A Coordinate Gradient Descent Method for Linearly Constrained Smooth Optimization

Sangwoon Yun
Mathematics, National University of Singapore
Singapore

INFORMS Annual Meeting
November 5, 2007

(Joint work with Paul Tseng)

Talk Outline

- SVM (Dual) Quadratic Program
- General Problem Model
- Coordinate Gradient Descent Method
- Convergence Results
- Complexity Bound
- Index Subset Selection
- Numerical Experiance on SVM QP
- Extension
- Conclusions & Future Work

SVM (Dual) Quadratic Program

$$\begin{array}{rl} \min_x & \frac{1}{2}x^TQx - e^Tx \\ \text{subject to} & 0 \le x_i \le C, \quad i = 1, ..., n, \\ & a^Tx = 0, \end{array}$$

where $a \in \{-1, 1\}^n$, $0 < C \le \infty$, $e = [1, ..., 1]^T$, $Q \in \Re^{n\times n}$ is a sym. pos. semidef. with $Q_{ij} = a_i a_j K(z_i, z_j)$, $K : \Re^p \times \Re^p \to \Re$ (“kernel function”), and $z_i \in \Re^p$ (“ith data point”), $i = 1, ..., n$.

Popular choices of K:

- Linear kernel $K(z_i, z_j) = z_i^T z_j$
- Radial basis function kernel $K(z_i, z_j) = \exp(-\gamma\|z_i - z_j\|^2)$
- Sigmoid kernel $K(z_i, z_j) = \tanh(\gamma z_i^T z_j)$

where γ is a constant.

Q is an $n \times n$ fully dense matrix and even indefinite. ($n \ge 5000$)

Interior-point methods cannot be directly applied, except in the case of linear kernel.

Previous methods

Decomposition methods based on iterative block-coordinate descent have become popular for solving SVM QP.

- Joachims (98)
- Platt (99)
- Chang et al. (00)
- Keerthi et al. (00)
- Hush and Scovel (03)
- Palagi and Sciandrone (05)
- Fan et al. (05)

Decomposition methods use search directions of small support (i.e., few nonzeros) and achieve linear convergence under additional assumptions such as Q being positive definite.

General Problem Model

$$\begin{array}{ll} \min\limits_{x\in\Re^n} & f(x) \\ \text{s.t.} & x \in X := \{x \mid l \le x \le u,\ Ax = b\}, \end{array}$$

$f : \Re^n \to \Re$ is smooth.

$A \in \Re^{m\times n}$, $b \in \Re^m$, and $l \le u$ (possibly with $-\infty$ or ∞ components).

- For SVM QP, f is quadratic (possibly nonconvex) and $m = 1$.

Coord. Gradient Descent Method

Descent direction.

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

$$\boxed{\min_{x+d \in X,\ d_j = 0\ \forall j \notin \mathcal{J}} \{\nabla f(x)^T d + \frac{1}{2} d^T H d\}.}$$

direc. subprob

Let $d_H(x; \mathcal{J})$ and $q_H(x; \mathcal{J})$ be the opt. soln and obj. value of the direc. subprob.

Facts:

- $q_H(x; \mathcal{N}) = 0 \ \Leftrightarrow \ x \in X$ is a stationary point of f over X. stationarity
- $q_H(x; \mathcal{J}) \leq -\frac{1}{2} d^T H d$ where $d = d_H(x; \mathcal{J})$.

Choose α: Armijo rule

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

$f(x + \alpha d) - f(x) \le \sigma \alpha q_H(x; \mathcal{J}) \quad (0 < \beta < 1,\ 0 < \sigma < 1).$

For a QP, the minimization rule or the limited minimization rule can also be used.

Choose $\mathcal{J}$: Gauss-Southwell-q rule

$$q_D(x; \mathcal{J}) \le \upsilon\, q_D(x; \mathcal{N}),$$

Where $0 < \upsilon \le 1$, $D \succ 0_n$ is diagonal.

Convergence Results

Global convergence If

- $0 < \underline{\lambda} \leq \lambda_i(D), \lambda_i(H) \leq \bar{\lambda} \; \forall i$,
- $\mathcal{J}$ is chosen by Gauss-Southwell-q rule,
- α is chosen by Armijo rule,

then every cluster point of the x-sequence generated by CGD method is a stationary point of f over X.

Local convergence rate If

- $0 < \underline{\lambda} \le \lambda_i(D), \lambda_i(H) \le \bar{\lambda}\ \forall i$,
- $\mathcal{J}$ is chosen by Gauss-Southwell-q rule,
- α is chosen by Armijo rule,

in addition, if f satisfies any of the following assumptions, then the x-sequence generated by CGD method converges at R-linear rate.

C1 f is strongly convex. ∇f is Lipschitz cont. on X

C2 f is (nonconvex) quadratic. (e.g., SVM QP)

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 $\Re^m$.

C4 $f(x) = \max_{y\in Y}\{(Ex)^T y - g(y)\} + q^T x$, where $Y \subseteq \Re^m$ is polyhedral, $E \in \Re^{m\times n}$, $q \in \Re^n$, g is strongly convex, ∇g is Lipschitz cont. on $\Re^m$.

Notes:

Proof of convergence rate uses a local error bound

- Error Bound

$$\operatorname{dist}(x, X^*) \leq \kappa \|d_I(x; \mathcal{N})\|_2 \quad \text{whenever } \|d_I(x; \mathcal{N})\|_2 \leq \epsilon,$$

for some $\kappa > 0$, $\epsilon > 0$, where X^* denotes the set of stationary points of f over X and $\operatorname{dist}(x, X^*) = \min_{x^* \in X^*} \|x - x^*\|_2$.

Complexity Bound

- $0 < \underline{\lambda} \le \lambda_i(D), \lambda_i(H) \le \bar{\lambda}\ \forall i$,
- $\mathcal{J}$ is chosen by Gauss-Southwell-q rule,
- α is chosen by Armijo rule,

in addition, if f is convex with Lipschitz continuous gradient, then the number of iterations for achieving ϵ-optimality is

$$O\left(\frac{Lr^0}{\upsilon\epsilon} + \max\left\{0, \frac{L}{\upsilon}\ln\left(\frac{e^0}{r^0}\right)\right\}\right),$$

where $r^0 = \max_{x\in X}\left\{\mathrm{dist}(x, X^*)^2 \mid f(x) \le f(x^0)\right\}$, $e^0 = f(x^0) - \min_{x\in X} f(x)$, and L is a Lipschitz constant.

The constant in $O(\cdot)$ depends on $\underline{\lambda}$, $\bar{\lambda}$, σ, β.

When specialized to SVM QP, our complexity bound for achieving ϵ-optimality compares favorably with existing bounds.

Index Subset Selection

Elementary vector (Rockafellar, 1969)

- For any $d \in \Re^n$, the support of d is $\mathrm{supp}(d) := \{j \in \mathcal{N} \mid d_j \neq 0\}$.
- A d' is *conformal* to d if $\mathrm{supp}(d') \subseteq \mathrm{supp}(d)$ and $d'_j d_j \geq 0 \; \forall j \in \mathcal{N}$.
- A nonzero d is an *elementary vector* of $\mathrm{Null}(A)$ if $d \in \mathrm{Null}(A)$ and there is no nonzero $d' \in \mathrm{Null}(A)$ that is conformal to d and $\mathrm{supp}(d') \neq \mathrm{supp}(d)$.
- Each elementary vector d satisfies $|\mathrm{supp}(d)| \leq \mathrm{rank}(A) + 1$.

Find $\mathcal{J}$ with $|\mathcal{J}| = 2$ in $O(n)$ opers. (SVM QP, $m = 1$)

- Step 1: Find $d_D(x;\mathcal{N})$ in $O(n)$ opers. by solving a cont. quad. knapsack problem:

$$\begin{array}{rl} \min_d & \frac{1}{2}d^TDd + \nabla f(x)^T d \\ \text{subject to} & l \le x + d \le u, \\ & Ad = 0, \end{array}$$

  Where $D \succ 0_n$ is diagonal.

- Step 2: Find a *conformal realization* of $d_D(x;\mathcal{N})$:

  $$d_D(x;\mathcal{N}) = \sum_{i=1}^{r} d^i$$ where d^i is an elementary vector of $\mathrm{Null}(A)$

  and $r \le n - 1$.

  Choose $\mathcal{J} = \mathrm{supp}(d^{\bar{i}})$ where $\bar{i} \in \underset{i\in\{1,\ldots,r\}}{\arg\min}\ g^Td^i + \frac{1}{2}(d^i)^TDd^i$.

This finds a $\mathcal{J}$ satisfying $|\mathcal{J}| = 2$ and $q_D(x;\mathcal{J}) \le \frac{1}{n-1}q_D(x;\mathcal{N})$ in $O(n)$ opers.

Numerical Experience on SVM QP

- Implement CGD method in Fortran.

- Choose $\mathcal{J}$ by Gauss-Southwell-q rule with

$$D = \text{diag}\left[\max\{Q_{jj}, 10^{-5}\}\right]_{j=1,\ldots,n},$$

as described in previous slide.

- Our implementation of the CGD method has the form

$$x^{\text{new}} = x + d_Q(x; \mathcal{J}),$$

with $|\mathcal{J}| = 2$. This corresponds to the CGD method with α chosen by the minimization rule. (The choice of H is actually immaterial here.)

- Compute $d_D(x, \mathcal{N})$ and $q_D(x; \mathcal{N})$ by using a linear-time Fortran code `k1vfo` provided by Krzysztof Kiwiel.

- $x^0 = 0$: $O(n)$ opers. to compute gradient $Qx^0 - e$.
  (for general x^0, $O(n^2)$ opers.)

- $O(n)$ opers. per iteration to update gradient $Qx - e$ since $|\mathcal{J}| = 2$.

- The CGD method is terminated when $-q_D(x;\mathcal{N}) \le 10^{-5}$.

- Additional refinements such as caching most recently used columns of Q and using supports of 3 elementary vectors for a conformal realization of $d_D(x;\mathcal{N})$ are used to speed up the method.

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

- Comparison with LIBSVM (version 2.83), which chooses $\mathcal{J}$ differently, but with the same cardinality of 2.

Test results ($\gamma = 1/p$:default values of LIBSVM)

Data	n/p	C/kernel	LIBSVM	CGD-3pair
			iter/obj/cpu	iter/obj/cpu
a7a	16100/122	1/lin	64108/-5699.253/1.3	56869/-5699.246/6.3
		10/lin	713288/-56875.57/4.6	322000/-56873.58/32.8
		1/rbf	4109/-5899.071/1.3	4481/-5899.070/1.0
		10/rbf	10385/-55195.29/1.4	16068/-55195.30/2.0
		1/sig	3941/-6095.529/1.7	4201/-6095.529/1.2
		10/sig	9942/-57878.56/1.7	10890/-57878.57/1.8
ijcnn1	49990/22	1/lin	16404/-8590.158/3.0	20297/-8590.155/6.5
		10/lin	155333/-85441.01/4.2	155274/-85441.00/46.9
		1/rbf	5713/-8148.187/4.6	6688/-8148.187/3.8
		10/rbf	6415/-61036.54/3.5	12180/-61036.54/4.8
		1/sig	6796/-9156.916/7.0	6856/-9156.916/5.0
		10/sig	10090/-88898.40/6.4	12420/-88898.39/6.5
w7a	24692/300	1/lin	66382/-765.4115/0.4	72444/-765.4116/8.2
		10/lin	662877/-7008.306/1.1	493842/-7008.307/60.6
		1/rbf	1550/-1372.011/0.4	1783/-1372.010/0.5
		10/rbf	4139/-10422.69/0.4	4491/-10422.70/0.8
		1/sig	1477/-1427.453/0.4	2020/-1427.455/0.4
		10/sig	2853/-11668.85/0.3	5520/-11668.86/0.9

- CGD-3pair is slower than LIBSVM when the linear kernel is used, due to the greater times spent in finding $d_D(x;\mathcal{N})$ and for updating the gradient.
- CGD-3pair is comparable to LIBSVM in speed and solution quality for nonlinear kernel.

Extension

In order to find sparse solution, a nonsmooth function P is added in the objective function (e.g. $P(x) = \|x\|_1$).

Linearly Constrained Nonsmooth Optimization

$$\begin{array}{ll} \min\limits_{x \in \Re^n} & f(x) + cP(x) \\ \text{s.t.} & x \in X := \{x \mid l \le x \le u,\ Ax = b\}. \end{array}$$

$P : \Re^n \to (-\infty, \infty]$ is proper, convex, lsc, and $P(x) = \sum_{j=1}^n P_j(x_j)$ ($x = (x_1, ..., x_n)^T$).

The CGD method can be extended to solve the linearly constrained nonsmooth optimization problem.

Conclusions & Future Work

1. The CGD method is the first globally convergent block-coordinate update method for general linearly constrained optimization.

2. It is implementable in $O(n)$ opers. per iteration when f is quadratic and $m = 1$ and is suited for large scale problems with n large and m small.

3. For SVM QP, numerical results show that CGD method can be competitive with state-of-the-art SVM code on large data classification problems when a nonlinear kernel is used.

4. The CGD-3pair can be further speeded up by omitting infrequently updated components from computation ("shrinkage"), as is done in state-of-the-art SVM codes LIBSVM and SVM^{light}.

5. For large-scale applications such as ν-SVM, $m = 2$. A conformal realization can be found in $O(n \log n)$ operations when $m = 2$. However, this can still be slow. Can this be improved to $O(n)$ operations?

Thank you!

Tseng, P. and Yun S., A coordinate gradient descent method for linearly constrained smooth optimization and support vector machines training.
Tseng, P. and Yun S., A coordinate gradient descent method for constrained nonsmooth optimization and bi-level optimization.
(PDF file available at http://www.math.washington.edu/˜sangwoon/)

Support Vector Classification

- Training points : $z_i \in \Re^p$, $i = 1, ..., n$.

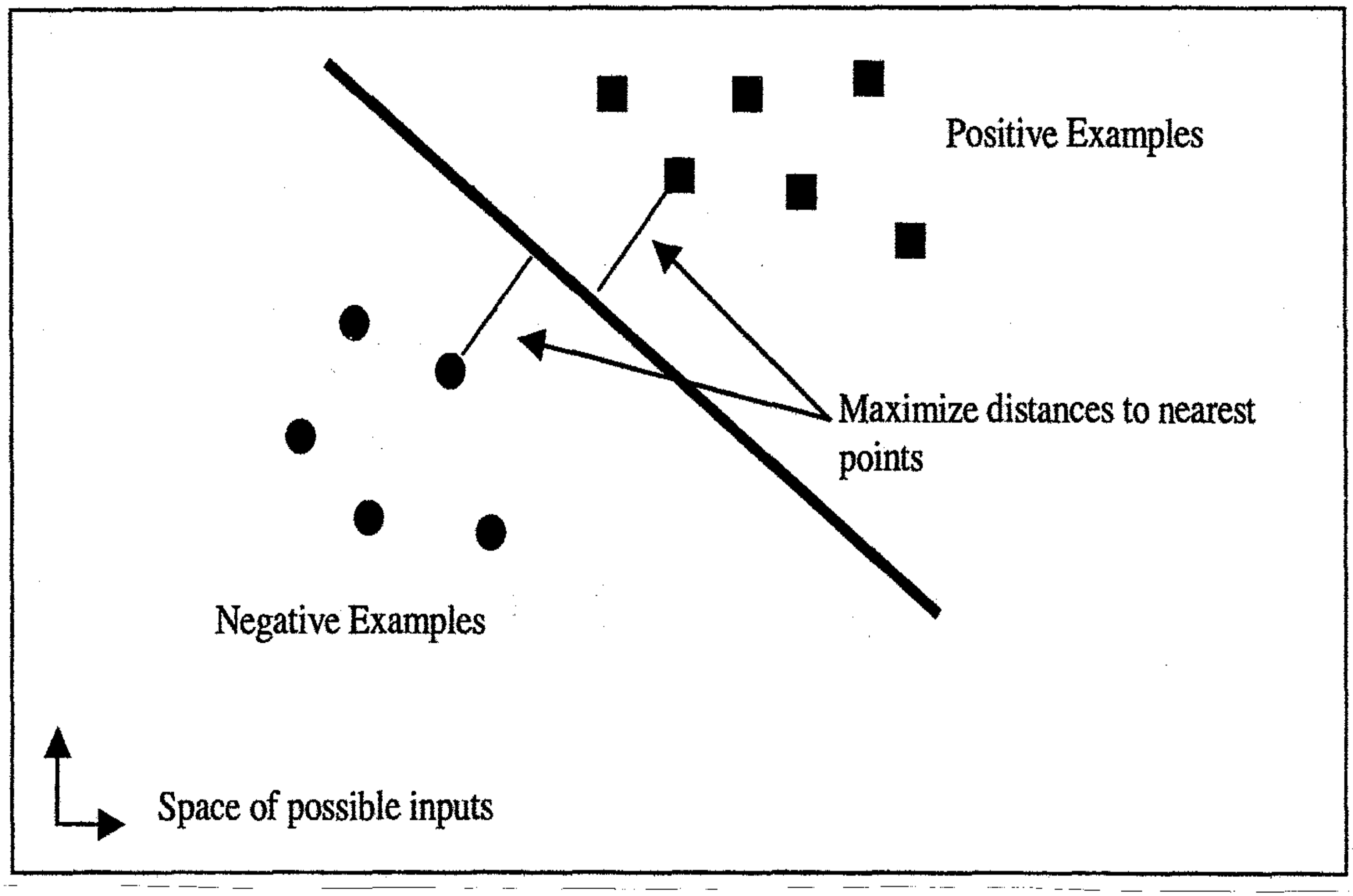
- Consider a simple case with two classes (linear separable case):

  Define a vector a :

$$a_i = \begin{cases} 1 & \text{if } z_i \text{ in class } 1 \\ -1 & \text{if } z_i \text{ in class } 2 \end{cases}$$

- A hyperplane ($0 = w^T z - b$) separates data with the maximal margin.

Margin is the distance of the hyperplane to the nearest of the positive and negative points.
Nearest points lie on the planes $\pm 1 = w^T z - b$


Positive Examples
Maximize distances to nearest points
Negative Examples
Space of possible inputs

SVM Optimization Problem

- The (original) Optimization Problem

$$\begin{array}{ll} \min\limits_{w,b} & \frac{1}{2}\|w\|_2^2 \\ \text{s.t.} & a_i \left(w^T z_i - b\right) \geq 1, \;\; i = 1, ..., n. \end{array}$$

- The Modified Optimization Problem (allows, but penalizes, the failure of a point to reach the correct margin, by Cortes and Vapnik, 1995)

$$\begin{array}{ll} \min\limits_{w,b,\xi} & \frac{1}{2}\|w\|_2^2 + C\sum\limits_{i=1}^{n} \xi_i \\ \text{s.t.} & a_i \left(w^T z_i - b\right) \geq 1 - \xi_i, \quad \xi_i \geq 0, \;\; i = 1, ..., n. \end{array}$$

Caching and Other Choices of $\mathcal{J}$

Using `k1vfo` and updating the gradient are the dominant computations.

- Cache the most recently used columns of Q, up to a user-specified limit `maxCN`, when updating the gradient $Qx - e$.

- There exists at least one elementary vector in this realization whose support $\mathcal{J}$ satisfies

$$q_D(x;\mathcal{J}) \leq \frac{1}{n-1} q_D(x;\mathcal{N}).$$

- From among all such $\mathcal{J}$, we find the best one (i.e., has the least $q_Q(x;\mathcal{J})$ value) and make this our choice for index subset.

- In addition, find from among all such $\mathcal{J}$ the second-best and third-best ones, if they exist. (In our tests, they always exist.)

- If the second-best one is disjoint from the best one, we make it the next index subset, and if the third-best one is disjoint from both the best and the second-best, we make it the second-next index subset.

- The procedure of selecting 3 (possible) pairs of the index subset is repeated at least once every 3 consecutive iterations.