Advanced Topics in Data Sciences

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 7: Randomized Linear Algebra

Laboratory for Information and Inference Systems (LIONS)
École Polytechnique Fédérale de Lausanne (EPFL)

EE-731 (Spring 2016)

License Information for Mathematics of Data Slides

Outline

- Coordinate descent methods (cont.)
  1. Coordinate descent methods for composite functions
  2. Coordinate descent primal-dual algorithm

- Randomized Linear Algebra
  1. Randomized matrix decompositions
  2. Comparison to classical methods

Recommended reading material:

- Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions,* SIAM review 53.2 (2011): 217-288. vspace1mm
- Michael W Mahoney, *Randomized algorithms for matrices and data,* Foundations and Trends in Machine Learning 3.2 (2011): 123-224.

Recall: Randomized CD algorithm

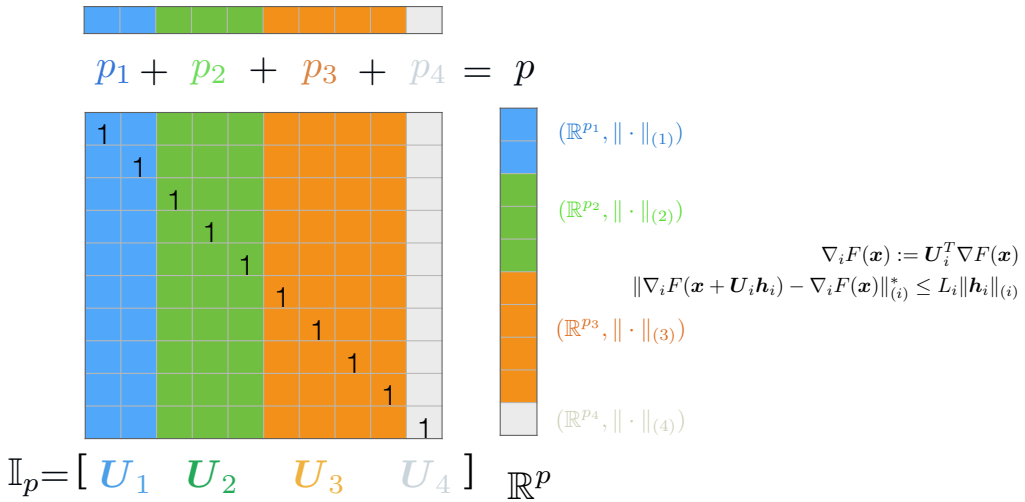
Randomized coordinate descent algorithm

1. Choose $\theta \in \mathbb{R}$ and $\mathbf{x}^0 \in \mathbb{R}^p$.

2. For $k = 0, 1, \ldots$ perform:

2a. Choose $i_k = \mathcal{A}_\theta$.

2b. Update $\mathbf{x}^{k+1} = \mathbf{x}^k - L_{i_k}^{-1} \boldsymbol{U}_{i_k} \left[\nabla_{i_k} f(\mathbf{x}^k)\right]^\#$.



- Sharp-operator : $[\mathbf{x}]^\# = \arg\max_{\boldsymbol{s} \in \mathbb{R}^p} \ \langle \mathbf{x}, \boldsymbol{s} \rangle - (1/2)\|\boldsymbol{s}\|^2 \implies$ for ℓ_2 norm, $[\mathbf{x}]^\# = \mathbf{x}$.
- $\mathcal{A}_\theta$ generates $i \in \{1, \ldots, s\}$ with probability $L_i^\theta / \sum_{j=1}^s L_j^\theta \implies$ for $\theta = 0$, uniform distribution.

Coordinate descent for composite minimization problem

Problem (Composite convex minimization)

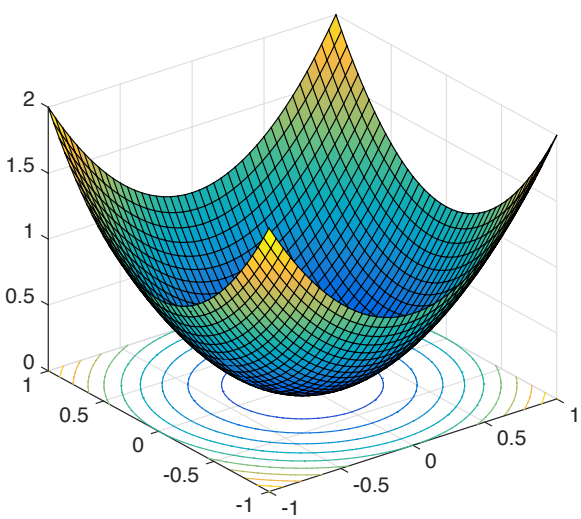
Consider the following unconstrained composite convex minimization problem:

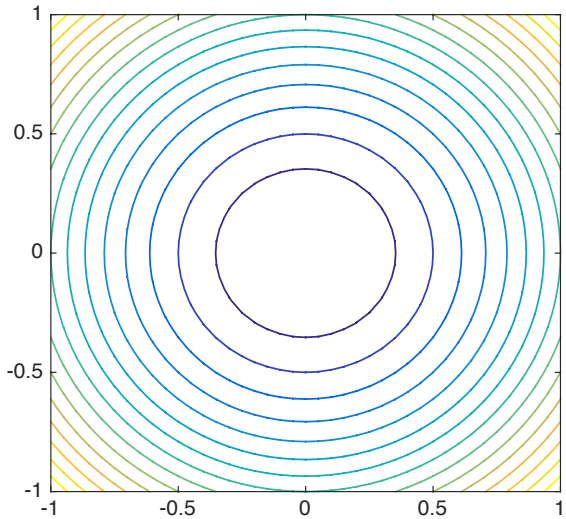
$$F^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \{F(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x})\}$$

- f *and* g *are both proper, closed, and convex.*
- ∇f *is* L*-Lipschitz continuous.*
- g *is possibly non-smooth.*
- *The solution set* $\mathcal{S}^\star := \{\mathbf{x}^\star \in dom(F) : F(\mathbf{x}^\star) = F^\star\}$ *is nonempty.*

Next: Examples that illustrates we need an additional assumption for CD to work for composite problems!

CD does not always converge for composite convex problems!





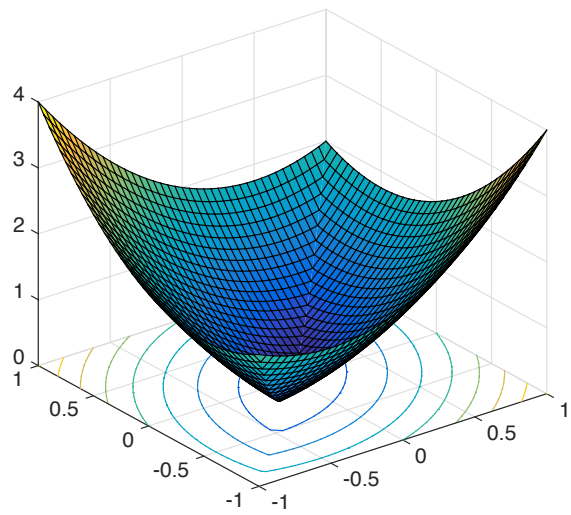
Smooth objective function: $f(\mathbf{x}) = \|\mathbf{x}\|_2^2$

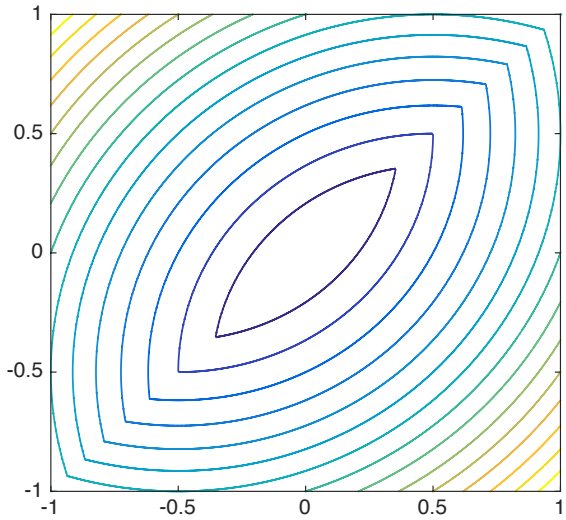
$f(\mathbf{x})$ is minimized along each coordinate axis, if and only if $\mathbf{x}$ is the global optimum.

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = 0, \quad \text{for } i = 1, \dots, p \quad \Longleftrightarrow \quad \nabla f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \dots, \frac{\partial f(\mathbf{x})}{\partial x_p} \right]^T = \mathbf{0}.$$

What if $f(\mathbf{x})$ is non-smooth?

CD does not always converge for composite convex problems!





Composite (non-smooth) objective function: $F(\mathbf{x}) = \|\mathbf{x}\|_2^2 + |x_1 - x_2|$

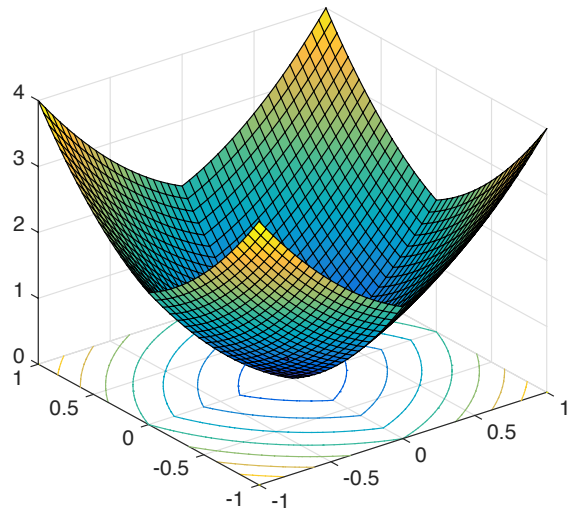
~~$F(\mathbf{x})$ is minimized along each coordinate axis, if and only if $\mathbf{x}$ is the global optimum.~~

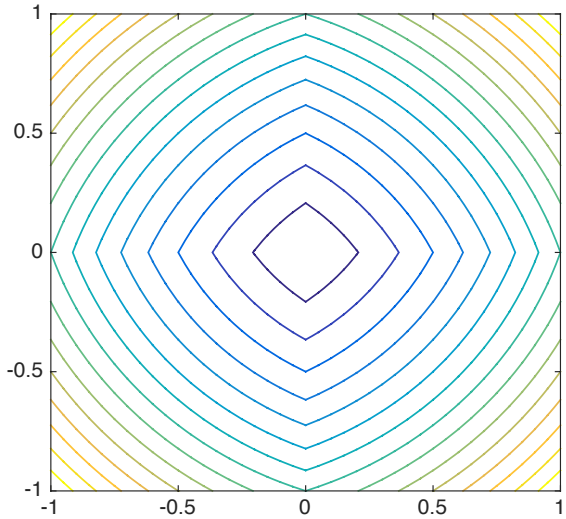
Statement above is not valid anymore!

Consider the point $(0.5, 0.5)$ as a counter example.

This is why we need an additional assumption!

CD does not always converge for composite convex problems!





Composite (non-smooth) objective function: $F(\mathbf{x}) = \|\mathbf{x}\|_2^2 + \|\mathbf{x}\|_1$

Denote $f(\mathbf{x}) := \|\mathbf{x}\|_2^2$ the smooth part and $g(\mathbf{x}) := \|\mathbf{x}\|_1$ the non-smooth part.

Assume that the non-smooth part is seperable: $g(\mathbf{x}) = \sum_{i=1}^{p} g_i(\mathbf{x}_i)$.

Then, $F(\mathbf{x})$ is minimized along each coordinate axis, if and only if $\mathbf{x}$ is the global optimum.

Coordinate descent for composite minimization problem

Problem (Composite convex minimization)

Consider the following unconstrained composite convex minimization problem:

$$\boxed{F^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \{F(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x})\}}$$

- f *and* g *are both proper, closed, and convex.*
- ∇f *is* L*-Lipschitz continuous.*
- g *is possibly non-smooth.*
- $\mathcal{S}^\star := \{\mathbf{x}^\star \in dom(F) : F(\mathbf{x}^\star) = F^\star\} \neq \emptyset$.

• g is separable: $g(\mathbf{x}) = \sum_{i=1}^p g_i(x_i)$, where $g_i : \mathbb{R} \to \mathbb{R}$ for all i, e.g.,
  - Unconstrained: $g(\mathbf{x}) =$ constant.
  - Box constrained: $g(\mathbf{x}) = \sum_{i=1}^s \mathbb{1}_{[a_i,b_i]}(x_i)$.
  - ℓ_q norm regularization: $g(\mathbf{x}) = \|\mathbf{x}\|_q^q$ where $q \geq 1$.

• g is block-separable: $p \times p$ identity matrix can be partitioned into column submatrices $\boldsymbol{U}_i$, $i = 1, \ldots, s$ such that $g(\mathbf{x}) = \sum_{i=1}^s g_i(\boldsymbol{U}_i^T \mathbf{x})$. Block-separable examples include group-sparse regularizers.

Examples: Composite convex problems with separable g

Example (LASSO)

$$\min_{\mathbf{x}} \underbrace{\frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2}_{f(\mathbf{x})} + \underbrace{\lambda\|\mathbf{x}\|_1}_{g(\mathbf{x})}.$$

Example (Support vector machine (SVM) with squared hinge loss)

$$\min_{\mathbf{x}} \underbrace{C\sum_i \max\{y_i(\boldsymbol{w}_i^T\mathbf{x}-b),0\}^2}_{g(\mathbf{x})} + \underbrace{\frac{1}{2}\|\mathbf{x}\|_2^2}_{f(\mathbf{x})}.$$

Example (SVM: dual form with bias term)

$$\min_{0\le\mathbf{x}\le C\mathbb{1}} \underbrace{\frac{1}{2}\sum_{i,j} x_i x_j y_i y_j \boldsymbol{K}(\boldsymbol{w}_i,\boldsymbol{w}_j)}_{f(\mathbf{x})} - \underbrace{\sum_i x_i}_{g(\mathbf{x})}.$$

Examples: Composite convex problems with separable g

Example (Logistic regression with ℓ_q norm regularization)

$$\min_{\mathbf{x}} \underbrace{\frac{1}{p}\sum_i \log(1+\exp(-b_i \boldsymbol{w}_i^T \mathbf{x}))}_{g(\mathbf{x})} + \underbrace{\lambda \|\mathbf{x}\|_q^q}_{f(\mathbf{x})}.$$

Example (Semi-supervised learning with Tikhonov regularization)

$$\min_{\mathbf{x}} \underbrace{\sum_{i \in \{\text{labeled data}\}} (\mathbf{x}_i - \mathbf{y}_i)^2}_{g(\mathbf{x})} + \underbrace{\lambda \mathbf{x}^T \boldsymbol{L} \mathbf{x}}_{f(\mathbf{x})}.$$

Example (Relaxed linear programming)

$$\min_{\mathbf{x}\geq 0} \boldsymbol{c}^T \mathbf{x} \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{b} \;\Rightarrow\; \min_{\mathbf{x}\geq 0} \underbrace{\boldsymbol{c}^T \mathbf{x}}_{g(\mathbf{x})} + \underbrace{\lambda \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2}_{f(\mathbf{x})}.$$

Randomized proximal coordinate descent algorithm

$$F^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \left\{ F(\mathbf{x}) := f(\mathbf{x}) + \sum_{i=1}^{s} g_i(\mathbf{x}_i) \right\}$$

Randomized coordinate descent for composite functions (RCDC)

1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$ and $(\gamma_k)_{k\in\mathbb{N}} \in]0, +\infty[^{\mathbb{N}}$.
2. For $k = 0, 1, \ldots$ perform:
2a. Pick $i_k \in \{1, \ldots, s\}$ uniformly at random.
2b. Update coordinate i_k:

$$\mathbf{x}_{i_k}^{k+1} = \underset{\mathbf{v}\in\mathbb{R}^{p_{i_k}}}{\arg\min}\ g_{i_k}(\mathbf{v}) + \left\langle \mathbf{v}, \nabla_{i_k} f(\mathbf{x}^k) \right\rangle + \frac{1}{2\alpha_k} \|\mathbf{v} - \mathbf{x}_{i_k}^k\|_{(i_k)}^2.$$

- If $\|\cdot\|_{(i_k)} = \|\cdot\|_2$, then we can simplify the update rule as

$$\mathbf{x}_{i_k}^{k+1} = \mathrm{prox}_{\alpha_k g_{i_k}} \left(\mathbf{x}_{i_k}^k - \alpha_k \nabla_{i_k} f(\mathbf{x}^k) \right).$$

Convergence of RCDC

Suppose that ∇f_i is Lipschitz continuous with respect to some norm $\|\cdot\|_{(i)}$ for $i = 1, 2, \dots s$, that is

$$\|\nabla_i f(\mathbf{x} + \boldsymbol{U}_i \mathbf{t}) - \nabla_i f(\mathbf{x})\|_{(i)}^* \leq L_i \|\mathbf{t}\|_{(i)}, \qquad \forall \mathbf{t} \in \mathbb{R}^{p_i}.$$

Theorem (Convergence without strong convexity [7])

Choose a target confidence $0 < \rho < 1$. *For any target accuracy* $\varepsilon < F(\mathbf{x}^0) - F^\star$,

$$\mathbb{P}(F(\mathbf{x}^k) - F^\star \leq \varepsilon) \geq 1 - \rho, \quad \textit{for any} \quad k \geq \frac{2sD_L}{\varepsilon}(1 - \log(\rho)) + 2 - \frac{2sD_L}{F(\mathbf{x}^0) - F^\star},$$

where

$$D_L := \max\left\{F(\mathbf{x}^0) - F^\star, \max_{\mathbf{y}} \max_{\mathbf{x}^\star \in \mathcal{X}^\star} \Big\{ \underbrace{\sum_{i=1}^{s} L_i \|\mathbf{y}_i - \mathbf{x}_i^\star\|_{(i)}^2}_{:= \|\mathbf{y} - \mathbf{x}^\star\|_L^2} : F(\mathbf{y}) \leq F(\mathbf{x}^0) \Big\} \right\}.$$

Convergence of RCDC

Suppose that ∇f_i is Lipschitz continuous with respect to some norm $\|\cdot\|_{(i)}$ for $i = 1, 2, \dots s$, that is

$$\|\nabla_i f(\mathbf{x} + \boldsymbol{U}_i \mathbf{t}) - \nabla_i f(\mathbf{x})\|_{(i)}^* \leq L_i \|\mathbf{t}\|_{(i)}, \qquad \forall \mathbf{t} \in \mathbb{R}^{p_i}.$$

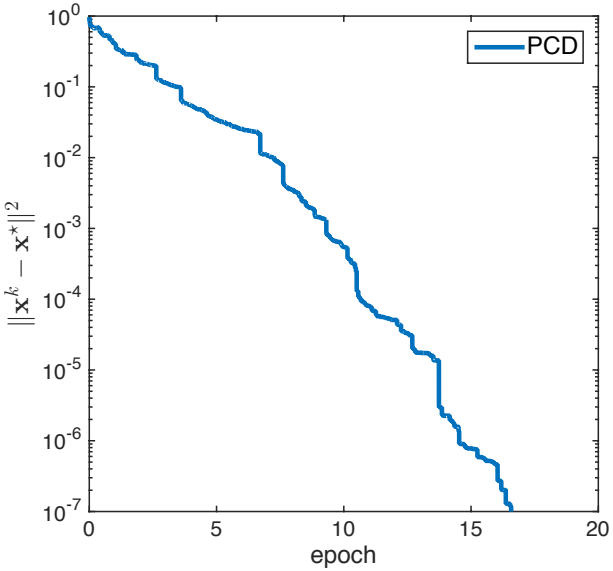
Theorem (Convergence with strong convexity [8])

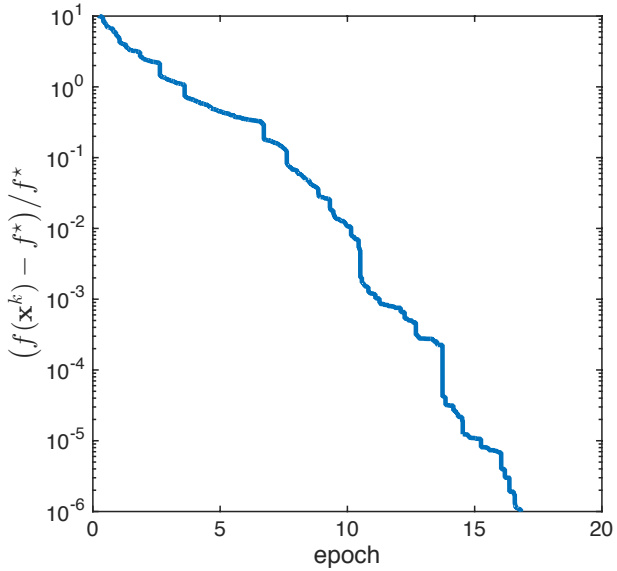
Suppose f is a strongly convex function with convexity constant μ. Let us set $\alpha_k = 1/L_{max}$ for all k, where $L_{max} = \max_i L_i$, then

$$\mathbb{E}[F(\mathbf{x}^k) - F^\star] \leq \left(1 - \frac{\mu}{sL_{\max}}\right)^k \left(F(\mathbf{x}^0) - F^\star\right).$$

Example: LASSO

$$\min_{\mathbf{x}} \left\{ f(\mathbf{x}) := \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda \|\mathbf{x}\|_1 : \mathbf{x} \in \mathbb{R}^p \right\}$$





Synthetic problem setup

- $\mathbf{A} := \mathrm{randn}(n, p)$ - standard Gaussian $\mathcal{N}(0, \mathbb{I})$, with $n = 1000$, $p = 500$.
- $\mathbf{x}^\natural \in \mathbb{R}^p$ is 50-sparse with Gaussian i.i.d. entries, normalized to $\|\mathbf{x}^\natural\|_2 = 1$.
- $\mathbf{b} := \mathbf{A}\mathbf{x}^\natural + \mathbf{w}$, where $\mathbf{w}$ is Gaussian white noise. SNR is 30dB.
- $\theta = 0$, so coordinates are chosen uniformly random.
- $\lambda := 10^{-2}$.

Accelerated parallel proximal coordinate descent method

Accelerated parallel proximal coordinate descent method (APPROX)

1. Choose $\mathbf{v}^0 = \mathbf{x}^0 \in \mathbb{R}^p$ and $\alpha_0 = \tau/s$.
2. For $k = 0, 1, \ldots$ perform:
2a. $\mathbf{y}^k = (1-\alpha_k)\mathbf{x}^k + \alpha_k \mathbf{v}^k$.
2b. Generate a random set of coordinate blocks $\mathcal{S}_k$ with uniform block sampling.
2c. For $i \in \mathcal{S}_k$, perform:

$$\mathbf{v}_i^{k+1} = \arg\min_{\mathbf{v}\in\mathbb{R}^{p_i}} \left\{ \left\langle \mathbf{v} - \mathbf{y}_i^k, \nabla_i f(\mathbf{y}^k) \right\rangle + \frac{s\alpha_k\sigma_i}{2\tau} \|\mathbf{v} - \mathbf{v}_i^k\|_{(i)}^2 + g_i(\mathbf{v}) \right\}.$$

2d. $\mathbf{x}_i^{k+1} = \mathbf{y}_i^k + \frac{s\alpha_k}{\tau}(\mathbf{v}_i^{k+1} - \mathbf{v}_i^k)$.
3. $\alpha_{k+1} = \frac{1}{2}\left(\sqrt{\alpha_k^4 + 4\alpha_k^2} - \alpha_k^2 \right)$.

- Uniform block sampling: $\mathbb{P}(i \in \mathcal{S}) = \mathbb{P}(j \in \mathcal{S})$ for all $i, j \in \{1, 2, \ldots, s\}$.
- $\tau = \mathbb{E}[|\mathcal{S}|]$.
- $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_s) \in \mathbb{R}_+^s$ satisfy $\forall \mathbf{x}, \mathbf{h} \in \mathbb{R}^p$:

$$\mathbb{E}\left[f\left(\mathbf{x} + \sum_{i\in\mathcal{S}} \boldsymbol{U}_i \mathbf{h}_i \right) \right] \leq f(\mathbf{x}) + \frac{\tau}{s}\left(\langle \nabla f(\mathbf{x}), \mathbf{h} \rangle + \frac{1}{2} \left\| \sum_{i\in\mathcal{S}} \boldsymbol{U}_i \mathbf{h}_i \right\|_{\boldsymbol{\sigma}}^2 \right),$$

where $\|\mathbf{x}\|_{\boldsymbol{\sigma}}^2 := \sum_{i=1}^s \sigma_i \|\mathbf{x}_i\|_{(i)}^2$.

Rate of convergence of APPROX: $\mathcal{O}(1/k^2)$

- Uniform block sampling: $\mathbb{P}(i \in \mathcal{S}) = \mathbb{P}(j \in \mathcal{S})$ for all $i, j \in \{1, 2, \dots, s\}$.
- $\tau = \mathbb{E}[|\mathcal{S}|]$.
- $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_s) \in \mathbb{R}_+^s$ satisfy $\forall \mathbf{x}, \mathbf{h} \in \mathbb{R}^p$:

$$\mathbb{E}\left[f\left(\mathbf{x} + \sum_{i \in \mathcal{S}} \boldsymbol{U}_i \mathbf{h}_i\right)\right] \leq f(\mathbf{x}) + \frac{\tau}{s}\left(\langle \nabla f(\mathbf{x}), \mathbf{h}\rangle + \frac{1}{2}\Big\|\sum_{i \in \mathcal{S}} \boldsymbol{U}_i \mathbf{h}_i\Big\|_{\boldsymbol{\sigma}}^2\right),$$

where $\|\mathbf{x}\|_{\boldsymbol{\sigma}}^2 := \sum_{i=1}^s \sigma_i \|\mathbf{x}_i\|_{(i)}^2$.

Theorem ([4])

Let $\{\mathbf{x}^k\}_{k \geq 0}$ be a sequence generated by APPROX. Then, for any optimal point $\mathbf{x}^\star$, we have

$$\mathbb{E}[F(\mathbf{x}^k) - F^\star] \leq \frac{4s^2}{((k-1)\tau + 2s)^2} C,$$

where

$$C = \left(1 - \frac{\tau}{s}\right)\left(F(\mathbf{x}^0) - F^\star\right) + \frac{1}{2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_{\boldsymbol{\sigma}}^2.$$

Coordinate descent primal-dual algorithm

Problem (Composite minimization problem with linear operator)

Consider the following minimization problem

$$\boxed{F^\star := \min_{\mathbf{x}\in\mathbb{R}^p} \Big\{ f(\mathbf{x}) + g(\mathbf{x}) + h(\mathbf{A}\mathbf{x}) \Big\}.}$$

- g *and* h *are convex,* f *is convex and differentiable.*
- $\mathbf{A} \in \mathbb{R}^{q\times p}$.

This problem can be transformed to finding saddle points of the Lagrangian function

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{x}) + \langle \mathbf{y}, \mathbf{A}\mathbf{x} \rangle - h^\star(\mathbf{y}),$$

where $h^\star : \mathbf{y} \mapsto \sup_{\mathbf{z}} \ \langle \mathbf{y}, \mathbf{z} \rangle - h(\mathbf{z})$ is the Fenchel-Legendre transform of h.

Examples

Example (Total variation + ℓ_1 regularized least squares regression)

$$\min_{\mathbf{x}\in\mathbb{R}^p} \underbrace{\frac{1}{2}\|\boldsymbol{M}\mathbf{x}-\mathbf{b}\|_2^2}_{f(\mathbf{x})} + \underbrace{\alpha r\|\mathbf{x}\|_1}_{g(\mathbf{x})} + \underbrace{\alpha(1-r)\|\mathbf{A}\mathbf{x}\|_{2,1}}_{h(\mathbf{A}\mathbf{x})},$$

where

$$\|\mathbf{A}\mathbf{x}\|_{2,1} = \sum_j \|\mathbf{A}_{ij}\mathbf{x}_i\|_2, \quad \mathbf{x} = (\mathbf{x}_i)_i.$$

Example (Dual SVM)

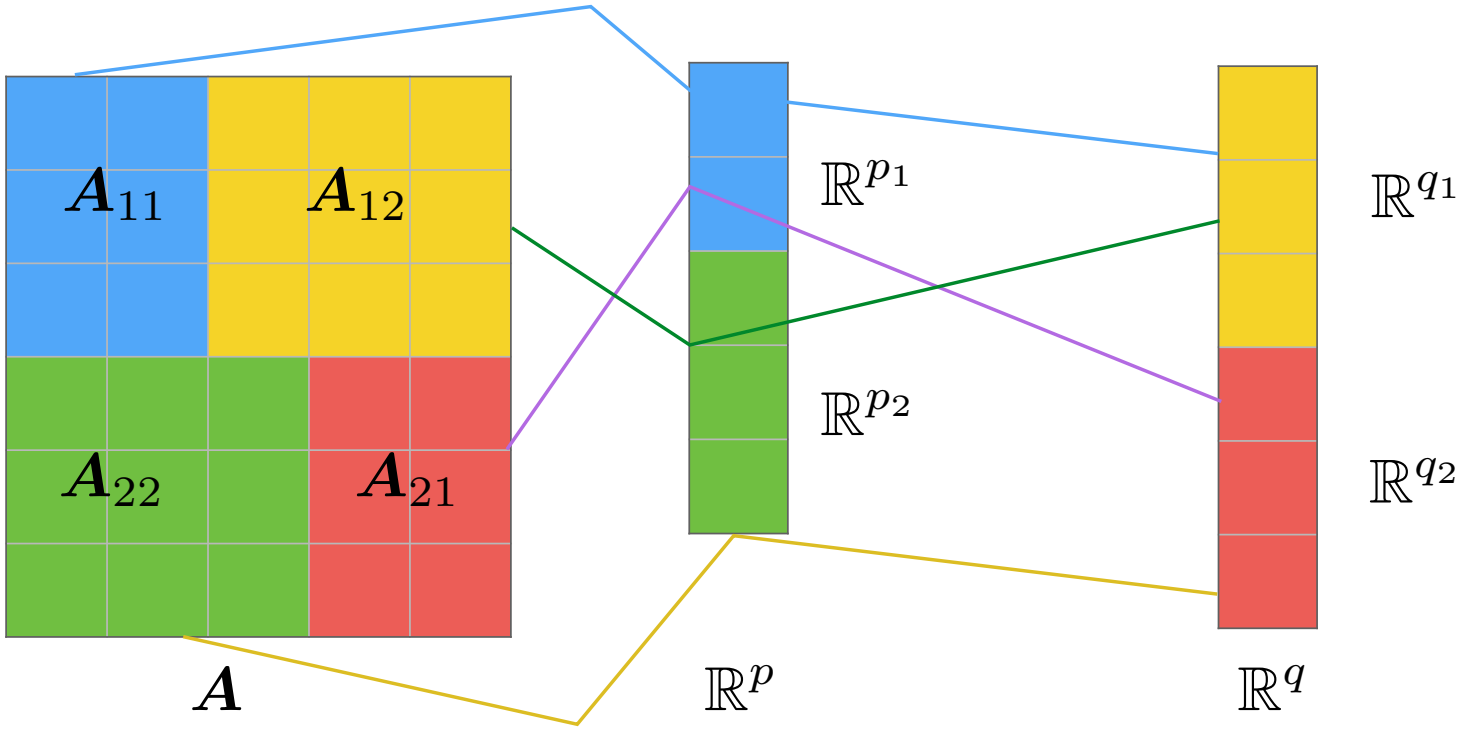
$$\min_{\mathbf{x}\in\mathbb{R}^p} \underbrace{\frac{1}{2\lambda}\|\mathbf{A}(\mathbf{b}\odot\mathbf{x})\|_2^2 - \boldsymbol{e}^T\mathbf{x}}_{f(\mathbf{x})} + \underbrace{\sum_{i=1}^p \iota_{[0,C_i]}(x_i)}_{g(\mathbf{x})} + \underbrace{\iota_{\mathbf{b}^\perp}(\mathbf{x})}_{h(\mathbb{I}_p\mathbf{x})},$$

where $\mathbf{b}\odot\mathbf{x}$ is the component-wise multiplications of two vectors $\mathbf{b}$ and $\mathbf{x}$ and

$$\mathbf{b}^\perp = \left\{\mathbf{x}\in\mathbb{R}^p \mid \mathbf{b}^T\mathbf{x} = 0\right\}.$$

Set up

1. $\mathbb{R}^p = \mathbb{R}^{p_1} \times \ldots \times \mathbb{R}^{p_s}$ and $\mathbb{R}^q = \mathbb{R}^{q_1} \times \ldots \times \mathbb{R}^{q_t}$. Hence, matrix $\mathbf{A} \in \mathbb{R}^{q \times p}$ can be decomposed in blocks of matrices $\mathbf{A}_{ij} \in \mathbb{R}^{q_j \times p_i}$.



2. $\mathbb{I}(j)$ indices the set of nonzero q_j-rows matrices and m_j its cardinal.
3. $\mathbb{J}(i)$ indices the set of nonzero p_i-columns matrices.

Coordinate descent primal-dual algorithm

Coordinate descent primal-dual algorithm

1. Choose $\sigma = (\sigma_1, \dots, \sigma_t)$, $\tau = (\tau_1, \dots, \tau_s)$, $\mathbf{x}^0 \in \mathbb{R}^p$, $\mathbf{y}^0 \in \mathbb{R}^q$ and initialize

$$\begin{cases} (\forall i \in \{1, \dots, s\}) & \boldsymbol{w}_i^0 = \sum_{j \in \mathbb{J}(i)} \mathbf{A}_{ji}^T \mathbf{y}_j^0(i). \\ (\forall j \in \{1, \dots, t\}) & \mathbf{z}_j^0 = (1/m_j) \sum_{i \in \mathbb{I}(j)} \mathbf{y}_j^0(i). \end{cases}$$

2. For $k = 0, 1, \dots$ perform:

2a. Choose $i_k \in \{1, \dots, s\}$ at random and uniformly.

2b. Compute:

$$\begin{cases} \bar{\mathbf{y}}^{k+1} = \operatorname{prox}_{\sigma h^\star}\Big(\mathbf{z}^k + \sigma \odot (\mathbf{A}\mathbf{x}^k)\Big) \\ \bar{\mathbf{x}}^{k+1} = \operatorname{prox}_{\tau g}\Big(\mathbf{x}^k - \tau \odot (\nabla f(\mathbf{x}^k) + 2\mathbf{A}^T \bar{\mathbf{y}}^{k+1} - \boldsymbol{w}^k)\Big). \end{cases}$$

2c. Update:

2c1. For $i = i_{k+1}$ and for each $j \in \mathbb{J}(i_{k+1})$:

$$\begin{cases} \mathbf{x}_i^{k+1} = \bar{\mathbf{x}}_i^{k+1}, \quad \mathbf{y}_j^{k+1}(i) = \bar{\mathbf{y}}_j^{k+1}(i) \\ \boldsymbol{w}_i^{k+1} = \boldsymbol{w}_i^k + \sum_{j \in \mathbb{J}(i)} \mathbf{A}_{ji}^\star \Big(\mathbf{y}_j^{k+1}(i) - \mathbf{y}_j^k(i)\Big) \\ \mathbf{z}_j^{k+1} = \mathbf{z}_j^k + \frac{1}{m_j}\Big(\mathbf{y}_j^{k+1}(i) - \mathbf{y}_j^k(i)\Big). \end{cases}$$

2c2. Otherwise: $\mathbf{x}_i^{k+1} = \mathbf{x}_i^k$, $\boldsymbol{w}_i^{k+1} = \boldsymbol{w}_i^k$, $\mathbf{z}_j^{k+1} = \mathbf{z}_j^k$, $\mathbf{y}_j^{k+1}(i) = \mathbf{y}_j^{k+1}(i)$.

- $\operatorname{prox}_{\tau g} : \mathbf{x} \mapsto \arg\min_{\mathbf{y}} \{f(\mathbf{y}) + (1/2)\|\mathbf{x} - \mathbf{y}\|_\tau^2\}$, where $\|\mathbf{x}\|_\tau^2 = \sum_{i=1}^s \tau_i^{-1} \|\mathbf{x}_i\|_{(i)}^2$.

Example

$$\min_{\mathbf{x}\in\mathbb{R}^p} \underbrace{\frac{1}{2\lambda}\|\mathbf{A}(\mathbf{b}\odot\mathbf{x})\|_2^2 - \boldsymbol{e}^T\mathbf{x}}_{f(\mathbf{x})} + \underbrace{\sum_{i=1}^{p} \iota_{[0,C_i]}(x_i)}_{g(\mathbf{x})} + \underbrace{\iota_{\mathbf{b}^\perp}(\mathbf{x})}_{h(\mathbb{I}_p\mathbf{x})}$$

We have: $\mathbb{R}^p = \mathbb{R} \times \ldots \times \mathbb{R}$ and

$$\mathbb{I}_p = \begin{pmatrix} \boxed{1} & \boxed{0} & \ldots & \boxed{0} \\ \boxed{0} & \boxed{1} & \ldots & \boxed{0} \\ \vdots & \vdots & \vdots & \vdots \\ \boxed{0} & \boxed{0} & \ldots & \boxed{1} \end{pmatrix}$$

- $\mathbb{J}(i) = \{i\}$ and $\mathbb{I}(j) = \{j\}$.
- m_j is the number of nonzero elements in jth column, i.e., 1.

Example (cont.)

$$\min_{\mathbf{x}\in\mathbb{R}^p} \underbrace{\frac{1}{2\lambda}\|\mathbf{A}(\mathbf{b}\odot\mathbf{x})\|_2^2 - \boldsymbol{e}^T\mathbf{x}}_{f(\mathbf{x})} + \underbrace{\sum_{i=1}^{p}\iota_{[0,C_i]}(x_i)}_{g(\mathbf{x})} + \underbrace{\iota_{\mathbf{b}^\perp}(\mathbf{x})}_{h(\mathbb{I}_p\mathbf{x})}$$

1. Choose $\sigma = (\sigma_1, \ldots, \sigma_p)$, $\tau = (\tau_1, \ldots, \tau_p)$, $\mathbf{x}^0 \in \mathbb{R}^p$, $\mathbf{y}^0 \in \mathbb{R}^p$.
2. For $k = 0, 1, \ldots$ perform:
2a. Choose $i_k \in \{1, \ldots, p\}$ at random and uniformly.
2b. Compute:

$$\begin{cases} \bar{\mathbf{y}}^{k+1} = \operatorname{prox}_{\sigma h^\star}\left(\mathbf{y}^k + \sigma \odot (\mathbf{A}\mathbf{x}^k)\right) \\ \bar{\mathbf{x}}^{k+1} = \operatorname{prox}_{\tau g}\left(\mathbf{x}^k - \tau \odot (\nabla f(\mathbf{x}^k) + \mathbf{A}^T(2\bar{\mathbf{y}}^{k+1} - \mathbf{y}^k))\right). \end{cases}$$

2c. Update:

$$(\mathbf{x}_i^{k+1}, \mathbf{y}_i^{k+1}) = \begin{cases} (\bar{\mathbf{x}}_i^{k+1}, \bar{\mathbf{y}}_{i_k}^{k+1}), & \text{if } i = i_{k+1}, \\ (\mathbf{x}_i^k, \mathbf{y}_i^k), & \text{otherwise.} \end{cases}$$

- $\nabla f(\mathbf{x}) = \lambda^{-1}\mathbf{b}^T \odot (\mathbf{A}^T\mathbf{A}(\mathbf{b}\odot\mathbf{x})) - \boldsymbol{e}^T$.
- $\operatorname{prox}_{\tau g}\mathbf{x} = P_{[0,\boldsymbol{C}]}\mathbf{x}$.
- $\operatorname{prox}_{\sigma h^*}\mathbf{x} = \mathbf{x} - \sigma \odot \operatorname{prox}_{\sigma^{-1}h}(\sigma^{-1}\odot\mathbf{x}) = \mathbf{x} - \sigma \odot P_{\mathbf{b}^\perp}^{\sigma^{-1}}(\sigma^{-1}\odot\mathbf{x})$.

Convergence's results

Theorem ([3])

Suppose that for every $i \in \{1, \dots, s\}$*:*

1. *There exists* $\beta_i \geq 0$ *such that*

$$(\forall \mathbf{x} \in \mathbb{R}^p)(\forall \mathbf{u} \in \mathbb{R}^{p_i}) \quad f(\mathbf{x} + \boldsymbol{U}_i \mathbf{u}) \leq f(\mathbf{x}) + \langle \boldsymbol{U}_i \mathbf{u}, \nabla f(\mathbf{x}) \rangle + \frac{\beta_i}{2} \|\mathbf{u}\|_{(i)}^2.$$

2. $\tau_i < 1/(\beta_i + \rho(\boldsymbol{B}))$*, where* $\boldsymbol{B} = \sum_{j \in \mathbb{J}(i)} m_j \sigma_j \mathbf{A}_{ji}^T \mathbf{A}_{ji}$ *and* $\rho(\boldsymbol{B})$ *denotes the spectral radius of* $\boldsymbol{B}$*, i.e., the maximum of absolute values of eigenvalues of* $\boldsymbol{B}$*.*

Then

1. $\mathbf{x}^k \to \mathbf{x}^\star$*.*
2. $\mathbf{y}_j^k(i) \to \mathbf{y}_j^\star$ *for every* $j \in \{1, \dots, t\}$ *and every* $i \in \mathbb{I}(j)$*.*

- **Note.** No result on convergence's rate!

Outline

- Coordinate descent methods (cont.)
  1. Coordinate descent methods for composite functions
  2. Coordinate descent primal-dual algorithm

- Randomized Linear Algebra
  1. Randomized matrix decompositions
  2. Comparison to classical methods

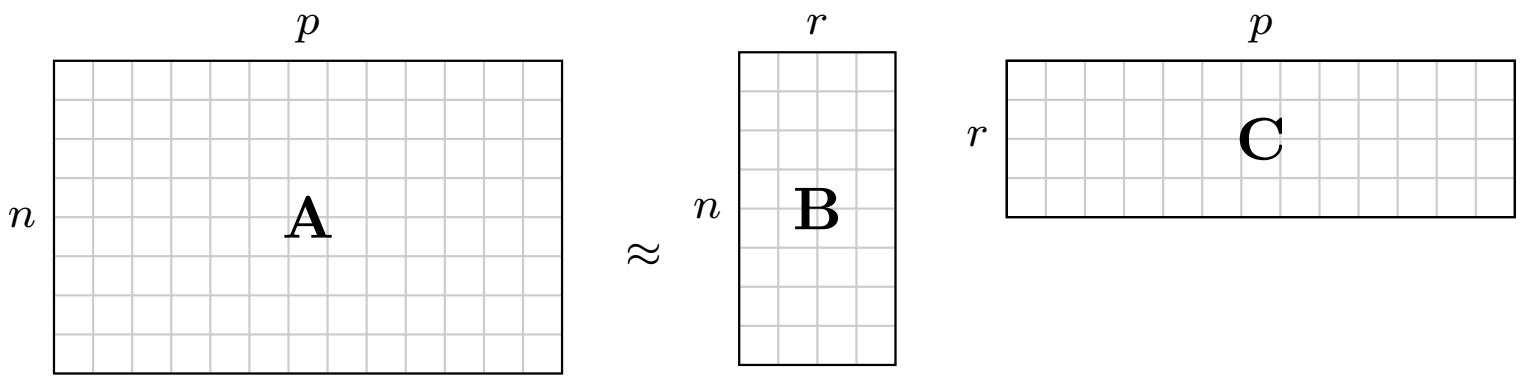
Top-Ten Algorithms of 20th century [2]:

- 1946: Monte Carlo Method
- 1947: Simplex Method for Linear Programming
- 1950: Krylov Subspace Iteration Method
- **1951: The Decompositional Approach to Matrix Computations**
- 1957: The Fortran Optimizing Compiler
- 1959: QR Algorithm for Computing Eigenvalues
- 1962: Quicksort Algorithms for Sorting
- 1965: Fast Fourier Transform.
- 1977: Integer Relation Detection
- 1987: Fast Multipole Method

Matrix Decompositions

- Cholesky, Schur, eigenvalue, QR and singular value decompositions (SVD) etc.
- Allows software packages that can be used to solve different linear algebra problems
- SVD and QR decompositions have $\mathcal{O}(np \min\{n, p\})$ complexity
- This can be the major computational bottleneck due to their superlinear dependence on matrix size
- Real data is often noisy, so it makes sense to sacrife accuracy for speed-up.

Example-I: Matrix vector multiplication



Advantages

1. Faster computation: $\mathcal{O}(np)$ instead of $\mathcal{O}(r(n+p))$.
2. Lower memory: $\mathcal{O}(np)$ instead of $\mathcal{O}(r(n+p))$.

• The approximation costs $\mathcal{O}(np\log(r) + r^2(n+p))$ with state-of-the-art when the decomposition is SVD.

Example-II: Robust Principal Component Analysis (RPCA)

For certain applications such as video surveillance, we need to solve

$$\min_{X=L+S} \|L\|_\star + \lambda \|S\|_1$$

which requires computation of the proximal operator of the nuclear norm

$$Z^* = \arg\min_Z \frac{1}{2}\|X - Z\|_F^2 + \lambda \|Z\|_\star.$$

We can only need to compute $Z^* = U_r \Sigma_r V_r$ where U_r and V_r contain the first r left and right singular vectors and Σ_r is a diagonal matrix with the first r singular values on its diagonal.

Complexities

- Truncated SVD with classical methods has $\mathcal{O}(npr)$ complexity .
- The randomized approach can cost as low as $\mathcal{O}(np\log(r) + r^2(n+p))$ operations.

Example-II: Robust Principal Component Analysis (RPCA)

Example

To compute the proximity operator for the nuclear norm in robust PCA for video background subtraction, we try

1. Lanczos-based SVD using PROACK software
2. Randomized factorization

The matrix to be decomposed has dimensions 61440 x 17884 (8.1 GB) and is taken from a video sequence. We see that U

- Faster even with one core
- Accuracies are indistinguishable
- Randomized method scales much better for parallel computation

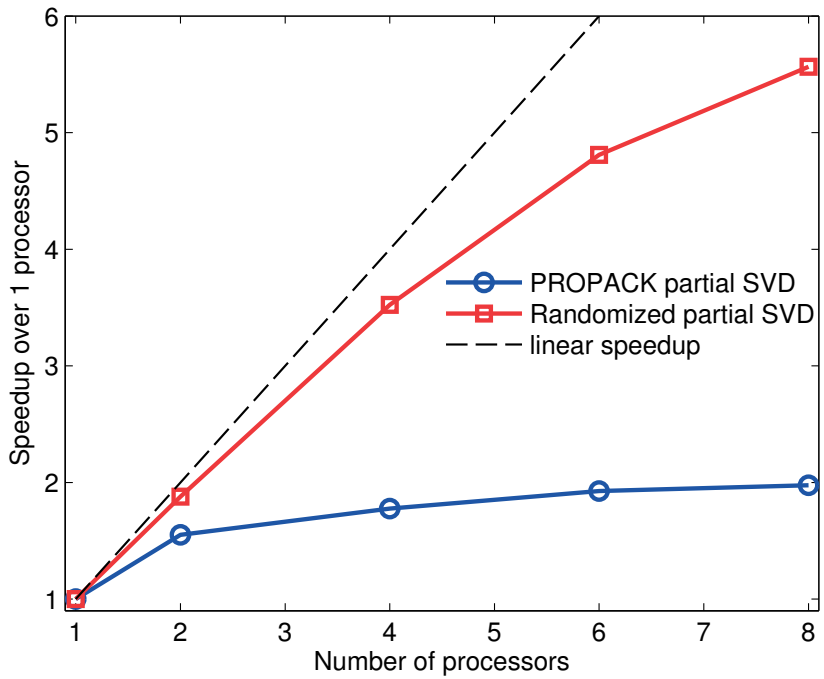


Figure: Computing the top 5 singular vectors of a 10^9 entry matrix using varying number of computer cores [1]

Randomized low rank decompositions: How do we do it ?

Step-1: Finding a range

- Apply a *randomized* algorithm to find an orthogonal low-dimensional basis $\mathbf{Q} \in \mathbb{R}^{n \times l}$ with $l \ll p$ that can well represent the matrix $\mathbf{A}$
- In other words, $\mathbf{Q}$, when approximated on its span, should well approximate $\mathbf{A}$:

$$\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^* \mathbf{A}$$

where $\mathbf{Q}\mathbf{Q}^*$ is the projection onto the subspace spanned by the basis $\mathbf{Q}$

Step-2: Decomposition

- Reduce the dimension using $\mathbf{Q}$ as the approximation above suggests
- Apply *classical linear* algebra which is no more prohibitive at these dimensions
- Obtain the desired decomposition

Randomized low rank decompositions: How do we do it ?

Step-1: Finding a range

- Apply a *randomized* algorithm to find an orthogonal low-dimensional basis $\mathbf{Q} \in \mathbb{R}^{n \times l}$ with $l \ll p$ that can well represent the matrix $\mathbf{A}$
- In other words, $\mathbf{Q}$, when approximated on its span, should well approximate $\mathbf{A}$:

$$\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^* \mathbf{A}$$

where $\mathbf{Q}\mathbf{Q}^*$ is the projection onto the subspace spanned by the basis $\mathbf{Q}$

Step-2: Decomposition

- Reduce the dimension using $\mathbf{Q}$ as the approximation above suggests
- Apply *classical linear* algebra which is no more prohibitive at these dimensions
- Obtain the desired decomposition

Step-1: Finding the range

Step-1: Finding a range

Given $\mathbf{A} \in \mathbb{R}^{n \times p}$, find $\mathbf{Q} \in \mathbb{R}^{n \times l}$ such that

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^* \mathbf{A}\| \approx \min_{\mathsf{rank}(\mathbf{X}) \leq r} \|\mathbf{A} - \mathbf{X}\|$$

- r is the target rank,
- $l = r + s$ number of columns used
- s is the number of oversamples

Method: Obtain random vectors in the range of $\mathbf{A}$ by multiplying it with random vectors

- From these vectors we can find an orthogonal basis $\mathbf{Q}$
- There exists a $\mathbf{Q} \in \mathbb{R}^{n \times r}$ which gives the optimum value of the above minimization problem (guess what it is !)
- But for a better approximation we oversample it: $\mathbf{Q} \in \mathbb{R}^{n \times (r+s)}$

Step-1: Finding the range

Step-1: Finding a range

Given $\mathbf{A} \in \mathbb{R}^{n \times p}$, find $\mathbf{Q} \in \mathbb{R}^{n \times l}$ such that

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^* \mathbf{A}\| \approx \min_{\text{rank}(\mathbf{X}) \leq r} \|\mathbf{A} - \mathbf{X}\|$$

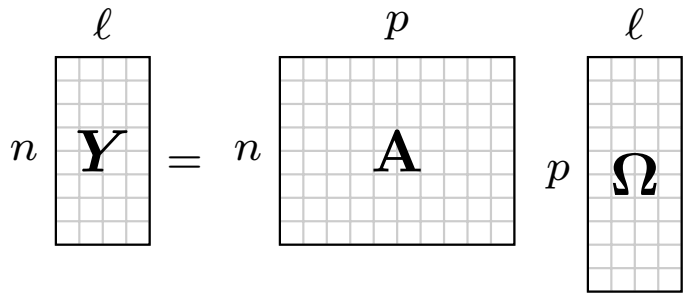
- r is the target rank,
- $l = r + s$ number of columns used
- s is the number of oversamples

Method: Obtain random vectors in the range of $\mathbf{A}$ by multiplying it with random vectors

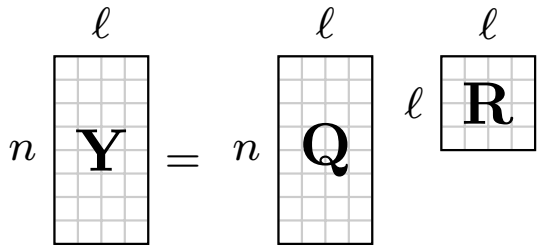
- From these vectors we can find an orthogonal basis $\mathbf{Q}$
- There exists a $\mathbf{Q} \in \mathbb{R}^{n \times r}$ which gives the optimum value of the above minimization problem (guess what it is !)
- But for a better approximation we oversample it: $\mathbf{Q} \in \mathbb{R}^{n \times (r+s)}$

Step-1: Finding a range

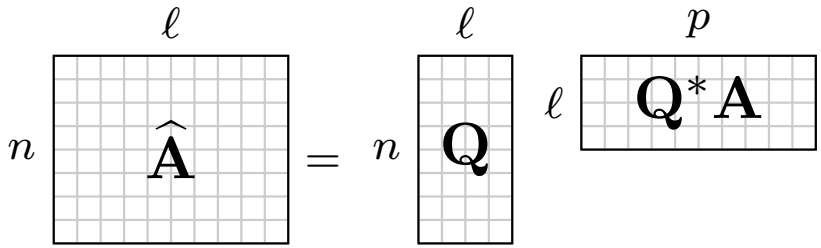
1. Multiply $\mathbf{A\Omega}$ for $\mathbf{\Omega}_{i,j} \sim \mathcal{N}(0,1)$, at cost $\mathcal{O}(np\ell)$ (or less)



2. Compute thin QR factorization of $\mathbf{Y}$, at a cost of $\mathcal{O}(n\ell^2)$ (e.g. with Gram-Schmidt)



3. Final multiply $\mathbf{Q}^*\mathbf{A}$, at cost $\mathcal{O}(np\ell)$



Random Sampling: Geometric Interpretation

- $A = [a_1, a_2, a_3, a_4]$
- $n = 3, p = 4$, rank is $r = 2$
- $Y = A\Omega$

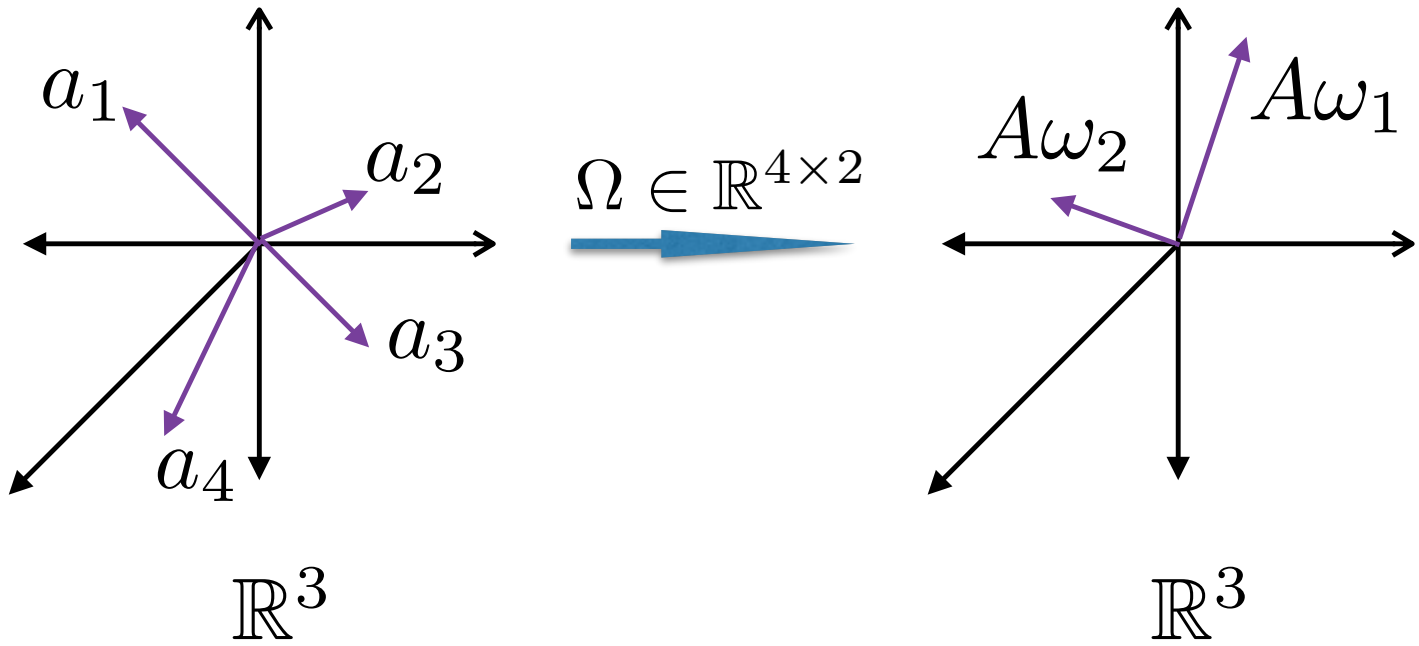


Figure: Random sampling can span the range

- Column selection would also work, but we need to be careful about how we select the columns. (next lecture)

Mathematical Intuition: What does randomness bring ?

Randomness

- $\Omega = [\omega_1, \omega_2, ..., \omega_r]$ has linearly independent columns.
- No linear combination of the columns can be in the null space of A.

Claim: Range(A) = Range$(Y = A\Omega)$

Claim: $A = QQ^*A$, when A is rank-r.

In practice we have

$$X = A + E$$

where A is best rank-r approximation to X. By random sampling, we aim to span the range of A with $X\omega_1, ..., X\omega_r$. However they're distorted by $E\omega_i$. That is the reason why we oversample and take $\ell = r + s$ columns.

Theoretical Guarentees

Best rank-r errors

Note the different optimal errors in spectral and Frobenius norms. Let σ_i be the i^{th} singular value of $\mathbf{A}$. Then

$$\sigma_{r+1} = \min_{\text{rank}(\mathbf{B}) \leq r} \|\mathbf{A} - \mathbf{B}\| \qquad \textit{vs.} \qquad \left(\sum_j \sigma_j^2\right)^{1/2} = \min_{\text{rank}(\mathbf{B}) \leq r} \|\mathbf{A} - \mathbf{B}\|_F$$

Theorem (expected Frobenius error [6])

Let $\widehat{\mathbf{A}}_{(l)}$ be the approximation with l columns obtained above. The average error decreases with the oversampling rate s. In particular for $r \geq 2, s \geq 2$ and $l = r + s \leq \min\{n, p\}$

$$\mathbb{E}\|\widehat{\mathbf{A}}_{(l)} - \mathbf{A}\|_F = \sqrt{1 + \frac{r}{s-1}} \left(\sum_{j>r} \sigma_j^2\right)^{1/2}$$

Theoretical Guarentees

Best rank-r errors

Note the different optimal errors in spectral and Frobenius norms. Let σ_i be the i^{th} singular value of $\mathbf{A}$. Then

$$\sigma_{r+1} = \min_{\text{rank}(\mathbf{B}) \leq r} \|\mathbf{A} - \mathbf{B}\| \qquad \textit{vs.} \qquad \left(\sum\nolimits_j \sigma_j^2\right)^{1/2} = \min_{\text{rank}(\mathbf{B}) \leq r} \|\mathbf{A} - \mathbf{B}\|_F$$

Theorem (expected Frobenius error [6])

Let $\widehat{\mathbf{A}}_{(l)}$ be the approximation with l columns obtained above. The average error decreases with the oversampling rate s. In particular for $r \geq 2, s \geq 2$ and $l = r + s \leq \min\{n, p\}$

$$\mathbb{E}\|\widehat{\mathbf{A}}_{(l)} - \mathbf{A}\|_F = \sqrt{1 + \frac{r}{s-1}} \left(\sum_{j>r} \sigma_j^2\right)^{1/2}$$

Theoretical Guarentees

Theorem (simple spectral bound [6])

Let $\mathbf{A} \in \mathbb{R}^{n \times p}$ with $n \geq p$ be the matrix that is randomly approximated as above. Let also $r \geq 2, s \geq 2$ and $l = r + s \leq \min\{n, p\}$. Then the following holds:

$$\mathbb{E}\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^* \mathbf{A}\| \leq \left(1 + \frac{4\sqrt{r+s}}{s-1}\sqrt{p}\right) \sigma_{r+1}$$

Theorem (deterministic spectral bound [6])

Furthermore, the following deterministic bounds also holds:

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^* \mathbf{A}\| \leq \left[1 + 9\sqrt{r+s} \cdot \min(n, p)\sqrt{p}\right] \sigma_{r+1}$$

with probability at least $1 - 3 \cdot s^{-s}$. under some mild assumptions on p.

- In practice an oversampling of $s = 5$ is sufficient

Theoretical Guarentees

Theorem (simple spectral bound [6])

Let $\mathbf{A} \in \mathbb{R}^{n \times p}$ with $n \geq p$ be the matrix that is randomly approximated as above. Let also $r \geq 2, s \geq 2$ and $l = r + s \leq \min\{n, p\}$. Then the following holds:

$$\mathbb{E}\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\| \leq \left(1 + \frac{4\sqrt{r+s}}{s-1}\sqrt{p}\right)\sigma_{r+1}$$

Theorem (deterministic spectral bound [6])

Furthermore, the following deterministic bounds also holds:

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\| \leq \left[1 + 9\sqrt{r+s} \cdot \min(n,p)\sqrt{p}\right]\sigma_{r+1}$$

with probability at least $1 - 3 \cdot s^{-s}$. under some mild assumptions on p.

- In practice an oversampling of $s = 5$ is sufficient

Structured Random Matrices

Motivation

- If we have a fast way of multiplying $\mathbf{Y} = \mathbf{A\Omega}$, then random projection method is attractive.
- Use of structured matrices such as Fourier or Hadamard allows a faster matrix multiplication (e.g. using FFT)
- If A has a fast spectral decay, then this approach below works as well as Gaussian matrices.

Theorem

If $\mathbf{\Omega}$ is a subsampled random Fourier transform matrix (SRFT) of dimensions $p \times \ell$ with $\ell \geq (r + \log n) \log r$, then

$$\|\mathbf{A} - \mathbf{QQ}^* \mathbf{A}\|_2 \leq \sqrt{1 + \frac{7p}{\ell}} \cdot \sigma_{r+1}$$

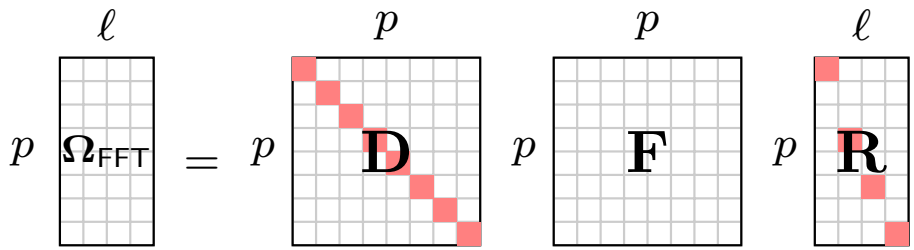
except with probability $\mathcal{O}(r^{-1})$. [6]

- We need more oversampling for a decent performance : $s = 20$

Structured Random Matrices

A subsampled random Fourier transform matrix is a $p \times l$ matrix of the form

$$\mathbf{\Omega}_{\text{FFT}} = \sqrt{\frac{p}{l}}\mathbf{DFR}$$



where

- $\mathbf{D} \in \mathbb{R}^{p \times p}$ is diagonal matrix with entries that are independent RVs uniformly distributed on the complex unit circle
- $\mathbf{F}$ is $p \times p$ is the unitary DFT Matrix
- $\mathbf{R}$ is a $p \times l$ matrix whose l columns are drawn uniformly from the identity matrix without replacement.

Using Fast Fourier Transform (FFT), cost of $\mathbf{Y} = \mathbf{A\Omega}$ reduces to $\mathcal{O}(np \log \ell)$!
(compare with direct method that costs $\mathcal{O}(np\ell)$)

Step-2: Forming the decomposition

- So far we have obtained:
$$\mathbf{A} \approx \mathbf{Q}(\mathbf{Q}^* \mathbf{A})$$
- Multiplying $(\mathbf{Q}^* \mathbf{A})$ costs $\mathcal{O}(npl)$
- This is the bottleneck and could be avoided and reduced to $\mathcal{O}(l^2(n+p))$ using row extraction method (next lecture) but at the expense of worse error bound.
- For the moment we work with the product $\mathbf{Q}^* \mathbf{A}$ and decompose it.
- Let $\mathbf{B} = \mathbf{Q}$ and $\mathbf{C} = \mathbf{Q}^* \mathbf{A}$ in our low rank approximation $\mathbf{A} \approx \mathbf{BC}$
- Indeed this is the partial QR decomposition using Randomized Linear Algebra
- We now form partial singular value decomposition out of this

Classical Methods for partial Singular Value Decomposition

via Full SVD

- The full SVD of a $n \times p$ matrix is computed and truncated.
- It costs $\mathcal{O}(np \min\{n, p\})$.
- Stable but very expensive.

Krylov Subspace methods

- The idea is to choose a random initial vector ω and apply successively a Hermitian operator $\mathbf{H}$ to form the subspace

$$\mathcal{K}_r(\mathbf{H}, \omega) = span\{b, \mathbf{H}b, \mathbf{H}^2 b, ..., \mathbf{H}^{r-1} b\}$$

  to find of the eigenvectors of $\mathbf{H}$ (or first few of them)
- One of the best methods in numerical linear algebra such as Arnoldi and Lanczos algorithms are based on this idea [5]
- It might vary but typically costs $\mathcal{O}(rnp + r^2(n + p))$
- It requires $\mathcal{O}(k)$ passes over the data

Classical Methods for partial Singular Value Decomposition

via Full SVD

- The full SVD of a $n \times p$ matrix is computed and truncated.
- It costs $\mathcal{O}(np \min\{n, p\})$.
- Stable but very expensive.

Krylov Subspace methods

- The idea is to choose a random initial vector ω and apply successively a Hermitian operator $\mathbf{H}$ to form the subspace

$$\mathcal{K}_r(\mathbf{H}, \omega) = span\{b, \mathbf{H}b, \mathbf{H}^2 b, ..., \mathbf{H}^{r-1} b\}$$

  to find of the eigenvectors of $\mathbf{H}$ (or first few of them)
- One of the best methods in numerical linear algebra such as Arnoldi and Lanczos algorithms are based on this idea [5]
- It might vary but typically costs $\mathcal{O}(rnp + r^2(n + p))$
- It requires $\mathcal{O}(k)$ passes over the data

Classical Methods for partial Singular Value Decomposition

Computing a partial SVD using QR

- Use Businger-Golub or strong rank revealing QR algorithm to form $\mathbf{A} \approx \mathbf{QR}$ where $\mathbf{Q} \in \mathbb{R}^{n \times \ell}$ and $\mathbf{R} \in \mathbb{R}^{\ell \times p}$ [5]
- Then transform this to SVD as above.
- This also costs $\mathcal{O}(npr)$ but more robust compared to Krylov methods.
- It requires $\mathcal{O}(k)$ passes over the data

A comparison

- Classical techniques require at least $\mathcal{O}(npr)$ whereas randomized algorithms can be implemented with $\mathcal{O}(np\log(l) + l^2(n+p))$.
- In the slow memory environment, the figure of merit is not the flop counts, but number of passes over the data.
- All these classical techniques require many passes over the matrix and whereas randomized algorithms require a constant number of passes over the data. [6]
- Randomized methods are highy parallelizable, because $\mathbf{Y} = \mathbf{A}\Omega$ can be efficiently implemented in modern architectures: GPUs, distributed computing, multi-core processors.

Classical Methods for partial Singular Value Decomposition

Computing a partial SVD using QR

- Use Businger-Golub or strong rank revealing QR algorithm to form $\mathbf{A} \approx \mathbf{QR}$ where $\mathbf{Q} \in \mathbb{R}^{n \times \ell}$ and $\mathbf{R} \in \mathbb{R}^{\ell \times p}$ [5]
- Then transform this to SVD as above.
- This also costs $\mathcal{O}(npr)$ but more robust compared to Krylov methods.
- It requires $\mathcal{O}(k)$ passes over the data

A comparison

- Classical techniques require at least $\mathcal{O}(npr)$ whereas randomized algorithms can be implemented with $\mathcal{O}(np\log(l) + l^2(n+p))$.
- In the slow memory environment, the figure of merit is not the flop counts, but number of passes over the data.
- All these classical techniques require many passes over the matrix and whereas randomized algorithms require a constant number of passes over the data. [6]
- Randomized methods are highy parallelizable, because $\mathbf{Y} = \mathbf{A\Omega}$ can be efficiently implemented in modern architectures: GPUs, distributed computing, multi-core processors.

References I

[1] Volkan Cevher, Steffen Becker, and Martin Schmidt.
Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics.
Signal Processing Magazine, IEEE, 31(5):32–43, 2014.

[2] Jack Dongarra and Francis Sullivan.
Guest editors? introduction: The top 10 algorithms.
Computing in Science & Engineering, 2(1):22–23, 2000.

[3] Olivier Fercoq and Pascal Bianchi.
A coordinate descent primal-dual algorithm with large step size and possibly non separable functions.
http://arxiv.org/abs/1508.04625, Aug. 2015.

[4] Olivier Fercoq and Peter Richatárik.
Accelerated, parallel and proximal coordinate descent.
SIAM J. Optim., 25:1997–2023, 2016.

[5] Gene H Golub and Charles F Van Loan.
Matrix computations, volume 3.
JHU Press, 2012.

References II

[6] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp.
Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions.
SIAM review, 53(2):217–288, 2011.

[7] Peter Richatárik and Martin Takac.
Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function.
Math. Program., 144:1–38, 2014.

[8] Stephen J Wright.
Coordinates descent algorithms.
Math. Program., 151:3–34, 2015.