

LECTURE 7: COORDINATE DESCENT METHODS (CONT.) AND RANDOMIZED LINEAR ALGEBRA

1 Coordinate descent methods for composite functions

In the last lecture, we have learned the variants of coordinate descent method for unconstrained smooth convex minimization problems. As the underlying convex problem has instead composite objective in many real world applications in convex optimization and machine learning, it is natural ask whether coordinate descent methods can be used in this setting. Before investigating this question in details, let us recall the following basic template of the composite convex minimization problem:

$$F^* = \min_{\mathbf{x} \in \mathbb{R}^p} \{F(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x})\}. \quad (1)$$

Here, we assume that both f and g are proper, closed and convex functions, and the solution set $\mathcal{S}^* := \{\mathbf{x}^* \in \text{dom}(F) : F(\mathbf{x}^*) = F^*\}$ is nonempty. Typically in the applications, smooth term f (smooth means that ∇f is Lipschitz continuous) represents the data fidelity and the nonsmooth term g is a regularizer which encourages some desired structures and/or constraints in the solution.

While these problems can be solved using the subgradient methods, considering the subgradient of overall nonsmooth objective F , these methods are inefficient in terms of iteration complexity. On the other hand, if the proximal operator of the nonsmooth part,

$$\text{prox}_g(\mathbf{x}) \equiv \arg \min_{\mathbf{y} \in \mathbb{R}^p} \left\{ g(\mathbf{y}) + \frac{1}{2} \|\mathbf{y} - \mathbf{x}\|_2^2 \right\}, \quad (2)$$

is cheap to compute, then solving (1) is as efficient as solving an unconstrained smooth convex minimization problem in terms of complexity. Fortunately, proximal operator of many well-known convex regularization functions can be computed either analytically or very efficiently. Table 1 provides a non-exhaustive list of proximal operators of common regularization functions and their computational complexities.

Name	Function	Proximal operator	Complexity
ℓ_1 -norm	$f(\mathbf{x}) := \ \mathbf{x}\ _1$	$\text{prox}_{\lambda f}(\mathbf{x}) = \text{sign}(\mathbf{x}) \otimes [\mathbf{x} - \lambda]_+$	$\mathcal{O}(p)$
ℓ_2 -norm	$f(\mathbf{x}) := \ \mathbf{x}\ _2$	$\text{prox}_{\lambda f}(\mathbf{x}) = [1 - \lambda/\ \mathbf{x}\ _2]_+ \mathbf{x}$	$\mathcal{O}(p)$
Support function	$f(\mathbf{x}) := \max_{\mathbf{y} \in \mathcal{C}} \mathbf{x}^T \mathbf{y}$	$\text{prox}_{\lambda f}(\mathbf{x}) = \mathbf{x} - \lambda \pi_{\mathcal{C}}(\mathbf{x})$	$\mathcal{O}(p)$
Box indicator	$f(\mathbf{x}) := \delta_{[\mathbf{a}, \mathbf{b}]}(\mathbf{x})$	$\text{prox}_{\lambda f}(\mathbf{x}) = \pi_{[\mathbf{a}, \mathbf{b}]}(\mathbf{x})$	$\mathcal{O}(p)$
Positive semidefinite cone indicator	$f(\mathbf{X}) := \delta_{\mathbb{S}_+^p}(\mathbf{X})$	$\text{prox}_{\lambda f}(\mathbf{X}) = \mathbf{U}[\Sigma]_+ \mathbf{U}^T$, where $\mathbf{X} = \mathbf{U}\Sigma\mathbf{U}^T$	$\mathcal{O}(p^3)$
Hyperplane indicator	$f(\mathbf{x}) := \delta_{\mathcal{X}}(\mathbf{x}), \mathcal{X} := \{\mathbf{x} : \mathbf{a}^T \mathbf{x} = b\}$	$\text{prox}_{\lambda f}(\mathbf{x}) = \pi_{\mathcal{X}}(\mathbf{x}) = \mathbf{x} + \left(\frac{b - \mathbf{a}^T \mathbf{x}}{\ \mathbf{a}\ _2}\right) \mathbf{a}$	$\mathcal{O}(p)$
Simplex indicator	$f(\mathbf{x}) = \delta_{\mathcal{X}}(\mathbf{x}), \mathcal{X} := \{\mathbf{x} : \mathbf{x} \geq 0, \mathbf{1}^T \mathbf{x} = 1\}$	$\text{prox}_{\lambda f}(\mathbf{x}) = (\mathbf{x} - \nu \mathbf{1})$ for some $\nu \in \mathbb{R}$, which can be efficiently calculated	$\tilde{\mathcal{O}}(p)$
Convex quadratic	$f(\mathbf{x}) := (1/2)\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{q}^T \mathbf{x}$	$\text{prox}_{\lambda f}(\mathbf{x}) = (\lambda \mathbf{I} + \mathbf{Q})^{-1} \mathbf{x}$	$\mathcal{O}(p \log p) \rightarrow \mathcal{O}(p^3)$
Square ℓ_2 -norm	$f(\mathbf{x}) := (1/2)\ \mathbf{x}\ _2^2$	$\text{prox}_{\lambda f}(\mathbf{x}) = (1/(1 + \lambda))\mathbf{x}$	$\mathcal{O}(p)$
log-function	$f(\mathbf{x}) := -\log(x)$	$\text{prox}_{\lambda f}(x) = ((x^2 + 4\lambda)^{1/2} + x)/2$	$\mathcal{O}(1)$
log det-function	$f(\mathbf{x}) := -\log \det(\mathbf{X})$	$\text{prox}_{\lambda f}(\mathbf{X})$ is the log-function prox applied to the individual eigenvalues of \mathbf{X}	$\mathcal{O}(p^3)$

Table 1: Non-exhaustive list of proximal operators of common regularization functions with computational complexities. Here: $[\mathbf{x}]_+ := \max\{0, \mathbf{x}\}$ and $\delta_{\mathcal{X}}$ is the indicator function of the convex set \mathcal{X} , sign is the sign function, \mathbb{S}_+^p is the cone of symmetric positive semidefinite matrices. For more functions, see [2, 9].

Recall that in the previous lecture, we underlined some important features of coordinate descent methods in comparison with the stochastic gradient methods for unconstrained smooth convex minimization problems. As you would remember, the most salient feature is the decay of the variance between the gradient estimates and the exact gradient in coordinate descent methods. Once again, recall that this variance was the main reason for the choice of diminishing learning rates in the stochastic gradient descent methods.

Now we will investigate a similar property and compare the coordinate descent methods for smooth minimization and the composite minimization templates in convex optimization. We will pursue this investigation by considering a toy numerical example.

First, let us consider the example given in Figure 1, the smooth minimization problem with the objective function $f(\mathbf{x}) = \|\mathbf{x}\|_2^2$. Clearly, \mathbf{x}^* is a global optimum if and only if $f(\mathbf{x})$ is minimized along each coordinate axis at $\mathbf{x} = \mathbf{x}^*$ (Recall that, on the other hand, an unbiased estimate of the gradient is not necessarily the zero vector at the global optimum).

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

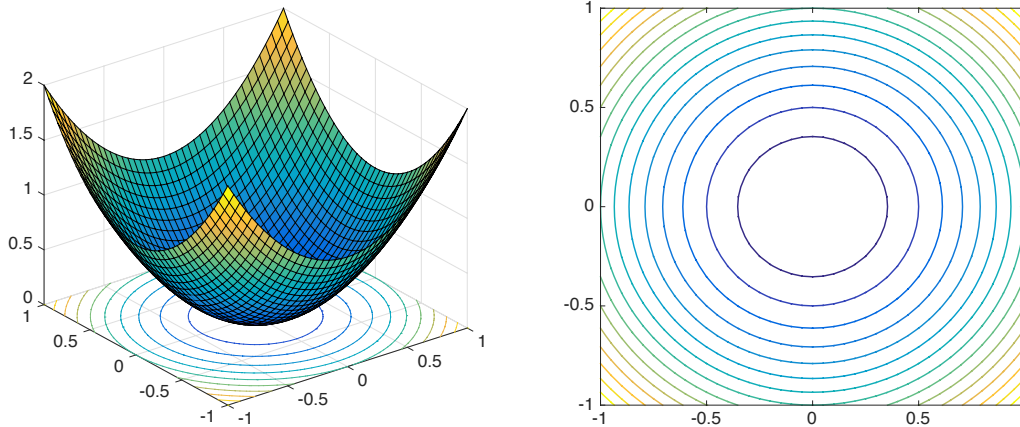


Figure 1: Smooth objective function: $f(\mathbf{x}) = \|\mathbf{x}\|_2^2$

Unfortunately, the same equivalence does not hold in the composite convex minimization formulation. To see this, let us consider the example given in Figure 2, which corresponds to the minimization of $F(\mathbf{x}) = \|\mathbf{x}\|_2^2 + |x_1 - x_2|$. As a counter example, you can easily verify that the point $(0.5, 0.5)$ is minimized along both coordinate axis, while it is not an optimum point.

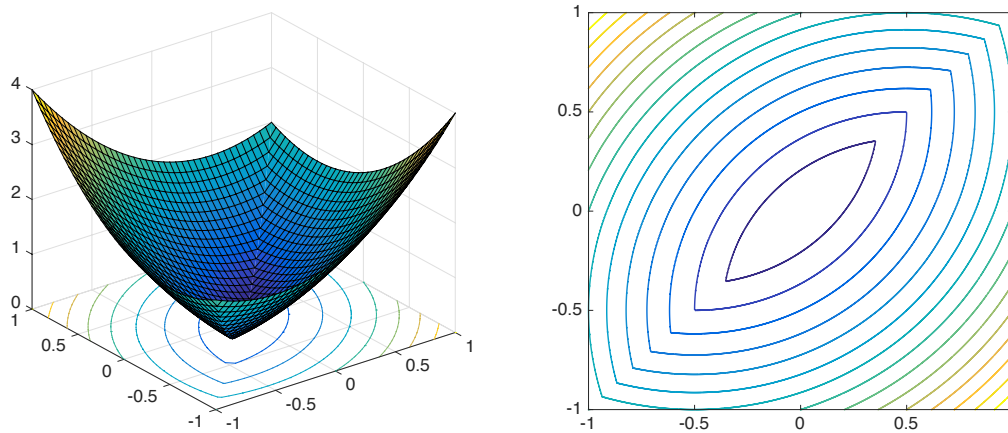


Figure 2: Composite objective function: $F(\mathbf{x}) = \|\mathbf{x}\|_2^2 + |x_1 - x_2|$.

These examples point out that the bare coordinate descent method does not work for composite convex minimization problems generally speaking. However, we can overcome this obstacle by adopting the following mild assumption: **The nonsmooth part g is (block-)coordinatewise separable, i.e., $g(\mathbf{x}) = \sum_{i=1}^p g_i(x_i)$.** Then, F is minimized along each coordinate axis at $\mathbf{x} = \mathbf{x}^*$, if and only if \mathbf{x}^* is the global optimum. Figure 3 shows a simple example for this case where $F(\mathbf{x}) = \|\mathbf{x}\|_2^2 + \|\mathbf{x}\|_1$.

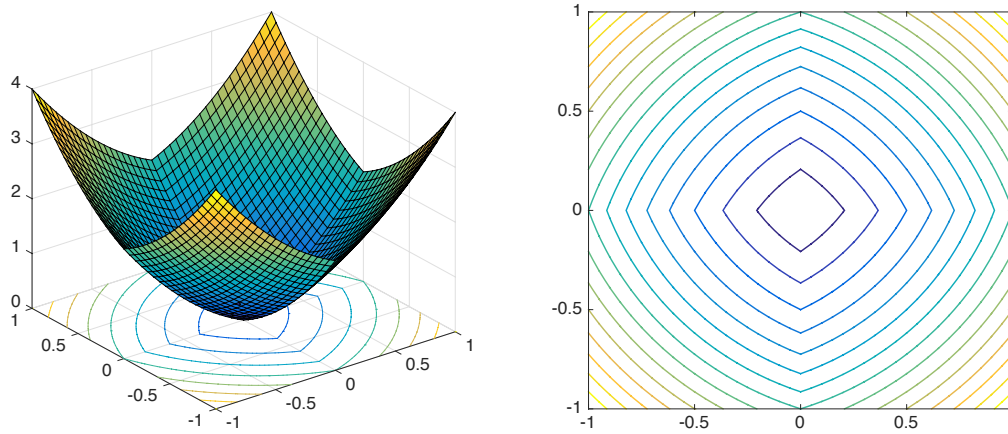


Figure 3: Composite objective with separable nonsmooth part: $F(\mathbf{x}) = \|\mathbf{x}\|_2^2 + \|\mathbf{x}\|_1$.

To sum up, in this section we consider coordinate descent methods for composite convex minimization problem (1), where we assume that

1. g is separable:

$$g(\mathbf{x}) = \sum_{i=1}^p g_i(x_i), \quad \text{where } g_i: \mathbb{R} \rightarrow \mathbb{R} \text{ for all } i, \text{ e.g.,}$$

2. g is block-separable: $p \times p$ identity matrix can be partitioned into column submatrices $U_i, i = 1, \dots, s$ such that

$$g(\mathbf{x}) = \sum_{i=1}^s g_i(U_i^T \mathbf{x}).$$

Examples include unconstrained smooth minimization (with the choice of $g(\mathbf{x}) = \text{constant}$), box constrained minimization ($g(\mathbf{x}) = \sum_{i=1}^s \mathbb{1}_{[a_i, b_i]}(x_i)$), ℓ_q norm regularization: $g(\mathbf{x}) = \|\mathbf{x}\|_q^q$ where $q \geq 1$. Block-separable examples include group-sparse regularizers. Some of the more sophisticated examples of great interest are the followings:

- LASSO formulation (least squares with ℓ_1 norm regularization):

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

- Support vector machine (SVM) with squared hinge loss:

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

- SVM: dual form with bias term:

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

- Logistic regression with ℓ_q norm regularization:

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

- Semi-supervised learning with Tikhonov regularization

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

- Relaxed linear programming:

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

Finally, before the discussion of the theoretical aspects of coordinate descent methods in composite convex minimization template, let us recall the necessary notation for (block) partitioning that we have used in lecture 6. This notation is summarized in Figure 4.

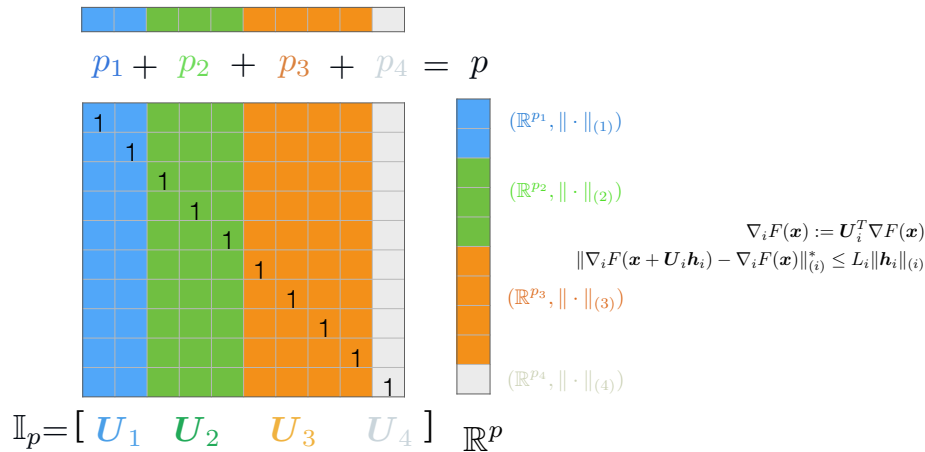


Figure 4: Notation for block separability (from lecture 6).

1.1 Randomized proximal coordinate descent (PCD)

We start with the simplest variant of the coordinate descent methods for composite convex minimization formulation. The pseudocode of this algorithm is given below:

Randomized proximal coordinate descent (PCD)
<ol style="list-style-type: none"> 1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$ and $(\gamma_k)_{k \in \mathbb{N}} \in]0, +\infty[^\mathbb{N}$. 2. For $k = 0, 1, \dots$ perform: <ol style="list-style-type: none"> 2a. Pick $i_k \in \{1, \dots, s\}$ uniformly at random. 2b. Update coordinate i_k: $\mathbf{x}_{i_k}^{k+1} = \arg \min_{\mathbf{v} \in \mathbb{R}^{p_{i_k}}} g_{i_k}(\mathbf{v}) + \langle \mathbf{v}, \nabla_{i_k} f(\mathbf{x}^k) \rangle + \frac{1}{2\alpha_k} \ \mathbf{v} - \mathbf{x}_{i_k}^k\ _{(i_k)}^2.$

Remark: If $\|\cdot\|_{(i_k)} = \|\cdot\|_2$, then the update simplifies to $\mathbf{x}_{i_k}^{k+1} = \text{prox}_{\alpha_k g_{i_k}}(\mathbf{x}_{i_k}^k - \alpha_k \nabla_{i_k} f(\mathbf{x}^k))$.

Theorem 1. [Convergence without strong convexity [10]]

Choose a target confidence $0 < \rho < 1$. Suppose that ∇f_i is Lipschitz continuous with respect to some norm $\|\cdot\|_{(i)}$ for $i = 1, 2, \dots, s$, that is

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

For any target accuracy $\varepsilon < F(\mathbf{x}^0) - F^*$,

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

where

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

Theorem 2. [Convergence with strong convexity [12]]

Suppose f is a strongly convex function with convexity constant μ . Also suppose that ∇f_i is Lipschitz continuous with respect to some norm $\|\cdot\|_{(i)}$ for $i = 1, 2, \dots, s$, that is

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

Let us set $\alpha_k = 1/L_{\max}$ for all k , where $L_{\max} = \max_i L_i$, then

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

We provide empirical performance of PCD on the following simple problem formulation (LASSO) in Figure 5:

$$\min_{\mathbf{x} \in \mathbb{R}^p} \left\{ f(\mathbf{x}) := \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \rho \|\mathbf{x}\|_1 \right\}. \quad (3)$$

We consider a synthetic problem setup, where each entry of the measurement matrix \mathbf{A} follows the standard Gaussian distribution $\mathcal{N}(0, 1)$. Problem dimensions n and p are 1000 and 500 respectively. We generate a 50-sparse true signal \mathbf{x}^\dagger , where the non-zero entries are randomly chosen with Gaussian distribution with 0 mean, and the signal is normalized to $\|\mathbf{x}^\dagger\|_1 = 1$. We generate noisy measurements $\mathbf{b} := \mathbf{A}\mathbf{x}^\dagger + \mathbf{w}$, where \mathbf{w} is Gaussian white noise and the SNR is 30dB. Regularization constant λ is chosen as 10^{-2} . Coordinates are chosen from uniform distribution. Empirical rate is linear in accordance with the theorem 2.

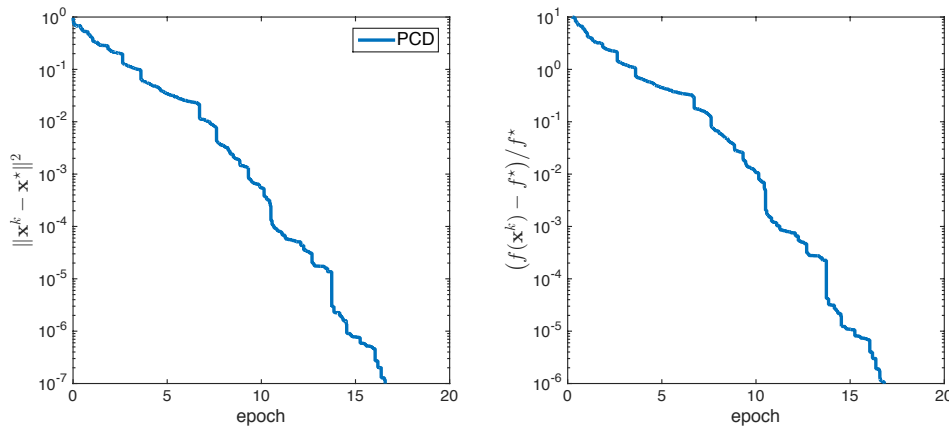


Figure 5: Empirical performance of PCD on LASSO problem.

Recall that the optimal rates for the stochastic gradient methods are $O(1/k)$ and $O(1/\sqrt{k})$, with and without strong convexity assumptions respectively. Instead, coordinate descent methods reach the rate of first order gradient methods' optimal rates, and converge linearly with strong convexity, and with $O(1/k)$ rate without strong convexity assumptions. In the next section, we show that the latter can be improved to $O(1/k^2)$ by an acceleration step.

1.2 Accelerated parallel proximal coordinate descent method (APPROX)

In this section, we consider the accelerated variant of the proximal coordinate gradient methods. We adopt the following notation

- **Uniform block sampling:** $\mathbb{P}(i \in S) = \mathbb{P}(j \in S)$ for all $i, j \in \{1, 2, \dots, s\}$,
- $\tau = \mathbb{E}[|S|]$.

Let us assume that there exist positive real numbers $\sigma = (\sigma_1, \dots, \sigma_s) \in \mathbb{R}_+^s$ that satisfy $\forall \mathbf{x}, \mathbf{h} \in \mathbb{R}^p$:

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

where $\|\mathbf{x}\|_{\sigma}^2 := \sum_{i=1}^s \sigma_i \|\mathbf{x}_i\|_{(i)}^2$, then the pseudocode of this variant follows below:

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 $k = 0, 1, \dots$ 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:

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

- 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. $\alpha_{k+1} = \frac{1}{2}(\sqrt{\alpha_k^4 + 4\alpha_k^2 - \alpha_k^2})$.

Theorem 3. [Mean convergence of APPROX [5]]

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

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

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

Remark: Note that APPROX applies full dimensional vector updates on \mathbf{y} . As you would remember, we have discussed in lecture 6 that one of the main advantages of the coordinate descent methods is one dimensional updates on the decision variables. Clearly, full dimensional updates of APPROX limits the efficiency compared to PCD. However, under mild conditions that is satisfied in most of the machine learning applications, APPROX possess an efficient implementation which avoids high dimensional updates on \mathbf{y} . See [5] and [8] for details.

2 Coordinate descent primal-dual algorithm

2.1 Composite minimization problem

In this section, we consider the following composite minimization problem:

$$F^* := \min_{\mathbf{x} \in \mathbb{R}^p} \{f(\mathbf{x}) + g(\mathbf{x}) + h(\mathbf{A}\mathbf{x})\}, \quad (4)$$

where g and h are convex, f is convex and differentiable, and $\mathbf{A} \in \mathbb{R}^{q \times p}$. This problem can be transformed to finding saddle points of the Lagrangian function

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

where $h^*: \mathbf{y} \mapsto \sup_{\mathbf{z}} \langle \mathbf{y}, \mathbf{z} \rangle - h(\mathbf{z})$ is the Fenchel-Legendre transform of h . The initial variable \mathbf{x} is called primal variable and the new variable \mathbf{y} is called dual variable which plays a similar role as Lagrange multiplier in the setting of constrained optimization. It is known that \mathcal{L} is convex-concave and hence tools from convex optimization could be applied. A such method was proposed in [11] to solve (4).

Let us give some practical examples that have this problem formulation:

Example 1. Total variation + ℓ_1 regularized least squares regression

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

where

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

Example 2. Dual SVM

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

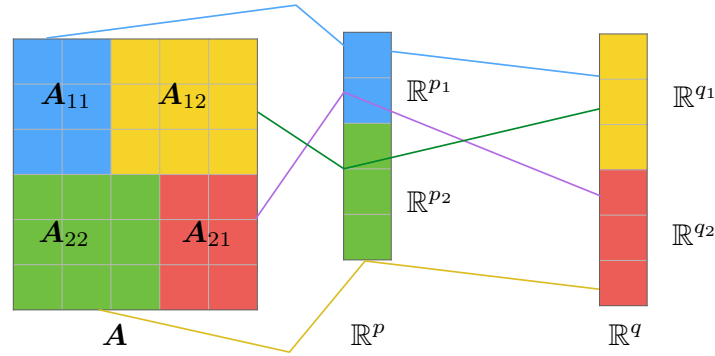
where $\mathbf{b} \odot \mathbf{x}$ is the component-wise multiplications of two vectors \mathbf{b} and \mathbf{x} and

$$\mathbf{b}^\perp = \{\mathbf{x} \in \mathbb{R}^p \text{ s.t. } \mathbf{b}^T \mathbf{x} = 0\}.$$

2.2 The algorithm

Recent trends in optimization intend to reduce the computational cost by explore the separable structure of (4), e.g., blocks of coordinates. In this approach, optimization problems in high dimensions can be solved in smaller dimensions. We therefore suppose that we can decompose the spaces into blocks of coordinates and the involved matrix into block-matrices as follow:

$$\mathbb{R}^p = \mathbb{R}^{p_1} \times \dots \times \mathbb{R}^{p_s}, \quad \mathbb{R}^q = \mathbb{R}^{q_1} \times \dots \times \mathbb{R}^{q_r}, \quad \mathbf{A} \in \mathbb{R}^{q \times p} = [\mathbf{A}_{ij}]_{i,j}, \quad \mathbf{A}_{ij} \in \mathbb{R}^{q_j \times p_i}.$$



Before presenting the algorithm, we introduce the following notation: $\mathbb{I}(j)$ are the indices of nonzero q_j -rows matrices and m_j its cardinal, $\mathbb{J}(i)$ are the indices of nonzero p_i -columns matrices. The proximal operator of a convex function $g: \mathbb{R}^p \rightarrow (0, +\infty]$ of coefficient $\tau = (\tau_i)_{1 \leq i \leq s}$, is defined as follow

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

In particular, the τ -projection onto a convex set \mathbf{C} is

$$P_{\mathbf{C}}^{\tau}: \mathbf{x} \mapsto \arg \min_{\mathbf{y}} \|\mathbf{x} - \mathbf{y}\|_{\tau}^2.$$

The algorithm below combines the block-coordinate descent approach with method in [11].

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 $\begin{cases} (\forall i \in \{1, \dots, s\}) & \mathbf{w}_i^0 = \sum_{j \in \mathbb{I}(i)} \mathbf{A}_{ji}^T \mathbf{y}_j^0(i). \\ (\forall j \in \{1, \dots, t\}) & \mathbf{z}_j^0 = (1/m_j) \sum_{i \in \mathbb{I}(j)} \mathbf{y}_j^0(i). \end{cases}$
2.	For $k = 0, 1, \dots$ perform: <ul style="list-style-type: none"> 2a. Choose $i_k \in \{1, \dots, s\}$ at random and uniformly. 2b. Compute: $\begin{cases} \bar{\mathbf{y}}^{k+1} = \text{prox}_{\sigma_{h^*}}(\mathbf{z}^k + \sigma \odot (\mathbf{A}\mathbf{x}^k)) \\ \bar{\mathbf{x}}^{k+1} = \text{prox}_{\tau g}(\mathbf{x}^k - \tau \odot (\nabla f(\mathbf{x}^k) + 2\mathbf{A}^T \bar{\mathbf{y}}^{k+1} - \mathbf{w}^k)). \end{cases}$ 2c. Update: <ul style="list-style-type: none"> 2c1. For $i = i_{k+1}$ and for each $j \in \mathbb{I}(i_{k+1})$: $\begin{cases} \mathbf{x}_i^{k+1} = \bar{\mathbf{x}}_i^{k+1}, & \mathbf{y}_j^{k+1}(i) = \bar{\mathbf{y}}_j^{k+1}(i) \\ \mathbf{w}_i^{k+1} = \mathbf{w}_i^k + \sum_{j \in \mathbb{I}(i)} \mathbf{A}_{ji}^* (\mathbf{y}_j^{k+1}(i) - \mathbf{y}_j^k(i)) \\ \mathbf{z}_j^{k+1} = \mathbf{z}_j^k + \frac{1}{m_j} (\mathbf{y}_j^{k+1}(i) - \mathbf{y}_j^k(i)). \end{cases}$ 2c2. Otherwise: $\mathbf{x}_i^{k+1} = \bar{\mathbf{x}}_i^{k+1}$, $\mathbf{w}_i^{k+1} = \mathbf{w}_i^k$, $\mathbf{z}_j^{k+1} = \mathbf{z}_j^k$, $\mathbf{y}_j^{k+1}(i) = \mathbf{y}_j^k(i)$.

Let us write down the algorithm for dual SVM:

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

In this case $\mathbb{R}^p = \mathbb{R} \times \dots \times \mathbb{R}$ and

$$\mathbf{A} = \mathbb{I}_p = \begin{pmatrix} \boxed{1} & \boxed{0} & \dots & \boxed{0} \\ \boxed{0} & \boxed{1} & \dots & \boxed{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boxed{0} & \boxed{0} & \dots & \boxed{1} \end{pmatrix},$$

that is $\mathbf{A}_{ij} = \delta_{ij}$. Hence, $\mathbb{J}(i) = \{i\}$ and $\mathbb{I}(j) = \{j\}$ and since m_j is the number of nonzero elements in j th column, $m_j = 1$. The above primal-dual block coordinate descent algorithm is given for this specific example can be written down as follow.

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

$$\begin{cases} \bar{\mathbf{y}}^{k+1} = \text{prox}_{\sigma_{h^*}}(\mathbf{y}^k + \sigma \odot (\mathbf{A}\mathbf{x}^k)) \\ \bar{\mathbf{x}}^{k+1} = \text{prox}_{\tau g}(\mathbf{x}^k - \tau \odot (\nabla f(\mathbf{x}^k) + \mathbf{A}^T(2\bar{\mathbf{y}}^{k+1} - \mathbf{y}^k))). \end{cases}$$
 - 2c. Update:

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

The gradient and the proximal operators can be computed explicitly as:

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

The following theorem states that the iterative sequence converges to a solution but at the moment no result on convergence's rate is available.

Theorem 4 ([4]). Let $[\mathbf{U}_1, \dots, \mathbf{U}_s]$ be block decomposition of identity matrix \mathbb{I}_p and suppose that for every $i \in \{1, \dots, s\}$:

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

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

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

Then

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

3 Randomized linear algebra

3.1 Introduction

20th century witnessed emergence of many powerful algorithms and they became integral part of our lives. Below is a list of these algorithms compiled by [3]:

- 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

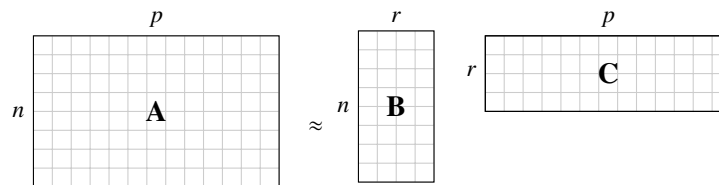
Among them some pertain to linear algebra and one of them, the decompositional approach to matrix computations, is a general framework rather than being a specific algorithm. This approach basically foresees the use of matrix decompositions for the solution of different linear algebra problems. For example if the problem at hand is solving a system of linear equations with a Hermitian positive matrix, we'd better use Cholesky decomposition that will allow us to find the unknown efficiently by simply doing forward and backward substitutions. If the problem is a low rank approximation of a matrix, then singular value decomposition (SVD) provides us the solution.

Once we have a decomposition of the matrix, we can reuse it efficiently for our purpose, whether it is solving a system of equations or finding the low-rank approximation. However these decompositions need to be implemented efficiently. There are many software packages that provide us efficient decompositions, but the computational complexity depends on the problem size and at high dimensions this is a bottleneck. For example both SVD and QR decompositions have $O(np \min\{n, p\})$ complexity.

This super-linear dependence on the data size is a challenge in the era of Big Data. On the other hand the classical algorithms for these compositions are endowed with the ability to provide high accuracy solutions. However in the real life problems the data is usually noisy therefore one can think of sacrificing accuracy for speed-up of the computation.

It turns out the one answer to this challenge is the recently investigated randomized linear algebra which proposes using a randomly found low rank approximation to a matrix that is given in a decomposition form. Before going into the description of this method, we give two examples to motivate.

Example-1: Matrix-vector multiplication



Let us take the matrix approximation given above. The advantages of this approximative decomposition are two-fold in matrix-vector multiplication:

- Faster computation: $O(r(n + p))$ instead of $O(np)$
- Lower memory usage: $O(r(n + p))$ instead of $O(np)$

In the regime $r \ll p$, this is a significant improvement at the cost of some approximation error.

Example-2: Robust principal component analysis (RPCA)

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

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

which requires computation of the proximal operator of the nuclear norm

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

We 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.

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

To compute the proximity operator for the nuclear norm in robust PCA for video background subtraction, we try both classical Lanczos-based SVD using PROACK software and randomized factorization. The matrix to be decomposed has dimensions 61440×17884 (8.1 GB) and is taken from a video sequence. In Figure 6 we also see that randomized approach scales much better for parallel computation.

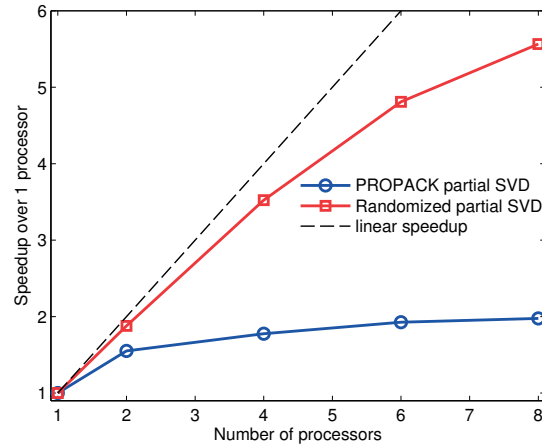


Figure 6: Computing the top 5 singular vectors of a 10^9 entry matrix using varying number of computer cores [1]

3.2 Randomized matrix decompositions

We have seen above some advantages of having an approximative matrix decomposition. Now let's see how we obtain this kind of approximations via randomization. Basically the procedure can be analyzed in two steps:

Step-1: Finding a range

- Apply a randomized algorithm to find an orthogonal low-dimensional basis $\mathbf{Q} \in \mathbb{R}^{n \times \ell}$ with $\ell \ll p$ 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

3.2.1 Finding a range

Given $\mathbf{A} \in \mathbb{R}^{n \times p}$, we're interested in finding $\mathbf{Q} \in \mathbb{R}^{n \times \ell}$ such that

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

where r is the target rank, $\ell = r + s$ number of columns used and s is the number of oversamples. We do this by obtaining random vectors in the range of \mathbf{A} by multiplying it with random vectors. This step is parallelizable if needed, i.e. one can use different cores for multiplying with different vectors and the merge them. From these vectors we can find an orthogonal basis \mathbf{Q} . Eckart-Young-Mirsky theorem says that there exists a $\mathbf{Q} \in \mathbb{R}^{n \times r}$ which gives the optimum solution of the above minimization problem and it is found by (partial) SVD. But since we cannot afford SVD at high dimension (remember this is the basic motivation for the randomized approximations), it is expected that we have a suboptimal solution: Oversampling $\ell = r + s$ columns.

For a better visualization of the procedure, the steps are listed below:

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

$$\begin{array}{c} \ell \\ \boxed{\mathbf{Y}} \end{array} = \begin{array}{c} p \\ \boxed{\mathbf{A}} \end{array} \begin{array}{c} \ell \\ \boxed{\mathbf{\Omega}} \end{array}$$

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

$$n \begin{matrix} \ell \\ \mathbf{Y} \end{matrix} = n \begin{matrix} \ell \\ \mathbf{Q} \end{matrix} \ell \begin{matrix} \ell \\ \mathbf{R} \end{matrix}$$

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

$$n \begin{matrix} \ell \\ \widehat{\mathbf{A}} \end{matrix} = n \begin{matrix} \ell \\ \mathbf{Q} \end{matrix} \ell \begin{matrix} p \\ \mathbf{Q}^* \mathbf{A} \end{matrix}$$

It is important to note that this procedure requires only 2 passes over the data matrix \mathbf{A} and this is an important advantage when the matrix is too big to store in fast memory.

3.2.2 Random projection: geometric interpretation

By multiplying our matrix \mathbf{A} with a random matrix Ω which has less number of columns, we can span the range of \mathbf{A} . For a geometric interpretation, let's take $\mathbf{A} = [a_1, a_2, a_3, a_4], n = 3, p = 4, \text{rank} = r = 2$. In Figure 7, we see that the matrix $\mathbf{A}\Omega$ spans the column space of \mathbf{A} , which is the plane \mathbb{R}^2 , with less number of vectors.

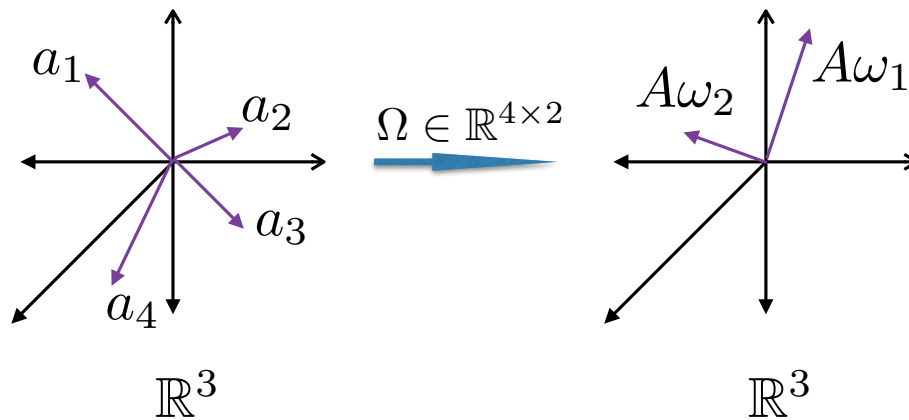


Figure 7: Random sampling can span the range

The random matrix Ω has the following features:

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

Using these, it can be shown that range of the matrix \mathbf{A} is equal to the range of $\mathbf{Y} = \mathbf{A}\Omega$ and $\mathbf{A} = \mathbf{Q}\mathbf{Q}^* \mathbf{A}$, when \mathbf{A} is rank- r . However, in practice the matrix \mathbf{A} is not a rank- r matrix, but 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 $X\omega_1, \dots, X\omega_r$. However they're distorted by $E\omega_i$. That is intuitively the reason why we oversample and take $\ell = r + s$ columns.

3.2.3 Theoretical guarantees

As it is also seen by looking at the theorems given below, the oversampling decreases the error probability or provides a tighter bound on the expected error. In practice an oversampling of $s = 5$ is usually sufficient.

Theorem 5 (simple spectral bound [7]). 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:

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

where σ_i is the i^{th} singular value of \mathbf{A} .

Theorem 6 (deterministic spectral bound [7]). 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 .

3.3 Structured random matrices

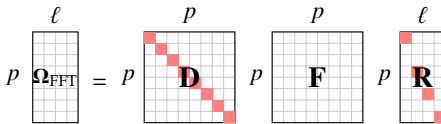
The multiplication $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$ costs $\mathcal{O}(np\ell)$. Therefore if we have a faster way of implementing this multiplication, the random projection method described above becomes more attractive. It turns out that the use of structured matrices such as Fourier or Hadamard allows a faster matrix multiplication using fast Fourier transformation (FFT). In particular using the subsampled random Fourier transform matrix defined below, the cost of $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$ decreases to $\mathcal{O}(np \log \ell)$. If \mathbf{A} has a fast spectral decay, then this approach works as well as Gaussian matrices used above. The slight drawback is that we need more oversampling for a decent performance, however $s = 20$ works fine in practice. This method has the following guarantee:

Theorem 7. 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

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

except with probability $\mathcal{O}(r^{-1})$. [7]

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

$$\mathbf{\Omega}_{\text{FFT}} = \sqrt{\frac{p}{\ell}} \mathbf{D}\mathbf{F}\mathbf{R}$$


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 $p \times \ell$ matrix whose ℓ columns are drawn uniformly from the identity matrix without replacement.

So far we have obtained:

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

using two different type of random matrices: Gaussian and Fourier. However the next step requires multiplying $\mathbf{Q}^*\mathbf{A}$ and it costs $\mathcal{O}(npl)$. This is the bottleneck and could be avoided and reduced to $\mathcal{O}(p^2(n+p))$ 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 using classical linear algebra, since the product matrix has an affordable size. Indeed $\mathbf{A} \approx \mathbf{Q}(\mathbf{Q}^*\mathbf{A})$ is an approximation to the partial QR decomposition and we can then form a partial singular value decomposition out of this decomposition. (next lecture).

3.4 Comparison to classical methods

Obtaining a low rank approximation to a given matrix can also be carried out via full SVD. In this method, the full SVD of a $n \times p$ matrix is computed and truncated. This method is stable, however it is computationally quite expensive: $O(np \min\{n, p\})$. On the other hand Krylov subspace methods choose a random initial vector ω and apply successively a Hermitian operator \mathbf{H} to find the eigenvectors of \mathbf{H} (or first few of them). Well known methods in numerical linear algebra such as Arnoldi and Lanczos algorithms are based on this idea [6]. It might vary but typically the cost is $O(rnp + r^2(n + p))$. These methods require $O(k)$ passes over the data.

Another classical approach to low rank approximations is to use partial QR. This approach consists of using 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 $\mathbf{R} \in \mathbb{R}^{\ell \times p}$ [6]. This method costs $O(npr)$ but more robust compared to Krylov methods. Like Krylov subspace methods, it requires $O(k)$ passes over the data.

In summary, classical techniques require at least $O(npr)$ whereas randomized algorithms can be implemented with a cost of $O(np \log(\ell) + \ell^2(n + p))$. On the other hand, 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 small constant number of passes over the data [7]. Randomized methods are also highly parallelizable, because the multiplication with a random matrix can be efficiently implemented in modern architectures: GPUs, distributed computing, multi-core processors etc.

References

- [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] P. Combettes and J.-C. Pesquet. Signal recovery by proximal forward-backward splitting. In *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212. Springer-Verlag, 2011.
- [3] Jack Dongarra and Francis Sullivan. Guest editors introduction: The top 10 algorithms. *Computing in Science & Engineering*, 2(1):22–23, 2000.
- [4] O. Fercoq and P. Bianchi. A Coordinate Descent Primal-Dual Algorithm with Large Step Size and Possibly Non Separable Functions. *ArXiv e-prints*, August 2015.
- [5] Olivier Fercoq and Peter Richtárik. Accelerated, parallel and proximal coordinate descent. *SIAM J. Optim.*, 25:1997–2023, 2016.
- [6] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.
- [7] 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.
- [8] Y. T. Lee and A. Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. arXiv:1305.1922, 2013.
- [9] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):123–231, 2013.
- [10] Peter Richtárik and Martin Takac. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Math. Program.*, 144:1–38, 2014.
- [11] Bang Cong Vu. A splitting algorithm for dual monotone inclusions involving cocoercive operators. *Adv. Comput. Math.*, 38(3):667–681, 2013.
- [12] Stephen J Wright. Coordinates descent algorithms. *Math. Program.*, 151:3–34, 2015.