# IE510 Applied Nonlinear Programming

# Lecture 7: Coordinate Descent Methods

Ruoyu Sun

Mar 1 and 6, 2018. Week 7 Thur and Week 8 Tues.

# Questions for Lecture 6 on Eigenvalues

- **Q1**: For what kind of algorithms/problems can you analyze its behavior by “eigenvalues”? What eigenvalues?

- **Q2**: Suppose you computed eigenvalues $\lambda_1, \ldots, \lambda_n$ of the update matrix $M$. What can you say about the algorithm?

- **Q3**: Your colleague is tuning parameters for an algorithm. Could you provide some tips? (lots of answers)

# Questions for Lecture 6 on Eigenvalues

- **Q1**: For what kind of algorithms/problems can you analyze its behavior by "eigenvalues"? What eigenvalues?

- **Q2**: Suppose you computed eigenvalues $\lambda_1, \ldots, \lambda_n$ of the update matrix $M$. What can you say about the algorithm?

  $\max\{|\lambda_i|\} > 1$ : diverge; $\max\{|\lambda_i| < 1$, converge.

  $\lambda_i$'s complex $\Rightarrow$ oscillation

  Multi-stage : $\lambda_i$'s.

- **Q3**: Your colleague is tuning parameters for an algorithm. Could you provide some tips? (lots of answers)

  - Start from small scale problems, and understand it.
  - Check eigenvalues of Hessian; see the relation between Hessian and parameters: when parameters change, how do the eigenvalues change?

# Review of Lecture 6 on Eigenvalues

- Eigenvalues of what?
  - Quadratic case: update matrix in a matrix recursion form
  - Non-quadratic case + GD: Hessian (as approximation)

  + other algorithms: fixed-point equations + approximate.
- Effects? Suppose $\mu_1 \leq \cdots \leq \mu_n < 1$ in eig$(M)$:
  - Worst-case rate $\mu_n$, i.e., converge faster than the sequence $\{\mu_n^r\}$
  - Multiple stages; slope $\sim \frac{1}{\mu_k}$, length $\sim \frac{1}{1-\mu_k}$ (see Lecture 6 slides)
  - Complex $\mu_k$ results in fluctuation in stage $k$
- GD and HB: multi-stage. Various shapes of plots – determined by eigenvalues of $M$
- Practical issues

  $\frac{1}{1-0.9}, \frac{1}{1-0.92}, \frac{1}{1-0.95}, \cdots$

  $0.9, 0.92, 0.95 \cdots$
  - Sensitivity and "feature" of parameter (e.g. tune $1/(1-\beta)$, not $\beta$)
  - Guess the optimal parameter of HB via simulation observation: straight $\&$ all-time fluctuation

# High-Level Lessons

- Belief: something can explain the world (algorithm)
  - Eigenvalues are powerful tools, but not always work
  - Something can! (drive of modern science)

AI is not Alchemy
Theory is far behind practice

- LeCun: "scientific methodology"
  - How to understand the world: observe and explain
  - Reductionism: Lecture 4 and 6. (v.s. Holism)

Logistic regression < 1) 1-dim, 2) n-dim

- Future ~~works~~ workers: scientific training is increasingly important (v.s. "skills")
  - Why? AI. side: GDP spectrum

# Announcement: Project Proposal

- **Tentative topic**: $\geq$ 1 paragraph ($\geq$ 10 lines), by Mar 10 Sat. 11:59pm. Email to me and TA.
  - What is your tentative topic: what kind of task, preliminary thought
- $\geq$ **1-page proposal** due: Apr 1 11:59pm.
  - Describe problem/motivation and what you've done.
  - Do some work before submitting it. Allow you to test ideas/change plan.
  - More details later on website.
- Final report due: May 14 11:59pm
- What should you do? One simple way: pick any algorithm (SVRG, BFGS, Adam, etc.), analyze its behavior in
  - linear regression, logistic regression, other convex problems,
  - matrix factorization, 2-layer neural-nets, NMF and other nonconvex problems

# Today

- We've discussed GD and two types of momentum.
- Coming weeks: another big class of methods
  - Open the door to large scale optimization
- Today: Coordinate Descent Methods basics
- After today's course, you will be able to
  - define coordinate descent
  - identify problems that are suitable for CD
  - explain the benefit of CD
- More advanced goal: understand "first principle" of algorithm design

# Outline

# Motivation: Minimize $(xy-1)^2$

- Recall homework 2 problem 4: solve

$$\min_{x,y}(xy-1)^2.$$

$(x-1)^2$.

- It's not easy to apply existing theory of GD to it!
- Observation: nonconvex in $(x,y)$, but convex in $x$ only (fixed $y$), or $y$ only (fixed $x$).
- CD: fixed $y$, update $x$ by $x \leftarrow \arg\min_x (xy-1)^2$
  fix $x$, update $y$ by $y \leftarrow \arg\min_y (xy-1)^2$
- **Advantage**: no stepsize tuning (if close-form solution exists).

# Coordinate Descent

- Consider a general problem

$$\min_{x_i \in \mathcal{X}_i} f(x_1, x_2, \ldots, x_n)$$

- The function may be complicated (not necessarily smooth). Maybe one at a time?

- **Coordinate descent** (CD) (computer form):
  For $i = 1, 2, \ldots, n, 1, 2, \ldots, n, \ldots,$

$$x_i \leftarrow \operatorname*{argmin}_{x_i \in \mathcal{X}_i} f(x_1, x_2, \ldots, x_n), \tag{1}$$

  where in this update $x_{-i} \triangleq (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ is fixed.

- Example: $n = 2$, alternating minimization
  $x_1^+ \leftarrow \operatorname{argmin}_{x_1} f(x_1, x_2), \quad x_2^+ \leftarrow \operatorname{argmin}_{x_2} f(x_1^+, x_2)$

# Coordinate Descent: Math Form

- If we denote each iterate by a distinct notation $x_k^r$, we have a more mathematical form.

  3 forms: $x \leftarrow h(x), \quad x^+ = h(x), \quad x^{r+1} = h(x^r).$

- **Coordinate descent** (CD) (math form):

  Initial point $x^0 = (x_1^0, x_2^0, \ldots, x_n^0)$.

  For $r = 0, 1, 2, \ldots$, update $x_1, \ldots, x_n$ sequentially as:

$$x_1^{r+1} \in \operatorname*{argmin}_{x_1 \in \mathcal{X}_1} f(x_1, x_2^r, \ldots, x_n^r),$$

$$x_k^{r+1} \in \operatorname*{argmin}_{x_k \in \mathcal{X}_k} f(x_1^{r+1}, \ldots, x_{k-1}^{r+1}, x_k, x_{k+1}^r, \ldots, x_n^r), \tag{2}$$

$$x_n^{r+1} \in \operatorname*{argmin}_{x_n \in \mathcal{X}_n} f(x_1^{r+1}, \ldots, x_{n-1}^{r+1}, , x_n^r).$$

# CD and GD: Difference and Relation

- Why is a problem $x_1^2 x_3 + \sin(x_2 x_1) + \cdots + x_1 e^{x_n x_2}$ difficult?
  - Complicated form
  - Too many variables
- What problems are (relatively) easy?
  - Quadratic problems: simple in function form
  - 1-dim problems: simple in dimension
- GD: approximate by quadratic, iteratively
- CD: solve 1-dim problem, iteratively

# CD and GD: Difference and Relation

- Why is a problem $x_1^2 x_3 + \sin(x_2 x_1) + \cdots + x_1 e^{x_n x_2}$ difficult?
  - Complicated form
  - Too many variables
- What problems are (relatively) easy?
  - Quadratic problems: simple in function form
  - 1-dim problems: simple in dimension
- GD: approximate by quadratic, iteratively
- CD: solve 1-dim problem, iteratively

# Variants of CD

- What if the subproblem has no closed form solution?

  One choice: Use whatever algorithms to solve the subproblem! Say, 50 iteration of GD.

- **Variant 1**: **CGD** (coordinate gradient descent) hw2 p1
  Apply one gradient update of $x_i$

$$x_i \leftarrow x_i + \alpha \nabla_i f(x).$$

  How to pick stepsize?

- What if I want to update a chunk of coordinates?
- **Variant 2**: **BCGD** (block coordinate gradient descent).
  - Split $x \in \mathbb{R}^N$ into $x_1, \ldots, x_n$, where $x_i$ is $d_i$-dim vector.
  - Update each block $x_i$ by $\partial f / \partial x_i$.

# Variants of CD

- What if the subproblem has no closed form solution?

  One choice: Use whatever algorithms to solve the subproblem! Say, 50 iteration of GD.

- **Variant 1**: **CGD** (coordinate gradient descent) hw2 p1
  Apply one gradient update of $x_i$

$$x_i \leftarrow x_i + \alpha \nabla_i f(x).$$

  How to pick stepsize?

- What if I want to update a chunk of coordinates?
- **Variant 2**: **BCGD** (block coordinate gradient descent).
  - Split $x \in \mathbb{R}^N$ into $x_1, \ldots, x_n$, where $x_i$ is $d_i$-dim vector.
  - Update each block $x_i$ by $\partial f / \partial x_i$.

# Variants of CD (cont'd)

e.g. $n=3$, (123), (123), (123), ...

- Do we have to update $x_1, \ldots, x_n$ sequentially?
- **Variant 3**: different update order.
  - Double sweep:
    $(1, 2, \ldots, n-1, n, n-1, \ldots, 1), (1, 2, \ldots, n-1, n, n-1, \ldots, 1)$
  - Randomly permuted (shuffled):
    - pick a random permutation $\sigma$,
    - then at round $r$, use order $\sigma(1), \ldots, \sigma(n)$
  - Randomized: pick $i$ indep. randomly from $(1, 2, \ldots, n)$
  - Greedy: pick the "best" coordinate/block
    - Gauss Southwell: pick $i$ that maximize $\|\nabla_i f(x)\|$
    - MBI (Maximum Block Improvement): pick $i$ that minimize $f(x)$, or $i = \operatorname{argmin} f(x_i, x^r_{-i})$.
  - Essentially cyclic: every $i$ is updated at least once in $K$ iterations

# Variants of CD (cont'd)

- Do we have to update $x_1, \ldots, x_n$ sequentially?
- **Variant 3**: different update order.
  - Double sweep: $n=3$: (1 2 3 2 1), (1 2 3 2 1), - -
    $(1, 2, \ldots, n-1, n, n-1, \ldots, 1), (1, 2, \ldots, n-1, n, n-1, \ldots, 1)$
  - Randomly permuted (shuffled): $n=3$: (1 2 3), (3 1 2), (3 2 1), - -
    - pick a random permutation $\sigma$,
    - then at round $r$, use order $\sigma(1), \ldots, \sigma(n)$

    $n=3$: (1 3 1 2 3 3 1 2 2 2 3 - -)
  - Randomized: pick $i$ indep. randomly from $(1, 2, \ldots, n)$
  - Greedy: pick the "best" coordinate/block
    - Gauss Southwell: pick $i$ that maximize $\|\nabla_i f(x)\|$
    - MBI (Maximum Block Improvement): pick $i$ that minimize $f(x)$, or $i = \operatorname{argmin} f(x_i, x^r_{-i})$.
  - Essentially cyclic: every $i$ is updated at least once in $K$ iterations

Stopping: $\|x^r - x^{r-1}\|$. $\|f(x^r) - f(x^{r-1})\|$

# Outline

# Example 1a: Linear Regression

- Consider solving

$$\min_{x_1,x_2,\ldots,x_d} \frac{1}{2}\|Ax-b\|^2 = \frac{1}{2}\|A_1x_1+\cdots+A_nx_n-b\|^2.$$

n×d d×1

- GD: each iteration computes $A^T(Ax-b)$
  - takes time O(nd)
  - requires storage O(nd)

If $n \approx 100, d \approx 100$, fine; what if $n,d \approx 10^6$?

- Try to save time/storage. Update one coordinate $x_i$

$$0 = \nabla_i f(x) = A_i^T(Ax-b) = A_i^T(A_ix_i + A_{-i}x_{-i} - b),$$

so the update of $x_i$ is

$$x_i = \frac{A_i^T(b - A_{-i}x_{-i})}{A_i^TA_i}.$$

- Time _____, storage _____

# Example 1a: Linear Regression

- Consider solving

$A = [A_1, A_2, \cdots, A_n] \in \mathbb{R}^{d \times n}$.

$$\min_{x_1, x_2, \ldots, x_d} \frac{1}{2}\|Ax - b\|^2 = \frac{1}{2}\|A_1 x_1 + \cdots + A_n x_n - b\|^2. \quad (1)$$

- GD: each iteration computes $A^T(Ax - b)$
  - takes time $O(nd)$
  - requires storage $O(nd)$.

- Try to save time/storage. Update one coordinate $x_i$ $\min f(x_i, \cdots)$

$$0 = \nabla_i f(x) = A_i^T(Ax - b) = A_i^T(A_i x_i + A_{-i} x_{-i} - b),$$

desired var, fixed.

so the update of $x_i$ is

$$x_i = \frac{A_i^T(b - A_{-i}x_{-i})}{A_i^T A_i}.$$

$A_{-i} x_{-i} = A_1 x_1 + \cdots + A_{i-1} x_{i-1} + A_{i+1} x_{i+1} + \cdots + A_n x_n$

Let $b=0$, $n=3$. $A_i^T A_i = \|A_i\|^2 = 1$, $i=1$; $A_1 \in \mathbb{R}^{d \times 1}$

$x_1 = -A_1^T (A_2 x_2 + A_3 x_3)$. ($d \times 1$, $1 \times 1$)

$2 \cdot 2d = 4d$ ($d$, $d$)

$n^3$ $^2d$

- Time $O(nd)$, storage $O(nd)$.

# Linear Regression: A Clever Trick

- There seems to be no benefit? How to update faster?
- Clever trick: store $r = b - Ax$.

$$x_i^+ = \frac{A_i^T(b - A_{-i}x_{-i})}{A_i^T A_i} = \frac{A_i^T r}{\|A_i\|^2} + x_i. \tag{3}$$

- Update procedure and time:

$$x_i^+ = \frac{A_i^T r}{A_i^T A_i} + x_i, \tag{4a}$$

$$r^+ \leftarrow \tag{4b}$$

- Each iteration takes time _____; one round takes time _____. Storage ____

Qi's proposal: store $r_i = b - A_{-i} x_{-i}$. Issue: Need to store $r_1, r_2, \ldots, r_n$.

# Linear Regression: A Clever Trick

- There seems to be no benefit? How to update faster?
- Clever trick: store $r = b - Ax$.

Compute r from scratch takes time O(nd). However, the trick is to "update" r, not "compute" r at each iteration.

$$x_i^+ = \frac{A_i^T (b - A_{-i} x_{-i})}{A_i^T A_i} = \frac{A_i^T r}{\|A_i\|^2} + x_i. \tag{3}$$

- Update procedure and time:

$$x_i^+ = \frac{A_i^T r}{A_i^T A_i} + x_i, \tag{4a}$$

$$r^+ \leftarrow r + A_i (x_i - x_i^+). \tag{4b}$$

O(d) (time of (4a)); O(d) (time of (4b))

$r^+ = b - A_1 x_1 - \cdots - A_i x_i^+ - A_{i+1} x_{i+1} - \cdots$
$= b - A_{-i} x_{-i} - A_i x_i^+$
$= b - A_{-i} x_{-i} - A_i x_i + (A_i x_i - A_i x_i^+)$
$= r + (A_i x_i - A_i x_i^+)$

- Each iteration takes time O(d); one round takes time O(nd). Storage O(d).

# Linear Regression: Comparison of GD and CD

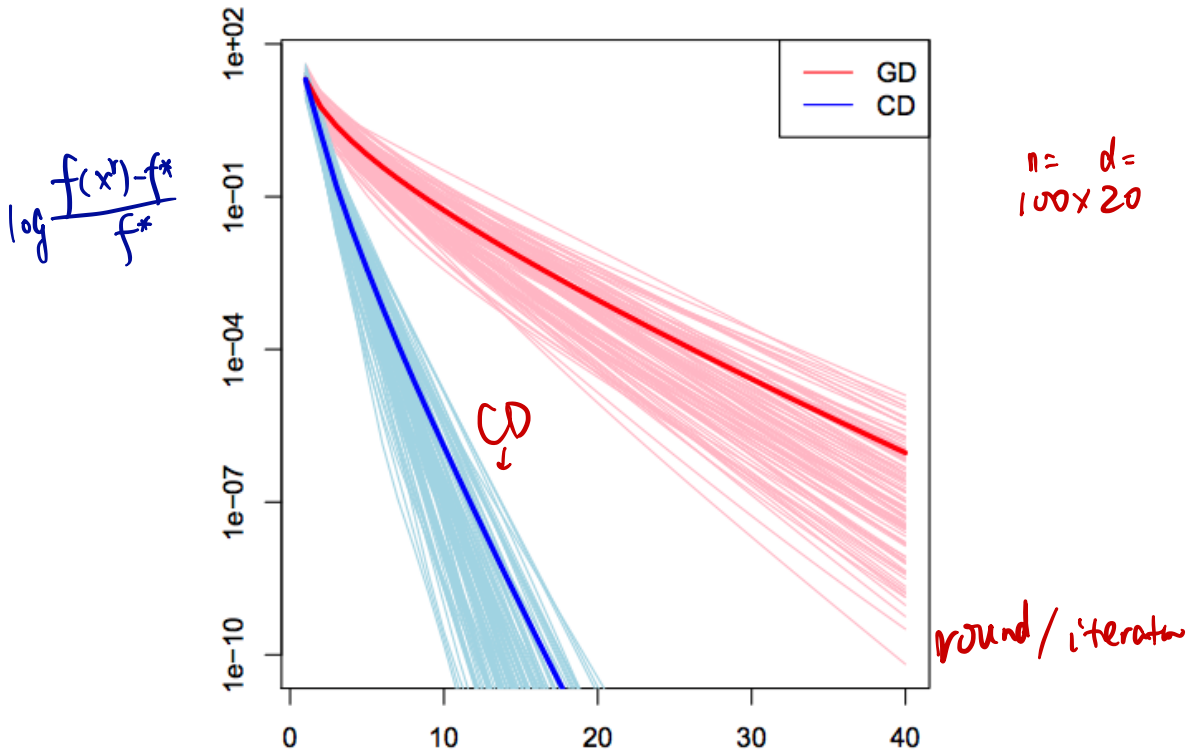


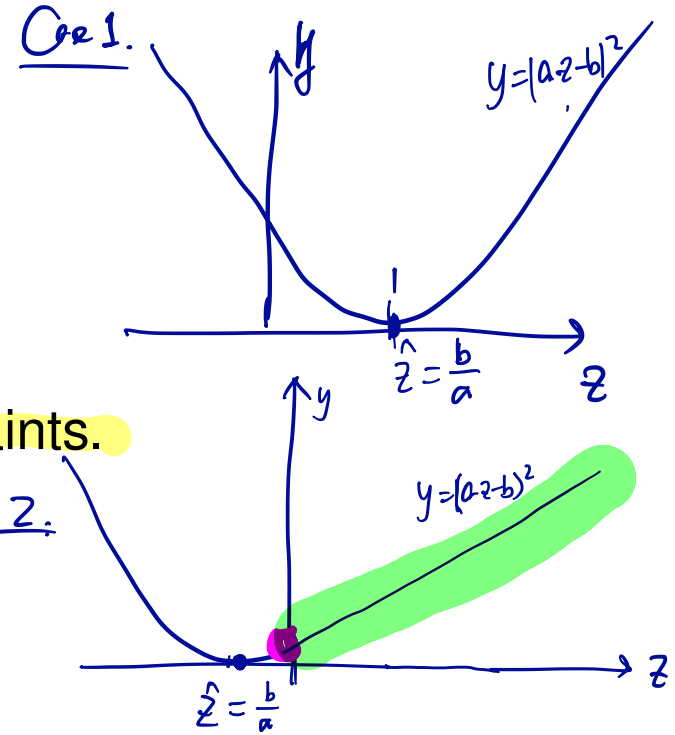

Figure: from Gordan, Tibshirani lecture
https://www.cs.cmu.edu/ ggordon/10725-F12/slides/25-coord-desc.pdf

# Example 1b: Nonnegative Least Squares

- Problem $\min_{x \in \mathbb{R}^n, x \geq 0} \|Ax - b\|^2$. $= f(x)$
- Application: radiotherapy [Zarepisheh, Lei, Ye'2018], linear programming
- CD: $x_i \leftarrow \arg\min_{x_i} f(x_i, x_{-i})$.

  subproblem is $\min_{z \in \mathbb{R}} (az - b)^2$ s.t. $z \geq 0$.

  $z^* = \max\{\frac{b}{a}, 0\}$.
- **Remark**: CD can handle simple constraints.




End of Mar. 1 lecture.

Mar. 6, Week 8 Tues.

Review:

1) What are the variants of CD?

or What're choices when you implement CD?

Choice1: Update order

Choice2: Block size

Choice3: Subproblem solver (exact minimization. one gradient step; five Newton steps; etc.)
- stepsize choice
- momentum or not

How to remember the three?

Naive CD: for cyclical order $1, 2, \cdots, n$, pick coordinate $i$, update $x_i \leftarrow \underset{x_i}{\operatorname{argmin}} f(x_i, x_{-i})$

cyclical order: Choice 1; coordinate $i$: Choice 2; $\underset{x_i}{\operatorname{argmin}}$: Choice 3

2) What is the trick when applying CD to linear regression?

- Store residual.

# Example 1c: LASSO and Compressive Sensing

Probably the hottest topic between 2004-2012 in EE/Stats/Applied math/optimization.

[PDF] **Regression shrinkage and selection via the lasso**
R Tibshirani - Journal of the Royal Statistical Society. Series B …, 1996 - JSTOR
We propose a new method for estimation in linear models. Thelasso'minimizes the residual sum of squares subject to the sum of the absolute value of the coefficients being less than a constant. Because of the nature of this constraint it tends to produce some coefficients that are exactly 0 and hence gives interpretable models. Our simulation studies suggest that the lasso enjoys some of the favourable properties of both subset selection and ridge regression. It produces interpretable models like subset selection and exhibits the stability of
☆ 99 Cited by 23012 Related articles All 65 versions »

**Atomic decomposition by basis pursuit**
SS Chen, DL **Donoho**, MA Saunders - SIAM review, 2001 - SIAM
The time-frequency and time-scale communities have recently developed a large number of overcomplete waveform dictionaries---stationary wavelets, wavelet packets, cosine packets, chirplets, and warplets, to name a few. Decomposition into overcomplete systems is not
☆ 99 Cited by 10586 Related articles All 46 versions »

**Compressed sensing**
DL **Donoho** - IEEE Transactions on information theory, 2006 - ieeexplore.ieee.org
Suppose x is an unknown vector in Ropf m (a digital image or signal); we plan to measure n general linear functionals of x and then reconstruct. If x is known to be compressible by transform coding with a known transform, and we reconstruct via the nonlinear procedure
☆ 99 Cited by 20721 Related articles All 31 versions »

**Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information**
EJ Candès, J Romberg, T Tao - IEEE Transactions on …, 2006 - ieeexplore.ieee.org
This paper considers the model problem of reconstructing an object from incomplete frequency samples. Consider a discrete-time signal f/spl isin/C/sup N/and a randomly chosen set of frequencies/spl Omega/. Is it possible to reconstruct f from the partial knowledge of its Fourier coefficients on the set/spl Omega/? A typical result of this paper is as follows. Suppose that f is a superposition of| T| spikes f (t)=/spl sigma//sub/spl tau//spl isin/T/f (/spl tau/)/spl delta/(t-/spl tau/) obeying| T|/spl les/C/sub M//spl middot/(log N)/sup-
☆ 99 Cited by 12999 Related articles All 41 versions

Candes, T. Tao

# Example 1c: LASSO formulation

- To predict NBA salary. $n = 600$, but features $d = 10^5$.
- Too many features, but only a few can explain the salary

$$\text{salary} = w_1 \cdot \text{scores} + w_2 \cdot \text{WS} + w_3 \cdot \text{PER} + \cdots + w_d \cdot \text{favorite color}.$$

- **Step 1**: Write ideal formulation:

$$\min_w \frac{1}{2}\|Aw - b\|^2 + \|w\|_0.$$

- **Step 2**: Write a related-but-simpler formulation:

$$\min_w \frac{1}{2}\|Aw - b\|^2 + \|w\|_1.$$

  Under suitable conditions, $\ell_1$-minimization $\Leftrightarrow$ $\ell_0$-minimization

- **Step 3**: Find algorithm to solve it.
  - GD doesn't work well (non-smooth).
  - How about CD?

# Example 1c: LASSO formulation

- To predict NBA salary. $n = 600$, but features $d = 10^5$.
- Too many features, but only a few can explain the salary

$$\text{salary} = w_1 \cdot \text{scores} + w_2 \cdot \text{WS} + w_3 \cdot \text{PER} + \cdots + w_d \cdot \text{favorite color}.$$

- **Step 1**: Write ideal formulation:

$w = \begin{pmatrix} w_1 \\ \vdots \\ w_d \end{pmatrix}$ is sparse.

$$\min_w \frac{1}{2}\|Aw - b\|^2 + \lambda \|w\|_0.$$

$\|w\|_0$ = # of nonzero entries in $w$

prediction error → sparsity

- **Step 2**: Write a related-but-simpler formulation:

$$\min_w \frac{1}{2}\|Aw - b\|^2 + \|w\|_1.$$

Under suitable conditions, $\ell_1$-minimization $\Leftrightarrow$ $\ell_0$-minimization

- **Step 3**: Find algorithm to solve it.
  - GD doesn't work well (non-smooth).
  - How about CD?

# Example 1c: LASSO formulation

- To predict NBA salary. $n = 600$, but features $d = 10^5$.
- Too many features, but only a few can explain the salary

  salary $= w_1 \cdot$ scores $+ w_2 \cdot$ WS $+ w_3 \cdot$ PER $+ \cdots + w_d \cdot$ favorite color.

- **Step 1**: Write ideal formulation:

$$\min_w \frac{1}{2}\|Aw - b\|^2 + \|w\|_0.$$

discrete problem

- **Step 2**: Write a related-but-simpler formulation:

continuous problem

$$\min_w \frac{1}{2}\|Aw - b\|^2 + \|w\|_1.$$

  Under suitable conditions, $\ell_1$-minimization $\Leftrightarrow$ $\ell_0$-minimization

- **Step 3**: Find algorithm to solve it.
  - GD doesn't work well (non-smooth).
  - How about CD?

# Example 1c: LASSO formulation

- To predict NBA salary. $n = 600$, but features $d = 10^5$.
- Too many features, but only a few can explain the salary

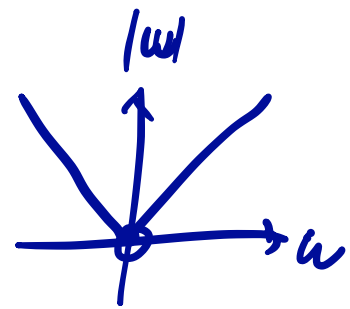
$$\text{salary} = w_1 \cdot \text{scores} + w_2 \cdot \text{WS} + w_3 \cdot \text{PER} + \cdots + w_d \cdot \text{favorite color}.$$

- **Step 1**: Write ideal formulation:

$$\min_w \frac{1}{2}\|Aw - b\|^2 + \|w\|_0.$$

- **Step 2**: Write a related-but-simpler formulation:

$$\min_w \frac{1}{2}\|Aw - b\|^2 + \|w\|_1.$$




Under suitable conditions, $\ell_1$-minimization $\Leftrightarrow$ $\ell_0$-minimization

- **Step 3**: Find algorithm to solve it.
  - GD doesn't work well (non-smooth).
  - How about CD?

# Example 1c: algorithm

- 1-dim problem?

$$\min_{w_i} \frac{1}{2}(\|A_i\|^2 w_i - (A_i^T r + \|A_i\|^2 w_i^{\text{old}}))^2 + \lambda |w_i|.$$

- Soft-thresholding: $x^+ = \text{argmin}_x (x - c)^2 + \lambda |x|$,

$$x^+ = \begin{cases} c + \lambda, & c < -\lambda, \\ 0, & c \in [-\lambda, \lambda] \\ c + \lambda, & c > \lambda. \end{cases}$$

- CD:

- **glmnet**: popular package for solving LASSO. Use CD.

# Example 1c: algorithm

- 1-dim problem? $(ax-c)^2+|x|_1,\ x\in\mathbb{R}$.

$$\min_{w_i} \frac{1}{2}(\|A_i\|^2 w_i - (A_i^T r + \|A_i\|^2 w_i^{\text{old}}))^2 + \lambda|w_i|.$$

- Soft-thresholding: $x^+ = \text{argmin}_x (x-c)^2 + \lambda|x|$,

$$x^+ = \begin{cases} c+\lambda, & c < -\lambda, \\ 0, & c \in [-\lambda, \lambda] \\ c+\lambda, & c > \lambda. \end{cases}$$

- CD:

- **glmnet**: popular package for solving LASSO. Use CD.

# Example 1c: algorithm

- 1-dim problem?

$$\min_{w_i} \frac{1}{2}(\|A_i\|^2 w_i - (A_i^T r + \|A_i\|^2 w_i^{\text{old}}))^2 + \lambda |w_i|.$$

- Soft-thresholding: $x^+ = \text{argmin}_x (x - c)^2 + \lambda |x|$,

$$x^+ = \begin{cases} c + \lambda, & c < -\lambda, \\ 0, & c \in [-\lambda, \lambda] \\ c + \lambda, & c > \lambda. \end{cases}$$

- CD: store $r = b - Ax$. At each round $k$, for $i = 1, 2, \cdots, n$, $x_i \leftarrow \arg\min_{x_i} f(x_i, x_{-i})$ update $r$.
- **glmnet**: popular package for solving LASSO. Use CD.

# Example 1c: brief history of CD for LASSO

History:

Idea appeared in Fu (1998), and again in Daubechies et al. (2004), but was not popular

Friedman et al. (2007) really sparked interest in statistics and ML community

People later realize that the convergence of CD for LASSO (and more general problems) has been thoroughly studied in [Tseng 2001].

# Summary of $f(Ax)$ type problem

Linear system $Ax = b$ is the most fundamental computational problem.

Many problems in machine learning and engineering are of the form $f(Ax) + G(x)$, where $G$ is a regularizer.

- Least squares and linear regression
  - One example: medical imaging, Reynolds et al. 2011
- Nonnegative least squares
- LASSO (+extension: group LASSO, elastic net); package glmnet
- SVM (package libsvm)

CD can solve these problems very fast – with the "residual" trick.

# Outline

# Most Basic of This Type

The example at the beginning of the lecture:

$$f(x, y) = (xy - 1)^2.$$

CD:

$$\begin{cases} x^{new} \leftarrow \arg\min_x f(x, y^{old}) \\ \nabla_x f = 2(xy-1)\cdot y \overset{set}{=} 0 \Rightarrow xy^2 - y = 0 \\ \qquad\qquad \Rightarrow x = \frac{y}{y^2}. \\ y \leftarrow \arg\min_y f(x^{new}, y). \end{cases}$$

# Example 2: Matrix Factorization

$= \sum_{(i,j)} (M_{ij} - x_i y_j)^2$.

- **Rank-1 case**: $\min_{x,y} \|M - xy^T\|_F^2$, where $M \in \mathbb{R}^{m\times n}, x \in \mathbb{R}^{m\times 1}, y \in \mathbb{R}^{n\times 1}$.

Write down how to solve it.

- Version 1 of BCD: 2 blocks $(x, y)$ alternating.

Set $\frac{\partial f}{\partial x} = (M - xy^T)y = 0$, $\frac{\partial f}{\partial y} = (M - xy^T)^T x = 0$.

$$\begin{cases} x^{new} \leftarrow My/\|y\|^2, \\ y \leftarrow M^T x^{new}/\|x^{new}\|^2. \end{cases}$$

Same (parallel)

- Version 2: Per-coordinate update, $(x_1, \dots, x_m, y_1, \dots, y_n)$, $(m+n)$ blocks.

$$\begin{cases} x_i^{new} \leftarrow \frac{\sum_j M_{ij} y_j}{\sum_j y_j^2} = \frac{\langle M_{i,:}, y\rangle}{\|y\|^2}, & i = 1, 2, \dots, m. \\ y_j \leftarrow \frac{\sum_i M_{ij} x_i^{new}}{\|x_i^{new}\|^2} = \frac{\langle M_{:,j}, x^{new}\rangle}{\|x^{new}\|^2}, & j = 1, \dots, n. \end{cases}$$

The update of $x_i$ does not depend on other $x_j$'s,
So update $x_i$'s in parallel
$\Leftrightarrow$ update $x_i$'s sequentially

- **Rank-r case**: $\min_{X,Y} \|M - XY^T\|_F^2$, where $M \in \mathbb{R}^{m\times n}, X \in \mathbb{R}^{m\times r}, Y \in \mathbb{R}^{n\times r}$.

Several versions of CD exist.

$Ax=b$.
$Ax=\lambda x \Leftrightarrow \min \|A-xx^T\|_F^2$.

Two fundamental problems

# Extension: NMF, tensor decomposition, matrix completion, etc.

$f(xy)$
$\rightarrow f(xyz)$
$\downarrow$
$f(x\otimes(y\otimes z))$

- Matrix factorization (MF) has wide applications
  - Big data matrix always has low rank structure (recall Lecture 6, NHL data matrix)
- Many such MF problems can be solved by CD.
- Extension 1: NMF. nonnegative MF. 10,000 papers. e.g. blind source separation
- Extension 2: Tensor decomposition [Kolda-09] most popular algorithm: ALS (Alternating Least Squares) most cited in SIAM.

- Extension 3: Matrix completion. [Sun-2015] Netflix prize. $8 million 2yo
- Extension 4: Deep learning (not well-known) back propogation has two versions. GD vs CD.

# Outline

# Possible Failure on Nonsmooth Problems

When talking about CD, many people worry about convergence.

There is a reason to worry:

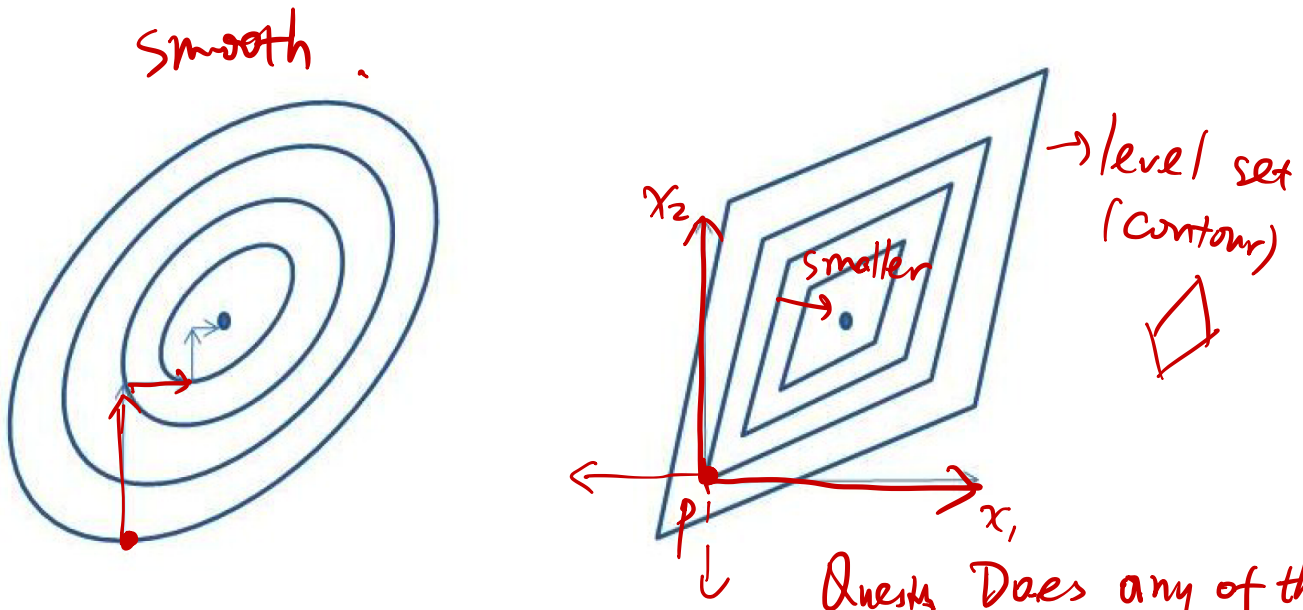



Figure: Comparison of CD on smooth and nonsmooth problems

# Convergence Theory

But there is a reason not to worry:

**Theorem 7.1** (Bertsekas '99 corollary). If f(x) is a continuously differentiable function, and $x_i \in \mathcal{X}_i$ is closed convex subset of $\mathbb{R}^{d_i}$. Suppose

e.g. $\mathcal{X}_i = \{x_i \geq 0\}$.

- $f$ is strictly convex in each block $x_i$, and has a minimum.

Let $\{x^r\}$ be the sequence generated by BCD. Then, every limit point of $x^r$ is a stationary point.

Simply put, smooth + per-block strict convexity implies convergence.

Remark: [Bertsekas '99] version 2 presented a wrong result; corrected in version 3. More general version than the one presented here (no need of per-block convexity)

# Applying to Examples

Apply the theory to examples?

- $f(Ax)$-type problems, e.g., linear regression: $\min_x \|A_1x_1 + \cdots + A_nx_n - b\|^2$.
  - per-block problem $\min_x \|A_ix + c\|^2$.
  - When $A_i^T A_i$ is full-rank, subproblem strictly convex
- $f(XY)$-type problems, e..g, matrix factorization $\min_{x,y\in\mathbb{R}^n} \|M - xy^T\|_F^2$.
  - Depending on blocks. If choose $x$ to be block, then coefficient matrix $y^Ty = \|y\|^2$.
  - When $\|y\| \neq 0$, subproblem strictly convex.
  - If the iterates always satisfy $\|x\| \neq 0, \|y\| \neq 0$, then every limit point is stationary.
  - Again, limit point may _____, need _____ to ensure _____

1-dim: $(ax-b)^2$, ...any cvx? Answer: Not necessarily. If $a=0$, no, $a\neq 0$, yes.

$A = \begin{bmatrix} | & | & \times & \times & \times \\ | & | & \times & \times & \times \\ | & | & \times & \times & \times \end{bmatrix}$, $A_i = \begin{bmatrix} | & | \\ | & | \\ | & | \end{bmatrix}$, $A_i^TA_i$ not full-rank, the subproblem is not strictly convex.

1-d.m: $(xy-1)^2$, subproblem strictly convex?

If $x\neq 0, y\neq 0$, then yes; otherwise, no.

not exist, bounded level set, limit point exists.

# Stronger Convergence Results

We have a very good understanding of convergence of CD now:

Tseng 2001, Convergence of a block coordinate descent method for nondifferentiable minimization.

Razaviyayn, Hong and Luo, 2013: A unified convergence analysis of block successive minimization methods for nonsmooth optimization.

BSUM: Block successive upper bound minimization

**First lesson**: Adding $\lambda\|\cdot\|^2$ term to the subproblem can guarantee convergence in many cases.

not $\lambda\|x_i\|^2$.

- Solve $x_i \leftarrow \operatorname{argmin}_{x_i} f(x_i, x_{-i}^{\text{old}}) + \lambda\|x_i - x_i^{\text{old}}\|^2$.
- Again, not guarantee limit point exists

**Second lesson**: inexact versions of BCD, e.g., BCGD, also converge (for smooth problems, and many non-smooth problems)

Subproblem not solved exactly. e.g. $x_i^{\text{new}} = x_i + \alpha \nabla_i f(x) \neq \operatorname{argmin}_{x_i} f(x_i, x_{-i})$

$x_i^{\text{new}} = x_i + \alpha \nabla_i f(x)$: CGD

$\operatorname{argmin}_{x_i} f(x_i, x_{-i})$: CD, exact version

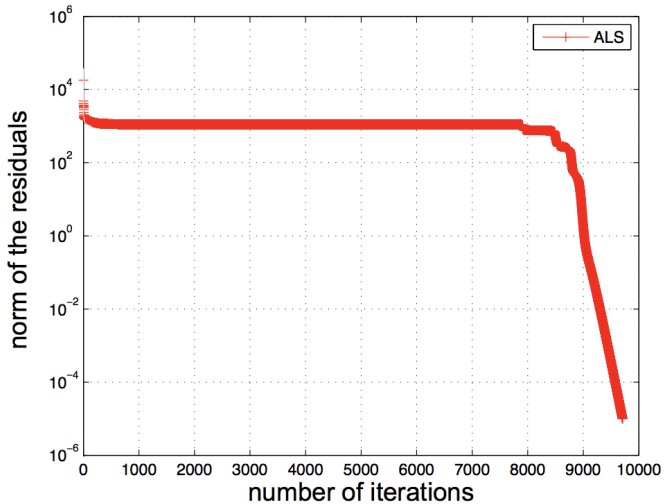
# Is Theory Needed?

Maybe in practice BCD just works?

In tensor decomposition field, the convergence behavior of BCD (called ALS) was a major issue.

Comon et al. 2009 “Tensor Decompositions, Alternating Least Squares and other Tales. ”.

Swamp effect reported.




partially because subproblems are degenerate (not strictly convex), or close to degenerate.

Figure: swamp effect in ALS for tensor decomposition

Adding $\lambda\|x_i - x_i^{\text{old}}\|^2$ can guarantee convergence (every limit point is stationary); and improve performance in practice.

# Convergence Speed

Two applications: $f(XY)$ and $f(Ax)$.

You may say:
CD for $f(XY)$ is better since GD has Lipschitz issue, fine.

Why using CD for linear regression, LASSO, etc.?

- In practice, faster (not convincing enough; maybe using different data GD is faster?)
- But why? (a bit theory?)

# Direct Relation of GD and CGD

Let's come back to $\min_x \|Ax - b\|^2$. $= \|A_1x_1 + \cdots + A_nx_n - b\|^2$

- CD: at each round, for $i = 1, \ldots, n$,

$$x_i^+ = \frac{A_i^T(b - A_{-i}x_{-i})}{A_i^T A_i} = \frac{A_i^T(b - Ax)}{\|A_i\|^2} + x_i. \qquad (5)$$

$\frac{\partial f}{\partial x_i}$.

GD: $x^+ = x - \alpha \nabla f(x) = x - \alpha A^T(Ax - b)$.

- GD: $x_i^+ = x_i + \alpha A_i^T(b - Ax)$, for all $i$ simultaneously.
- **Observation**: CD is a sequential version of GD, with stepsize $1/L_i$, where $L_i = \|A_i\|^2$. two differences: 1) stepsize; 2) sequential

**GD v.s. BCGD**: Parallel v.s. Sequential.
(Jacobi) (Gauss-Seidel)

Question (try at home): What if using stepsize $\frac{1}{L}$ for updating $x_i$?

# Why CD Faster

**Lesson** (non-rigorous): CD faster than GD because larger allowable stepsize!

$\frac{1}{\|A_i\|^2} \gg \frac{1}{L}$, $\forall i$.

- A lesson that many optimization experts don't know!
- This is also (almost) true for SGD (stochastic GD) we discuss next time

Rigorous results are known for randomized BCGD.

- Recent result: cyclic BCD can be very slow [Sun, Ye'16].

See IE598 course website (linked from my homepage "teaching"), post on "coordinate descent" for more details.

This might be true for $f(XY)$-type problem: largely open!

# Summary: Advantage of CD

When should you use CD? Subproblems easy to solve.

- Either closed form
- Or partial gradient easy to compute

What are the main advantages of CD (if applicable)?

- No parameter tuning.
- Cheap iteration (and... overall faster)
- Small storage.

More advantages?

- **Simple**!
- Orthogonal to other ideas! So can combine!

# Summary: Advantage of CD

When should you use CD? Subproblems easy to solve.

- Either closed form
- Or partial gradient easy to compute

What are the main advantages of CD (if applicable)?

- No parameter tuning.
- Cheap iteration (and... overall faster)
- Small storage.

More advantages?

- **Simple**!
- Orthogonal to other ideas! So can combine!

# Summary: Advantage of CD

When should you use CD? Subproblems easy to solve.

- Either closed form
- Or partial gradient easy to compute

What are the main advantages of CD (if applicable)?

- No parameter tuning.
- Cheap iteration (and... overall faster)
- Small storage.

More advantages?

- **Simple**!
- Orthogonal to other ideas! So can combine!

e.g. CD+momentum (not trivial).

# Summary

In this lecture, we learned the following:

- Different variants of CD
- Trick of CD for linear regression and beyond
- Apply CD to matrix factorization and beyond
- Convergence: smooth + per-block strict convexity (can add $\|x_i - x_i^{\text{old}}\|^2$ term)
- Convergence speed: typically faster than GD, since bigger stepsize allowed

# Summary

In this lecture, we learned the following:

- Different variants of CD
- Trick of CD for linear regression and beyond
- Apply CD to matrix factorization and beyond
- Convergence: smooth + per-block strict convexity (can add $\|x_i - x_i^{\text{old}}\|^2$ term)
- Convergence speed: typically faster than GD, since bigger stepsize allowed