OPTIMIZATION FOR MACHINE LEARNING

December 2, 2024

Last week of (this) class!

- Today: Last lecture (Sparsity & Coordinate descent)
- Thursday: Last lab + project presentation

Exam: December 9, 2.15-4.15?
OPEN BOOK!

SPARSE OPTIMIZATION AND COORDINATE DESCENT

→ Last two lectures: Regularized problems ⇒ Proximal methods

Sparsity-inducing regularizers ⇒ ISTA, FISTA
(≡ proximal gradient)

→ Today: Not only sparse solutions but also
- Sparse data
- Sparse updates ($x_k \to x_{k+1}$)

(1) Basics of coordinate descent methods

Problem: $\underset{x \in \mathbb{R}^d}{\text{minimize}}\ f(x)$ $\quad f: \mathbb{R}^d \to \mathbb{R}$ $\quad C^1$

$$\forall x \in \mathbb{R}^d, \quad \nabla f(x) = \begin{bmatrix} \nabla_1 f(x) \\ \vdots \\ \nabla_d f(x) \end{bmatrix} \qquad \nabla_j f(x) = \frac{\partial f}{\partial x_j}(x) \quad \forall j = 1 \dots d$$

$$= \sum_{j=1}^{d} \nabla_j f(x)\, e_j \quad , \quad e_j = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \leftarrow j \quad \in \mathbb{R}^d$$

Coordinate descent iteration

$$x_{k+1} = x_k - \alpha_k \left[\nabla_{j_k} f(x_k)\right] e_{j_k}$$

where $\alpha_k > 0$ and $j_k \in \{1, \dots, d\}$

Equivalent: $\forall j = 1..d$, $[x_{k+1}]_j = \begin{cases} [x_k]_j & \text{if } j \neq j_k \\ [x_k]_j - \alpha_k \nabla_{j_k} f(x_k) & \text{if } j = j_k \end{cases}$

$\longrightarrow$ Sparse update: Only 1 coordinate of the iterate is updated

Block coordinate descent

$$x_{k+1} = x_k - \alpha_k \sum_{j \in B_k} \nabla_j f(x_k) e_j$$

for some $B_k \subseteq \{1, \dots, d\}$

$|B_k| = 1 \Rightarrow$ Basic CD

$|B_k| = d \Rightarrow$ GD

Choosing j_k (and B_k)

- Cyclic CD: Cycle through $\{1, \dots, d\}$: $j_0 = 1, j_1 = 2, \dots, j_{d-1} = d, j_d = 1$ ...
- Randomized Cyclic CD: Every d iterations, compute a random permutation of $\{1, \dots, d\} \to \{\sigma(1), \dots, \sigma(d)\}$ and then take the coordinates $\sigma(1), \dots, \sigma(d)$ in that order
- Randomized CD: Pick j_k uniformly at random in $\{1, \dots, d\}$

Connection with stochastic gradient

↳ In general, SG and CD give different results when applied to the same problem

↳ But for certain classes of problems, SG applied to that problem is equivalent to coordinate descent (CD) applied to its dual problem

Consider

$$(P) \quad \underset{x \in \mathbb{R}^d}{\text{minimize}} \quad \frac{1}{n} \sum_{i=1}^{n} \ell_i(a_i^T x) + \lambda \Omega(x)$$

where $a_1, \dots, a_n \in \mathbb{R}^d$ are data vectors

$\ell_1, \dots, \ell_n$ are convex (loss) functions from $\mathbb{R}$ to $\mathbb{R}$

C^1, that depend on the data point

(Ex: $\ell_i(h) = \frac{1}{2}(h - y_i)^2$)

$\Omega : \mathbb{R}^d \to \mathbb{R}$ convex, C^1

Duality theory says that solving (P) is equivalently to solving

$$(D) \quad \underset{z \in \mathbb{R}^n}{\text{maximize}} \quad -\left[\frac{1}{n} \sum_{i=1}^{n} \ell_i^*(-[z]_i) + \lambda \Omega^*\left(\frac{1}{\lambda n} A^T z \right) \right]$$

where $A = \begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix} \in \mathbb{R}^{n \times d}$

and for any convex function $\phi : \mathbb{R}^m \to \mathbb{R}$, $\phi^* : \mathbb{R}^m \to \mathbb{R}$ is the Fenchel dual function defined by

$$\phi^*(z) = \sup_{x \in \mathbb{R}^n} \{ z^T x - \phi(x) \}$$

Suppose that we apply SG to (P) with random indices $i_0, i_1, \dots$ starting from x_0 $\Rightarrow$ Iterates $x_0, x_1, \dots$

Suppose that we apply CD to (D) with the same random indices than SG $(i_0, i_1, \dots)$ starting from $z_0 \in \mathbb{R}^m$ such that $x_0 = \frac{1}{\lambda_m} A^T z_0$

$\Rightarrow$ Iterates $z_0, z_1, \dots$ such that

$$\forall k \in \mathbb{N}, \quad x_k = \frac{1}{\lambda_m} A^T z_k = \frac{1}{\lambda_m} \sum_{i=1}^{m} [z_k]_j \cdot a_j$$

randomized ↑ stochastic

↳ Explains why randomized CD is sometimes called stochastic dual ascent in the literature

(D) ↑ ↑ actually a maximization problem

CD $x_{k+1} = x_k - \alpha_k \left[\nabla_{j_k} f(x_k) \right] e_{j_k}$

Coordinate Ascent: $x_{k+1} = x_k + \alpha_k \left[\nabla_{j_k} f(x_k) \right] e_{j_k}$

(2) Applications of coordinate descent

Q: When is coordinate descent interesting (more interesting than gradient descent) ?

CD Iteration: $x_{k+1} = x_k - \alpha_k \left[\nabla_{j_k} f(x_k) \right] e_{j_k}$, $j_k \in \{1, \dots, d\}$

$$\nabla_{j_k} f(x_k) = \frac{\partial f}{\partial [x]_{j_k}}(x_k)$$

⊕ Only update one coordinate of x_k to get x_{k+1}
Only need one partial derivative, possibly requiring less calculations than a full gradient...

⊖ ... but in general, the entire vector x_k is required to compute the partial derivative ⇒ cost of accessing x_k does not match the updating cost

One may also only have access to a code returning $\nabla f(x_k)$

↳ We would like to apply CD to problems for which partial derivatives can be computed without accessing all the coordinates of the input

⇒ Ideal class : Separable functions

$f : \mathbb{R}^d \to \mathbb{R}$ is separable if there exist $f_1, \dots, f_d : \mathbb{R} \to \mathbb{R}$ s.t. that

$$\forall x \in \mathbb{R}^d, \quad f(x) = \sum_{j=1}^{d} f_j([x]_j)$$

Ex) $f(x) = \|x\|^2 = x_1^2 + x_2^2 + \dots + x_d^2$

$$f_i([x]_i) = [x]_i^2$$

→ More general class: partially separable functions

$$f(x) = \sum_{j=1}^{J} f_j\left([x]_{B_j}\right)$$

where $B_j \subseteq \{1, \dots, d\}$ $\forall j = 1 \dots J$

$$[x]_{B_j} = \left[[x]_l \right]_{l \in B_j} \in \mathbb{R}^{|B_j|}$$

and $|B_j \cap B_{j'}|$ is small

$f(x) = x_1^2 e^{-x_2} + x_2^2 + x_3^2 + x_4^2 e^{-x_2}$

f : partially separable

$B_1 = \{1, 2\}$

$B_2 = \{3\}$

$B_4 = \{2, 4\}$

In ML

- Many regularizers are partially separable

$$\|x\|_1 = \sum_{j=1}^{d} |[x]_j| \quad , \quad \|x\|^2 = \sum_{j=1}^{d} [x]_j^2 \quad \Big) \text{ Separable}$$

Group LASSO: $x \mapsto \sum_{g \in G} \|x_g\|$ partially separable (depending on what the groups are)

- Partially separable objective/data-fitting terms with linear models and sparse data

$$x \mapsto \frac{1}{n} \sum_{i=1}^{n} l_i(a_i^T x)$$

a_i's are sparse vectors

Ex) Linear regression with sparse data and ℓ_2/ridge regularization

$$\underset{x \in \mathbb{R}^d}{\text{minimize}} \quad \varphi(x) = \frac{1}{2n} \|Ax - y\|^2 + \frac{\lambda}{2} \|x\|^2$$

$$\varphi(x) = \frac{1}{2n}\sum_{i=1}^{n}\left(a_i^T x - y_i\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{d}[x]_j^2$$

Sparse a_i $\Rightarrow$ $(a_i^T x - y_i)^2$ only depends on a few coordinates of x

$$\forall j \in \{1,\dots,d\} \quad \nabla_j \varphi(x) = \left[\frac{1}{n}A^T(Ax-y)\right]_j + \lambda [x]_j$$

$$= \frac{1}{n}\sum_{i=1}^{n}(a_i^T x - y_i)[a_i]_j + \lambda [x]_j$$

$[x]_j$: one coordinate of x

$$= \frac{1}{n}\sum_{i:\,[a_i]_j \neq 0}(a_i^T x - y_i)[a_i]_j$$

with sparse a_i's, $|\{i : [a_i]_j \neq 0\}| \ll n$

$\Rightarrow$

$\Rightarrow$ each $a_i^T x$ only involves a small fraction of the coordinates of x

$\longrightarrow$ Proximal subproblems with separable regularizers

(ex) $\underset{x\in\mathbb{R}^d}{\text{minimize}}\ f(x) + \lambda\|x\|_1$; $f \in C^1$

Proximal (gradient) subproblem

$$x_{k+1} = \operatorname{argmin}\left\{ f(x_k) + \nabla f(x_k)^T(x - x_k) + \frac{1}{2\alpha_k}\|x - x_k\|^2 + \lambda\|x\|_1 \right\}$$

$\nabla f(x_k)^T(x - x_k)$: linear (separable)

$\frac{1}{2\alpha_k}\|x - x_k\|^2$: separable

$\lambda\|x\|_1$: separable!

the objective is separable

$$\frac{1}{2\alpha_k}\|x\|^2 - \alpha_k x_k^T x + \frac{1}{2\alpha_k}\|x_k\|^2$$

$\frac{1}{2\alpha_k}\|x\|^2$: separable

$\frac{1}{2\alpha_k}\|x_k\|^2$: constant

→ Can apply CD to the proximal subproblem

→ Can also replace the proximal subproblem by an update over one coordinate (or a block of coordinates)

⇒ Proximal coordinate descent !

In our example,

$$x_{k+1} = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ f(x_k) + \nabla_{j_k} f(x_k)\left([x]_{j_k} - [x_k]_{j_k}\right) + \frac{1}{2\alpha_k}\left([x]_{j_k} - [x_k]_{j_k}\right)^2 + \lambda \left| [x]_{j_k} \right| \right\}$$

$j_k \in \{1, \dots, d\}$

NB: Can also do Stochastic CD, Accelerated CD (for convex functions), CD with subgradients, ...

(3) Theory of Coordinate descent

→ Our goal: . Get convergence rates for a CD approach under reasonable assumptions (strongly convex, $C^{1,1}_L$)

. Cost unit: # of iterations
# of coordinate updates

. Overall, we want to guarantee convergence of coordinate descent (asymptotically, e.g. $f(x_k) \xrightarrow[k \to \infty]{} \min_{x \in \mathbb{R}^d} f(x)$)

a) CD methods do not converge in general (Powell, 1973)

Consider $f : \mathbb{R}^3 \to \mathbb{R}$

$$x \longmapsto -\left([x]_1[x]_2 + [x]_2[x]_3 + [x]_3[x]_1\right) + \sum_{j=1}^{3}\left(|[x]_j| - 1\right)_+$$

$(t)_+ = \max(t, 0)$

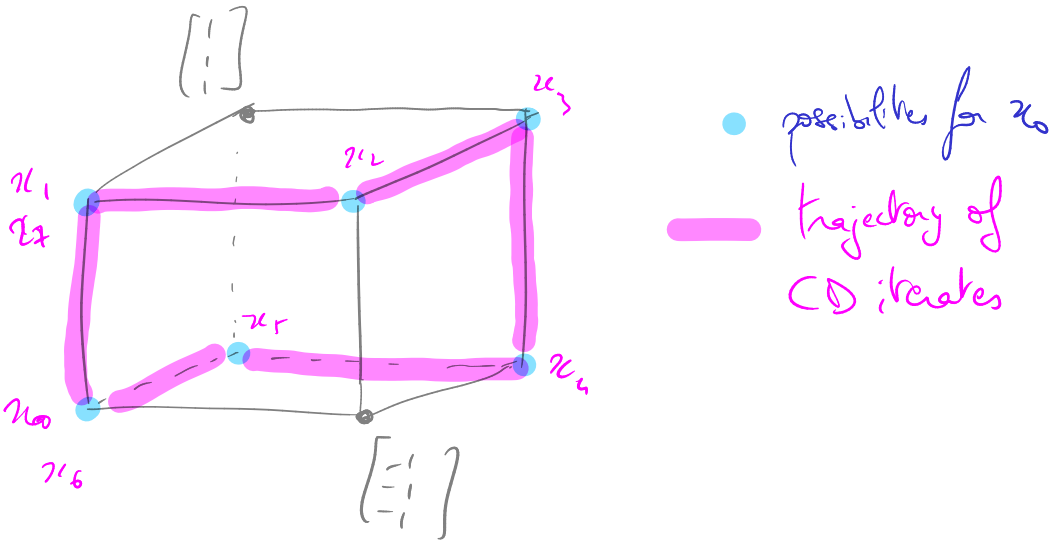
f C¹, nonconvex

$$\operatorname*{argmin}_{x \in \mathbb{R}^3} f(x) = \left\{ \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \right\}$$

→ Cyclic CD $(j_0 = 1, j_1 = ?, j_2 = 3, j_3 = 1, \dots)$

with $x_0 \in \left\{ \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix} \mid \varepsilon_i \in \{-1, 1\} \right\} \setminus \left\{ \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \right\}$

Visualization



→ Assume that $\alpha_k = \operatorname*{argmin}_{\alpha > 0} f\left(x_k - \alpha \left[\nabla_{j_k} f(x_k)\right] e_{j_k}\right)$ "Exact line search"

→ Show the limitations of (cyclic) CD, and it explains why

these methods were discarded in the 1970s-1980s and then became trendy again for large-scale calculations

$\Rightarrow$ Randomized CD has been the most successful approach

b) Convergence rates for randomized CD and randomized cyclic CD

Setup: $\underset{x \in \mathbb{R}^d}{\text{minimize}}\ f(x)$, f μ-strongly convex

$C^{1,1}_L$ (∇f L-Lipschitz)

- $\nabla_j f$ is L_j-Lipschitz continuous for every $j = 1..d$.

$\Rightarrow$ If $L_{max} = \max_{1 \le j \le d} L_j$, $\quad L_{max} \le L \le d L_{max}$

Th) Convergence rate for randomized coordinate descent.

Suppose that we apply CD with j_k drawn uniformly at random $\forall k$ and $\alpha_k = \frac{1}{L_{j_k}}$ $\forall k$. (similar results for $\alpha_k = \frac{1}{L_{max}}$)

Then, $\forall K \in \mathbb{N}$,

$$\mathbb{E}\left[f(x_K) - \min_{x \in \mathbb{R}^d} f(x)\right] \le \left(1 - \frac{\mu}{d L_{max}}\right)^K \left(f(x_0) - \min_{x \in \mathbb{R}^d} f(x)\right)$$

$\mathbb{E}$ over $j_0, \dots, j_{K-1}$

→ For GD with stepsize $\frac{1}{L}$, $f(x_k) - \min_{x \in \mathbb{R}^d} f(x) \leq \left(1 - \frac{\mu}{L}\right)^k \left(f(x_0) - \min_{x \in \mathbb{R}^d} f(x)\right)$

$L \leq dL_{max} \Rightarrow -\frac{1}{L} \leq -\frac{1}{dL_{max}} \Rightarrow \left(1 - \frac{\mu}{L}\right)^k \leq \left(1 - \frac{1}{dL_{max}}\right)^k$

⇒ In terms of rate of convergence, GD is better

⇒ But an iteration of GD updates d coordinates

while ——— CD updates 1 coordinate

Using a budget of C coordinate updates, we can perform

C iterations of CD ⇒ rate $\left(1 - \frac{\mu}{dL_{max}}\right)^C$

but only C/d ——— GD ⇒ rate $\left(1 - \frac{\mu}{L}\right)^{C/d}$

Interpretation:

. Like stochastic gradient, comparing CD with GD in terms of iterations is not necessarily fair, especially if the cost of | updating coordinates / computing partial derivatives | is the dominant cost

⇒ More favorable comparison in terms of coordinate updates, but no clear winner (because of the way the rates depend on d) in general

↳ Specialized results for partially separable functions

. Unlike SG, randomized CD has a linear rate of convergence ($O(t^k)$) on strongly convex functions, like GD.

⇒ Every step is taken along a direction of decrease for the function

Th] Randomized cyclic coordinate descent:

After K iterations of the method with $\alpha_k = \frac{1}{L_{max}}$)

$$\mathbb{E}\left[f(x_K) - \min_{x \in \mathbb{R}^d} f(x)\right] \leq \left(1 - \frac{\mu}{2L_{max}\left(1 + \frac{dL^2}{L_{max}^2}\right)}\right)^{K/d} \left(f(x_0) - \min_{x \in \mathbb{R}^d} f(x)\right)$$

→ Worse rate than randomized CD ($O(\epsilon^{K/d})$ vs $O(\epsilon^K)$)

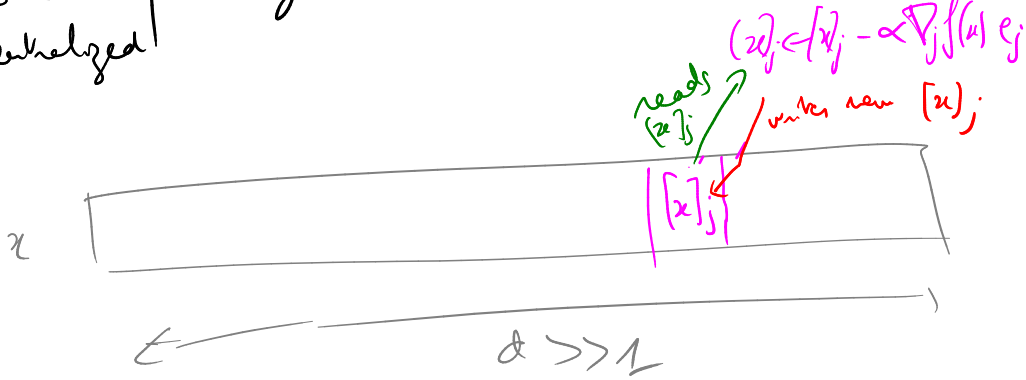
→ Can improve with (partial) separability

(4) CD in distributed settings

Main motivation for using CD: Run iterations of (block) CD in parallel

a) Synchronous CD

Setup: Shared (centralized) memory + read/write access



Iteration k

- Distribute (blocks of) coordinates to processors
- Each processor performs an update
- Synchronization: update the iterate in memory

⊕ Works well for separable problems

⊖ Synchronization requires to wait for all processors

b) Asynchronous CD

↳ Processors read and write asynchronously

Algorithm

Shared: iterate x, counter $k=0$.

Repeat (for all processors)

- Draw $j_k \in \{1, \dots, d\}$
- Update: $x_{k+1} = x_k - \alpha_k \nabla_{j_k} f(\hat{x}_k) e_{j_k}$

  ($\hat{x}_k$: value of x at the time it was read)
- $k = k+1$

→ No convergence rate for that method under simple assumptions but asymptotic convergence can be guaranteed!

→ Under additional assumptions, can get CV rates

⇒ HOGWILD! (NeurIPS test-of-time-award in 2020)

Ré, Recht, Wright, Liu, ...

⇒ Asynchronous CD on a dual problem

used to define ↳ Asynchronous SG method

Takeaway:

- CD: good for (partially) separable problems
- Theory, but mostly practical concerns about $d \gg 1$
- Works in parallel

Exercise: $\underset{x \in \mathbb{R}^d}{\text{minimize}} \; f(x) + \frac{1}{2}\|Lx\|^2$ $\quad f \in C^1$

$$L = \begin{bmatrix} -2 & 1 & 0 & & \\ 1 & \ddots & \ddots & \ddots & \\ 0 & \ddots & \ddots & \ddots & 1 \\ & & \ddots & 1 & -2 \end{bmatrix} \qquad L_{ij} = \begin{cases} 1 & \text{if } i = j+1 \text{ or } i = j-1 \\ -2 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

a) Prox gradient iteration

$$x_{k+1} = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2\alpha_k}\|x - x_k\|^2 + \frac{1}{2}\|Lx\|^2 \right\}$$

$$= \operatorname{prox}_{\alpha_k \frac{1}{2}\|L\cdot\|^2}\left(x_k - \alpha_k \nabla f(x_k) \right)$$

b) Is it easy to compute?

Yes if solving strongly convex quadratic problems is easy

c) Special case: $f(x) = \frac{1}{2n}\|Ax - y\|^2$ with $A = I_d \in \mathbb{R}^{d \times d}$

Then $\nabla f(x_k) = \frac{1}{n} A^T (A x_k - y) = \frac{1}{d}(x_k - y)$

Proximal subproblem

$$\underset{x \in \mathbb{R}^d}{\text{minimize}} \left\{ f(x_k) + \frac{1}{d}(x_k - y)^T (x - x_k) + \frac{1}{2\alpha_k}\|x - x_k\|^2 + \frac{1}{2}\|Lx\|^2 \right\}$$

d) $d = 0 \Rightarrow$ Problem separable

Write down the iteration of proximal CD in that case

$$x_{k+1} = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \Big\{ f(x_k) + \frac{1}{d}\big[x_k - y\big]_{j_k}\big[x - x_k\big]_{j_k} + \frac{1}{2\alpha_k}\big[x\big]_{j_k}^2 - \frac{1}{\alpha_k}\big[x\big]_{j_k}\big[x_k\big]_{j_k} + \frac{1}{2\alpha_k}\big[x_k\big]_{j_k}^2 \Big\}$$

CD on that problem

$$x_{k+1} = x_k - \alpha_k \nabla_{j_k} f(x_k)\, e_j$$

$$= x_k - \frac{\alpha_k}{d}\big[x - x_k\big]_{j_k} e_j$$

e) When $\lambda > 0$, justify that the proximal subproblem is partially separable and propose a way to select coordinate(s) for (block) proximal CD.

$$\|Lx\|^2 = x^T \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}^2 x$$

$$= \left\| \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix} x \right\|^2, \quad Lx = \begin{bmatrix} 2[x]_1 - [x]_2 \\ -[x]_1 + 2[x]_2 - [x]_3 \\ \vdots \\ -[x]_{d-2} + 2[x]_{d-1} - [x]_d \\ -[x]_{d-1} + 2[x]_d \end{bmatrix}$$