A Coordinate Gradient Descent Method for ℓ_1-regularized Convex Minimization

Sangwoon Yun
Singapore-MIT Alliance, National University of Singapore

January 22, 2009

(Joint work with Kim-Chuan Toh (NUS))

Outline

- ℓ_1-regularized Linear Least Squares problem $\&$ ℓ_1-regularized Logistic Regression problem
- General Problem Model: ℓ_1-regularized Convex Minimization
- Coordinate Gradient Descent Method
- Convergence Results
- Numerical Experience

ℓ_1-regularized Linear Least Squares problem & ℓ_1-regularized Logistic Regression problem

ℓ_1-regularized linear least squares (ℓ_1LS) problem

Find x so that $Ax - b \approx 0$ and x has "few" nonzeros.

Formulate this as an unconstrained convex optimization problem:

$$\min_{x \in \Re^n} \ \frac{1}{2}\|Ax - b\|_2^2 + \mu \|x\|_1,$$

where $\mu > 0$, $A \in \Re^{m \times n}$ is dense, $m < n$ or even $m \ll n$, and $b \in \Re^m$.

Ex: Basis Pursuit Denoising problem in Compressed Sensing.

Recent methods

- Homotopy methods (Osborne et al. '00, Donoho $\&$ Tsaig '06).
- Interior Point method (Kim et al. '07:l1-ls).
- Iterative Shrinkage/Thresholding (IST) method (Figueiredo and Nowak '03, Daubechies et al. '04).
- Gradient Projection method (Figueiredo et al. '07:GPSR).
- Fixed-Point Continuation (FPC) method (Hale et al. '07).

ℓ_1-regularized logistic regression (ℓ_1LR) problem

Proposed as a promising method for feature selection in classification problems.

$$\min_{w\in\Re^{n-1},v\in\Re} \frac{1}{m}\sum_{i=1}^{m}\log(1+\exp(-(w^T a_i + v b_i))) + \mu\|w\|_1,$$

where $a_i = b_i z_i$ and $(z_i, b_i) \in \Re^{n-1} \times \{-1, 1\},\ i = 1, ..., m$ are a given set of (observed or training) examples.

Recent methods

- Genkin et al. '06.
- Koh et al. '07: Interior Point (l1-logreg) method.
- Wright et al. '07 (SpaRSA).

General Problem Model: ℓ_1-regularized Convex Minimization

P

$$\min_x \quad F_\rho(x) := f(x) + \rho^T |x|$$

$f : \Re^n \to \Re$ is smooth (i.e., continuously differentiable) and convex.

ρ is a given nonnegative weight vector.

$|x|$ denotes obtained from x by taking absolute values.

- For ℓ_1LS, $\rho_i = \mu$ for all i.
- For ℓ_1LR, $\rho_i = \mu$ for all $i = 1, ..., n-1$ and $\rho_n = 0$.

Coord. Gradient Descent Method

Descent direction.

For $x \in \Re^n$, choose $\mathcal{J}(\neq \emptyset) \subseteq \mathcal{N} = \{1, ..., n\}$ and $H \succ 0_n$, Then solve

$$\min_{d | d_j = 0 \ \forall j \notin \mathcal{J}} \{\nabla f(x)^T d + \frac{1}{2} d^T H d + \rho^T |x + d| - \rho^T |x|\}$$

direc. subprob

Let $d^H(x; \mathcal{J})$ and $q^H(x; \mathcal{J})$ be the opt. soln & obj. value of the direc. subprob.

Facts:

- $d^H(x; \mathcal{N}) = 0 \ \Leftrightarrow F'_\rho(x; d) \geq 0 \ \forall d \in \Re^n$. stationarity
- H is diagonal $\Rightarrow d^H(x; \mathcal{J}) = \sum_{j \in \mathcal{J}} d^H(x; j)$, $q^H(x; \mathcal{J}) = \sum_{j \in \mathcal{J}} q^H(x; j)$. separab.
- $q^H(x; \mathcal{J}) \leq -\frac{1}{2} d^T H d$ where $d = d^H(x; \mathcal{J})$.

Stepsize: **Armijo rule**

Choose α to be the largest element of $\{\beta^k\}_{k=0,1,\ldots}$ satisfying

$$F_\rho(x+\alpha d) - F_\rho(x) \leq \sigma\alpha q^H(x;\mathcal{J}) \quad (0<\beta<1,\, 0<\sigma<1).$$

For the ℓ_1-regularized linear least squares problem, the minimization rule

$$\alpha \in \arg\min\{F_\rho(x+td) \mid t \geq 0\}$$

or the limited minimization rule

$$\alpha \in \arg\min\{F_\rho(x+td) \mid 0 \leq t \leq s\},$$

where $0 < s < \infty$, can also be used.

Choose $\mathcal{J}$:

- Gauss-Southwell-r rule:

$$\|d^D(x;\mathcal{J})\|_\infty \geq \upsilon \|d^D(x;\mathcal{N})\|_\infty$$

where $0 < \upsilon \leq 1$, $D \succ 0_n$ is diagonal (e.g., $D = \mathrm{diag}(H)$).

- Gauss-Southwell-q rule:

$$q^D(x;\mathcal{J}) \leq \upsilon \; q^D(x;\mathcal{N}),$$

Where $0 < \upsilon \leq 1$, $D \succ 0_n$ is diagonal (e.g., $D = \mathrm{diag}(H)$).

- This coord. grad. descent approach may be viewed as a hybrid of coordinate descent and IST (or fixed point) method.

- If H is diagonal, then subproblems can be solved in parallel:

$$d^H(x;\mathcal{N})_j = -\text{median}\{(\nabla f(x)_j - \rho_j)/H_{jj}, x_j, (\nabla f(x)_j + \rho_j)/H_{jj}\}.$$

Convergence Results

Global convergence If

- $0 \prec \underline{\lambda} I \preceq D,\ H \preceq \bar{\lambda} I,$
- α is chosen by Armijo rule,

then every cluster point of the x-sequence generated by CGD method using GS-r or GS-q rule is a stationary point of F_ρ.

Local convergence rate In addition, if f satisfy any of the following assumptions, then the x-sequence generated by CGD method using GS-q rule converges at R-linear rate.

C1: f is strongly convex, ∇f is Lipschitz cont.
C2: f is quadratic.
C3: $f(x) = g(Ex) + q^T x$, where $E \in \Re^{m \times n}$, $q \in \Re^n$, g is strongly convex, ∇g is Lipschitz cont. on $\Re^m$.

Numerical Experience

1. ℓ_1-regularized linear least squares problem (Compressed Sensing):

$$\min_{x \in \Re^n} \|Ax - b\|_2^2 + \bar{\mu}\|x\|_1 \qquad (\bar{\mu} > 0) \tag{1}$$

- Implement CGD method in Matlab.

- Choose H as a constant multiple of the identity matrix

$$H = \theta I,$$

Initially, we set $\theta^{\text{init}} = \|Au\|_2^2$ where u is a random unit vector. Then updated as follows:

$$\theta^{\text{new}} = \begin{cases} \max\{\theta/\alpha, 1\} & \text{if } \alpha > 10 \\ \min\{\theta/\alpha, 1\} & \text{if } \alpha < 10^{-1} \\ \theta & \text{otherwise.} \end{cases}$$

- Choose $\mathcal{J}$ by Gauss-Southwell-r rule,

$$\mathcal{J} = \left\{ j \mid |d_j^H(x;\mathcal{N})| \geq \upsilon \|d^H(x;\mathcal{N})\|_\infty \right\}.$$

Or by Gauss-Southwell-q rule,

$$\mathcal{J} = \left\{ j \mid q^H(x;j) \leq \upsilon \min_i q^H(x;i) \right\}.$$

- Choose α by the minimization rule.

$$\alpha \in \arg\min \quad \{ \frac{a_1}{2} t^2 - a_2 t + c \sum_{j \in \mathcal{J}} |x_j + t d_j| + a_3 \mid t > 0 \},$$

where $a_1 = \|Ad\|_2^2$, $a_2 = (Ad)^T(b - Ax)$, $a_3 = \frac{1}{2}\|b - Ax\|_2^2$, and $d = d^H(x;\mathcal{J})$.

- Termination Criterion:

$$\|Hd^H(x;\mathcal{N})\|_\infty \leq 10^{-3} \qquad \text{or} \qquad \frac{\|Hd^H(x;\mathcal{N})\|_\infty}{\max\{1,\|x\|_\infty\}} \leq 10^{-3}$$

- Continuation Strategy: solving a sequence of ℓ_1LS problems defined by a decreasing sequence $\{\mu^0,\mu^1,\ldots,\mu^\ell=\bar{\mu}\}$.

Initially, we set $\mu^0 = 0.01\|A^Tb\|_\infty$ and update $\mu^{\text{new}} = \max\{0.25\mu,\bar{\mu}\}$ whenever the following criterion is satisfied:

$$\frac{\|Hd^H(x;\mathcal{N})\|_\infty}{\max\{1,\|x\|_\infty\}} \leq \max\{10^{\lfloor log(\mu)\rfloor},10^{-3}\}.$$

- $A \in \Re^{m \times n}$ is obtained by filling it with independent samples of the Gaussian distribution and then orthonormalizing the rows ($m = 2048, n = 8192$).

the original signal x_s contains 320 randomly placed ± 1 spikes.

$b = Ax_s + \xi$, where ξ is Gaussian white noise with variance $(0.01\|Ax_s\|_2)^2$.

- Comparison with l1-ls (interior-point), GPSR-BB (gradient projection), and FPC-BB (fixed-point continuation).

- Require only matrix-vector mults. involving A and A^T at each iteration.

Matrix-vector mults. involving A^T cost $O(mn)$ opers. (same as the computational cost of other algorithms).

But matrix-vector mults. involving A cost $O(m|\mathcal{J}|)$ opers. (only need to evaluate Ad).

Sorting is needed to find the stepsize, the cost is $O(|\mathcal{J}| \ln |\mathcal{J}|)$.

Test Results

- To perform this comparison, first ran l1-ls and then each of the others until each reached the same objective value reached by l1-ls (10 random instances).

	l1-ls	CGD-GS-q	CGD-GS-r	GPSR-BB	FPC-BB
	$n = 8192, m = 2048, \bar{\mu} = 0.05\|\|A^T b\|\|_\infty$				
mean iterations	12	17	21	28	39
mean nnz(x)	842	399	437	574	629
mean CPU time	2.0e+01	9.2e-01	1.1e+00	2.8e+00	3.8e+00
mean error	1.4e-01	1.5e-01	1.5e-01	1.6e-01	1.5e-01
	$n = 8192, m = 2048, \bar{\mu} = 0.01\|\|A^T b\|\|_\infty$				
mean iterations	13	39	45	111	57
mean nnz(x)	1026	800	1122	928	1204
mean CPU time	2.8e+01	2.0e+00	2.3e+00	1.1e+01	5.5e+00
mean error	3.4e-02	4.6e-02	4.5e-02	4.0e-02	3.7e-02
	$n = 8192, m = 2048, \bar{\mu} = 0.005\|\|A^T b\|\|_\infty$				
mean iterations	13	48	53	1001	58
mean nnz(x)	1245	1206	1704	8175	1588
mean CPU time	3.2e+01	2.5e+00	2.8e+00	9.4e+01	5.6e+00
mean error	2.3e-02	2.8e-02	2.8e-02	7.1e-01	3.1e-02

2. ℓ_1-regularized logistic regression problem:

$$\min_{w\in\Re^{n-1},v\in\Re} \frac{1}{m}\sum_{i=1}^{m}\log(1+\exp(-(w^T a_i + v b_i))) + \bar{\mu}\|w\|_1, \qquad (\bar{\mu} > 0)$$

- Implement CGD method in Matlab.

- Choose H as a diagonal Hessian approximation

$$H = \operatorname{diag}\left[\min\{\max\{\nabla^2 f(x)_{jj}, 10^{-10}\}, 10^{10}\}\right]_{j=1,\ldots,n},$$

where $x = ((w)^T, v)^T$.

- Choose $\mathcal{J}$ by Gauss-Southwell-r rule, or by Gauss-Southwell-q rule.

- Choose α by the Armijo rule.

- The CGD method is terminated when

$$\|Hd^H(x;\mathcal{N})\|_\infty \leq 10^{-6}.$$

- Numerical tests on some large two-class (sparse) data classification problems.

- Comparison with l1-logreg (interior-point) and SpaRSA.

Test Results

- To perform this comparison, first ran l1-logreg and then each of the others until each reached the same objective value reached by l1-logreg.

	l1-logreg (10^{-4})	CGD-GS-q	CGD-GS-r	SpaRSA
leu	$n = 7130, m = 38, \bar{\mu} = 0.01\mu_{\max}$			
iterations	25	92	146	fail
obj value	3.14477e-02	3.14458e-02	3.14212e-02	
CPU time	1.1e+00	6.6e-01	1.0e+00	
rcv1	$n = 47237, m = 20242, \bar{\mu} = 0.01\mu_{\max}$			
iterations	28	118	105	191
obj value	2.12420e-01	2.12420e-01	2.12420e-01	2.12420e-01
CPU time	1.1e+01	7.5e+00	6.9e+00	7.5e+00
real-sim	$n = 20959, m = 72309, \bar{\mu} = 0.01\mu_{\max}$			
iterations	22	71	72	88
obj value	2.14289e-01	2.14289e-01	2.14287e-01	2.14289e-01
CPU time	1.4e+01	7.5e+00	7.9e+00	7.6e+00

$$\mu_{\max} = \frac{1}{m}\left\|\frac{m_-}{m}\sum_{b_i=1} a_i + \frac{m_+}{m}\sum_{b_i=-1} a_i\right\|_\infty$$

- Randomly generated problem (Features of positive (negative) examples are independent and identically distributed, dense).

	l1-logreg 10^{-4}	CGD-GS-q 10^{-6}	CGD-GS-r 10^{-6}
10 random	$n = 10001, m = 1000, \bar{\mu} = 0.01\mu_{\max}$		
mean iterations	20	230	260
mean CPU time	2.4e+02	1.2e+01	1.4e+01
10 random	$n = 1001, m = 100, \bar{\mu} = 0.01\mu_{\max}$		
mean iterations	17	187	220
mean CPU time	1.9e-01	2.4e-01	2.9e-01
10 random	$n = 1001, m = 10000, \bar{\mu} = 0.01\mu_{\max}$		
mean iterations	16	82	88
mean CPU time	2.3e+02	4.8e+00	5.2e+00
10 random	$n = 101, m = 1000, \bar{\mu} = 0.01\mu_{\max}$		
mean iterations	14	60	66
mean CPU time	1.9e-01	6.8e-02	9.0e-02

- The computational cost for the search direction of l1-logreg is $O(\min(n,m)^2 \max(n,m))$ opers. per iteration. In contrast, the computational cost of CGD is $O(mn)$ opers. per iteration.

Conclusions

1. Numerical results shows that CGD method is efficient not only in minimizing a quadratic convex smooth function with ℓ_1-regularization but also in minimizing a nonquadratic convex smooth function with ℓ_1-regularization.

2. The Barzilai-Borwein steps accelerate the convergence of GPSR and FPC. But the performance of CGD method using BB steps was generally worse than that of CGD method using the minimization rule when it was applied for solving ℓ_1-regularized linear least squares problems. Can the efficiency be further improved by using nonmonotone descent?

3. When CGD method with continuation was applied to solve ℓ_1-regularized logistic regression problems, its overall performance was worse than that of CGD method without continuation. Can other acceleration techniques be developed for solving ℓ_1-regularized logistic regression problems?

Thank you!

Yun S. and Toh K.-C., A coordinate gradient descent method for ℓ_1-regularized convex minimization.
(PDF file available at http://www.math.nus.edu.sg/ matys/index.html)

Discrete Cosine Transform:

- $A \in \Re^{m \times n}$ is a partial DCT matrix whose m rows were chosen at random (uniformly) from the $n \times n$ discrete cosine transform (DCT) matrix ($m = 2048, n = 8192$).

the original signal x_s contains 320 randomly placed ± 1 spikes.

$b = Ax_s + \xi$, where ξ is Gaussian white noise with variance $(0.01\|Ax_s\|_2)^2$.

- Comparison with l1-ls, GPSR-BB, and FPC-BB.

- Require only matrix-vector mults. involving A and A^T at each iteration.

By using fast transform operators, such as the FFT, matrix-vector mults. involving A and A^T cost $O(n \ln n)$ opers.

But we lose the advantage of "sparse" multiplication.

Test Results

- To perform this comparison, first ran l1-ls and then each of the others until each reached the same objective value reached by l1-ls (10 random instances).

	l1-ls	CGD-GS-q	CGD-GS-r	GPSR-BB	FPC-BB
	$n = 8192, m = 2048, \bar{\mu} = 0.01\|\|A^T b\|\|_\infty$				
mean iterations	13	46	46	119	58
mean nnz(x)	912	830	1090	917	1215
mean CPU time	2.6e+00	4.2e-01	4.1e-01	9.4e-01	4.3e-01
mean error	3.3e-02	4.4e-02	4.3e-02	3.8e-02	3.6e-02
	$n = 8192, m = 2048, \bar{\mu} = 0.005\|\|A^T b\|\|_\infty$				
mean iterations	13	55	54	1001	60
mean nnz(x)	1204	1199	1584	8179	1593
mean CPU time	2.9e+00	5.2e-01	4.9e-01	7.9e+00	4.6e-01
mean error	2.2e-02	2.8e-02	2.8e-02	7.2e-01	3.1e-02