A Brief Overview of Practical Optimization Algorithms in the Context of Relaxation

Zhouchen Lin

Peking University

April 22, 2018

Too Many Opt. Problems!

Compressed Sensing: $\min_{\mathbf{x}} \|\mathbf{x}\|_1, \quad s.t. \quad \mathbf{A}\mathbf{x} = \mathbf{b}.$

RPCA w/ Missing Value: $\min \|\mathbf{A}\|_* + \lambda\|\mathbf{E}\|_1, \quad s.t. \quad \pi_\Omega(\mathbf{A} + \mathbf{E}) = \mathbf{d}.$

LASSO: $\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2, \quad s.t. \quad \|\mathbf{x}\|_1 \le \varepsilon.$

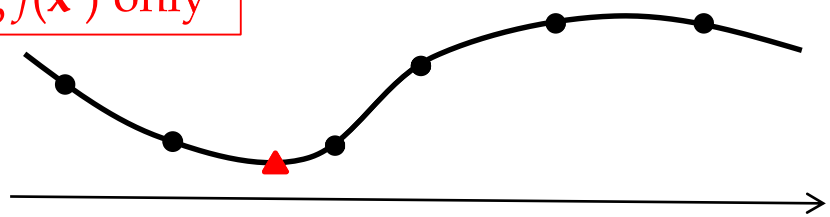
Image Restoration: $\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda||\nabla\mathbf{x}||_1, \quad s.t. \quad 0 \le \mathbf{x} \le 255.$

Covariance Selection:
$$\min_{\mathbf{X}} \operatorname{tr}(\mathbf{\Sigma}\mathbf{X}) - \log(\det(\mathbf{X})) + \rho\mathbf{e}^T|\mathbf{X}|\mathbf{e},$$
$$s.t.\ \lambda_{\min}\mathbf{I} \preccurlyeq \mathbf{X} \preccurlyeq \lambda_{\max}\mathbf{I}.$$

Pose Estimation:
$$\min_{\mathbf{Q}} \operatorname{tr}(\mathbf{W}\mathbf{Q}),$$
$$s.t.\ \operatorname{tr}(\mathbf{A}_i\mathbf{Q}) = 0, i = 1, \cdots, m, \mathbf{Q} \succcurlyeq \mathbf{0}, \operatorname{rank}(\mathbf{Q}) \le 1.$$

Too Many Opt. Algorithms!

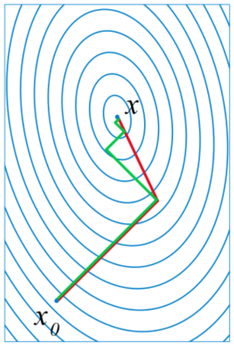
- Zero-th order algorithms: $\boxed{\text{Using } f(\mathbf{x}^k) \text{ only}}$
  - Interpolation Methods
  - Pattern Search Methods
  - …



- First order algorithms: $\boxed{\text{Using } f(\mathbf{x}^k) \text{ \& } \nabla f(\mathbf{x}^k) \text{ only}}$
  - Coordinate Descent
  - (Stochastic) Gradient/Subgradient Descent
  - Conjugate Gradient
  - Quasi-Newton/Hessian-Free Methods
  - (Augmented) Lagrangian Method of Multipliers
  - …

$$\mathbf{x}_i^{k+1} = \mathrm{argmin}_{\mathbf{x}_i} f(\mathbf{x}_1^{k+1}, \cdots, \mathbf{x}_{i-1}^{k+1}, \mathbf{x}_i, \mathbf{x}_{i+1}^k, \cdots, \mathbf{x}_n^k).$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha^k \nabla f(\mathbf{x}^k).$$

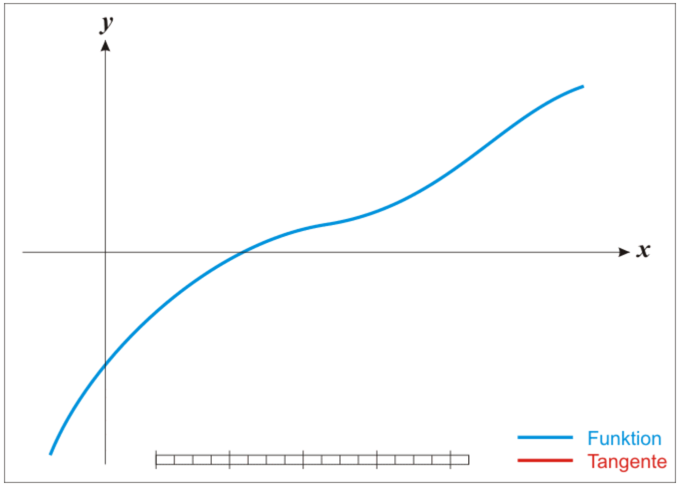


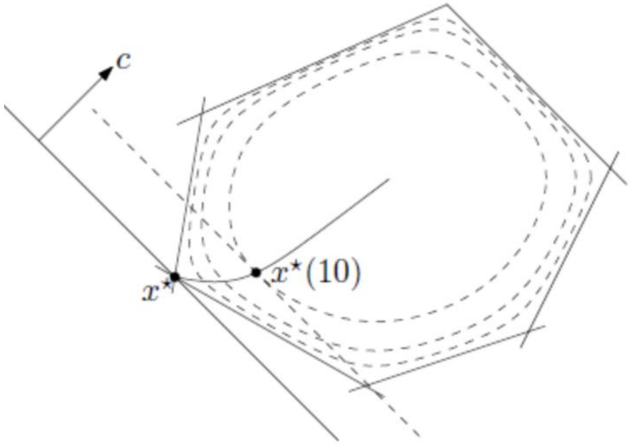
$$\sim \mathbf{H}_f^{-1}(\mathbf{x}^k)$$

$$\sim \mathbf{H}_f(\mathbf{x}^k)\mathbf{g} = \lim_{\varepsilon \to 0} \frac{\nabla f(\mathbf{x}^k + \varepsilon \mathbf{g}) - \nabla f(\mathbf{x}^k)}{\varepsilon}$$

Too Many Opt. Algorithms!

- Second order algorithms: Using $f(\mathbf{x}^k)$, $\nabla f(\mathbf{x}^k)$ & $H_f(\mathbf{x}^k)$ only
  - Newton's Method
  - Sequential Quadratic Programming
  - Interior Point Methods
  - …





Questions

- Can we master optimization algorithms more systematically?
- Is there a methodology to tweak existing algorithms (without proofs)?

Model Optimization Problem & Algorithm

$$\min_{\mathbf{x}} \boxed{f(\mathbf{x})}, \quad s.t.\ \mathbf{x} \in \boxed{\mathcal{C}.}$$

relaxing the objective function

relaxing the constraint

$$\mathbf{x}_{k+1} = \boxed{\phi(\mathbf{x}_k, \nabla f(\mathbf{x}_k))}.$$

relaxing the update process

Relaxing the Objective Function

- Majorization Minimization

$$\min_{\mathbf{x}} \ f(\mathbf{x}), \quad s.t.\ \mathbf{x} \in \mathcal{C}.$$

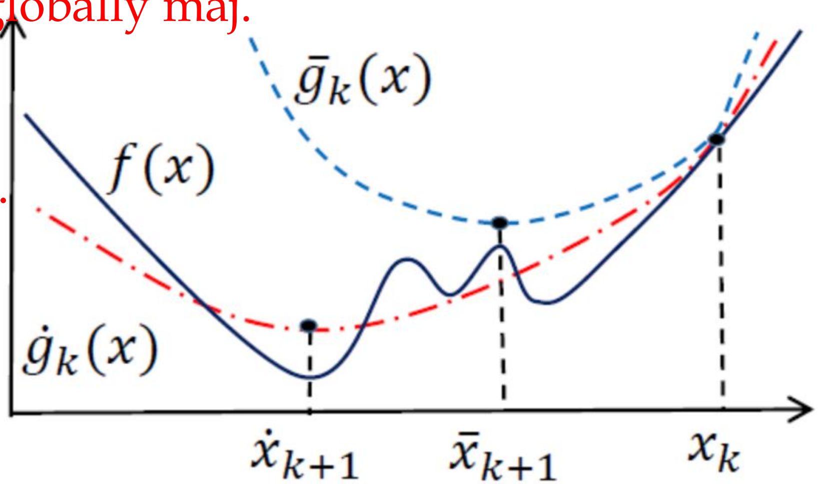
$$\min_{\mathbf{x}} \ g_k(\mathbf{x}), \quad s.t.\ \mathbf{x} \in \mathcal{C}.$$

1. $f(\mathbf{x}) \leq g_k(\mathbf{x}), \forall \mathbf{x} \in \mathcal{C};$ globally majorant
2. $f(\mathbf{x}_k) = g_k(\mathbf{x}_k).$

$$\boxed{\mathbf{x}_{k+1} = \operatorname*{argmin}_{\mathbf{x}} g_k(\mathbf{x}) \Longrightarrow f(\mathbf{x}_{k+1}) \leq g_k(\mathbf{x}_{k+1}) \leq g_k(\mathbf{x}_k) = f(\mathbf{x}_k).}$$

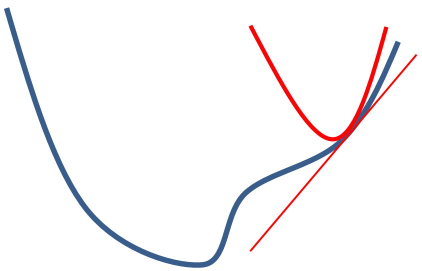
globally maj.

locally maj.



Relaxing the Objective Function

- Majorization Minimization
  - How to choose the majorant function?



$$\min_{\mathbf{x}} \; g_k(\mathbf{x}), \quad s.t. \; \mathbf{x} \in \mathcal{C}.$$

$$g_k(\mathbf{x}) = f(\mathbf{x}_k) + \langle \nabla f(\mathbf{x}_k), \mathbf{x} - \mathbf{x}_k \rangle + \frac{1}{2\alpha} \|\mathbf{x} - \mathbf{x}_k\|^2.$$

$$\mathbf{x}_{k+1} = \mathcal{P}_{\mathcal{C}}(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)).$$

projected gradient descent

$$(\mathcal{C} = \mathbb{R}^n)$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k).$$

gradient descent

Relaxing the Objective Function

- Majorization Minimization
  - How to choose the stepsize α?

$$f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)) < f(\mathbf{x}_k).$$

locally majorant

$$\Downarrow$$

$$\text{If not satisfied } \alpha \leftarrow \mu\alpha. \ (\mu \in (0,1))$$

backtracking

If f is L-smooth, i.e., $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \le L\|\mathbf{x}-\mathbf{y}\|$, then we may choose

$$\alpha = L^{-1}.$$

globally majorant

$$f(\mathbf{x}) \le f(\mathbf{x}_k) + \langle \nabla f(\mathbf{x}_k), \mathbf{x} - \mathbf{x}_k \rangle + \frac{1}{2\alpha}\|\mathbf{x}-\mathbf{x}_k\|^2 \triangleq g_k(\mathbf{x}).$$

$$f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)) < f(\mathbf{x}_k) - \beta\alpha\|\nabla f(\mathbf{x})\|^2. \ (\beta \in (0,1))$$

Armijo's rule

All the above are relaxation of exact line search for stepsize:

$$\alpha = \mathrm{argmin}_\alpha f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)).$$

Relaxing the Objective Function

- Majorization Minimization
  - How to choose the majorant function?

$$\min_{\mathbf{x}} \; g_k(\mathbf{x}), \quad s.t. \; \mathbf{x} \in \mathcal{C}.$$

$$g_k(\mathbf{x}) = f(\mathbf{x}_k) + \langle \nabla f(\mathbf{x}_k), \mathbf{x} - \mathbf{x}_k \rangle + \frac{1}{2\alpha}\|\mathbf{x} - \mathbf{x}_k\|^2.$$

asymptotic smoothness

$$g_k(\mathbf{x}) - f(\mathbf{x}) \text{ is smooth.}$$

$$g_k(\mathbf{x}) \geq f(\mathbf{x}), \;\; \forall \mathbf{x}$$

globally majorant

$$\lim_{k\to\infty} \nabla g_k(\mathbf{x}_k, \mathbf{d}) - \nabla f(\mathbf{x}_k; \mathbf{d}) = 0.$$

$$g_k(\mathbf{x}_{k+1}) \geq f(\mathbf{x}_{k+1}).$$

locally majorant

Relaxed Majorization Minimization

Robust Matrix Factorization: $\min_{\mathbf{U}\in\mathcal{C}_U, \mathbf{V}\in\mathcal{C}_V} \|\mathbf{W}\odot(\mathbf{M}-\mathbf{U}\mathbf{V}^T)\|_1 + R_u(\mathbf{U}) + R_v(\mathbf{V}).$

C. Xu, Z. Lin, and H. Zha, Relaxed Majorization-Minimization for Non-smooth and Non-convex Optimization, AAAI 2016.
Zhouchen Lin, Chen Xu,and Hongbin Zha, Robust Matrix Factorization by Majorization-Minimization, IEEE TPAMI, 2018.

Relaxing the Objective Function

- Majorization Minimization
  - How to choose the majorant function?

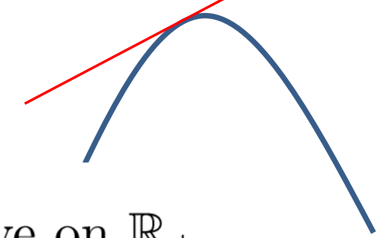
$$\min_{\mathbf{x}} \; f(\mathbf{x}) + h(\mathbf{x}), \quad s.t. \; \mathbf{x} \in \mathcal{C},$$

where f is convex and h is concave.

$$g_k(\mathbf{x}) = f(\mathbf{x}) + \langle \partial h(\mathbf{x}_k), \mathbf{x} - \mathbf{x}_k \rangle + h(\mathbf{x}_k),$$

convex concave procedure (CCCP)

where ∂h is a super-gradient of h.



low-rankness regularizer

$$\min_{\mathbf{X}} \boxed{\sum_{i=1}^{\min(m,n)} h(\sigma_i(\mathbf{X}))} + f(\mathbf{X}), \text{ where } h \text{ is concave on } \mathbb{R}_+.$$

$$h(\sigma_i) \leq h(\sigma_i^k) + w_i^k(\sigma_i - \sigma_i^k), \quad w_i^k \in \partial h(\sigma_i).$$

A. L. Yuille, A. Rangarajan, The Concave-Convex Procedure. Neural Computation 15(4): 915-936 (2003).
C. Lu, J. Tang, S. Yan, Z. Lin, Generalized nonconvex nonsmooth low-rank minimization, CVPR 2014.
C. Lu, J. Tang, S. Yan, Z. Lin, Nonconvex nonsmooth low-rank minimization via iteratively reweighted nuclear norm, IEEE TIP 2016.
Canyi Lu, Yunchao Wei, Zhouchen Lin, and Shuicheng Yan, Proximal Iteratively Reweighted Algorithm with Multiple Splitting for Nonconvex Sparsity Optimization, AAAI 2014.

Relaxing the Objective Function

- Majorization Minimization
  - How to choose the majorant function?

$$\min_{\mathbf{x}} \; f(\mathbf{x}), \quad s.t. \; \mathbf{x} \in \mathcal{C}, \quad \text{where } f(\mathbf{x}) = \min_{\mathbf{y} \in \mathcal{Y}} h(\mathbf{x}, \mathbf{y}).$$

Variational surrogate: $g_k(\mathbf{x}) = h(\mathbf{x}, \mathbf{y}_k^*), \quad \text{where } \mathbf{y}_k^* = \operatorname*{argmin}_{\mathbf{y} \in \mathcal{Y}} h(\mathbf{x}_k, \mathbf{y}).$

Schatten-p norm: $\|\mathbf{X}\|_{S_p} = \left(\sum_i \sigma_i^p(\mathbf{X}) \right)^{1/p}$, low-rankness regularizer.

Theorem 1. *With compatiable dimensions and* $\frac{1}{p} = \sum_{i=1}^{I} \boxed{\frac{1}{p_i}}$:

$$\frac{1}{p} \|\mathbf{X}\|_{S_p}^p = \min_{\mathbf{X} = \Pi_{i=1}^{I} \mathbf{X}_i} \sum_{i=1}^{I} \frac{1}{p_i} \|\mathbf{X}_i\|_{S_{p_i}}^{p_i}.$$

If $0 < p < 1$, we can still choose $p_i \geq 1$.

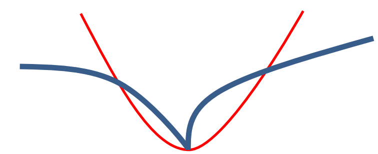
Chen Xu, Zhouchen Lin, and Hongbin Zha, A Unified Convex Surrogate for the Schatten-p Norm, AAAI 2017.
Fanhua Shang, Yuanyuan Liu, James Cheng, Zhi-Quan Luo, and Zhouchen Lin, Bilinear Factor Matrix Norm Minimization for Robust PCA: Algorithms and Applications, IEEE TPAMI, 2018.

Relaxing the Objective Function

- Majorization Minimization
  - How to choose the majorant function?

$$\min_{\mathbf{x}} \; f(\mathbf{x}) + h(\mathbf{x}), \quad s.t.\ \mathbf{x} \in \mathcal{C},$$

where f is convex and h is non-convex.



$$g_k(\mathbf{x}) = f(\mathbf{x}) + \mathbf{x}^T \mathbf{H}(\mathbf{x}_k)\mathbf{x},$$

where $\mathbf{H}(\mathbf{x}_k)$ satisfies: $\mathbf{H}(\mathbf{x}_k) \succeq \mathbf{0}$, $\mathbf{x}_k^T \mathbf{H}(\mathbf{x}_k)\mathbf{x}_k = h(\mathbf{x}_k)$ and $\mathbf{x}^T \mathbf{H}(\mathbf{x}_k)\mathbf{x} \geq h(\mathbf{x}), \forall \|\mathbf{x}\| \geq \varepsilon_k$.

l_p-norm, sparsity regularizer

$$h(\mathbf{x}) = \boxed{\|\mathbf{x}\|_p^p} \Longrightarrow \mathbf{x}^T \mathbf{H}(\mathbf{x}_k)\mathbf{x} = \mathbf{x}^T \operatorname{Diag}(|x_{k,i}|^{p-2})\mathbf{x}.$$

$$h(\mathbf{X}) = \boxed{\operatorname{tr}\left((\mathbf{X}\mathbf{X}^T)^{p/2}\right)} \Longrightarrow \langle \mathbf{X}, \mathbf{H}(\mathbf{X}_k)\mathbf{X}\rangle = \operatorname{tr}\left((\mathbf{X}_k\mathbf{X}_k^T)^{p/2-1}\mathbf{X}\mathbf{X}^T\right).$$

Schatten-p norm, low-rankness regularizer

Iteratively Reweighted Least Squares

E. Candès, M.B. Wakin, S.P. Boyd, Enhancing sparsity by reweighted l_1 minimization, Journal of Fourier Analysis and Applications 14 (5–6) (2008) 877–905.

C. Lu, Z. Lin, S. Yan, Smoothed low rank and sparse matrix recovery by iteratively reweighted least squared minimization, IEEE TIP, 2015.

Relaxing the Objective Function

- Majorization Minimization
  - How to choose the majorant function?

Other choices:

- Julien Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. SIAM Journal on Optimization, 25(2):829–855, 2015.

Relaxing the Constraints

- What if no relaxation?
  - Only works for simple constraints

λ has to go to ∞

Penalty method: $\mathcal{A}(\mathbf{x}) = \mathbf{b} \Longrightarrow \lambda\|\mathcal{A}(\mathbf{x}) - \mathbf{b}\|^2$.
Barrier method: $\mathbf{x} \geq \mathbf{0} \Longrightarrow -\lambda\sum_i \log x_i$, $\mathbf{X} \succeq \mathbf{0} \Longrightarrow -\lambda \log\det \mathbf{X}$.

But what if: $\mathcal{A}(\mathbf{x}) = \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$ or even more complex?

Relaxing the Constraints

- Method of Lagrange Multipliers

Model problem:

$$\min_{\mathbf{x}} \ f(\mathbf{x}), \quad \mathcal{A}(\mathbf{x}) = \mathbf{b}, \mathbf{x} \in \mathcal{C}.$$

Lagrangian function:

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \langle \boldsymbol{\lambda}, \mathcal{A}(\mathbf{x}) - \mathbf{b} \rangle .$$

may not have a solution

$$\mathbf{x}_{k+1} = \operatorname*{argmin}_{\mathbf{x} \in \mathcal{C}} L(\mathbf{x}, \boldsymbol{\lambda}_k) = \operatorname*{argmin}_{\mathbf{x} \in \mathcal{C}} f(\mathbf{x}) + \langle \boldsymbol{\lambda}_k, \mathcal{A}(\mathbf{x}) - \mathbf{b} \rangle ,$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \boxed{\alpha_k} (\mathcal{A}(\mathbf{x}_{k+1}) - \mathbf{b}).$$

$\mathcal{A}(\mathbf{x}) = \mathbf{b}, \mathbf{x} \in \mathcal{C}$ is achieved only when convergence!

not easy to choose

Augmented Lagrangian function:

need not go to ∞

$$\tilde{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \langle \boldsymbol{\lambda}, \mathcal{A}(\mathbf{x}) - \mathbf{b} \rangle + \frac{\mu}{2} \|\mathcal{A}(\mathbf{x}) - \mathbf{b}\|^2.$$

$$\mathbf{x}_{k+1} = \operatorname*{argmin}_{\mathbf{x} \in \mathcal{C}} \tilde{L}(\mathbf{x}, \boldsymbol{\lambda}_k) = \operatorname*{argmin}_{\mathbf{x} \in \mathcal{C}} f(\mathbf{x}) + \langle \boldsymbol{\lambda}_k, \mathcal{A}(\mathbf{x}) - \mathbf{b} \rangle + \boxed{\frac{\mu_k}{2}} \|\mathcal{A}(\mathbf{x}) - \mathbf{b}\|^2,$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \mu_k (\mathcal{A}(\mathbf{x}_{k+1}) - \mathbf{b}).$$

Relaxing the Constraints

- Method of Lagrange Multipliers

$$\mathbf{x}_{k+1} = \operatorname*{argmin}_{\mathbf{x}\in\mathcal{C}} f(\mathbf{x}) + \langle \boldsymbol{\lambda}_k, \mathcal{A}(\mathbf{x}) - \mathbf{b} \rangle + \frac{\mu_k}{2}\|\mathcal{A}(\mathbf{x}) - \mathbf{b}\|^2$$

$$= \operatorname*{argmin}_{\mathbf{x}\in\mathcal{C}} f(\mathbf{x}) + \boxed{\frac{\mu_k}{2}\|\mathcal{A}(\mathbf{x}) - \mathbf{b} + \boldsymbol{\lambda}_k/\mu_k\|^2.}$$

$\leq$ majorant

$$\mathbf{x}_{k+1} = \operatorname*{argmin}_{\mathbf{x}\in\mathcal{C}} f(\mathbf{x}) + \boxed{\mu_k \langle \mathcal{A}^*(\mathcal{A}(\mathbf{x}_k) - \mathbf{b} + \boldsymbol{\lambda}_k/\mu_k), \mathbf{x} - \mathbf{x}_k \rangle + \frac{\beta_k}{2}\|\mathbf{x} - \mathbf{x}_k\|^2}$$

$$= \operatorname*{argmin}_{\mathbf{x}\in\mathcal{C}} f(\mathbf{x}) + \frac{\beta_k}{2}\left\|\mathbf{x} - \mathbf{x}_k + \mu_k\beta_k^{-1}\mathcal{A}^*(\mathcal{A}(\mathbf{x}_k) - \mathbf{b} + \boldsymbol{\lambda}_k/\mu_k)\right\|^2.$$

$\boxed{\text{Majorant condition: } \beta_k \geq \mu_k\|\mathcal{A}\|^2.}$

When $f(\mathbf{x})$ is separable, i.e., $f(\mathbf{x}) = \sum_i^n f_i(\mathbf{x}_i)$, $\mathbf{x} = (\mathbf{x}_i^T, \cdots, \mathbf{x}_n^T)^T$, the above becomes Linearized Alternating Direction Method with Parallel Splitting and Adaptive Penalty (LADMPSAP).

Zhouchen Lin, Risheng Liu, and Huan Li, Linearized Alternating Direction Method with Parallel Splitting and Adaptive Penalty for Separable Convex Programs in Machine Learning, Machine Learning, 2015

Relaxing the Constraints

- Method of Lagrange Multipliers

$$\mathbf{x}_{k+1} = \operatorname*{argmin}_{\mathbf{x}\in\mathcal{C}} f(\mathbf{x}) + \frac{\beta_k}{2}\left\|\mathbf{x} - \mathbf{x}_k + \mu_k\beta_k^{-1}\mathcal{A}^*(\mathcal{A}(\mathbf{x}_k) - \mathbf{b} + \boldsymbol{\lambda}_k/\mu_k)\right\|^2.$$

Reformulate the problem as:

$$\min_{\mathbf{x},\mathbf{y}} \; f(\mathbf{x}), \quad \mathcal{A}(\mathbf{x}) = \mathbf{b}, \mathbf{x} = \mathbf{y}, \mathbf{y} \in \mathcal{C}.$$

can be further relaxed if *f* is still complex

proximal operator

relatively easy to solve

$$\mathbf{x}_{k+1} = \boxed{\operatorname{argmin} \boxed{f(\mathbf{x})} + \frac{\beta_k}{2}\|\mathbf{x} - \tilde{\mathbf{x}}_k\|^2},$$

$$\mathbf{y}_{k+1} = \boxed{\mathcal{P}_{\boxed{\mathcal{C}}}(\tilde{\mathbf{y}}_k).}$$

projection operator

More auxiliary variables can be introduced if $\mathcal{C}$ is still complex, e.g.,

$$\mathbf{x} \in \cap_i \mathcal{C}_i \Longrightarrow \mathbf{x} = \mathbf{y}_i, \mathbf{y}_i \in \mathcal{C}_i.$$

Zhouchen Lin, Risheng Liu, and Huan Li, Linearized Alternating Direction Method with Parallel Splitting and Adaptive Penalty for Separable Convex Programs in Machine Learning, Machine Learning, 2015

Relaxing the Constraints

- Method of Lagrange Multipliers

More investigations:

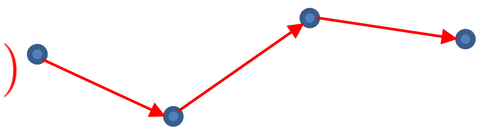
- Zhouchen Lin, Risheng Liu, and Huan Li, Linearized Alternating Direction Method with Parallel Splitting and Adaptive Penalty for Separable Convex Programs in Machine Learning, Machine Learning, 2015.
- Canyi Lu, Jiashi Feng, Shuicheng Yan, and Zhouchen Lin, A Unified Alternating Direction Method of Multipliers by Majorization Minimization, IEEE Trans. Pattern Analysis and Machine Intelligence, 2018.

Relaxing the Update Process

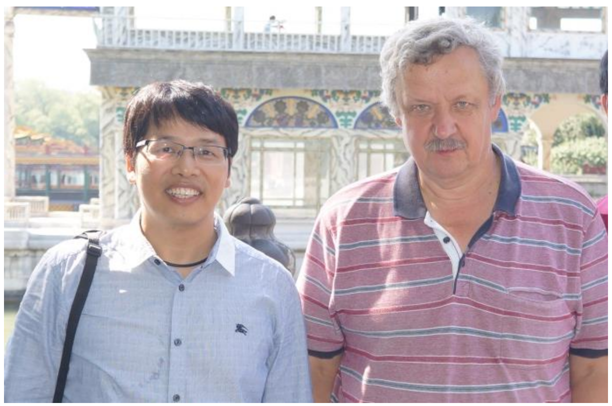
- Relaxing the location to compute gradient

$$\min_{\mathbf{x}} \; f(\mathbf{x})$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k) : \quad O(k^{-1})$$



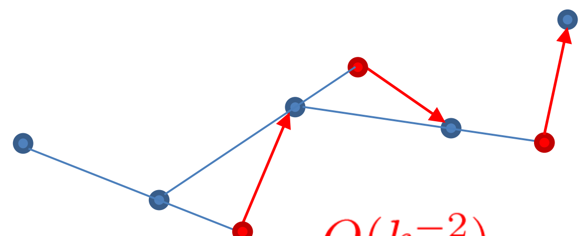
Nesterov's acceleration for L-smooth function f:



$$\mathbf{x}_k = \mathbf{y}_k - L^{-1} \nabla f(\boxed{\mathbf{y}_k}),$$

$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2},$$

$$\mathbf{y}_{k+1} = \mathbf{x}_k + \frac{t_k - 1}{t_{k+1}} (\mathbf{x}_k - \mathbf{x}_{k-1}),$$



$O(k^{-2})$

where $\mathbf{x}_0 = \mathbf{y}_1 = \mathbf{0}$ and $t_1 = 1$.

$$\min_{\mathbf{x}} \; f(\mathbf{x}) + g(\mathbf{x}) \qquad \boxed{\mathbf{x}_k = \mathrm{Prox}_{L^{-1}g}(\mathbf{y}_k - L^{-1} \nabla f(\mathbf{y}_k))}$$

Y. Nesterov, A method of solving a convex programming problem with convergence rate O(1/k^2), Soviet Mathematics Doklady 27 (2) (1983) 372–376.

A. Beck,M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM Journal on Imaging Sciences 2 (1) (2009) 183–202.

Relaxing the Update Process

- Relaxing the location to compute gradient

Monotone APG for non-convex programs:

$$
\begin{aligned}
\mathbf{y}_k &= \mathbf{x}_k + \frac{t_{k-1}}{t_k}(\mathbf{z}_k - \mathbf{x}_k) + \frac{t_{k-1}-1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1}),\\
\mathbf{z}_{k+1} &= \operatorname{prox}_{L^{-1}g}(\mathbf{y}_k - L^{-1}\nabla f(\mathbf{y}_k)),\\
\boxed{\mathbf{v}_{k+1} = \operatorname{prox}_{L^{-1}g}(\mathbf{x}_k - L^{-1}\nabla f(\mathbf{x}_k)),} &\longleftarrow \text{monitor}\\
t_{k+1} &= \frac{\sqrt{4(t_k)^2+1}+1}{2},\\
\mathbf{x}_{k+1} &= \begin{cases} \mathbf{z}_{k+1}, & \text{if } F(\mathbf{z}_{k+1}) \le F(\boxed{\mathbf{v}_{k+1}}), \\ \boxed{\mathbf{v}_{k+1}}, & \text{otherwise.} \end{cases} \longleftarrow \text{corrector}
\end{aligned}
$$

changed $\mathbf{x}_k$ to $\mathbf{v}_{k+1}$

Converges at non-convex case, maintains $O(k^{-2})$ convergence rate at convex case.

Canyi Lu, Huan Li, Zhouchen Lin, and Shuicheng Yan, Fast Proximal Linearized Alternating Direction Method of Multiplier with Parallel Splitting, pp. 739-745, AAAI 2016.

Huan Li and Zhouchen Lin, Accelerated Proximal Gradient Methods for Nonconvex Programming, NIPS 2015.

Relaxing the Update Process

- Relaxing the evaluation of gradient
  - Stochastic gradient descent (SGD)

$$\min_{\mathbf{x}} \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) \Longrightarrow f_{i_j}(\mathbf{x}^k), \nabla f_{i_j}(\mathbf{x}^k)$$

Variance Reduction: Compute the full gradient $\frac{1}{n}\sum_{i=1}^{n} \nabla f_{i_j}(\mathbf{x})$ periodically and use it to correct the stochastic gradient, so that the variance in the stochastic gradient can be reduced.

Algorithm 1 Serial SVRG

Input x_0^0, epoch length m, step size γ, and $S = \lfloor K/m \rfloor$.
for $s = 0$ to $S-1$ do
  $g^s = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\boxed{x_0^s})$, snapshot vector
  for $k = 0$ to $m-1$ do
    Randomly sample i_k from $1, 2, \cdots, n$,
    $\boxed{v_k^s} = \boxed{\nabla f_{i_k}(x_k^s)} - \nabla f_{i_k}(x_0^s) + g^s$,
    $x_{k+1}^s = x_k^s - \boxed{\gamma} v_k^s$, fixed stepsize
  end for k.
  $x_0^{s+1} = x_m^s$,
end for s.

$$\mathbf{v}_k^s = \nabla f_{i_k}(\mathbf{x}_k^s) - \nabla f_{i_k}(\mathbf{x}_0^s) + \nabla f(\mathbf{x}_0^s).$$

$$\mathbf{x}_k^s, \mathbf{x}_0^s \approx \mathbf{x}^*$$

$$\Downarrow$$

$$\mathbf{v}_k^s \approx \nabla f_{i_k}(\mathbf{x}^*) - \nabla f_{i_k}(\mathbf{x}^*) + \nabla f(\mathbf{x}^*) = 0.$$

$$\nabla f_{i_k}(\mathbf{x}^*) \neq \mathbf{0} \; (\nabla f(\mathbf{x}^*) = \mathbf{0})$$

R. Johnson and T. Zhang, Accelerating stochastic gradient descent using predictive variance reduction. NIPS 2013.

Relaxing the Update Process

- Relaxing the update of variable
  - Asynchronous update

Algorithm 1 Serial SVRG

Input $\mathbf{x}_0^0$, epoch length m, step size γ, and $S = \lfloor K/m \rfloor$.

for $s = 0$ **to** $S-1$ **do**

$\quad \mathbf{g}^s = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x}_0^s)$,

$\quad$ **for** $k = 0$ **to** $m-1$ **do**

$\quad\quad$ Randomly sample i_k from $1, 2, \cdots, n$,

$\quad\quad \mathbf{v}_k^s = \nabla f_{i_k}(\boxed{\mathbf{x}_k^s}) - \nabla f_{i_k}(\mathbf{x}_0^s) + \mathbf{g}^s$,

$\quad\quad \mathbf{x}_{k+1}^s = \mathbf{x}_k^s - \gamma \mathbf{v}_k^s$,

$\quad$ **end for** k.

$\quad \mathbf{x}_0^{s+1} = \mathbf{x}_m^s$,

end for s.

Algorithm 2 ASVRG

Input $\mathbf{x}_0^0$, epoch length m, step size γ, and $S = \lfloor K/m \rfloor$.

for $s = 0$ **to** $S-1$ **do**

$\quad \mathbf{g}^s = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x}_0^s)$,

$\quad$ **for** $k = 0$ **to** $m-1$ **do**

$\quad\quad$ Randomly sample i_k from $1, 2, \cdots, n$,

$\quad\quad \mathbf{v}_{j(k)}^s = \nabla f_{i_k}(\boxed{\mathbf{x}_{j(k)}^s}) - \nabla f_{i_k}(\mathbf{x}_0^s) + \mathbf{g}^s$,

$\quad\quad \mathbf{x}_{k+1}^s = \mathbf{x}_k^s - \gamma \mathbf{v}_{j(k)}^s$,

$\quad$ **end for** k.

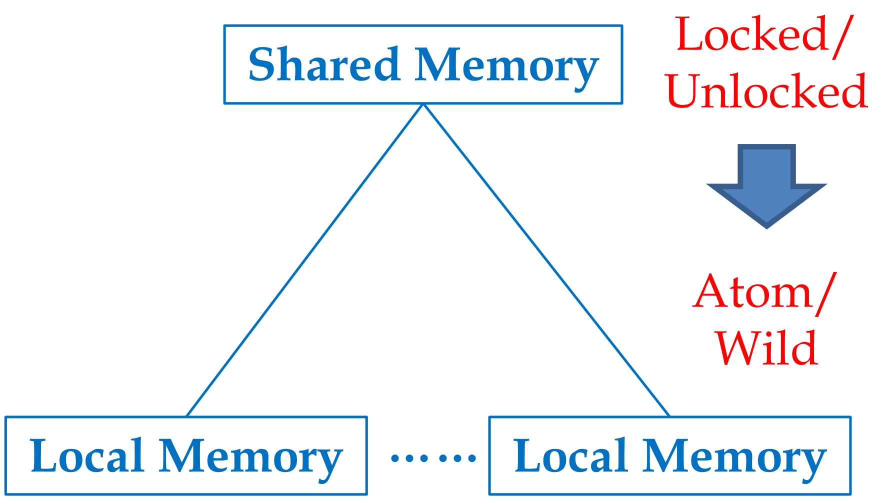
$\quad \mathbf{x}_0^{s+1} = \mathbf{x}_m^s$,

end for s.

R. Johnson and T. Zhang, Accelerating stochastic gradient descent using predictive variance reduction. NIPS 2013.
Cong Fang and Zhouchen Lin, Parallel Asynchronous Stochastic Variance Reduction for Nonconvex Optimization, AAAI 2017.

Relaxing the Update Process

- Relaxing the update of variable
  - Asynchronous update



Algorithm 2 ASVRG

Input $\mathbf{x}_0^0$, epoch length m, step size γ, and $S = \lfloor K/m \rfloor$.

for $s = 0$ **to** $S-1$ **do**

  $\mathbf{g}^s = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x}_0^s)$,

  for $k = 0$ **to** $m-1$ **do**

    Randomly sample i_k from $1, 2, \cdots, n$,

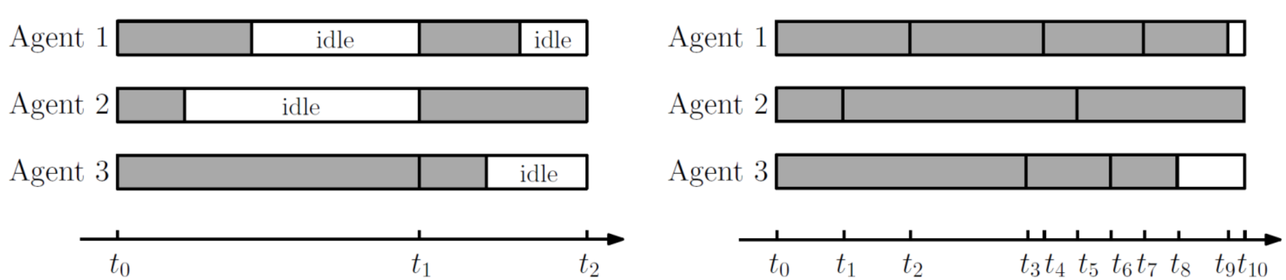
    $\mathbf{v}_{j(k)}^s = \nabla f_{i_k}(\mathbf{x}_{j(k)}^s) - \nabla f_{i_k}(\mathbf{x}_0^s) + \mathbf{g}^s$,

    $\mathbf{x}_{k+1}^s = \mathbf{x}_k^s - \gamma \mathbf{v}_{j(k)}^s$,

  end for k.

  $\mathbf{x}_0^{s+1} = \mathbf{x}_m^s$,

end for s.



(a) Sync-parallel computing (b) Async-parallel computing

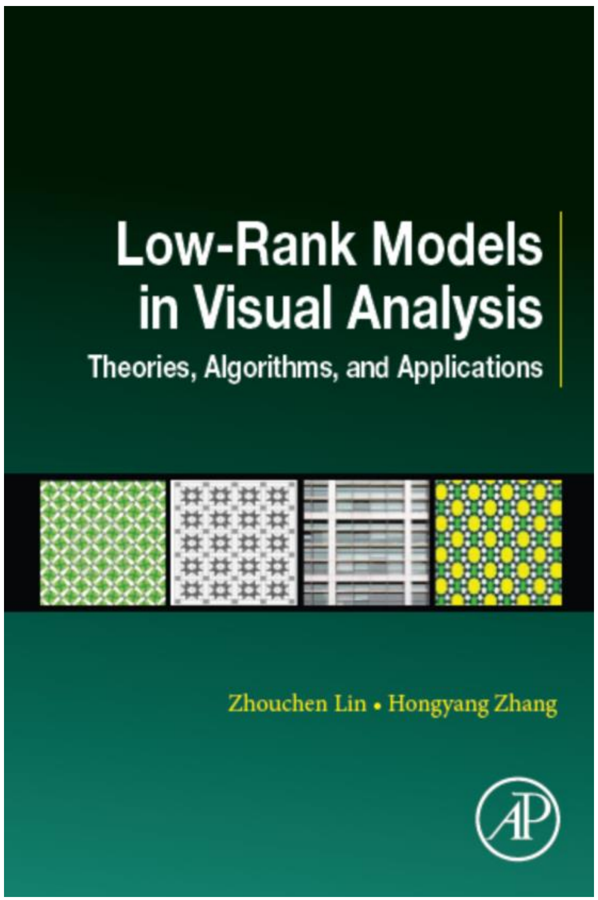
Cong Fang and Zhouchen Lin, Parallel Asynchronous Stochastic Variance Reduction for Nonconvex Optimization, AAAI 2017.

Conclusions

- Relaxation is good and even necessary for optimization
- The same for life!

Thanks!

- zlin@pku.edu.cn
- http://www.cis.pku.edu.cn/faculty/vision/zlin/zlin.htm



Recruitment: PostDocs (**540K** RMB/year) and **Faculties** in **machine learning** related areas
Please Google me and visit my webpage!