Fast convex optimization for eigenproblems and beyond

Vien V. Mai and Mikael Johansson
KTH - Royal Institute of Technology

Interplay of optimization and numerical linear algebra

Many optimization algorithms involve solving linear algebra problems

- preconditioning
- matrix factorization
- eigenvalue decomposition
- largest, smallest eigenvalues

Optimization helps numerical linear algebra

- approximating matrix-vector multiplications, matrix inverse, etc

Recent advances in solving large-scale learning problems

Stochastic optimization

Mostly dealing with empirical-risk minimization

$$\underset{\boldsymbol{x}\in\mathbb{R}^d}{\text{minimize}}\ \frac{1}{n}\sum_{i=1}^{n} f_i\left(\boldsymbol{x}\right)$$

SGD, SAG, SAGA, SVRG, and their variants

Randomized numerical linear algebra:

Efficient computations of huge matrices

randomized matrix multiplication, low rank approximation, sketching, ...

[Bottou et al., 2018], [Drineas and Mahoney, 2018]

Two highly structured problems

Generalized eigenproblem

Finding the solution $(\lambda, \boldsymbol{w})$ of the equation

$$\mathbf{A}\boldsymbol{w} = \lambda \mathbf{B}\boldsymbol{w}$$

Key problem in PCA, CCA, Fisher LDA, SVD, etc

Elastic net problem

$$\underset{\boldsymbol{x}\in\mathbb{R}^d}{\text{minimize}} \left\{ \frac{1}{2n} \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \frac{\gamma_2}{2} \|\boldsymbol{x}\|_2^2 + \gamma_1 \|\boldsymbol{x}\|_1 \right\},$$

Ubiquitous in statistics, signal/image processing

Focus of this talk

The interaction of optimization and numerical lin.alg. in these two problems

Contents

Generalized eigenvalue problem (GEV)

Top-k generalized eigenvalue problem:

$$\begin{aligned} \underset{\boldsymbol{w}_i}{\text{maximize}} \quad & |\boldsymbol{w}_i^\top \mathbf{A} \boldsymbol{w}_i| \\ \text{subject to} \quad & \boldsymbol{w}_i^\top \mathbf{B} \boldsymbol{w}_i = 1 \\ & \boldsymbol{w}_i^\top \mathbf{B} \boldsymbol{w}_j = 0 \quad \forall j \in \{1, 2, \ldots, i-1\}. \end{aligned}$$

Can be computed as top-k eigenvectors of

$$\mathbf{M} = \mathbf{B}^{-1/2} \mathbf{A} \mathbf{B}^{-1/2}$$

and then multiply by $\mathbf{B}^{-1/2}$, but forming $\mathbf{B}^{-1/2}$ is very **costly**

Q: can we develop a scalable, linearly convergent algorithm?

Classical methods

Power method:

- oldest and simplest one
- iteration complexity: $\mathcal{O}\left(\frac{1}{\Delta}\log\frac{1}{\epsilon}\right)$ (Δ: **relative eigenvalue gap**)
- converge very slowly if Δ is small

Lanczos method:

- a significantly more sophisticated method
- iteration complexity: $\mathcal{O}\left(\frac{1}{\sqrt{\Delta}}\log\frac{1}{\epsilon}\right)$ (optimal w.r.t. Δ)

Both require multiple matrix-vector products of the form $\mathbf{B}^{-1}\boldsymbol{w}$

➤ running times are **superlinear** in the input size $(\mathrm{nnz}(\mathbf{A}), \mathrm{nnz}(\mathbf{B}))$

More recent work

Shamir (2015):

- combines variance reduction with Oja's algorithm
- first linear convergence stochastic PCA

Gaber et al. (2016):

- based on shift-and-inverse technique
- improves the iteration complexity in Shamir (2015)

De Sa et al. (2017):

- Oja's method + momentum + variance reduction + minibatching
- linear convergence for large enough minibatch sizes

Ge et al. (2016):

- performs inexact power method
- same iteration complexity as the power method

and more (e.g., online PCA, doubly acceleration,...)

A closer look at the Power method

Given a feasible $\boldsymbol{x}_0$, the power method works as follows:

$$\boldsymbol{x}_k = \mathbf{B}^{-1}\mathbf{A}\boldsymbol{x}_{k-1} = \left(\mathbf{B}^{-1}\mathbf{A}\right)^k \boldsymbol{x}_0 = p_k\left(\mathbf{B}^{-1}\mathbf{A}\right)\boldsymbol{x}_0,$$

where $p_k(x) = x^k$ is a degree-k polynomial and

$$p_k\left(\mathbf{B}^{-1}\mathbf{A}\right) = \boldsymbol{U}\begin{bmatrix} \lambda_1^k & 0 & \dots & 0 \\ 0 & \lambda_2^k & \dots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \lambda_d^k \end{bmatrix}\boldsymbol{U}^\top.$$

Suppose that $|\lambda_2| = (1-\Delta)|\lambda_1|$, then

- $\left(\mathbf{B}^{-1}\mathbf{A}\right)^k$ is dominated by $\lambda_1^k \boldsymbol{u}_1\boldsymbol{u}_1^\top$
- needs $k = \mathcal{O}\left(\frac{1}{\Delta}\log\frac{1}{\epsilon}\right)$ iterations until $|\lambda_2|^k \le \epsilon|\lambda_1|^k$

Polynomial approximation of x^k

Proposition. For any positive integers k and d, there is a degree-d polynomial $p_{k,d}(x)$ satisfying

$$\sup_{x\in[-1,1]} \left| p_{k,d}(x) - x^k \right| \le 2e^{-d^2/2k}.$$

Can approximate x^k by any accuracy δ using a polynomial of degree

$$d = \lceil \sqrt{2\ln(2/\delta)\, k} \rceil.$$

Essentially optimal: approx. of x^k over $[-1,1]$ requires degree $\Omega(\sqrt{k})$.

[Sachdeva and Vishnoi, 2013]

Polynomial approximation in numerical methods

Key (implicit) idea behind the quadratic savings in #iters. of many methods

- $\mathcal{O}\left(\sqrt{\kappa}\log\frac{1}{\epsilon}\right)$ (CG) v.s. $\mathcal{O}\left(\kappa\log\frac{1}{\epsilon}\right)$ (Richardson iteration)
- $\mathcal{O}\left(\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}\right)$ (AGD) v.s. $\mathcal{O}\left(\frac{L}{\mu}\log\frac{1}{\epsilon}\right)$ (GD)
- $\mathcal{O}\left(\frac{1}{\sqrt{\Delta}}\log\frac{1}{\epsilon}\right)$ (Lanczos) v.s. $\mathcal{O}\left(\frac{1}{\Delta}\log\frac{1}{\epsilon}\right)$ (power method)

They all essentially use stronger polynomials than their counterparts.

Definition. The degree k **Chebyshev polynomial** of the first kind, $T_k(z)$, is defined recursively as follows:

$$\begin{aligned} &T_0(z) = 1,\ T_1(z) = z, \\ &T_k(z) = 2zT_{k-1}(z) - T_{k-2}(z) \quad \text{for} \quad k \geq 2. \end{aligned}$$

Scalable GEV: key ideas

Reduce #iters. using **momentum-acceleration** inspired by poly. approx.

Reduce computational cost per iter using **convex optimization**

$$\boldsymbol{w}_{k+1} \approx \mathbf{B}^{-1}\mathbf{A}\boldsymbol{w}_k$$

by solving

$$\underset{\boldsymbol{w}}{\text{minimize}}\ \frac{1}{2}\boldsymbol{w}^\top \mathbf{B}\boldsymbol{w} - \boldsymbol{w}^\top \mathbf{A}\boldsymbol{w}_k$$

to some target precision (under our control).

Improve performance by well-designed **initialization**

Novel algorithm

Algorithm 1 Power Method

```
Require: Initial points $\boldsymbol{w}_0$.
1: $\boldsymbol{w}_0 \leftarrow \boldsymbol{w}_0 / \|\boldsymbol{w}_0\|_{\mathbf{B}}$
2: for $k = 0, 1, \dots, s-1$ do
3:     $\tilde{\boldsymbol{w}}_{k+1} \leftarrow \mathbf{B}^{-1}\mathbf{A}\boldsymbol{w}_k$
4:     $\boldsymbol{w}_{k+1} \leftarrow \frac{\tilde{\boldsymbol{w}}_{k+1}}{\|\tilde{\boldsymbol{w}}_{k+1}\|_{\mathbf{B}}}$
5: end for
Ensure: $\boldsymbol{w}_s$
```

Algorithm 2 Noisy Accelerated Power Method

```
Require: Initial points $\boldsymbol{w}_{-1} = 0$, $\boldsymbol{w}_0$.
1: $\boldsymbol{w}_0 \leftarrow \boldsymbol{w}_0 / \|\boldsymbol{w}_0\|_{\mathbf{B}}$
2: for $k = 0, 1, \dots, s-1$ do
3:     $\alpha_k \leftarrow \boldsymbol{w}_k^\top \mathbf{A}\boldsymbol{w}_k$
4:     $\tilde{\boldsymbol{w}}_{k+1} \approx \operatorname{argmin}\left\{\frac{1}{2}\boldsymbol{w}^\top\mathbf{B}\boldsymbol{w} - \boldsymbol{w}^\top\mathbf{A}\boldsymbol{w}_k\right\}$
       (initialize the solver with $\alpha_k \boldsymbol{w}_k$)
5:     $\tilde{\boldsymbol{w}}_{k+1} \leftarrow \tilde{\boldsymbol{w}}_{k+1} - \beta\boldsymbol{w}_{k-1}$
6:     $\boldsymbol{w}_k \leftarrow \frac{\boldsymbol{w}_k}{\|\tilde{\boldsymbol{w}}_{k+1}\|_{\mathbf{B}}}$, $\boldsymbol{w}_{k+1} \leftarrow \frac{\tilde{\boldsymbol{w}}_{k+1}}{\|\tilde{\boldsymbol{w}}_{k+1}\|_{\mathbf{B}}}$
7: end for
Ensure: $\boldsymbol{w}_s$
```

Theoretical performance of `NAPI`

Let λ_i be the eigenvalues of $\mathbf{B}^{-1}\mathbf{A}$ satisfying $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_d|$.

- If $|\lambda_2| \leq 2\sqrt{\beta} < |\lambda_1|$, then

$$\sin^2\theta\left(\boldsymbol{w}_s, \boldsymbol{v}_1\right) \leq \left(\frac{1}{2} + \frac{\sqrt{\beta}}{|\lambda_1| + \sqrt{\lambda_1^2 - 4\beta}}\right)^{2s} \tan^2\theta_0.$$

- If $\beta = |\lambda_2|^2/4$, the number of iterations needed is of order

$$\mathcal{O}\left(\frac{1}{\sqrt{\Delta}} \log \frac{\tan\theta_0}{\epsilon}\right).$$

- If we use Nesterov's accelerated method, the total running time is:

$$\mathcal{O}\left(\frac{\mathrm{nnz}\left(\mathbf{B}\right)\sqrt{\kappa}}{\sqrt{\Delta}} \log \frac{\tan\theta_0}{\epsilon} + \frac{\mathrm{nnz}\left(\mathbf{A}\right)}{\sqrt{\Delta}} \log \frac{\tan\theta_0}{\epsilon}\right).$$

Discussion

+ linearly convergent algorithm matching the asymptotic iteration complexity of the Lanczos method

+ much lower computational cost: **linear running time** in the input-size

➤ allows to operate on large-scale data sets

– requires the prior knowledge of λ_2 to achieve the optimal rate

Rayleigh quotient may give a reasonable estimate

Application to canonical correlation analysis (CCA)

Given two views of a data set: $(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)$, $\boldsymbol{x}_i \in \mathbb{R}^{d_1}$, $\boldsymbol{y}_i \in \mathbb{R}^{d_2}$

Let $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]^\top$ and $\mathbf{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n]^\top$ and define:

$$\boldsymbol{\Sigma}_{11} = \frac{1}{n}\mathbf{X}^\top\mathbf{X} + \gamma_1\mathbf{I}, \;\; \boldsymbol{\Sigma}_{22} = \frac{1}{n}\mathbf{Y}^\top\mathbf{Y} + \gamma_2\mathbf{I}, \;\; \boldsymbol{\Sigma}_{12} = \frac{1}{n}\mathbf{X}^\top\mathbf{Y}.$$

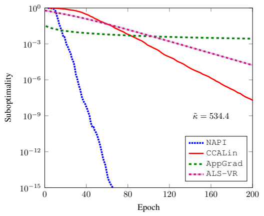
CCA maximizes the empirical cross-correlation between $\mathbf{X}$ and $\mathbf{Y}$ by solving:

$$\begin{aligned} \underset{\mathbf{x},\mathbf{y}}{\text{maximize}} \quad & \boldsymbol{x}^\top \boldsymbol{\Sigma}_{12} \boldsymbol{y} \\ \text{subject to} \quad & \boldsymbol{x}^\top \boldsymbol{\Sigma}_{11} \boldsymbol{x} = \boldsymbol{y}^\top \boldsymbol{\Sigma}_{22} \boldsymbol{y} = 1. \end{aligned}$$

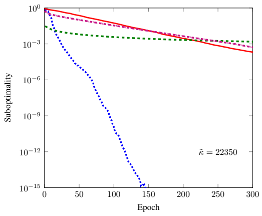
Reducing to the generalized eigenproblem:

$$\begin{bmatrix} \mathbf{0} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{12}^\top & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}$$

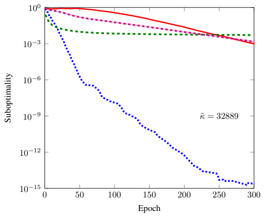
Experimental results



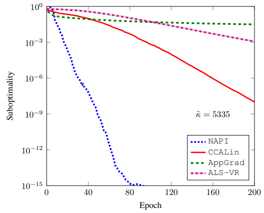
(a) Mediamill, $\gamma_1 = \gamma_2 = 10^{-3}$



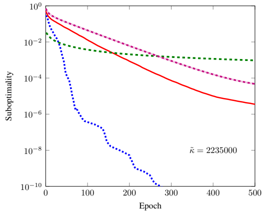
(b) MNIST, $\gamma_1 = \gamma_2 = 10^{-3}$



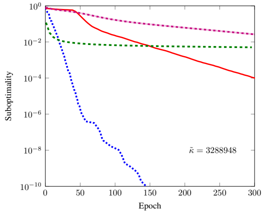
(c) XRMB, $\gamma_1 = \gamma_2 = 10^{-3}$



(d) Mediamill, $\gamma_1 = \gamma_2 = 10^{-4}$



(e) MNIST, $\gamma_1 = \gamma_2 = 10^{-4}$



(f) XRMB, $\gamma_1 = \gamma_2 = 10^{-5}$

[Wang et al., 2017], [Ma et al., 2015]

Contents

Elastic net problem

Consider the minimization problem

$$\underset{\boldsymbol{x}\in\mathbb{R}^d}{\text{minimize}} \left\{ \frac{1}{2n} \left\| \mathbf{A}\boldsymbol{x} - \boldsymbol{b} \right\|_2^2 + \frac{\gamma_2}{2} \left\| \boldsymbol{x} \right\|_2^2 + \gamma_1 \left\| \boldsymbol{x} \right\|_1 \right\}$$

- $\gamma_1 = 0$: Ridge regression
- $\gamma_2 = 0$: Lasso

Can be written as a finite-sum minimization problem

Hessian of the smooth part

$$\nabla^2 f(\boldsymbol{x}) = \frac{1}{n}\mathbf{A}^\top\mathbf{A} + \gamma_2\mathbf{I} = \boldsymbol{C} + \gamma_2\mathbf{I}$$

State-of-the-art implementations are mostly **2nd-order** methods, e.g., `gmlnet`

Related work

Deterministic first-order methods:

- Proximal gradient descent: $O\left(dn\kappa \log \frac{1}{\epsilon}\right)$
- `FISTA`: $O\left(dn\sqrt{\kappa} \log \frac{1}{\epsilon}\right)$

$\kappa = \frac{\lambda_1(\boldsymbol{C}+\gamma_2\mathbf{I})}{\lambda_d(\boldsymbol{C}+\gamma_2\mathbf{I})}$

Stochastic first-order methods:

- `ProxSVRG`: $O\left(d\left(n+\tilde{\kappa}\right) \log \frac{1}{\epsilon}\right)$
- Katyusha: $O\Big(d\big(n+\sqrt{n\tilde{\kappa}}\big) \log \frac{1}{\epsilon}\Big)$

$\tilde{\kappa} = \frac{\mathrm{tr}(\boldsymbol{C}+\gamma_2\mathbf{I})}{\lambda_d(\boldsymbol{C}+\gamma_2\mathbf{I})}$

Ridge regression:

- `SCSVRG`: $O\Big(d\big(n+\kappa_{\mathrm{avg}}\big) \log \frac{1}{\epsilon}\Big)$

$\kappa_{\mathrm{avg}} = \frac{\mathrm{tr}\left(\boldsymbol{H}^{-1}(\boldsymbol{C}+\gamma_2\mathbf{I})\right)}{\lambda_d(\boldsymbol{C}+\gamma_2\mathbf{I})}$

[Gonen, Orabona, and Shalev-Shwartz, 2016]

An observation

Practical data sets often have few strong dominant features

Other coordinates are highly correlated with the stronger features

➤ rapid decaying spectrum of the matrix $\boldsymbol{C}$

Large gain in conditioning:

$$\frac{\tilde{\kappa}}{\kappa_{\mathrm{avg}}} = \frac{\sum_{i=1}^{r} \lambda_i + \sum_{i>r} \lambda_i}{r\lambda_r + \sum_{i>r} \lambda_i} \gg 1$$

- $r\lambda_r \ll \sum_{i=1}^{r} \lambda_i$
- $\sum_{i>r} \lambda_i \ll r\lambda_r$

Q. Can we achieve this time complexity for the Elastic net?

Building block 1: Block Lanczos [Musco and Musco, 2015]

BL outputs a rank-r approximation of $\mathbf{A}$ as $\boldsymbol{U}_r \boldsymbol{\Sigma}_r \boldsymbol{V}_r^\top$ in time $O(rnd)$

- $\boldsymbol{U}_r, \boldsymbol{V}_r \in \mathbb{R}^{d \times r}$
- $\boldsymbol{\Sigma}_r \in \mathbb{R}^{r \times r}$

Considering the approximate Hessian of the form

$$\boldsymbol{H} = \boldsymbol{V}_r \left(\boldsymbol{\Sigma}_r^2 + \gamma_2 \mathbf{I} \right) \boldsymbol{V}_r^\top + \left(\sigma_r^2 + \gamma_2 \right) \left(\mathbf{I} - \boldsymbol{V}_r \boldsymbol{V}_r^\top \right) \tag{1}$$

➤ Can compute $\boldsymbol{H}^{-1}\boldsymbol{x}$ in time $O(rd)$

Building block 2: Inexact scaled ProxSVRG (IASVRG)

$$\underset{\boldsymbol{x}\in\mathbb{R}^d}{\text{minimize}}\ \frac{1}{n}\sum_{i=1}^{n} f_i\left(\boldsymbol{x}\right) + h\left(\boldsymbol{x}\right)$$

Algorithm.

Outer loop: computes $\nabla f\left(\tilde{\boldsymbol{x}}_s\right)$

Inner loop: repeats

1. $\boldsymbol{y}_k \leftarrow \frac{1}{1+\tau}\boldsymbol{x}_k + \frac{\tau}{1+\tau}\boldsymbol{z}_k$
2. $\boldsymbol{v}_k \leftarrow \nabla f_{\mathcal{B}_k}\left(\boldsymbol{y}_k\right) - \nabla f_{\mathcal{B}_k}\left(\tilde{\boldsymbol{x}}_s\right) + \nabla f\left(\tilde{\boldsymbol{x}}_s\right)$
3. $\boldsymbol{x}_{k+1} \approx \operatorname{prox}_{\eta h}^{\boldsymbol{H}}\left(\boldsymbol{y}_k - \eta\boldsymbol{H}^{-1}\boldsymbol{v}_k\right)$
4. $\boldsymbol{z}_{k+1} \leftarrow \boldsymbol{z}_k + \tau\left(\boldsymbol{y}_k - \boldsymbol{z}_k\right) + \frac{\tau}{\mu\eta}\left(\boldsymbol{x}_{k+1} - \boldsymbol{y}_k\right)$

Solving subproblems

Approximate proximal Newton updates

$$\boldsymbol{x}_{k+1} \approx \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^d} \frac{1}{2\eta} \left\|\boldsymbol{x} - \boldsymbol{y}_k + \eta \boldsymbol{H}^{-1}\boldsymbol{v}_k\right\|_{\boldsymbol{H}}^2 + \gamma_1 \left\|\boldsymbol{x}\right\|_1$$

Much easier to solve than the original problem

- $\kappa_{\mathrm{sub}} = \lambda_1/\lambda_r \ll \lambda_1/\lambda_d = \kappa$
- gradient computed in time $O(rd)$ instead of $O(nd)$

With **warm-start**, it suffices to reduce the residual by constant factor

FISTA can find a desired solution in at most

$$O\left(r\sqrt{\kappa_{\mathrm{sub}}}\log\kappa_{\mathrm{sub}}\right)$$

gradient evaluations

Performance guarantees of IASVRG

Theorem. Let $\eta = \frac{1}{L_{\text{avg}}}$, $\tau = \sqrt{\frac{\mu}{2L_{\text{avg}}}}$, and $b \geq \sqrt{\frac{L_{\text{avg}}}{\mu}}$. If $T \geq c\sqrt{\frac{L_{\text{avg}}}{\mu}}$, then, it holds that

$$\mathbb{E}\left\{F\left(\tilde{\boldsymbol{x}}_s\right) - F\left(\boldsymbol{x}^\star\right)\right\} \leq \frac{2}{3}\,\mathbb{E}\left\{F\left(\tilde{\boldsymbol{x}}_{s-1}\right) - F\left(\boldsymbol{x}^\star\right)\right\}.$$

Remark. Direct proof based on explicit Lyapunov function:

$$V_k = F\left(\boldsymbol{x}_k\right) - F\left(\boldsymbol{x}^\star\right) + \frac{\mu}{2}\left\|\boldsymbol{z}_k - \boldsymbol{x}^\star\right\|_{\boldsymbol{H}}^2$$

Avoids the use of sophisticated stochastic estimate sequences

Main result

Suppose that the approximate Hessian matrix $\boldsymbol{H}$ is given by (1)

Invoking `IASVRG` with $f(\boldsymbol{x}) = \frac{1}{2n}\left\|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\right\|_2^2 + \frac{\gamma_2}{2}\left\|\boldsymbol{x}\right\|_2^2$, $h(\boldsymbol{x}) = \gamma_1\left\|\boldsymbol{x}\right\|_1$

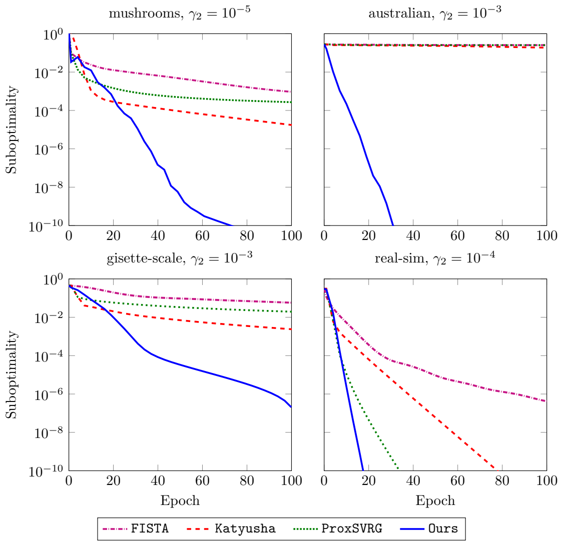
Suppose that the subproblems are solved by `FISTA`

The time complexity of the proposed method is

$$O\Big(d\,(n + \kappa_{\text{avg}}) \log \frac{1}{\epsilon}\Big)$$

Same as **stochastic 1st-order** methods with $\tilde{\kappa}$ replaced by κ_{avg}

Experimental results


mushrooms, $\gamma_2 = 10^{-5}$
australian, $\gamma_2 = 10^{-3}$
gisette-scale, $\gamma_2 = 10^{-3}$
real-sim, $\gamma_2 = 10^{-4}$
Suboptimality
Epoch
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
FISTA
Katyusha
ProxSVRG
Ours