i \mathbf{x}_i$$

- Initial $\mathbf{w}$

$$\boldsymbol{\alpha} = \mathbf{0} \quad \Rightarrow \quad \mathbf{w} = \mathbf{0}$$

- Give up maintaining $\nabla f(\boldsymbol{\alpha})$
- Select i for update

  Sequential, random, or

  Permuting $1, \ldots, l$ every l steps

Algorithms for Linear and Nonlinear SVM

Linear:

- While $\boldsymbol{\alpha}$ is not optimal
  1. Select the ith element for update
  2. $\bar{\alpha}_i \leftarrow \min\left(\max\left(\alpha_i - \frac{y_i \mathbf{w}^T \mathbf{x}_i - 1}{Q_{ii}}, 0\right), C\right)$
  3. $\mathbf{w} \leftarrow \mathbf{w} + (\bar{\alpha}_i - \alpha_i) y_i \mathbf{x}_i$

Nonlinear:

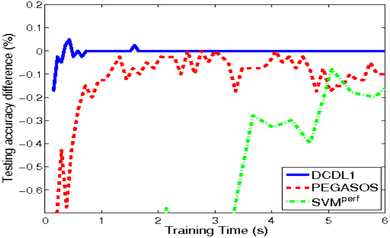
- While $\boldsymbol{\alpha}$ is not optimal
  1. Select the ith element using $\nabla f(\boldsymbol{\alpha})$
  2. $\bar{\alpha}_i \leftarrow \min\left(\max\left(\alpha_i - \frac{\sum_{j=1}^{l} Q_{ij}\alpha_j - 1}{Q_{ii}}, 0\right), C\right)$
  3. $\nabla_s f(\boldsymbol{\alpha}) \leftarrow \nabla_s f(\boldsymbol{\alpha}) + Q_{si}(\bar{\alpha}_i - \alpha_i), \forall s$

Analysis

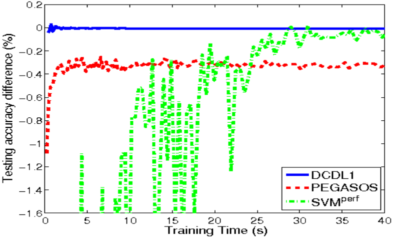
- Decomposition method for nonlinear (also linear): $O(ln)$ per iteration (used in LIBSVM)
- New way for linear: $O(n)$ per iteration (used in LIBLINEAR)
- Faster if # iterations not l times more
- Experiments

Problem	l: # data	n: # features
news20	19,996	1,355,191
yahoo-japan	176,203	832,026
rcv1	677,399	47,236
yahoo-korea	460,554	3,052,939

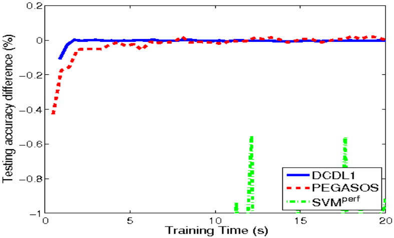
Testing Accuracy versus Training Time



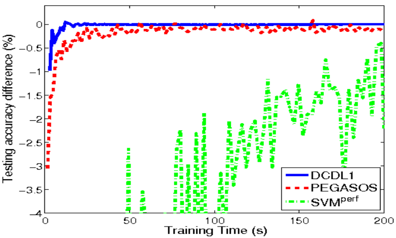
news20



yahoo-japan



rcv1



yahoo-korea

Outline

- Linear versus Nonlinear Classification
- Review of SVM Training
- Large-scale Linear SVM
- Discussion and Conclusions

Limitation

- A few seconds for million data; Too good to be true?
- Less effective if C is large (or data not scaled)

  Same problem occurs for training nonlinear SVMs
- But no need to use large C

  Same model after $C \geq \bar{C}$ [Keerthi and Lin, 2003]

  $\bar{C}$ is small for document data (if scaled)

Limitation (Cont'd)

- Less effective if # features small

  Should solve primal: # variables = # features

  Why not using kernels with nonlinear mappings?

Comparing Different Training Methods

- $O(ln)$ versus $O(n)$ per iteration
- Generally, the new method for linear is much faster
  Especially for document data
- But can always find weird cases where LIBSVM faster than LIBLINEAR
- Apply the right approach to the right problem is essential
- One must be careful on comparing training algorithms

Software Issue

- Large data $\Rightarrow$ may need different training strategies for different problems
- But we pay the price of complicating software packages
- The success of LIBSVM and SVM^{light}
  Simple and general
- They cover both linear/nonlinear
- General versus special: always an issue

Other Methods for Linear SVM

- $\mathbf{w}$ is the key to reduce $O(ln)$ to $O(n)$ per iteration

$$\mathbf{w} = \sum_{j=1}^{l} y_j \alpha_j \mathbf{x}_j \in R^n$$

- Many optimization methods can be used
- We can now solve primal: **$\mathbf{w}$ not infinite any more**

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

- We used decomposition method as an example as it works for both linear and nonlinear
  Easily see the striking difference with/without $\mathbf{w}$

Other Linear Classifiers

- Logistic regression, maximum entropy, conditional random fields (CRF)

  All linear classifiers
- In the past, SVM training is considered very different from them
- For the linear case, things are very related
- Many interesting findings; but no time to show details

What if Data Are Even Larger?

- We see I/O costs more than computing
- Large-scale document classification on a single computer essentially a solved problem
- Challenges:

  What if data larger than computer RAM?

  What if data distributedly stored?
- Document classification in a data center environment is an interesting research direction

Conclusions

- For certain problems, linear classifiers as accurate as nonlinear, and more efficient for training/testing
- However, we are **not** claiming you shouldn't use kernels any more
- For large data, right approaches are essential

  Machine learning researchers should clearly tell people when to use which methods
- You are welcome to try our software

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

  http://www.csie.ntu.edu.tw/~cjlin/liblinear