FISTA algorithm: so fast ?

Mathis Scheffler and Constantin Vaillant-Tenzer

January 8, 2025

Objectives of the FISTA Algorithm

- Minimize a function of the form:

$$\min_{x} \left\{ F(x) = f(x) + g(x) \right\}$$

- $x \in \mathscr{R}^d$
- $f(x)$: convex and smooth (differentiable) function with L-Lipschitz gradient
- $g(x)$: convex, and lower-semi continuous (potentially non-smooth and non continuous function) function

Algorithm Steps : FISTA

Algorithm 1 FISTA

Initialize: $y_1 = x_0$, $t_1 = 1$

for $k = 1, 2, \dots$ **do**

$x_k = \text{prox}_{\lambda g}(y_k - \lambda \nabla f(y_k))$

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

$y_{k+1} = x_k + (\frac{t_k - 1}{t_{k+1}})(x_k - x_{k-1})$

end for

$$\text{Prox}_{\gamma,g}(\tau) :=_{\theta\in\Theta} \left(g(\theta) + \frac{1}{2\gamma}\|\theta-\tau\|^2\right)$$

What happens if ∇f is untracktable ?

- Suppose that ∇f is untracktable and that :

$$\nabla f(\theta) = \int_X H(\theta, x)\, \pi_\theta(dx)$$

- We can approximate stochastically ∇f !

Stochastic Fista or Perturbed Fista (P-FISTA)

- Classic Fista:

$$x_k = \mathrm{prox}_{\gamma g}\left(y_k - \gamma \nabla f(y_k)\right)$$

- p-Fista:

$$x_k = \mathrm{prox}_{\gamma g}\left(y_k - \gamma H_{n+1}\right)$$

with H_{n+1} a stochastic approximation of $\nabla f(y_k)$ obtained with MCMC

Does P-FISTA theoretically works ?

Assumptions :

1. $$\mathscr{L} = \arg\min_x \left(f + g\right) \neq 0$$

2. $$t_0 = 1; t_n \geq 1; \gamma_n \in \left]0, \frac{1}{L}\right]; \tau_n := \gamma_n t_{n-1}^2 - \gamma_{n+1} t_n (t_n - 1) > 0.$$

3. .

   With

   $$z_n := \theta_n + t_n \left(\mathrm{Prox}_{\gamma_{n+1}, g}\left(\vartheta_n - \gamma_{n+1} \nabla f(\vartheta_n)\right) - \theta_n\right)$$

   and

   $$\eta_{n+1} := H_{n+1} - \nabla f(\vartheta_n)$$

   Then

   The series $\sum_n \gamma_{n+1} t_n \langle z_n - \theta_*, \eta_{n+1} \rangle$ exists for some $\theta_* \in \mathscr{L}$.

Does P-FISTA theoretically works ?

Results:

1. The algorithm converges to a minimum of f+g
2. Under conditions such as the following :
   - $(a \in [0,1], c = 3-2a)$ or $(a \in [1,2[, c = 2-a)$and$b > 1$
   - t_n = O(n)
   - $\gamma_n = \gamma n^{-a}$
   - $m_n = m(\ln n)^b n^c$ = Samples to estimate H_n

   We obtain convergence at rate n^{a-2}. The optimal rate is Fista's rate ie. $0(n^{-2})$ but taking a = 0 leads to a huge computation as it would imply that $n^3 = o(m_n)$.

Numerical Results Implementation

Performance of FISTA:

- Slow in Python (3,5 days on an i7-12700K processor) - with $p = 10$ and $N = 25$.
- Bottleneck: Large sample size.

Processing Time Distribution:

- Algo 1 (P-PG, $m_n = O(\sqrt{n})$, $t_n = 1$): 4%
- Algo 2 (P-FISTA, $m_n = O(n^3)$, $t_n = O(n)$): 19%
- Algo 3 (P-FISTA, $m_n = O(n^3)$, $t_n = O(\sqrt{n})$: 31%
- Algo 4 (P-FISTA, $m_n = O(n^3)$, $t_n = O(n^\epsilon)$: 45%
- Algo 5 (P-PG, $m_n = 10$, S_n computed iteratively): 1%

Limitations:

- Convergence threshold: 10^{-2} (not all 2000 steps completed).
- Sample cap: 10,000 (growth stops after 22 iterations in algorithms 2–4).

Optimization Attempts:

- Parallelization with JAX: Faster but limited.
- Potential C implementation: Estimated ×87 speedup (Heer and al. 2023).

Numerical analysis

Study of the sparcity of the non-zero components of θ_n:

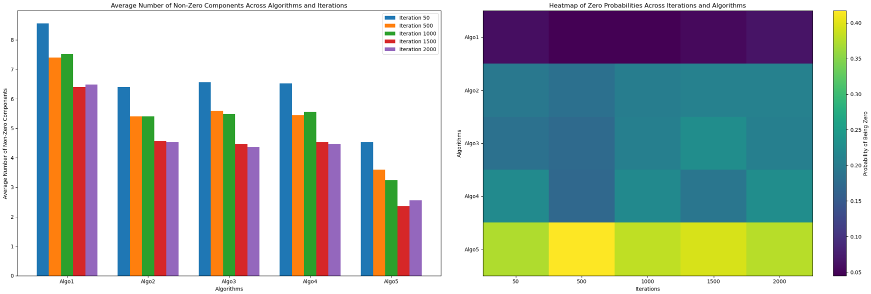


Figure: [left]: number of non-zeros components θ_n, [right]: probability to be non-nul for each component avec n iterations.

Our code is available here: `https://github.com/cvt8/fista_sofast`.

References I

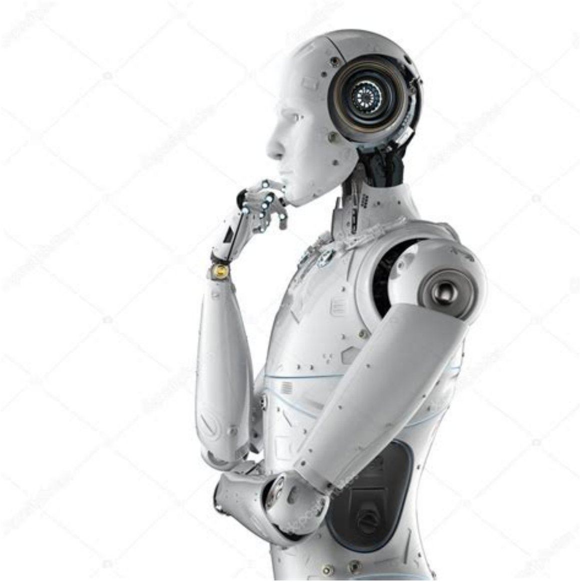
Beck, Amir and Marc Teboulle (2009). “A fast iterative shrinkage-thresholding algorithm for linear inverse problems”. In: *SIAM journal on imaging sciences* 2.1, pp. 183–202.

Fort, Gersende et al. (2018). “Stochastic FISTA algorithms: so fast?” In: *2018 IEEE Statistical Signal Processing Workshop (SSP)*. IEEE, pp. 796–800.

Heer, Niklas and et al. (2023). *Speed Comparison of Programming Languages.* Accessed: 2025-01-05. URL: `https://github.com/niklas-heer/speed-comparison`.

Wolff, Ulli (1989). “Collective Monte Carlo updating for spin systems”. In: *Physical Review Letters* 62.4, p. 361.

Questions



Metropolis-Hastings Algorithm

- Goal: Sample from a target distribution $p(x)$ when direct sampling is difficult
- Algorithm Steps:
  1. Start with initial state x_t
  2. Propose new state x' from proposal distribution $q(x'|x_t)$
  3. Calculate acceptance ratio:

$$\alpha = \min\left(1, \frac{p(x')q(x_t|x')}{p(x_t)q(x'|x_t)}\right)$$

  4. Accept x' with probability α:
     - If accepted: $x_{t+1} = x'$
     - If rejected: $x_{t+1} = x_t$
- Key feature: Only needs to know $p(x)$ up to a normalizing constant
- Converges to the target distribution as $t \to \infty$

Gibbs Sampling

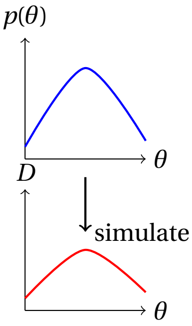
- Special case of MH with acceptance rate always 1
- Used when sampling from conditional distributions is easier
- For variables $\mathbf{x} = (x_1, \ldots, x_n)$:
  1. Initialize $\mathbf{x}^{(0)}$
  2. For each iteration t:

$$
\begin{aligned}
x_1^{(t+1)} &\sim p(x_1 | x_2^{(t)}, x_3^{(t)}, \ldots, x_n^{(t)}) \\
x_2^{(t+1)} &\sim p(x_2 | x_1^{(t+1)}, x_3^{(t)}, \ldots, x_n^{(t)}) \\
&\vdots \\
x_n^{(t+1)} &\sim p(x_n | x_1^{(t+1)}, \ldots, x_{n-1}^{(t+1)})
\end{aligned}
$$

- Advantages:
  - No tuning of proposal distributions needed
  - Higher acceptance rate than MH
  - Works well for high-dimensional problems

Approximate Bayesian Computation (ABC)

Key Idea: Approximate posterior when likelihood is intractable



Algorithm:

1. Sample $\theta^* \sim p(\theta)$ from prior
2. Simulate data $D^* \sim p(D|\theta^*)$
3. Compute summary statistics $S^* = S(D^*)$
4. Accept if $\rho(S^*, S_{obs}) \leq \varepsilon$

Key Components:

- Summary statistics $S(\cdot)$
- Distance metric $\rho(\cdot, \cdot)$
- Tolerance threshold ε

Variants:

- ABC-MCMC: Uses Markov Chain exploration

Wolf Sampling (Wolff 1989)

Context: Designed for Monte Carlo simulations of spin-lattice systems (e.g., Ising model).
Uses cluster updates to improve efficiency.

Main Steps:

1. **Initialization:** Select a random spin s_0 as the cluster seed.
2. **Cluster Growth:**
   - Add aligned neighboring spins to the cluster with probability $P_{\text{add}} = 1 - e^{-2\beta J}$, where β is the inverse temperature and J the interaction energy.
3. **Cluster Flip:** Flip all spins in the cluster.

Advantages:

- Reduces correlations between successive configurations.
- Effective near the critical temperature.

Numerical Simulations: Overview

- Goal: Compare five algorithms in optimizing a likelihood function in binary graphical models.
- Setting:
  - Sparse matrix θ with $p = 100, N = 250$.
  - Penalty term: $g(\theta) = \lambda \sum |\theta_{ij}| + \mu \sum \theta_i^2$.
- Algorithms:
  - Alg1: Perturbed Proximal Gradient (P-PG).
  - Alg2–4: Variants of Perturbed FISTA (P-FISTA).
  - Alg5: Averaging-based Perturbed Proximal Gradient.

Algorithm 1: Perturbed Proximal Gradient (P-PG)

- Update rule:
$$\theta_{n+1} = \mathrm{Prox}_{\gamma_n, g}\left(\theta_n - \gamma_n H_n\right),$$
where H_n is a Monte Carlo approximation of the gradient.
- Parameters:
  - Step size: $\gamma_n = O(1/\sqrt{n})$.
  - Sample size: $m_n = O(\sqrt{n})$.
- Results:
  - Convergence rate: $O(1/n)$ (logarithmic terms ignored).
  - Baseline for comparison.

Algorithm 2: P-FISTA with $t_n = O(n)$

- Accelerated update rule:

$$\theta_{n+1} = \mathrm{Prox}_{\gamma_n, g}\left(\varphi_n - \gamma_n H_n\right),$$

  where $\varphi_n = \theta_n + \frac{t_{n-1}-1}{t_n}(\theta_n - \theta_{n-1})$.
- Parameters:
  - Step size: $\gamma_n = O(1)$.
  - Sample size: $m_n = O(n^3)$.
  - Momentum: $t_n = O(n)$.
- Results:
  - In practice, slower convergence than Alg1.
  - Optimal convergence rate $O(1/n^2)$.

Algorithm 3: P-FISTA with $t_n = O(\sqrt{n})$

- Same update rule as Algorithm 2.
- Parameters:
  - Step size: $\gamma_n = O(1)$.
  - Sample size: $m_n = O(n^3)$.
  - Momentum: $t_n = O(\sqrt{n})$.
- Results:
  - Convergence slower than $t_n = O(n)$ but better than Alg1.
  - Computational cost remains high due to $m_n = O(n^3)$.

Algorithm 4: P-FISTA with $t_n = O(n^\epsilon)$

- Same update rule as Algorithm 2.
- Parameters:
  - Step size: $\gamma_n = O(1)$.
  - Sample size: $m_n = O(n^3)$.
  - Momentum: $t_n = O(n^\epsilon), \epsilon \ll 1$.
- Results:
  - Practical trade-off between convergence speed and stability.
  - The slower convergence in practice
  - Requires further theoretical analysis.

Algorithm 5: Averaging-based P-PG

- Update rule:

$$S_{n+1} = (1-\delta_{n+1})S_n + \delta_{n+1}\frac{1}{m_n}\sum_{j=1}^{m_n} S(X_j),$$

  where $\delta_n = O(n^{-0.9})$.
- Parameters:
  - Sample size: $m_n = O(1)$.
  - Uses all past samples iteratively.
- Results:
  - Improves convergence rate significantly.
  - Best performance in the simulations.