From Extrapolation to Quasi-Newton:

Stabilizing Type-I Anderson Mixing for Memory-Efficient, Line-Search Free and Black-Box Acceleration

Junzi Zhang

Stanford ICME, *junziz@stanford.edu*

Joint works with Brendan O'Donoghue, Anqi Fu, Stephen P. Boyd
Xin Guo, Anran Hu and Renyuan Xu

June 14, 2019

Overview

Fixed-point problems

- We consider solving a fixed-point problem $x = f(x)$, where $f : \mathbb{R}^n \to \mathbb{R}^n$ is potentially non-smooth.

[1] $\|x\|_H = \sqrt{x^T H x}$ for some PSD matrix H

Fixed-point problems

- We consider solving a fixed-point problem $x = f(x)$, where $f : \mathbb{R}^n \to \mathbb{R}^n$ is potentially non-smooth.
- **Assumption**: f is **non-expansive** in l_2 (or H-norm[1]), *i.e.*,

$$\|f(x) - f(y)\|_2 \leq \|x - y\|_2 \text{ for any } x,\ y \in \mathbb{R}^n$$

  or **contractive** in an arbitrary norm $\|\cdot\|$.
- Simplest solution: averaged iteration, a.k.a. Krasnosel'skiĭ-Mann (KM) iteration

$$x^{k+1} = (1-\alpha)x^k + \alpha f(x^k),\ \alpha \in (0,1).$$

- Convergence is robust, but sublinear in theory and slow in practice: can we **(safely)** do better?

[1] $\|x\|_H = \sqrt{x^T H x}$ for some PSD matrix H

Why non-smooth non-expansive fixed-point problems?

Many (potentially complicated) algorithms in optimization and beyond can be reformulated as **"black-box" fixed-point** problems.

Examples:

- (Any) convex optimization with no strong convexity
  - $\text{minimize}_{x \in C}\ F(x)$, C is convex, F is convex and L-strongly smooth.

Why non-smooth non-expansive fixed-point problems?

Many (potentially complicated) algorithms in optimization and beyond can be reformulated as **"black-box" fixed-point** problems.

Examples:

- (Any) convex optimization with no strong convexity
  - $\text{minimize}_{x\in C}\ F(x)$, C is convex, F is convex and L-strongly smooth.
  - Projected gradient descent: $x^{k+1} = \Pi_C\left(x^k - \frac{1}{L}\nabla F(x^k)\right)$.

Why non-smooth non-expansive fixed-point problems?

Many (potentially complicated) algorithms in optimization and beyond can be reformulated as **"black-box" fixed-point** problems.

Examples:

- (Any) convex optimization with no strong convexity
  - $\text{minimize}_{x\in C}\ F(x)$, C is convex, F is convex and L-strongly smooth.
  - Projected gradient descent: $x^{k+1} = \Pi_C\left(x^k - \frac{1}{L}\nabla F(x^k)\right)$.
  - Optimality $\Leftrightarrow$ $x = f(x)$, $f(x) := \Pi_C\left(x - \frac{1}{L}\nabla F(x)\right)$.

Why non-smooth non-expansive fixed-point problems?

Many (potentially complicated) algorithms in optimization and beyond can be reformulated as **"black-box" fixed-point** problems.

Examples:

- (Any) convex optimization with no strong convexity
  - $\text{minimize}_{x \in C}\ F(x)$, C is convex, F is convex and L-strongly smooth.
  - Projected gradient descent: $x^{k+1} = \Pi_C\left(x^k - \frac{1}{L}\nabla F(x^k)\right)$.
  - Optimality $\Leftrightarrow$ $x = f(x)$, $f(x) := \Pi_C\left(x - \frac{1}{L}\nabla F(x)\right)$.
  - Projection is non-differentiable and non-expansive, but non-contractive without strong convexity.

Why non-smooth non-expansive fixed-point problems?

Many (potentially complicated) algorithms in optimization and beyond can be reformulated as **"black-box" fixed-point** problems.

Examples:

- Discounted Markov decision processes (MDP)
  - Value iteration: $x^{k+1} = Tx^k$, where T is the Bellman operator:

$$(Tx)_s = \max_{a=1,\ldots,A} R(s,a) + \gamma \sum_{s'=1}^{S} P(s,a,s')x_{s'}.$$

  - Optimality $\Leftrightarrow x = Tx$.
  - Contractive in l_∞, but still non-differentiable due to max.

Why non-smooth non-expansive fixed-point problems?

Many (potentially complicated) algorithms in optimization and beyond can be reformulated as **"black-box" fixed-point** problems.

Examples:

- Nash equilibrium in a multiplayer game $\Leftrightarrow$ monotone inclusion problem $\Leftrightarrow$ non-smooth non-expansive fixed-point problem.

Acceleration by extrapolation

Algorithm 1 Extrapolation framework

Input: initial point x_0, fixed-point mapping $f : \mathbb{R}^n \to \mathbb{R}^n$.
for $k = 0, 1, \ldots$ **do**
  Choose m_k (*e.g.*, $m_k = \min\{m, k\}$ for some integer $m \geq 0$).
  Select weights α_j^k based on the last m_k iterations, with $\sum_{j=0}^{m_k} \alpha_j^k = 1$.
  $x^{k+1} = \sum_{j=0}^{m_k} \alpha_j^k f(x^{k-m_k+j})$.

Such a framework subsumes many different algorithms, among which one of the most natural and popular method is Anderson acceleration (1965):

$$\text{minimize} \quad \|\sum_{j=0}^{m_k} \alpha_j g(x^{k-m_k+j})\|_2^2 \quad \text{subject to} \quad \sum_{j=0}^{m_k} \alpha_j = 1,$$

where $g(x) := x - f(x)$ is the residual.

Literature comments

- Also known as **Type-II Anderson acceleration** (AA-II), Anderson/Pulay mixing, Pulay's direct inversion iterative subspace (DIIS), nonlinear GMRES, minimal polynomial extrapolation (MPE), reduced rank extrapolation (RRE), etc.

Literature comments

- Also known as **Type-II Anderson acceleration** (AA-II), Anderson/Pulay mixing, Pulay's direct inversion iterative subspace (DIIS), nonlinear GMRES, minimal polynomial extrapolation (MPE), reduced rank extrapolation (RRE), etc.
- Widely used in computational quantum chemistry and material sciences, and recently introduced to optimization applications
  - MLE, matrix completion, K-means, computer vision and deep learning.

Literature comments

- Also known as **Type-II Anderson acceleration** (AA-II), Anderson/Pulay mixing, Pulay's direct inversion iterative subspace (DIIS), nonlinear GMRES, minimal polynomial extrapolation (MPE), reduced rank extrapolation (RRE), etc.
- Widely used in computational quantum chemistry and material sciences, and recently introduced to optimization applications
  - MLE, matrix completion, K-means, computer vision and deep learning.
- Equivalent to **multi-secant quasi-Newton** methods (see below) – development separated from the main-stream, connection established very recently in Fang and Saad 2009.
  - Extrapolation: good for intuition.
  - Quasi-Newton: good for derivations.

From extrapolation to quasi-Newton

- Recall the selection of α_j^k in AA-II (constrained least-squares):

$$\text{minimize} \quad \|\textstyle\sum_{j=0}^{m_k} \alpha_j g(x^{k-m_k+j})\|_2^2 \quad \text{subject to} \quad \textstyle\sum_{j=0}^{m_k} \alpha_j = 1,$$

- Reformulation: minimize $\|g_k - Y_k\gamma\|_2$
  - variable $\gamma = (\gamma_0, \ldots, \gamma_{m_k-1})$.
  - $g_i = g(x^i)$, $Y_k = [y_{k-m_k} \ \ldots \ y_{k-1}]$ with $y_i = g_{i+1} - g_i$ for each i.
  - $\alpha_0 = \gamma_0$, $\alpha_i = \gamma_i - \gamma_{i-1}$ for $1 \leq i \leq m_k - 1$ and $\alpha_{m_k} = 1 - \gamma_{m_k-1}$.

From extrapolation to quasi-Newton

- Recall the selection of α_j^k in AA-II (constrained least-squares):

$$\text{minimize} \quad \|\sum_{j=0}^{m_k} \alpha_j g(x^{k-m_k+j})\|_2^2 \quad \text{subject to} \quad \sum_{j=0}^{m_k} \alpha_j = 1,$$

- Reformulation: minimize $\|g_k - Y_k\gamma\|_2$
  - variable $\gamma = (\gamma_0, \ldots, \gamma_{m_k-1})$.
  - $g_i = g(x^i)$, $Y_k = [y_{k-m_k} \ \ldots \ y_{k-1}]$ with $y_i = g_{i+1} - g_i$ for each i.
  - $\alpha_0 = \gamma_0$, $\alpha_i = \gamma_i - \gamma_{i-1}$ for $1 \leq i \leq m_k - 1$ and $\alpha_{m_k} = 1 - \gamma_{m_k-1}$.
- $x^{k+1} = \sum_{j=0}^{m_k} \alpha_j^k f(x^{k-m_k+j}) = x^k - H_k g_k$,
  - $H_k := I + (S_k - Y_k)(Y_k^T Y_k)^{-1} Y_k^T$.
  - $H_k = \text{argmin}_{HY_k=S_k} \|H - I\|_F$: **approximate inverse Jacobian** of g.
  - multi-secant type-II **(bad)** Broyden's (quasi-Newton) method.

Type-I Anderson acceleration

- Why not consider the **type-I (good)** counterpart?

Type-I Anderson acceleration

- Why not consider the **type-I (good)** counterpart?
- Instead of inverse Jacobian (which itself **may not exist**), consider $B_k := \text{argmin}_{BS_k = Y_k} \|B_k - I\|_F$: approximate Jacobian of g.
- $x^{k+1} = x^k - B_k^{-1} g_k$, with $B_k^{-1} = I + (S_k - Y_k)(S_k^T Y_k)^{-1} S_k^T$.

Type-I Anderson acceleration

- Why not consider the **type-I (good)** counterpart?
- Instead of inverse Jacobian (which itself **may not exist**), consider $B_k := \text{argmin}_{BS_k=Y_k} \|B_k - I\|_F$: approximate Jacobian of g.
- $x^{k+1} = x^k - B_k^{-1} g_k$, with $B_k^{-1} = I + (S_k - Y_k)(S_k^T Y_k)^{-1} S_k^T$.

Algorithm 2 Type-I Anderson Acceleration (AA-I)

1: **for** $k = 0, 1, \ldots$ **do**
2: Choose $m_k \leq m$ (e.g., $m_k = \min\{m, k\}$ for some integer $m \geq 0$).
3: Compute $\tilde{\gamma}^k = (S_k^T Y_k)^{-1}(S_k^T g_k)$.
4: $\alpha_0^k = \tilde{\gamma}_0^k$, $\alpha_i^k = \tilde{\gamma}_i^k - \tilde{\gamma}_{i-1}^k$ $(1 \leq i \leq m_k - 1)$ and $\alpha_{m_k}^k = 1 - \tilde{\gamma}_{m_k-1}^k$.
5: $x^{k+1} = \sum_{j=0}^{m_k} \alpha_j^k f(x^{k-m_k+j})$.

Good news and bad news

Good news:

- Compared to **AA-II**: early experiments applying AA to SCS (a popular convex optimization solver) show obvious advantage of AA-I over AA-II on some benchmark problems.

Good news and bad news

Good news:

- Compared to **AA-II**: early experiments applying AA to SCS (a popular convex optimization solver) show obvious advantage of AA-I over AA-II on some benchmark problems.
- Compared to **LBFGS** and **restarted Broyden**:
    - AA is *memory efficient* (AA-I with $m = 5 - 10$ beats LBFGS/restarted Broyden with $m = 200 - 500$)

Good news and bad news

Good news:

- Compared to **AA-II**: early experiments applying AA to SCS (a popular convex optimization solver) show obvious advantage of AA-I over AA-II on some benchmark problems.
- Compared to **LBFGS** and **restarted Broyden**:
  - AA is *memory efficient* (AA-I with $m = 5 - 10$ beats LBFGS/restarted Broyden with $m = 200 - 500$)
  - AA is *line-search free*: just accept or reject is the best practice

Good news and bad news

Good news:

- Compared to **AA-II**: early experiments applying AA to SCS (a popular convex optimization solver) show obvious advantage of AA-I over AA-II on some benchmark problems.
- Compared to **LBFGS** and **restarted Broyden**:
  - AA is *memory efficient* (AA-I with $m = 5 - 10$ beats LBFGS/restarted Broyden with $m = 200 - 500$)
  - AA is *line-search free*: just accept or reject is the best practice
  - AA is suitable to be used in a completely *black-box* way
    - PGD: don't separate the gradient step and projection
    - ADMM: don't separate the primal and dual steps

Good news and bad news

Good news:

- Compared to **AA-II**: early experiments applying AA to SCS (a popular convex optimization solver) show obvious advantage of AA-I over AA-II on some benchmark problems.
- Compared to **LBFGS** and **restarted Broyden**:
  - AA is *memory efficient* (AA-I with $m = 5 - 10$ beats LBFGS/restarted Broyden with $m = 200 - 500$)
  - AA is *line-search free*: just accept or reject is the best practice
  - AA is suitable to be used in a completely *black-box* way
    - PGD: don't separate the gradient step and projection
    - ADMM: don't separate the primal and dual steps
- **SCS** itself is a non-smooth and non-expansive fixed-point iteration.

Good news:

- Compared to **AA-II**:

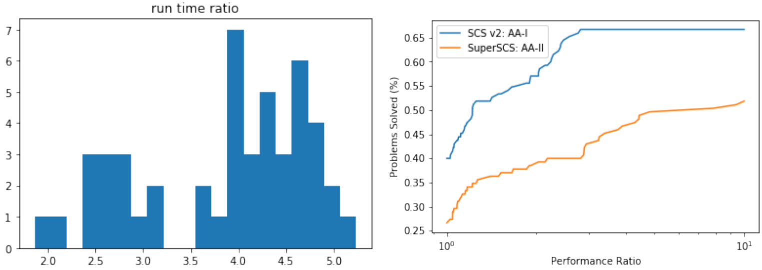
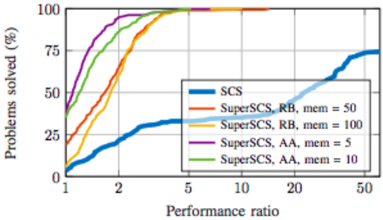


Figure: Left: histogram of run time ratio between SuperSCS (AA-II) and SCS v2 (AA-I). Right: DM profile of run time.

Good news and bad news

Good news:

- Compared to **restarted Broyden**::



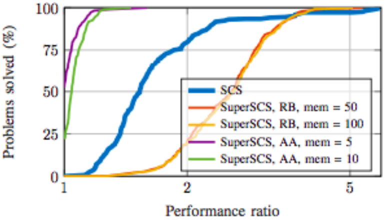


Figure: DM profile. left: sparse PCA; right: sparse logistic regression. *SuperSCS: fast and accurate large-scale conic optimization.* Sopasakis, et al., 2019.

Good news and bad news

Bad news:

- **Numerical challenge**: both AA-I and AA-II are subject to potential *numerical instability*, and AA-I is more severe.
  - AA-II: $Y_k^T Y_k$ (close to) singular (degenerate least-squares system).
  - AA-I: B_k can be (close to) singular.

Good news and bad news

Bad news:

- **Numerical challenge**: both AA-I and AA-II are subject to potential *numerical instability*, and AA-I is more severe.
  - AA-II: $Y_k^T Y_k$ (close to) singular (degenerate least-squares system).
  - AA-I: B_k can be (close to) singular.
- **Theoretical challenge**: local convergence theory exists with further smoothness assumptions, but *no global convergence*.

Good news and bad news

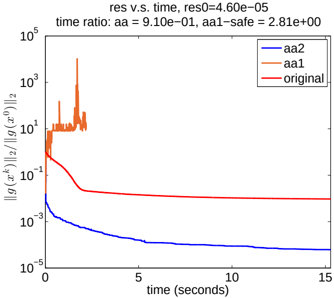
Bad news:

- **Numerical challenge**: both AA-I and AA-II are subject to potential *numerical instability*, and AA-I is more severe.
  - AA-II: $Y_k^T Y_k$ (close to) singular (degenerate least-squares system).
  - AA-I: B_k can be (close to) singular.
- **Theoretical challenge**: local convergence theory exists with further smoothness assumptions, but *no global convergence*.
- In general, most of the literature has been focused on AA-II:
  - AA-I is generally *missing both in theory and practice*.

Good news and bad news

Bad news:

- **Numerical challenge**: both AA-I and AA-II are subject to potential *numerical instability*, and AA-I is more severe.



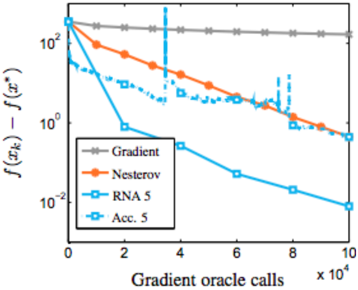


Figure: Convergence of Anderson accelerated gradient descent on ℓ_2 regularized logistic regression without stabilization. Left: AA-I vs AA-II. Right: AA-II v.s. stabilized AA-II (*Regularized Nonlinear Acceleration*, Scieur et al., 2016.)

Goal and contribution

- **Stabilize** AA-I with convergence beyond differentiability, locality and non-singularity
  - **Surprise:** stabilization also improves convergence consistently over both the original AA-I and AA-II.

Goal and contribution

- **Stabilize** AA-I with convergence beyond differentiability, locality and non-singularity
  - **Surprise:** stabilization also improves convergence consistently over both the original AA-I and AA-II.
- Develop a "plug-and-play" acceleration scheme based on the stabilized AA-I
  - View an arbitrary unaccelerated algorithm as a **black-box** fixed-point iteration problem.
  - For example, concatenate successive iterates in momentum algorithms.

Stabilization of AA-I: rank-one update

AA-I $\Longleftrightarrow$ Type-I Broyden's rank-one update with **orthogonalization**:

Proposition

Suppose that S_k is full rank, then B_k can be computed inductively from $B_k^0 = I$ as follows:

$$B_k^{i+1} = B_k^i + \frac{(y_{k-m_k+i} - B_k^i s_{k-m_k+i})\hat{s}_{k-m_k+i}^T}{\hat{s}_{k-m_k+i}^T s_{k-m_k+i}}, \quad i = 0, \dots, m_k - 1$$

with $B_k = B_k^{m_k}$. Here $\{\hat{s}_i\}_{i=k-m_k}^{k-1}$ is the Gram-Schmidt orthogonalization of $\{s_i\}_{i=k-m_k}^{k-1}$, i.e., $\hat{s}_i = s_i - \sum_{j=k-m_k}^{i-1} \frac{\hat{s}_j^T s_i}{\hat{s}_j^T \hat{s}_j}\hat{s}_j$, $\quad i = k - m_k, \dots, k-1$.

Stabilization of AA-I: 1. Powell-type regularization

Goal of regularization: avoid close to singularity ("lower bound" on B_k).

Stabilization of AA-I: 1. Powell-type regularization

Goal of regularization: avoid close to singularity ("lower bound" on B_k).

- AA-II: add ridge penalty (regularized nonlinear acceleration, 2016)

$$\text{minimize}_{\sum_{j=0}^{m_k} \alpha_j = 1} \quad \|\sum_{j=0}^{m_k} \alpha_j g(x^{k-m_k+j})\|_2^2 + \lambda \|\alpha\|_2^2$$

Help in extreme cases, but impede the convergence in general.

Stabilization of AA-I: 1. Powell-type regularization

Goal of regularization: avoid close to singularity ("lower bound" on B_k).

- AA-II: add ridge penalty (regularized nonlinear acceleration, 2016)

$$\text{minimize}_{\sum_{j=0}^{m_k}\alpha_j=1} \quad \|\sum_{j=0}^{m_k}\alpha_j g(x^{k-m_k+j})\|_2^2+\lambda\|\alpha\|_2^2$$

  Help in extreme cases, but impede the convergence in general.

- AA-I: Powell-type trick (turns out helpful also **in practice**!)
  - Replace y_{k-m_k+i} with $\tilde{y}_{k-m_k+i}=\theta_k^i y_{k-m_k+i}+(1-\theta_k^i)B_k^i s_{k-m_k+i}$, where $\theta_k^i=\phi_{\bar{\theta}}(\eta_k^i)$, with $\eta_k^i=\frac{\hat{s}_{k-m_k+i}{}^T(B_k^i)^{-1}y_{k-m_k+i}}{\|\hat{s}_{k-m_k+i}\|_2^2}$,

$$\phi_{\bar{\theta}}(\eta)=\begin{cases}1 & \text{if } |\eta|\geq\bar{\theta}\\ \frac{1-\textbf{sign}(\eta)\bar{\theta}}{1-\eta} & \text{if } |\eta|<\bar{\theta}.\end{cases}$$

Stabilization of AA-I: 1. Powell-type regularization

Goal of regularization: avoid close to singularity (“lower bound” on B_k).

- AA-II: add ridge penalty (regularized nonlinear acceleration, 2016)

$$\text{minimize}_{\sum_{j=0}^{m_k}\alpha_j=1} \quad \|\sum_{j=0}^{m_k}\alpha_j g(x^{k-m_k+j})\|_2^2+\lambda\|\alpha\|_2^2$$

  Help in extreme cases, but impede the convergence in general.

- AA-I: Powell-type trick (turns out helpful also **in practice**!)
  - Replace y_{k-m_k+i} with $\tilde{y}_{k-m_k+i}=\theta_k^i y_{k-m_k+i}+(1-\theta_k^i)B_k^i s_{k-m_k+i}$, where $\theta_k^i=\phi_{\bar{\theta}}(\eta_k^i)$, with $\eta_k^i=\frac{\hat{s}_{k-m_k+i}{}^T(B_k^i)^{-1}y_{k-m_k+i}}{\|\hat{s}_{k-m_k+i}\|_2^2}$,

$$\phi_{\bar{\theta}}(\eta)=\begin{cases}1 & \text{if } |\eta|\geq\bar{\theta}\\ \frac{1-\textbf{sign}(\eta)\bar{\theta}}{1-\eta} & \text{if } |\eta|<\bar{\theta}.\end{cases}$$

  - $|\det(B_k)|\geq\bar{\theta}^{m_k}>0$, and in particular, B_k is **invertible**!

Stabilization of AA-I: 2. Re-start checking

Goal of re-start: avoid blow-up ("upper bound" on B_k).

Goal of re-start: avoid blow-up ("upper bound" on B_k).

- $\hat{s}_{k-m_k+i}^T s_{k-m_k+i} = \|\hat{s}_{k-m_k+i}\|_2^2$ appears in the denominators: but $\hat{s}_{k-m_k+i}$ becomes 0 when $m_k > n$ due to orthogonalization.

Stabilization of AA-I: 2. Re-start checking

Goal of re-start: avoid blow-up ("upper bound" on B_k).

- $\hat{s}_{k-m_k+i}^T s_{k-m_k+i} = \|\hat{s}_{k-m_k+i}\|_2^2$ appears in the denominators: but $\hat{s}_{k-m_k+i}$ becomes 0 when $m_k > n$ due to orthogonalization.
- Solution: update $m_k = m_{k-1} + 1$. If $m_k = m + 1$ or $\|\hat{s}_{k-1}\|_2 < \tau \|s_{k-1}\|_2$, then reset $m_k = 1$.

Stabilization of AA-I: 2. Re-start checking

Goal of re-start: avoid blow-up (“upper bound” on B_k).

- $\hat{s}_{k-m_k+i}^T s_{k-m_k+i} = \|\hat{s}_{k-m_k+i}\|_2^2$ appears in the denominators: but $\hat{s}_{k-m_k+i}$ becomes 0 when $m_k > n$ due to orthogonalization.
- Solution: update $m_k = m_{k-1} + 1$. If $m_k = m + 1$ or $\|\hat{s}_{k-1}\|_2 < \tau \|s_{k-1}\|_2$, then reset $m_k = 1$.
- Then $\|B_k\|_2 \leq 3(1 + \bar{\theta} + \tau)^m / \tau^m - 2$!
- (Re)define $H_k := B_k^{-1}$: $\|H_k\|_2 \leq \left(3\left(\frac{1+\bar{\theta}+\tau}{\tau}\right)^m - 2\right)^{n-1} / \bar{\theta}^m$.

Stabilization of AA-I: 3. Safe-guard checking

Goal of safe-guard: avoid “wild” and “bad” extrapolation.

Stabilization of AA-I: 3. Safe-guard checking

Goal of safe-guard: avoid “wild” and “bad” extrapolation.

- Main idea: interleave AA-I steps with the vanilla KM iteration steps to safe-guard the decrease in residual norms g.

Stabilization of AA-I: 3. Safe-guard checking

Goal of safe-guard: avoid "wild" and "bad" extrapolation.

- Main idea: interleave AA-I steps with the vanilla KM iteration steps to safe-guard the decrease in residual norms g.
- Check if the current residual norm is sufficiently small, and replace it with $f_\alpha(x) = (1-\alpha)x + \alpha f(x)$ whenever not.

Stabilization of AA-I: 3. Safe-guard checking

Goal of safe-guard: avoid "wild" and "bad" extrapolation.

- Main idea: interleave AA-I steps with the vanilla KM iteration steps to safe-guard the decrease in residual norms g.
- Check if the current residual norm is sufficiently small, and replace it with $f_\alpha(x) = (1-\alpha)x + \alpha f(x)$ whenever not.
- Can be seen as a cheap alternative to the expensive line-search.

Stabilized AA-I

Combine Powell-type regularization, re-start checking and safe-guard checking (with some simplifications using Woodbury formula, etc.)

Algorithm 3 Stablized Type-I Anderson Acceleration (AA-I-S)

Input: initial point x_0, fixed-point mapping $f : \mathbb{R}^n \to \mathbb{R}^n$, regularization constants $\bar{\theta},\ \tau,\ \alpha \in (0,1)$, safe-guarding constants $D,\ \epsilon > 0$, max-memory $m > 0$.
Initialize $H_0 = I,\ m_0 = n_{AA} = 0,\ \bar{U} = \|g_0\|_2$, and compute $x^1 = \tilde{x}^1 = f_\alpha(x^0)$.
for $k = 1,\ 2,\ \ldots$ **do**
    $m_k = m_{k-1} + 1$.
    Compute $s_{k-1} = \tilde{x}^k - x^{k-1}$, $y_{k-1} = g(\tilde{x}^k) - g(x^{k-1})$.
    Compute $\hat{s}_{k-1} = s_{k-1} - \sum_{j=k-m_k}^{k-2} \frac{\hat{s}_j^T s_{k-1}}{\hat{s}_j^T \hat{s}_j} \hat{s}_j$.
    If $m_k = m + 1$ or $\|\hat{s}_{k-1}\|_2 < \tau \|s_{k-1}\|_2$ {Re-start checking}
        reset $m_k = 1$, $\hat{s}_{k-1} = s_{k-1}$, and $H_{k-1} = I$.
    Update H_k with {Powell-type regularization}, compute $\tilde{x}^{k+1} = x^k - H_k g_k$.
    If $\|g_k\| \le D\bar{U}(n_{AA} + 1)^{-(1+\epsilon)}$ {Safe-guard checking}
        $x^{k+1} = \tilde{x}^{k+1}$, $n_{AA} = n_{AA} + 1$.
    else $x^{k+1} = f_\alpha(x^k)$.

Global convergence

Theorem

Suppose that f is non-expansive in l_2-norm or contractive in an arbitrary norm, and assume that $\{x^k\}_{k=0}^{\infty}$ is generated by Algorithm 3. Then we have $\lim_{k\to\infty} x^k = x^\star$, where $x^\star = f(x^\star)$.

Key: bounds on H_k and B_k ensure that the deviation is not too much from the safe-guarding paths.

Implementation details

- **Hyper-parameters choice**: $\bar{\theta} = 0.01$, $\tau = 0.001$, $D = 10^6$, $\epsilon = 10^{-6}$, memory $m = 5$, averaging weight $\alpha = 0.1$.
- **Matrix-free updates**: instead of computing and storing H_k, we store $H_{k-j}\tilde{y}_{k-j}$ and $\frac{H_{k-j}^T \hat{s}_{k-j}}{\hat{s}_{k-j}^T (H_{k-j}\tilde{y}_{k-j})}$ for $j = 1, \ldots, m_k$, compute

$$d_k = g_k + \sum_{j=1}^{m_k} (s_{k-j} - (H_{k-j}\tilde{y}_{k-j})) \left(\frac{H_{k-j}^T \hat{s}_{k-j}}{\hat{s}_{k-j}^T (H_{k-j}\tilde{y}_{k-j})} \right)^T g_k,$$

  and then update $\tilde{x}^{k+1} = x^k - d_k$.
- **Problem scaling** is helpful when matrices are involed.

More examples: Problem + ALG $\Leftrightarrow$ black-box FP

General idea: rewrite an algorithm into $x^{k+1} = f(x^k)$ by **concatenation and neglecting (intermediate variables)**.
Apart from **PGD** ($\min_{x \in C} \; F(x)$) and **value iteration** (MDP):

More examples: Problem + ALG $\Leftrightarrow$ black-box FP

General idea: rewrite an algorithm into $x^{k+1} = f(x^k)$ by **concatenation and neglecting (intermediate variables)**.

Apart from **PGD** ($\min_{x \in C} \ F(x)$) and **value iteration** (MDP):

- Problem 1: find $x \in C \cap D$.

More examples: Problem + ALG $\Leftrightarrow$ black-box FP

General idea: rewrite an algorithm into $x^{k+1} = f(x^k)$ by **concatenation and neglecting (intermediate variables)**.

Apart from **PGD** ($\min_{x \in C} \; F(x)$) and **value iteration** (MDP):

- Problem 1: find $x \in C \cap D$.
- Algorithm – alternating projection: $x^{k+1} = f(x^k) = \Pi_C(\Pi_D(x^k))$.

More examples: Problem + ALG $\Leftrightarrow$ black-box FP

General idea: rewrite an algorithm into $x^{k+1} = f(x^k)$ by **concatenation and neglecting (intermediate variables)**.

Apart from **PGD** ($\min_{x \in C} \; F(x)$) and **value iteration** (MDP):

- Problem 1: find $x \in C \cap D$.
- Algorithm – alternating projection: $x^{k+1} = f(x^k) = \Pi_C(\Pi_D(x^k))$.
- FP: $x = \Pi_C(\Pi_D(x^k))$.

More examples: Problem + ALG $\Leftrightarrow$ black-box FP

General idea: rewrite an algorithm into $x^{k+1} = f(x^k)$ by **concatenation and neglecting (intermediate variables)**.

Apart from **PGD** ($\min_{x \in C} \; F(x)$) and **value iteration** (MDP):

- Problem 2: $\text{minimize}_x \; F(x) + \mu \|x\|_1$.
- Algorithm – ISTA: $x^{k+1} = S_{\alpha\mu}(x^k - \alpha \nabla F(x^k))$, with $S_\kappa(x)_i = \textbf{sign}(x_i)(|x_i| - \kappa)_+$ for $i = 1, \ldots, n$.
- FP: $x = S_{\alpha\mu}(x - \alpha \nabla F(x))$.

General idea: rewrite an algorithm into $x^{k+1} = f(x^k)$ by **concatenation and neglecting (intermediate variables)**.

Apart from **PGD** ($\min_{x \in C} \ F(x)$) and **value iteration** (MDP):

- Problem 3: $\text{minimize}_x \ \sum_{i=1}^m F_i(x)$.
- Algorithm – consensus DRS:

$$
\begin{aligned}
x_i^{k+1} &= \text{argmin}_{x_i} \ F_i(x_i) + (1/2\alpha)\|x_i - z_i^k\|_2^2, \\
z_i^{k+1} &= z_i^k + 2\bar{x}^{k+1} - x_i^{k+1} - \bar{z}^k, \quad i = 1, \ldots, m.
\end{aligned}
$$

- FP: f defined as the mapping from z^k to z^{k+1}.
- Wrong approach: apply AA to both x and z.

More examples: Problem + ALG $\Leftrightarrow$ black-box FP

General idea: rewrite an algorithm into $x^{k+1} = f(x^k)$ by **concatenation and neglecting (intermediate variables)**.

Apart from **PGD** ($\min_{x \in C} \ F(x)$) and **value iteration** (MDP):

- Problem 4: minimize$_x$ $c^T x$, subject to $Ax + s = b, \ s \in \mathcal{K}$.
- Algorithm – SCS ($\mathcal{C} = \mathbb{R}^n \times \mathcal{K}^* \times \mathbb{R}_+$):

$$
\begin{aligned}
\tilde{u}^{k+1} &= (I + Q)^{-1}(u^k + v^k) \\
u^{k+1} &= \Pi_{\mathcal{C}}(\tilde{u}^{k+1} - v^k) \\
v^{k+1} &= v^k - \tilde{u}^{k+1} + u^{k+1}.
\end{aligned}
$$

- FP (don't apply AA to u and v separately):

$$
f(u, v) = \begin{bmatrix} \Pi_{\mathcal{C}}((I + Q)^{-1}(u + v) - v) \\ v - (I + Q)^{-1}(u + v) + u \end{bmatrix}.
$$

General idea: rewrite an algorithm into $x^{k+1} = f(x^k)$ by **concatenation and neglecting (intermediate variables)**.

Apart from **PGD** ($\min_{x \in C} \; F(x)$) and **value iteration** (MDP):

- Problem 5: minimize$_x$ $\frac{1}{2}x^T Ax + b^T x + c$.
- Algorithm: momentum GD: $x^{k+1} = x^k - \alpha(Ax^k + b) + \beta(x^k - x^{k-1})$.
- FP (concatenate two successive iterates):

$$f(x', x) = \begin{bmatrix} x' - \alpha(Ax' + b) + \beta(x' - x) \\ x' \end{bmatrix}.$$

- Remember to concatenate, don't simply neglect x^{k-1} as in RNA.

Numerical examples

Gradient Descent: stabilization from **divergence** to **convergence**

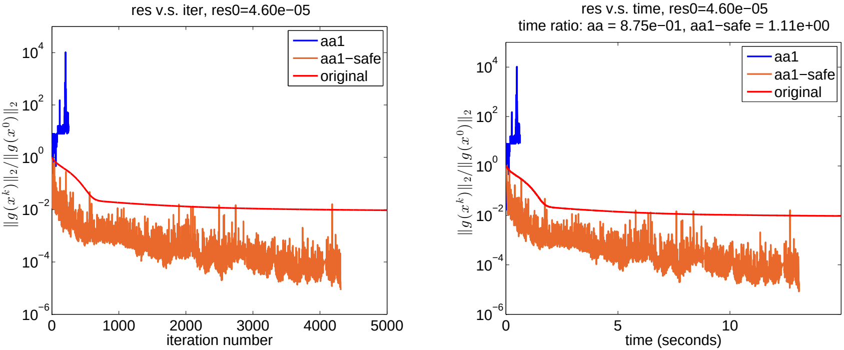


Figure: Gradient descent: regularized logistic regression. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

Numerical examples

SCS (ADMM): SOCP – nonsmoothness coming from projections

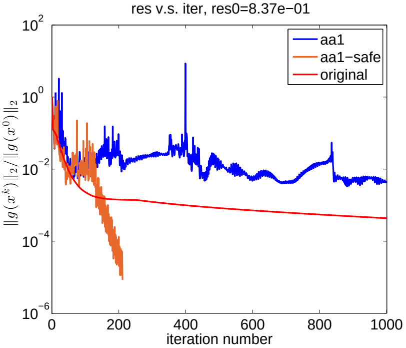


Figure: SCS: second-order cone program. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

Numerical examples

ISTA: elastic net regression – nonsmoothness coming from shrinkage

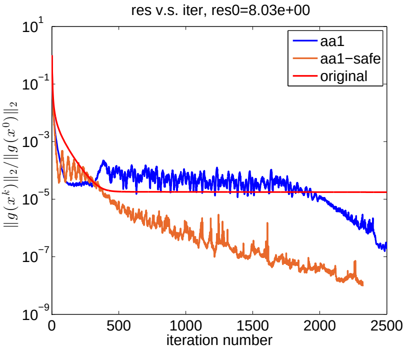


Figure: Iterative Shrinkage-Thresholding Algorithm: elastic-net linear regression. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

Numerical examples

MDP (value iteration) (discount factor $\gamma = 0.99$):

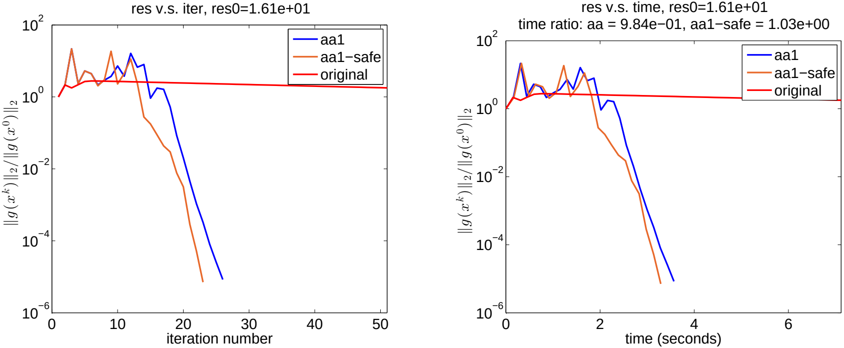


Figure: Value iteration: MDP. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

Effect of **different memories** *m*:

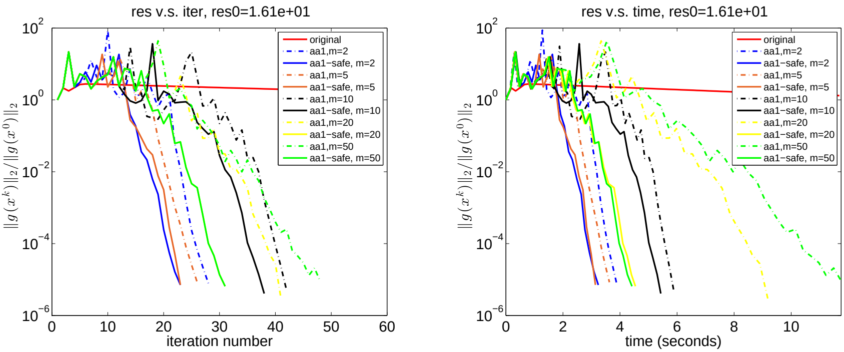


Figure: Value iteration: memory effect. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

Summary

- **Starting point**: Early empirical success in applying AA-I to SCS, but unstable performance

Summary

- **Starting point**: Early empirical success in applying AA-I to SCS, but unstable performance
- **Destination:**
  - the first globally convergent Anderson acceleration variant under very relaxed conditions.

Summary

- **Starting point**: Early empirical success in applying AA-I to SCS, but unstable performance
- **Destination:**
  - the first globally convergent Anderson acceleration variant under very relaxed conditions.
  - online pre-conditioning/stabilization tricks useful both in theory and practice (Powell, re-start and safe-guard).

Summary

- **Starting point**: Early empirical success in applying AA-I to SCS, but unstable performance
- **Destination:**
  - the first globally convergent Anderson acceleration variant under very relaxed conditions.
  - online pre-conditioning/stabilization tricks useful both in theory and practice (Powell, re-start and safe-guard).
  - flexible applications to many different problems/algorithms with general black-box fixed-point reformulation, with stable performance.

Summary

- **Starting point**: Early empirical success in applying AA-I to SCS, but unstable performance
- **Destination:**
  - the first globally convergent Anderson acceleration variant under very relaxed conditions.
  - online pre-conditioning/stabilization tricks useful both in theory and practice (Powell, re-start and safe-guard).
  - flexible applications to many different problems/algorithms with general black-box fixed-point reformulation, with stable performance.
  - Now being implemented and tested in SCS 2.0.

Beyond non-expansiveness (convexity)

- Our stabilization technique can actually be extended to generic **non-convex** optimization settings.

Beyond non-expansiveness (convexity)

- Our stabilization technique can actually be extended to generic **non-convex** optimization settings.
  - **Safe-guard** becomes central here (unlike non-expansive cases), and need to be exclusive designed for each algorithm.

Beyond non-expansiveness (convexity)

- Our stabilization technique can actually be extended to generic **non-convex** optimization settings.
  - **Safe-guard** becomes central here (unlike non-expansive cases), and need to be exclusive designed for each algorithm.
  - Example: We proposed **Anderson accelerated iPALM** [GHXZ2018] with an exclusive safe-guard for iPALM for computing the MLEs multivariate Hawkes processes.

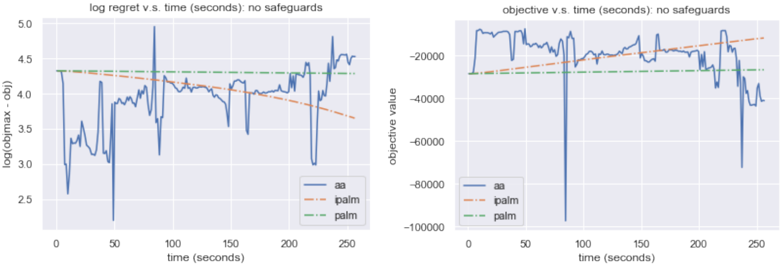


Figure: MLE of MHPs: exponential hawkes. No safe-guards. Left: log-regret v.s. time (seconds). Right: objective v.s. time (seconds).

Safe-guards in non-convex optimization

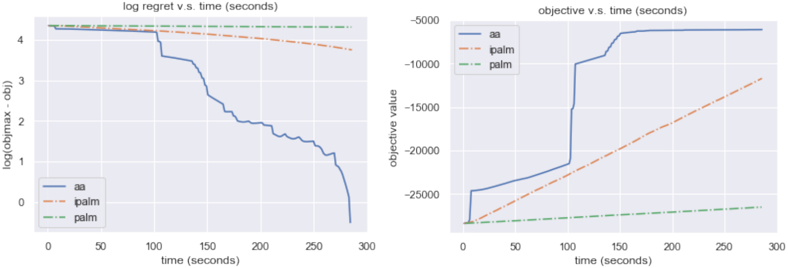


Figure: MLE of MHPs: exponential hawkes. With safe-guards. Left: log-regret v.s. time (seconds). Right: objective v.s. time (seconds).

Future work

- Can we extract some general design rules of safe-guards formally?

Future work

- Can we extract some general design rules of safe-guards formally?
  - Find a **balance** between practical efficiency and theoretical guarantee.

Future work

- Can we extract some general design rules of safe-guards formally?
  - Find a **balance** between practical efficiency and theoretical guarantee.
  - Failure example: apply AA-II to Nesterov, but require monotonic decrease in the objective values, which breaks the non-monotonic acceleration of Nesterov.

Future work

- Can we extract some general design rules of safe-guards formally?
  - Find a **balance** between practical efficiency and theoretical guarantee.
  - Failure example: apply AA-II to Nesterov, but require monotonic decrease in the objective values, which breaks the non-monotonic acceleration of Nesterov.
- More examples for applying AA-I:
  - Nesterov's accelerated gradient descent, Frank-Wolfe, stochastic gradient descent and its variants (*e.g.*, ADAM), ... (a ongoing tutorial paper).

Future work

- Can we extract some general design rules of safe-guards formally?
  - Find a **balance** between practical efficiency and theoretical guarantee.
  - Failure example: apply AA-II to Nesterov, but require monotonic decrease in the objective values, which breaks the non-monotonic acceleration of Nesterov.
- More examples for applying AA-I:
  - Nesterov's accelerated gradient descent, Frank-Wolfe, stochastic gradient descent and its variants (*e.g.*, ADAM), ... (a ongoing tutorial paper).
- Adaptive choices/line-search of the hyper-parameters in our stabilized AA-I.

References

Zhang, J., O'Donoghue, B, and Boyd, S. P. (2018).
Globally Convergent Type-I Anderson Acceleration for Non-Smooth Fixed-Point Iterations.
arXiv preprint arXiv:1808.03971.

X. Guo, A. Hu, R. Xu, and Zhang, J. (2018).
Consistency and Computation of Regularized MLEs for Multivariate Hawkes Processes.
arXiv preprint arXiv:1810.02955.

Thanks for listening!

Any questions?