Large-scale Linear Classification: Status and Challenges

Chih-Jen Lin
Department of Computer Science
National Taiwan University

San Francisco Machine Learning Meetup, October 30, 2014

Outline

Outline

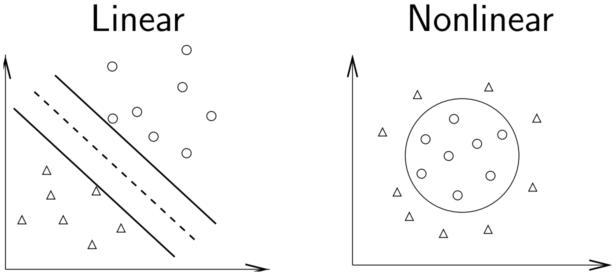
Linear Classification

- The model is a weight vector $\boldsymbol{w}$ (for binary classification)
- The decision function is

$$\mathrm{sgn}(\boldsymbol{w}^T \boldsymbol{x})$$

- Although many new and advanced techniques are available (e.g., deep learning), linear classifiers remain to be useful because of their simplicity
- We will give an overview of this topic in this talk

Linear and Kernel Classification



Linear: data in the original input space; nonlinear: data mapped to other spaces

Original: [height, weight]

Nonlinear: [height, weight, weight/height^2]

Kernel is one of the nonlinear methods

Linear and Nonlinear Classification

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

- Kernel methods: data mapped to another space

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

  $\phi(\boldsymbol{x})^T\phi(\mathbf{y})$ easily calculated; no good control on $\phi(\cdot)$
- Linear classification + feature engineering:

  Directly use $\boldsymbol{x}$ without mapping. But $\boldsymbol{x}$ may have been carefully generated. Full control on $\boldsymbol{x}$

We will focus on the 2nd type of approaches in this talk

Why Linear Classification?

- If $\phi(\boldsymbol{x})$ is high dimensional, decision function

$$\text{sgn}(\boldsymbol{w}^T \phi(\boldsymbol{x}))$$

  is expensive
- Kernel methods:

$$\boldsymbol{w} \equiv \sum_{i=1}^{l} \alpha_i \phi(\boldsymbol{x}_i) \text{ for some } \boldsymbol{\alpha}, K(\boldsymbol{x}_i, \boldsymbol{x}_j) \equiv \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

  New decision function: $\text{sgn}\left(\sum_{i=1}^{l} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x})\right)$
- Special $\phi(\boldsymbol{x})$ so calculating $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is easy. Example:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) \equiv (\boldsymbol{x}_i^T \boldsymbol{x}_j + 1)^2 = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j), \phi(\boldsymbol{x}) \in R^{O(n^2)}$$

Why Linear Classification? (Cont'd)

- Prediction

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

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

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

- Kernel: cost related to size of training data

  Linear: cheaper and simpler

Linear is Useful in Some Places

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

  But training and testing are much faster
- Especially document classification

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

  Large and sparse data
- Training millions of data in just a few seconds

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

	Linear		RBF Kernel	
Data set	Time	Accuracy	Time	Accuracy
MNIST38	0.1	96.82	38.1	99.70
ijcnn1	1.6	91.81	26.8	98.69
covtype	1.4	76.37	46,695.8	96.11
news20	1.1	96.95	383.2	96.90
real-sim	0.3	97.44	938.3	97.82
yahoo-japan	3.1	92.63	20,955.2	93.31
webspam	25.7	93.35	15,681.8	99.26

Size reasonably large: e.g., yahoo-japan: 140k instances and 830k features

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

	Linear		RBF Kernel	
Data set	Time	Accuracy	Time	Accuracy
MNIST38	0.1	96.82	38.1	99.70
ijcnn1	1.6	91.81	26.8	98.69
covtype	1.4	76.37	46,695.8	96.11
news20	1.1	96.95	383.2	96.90
real-sim	0.3	97.44	938.3	97.82
yahoo-japan	3.1	92.63	20,955.2	93.31
webspam	25.7	93.35	15,681.8	99.26

Size reasonably large: e.g., yahoo-japan: 140k instances and 830k features

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

	Linear		RBF Kernel	
Data set	Time	Accuracy	Time	Accuracy
MNIST38	0.1	96.82	38.1	99.70
ijcnn1	1.6	91.81	26.8	98.69
covtype	1.4	76.37	46,695.8	96.11
news20	1.1	96.95	383.2	96.90
real-sim	0.3	97.44	938.3	97.82
yahoo-japan	3.1	92.63	20,955.2	93.31
webspam	25.7	93.35	15,681.8	99.26

Size reasonably large: e.g., yahoo-japan: 140k instances and 830k features

Binary Linear Classification

- Training data $\{y_i, \boldsymbol{x}_i\}, \boldsymbol{x}_i \in R^n, i = 1, \ldots, l, \ y_i = \pm 1$
- l: # of data, n: # of features

$$\min_{\boldsymbol{w}} f(\boldsymbol{w}), \quad f(\boldsymbol{w}) \equiv \frac{\boldsymbol{w}^T \boldsymbol{w}}{2} + C \sum_{i=1}^{l} \xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i)$$

- $\boldsymbol{w}^T \boldsymbol{w}/2$: regularization term (we have no time to talk about L1 regularization here)
- $\xi(\boldsymbol{w}; \boldsymbol{x}, y)$: loss function: we hope $y \boldsymbol{w}^T \boldsymbol{x} > 0$
- C: regularization parameter

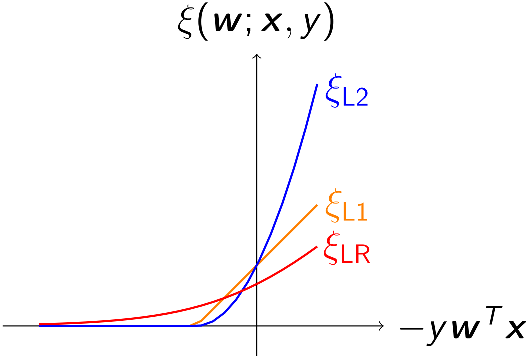
Loss Functions

- Some commonly used ones:

$$\xi_{\text{L1}}(\boldsymbol{w};\boldsymbol{x},y) \equiv \max(0, 1 - y\boldsymbol{w}^T\boldsymbol{x}), \tag{1}$$
$$\xi_{\text{L2}}(\boldsymbol{w};\boldsymbol{x},y) \equiv \max(0, 1 - y\boldsymbol{w}^T\boldsymbol{x})^2, \tag{2}$$
$$\xi_{\text{LR}}(\boldsymbol{w};\boldsymbol{x},y) \equiv \log(1 + e^{-y\boldsymbol{w}^T\boldsymbol{x}}). \tag{3}$$

- SVM (Boser et al., 1992; Cortes and Vapnik, 1995): (1)-(2)
- Logistic regression (LR): (3)

Loss Functions (Cont'd)



Their performance is usually similar

Optimization methods may be different because of differentiability

Outline

Optimization Methods

- Many unconstrained optimization methods can be applied
- For kernel, optimization is over a variable $\boldsymbol{\alpha}$ where

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

  We cannot minimize over $\boldsymbol{w}$ because it may be infinite dimensional
- However, for linear, minimizing over $\boldsymbol{w}$ or $\boldsymbol{\alpha}$ is ok

Optimization Methods (Cont'd)

Among unconstrained optimization methods,

- Low-order methods: quickly get a model, but slow final convergence
- High-order methods: more robust and useful for ill-conditioned situations

We will quickly discuss some examples and show both types of optimization methods are useful for linear classification

Optimization: 2nd Order Methods

- Newton direction (if twice differentiable)

$$\min_{\boldsymbol{s}} \quad \nabla f(\boldsymbol{w}^k)^T \boldsymbol{s} + \frac{1}{2} \boldsymbol{s}^T \nabla^2 f(\boldsymbol{w}^k) \boldsymbol{s}$$

- This is the same as solving Newton linear system

$$\nabla^2 f(\boldsymbol{w}^k)\boldsymbol{s} = -\nabla f(\boldsymbol{w}^k)$$

- Hessian matrix $\nabla^2 f(\boldsymbol{w}^k)$ too large to be stored

$$\nabla^2 f(\boldsymbol{w}^k) : n \times n, \quad n : \text{ number of features}$$

- But Hessian has a special form

$$\nabla^2 f(\boldsymbol{w}) = \mathcal{I} + CX^T DX,$$

Optimization: 2nd Order Methods (Cont'd)

- X: data matrix. D diagonal.
- Using Conjugate Gradient (CG) to solve the linear system. Only Hessian-vector products are needed

$$\nabla^2 f(\boldsymbol{w})\boldsymbol{s} = \boldsymbol{s} + C \cdot X^T(D(X\boldsymbol{s}))$$

- Therefore, we have a Hessian-free approach

Optimization: 1st Order Methods

- We consider L1-loss and the dual SVM problem

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & f(\boldsymbol{\alpha}) \\ \text{subject to} \quad & 0 \le \alpha_i \le C, \forall i, \end{aligned}$$

where

$$f(\boldsymbol{\alpha}) \equiv \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - \boldsymbol{e}^T \boldsymbol{\alpha}$$

and

$$Q_{ij} = y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j, \quad \boldsymbol{e} = [1, \ldots, 1]^T$$

- We will apply coordinate descent (CD) methods
- The situation for L2 or LR loss is very similar

1st Order Methods (Cont'd)

- Coordinate descent: a simple and classic technique
  Change one variable at a time
- Given current $\boldsymbol{\alpha}$. Let $\boldsymbol{e}_i = [0, \ldots, 0, 1, 0, \ldots, 0]^T$.

$$\min_{d} \; f(\boldsymbol{\alpha} + d\boldsymbol{e}_i) = \frac{1}{2} Q_{ii} d^2 + \nabla_i f(\boldsymbol{\alpha}) d + \text{constant}$$

- Without constraints

$$\text{optimal } d = -\frac{\nabla_i f(\boldsymbol{\alpha})}{Q_{ii}}$$

- Now $0 \leq \alpha_i + d \leq C$

$$\alpha_i \leftarrow \min\left(\max\left(\alpha_i - \frac{\nabla_i f(\boldsymbol{\alpha})}{Q_{ii}}, 0\right), C\right)$$

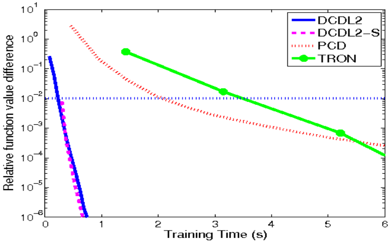
Comparisons

L2-loss SVM is used

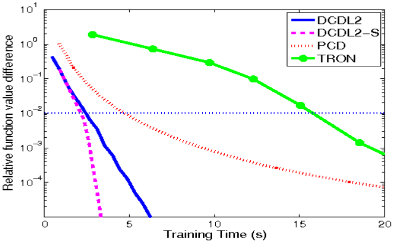
- DCDL2: Dual coordinate descent
- DCDL2-S: DCDL2 with shrinking
- PCD: Primal coordinate descent
- TRON: Trust region Newton method

This result is from Hsieh et al. (2008)

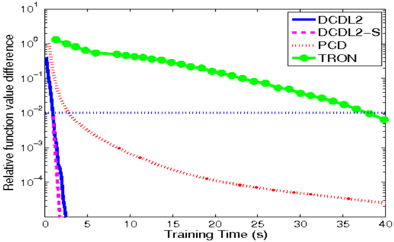
Objective values (Time in Seconds)



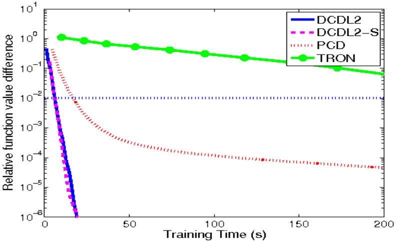
news20



rcv1



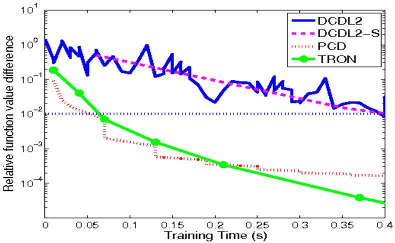
yahoo-japan



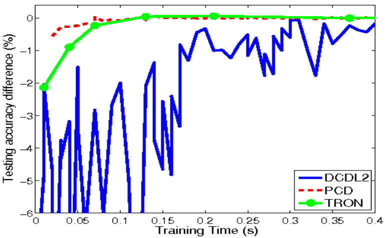
yahoo-korea

Low- versus High-order Methods

- We saw that low-order methods are efficient to give a model. However, high-order methods may be useful for difficult situations
- An example: # instance: 32,561, # features: 123



Objective value



Accuracy

features is small $\Rightarrow$ solving primal is more suitable

Outline

Outline

Dependency Parsing: an NLP Application

	Kernel		Linear	
	RBF	Poly-2	Linear	Poly-2
Training time	3h34m53s	3h21m51s	3m36s	3m43s
Parsing speed	0.7x	1x	1652x	103x
UAS	89.92	91.67	89.11	91.71
LAS	88.55	90.60	88.07	90.71

- We get faster training/testing, while maintain good accuracy
- But how to achieve this?

Linear Methods to Explicitly Train $\phi(\boldsymbol{x}_i)$

- Example: low-degree polynomial mapping:

$$\phi(\boldsymbol{x}) = [1, x_1, \ldots, x_n, x_1^2, \ldots, x_n^2, x_1 x_2, \ldots, x_{n-1}x_n]^T$$

- For this mapping, # features $= O(n^2)$
- Recall $O(n)$ for linear versus $O(nl)$ for kernel
- Now $O(n^2)$ versus $O(nl)$
- Sparse data

  $n \Rightarrow \bar{n}$, average # non-zeros for sparse data

  $\bar{n} \ll n \Rightarrow O(\bar{n}^2)$ may be much smaller than $O(l\bar{n})$

Handing High Dimensionality of $\phi(\boldsymbol{x})$

A multi-class problem with sparse data

n	Dim. of $\phi(\boldsymbol{x})$	l	$\bar{n}$	$\boldsymbol{w}$'s # nonzeros
46,155	1,065,165,090	204,582	13.3	1,438,456

- $\bar{n}$: average # nonzeros per instance
- Degree-2 polynomial is used
- Dimensionality of $\boldsymbol{w}$ is very high, but $\boldsymbol{w}$ is sparse

  Some training feature columns of $x_i x_j$ are entirely zero
- Hashing techniques are used to handle sparse $\boldsymbol{w}$

Discussion

- See more details in Chang et al. (2010)
- If $\phi(\boldsymbol{x})$ is too high dimensional, people have proposed projection or hashing techniques to use fewer features as approximations

  Examples: Kar and Karnick (2012); Pham and Pagh (2013)
- This has been used in computational advertising (Chapelle et al., 2014)

Outline

Example: Classifier in a Small Device

- In a sensor application (Yu et al., 2013), the classifier can use less than 16KB of RAM

Classifiers	Test accuracy	Model Size
Decision Tree	77.77	76.02KB
AdaBoost (10 trees)	78.84	1,500.54KB
SVM (RBF kernel)	85.33	1,287.15KB

- Number of features: 5
- We consider a degree-3 polynomial mapping

$$\text{dimensionality} = \binom{5+3}{3} + \text{ bias term} = 57.$$

Example: Classifier in a Small Device

- One-against-one strategy for 5-class classification

$$\binom{5}{2} \times 57 \times 4\text{bytes} = 2.28\text{KB}$$

  Assume single precision

- Results

SVM method	Test accuracy	Model Size
RBF kernel	85.33	1,287.15KB
Polynomial kernel	84.79	2.28KB
Linear kernel	78.51	0.24KB

Outline

Big-data Linear Classification

- Nowadays data can be easily larger than memory capacity
- Disk-level linear classification: Yu et al. (2012) and subsequent developments
- Distributed linear classification: recently an active research topic
- Example: we can parallelize the 2nd-order method discussed earlier. Recall the Hessian-vector product

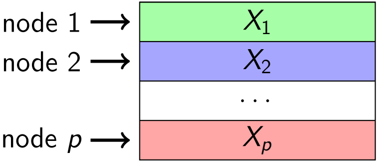
$$\nabla^2 f(\boldsymbol{w})\boldsymbol{s} = \boldsymbol{s} + C \cdot X^T(D(X\boldsymbol{s}))$$

Parallel Hessian-vector Product

- Hessian-vector products are the computational bottleneck

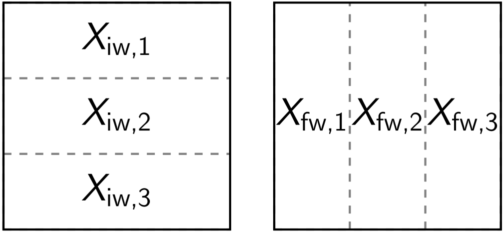
$$X^T D X \boldsymbol{s}$$

- Data matrix X is now distributedly stored



$$X^T D X \boldsymbol{s} = X_1^T D_1 X_1 \boldsymbol{s} + \cdots + X_p^T D_p X_p \boldsymbol{s}$$

Instance-wise and Feature-wise Data Splits



Instance-wise Feature-wise

- We won't have time to get into details. But their communication cost is different
- Data moved per Hessian-vector product
  Instance-wise: $O(n)$, Feature-wise: $O(l)$

Discussion: Dostributed Training or Not?

- One can always subsample data to one machine for deep analysis
- Deciding to do distributed classification or not is an issue
- In some areas distributed training has been successfully applied
- One example is CTR (click-through rate) prediction in computational advertising

Discussion: Platform Issues

- For the above-mentioned Newton methods, we have MPI and Spark implementations
- We are preparing the integration to Spark MLlib
- Other existing distributed linear classifiers include Vowpal_Wabbit from Yahoo!/Microsoft and Sibyl from Google
- Platforms such as Spark are still being rapidly changed. This is a bit annoying
- A carefully implementation may sometimes thousands times faster than a casual one

Discussion: Design of Distributed Algorithms

- On one computer, often we do batch rather than online learning

  Online and streaming learning may be more useful for big-data applications
- The example (Newton method) we showed is a synchronous parallel algorithms

  Maybe asynchronous ones are better for big data?

Outline

Resources on Linear Classification

- Since 2007, we have been actively developing the software LIBLINEAR for linear classification

  www.csie.ntu.edu.tw/~cjlin/liblinear
- A distributed extension (MPI and Spark) is now available
- An earlier survey on linear classification is Yuan et al. (2012)

  Recent Advances of Large-scale Linear Classification. *Proceedings of IEEE*, 2012

  It contains many references on this subject

Conclusions

- Linear classification is an old topic; but recently there are new and interesting applications
- Kernel methods are still useful for many applications, but linear classification + feature engineering are suitable for some others
- Linear classification will continue to be used in situations ranging from small-model to big-data applications

Acknowledgments

- Many students have contributed to our research on large-scale linear classification
- We also thank the partial support from National Science Council of Taiwan