Optimization and Machine Learning

Chih-Jen Lin
Department of Computer Science
National Taiwan University

Talk at 25th Simon Stevin Lecture, K. U. Leuven Optimization in Engineering Center, January 17, 2013

Outline

1. Introduction
2. Optimization methods for kernel support vector machines
3. Optimization methods for linear support vector machines
4. Discussion and conclusions

Outline

1 Introduction

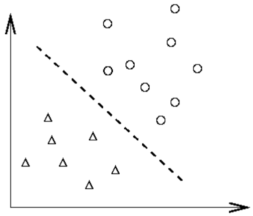
2 Optimization methods for kernel support vector machines

3 Optimization methods for linear support vector machines

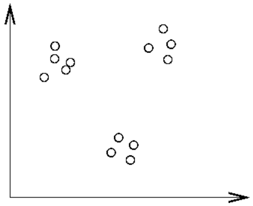
4 Discussion and conclusions

What is Machine Learning

- Extract knowledge from data
- Representative tasks: classification, clustering, and others



Classification



Clustering

- An old area, but many new and interesting applications/extensions: ranking, etc.

Data Classification

- Given training data in different classes (labels known)

  Predict test data (labels unknown)
- Classic example
  1. Find a patient's blood pressure, weight, etc.
  2. After several years, know if he/she recovers
  3. Build a machine learning model
  4. New patient: find blood pressure, weight, etc
  5. Prediction
- Two main stages: training and testing

Data Classification (Cont'd)

- Representative methods

  Nearest neighbor, naive Bayes

  Decision tree, random forest

  Neural networks, support vector machines

Why Is Optimization Used?

- Usually the goal of classification is to

  minimize the test error

- Therefore, many classification methods solve optimization problems

Optimization and Machine Learning

- Standard optimization packages may be directly applied to machine learning applications
- However, efficiency and scalability are issues
- Very often machine learning knowledge must be considered in designing suitable optimization methods
- We will discuss some examples in this talk

Outline

1 Introduction

2 Optimization methods for kernel support vector machines

3 Optimization methods for linear support vector machines

4 Discussion and conclusions

Kernel Methods

- Kernel methods are a class of classification techniques where major operations are conducted by kernel evaluations
- A representative example is support vector machine

Support Vector Classification

- Training data $(\mathbf{x}_i, y_i), i = 1, \ldots, l, \mathbf{x}_i \in R^n, y_i = \pm 1$
- Maximizing the margin (Boser et al., 1992; Cortes and Vapnik, 1995)

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

- High dimensional (maybe infinite) feature space

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

- $\mathbf{w}$: maybe infinite variables

Support Vector Classification (Cont'd)

- The dual problem (finite # variables)

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

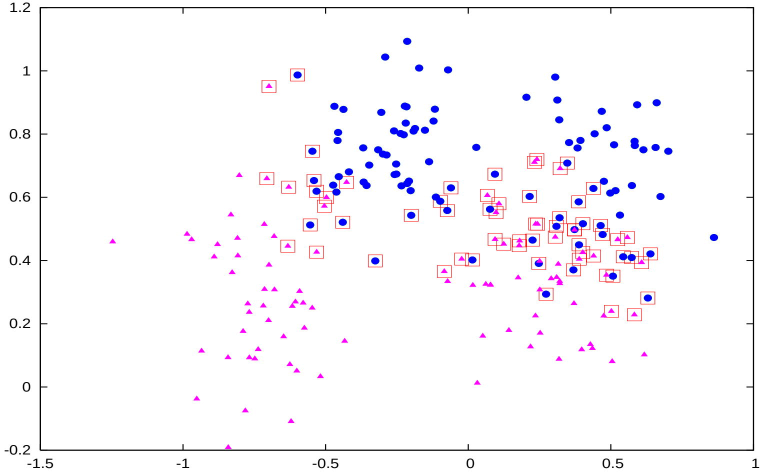
  where $Q_{ij} = y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ and $\mathbf{e} = [1, \ldots, 1]^T$
- At optimum

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

- Kernel: $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$; closed form
  Example: Gaussian (RBF) kernel: $e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}$

Support Vector Classification (Cont'd)

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



Large Dense Quadratic Programming

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

- $Q_{ij} \neq 0$, 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

Large Dense Quadratic Programming (Cont'd)

- For quadratic programming problems, traditionally we would use

  Newton or quasi Newton
- However, they cannot be directly applied here because Q cannot even be stored
- Currently, decomposition methods (a type of coordinate descent methods) are what used in practice

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:

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

Avoid Memory Problems

- The new objective function

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

- Only B columns of Q are needed
- $|B| \geq 2$ due to the equality constraint and in general $|B| \leq 10$ is used
- Calculated when used: trade time for space
- But is such an approach practical?

How Decomposition Methods Perform?

- Convergence not very fast. This is known because of using only first-order information
- But, no need to have very accurate $\boldsymbol{\alpha}$

  $$\text{decision function:} \quad \sum_{i=1}^{l} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b$$

  Prediction may still be correct with a rough $\boldsymbol{\alpha}$
- Further, in some situations,

  $$\# \text{ support vectors} \ll \# \text{ training points}$$

  Initial $\boldsymbol{\alpha}^1 = 0$, some instances never used

How Decomposition Methods Perform? (Cont'd)

- An example of training 50,000 instances using the software LIBSVM

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

- This was done on a typical desktop
- 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

How Decomposition Methods Perform? (Cont'd)

- Because many $\alpha_i = 0$ in the end, we can develop a shrinking techniques

  Variables are removed during the optimization procedure. Smaller problems are solved

Machine Learning Properties are Useful in Designing Optimization Algorithms

We have seen that special properties of SVM did contribute to the viability of decomposition method

- For machine learning applications, no need to accurately solve the optimization problem
- Because some optimal $\alpha_i = 0$, decomposition methods may not need to update all the variables
- Also, we can use shrinking techniques to reduce the problem size during decomposition methods

Differences between Optimization and Machine Learning

- The two topics may have different focuses. We give the following example
- The decomposition method we just discussed converges more slowly when C is large
- Using $C = 1$ on a date set

  # iterations: 508
- Using $C = 5,000$

  # iterations: 35,241

- Optimization researchers may rush to solve difficult cases of large C

  That's what I did before
- It turns out that large C less used than small C
- Recall that SVM solves

$$\frac{1}{2}\mathbf{w}^T\mathbf{w} + C(\text{sum of training losses})$$

- A large C means to overfit training data
- This does not give good testing accuracy

Outline

1 Introduction

2 Optimization methods for kernel support vector machines

3 Optimization methods for linear support vector machines

4 Discussion and conclusions

Linear and Kernel Classification

- We have

  Kernel $\Rightarrow$ map data to a higher space
  Linear $\Rightarrow$ use the original data

- Intuitively, kernel should give superior accuracy than linear
- There are even some theoretical results
- We optimization people may think there is no need to specially consider linear SVM
- However, this is wrong if we consider their practical use

Linear and Kernel Classification (Cont'd)

Methods such as SVM and logistic regression can used in two ways

- Kernel methods: data mapped to a higher dimensional space

  $$\mathbf{x} \Rightarrow \phi(\mathbf{x})$$

  $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ easily calculated; little control on $\phi(\cdot)$
- Linear classification + feature engineering:

  We have $\mathbf{x}$ without mapping. Alternatively, we can say that $\phi(\mathbf{x})$ is our $\mathbf{x}$; full control on $\mathbf{x}$ or $\phi(\mathbf{x})$

Linear and Kernel Classification (Cont'd)

- For some problems, accuracy by linear is as good as nonlinear

  But training and testing are much faster
- This particularly happens for document classification

  Number of features (bag-of-words model) very large

  Data very sparse (i.e., few non-zeros)
- Recently linear classification is a popular research topic.

Comparison Between Linear and Kernel (Training Time & Testing Accuracy)

			Linear		RBF Kernel	
Data set	#data	#features	Time	Accuracy	Time	Accuracy
MNIST38	11,982	752	0.1	96.82	38.1	99.70
ijcnn1	49,990	22	1.6	91.81	26.8	98.69
covtype	464,810	54	1.4	76.37	46,695.8	96.11
news20	15,997	1,355,191	1.1	96.95	383.2	96.90
real-sim	57,848	20,958	0.3	97.44	938.3	97.82
yahoo-japan	140,963	832,026	3.1	92.63	20,955.2	93.31
webspam	280,000	254	25.7	93.35	15,681.8	99.26

Therefore, there is a need to develop optimization methods for large linear classification

Comparison Between Linear and Kernel (Training Time & Testing Accuracy)

			Linear		RBF Kernel	
Data set	#data	#features	Time	Accuracy	Time	Accuracy
MNIST38	11,982	752	0.1	96.82	38.1	99.70
ijcnn1	49,990	22	1.6	91.81	26.8	98.69
covtype	464,810	54	1.4	76.37	46,695.8	96.11
news20	15,997	1,355,191	1.1	96.95	383.2	96.90
real-sim	57,848	20,958	0.3	97.44	938.3	97.82
yahoo-japan	140,963	832,026	3.1	92.63	20,955.2	93.31
webspam	280,000	254	25.7	93.35	15,681.8	99.26

Therefore, there is a need to develop optimization methods for large linear classification

Comparison Between Linear and Kernel (Training Time & Testing Accuracy)

			Linear		RBF Kernel	
Data set	#data	#features	Time	Accuracy	Time	Accuracy
MNIST38	11,982	752	0.1	96.82	38.1	99.70
ijcnn1	49,990	22	1.6	91.81	26.8	98.69
covtype	464,810	54	1.4	76.37	46,695.8	96.11
news20	15,997	1,355,191	1.1	96.95	383.2	96.90
real-sim	57,848	20,958	0.3	97.44	938.3	97.82
yahoo-japan	140,963	832,026	3.1	92.63	20,955.2	93.31
webspam	280,000	254	25.7	93.35	15,681.8	99.26

Therefore, there is a need to develop optimization methods for large linear classification

Why Linear is Faster in Training and Testing?

- Let's check the prediction cost

$$\mathbf{w}^T\mathbf{x} + b \quad \text{versus} \quad \sum_{i=1}^{l} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b$$

- If $K(\mathbf{x}_i, \mathbf{x}_j)$ takes $O(n)$, then

$$O(n) \quad \text{versus} \quad O(nl)$$

- Linear is much cheaper; reason:

  for linear, $\mathbf{x}_i$ is available

  but

  for kernel, $\phi(\mathbf{x}_i)$ is not

Optimization for Linear Classification

- Now a popular topic in both machine learning and optimization
- Most are based on first-order information: coordinate descent, stochastic gradient descent, or cutting plane
- The reason is again that no need to accurately solve optimization problems
- Let's see another development for linear classification

Optimization for Linear Classification (Cont'd)

- Martens (2010) and Byrd et al. (2011) propose the so called "Hessian free" approach
- Let's rewrite linear SVM as the following form

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

- What if we use a subset in the second term

$$\frac{C}{|B|}\sum_{i\in B}\max(1 - y_i\mathbf{w}^T\mathbf{x}_i, 0)$$

Optimization for Linear Classification (Cont'd)

- Then both gradient and Hessian-vector products can be cheaper
- That is, if there are enough data, then the average training loss should be similar
- This is a good example to take machine learning properties in designing optimization algorithms

Optimization for Linear Classification (Cont'd)

Lessons

- We must know the practical use of machine learning in order to design suitable optimization algorithms
- Here is how I started developing optimization algorithms for linear SVM
- In 2006, I visited at Yahoo! for six months. I learned that

  1. Document classification is heavily used
  2. Accuracy of linear and nonlinear is similar for documents

Outline

1. Introduction
2. Optimization methods for kernel support vector machines
3. Optimization methods for linear support vector machines
4. Discussion and conclusions

Machine Learning Software

- Algorithms discussed in this talk are related to my machine learning software
- LIBSVM (Chang and Lin, 2011):

  One of the most popular SVM packages; cited more than 11,000 times on Google Scholar
- LIBLINEAR (Fan et al., 2008):

  A library for large linear classification; popular in Internet companies
- The core of an SVM package is an optimization solver

Machine Learning Software (Cont'd)

- But designing machine learning software is quite different from optimization packages
- You need to consider prediction, validation, and others
- Also issues related to users (e.g., easy of use, interface, etc.) are very important for machine learning packages

Conclusions

- Optimization has been very useful for machine learning
- We need to take machine learning knowledge into account for designing suitable optimization algorithms
- The interaction between optimization and machine learning is very interesting and exciting.