CSCE 990 Lecture 8: Implementation*

Stephen D. Scott

February 28, 2006

*Most figures ©2002 MIT Press, Bernhard Schölkopf, and Alex Smola.

Introduction

- We know that the convex quadratic program representing our SVM optimization problem has a unique global optimum
- How do we efficiently find it?
- Unlike "classical" optimization problems, we sometimes must deal with extremely large, dense matrices

Outline

- Tricks of the trade: stopping criteria, restarting, caching, shrinking the training set
- Sparse greedy matrix approximation (SGMA)
- Interior point algorithms
- Subset selection methods: Chunking, working set algorithms
- Sequential minimal optimization (SMO)
- Sections 6.2.5, 6.4, 10.1–10.5, 10.7

Tricks of the Trade
Stopping Criteria

- Recall that one property of the dual is that its optimum equals the primal's optimum
- KKT-Gap is difference between primal and dual objective functions at a feasible solution
- Theorem 6.27 can bound KKT-Gap for SVMs

P10.1 Let f be feasible soln to problem of minimizing regularized risk functional $R_{\text{reg}}[f]$. Then

$$R_{\text{reg}}[f] \geq R_{\text{reg}}[f^*] \geq R_{\text{reg}}[f] - \text{Gap}[f]/(Cm)$$

where f^* is optimal feasible solution and

$$\text{Gap}[f] = \sum_{j=1}^{m} C \max\{0, 1-y_j f(x_j)\} + \alpha_j (y_j f(x_j) - 1)$$

for C-SVM and

$$\text{Gap}[f] = \sum_{j=1}^{m} \max\{0, \rho-y_j f(x_j)\} + \alpha_j (y_j f(x_j) - \rho)$$

for ν-SVM

Stopping Criteria
(cont'd)

- Can halt if gap is relatively smaller than ϵ:

$$\text{Gap}[f] \leq \epsilon \left(\frac{|R_{\text{reg}}| + |R_{\text{reg}}[f] - \text{Gap}[f]|}{2} \right)$$

  i.e. the gap is small compared to its "mid-point"

- Alternatively, can see that if f_i is solution at iteration i,

$$\min_i \{R_{\text{reg}}[f_i]\} \geq R_{\text{reg}}[f^*] \geq \max_i \{R_{\text{reg}}[f_i] - \text{Gap}[f_i]\}$$

  which is useful since Gap can increase when f_i improves

Tricks of the Trade
Restarting with Different Parameters

- Recall the C-SV classifier:

$$\begin{array}{ll} \underset{\mathbf{w}\in\mathcal{H}, b\in\mathbb{R}, \boldsymbol{\xi}\in\mathbb{R}^m}{\text{minimize}} & \tau(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i=1}^{m} \xi_i \\ \text{s.t.} & y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) \geq 1 - \xi_i, \ i = 1, \ldots, m \\ & \xi_i \geq 0, \ i = 1, \ldots, m \end{array}$$

- This can be thought of as minimizing the regularized risk functional

$$R_{\text{reg}}[f, C] := C \sum_{i=1}^{m} c(x_i, y_i, f(x_i)) + \Omega[f]$$

  where $c(\cdot)$ measures the average margin error of the training set

- How do we set C?

Restarting with Different Parameters
(cont'd)

- If f_C minimzes $R_{\text{reg}}[f, C]$, then for all $C' > C$

$$R_{\text{reg}}[f_C, C'] \geq R_{\text{reg}}[f_{C'}, C'] \geq R_{\text{reg}}[f_{C'}, C] \geq R_{\text{reg}}[f_C, C]$$

- Thus

$$\begin{aligned} R_{\text{reg}}[f_{C'}, C'] &\leq R_{\text{reg}}[f_C, C'] \\ &= C' \sum_{i=1}^{m} c(x_i, y_i, f_C(x_i)) + \Omega[f_C] \\ &= \frac{C'}{C}\left(C \sum_{i=1}^{m} c(x_i, y_i, f_C(x_i)) + \Omega[f_C] \right) \\ &= \left(\frac{C'}{C}\right) R_{\text{reg}}[f_C, C] \end{aligned}$$

Restarting with Different Parameters
(cont'd)

- Finally,

$$\left(\frac{C}{C'}\right) R_{\text{reg}}[f_{C'}, C'] \leq R_{\text{reg}}[f_C, C] \leq R_{\text{reg}}[f_{C'}, C']$$

- I.e. changes in $R_{\text{reg}}[f_C, C]$ are bounded by changes in C, so f_C is reasonable starting point for search for $f_{C'}$

- Thus can start with large C (i.e. focus on minimizing margin errors) and steadily decrease C to increase regularization

- By scaling parameters appropriately, can dramatically speed up training

Tricks of the Trade
Caching

- If kernel matrix K is too large to store in memory, may need to store most on disk, caching a relatively small amount in memory

  1. Row cache: cache m_c rows, each with m entries, and replace with LRU. Works well with e.g. SMO

  2. Element cache: store individual elements of K. Works well if most $\alpha_i = 0$, but significant overhead involved

  3. Function cache: Cheap way to update $f(x_j)$ (prediction of classifier on x_j). If e.g. first n alphas are changed in current update:

$$\begin{aligned} f^{\text{new}}(x_j) &= \sum_{i=1}^{m} \alpha_i^{\text{new}} k(x_i, x_j) + b \\ &= f^{\text{old}}(x_j) + \left[\sum_{i=1}^{n} \left(\alpha_i^{\text{new}} - \alpha_i^{\text{old}} \right) k(x_i, x_j) \right] \end{aligned}$$

Tricks of the Trade
Shrinking the Training Set

- Recall that only x_i for which $\alpha_i > 0$ affect the solution

- Thus can speed up training by dropping non-SVs from training set

- Don't want to do this too early, but at various points during optimization, can discard parts of training set

- Will cover subset selection schemes later

Sparse Greedy Matrix Approximation

- Cost of computing/storing/using for optimization entire $m \times m$ Gram matrix K is $\Theta(m^2)$

- Problematic when m (size of training set) is large, e.g. 10^5

- SGMA builds a sparse approximation $\tilde{K}$ of K

  – $\tilde{K}$ still $m \times m$, but represented by matrix $\alpha \in \mathbb{R}^{m \times n}$ for $n \ll m$

- Recall that for $x_i \in \mathcal{X}$, we can think of $k(x_i, \cdot) = k_i(\cdot)$ as a function that computes the dot product in feature space of $\Phi(x_i)$ and $\Phi(\cdot)$

  – So $K_{ij} = k_i(x_j)$

- SGMA approximates $k_i(\cdot)$ with

$$\tilde{k}_i(\cdot) := \sum_{j=1}^{n} \alpha_{ij} k(x_j, \cdot)$$

(w.l.o.g. assume that we use the first n training patterns $x_1, \ldots, x_n$ in the approximation)

Sparse Greedy Matrix Approximation
(cont'd)

- Goodness of the approximation will be the squared norm between the functions in feature space:

$$\left\| k_i(\cdot) - \tilde{k}_i(\cdot) \right\|_{\mathcal{H}}^2 = \left\langle k_i(\cdot) - \tilde{k}_i(\cdot), k_i(\cdot) - \tilde{k}_i(\cdot) \right\rangle_{\mathcal{H}}$$

- Let's hark back to Section 2.2.2, where if $f(\cdot) = \sum_{j=1}^{m} \alpha_j k(x_j, \cdot)$, then

$$\langle f, f \rangle_{\mathcal{H}} = \sum_{j,\ell=1}^{m} \alpha_j \alpha_\ell k(x_j, x_\ell)$$

- If we let $\alpha'_0 = 1$, $\alpha'_j = -\alpha_{ij}$, $x'_j = x_j$ and $x'_0 = x_i$, then

$$\begin{aligned} f(\cdot) &= k_i(\cdot) - \tilde{k}_i(\cdot) = k(x_i, \cdot) - \sum_{j=1}^{n} \alpha_{ij} k(x_j, \cdot) \\ &= \sum_{j=0}^{n} \alpha'_j k(x'_j, \cdot) \end{aligned}$$

and

$$\langle f, f \rangle_{\mathcal{H}} = K_{ii} - 2 \sum_{j=1}^{n} \alpha_{ij} K_{ij} + \sum_{j,\ell=1}^{n} \alpha_{ij} \alpha_{i\ell} K_{j\ell}$$

Sparse Greedy Matrix Approximation
(cont'd)

- Given basis functions $k_i(\cdot)$, goal is to find $\alpha \in \mathbb{R}^{m\times n}$ to minimize

$$\begin{aligned}\text{Err}(\alpha) &:= \sum_{i=1}^{m} \left\| k_i(\cdot) - \tilde{k}_i(\cdot) \right\|_{\mathcal{H}}^2 \\ &= \sum_{i=1}^{m} \left(K_{ii} - 2\sum_{j=1}^{n} \alpha_{ij} K_{ij} + \sum_{j,\ell=1}^{n} \alpha_{ij}\alpha_{i\ell} K_{j\ell} \right)\end{aligned}$$

- Differentiating wrt α and setting to 0 yields

$$\alpha_{\text{opt}} = K^{mn}\,(K^{nn})^{-1}$$

where K^{mn} is first n columns of K and K^{nn} is upper left submatrix of K

T10.2 The approximation $\tilde{K}$ of K is PD, $\text{Err}(\alpha_{\text{opt}}) = \text{tr}\,(K) - \text{tr}\left(\tilde{K}\right)$, and

$$\tilde{K} = \alpha K^{nn} \alpha^{\top}$$

- Thus given K and basis functions, can easily compute α_{opt}, $\text{Err}(\alpha_{\text{opt}})$, and $\tilde{K}$
- How do we choose basis functions?

Sparse Greedy Matrix Approximation
(cont'd)

- In general, it is intractable to choose as a basis the proper subset of the m functions that minimizes the objective function
- Alternative approach: given current set of n basis functions $k_1(\cdot), \ldots, k_n(\cdot)$, consider adding one more function
  - Random
  - Greedy: best out of all $m-n$ remaining
  - "Semi-greedy": best out of a subset of size N (e.g. $N = 59$)
- Given that $\alpha^{m,n}$ is the optimal matrix for the n chosen basis functions, and consider adding $k_{n+1}(\cdot)$. Then

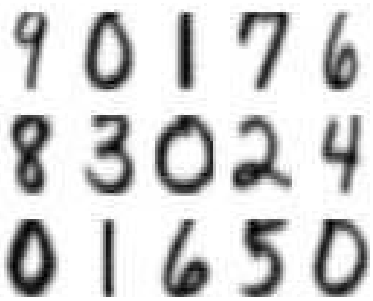
$$\text{Err}(\alpha^{m,n+1}) = \text{Err}(\alpha^{m,n}) - \eta^{-1}\|K^{mn}\mathbf{v} - \bar{\mathbf{k}}\|^2$$

where $\bar{\mathbf{k}} = (K_{1,n+1}, \ldots, K_{m,n+1})$, $\eta = (K_{n+1,n+1} - \mathbf{k}^{\top}(K^{nn})^{-1}\mathbf{k})$, $\mathbf{v} = ((K^{nn})^{-1}\mathbf{k})$, and $\mathbf{k} = (K_{n+1,1}, \ldots, K_{n+1,n})$

- Use the above to compute Err for each of the N new candidates and take the best; repeat until Err sufficiently small
- Can update α in $O(n^2)$ time

Sparse Greedy Matrix Approximation
Experiments

- Gaussian kernel, USPS data
- First ten basis functions correspond to 9 of the 10 digits



- Results (Fig. 10.3, p. 294) comparable to PCA, which requires computation of all m basis functions

Predictor-Corrector Methods

- Considers lower-order approximations when doing optimization of complex functions
- Predict update from lower-order approx, then correct it with higher-order version
- E.g. solving $f(x) = f_0 + ax + bx^2/2 = 0$:
  1. Start with $f_0 + ax = 0$, giving update of $x^{\text{pred}} = -f_0/a$
  2. Substitute x^{pred} into original equation and solve:

$$f_0 + ax^{\text{corr}} + \frac{b}{2}\left(\frac{f_0}{a}\right)^2 = 0$$

so

$$x^{\text{corr}} = -\frac{f_0}{a}\left(1 + \overbrace{\frac{bf_0}{2a^2}}^{\text{corr. term}}\right)$$

  3. Use x^{corr} as update
- Algorithm 6.5, p. 165

Interior Point Methods

- An interior point is a (x, α) pair satisfying both primal and dual constraints

- In general, start with a quadratic optimization problem:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2}x^\top Kx + c^\top x \\ \text{s.t.} & Ax + d + \xi = 0, \xi \geq 0 \end{array} \quad (1)$$

where K is $m \times m$ PD matrix, $x, c \in \mathbb{R}^m$, $A \in \mathbb{R}^{n \times m}$, and $d, \xi \in \mathbb{R}^n$

- Apply Theorem 6.26 to get KKT conditions:

$$\begin{aligned} Kx + A^\top \alpha + c &= 0 \\ Ax + d + \xi &= 0 \\ \alpha^\top \xi &= 0 \\ \alpha, \xi &\geq 0 \end{aligned}$$

- First two constraints are linear, but third is not, so we'll use predictor-corrector method

- First change third constraint to $\alpha^\top \xi = \mu > 0$ and decrease μ over time

Interior Point Methods
Linearization

- Start with initial values of x, α, ξ, and μ

- We'll compute updates Δx, etc. by expanding e.g. x to $x + \Delta x$:

$$\begin{array}{lcll} K\Delta x + A^\top \Delta\alpha & = & -Kx - A^\top \alpha - c & =: \rho_p \\ A\Delta x + \Delta\xi & = & -Ax - d - \xi & =: \rho_d \\ \alpha_i^{-1}\xi_i \Delta\alpha_i + \Delta\xi_i & = & \mu\alpha_i^{-1} - \xi_i - \alpha_i^{-1}\Delta\alpha_i\Delta\xi_i & =: \rho_{\text{KKT}_i} \end{array}$$

- Thus we get $\Delta\xi_i = \rho_{\text{KKT}_i} - \xi_i\Delta\alpha_i/\alpha_i$, and $A\Delta x - \xi\Delta\alpha/\alpha = \rho_d - \rho_{\text{KKT}}$:

$$\begin{bmatrix} K & A^\top \\ A & -D \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\alpha \end{bmatrix} = \begin{bmatrix} \rho_p \\ \rho_d - \rho_{\text{KKT}} \end{bmatrix}$$

where $D := \text{diag}(\alpha_1^{-1}\xi_1, \ldots, \alpha_n^{-1}\xi_n)$

- We've eliminated $\Delta\xi$, so we can solve for Δx^{pred} and $\Delta\alpha^{\text{pred}}$

- Now update ρ_{KKT} with $\Delta\alpha^{\text{pred}}$ (ρ_p, ρ_d are unchanged) and solve again to get Δx^{corr} and $\Delta\alpha^{\text{corr}}$, then solve for $\Delta\xi$

Interior Point Methods
(cont'd)

- Need to ensure that updates are not too large and negative to make the variables negative

- Shrink length of $(\Delta x, \Delta\alpha, \Delta\xi)$ by λ such that

$$\min \left\{ \frac{\alpha_1 + \lambda\Delta\alpha_1}{\alpha_1}, \ldots, \frac{\alpha_n + \lambda\Delta\alpha_n}{\alpha_n}, \frac{\xi_1 + \lambda\Delta\xi_1}{\xi_1}, \ldots, \frac{\xi_n + \lambda\Delta\xi_n}{\xi_n} \right\} \geq \epsilon$$

for e.g. $\epsilon = 0.05$

- To update μ, after getting new values of α and ξ, set

$$\mu = \frac{\alpha^\top \xi}{n} \left(\frac{1 - \lambda}{10 + \lambda} \right)^2$$

Interior Point Methods
Application to SVMs

- Recall the dual optimization problem for C-SVMs:

$$\begin{array}{ll} \underset{\boldsymbol{\alpha} \in \mathbb{R}^m}{\text{maximize}} & W(\boldsymbol{\alpha}) = \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j k(x_i, x_j) \\ \text{s.t.} & 0 \leq \alpha_i \leq C, \; i = 1, \ldots, m \\ & \sum_{i=1}^m \alpha_i y_i = 0 \end{array}$$

- Can put into the form of (1):

$$\begin{array}{ll} \underset{\boldsymbol{\alpha}, \mathbf{t} \in \mathbb{R}^m}{\text{minimize}} & \frac{1}{2}\alpha^\top Q\alpha + c^\top \alpha \\ \text{s.t.} & A\alpha = 0 \\ & \alpha + t = u \\ & \alpha, t \geq 0 \end{array} \quad (2)$$

where $Q_{ij} = y_i y_j k(x_i, x_j)$, $c = (-1, \ldots, -1) \in \mathbb{R}^m$, $u = (C, \ldots, C) \in \mathbb{R}^m$, $A = (y_1, \ldots, y_m)$

- Then dualize, linearize, and derive updates (Section 10.3.1)

- For large m, can use SGMA in optimization algorithm rather than all of Q

Subset Selection Methods

Working Set Algorithms

- A way of dealing with large data sets
- Focus on only a subset of the training patterns at any time, freezing the α variables for the other patterns
- Let $S_w \subset [m] = \{1, \ldots, m\}$ be working set and $S_f = [m] \setminus S_w$ be fixed set
- Now split the problem (2) into $Q = \begin{bmatrix} Q_{ww} & Q_{fw} \\ Q_{wf} & Q_{ff} \end{bmatrix}$, $c = (c_w, c_f)$, $A = (A_w, A_f)$, and $u = (u_w, u_f)$:

$$\begin{array}{ll} \underset{\boldsymbol{\alpha}_w, \mathbf{t}_w}{\text{minimize}} & \frac{1}{2}\alpha_w^\top Q_{ww}\alpha_w + \left[c_w + Q_{wf}\alpha_f\right]^\top \alpha_w \\ & + \left[\frac{1}{2}\alpha_f^\top Q_{ff}\alpha_f + c_f^\top \alpha_f\right] \\ \text{s.t.} & A_w\alpha_w = -A_f\alpha_f \\ & \alpha_w + t_w = u_w \\ & \alpha_w, t_w \geq 0 \end{array}$$

- Minimizing this also decreases (2)

Subset Selection Methods

(cont'd)

- When choosing the working set, want to base choice on what will speed up convergence
- Pick patterns whose Lagrange multipliers violate KKT conditions
- Want generally small working set (< 100), and balanced number of $+1$ and -1 labels
- In addition, can choose:
  1. those with largest contribution to KKT gap (P10.1)
  2. those with largest negative gradient of objective function at current solution

Sequential Minimal Optimization

- Extreme case of subset selection, with working set of size 2
- With only two active variables α_i and α_j, can analytically solve optimization problem

$$\begin{array}{ll} \underset{\alpha_i, \alpha_j}{\text{minimize}} & \frac{1}{2}\left[\alpha_i^2 Q_{ii} + \alpha_j^2 Q_{jj} + 2\alpha_i\alpha_j Q_{ij}\right] + c_i\alpha_i + c_j\alpha_j \\ \text{s.t.} & s\alpha_i + \alpha_j = \gamma \\ & 0 \leq \alpha_i \leq C_i, 0 \leq \alpha_j \leq C_j \end{array}$$

where $Q_{ij} = y_i y_j K_{ij}$, $s = y_i y_j$, $\gamma = y_i y_j \alpha_i^{\text{old}} + \alpha_j^{\text{old}}$, $c_i = y_i(f(x_i) - b - y_i) - \alpha_i K_{ii} - \alpha_j s K_{ij}$, C_i, C_j parameters weighting x_i, x_j

- Above values come from working set version of (2)

Sequential Minimal Optimization

(cont'd)

- Let $\xi = sc_j - c_i + \gamma s Q_{jj} - \gamma Q_{ij}$ and $\chi = Q_{ii} + Q_{jj} - 2sQ_{ij}$
- If $y_i = y_j$, let $L = \max\{0, \gamma - C_j\}$ and $H = \min\{C_i, \gamma\}$
- If $y_i \neq y_j$, let $L = \max\{0, \gamma\}$ and $H = \min\{C_i, C_j - \gamma\}$
  1. If $\chi = 0$, set $\alpha_i = L$ if $\xi > 0$ and $\alpha_i = H$ otherwise
  2. If $\chi > 0$, set $\alpha_i = \min\{\max\{L, \xi/\chi\}, H\}$
  3. Set $\alpha_j = \gamma - s\alpha_i$

Sequential Minimal Optimization
Updating b

- When choosing patterns to bring in, need to know our prediction on them, i.e. we need a current value of b
- When all αs optimal, can apply KKT conditions to sove for b; choose some $\alpha_i \in (0, C_i)$ and solve: $y_i(\langle \mathbf{w}, \Phi(x) \rangle + b) = 1$
  - But the αs aren't yet optimal!
- Thus we will estimate b by choosing the midpoint of a range of possible values

Sequential Minimal Optimization
Updating b (cont'd)

- Partition set of training patterns X into

$$I_0 = \{i \mid \alpha_i \in (0, C_i)\}$$
$$I_{+,0} = \{i \mid \alpha_i = 0, y_i = +1\}$$
$$I_{+,C} = \{i \mid \alpha_i = C_i, y_i = +1\}$$
$$I_{-,0} = \{i \mid \alpha_i = 0, y_i = -1\}$$
$$I_{-,C} = \{i \mid \alpha_i = C_i, y_i = -1\}$$

- Now define

$$e_{\text{hi}} := \min_{i \in I_0 \cup I_{+,0} \cup I_{-,C}} \{f(x_i) - y_i\}$$
$$e_{\text{lo}} := \min_{i \in I_0 \cup I_{-,0} \cup I_{+,C}} \{f(x_i) - y_i\}$$

  (f is based on setting b with a SV)
- Using KKT conditions (see Keerthi et al.), can show that optimality occurs iff $e_{\text{hi}} \geq 0 \geq e_{\text{lo}}$
- Further, using as a bias term $b_{\text{hi}} = b - e_{\text{hi}}$ for the "hi" sets and $b_{\text{lo}} = b - e_{\text{lo}}$ for the "lo" sets will yield optimality
- Thus can update b as $(b_{\text{hi}} + b_{\text{lo}})/2$
- Complete SMO pseudocode on p. 313

Topic summary due in 1 week!