Recent Advances in Large-scale Linear Classification

Chih-Jen Lin
Department of Computer Science
National Taiwan University

Talk at Asian Conference on Machine Learning, November, 2013

- Part of this talk is based on our survey paper (Yuan et al., 2012)

  Recent Advances of Large-scale Linear Classification. *Proceedings of IEEE*, 2012
- It's also related to our development of the software LIBLINEAR

  `www.csie.ntu.edu.tw/~cjlin/liblinear`
- Due to time constraints, we will give overviews instead of deep technical details.

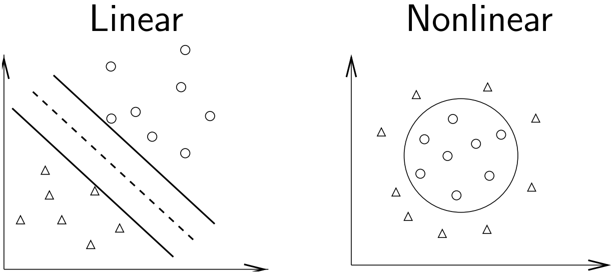
Outline

- Introduction
- Optimization Methods
- Extension of Linear Classification
- Discussion and Conclusions

Outline

- Introduction
- Optimization Methods
- Extension of Linear Classification
- Discussion and Conclusions

Linear and Nonlinear Classification



By linear we mean a linear function is used to separate data in the original input space

Original: [height, weight]

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

Kernel is one of the methods for nonlinear

Linear and Nonlinear Classification (Cont'd)

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

- Kernel methods: data mapped to another space

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

  $\phi(\mathbf{x})^T\phi(\mathbf{y})$ easily calculated; no good 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 $\phi(\mathbf{x})$

We will focus on linear here

Why Linear Classification?

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

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

is expensive. So kernel methods use

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

Then new decision function is $\text{sgn}\left(\sum_{i=1}^{l} \alpha_i K(\mathbf{x}_i, \mathbf{x})\right)$

- Special $\phi(\mathbf{x})$ so calculating $K(\mathbf{x}_i, \mathbf{x}_j)$ is easy. Example:

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

Why Linear Classification? (Cont'd)

- However, kernel is still expensive
- Prediction

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

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

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

- Nonlinear: more powerful to separate 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 $\{y_i, \mathbf{x}_i\}, \mathbf{x}_i \in R^n, i = 1, \ldots, l, \ y_i = \pm 1$
- l: # of data, n: # of features

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

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

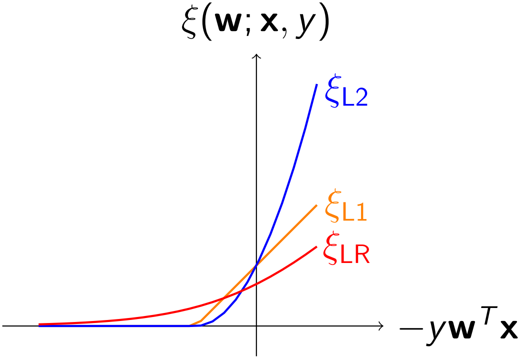
Loss Functions

- Some commonly used ones:

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

- SVM (Boser et al., 1992; Cortes and Vapnik, 1995): (1)-(2)
- Logistic regression (LR): (3); no reference because it can be traced back to 19th century

Loss Functions (Cont'd)



Their performance is usually similar

Loss Functions (Cont'd)

However, optimization methods for them may be different

ξ_{L1}: not differentiable
ξ_{L2}: differentiable but not twice differentiable
ξ_{LR}: twice differentiable

Outline

- Introduction
- Optimization Methods
- Extension of Linear Classification
- Discussion and Conclusions

Optimization: 2nd Order Methods

- Newton direction

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

- This is the same as solving Newton linear system

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

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

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

- But Hessian has a special form

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

Optimization: 2nd Order Methods (Cont'd)

- X: data matrix. D diagonal. For logistic regression,

$$D_{ii} = \frac{e^{-y_i \mathbf{w}^T \mathbf{x}_i}}{1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i}}$$

- Using CG to solve the linear system. Only Hessian-vector products are needed

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

- Therefore, we have a Hessian-free approach

2nd-order Methods (Cont'd)

- In LIBLINEAR, we use the trust-region + CG approach by Steihaug (1983); see details in Lin et al. (2008)
- What if we use L2 loss? It's differentiable but not twice differentiable

$$\xi_{L2}(\mathbf{w};\mathbf{x},y) \equiv \max(0, 1 - y\mathbf{w}^T\mathbf{x})^2$$

- We can use generalized Hessian (Mangasarian, 2002). Details not discussed here

Optimization: 1st Order Methods

- We consider L1-loss and the dual SVM problem

$$\begin{array}{ll} \min_{\boldsymbol{\alpha}} & f(\boldsymbol{\alpha}) \\ \text{subject to} & 0 \leq \alpha_i \leq C, \forall i, \end{array}$$

  where

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

  and

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

- We will apply coordinate descent 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 $\mathbf{e}_i = [0, \ldots, 0, 1, 0, \ldots, 0]^T$.

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

- Without constraints

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

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

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

1st Order Methods (Cont'd)

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

- $O(ln)$ cost; l:# data, n: # features. But we can define
$$\mathbf{u} \equiv \sum_{j=1}^{l} y_j \alpha_j \mathbf{x}_j,$$
- Easy gradient calculation: costs $O(n)$

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

1st Order Methods (Cont'd)

- All we need is to maintain $\mathbf{u}$

$$\mathbf{u} = \sum_{j=1}^{l} y_j \alpha_j \mathbf{x}_j,$$

- If

$$\bar{\alpha}_i : \text{ old } ; \qquad \alpha_i : \text{ new}$$

  then

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

  Also costs $O(n)$

References: first use for SVM probably by Mangasarian and Musicant (1999); Friess et al. (1998), but popularized for linear SVM by Hsieh et al. (2008)

1st Order Methods (Cont'd)

Summary of the dual coordinate descent method

- Given initial $\boldsymbol{\alpha}$ and find $\mathbf{u} = \sum_i y_i \alpha_i \mathbf{x}_i$.
- While $\boldsymbol{\alpha}$ is not optimal (Outer iteration)

  For $i = 1, \ldots, l$ (Inner iteration)

  (a) $\bar{\alpha}_i \leftarrow \alpha_i$

  (b) $G = y_i \mathbf{u}^T \mathbf{x}_i - 1$

  (c) If α_i can be changed

  $$\alpha_i \leftarrow \min(\max(\alpha_i - G/Q_{ii}, 0), C)$$
  $$\mathbf{u} \leftarrow \mathbf{u} + (\alpha_i - \bar{\alpha}_i) y_i \mathbf{x}_i$$

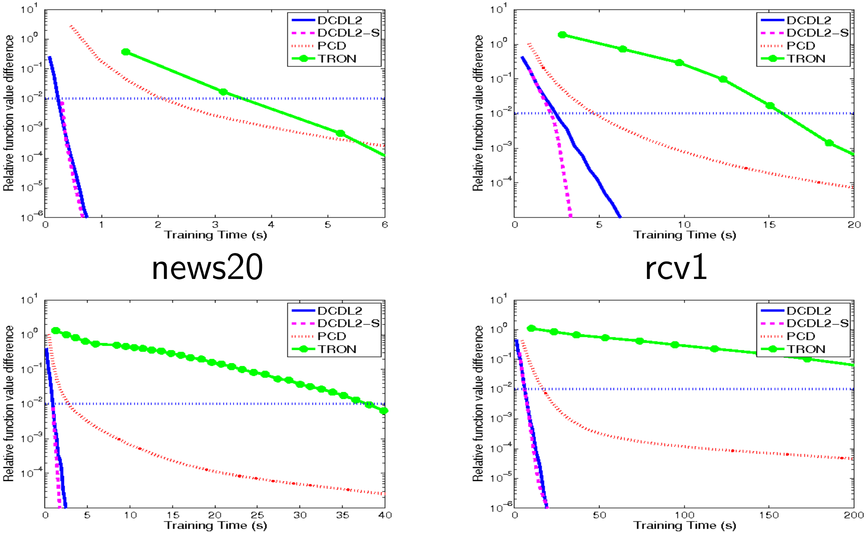
Comparisons

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

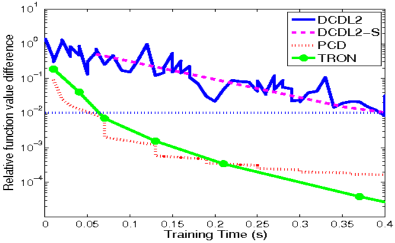
yahoo-japan yahoo-korea

Analysis

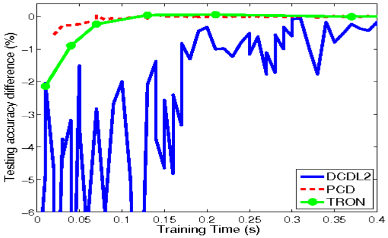
- First-order methods can quickly get a model
- But second-order methods are more robust and faster for ill-conditioned situations
- Both type of optimization methods are useful for linear classification

An Example When # Features Small

- # instance: 32,561, # features: 123



Objective value



Accuracy

If number of features is small, solving primal is more suitable

Outline

- Introduction
- Optimization Methods
- Extension of Linear Classification
- Discussion and Conclusions

Extension of Linear Classification

- Linear classification can be extended in different ways
- An important one is to approximate nonlinear classifiers
- Goal: better accuracy of nonlinear but faster training/testing
- Examples
  1. Explicit data mappings + linear classification
  2. Kernel approximation + linear classification
- I will focus on the first

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

- Example: low-degree polynomial mapping:

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

- For this mapping, # features $= O(n^2)$
- When is it useful?

  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 $O(l\bar{n})$

Example: Dependency Parsing

A multi-class problem with sparse data

n	Dim. of $\phi(\mathbf{x})$	l	$\bar{n}$	$\mathbf{w}$'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 $\mathbf{w}$ is too large, but $\mathbf{w}$ is sparse
- Some interesting Hashing techniques are used to handle sparse $\mathbf{w}$

Example: Dependency Parsing (Cont'd)

	LIBSVM		LIBLINEAR	
	RBF	Poly	Linear	Poly
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, but maintain good accuracy
- See detailed discussion in Chang et al. (2010)

Example: Classifier in a Small Device

- In a sensor application (Yu et al., 2013), the classifier must 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 mapping

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

Example: Classifier in a Small Device (Cont'd)

- 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

Example: Classifier in a Small Device (Cont'd)

- Running time (in seconds)

	LIBSVM	LIBLINEAR	
		Primal	Dual
Training time	30,519.10	1,368.25	4,039.20

- LIBSVM: polynomial kernel
- LIBLINEAR: training polynomial expansions
  primal: 2nd-order method; dual: 1st-order
- LIBLINEAR dual: slow convergence. Now

$$\#\text{data} \gg \#\text{features} = 57$$

Discussion

- Unfortunately, polynomial mappings easily cause high dimensionality. Some have proposed "projection" techniques to use fewer features as approximations

  Examples: Kar and Karnick (2012); Pham and Pagh (2013)
- Recently, ensemble of tree models (e.g., random forests or GBDT) become very useful. But under model-size constraints (the 2nd application), linear may still be the way to go

Outline

- Introduction
- Optimization Methods
- Extension of Linear Classification
- Discussion and Conclusions

Big-data Linear Classification

- Shared and distributed scenarios are very different
- Here I discuss more about distributed classification
- The major saving is parallel data loading
- But high communication cost is a big concern

Big-data Linear Classification (Cont'd)

- Data classification if often only one component of the whole workflow
- Example: distributed feature generation may be more time consuming than classification
- This explains why so far not many effective packages are available for big-data classification
- Many research and engineering issues remain to be solved

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
- Advantages of linear: because of working on $\mathbf{x}$, easier for feature engineering
- We expect that linear classification can be widely used in situations ranging from small-model to big-data classification

References I

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.

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.

T.-T. Friess, N. Cristianini, and C. Campbell. The kernel adatron algorithm: a fast and simple learning procedure for support vector machines. In *Proceedings of 15th Intl. Conf. Machine Learning*. Morgan Kaufman Publishers, 1998.

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.

P. Kar and H. Karnick. Random feature maps for dot product kernels. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 583–591, 2012.

References II

C.-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region Newton method for large-scale logistic regression. *Journal of Machine Learning Research*, 9:627–650, 2008. URL http://www.csie.ntu.edu.tw/~cjlin/papers/logistic.pdf.

O. L. Mangasarian. A finite Newton method for classification. *Optimization Methods and Software*, 17(5):913–929, 2002.

O. L. Mangasarian and D. R. Musicant. Successive overrelaxation for support vector machines. *IEEE Transactions on Neural Networks*, 10(5):1032–1037, 1999.

N. Pham and R. Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 239–247, 2013.

T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20:626–637, 1983.

T. Yu, D. Wang, M.-C. Yu, C.-J. Lin, and E. Y. Chang. Careful use of machine learning methods is needed for mobile applications: A case study on transportation-mode detection. Technical report, Studio Engineering, HTC, 2013. URL http://www.csie.ntu.edu.tw/~cjlin/papers/transportation-mode/casestudy.pdf.

G.-X. Yuan, C.-H. Ho, and C.-J. Lin. Recent advances of large-scale linear classification. *Proceedings of the IEEE*, 100(9):2584–2603, 2012. URL http://www.csie.ntu.edu.tw/~cjlin/papers/survey-linear.pdf.