The Proximal Alternating Linearized Minimization (PALM) algorithm

[Bolte et al., 2014]

Minimization of a sum of finite functions

$$\text{minimize}_{x,y}\Psi\left(x,y\right) := f\left(x\right) + g\left(y\right) + H\left(x,y\right)$$

- f(x), g(y) are proper, nonconvex, lower semicontinuous functions
- H(x,y) smooth, C^1, function

In a convex setting, a Gauss Seidel-type or coordinate descent optimization

$$x^{k+1} \in \text{argmin}_x \Psi\left(x, y^k\right)$$

$$y^{k+1} \in \text{argmin}_y \Psi\left(x^{k+1}, y\right)$$

converges when the minimum in each step is uniquely attained

- no convergence in the nonconvex setting, difficult subproblems

In the non-convex setting, the following proximal regularization of the Gauss Seidel scheme provides a non-increasing sequence of points

$$x^{k+1} \in \operatorname{argmin}_x \left\{ \Psi\left(x, y^k\right) + \frac{c_k}{2} \left\| x - x^k \right\|^2 \right\}$$
$$y^{k+1} \in \operatorname{argmin}_y \left\{ \Psi\left(x^{k+1}, y\right) + \frac{d_k}{2} \left\| y - y^k \right\|^2 \right\}$$

- More well-posed subproblems

Two drawbacks here

- solving for the minimum per iteration is a difficult problem, due to nonconvexity and nonsmoothness
- accumulation of computation errors

Assume a cost funtion with smooth h(x) and nonsmooth σ(x).

$$\Psi(x) = \sigma(x) + h(x)$$

A proximal forward-backward scheme is expressed as

$$x^{k+1} \in \operatorname{prox}_t^{\sigma}\left(x^k - \frac{1}{t}\nabla h\left(x^k\right)\right)$$

$$x^{k+1} \in \operatorname{argmin}_{x\in\mathbb{R}^d}\left\{\left\langle x - x^k, \nabla h\left(x^k\right)\right\rangle + \frac{t}{2}\left\|x - x^k\right\|^2 + \sigma\left(x\right)\right\}, \quad (t > 0)$$

PALM's main idea: replace Ψ with its linearization

$$\widehat{\Psi}\left(x, y^k\right) = \left\langle x - x^k, \nabla_x H\left(x^k, y^k\right)\right\rangle + \frac{c_k}{2}\left\|x - x^k\right\|^2 + f\left(x\right), \quad (c_k > 0)$$

$$\widehat{\widehat{\Psi}}\left(x^{k+1}, y\right) = \left\langle y - y^k, \nabla_y H\left(x^{k+1}, y^k\right)\right\rangle + \frac{d_k}{2}\left\|y - y^k\right\|^2 + g\left(y\right), \quad (d_k > 0)$$

PALM: Proximal Alternating Linearized Minimization

Initialization: start with any $(x^0, y^0) \in \mathbb{R}^n \times \mathbb{R}^m$.

For each $k = 0, 1, \ldots$ generate a sequence $\left\{\left(x^k, y^k\right)\right\}_{k\in\mathbb{N}}$ as follows:

Take $\gamma_1 > 1$, set $c_k = \gamma_1 L_1\left(y^k\right)$ and compute

$$x^{k+1} \in \operatorname{prox}_{c_k}^{f}\left(x^k - \frac{1}{c_k}\nabla_x H\left(x^k, y^k\right)\right).$$

Take $\gamma_2 > 1$, set $d_k = \gamma_2 L_2\left(x^{k+1}\right)$ and compute

$$y^{k+1} \in \operatorname{prox}_{d_k}^{g}\left(y^k - \frac{1}{d_k}\nabla_y H\left(x^{k+1}, y^k\right)\right).$$

- When there is no y term, PALM reduces to PFB
- The gradient of the smooth part H has to be globally Lipschitz continuous
- PALM converges to a critical point of Ψ.
- Convergence results for PALM exist for the case that the function Ψ to be minimized satisfies the so-called Kurdyka- Lojasiewicz (KL) property (sharpness of the gradient near critical points)

Example: PALM for BSS

$$\min_{\mathbf{A},\boldsymbol{\alpha}} \frac{1}{2}\|\mathbf{X} - \mathbf{A}\boldsymbol{\alpha}\boldsymbol{\Phi}^*\|_F^2 \; + \lambda\|\boldsymbol{\alpha}\|_1 + \imath_{\mathcal{C}}(\mathbf{A})$$

$$\mathbf{S} = \boldsymbol{\alpha}\boldsymbol{\Phi}^* \quad \boldsymbol{\alpha} \in \mathbb{R}^{N\times K}$$

[Feng & Kowalski, 2018]

Algorithm 1: BSS-PALM

Initialization : $\boldsymbol{\alpha}^{(1)} \in \mathbb{R}^{N\times K}$, $\mathbf{A}^{(1)} \in \mathbb{R}^{M\times N}$, $L^{1,(1)} = \|\mathbf{A}^{(1)}\|_2^2$, $L^{2,(1)} = \|\boldsymbol{\alpha}^{(1)}\boldsymbol{\Phi}^*\|_2^2$, $j = 1$;

repeat

1. $\nabla_{\boldsymbol{\alpha}} Q\left(\boldsymbol{\alpha}^{(j)}, \mathbf{A}^{(j)}\right) = -{\mathbf{A}^{(j)}}^T \left(\mathbf{X} - \mathbf{A}^{(j)}\boldsymbol{\alpha}^{(j)}\boldsymbol{\Phi}^*\right)\boldsymbol{\Phi}$;
2. $\boldsymbol{\alpha}^{(j+1)} = \mathcal{S}_{\lambda/L^{1,(j)}}\left(\boldsymbol{\alpha}^{(j)} - \frac{1}{L^{1,(j)}}\nabla_{\boldsymbol{\alpha}} Q(\boldsymbol{\alpha}^{(j)}, \mathbf{A}^{(j)})\right)$;
3. $\nabla_{\mathbf{A}} Q(\boldsymbol{\alpha}^{(j+1)}, \mathbf{A}^{(j)}) = -(\mathbf{X} - \mathbf{A}^{(j)}\boldsymbol{\alpha}^{(j+1)}\boldsymbol{\Phi}^*)\boldsymbol{\Phi}{\boldsymbol{\alpha}^{(j+1)}}^H$;
4. $\mathbf{A}^{(j+1)} = \mathcal{P}_{\mathcal{C}}\left(\mathbf{A}^{(j)} - \frac{1}{L^{2,(j)}}\nabla_{\mathbf{A}} Q(\boldsymbol{\alpha}^{(j+1)}, \mathbf{A}^{(j)})\right)$;
5. $L^{1,(j+1)} = \|\mathbf{A}^{(j+1)}\|_2^2$;
6. $L^{2,(j+1)} = \|\boldsymbol{\alpha}^{(j+1)}\boldsymbol{\Phi}^*\|_2^2$;
7. $j = j + 1$;

until *convergence*;

The stochastic asynchronous Proximal Alternating Linearized Minimization (PALM) algorithm

$$\underset{(x_1,\ldots,x_m)\in\mathcal{H}_1\times\ldots\times\mathcal{H}_m}{\text{minimize}} \; f(x_1,\ldots,x_m) + \sum_{j=1}^{m} r_j(x_j)$$

- f is a smooth, C^1 function, it can be the data fidelity term
- r is nonsmooth, it can be a structrural regularizer of the solution

The paper's idea facilitates parallel computation:
- assign the computation of each x_i to a different processor

Algorithm 1 SAPALM [Local view]

Input: $x \in \mathcal{H}$

All processors in parallel do

loop

  Randomly select a coordinate block $j \in \{1, \ldots, m\}$

  Read x from shared memory

  Compute $g = \nabla_j f(x) + \nu_j$

  Choose stepsize $\gamma_j \in \mathbb{R}_{++}$

  $x_j \leftarrow \mathbf{prox}_{\gamma_j r_j}(x_j - \gamma_j g)$

Main features:

- Inconsistent iterates. Other processors may write updates to x in the time required to read x from memory.
- Coordinate blocks. When the coordinate blocks x_j are low dimensional, it reduces the likelihood that one update will be immediately erased by another, simultaneous update.
- Noise. The noise ν ∈ H is a random variable that we use to model injected noise. It can be set to 0, or chosen to accelerate each iteration, or to avoid saddle points

Algorithm 2 SAPALM [Global view]

Input: $x^0 \in \mathcal{H}$

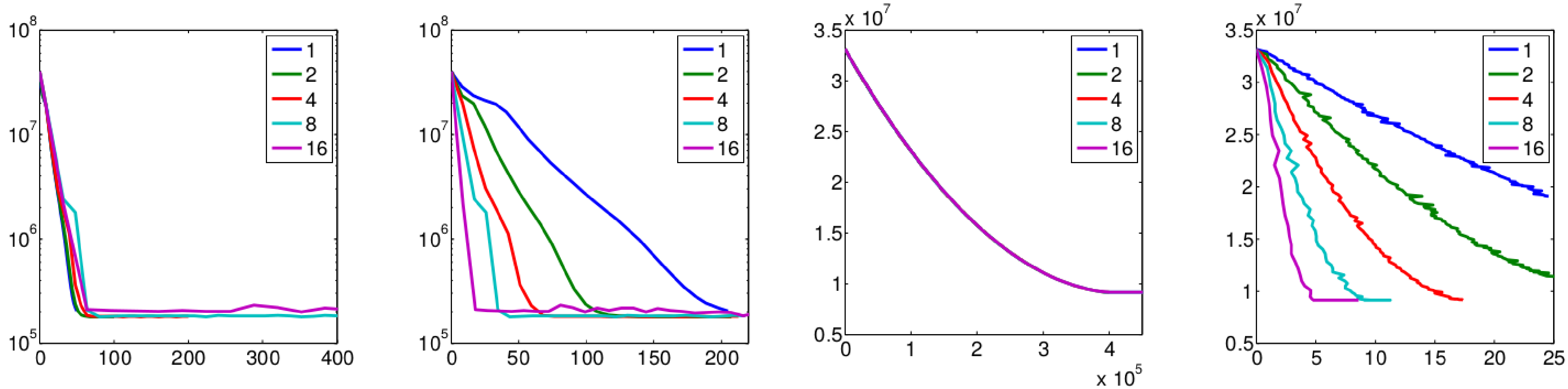
for $k \in \mathbb{N}$ **do**
  Randomly select a coordinate block $j_k \in \{1, \ldots, m\}$
  Read $x^{k-d_k} = (x_1^{k-d_{k,1}}, \ldots, x_m^{k-d_{k,m}})$ from shared memory
  Compute $g^k = \nabla_{j_k} f(x^{k-d_k}) + \nu_{j_k}^k$
  Choose stepsize $\gamma_{j_k}^k \in \mathbb{R}_{++}$
  for $j = 1, \ldots, m$ **do**
    if $j = j_k$ **then**
      $x_{j_k}^{k+1} \leftarrow \mathbf{prox}_{\gamma_{j_k}^k r_{j_k}}(x_{j_k}^k - \gamma_{j_k}^k g^k)$
    else
      $x_j^{k+1} \leftarrow x_j^k$

Stochastic Gradients: Noise due to stochastic approximations of delayed gradients.

- it allows us to prove convergence for a stochastic- or minibatch-gradient version of APALM, rather than requiring processors to compute a full (delayed) gradient.
- Stochastic gradients can be computed faster than their batch counterparts, allowing more frequent updates

Convergence theorem proves that the SAPALM sequence is summable and α-diminishing (expected error is below a threshold)

Numerical experiments



(a) Iterates vs objective (b) Time (s) vs. objective (c) Iterates vs. objective (d) Time (s) vs. objective

- Sparce PCA

$$\underset{X,Y}{\operatorname{argmin}} \frac{1}{2}||A - X^T Y||_F^2 + \lambda \|X\|_1 + \lambda \|Y\|_1$$

- Quadratically Regularized Firm Thresholding PCA with Asynchronous Stochastic Gradients

$$\underset{X,Y}{\operatorname{argmin}} \frac{1}{2}||A - X^T Y||_F^2 + \lambda(\|X\|_{\mathrm{Firm}} + \|Y\|_{\mathrm{Firm}}) + \frac{\mu}{2}(\|X\|_F^2 + \|Y\|_F^2)$$


That's all Folks!

Proximal operator over σ

$$\operatorname{prox}_t^{\sigma}(x) := \operatorname{argmin}\left\{\sigma(u) + \frac{t}{2}\|u - x\|^2 :\ u \in \mathbb{R}^d\right\}$$

Moreau proximal envelope associated to σ

$$m^{\sigma}(x,t) := \inf\left\{\sigma(u) + \frac{1}{2t}\|u - x\|^2 :\ u \in \mathbb{R}^d\right\}$$