The Proximal Alternating Linearized Minimization (PALM) algorithm

Minimization of a sum of finite functions

$$\min_{x,y} \Psi(x, y) := f(x) + g(y) + H(x, y)$$

- \(f(x), g(y)\) 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

$$x^{k+1} \in \arg\min_x \Psi(x, y^k)$$
$$y^{k+1} \in \arg\min_y \Psi(x^{k+1}, y)$$

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

\[ x^{k+1} \in \arg\min_x \left\{ \Psi (x, y^k) + \frac{c_k}{2} \| x - x^k \|^2 \right\} \]

\[ y^{k+1} \in \arg\min_y \left\{ \Psi (x^{k+1}, y) + \frac{d_k}{2} \| y - y^k \|^2 \right\} \]

- 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 function with smooth $h(x)$ and nonsmooth $\sigma(x)$.

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

A proximal forward-backward scheme is expressed as

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

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

PALM’s main idea: replace $\Psi$ with its linearization

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

$$\hat{\Psi}(x^{k+1}, y) = \left\langle y - y^k, \nabla_y H(x^{k+1}, y^k) \right\rangle + \frac{d_k}{2} \| y - y^k \|^2 + g(y), \quad (d_k > 0)$$
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 \(\{(x^k, y^k)\}_{k \in \mathbb{N}}\) as follows:

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

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

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

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

- 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 \(\Psi\).
- Convergence results for PALM exist for the case that the function \(\Psi\) to be minimized satisfies the so-called Kurdyka-Lojasiewicz (KL) property (sharpness of the gradient near critical points)
Example: PALM for BSS

\[
\min_{A, \alpha} \frac{1}{2} \|X - A\alpha\Phi^*\|_F^2 + \lambda \|\alpha\|_1 + \nu_C(A)
\]

\[
S = \alpha\Phi^* \quad \alpha \in \mathbb{R}^{N \times K}
\]

[Feng & Kowalski, 2018]

**Algorithm 1: BSS-PALM**

**Initialization:** \(\alpha^{(1)} \in \mathbb{R}^{N \times K}, A^{(1)} \in \mathbb{R}^{M \times N}, L_1^{(1)} = \|A^{(1)}\|_2^2, L_2^{(1)} = \|\alpha^{(1)}\Phi^*\|_2^2\), \(j = 1\);

repeat

1. \(\nabla_\alpha Q(\alpha^{(j)}, A^{(j)}) = -A^{(j)^T}(X - A^{(j)}\alpha^{(j)}\Phi^*)\Phi\);
2. \(\alpha^{(j+1)} = S_{\lambda/L_1^{(j)}}(\alpha^{(j)} - \frac{1}{L_1^{(j)}} \nabla_\alpha Q(\alpha^{(j)}, A^{(j)}))\);
3. \(\nabla_A Q(\alpha^{(j+1)}, A^{(j)}) = -(X - A^{(j)}\alpha^{(j+1)}\Phi^*)\Phi\alpha^{(j+1)H}\);
4. \(A^{(j+1)} = \mathcal{P}_C(A^{(j)} - \frac{1}{L_2^{(j)}} \nabla_A Q(\alpha^{(j+1)}, A^{(j)}))\);
5. \(L_1^{(j+1)} = \|A^{(j+1)}\|_2^2\);
6. \(L_2^{(j+1)} = \|\alpha^{(j+1)}\Phi^*\|_2^2\);
7. \(j = j + 1\);

until convergence;
The stochastic asynchronous Proximal Alternating Linearized Minimization (PALM) algorithm

\[
\begin{align*}
\text{minimize} & \quad f(x_1, \ldots, x_m) + \sum_{j=1}^{m} r_j(x_j) \\
(x_1, \ldots, x_m) & \in \mathcal{H}_1 \times \ldots \times \mathcal{H}_m
\end{align*}
\]

- \( f \) is a smooth, \( C^1 \) function, it can be the data fidelity term
- \( r \) is nonsmooth, it can be a structural 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} \\
1: \text{All processors in parallel do} \\
2: \textbf{loop} \\
3: \quad \text{Randomly select a coordinate block } j \in \{1, \ldots, m\} \\
4: \quad \text{Read } x \text{ from shared memory} \\
5: \quad \text{Compute } g = \nabla_j f(x) + \nu_j \\
6: \quad \text{Choose stepsize } \gamma_j \in \mathbb{R}_{++} \\
7: \quad x_j \leftarrow \text{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 \( \nu \in \mathcal{H} \) 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
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 $\alpha$-diminishing (expected error is below a threshold)
Numerical experiments

- Sparse PCA
  \[
  \arg \min_{X,Y} \frac{1}{2} \|A - X^T Y\|_F^2 + \lambda \|X\|_1 + \lambda \|Y\|_1
  \]

- Quadratically Regularized Firm Thresholding PCA with Asynchronous Stochastic Gradients
  \[
  \arg \min_{X,Y} \frac{1}{2} \|A - X^T Y\|_F^2 + \lambda (\|X\|_{\text{Firm}} + \|Y\|_{\text{Firm}}) + \frac{\mu}{2} (\|X\|_F^2 + \|Y\|_F^2)
  \]
That's all Folks!
Proximal operator over $\sigma$

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

Moreau proximal envelope associated to $\sigma$

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