E9 203 Compressed Sensing & Sparse SP

19 May 2021.

Last time:

- RIP results for subgaussian matrices
- RIP–ℓ_1 recovery connection
- JL Lemma.

Today:

- Algos for ℓ_1 regularization.
  [Source: K.P. Murphy: 'Machine learning'.]

① Coordinate descent

Optimize variables one by one, holding the others fixed.

$$x_j^* = \arg\min_z \underline{f(\underline{x} + z e_j) - f(x)}$$

e_j: j^{th} col. of I_N

$f(x)$ = cost fn. e.g. ℓ_1 regularized least squares.

Useful when the 1-d opt. pb. has an analytical soln.

(a) Round robin through coordinates

(b) Sample different coordinates at random.

Shooting algorithm: [Fu 1998, Wu & Lange 2008]

$$\underline{\underline{\hat{x}_j(c_j)}} \triangleq \begin{cases} \left(\frac{c_j+\lambda}{d_j}\right) & \text{if } c_j < -\lambda \\ 0 & \text{if } c_j \in [-\lambda, \lambda] \\ \left(\frac{c_j-\lambda}{d_j}\right) & \text{if } c_j > \lambda \end{cases}$$

$$\triangleq \underline{\text{soft}\left(\frac{c_j}{d_j} ; \frac{\lambda}{d_j}\right)}$$

where $\text{soft}(d;\delta) \triangleq \underline{\text{sign}(d)\,(|d|-\delta)_+}$

Consider

$$f(x) = \underline{\|y-Ax\|_2^2} + \underline{\underline{\lambda\|x\|_1}}$$

Local optimality condition

$$\underline{A^T(Ax-y)_j} \in \begin{cases} -\lambda & \text{if } x_j < 0 \\ [-\lambda, \lambda] & \text{if } x_j = 0 \\ \lambda & \text{if } x_j > 0. \end{cases}$$

We get $\hat{x}_j(c_j)$ above by setting

$$d_j = 2\sum_{i=1}^m \underline{\underline{A_{ij}^2}}, \quad c_j = 2\sum_{i=1}^m A_{ij}\left(y_i - \underbrace{A_{i,-j}^T}_{\text{i^{th} row of A, remove j^{th} component.}} \overset{\text{x with j^{th} component removed.}}{x_{-j}}\right)$$

Algorithm:

Init: $x = (A^TA + \lambda I)^{-1} A^T y$

Repeat

For $j = 1, 2, \ldots, N$

$$d_j = 2\sum_{i=1}^m a_{ij}^2$$

$$c_j = 2\sum_{i=1}^m a_{ij}\left(y_i - \overset{\text{i^{th} row of A}}{a_i^T} x + a_{ij} x_j\right)$$

$$x_j = \text{soft}\left(\frac{c_j}{d_j} ; \frac{\lambda}{d_j}\right)$$

Until convergence.

Problem: slow convergence.

② LARS and homotopy based methods

Active set methods: update many variables at a time.

Useful for generating solns. for a set of λ, i.e., regularization paths.

Exploit the fact that one can compute $\hat{x}(\lambda_k)$ from $\hat{x}(\lambda_{k-1})$ quickly if $\lambda_k \approx \lambda_{k-1}$ [<u>Warm start</u>]

$$\|y-Ax\|_2^2 + \underline{\underline{\lambda}\|x\|_1}$$

To find the soln. at λ_*, start at λ_{max} (large) and go down to λ_*.

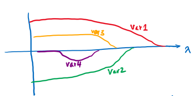
More computationally efficient than starting directly at $\lambda_* \Rightarrow$ Continuation methods
Homotopy methods.

LARS: <u>Least angle regression</u> and <u>shrinkage</u>.

Efficiently compute $\hat{x}(\lambda)$ for all possible λ!

$$\|y - Ax\|_2^2 + \lambda\|x\|_1$$

- Start with a large λ
  $\Rightarrow$ Only one variable, the for which the corresp. col. of A is most correlated w/ y, is chosen.
- <u>Decrease λ</u> till a <u>2^{nd} variable</u> becomes "active".
  That is, it has the same corr. with the <u>residual</u> as the first variable had with y.
- Interestingly, the λ at which the 2^{nd} var. becomes active can be found analytically (in closed form.)!
  So, at that λ, will have 2 active variables.
- Thus, the algo "jumps" to the next pt. in the regularization path where the active set changes.
- This process is repeated $\Rightarrow$ get soln $\hat{x}(\lambda)\ \forall \lambda$.



- Given the current active set, we solve a least-squares problem as $\lambda\downarrow$: turns out to be a <u>linear fn. of λ</u>.
  It is possible that as <u>$\lambda\downarrow$</u>, an entry in the active <u>set $\to 0$</u>. Then, it is <u>dropped</u>.
  (Otherwise, it is <u>like</u> OMP)

---

③ Proximal and gradient projection methods

Useful when homotopy methods become too slow, e.g. large scale problems.

Objective fn: $f(x) = L(x) + R(x)$

$L(x)$: loss fn, convex, differentiable

E.g. $L(x) = \frac{1}{2}\|y - Ax\|_2^2$

$R(x)$: Regularizer. Convex, not nec. differentiable.

E.g. $R(x) = \lambda \|x\|_1$.

To build intuition, let $A = I$. Then,

$$\boxed{f(x) = R(x) + \frac{1}{2}\|y - x\|_2^2}$$

Soln. $x_{opt} = \text{prox}_R(y)$

Proximal operator, R convex.

$$\text{Prox}_R(y) = \arg\min_z \left(R(z) + \frac{1}{2}\|y - z\|_2^2 \right).$$

Will use this inside an iterative algo, where we will want to stay close to the prev. iterate.

$$\boxed{\text{Prox}_R(x_k) = \arg\min_z \left(R(z) + \frac{1}{2}\|x_k - z\|_2^2 \right)}$$

Proximal gradient method

(a) How to solve;

(b) How to extend to a general A.

(a) : If $R(x) = \lambda\|x\|_1$

$\text{Prox}_R(x) = \text{soft}(x, \lambda)$

where $\text{soft}(a, \delta) \triangleq \text{sign}(a)\,(|a| - \delta)_+$

$x_+ \triangleq \max(x, 0)$

If $R(x) = \lambda\|x\|_0$,

$\text{Prox}_R(x) = \text{hard}(x, \sqrt{2\lambda})$

where $\text{hard}(u, a) = u\, \mathbb{1}_{\{|u| > a\}}$