Linear and Sublinear Linear Algebra Algorithms: Preconditioning Stochastic Gradient Algorithms with Randomized Linear Algebra

Michael W. Mahoney

ICSI and Dept of Statistics, UC Berkeley

(For more info, see:
http://www.stat.berkeley.edu/~mmahoney
or Google on "Michael Mahoney")

August 2015

Joint work with Jiyan Yang, Yin-Lam Chow, and Christopher Ré

Outline

RLA and SGD

- **SGD** (Stochastic Gradient Descent) methods[1]
  - Widely used in practice because of their scalability, efficiency, and ease of implementation.
  - Work for problems with general convex objective function.
  - Usually provide an asymptotic bounds on convergence rate.
  - Typically formulated in terms of differentiability assumptions, smoothness assumptions, etc.
- **RLA** (Randomized Linear Algebra) methods[2]
  - Better worst-case theoretical guarantees and better control over solution precision.
  - Less flexible (thus far), e.g., in the presence of constraints.
  - E.g., may use interior point method for solving constrained subproblem, and this may be less efficient than SGD.
  - Typically formulated (either TCS-style or NLA-style) for worst-case inputs.

[1] SGD: iteratively solve the problem by approximating the true gradient by the gradient at a single example.

[2] RLA: construct (with sampling/projections) a random sketch, and use that sketch to solve the subproblem or construct preconditioners for the original problem.

Can we get the "best of both worlds"?

Consider problems where both methods have something nontrivial to say.

Definition

Given a matrix $A \in \mathbb{R}^{n \times d}$, where $n \gg d$, a vector $b \in \mathbb{R}^n$, and a number $p \in [1, \infty]$, the *overdetermined ℓ_p regression problem* is

$$\min_{x \in \mathcal{Z}} f(x) = \|Ax - b\|_p.$$

Important special cases:

- Least Squares: $\mathcal{Z} = \mathbb{R}^d$ and $p = 2$.
  - Solved by eigenvector methods with $\mathcal{O}(nd^2)$ worst-case running time; or by iterative methods for which the running time depending on $\kappa(A)$.
- Least Absolute Deviations: $\mathcal{Z} = \mathbb{R}^d$ and $p = 1$.
  - Unconstrained ℓ_1 regression problem can be formulated as a linear program and solved by an interior-point method.

Outline

Deterministic ℓ_p regression as stochastic optimization

- Let $U \in \mathbb{R}^{n\times d}$ be a basis of the range space of A in the form of $U = AF$, where $F \in \mathbb{R}^{d\times d}$.
- The constrained overdetermined (deterministic) ℓ_p regression problem is equivalent to the (stochastic) optimization problem

$$\begin{aligned}\min_{x\in\mathcal{Z}} \|Ax-b\|_p^p &= \min_{y\in\mathcal{Y}} \|Uy-b\|_p^p \\ &= \min_{y\in\mathcal{Y}} \mathbb{E}_{\xi\sim P}\left[H(y,\xi)\right],\end{aligned}$$

  where $H(y,\xi) = \frac{|U_\xi y - b_\xi|^p}{p_\xi}$ is the randomized integrand and ξ is a random variable over $\{1,\ldots,n\}$ with distribution $P = \{p_i\}_{i=1}^n$.
- The constraint set of y is given by $\mathcal{Y} = \{y \in \mathbb{R}^d | y = F^{-1}x, x \in \mathcal{Z}\}$.

Brief overview of stochastic optimization

The *standard stochastic optimization problem* is of the form

$$\min_{x \in \mathcal{X}} f(x) = \mathbb{E}_{\xi \sim P} \left[F(x, \xi) \right], \tag{1}$$

where ξ is a random data point with underlying distribution P.

Two computational approaches for solving stochastic optimization problems of the form (1) based on Monte Carlo sampling techniques:

- **SA** (Stochastic Approximation):
  - Start with an initial weight x_0, and solve (1) iteratively.
  - In each iteration, a new sample point ξ_t is drawn from distribution P and the current weight is updated by its information (e.g., (sub)gradient of $F(x, \xi_t)$).
- **SAA** (Sampling Average Approximation):
  - Sample n points from distribution P independently, $\xi_1, \ldots, \xi_n$, and solve the Empirical Risk Minimization (ERM) problem,

$$\min_{x \in \mathcal{X}} \hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} F(x, \xi_i).$$

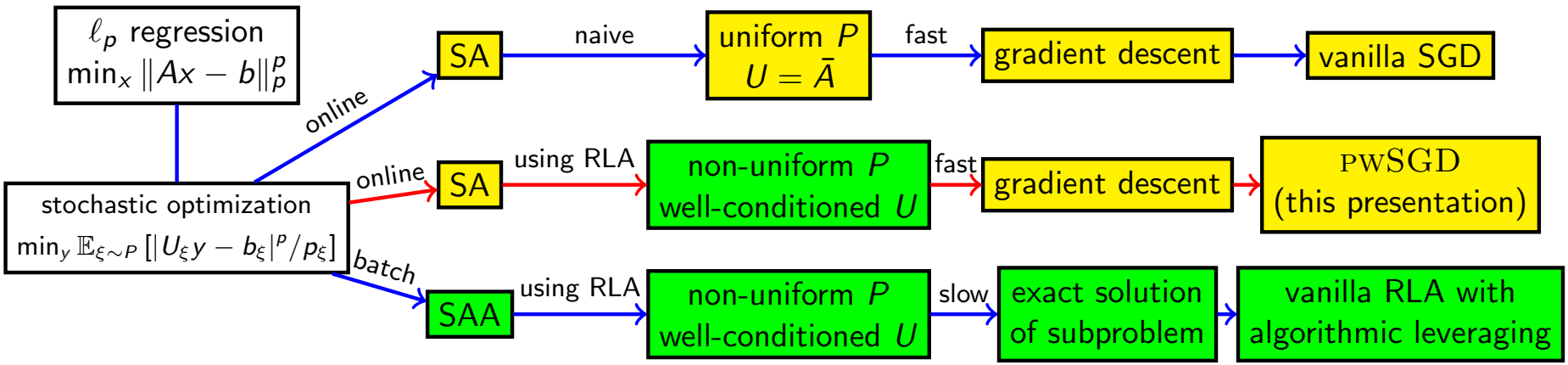
Solving ℓ_p regression via stochastic optimization

To solve this stochastic optimization problem, typically one needs to answer the following three questions.

- *(C1): How to sample: SAA (i.e., draw samples in a batch mode and deal with the subproblem) or SA (i.e., draw a mini-batch of samples in an online fashion and update the weight after extracting useful information)?*
- *(C2): Which probability distribution P (uniform distribution or not) and which basis U (preconditioning or not) to use?*
- *(C3): Which solver to use (e.g., how to solve the subproblem in SAA or how to update the weight in SA)?*

A unified framework for RLA and SGD

("Weighted SGD for Lp Regression with Randomized Preconditioning," Yang, Chow, Re, and Mahoney, 2015.)



- SA + "naive" P and U: **vanilla SGD** whose convergence rate depends (without additional niceness assumptions) on n
- SA + "smart" P and U: **pwSGD**
- SAA + "naive" P: **uniform sampling RLA algorithm** which may fail if some rows are extremely important (not shown)
- SAA + "smart" P: **RLA (with algorithmic leveraging or random projections)** which has strong worst-case theoretical guarantee and high-quality numerical implementations
- For unconstrained ℓ_2 regression (i.e., LS), SA + "smart" P + "naive" U recovers **weighted randomized Kaczmarz algorithm** [Strohmer-Vershynin].

A combined algorithm: pwSGD

("Weighted SGD for Lp Regression with Randomized Preconditioning," Yang, Chow, Re, and Mahoney, 2015.)

pwSGD:
Preconditioned weighted SGD consists of two main steps:

1. Apply RLA techniques for preconditioning and construct an importance sampling distribution.

2. Apply an SGD-like iterative phase with weighted sampling on the preconditioned system.

A closer look: "naive" choices of U and P in SA

Consider solving ℓ_1 regression; and let $U = A$.
If we apply the SGD with some distribution $P = \{p_i\}_{i=1}^n$, then the relative approximation error is

$$\frac{f(\hat{x}) - f(x^*)}{f(\hat{x})} = \mathcal{O}\left(\frac{\|x^*\|_2 \cdot \max_{1\le i\le n}\|A_i\|_1/p_i}{\|Ax^* - b\|_1}\right),$$

where $f(x) = \|Ax - b\|_1$ and x^* is the optimal solution.

- If $\{p_i\}_{i=1}^n$ is the uniform distribution, i.e., $p_i = \frac{1}{n}$, then

$$\frac{f(\hat{x}) - f(x^*)}{f(\hat{x})} = \mathcal{O}\left(n\frac{\|x^*\|_2 \cdot M}{\|Ax^* - b\|_1}\right),$$

  where $M = \max_{1\le i\le n}\|A_i\|_1$ is the maximum ℓ_1 row norm of A.

- If $\{p_i\}_{i=1}^n$ is proportional to the row norms of A, i.e., $p_i = \frac{\|A_i\|_1}{\sum_{i=1}^n \|A_i\|_1}$, then

$$\frac{f(\hat{x}) - f(x^*)}{f(\hat{x})} = \mathcal{O}\left(\frac{\|x^*\|_2 \cdot \|A\|_1}{\|Ax^* - b\|_1}\right).$$

In either case, the expected convergence time for SGD might blow up (i.e., grow with n) as the size of the matrix grows (*unless one makes extra assumptions*).

A closer look: "smart" choices of U and P in SA

- Recall that if U is a well-conditioned basis, then (by definition) $\|U\|_1 \leq \alpha$ and $\|y^*\|_\infty \leq \beta \|Uy^*\|_1$, for α and β depending on the small dimension d and not the large dimension n.

- If we use a well-conditioned basis U for the range space of A, and if we choose the sampling probabilities proportional to the row norms of U, i.e., leverage scores of A, then the resulting convergence rate on the relative error of the objective becomes

$$\frac{f(\hat{x}) - f(x^*)}{f(\hat{x})} = \mathcal{O}\left(\frac{\|y^*\|_2 \cdot \|U\|_1}{\|\bar{U}y^*\|_1}\right).$$

  where y^* is an optimal solution to the transformed problem.

- Since the condition number $\alpha\beta$ of a well-conditioned basis depends only on d, it implies that *the resulting SGD inherits a convergence rate in a relative scale that depends on d and is independent of n.*

Outline

A combined algorithm: pwSGD

("Weighted SGD for Lp Regression with Randomized Preconditioning," Yang, Chow, Re, and Mahoney, 2015.)

1. Compute $R \in \mathbb{R}^{d\times d}$ such that $U = AR^{-1}$ is an (α, β) well-conditioned basis U for the range space of A.
2. Compute or estimate $\|U_i\|_p^p$ with leverage scores λ_i, for $i \in [n]$.
3. Let $p_i = \frac{\lambda_i}{\sum_{j=1}^n \lambda_j}$, for $i \in [n]$.
4. Construct the preconditioner $F \in \mathbb{R}^{d\times d}$ based on R.
5. For $t = 1, \ldots, T$

   Pick ξ_t from $[n]$ based on distribution $\{p_i\}_{i=1}^n$.

$$c_t = \begin{cases} \operatorname{sgn}\left(A_{\xi_t} x_t - b_{\xi_t}\right)/p_{\xi_t} & \text{if } p = 1; \\ 2\left(A_{\xi_t} x_t - b_{\xi_t}\right)/p_{\xi_t} & \text{if } p = 2. \end{cases}$$

   Update x by

$$x_{t+1} = \begin{cases} x_t - \eta c_t H^{-1} A_{\xi_t} & \text{if } \mathcal{Z} = \mathbb{R}^d; \\ \underset{x\in\mathcal{Z}}{\arg\min}\ \ \eta c_t A_{\xi_t} x + \frac{1}{2}\|x_t - x\|_H^2 & \text{otherwise.} \end{cases}$$

   where $H = \left(FF^\top\right)^{-1}$.
6. $\bar{x} \leftarrow \frac{1}{T}\sum_{t=1}^T x_t$.
7. **Return** $\bar{x}$ for $p = 1$ or x_T for $p = 2$.

Some properties of pwSGD

("Weighted SGD for Lp Regression with Randomized Preconditioning," Yang, Chow, Re, and Mahoney, 2015.)

pwSGD has the following properties:

- It preserves the simplicity of SGD and the high quality theoretical guarantees of RLA.
- After "batch" preconditioning (on arbitrary input), unlike vanilla SGD, the convergence rate of the SGD phase only depends on the low dimension d, i.e., it is independent of the high dimension n.
- Such SGD convergence rate is superior to other related SGD algorithm such as the weighted randomized Kaczmarz algorithm.
- For ℓ_1 regression with size n by d, pwSGD returns an approximate solution with ϵ relative error in the objective value in $\mathcal{O}(\log n \cdot \text{nnz}(A) + \text{poly}(d)/\epsilon^2)$ time (for arbitrary input).
- For ℓ_2 regression, pwSGD returns an approximate solution with ϵ relative error in the objective value and the solution vector measured in prediction norm in $\mathcal{O}(\log n \cdot \text{nnz}(A) + \text{poly}(d) \log(1/\epsilon)/\epsilon)$ time.
- Empirically, pwSGD performs favorably compared to other competing methods, as it converges to a medium-precision solution, e.g., with ϵ roughly 10^{-3}, much more quickly.

Main theoretical bound (ℓ_1 Regression)

Let $f(x) = \|Ax - b\|_1$ and suppose $f(x^*) > 0$. Then there exists a step-size η such that after

$$T = d\bar{\kappa}_1^2(U)\hat{\kappa}^2(RF)\frac{c_1^2 c_2 c_3^2}{\epsilon^2}$$

iterations, pwSGD returns a solution vector estimate $\bar{x}$ that satisfies the expected relative error bound

$$\frac{\mathbb{E}\left[f(\bar{x})\right] - f(x^*)}{f(x^*)} \leq \epsilon.$$

(Above, $c_1 = \frac{1+\gamma}{1-\gamma}$, $c_2 = \frac{\|x^* - x_0\|_H^2}{\|x^*\|_H^2}$, $c_3 = \|Ax^*\|_1 / f(x^*)$ and $\hat{\kappa}^2(RF)$ relates to the condition number of RF.)

Recall: $\bar{\kappa}_1^2(U)$ is the condition number of the basis computed, which only depends on d; F is the preconditioner; γ is the quality of the approximate leverage scores.

Main theoretical bound (ℓ_2 Regression)

Let $f(x) = \|Ax - b\|_2$ and suppose $f(x^*) > 0$. Then there exists a step-size η such that after

$$T = c_1 \bar{\kappa}_2^2(U)\kappa^2(RF) \cdot \log\left(\frac{2c_2\kappa(U)\kappa^2(RF)}{\epsilon}\right) \cdot \left(1 + \frac{\kappa^2(U)\kappa^2(RF)}{c_3\epsilon}\right)$$

iterations, pwSGD returns a solution vector estimate x_T that satisfies the expected relative error bound

$$\frac{\mathbb{E}\left[\|A(x_T - x^*)\|_2^2\right]}{\|Ax^*\|_2^2} \leq \epsilon.$$

Furthermore, when $\mathcal{Z} = \mathbb{R}^d$ and $F = R^{-1}$, there exists a step-size η such that after

$$T = c_1 \bar{\kappa}_2^2(U) \cdot \log\left(\frac{c_2\kappa(U)}{\epsilon}\right) \cdot \left(1 + \frac{2\kappa^2(U)}{\epsilon}\right)$$

iterations, pwSGD returns a solution vector estimate x_T that satisfies the expected relative error bound

$$\frac{\mathbb{E}\left[f(x_T)\right] - f(x^*)}{f(x^*)} \leq \epsilon.$$

(Above, $c_1 = \frac{1+\gamma}{1-\gamma}$, $c_2 = \frac{\|x^* - x_0\|_H^2}{\|x^*\|_H^2}$, $c_3 = \|Ax^*\|_2^2 / f(x^*)^2$.)

Discussion on the choice of the preconditioner F

- Essentially, the convergence rates rely on $\kappa(RF)$. In general, there is a tradeoff between the convergence rate and the computation cost among the choices of the preconditioner F.
- When $F = R^{-1}$, the term $\kappa(RF)$ vanishes in the error bounds; however, an additional $\mathcal{O}(d^2)$ cost per iteration is needed in the SGD update.
- When $F = I$, no matrix-vector multiplication is needed when updating x; however, $\kappa(R) \approx \kappa(A)$ can be arbitrarily large, and this might lead to an ungraceful performance in the SGD phase.
- One can also choose F to be a diagonal preconditioner D, which scales R to have unit column norms. Theoretical results indicate that $\kappa(RD) \leq \kappa(R)$, while the additional cost per iteration to perform SGD updates with diagonal preconditioner is $\mathcal{O}(d)$.

Complexities

There exist choices of the preconditioner such that, with constant probability, one of the following events holds for pwSGD with $F = R^{-1}$. To return a solution $\tilde{x}$ with relative error ϵ on the objective,

- It runs in $time(R) + \mathcal{O}(\log n \cdot \text{nnz}(A) + d^3 \bar{\kappa}_1(U)/\epsilon^2)$ for unconstrained ℓ_1 regression.
- It runs in $time(R) + \mathcal{O}(\log n \cdot \text{nnz}(A) + time_{update} \cdot d\bar{\kappa}_1(U)/\epsilon^2)$ for constrained ℓ_1 regression.
- It runs in $time(R) + \mathcal{O}(\log n \cdot \text{nnz}(A) + d^3 \log(1/\epsilon)/\epsilon)$ for unconstrained ℓ_2 regression.
- It runs in $time(R) + \mathcal{O}(\log n \cdot \text{nnz}(A) + time_{update} \cdot d \log(1/\epsilon)/\epsilon^2)$ for constrained ℓ_2 regression.

In the above, $time(R)$ denotes the time for computing the matrix R and $time_{update}$ denotes the time for solving the optimization problem in update rule of pwSGD (quadratic objective with the same constraints).

Complexity comparisons

solver	complexity (general)	complexity (sparse)
RLA	$time(R) + \mathcal{O}(\text{nnz}(A)\log n + \bar{\kappa}_1^{\frac{3}{2}} d^{\frac{9}{2}}/\epsilon^3)$	$\mathcal{O}(\text{nnz}(A)\log n + d^{\frac{69}{8}} \log^{\frac{25}{8}} d/\epsilon^{\frac{5}{2}})$
randomized IPCPM	$time(R) + nd^2 + \mathcal{O}((nd + \text{poly}(d))\log(\bar{\kappa}_1 d/\epsilon))$	$\mathcal{O}(nd\log(d/\epsilon))$
pwSGD	$time(R) + \mathcal{O}(\text{nnz}(A)\log n + d^3\bar{\kappa}_1/\epsilon^2)$	$\mathcal{O}(\text{nnz}(A)\log n + d^{\frac{13}{2}} \log^{\frac{5}{2}} d/\epsilon^2)$

Table: Summary of complexity of several unconstrained ℓ_1 solvers that use randomized linear algebra. The target is to find a solution $\hat{x}$ with accuracy $(f(\hat{x}) - f(x^*))/f(x^*) \leq \epsilon$, where $f(x) = \|Ax - b\|_1$. We assume that the underlying ℓ_1 regression solver in RLA with algorithmic leveraging algorithm takes $\mathcal{O}(n^{\frac{5}{4}} d^3)$ time to return a solution. Clearly, pwSGD has a uniformly better complexity than that of RLA methods in terms of both d and ϵ, no matter which underlying preconditioning method is used.

solver	complexity (SRHT)	complexity (CW)
low-precision (projection)	$\mathcal{O}\left(nd\log(d/\epsilon) + d^3\log(nd)/\epsilon\right)$	$\mathcal{O}\left(\text{nnz}(A) + d^4/\epsilon^2\right)$
low-precision (sampling)	$\mathcal{O}\left(nd\log n + d^3\log d + d^3\log d/\epsilon\right)$	$\mathcal{O}\left(\text{nnz}(A)\log n + d^4 + d^3\log d/\epsilon\right)$
high-precision solvers	$\mathcal{O}\left(nd\log d + d^3\log d + nd\log(1/\epsilon)\right)$	$\mathcal{O}\left(\text{nnz}(A) + d^4 + nd\log(1/\epsilon)\right)$
pwSGD	$\mathcal{O}\left(nd\log n + d^3\log d + d^3\log(1/\epsilon)/\epsilon\right)$	$\mathcal{O}\left(\text{nnz}(A)\log n + d^4 + d^3\log(1/\epsilon)/\epsilon\right)$

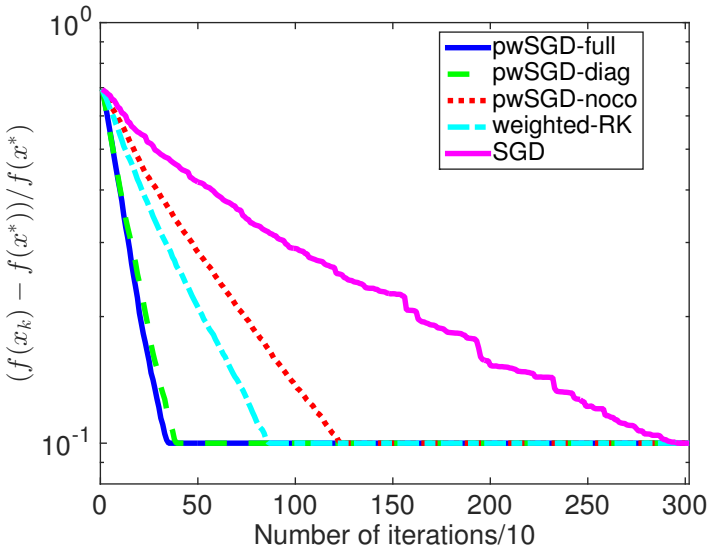
Table: Summary of complexity of several unconstrained ℓ_2 solvers that use randomized linear algebra. The target is to find a solution $\hat{x}$ with accuracy $(f(\hat{x}) - f(x^*))/f(x^*) \leq \epsilon$, where $f(x) = \|Ax - b\|_2$. When $d \geq 1/\epsilon$ and $n \geq d^2/\epsilon$, pwSGD is asymptotically better than the solvers listed above.

Connection to weighted randomized Kaczmarz algorithm

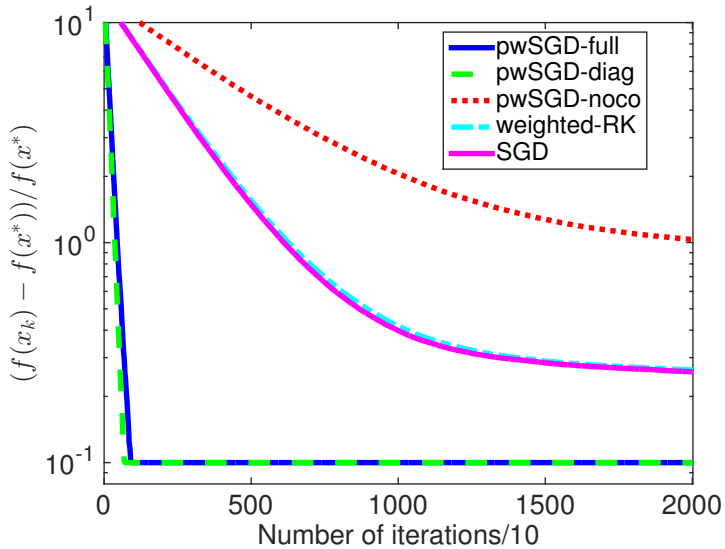
- Our algorithm pwSGD for least-squares regression is related to the weighted randomized Kaczmarz (RK) algorithm [Strohmer and Vershynin].
- Weighted RK algorithm can be viewed as an SGD algorithm with constant step-size that exploits a sampling distribution based on row norms of A, i.e., $p_i = \|A_i\|_2^2 / \|A\|_F^2$.
- In pwSGD, if the preconditioner $F = R^{-1}$ is used and the leverage scores are computed exactly, the resulting algorithm is equivalent to applying the weighted randomized Karczmarz algorithm on a well-conditioned basis U.
- Theoretical results indicate that weighted RK algorithm inherits a convergence rate that depends on condition number $\kappa(A)$ times the scaled condition number $\bar{\kappa}_2(A)$.
- The advantage of preconditioning in pwSGD is reflected here since $\kappa(U) \approx 1$ and $\hat{\kappa}_2(U) \approx \sqrt{d}$.

Outline

Comparison of convergence rates



(a) $\kappa(A) \approx 1$



(b) $\kappa(A) \approx 5$

Figure: Convergence rate comparison of several SGD-type algorithms for solving ℓ_2 regression on two synthetic datasets with condition number around 1 and 5, respectively. For each method, the optimal step-size is set according to the theory with target accuracy $|f(\hat{x}) - f(x^*)|/f(x^*) = 0.1$. The y-axis is showing the relative error on the objective, i.e., $|f(\hat{x}) - f(x^*)|/f(x^*)$.

On datasets with increasing condition number

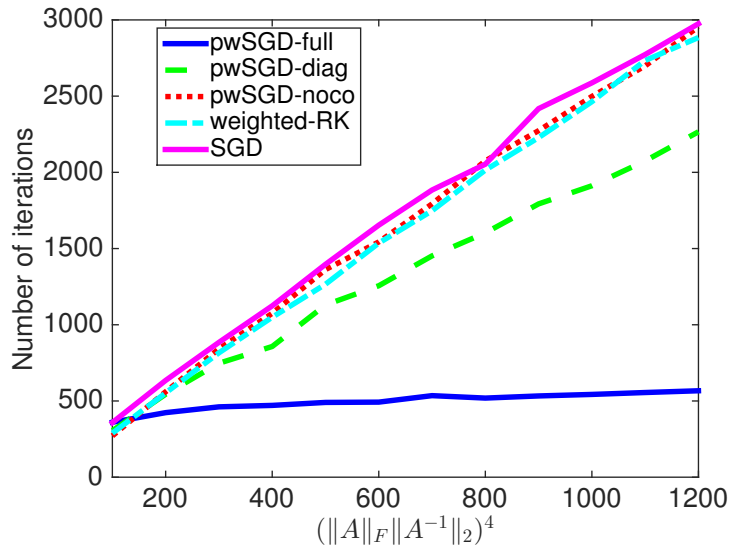
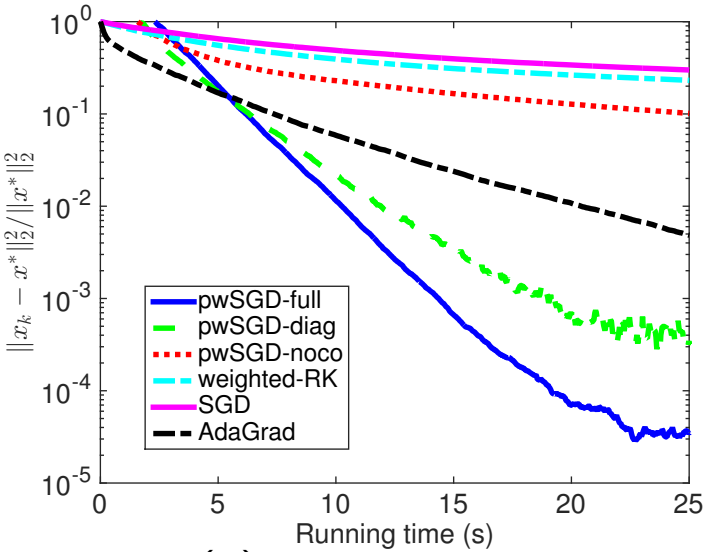
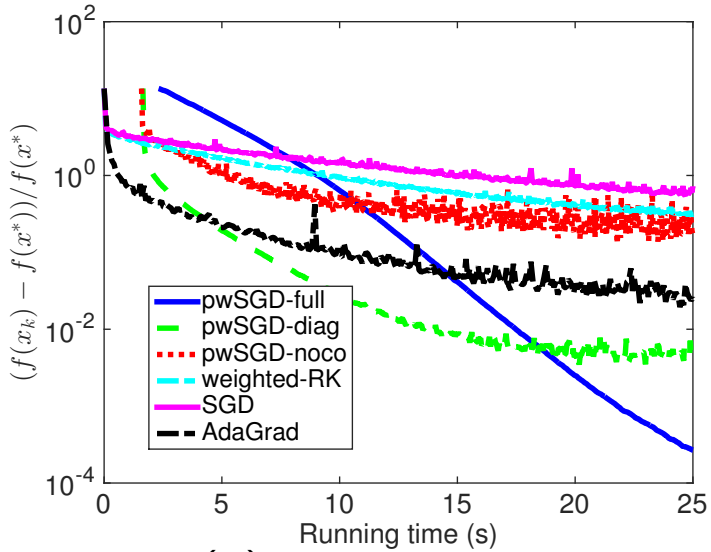


Figure: Convergence rate comparison of several SGD-type algorithms for solving ℓ_2 regression on synthetic datasets with increasing condition number. For each method, the optimal step-size is set according to the theory with target accuracy $|f(\hat{x}) - f(x^*)|/f(x^*) = 0.1$. The y-axis is showing the minimum number of iterations for each method to find a solution with the target accuracy.

Time-accuracy tradeoffs for ℓ_2 regression



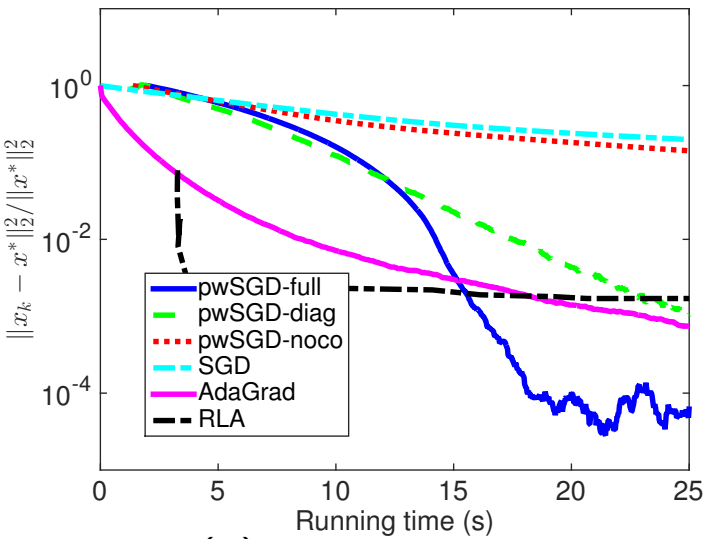
(a) ℓ_2 regression



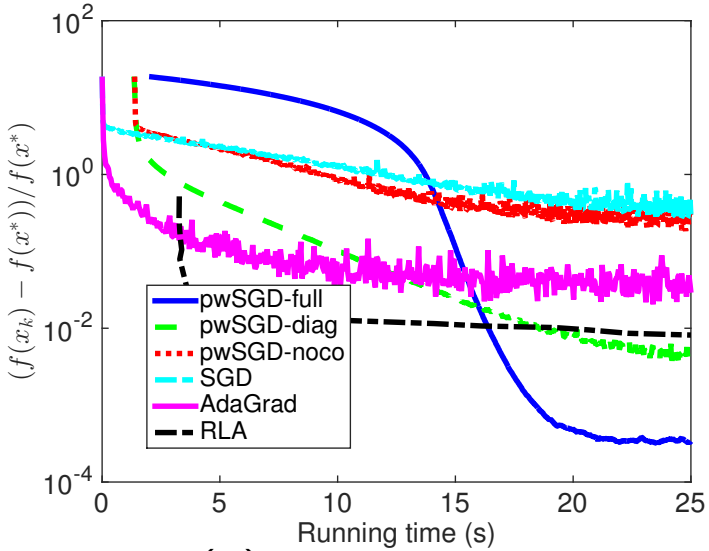
(b) ℓ_2 regression

Figure: Time-accuracy tradeoffs of several algorithms including pwSGD with three different choices of preconditioners on `year` dataset.

Time-accuracy tradeoffs for ℓ_1 regression



(a) ℓ_1 regression



(b) ℓ_1 regression

Figure: Time-accuracy tradeoffs of several algorithms including pwSGD with three different choices of preconditioners on `year` dataset.

Remarks

Compared with general RLA methods:

- For ℓ_2 regression, for which traditional RLA methods are well designed, pwSGD has a comparable complexity.
- For ℓ_1 regression, due to efficiency of SGD update, pwSGD has a strong advantage over traditional RLA methods.

Compared with general SGD methods:

- The RLA-SGD hybrid algorithm pwSGD works for problems in a narrower range, i.e., ℓ_p regression, but inherits the strong theoretical guarantees of RLA.
- Comparison with traditional SGD methods (convergence rates, etc.) depends on the specific objectives of interest and assumptions made.

Outline

Question

After viewing RLA and SGD from the stochastic optimization perspective and using that to develop our main algorithm, a natural question arises:

Can we do this for other optimization/regression problems?

To do so, we need to define "leverage scores" for them, since these scores play a crucial role in this stochastic framework.

Coreset methods

- In [Feldman and Langberg, 2011], the authors propose a framework for computing a "coreset" of $\mathcal{F}$ to a given optimization problem of the following form,

$$\mathrm{cost}(\mathcal{F}, x) = \min_{x \in \mathcal{X}} \sum_{f \in \mathcal{F}} f(x),$$

  where $\mathcal{F}$ is a set of function from a set $\mathcal{X}$ to $[0, \infty)$.

- Let $\bar{A} = \begin{pmatrix} A & b \end{pmatrix}$. The ℓ_p regression problem can be written as

$$\min_{x \in \mathcal{C}} \sum_{i=1}^{n} f_i(x),$$

  where $f_i(x) = |\bar{A}_i x|^p$, in which case one can define a set of functions $\mathcal{F} = \{f_i\}_{i=1}^n$.

A few notions

Sensitivities

Given a set of function $\mathcal{F} = \{f\}$ with size n, the *sensitivity* $m(f)$ of each function is defined as $m(f) = \lfloor \sup_{x \in \mathcal{X}} n \cdot \frac{f(x)}{\mathrm{cost}(\mathcal{F}, x)} \rfloor + 1$, and the *total sensitivity* $M(\mathcal{F})$ of the set of functions is defined as $M(\mathcal{F}) = \sum_{f \in \mathcal{F}} m(f)$.

Dimension of subspaces

The dimension of $\mathcal{F}$ is defined as the smallest integer d, such that for any $G \subset \mathcal{F}$,

$$|\{\textbf{Range}(G, x, r) | x \in \mathcal{X}, r \geq 0\}| \leq |G|^d,$$

where $\textbf{Range}(G, x, r) = \{g \in G | g(x) \leq r\}$.

Algorithm for computing a coreset

1. Initialize $\mathcal{D}$ as an empty set.
2. Compute the sensitivity $m(f)$ for each function $f \in \mathcal{F}$.
3. $M(\mathcal{F}) \leftarrow \sum_{f \in \mathcal{F}} m(f)$.
4. For $f \in \mathcal{F}$
   Compute probabilities
   $$p(f) = \frac{m(f)}{M(\mathcal{F})}.$$
5. For $i = 1, \ldots, s$
   Pick f from $\mathcal{F}$ with probability $p(f)$.
   Add $f/(s \cdot p(f))$ to $\mathcal{D}$.
6. Return $\mathcal{D}$.

Theoretical guarantee

Theorem

Given a set of functions $\mathcal{F}$ from $\mathcal{X}$ to $[0, \infty]$, if $s \geq \frac{cM(\mathcal{F})}{\epsilon^2}(\dim(\mathcal{F}') + \log\left(\frac{1}{\delta}\right))$, then with probability at least $1 - \delta$, the coreset method returns ϵ-coreset for $\mathcal{F}$.

That is,

$$(1-\epsilon)\sum_{f\in\mathcal{F}} f(x) \leq \sum_{f\in\mathcal{D}} f(x) \leq (1+\epsilon)\sum_{f\in\mathcal{F}} f(x).$$

Connection with RLA methods

("Weighted SGD for Lp Regression with Randomized Preconditioning," Yang, Chow, Re, and Mahoney, 2015.)

Fact. Coreset methods coincides the *RLA algorithmic leveraging* approach on LA problems; sampling complexities are the same up to constants!
We show that, when applied to ℓ_p regressions,

- Given $\bar{A} \in \mathbb{R}^{n\times(d+1)}$, let $f_i(x) = |\bar{A}_i x|^p$, for $i \in [n]$.
  Let λ_i be the i-th leverage score of $\bar{A}$. Then,

$$m(f_i) \leq n\beta^p\lambda_i + 1,$$

  for $i \in [n]$, and

$$M(\mathcal{F}) \leq n((\alpha\beta)^p + 1).$$

  This implies the notion of leverage score in RLA is equivalent to the notion of sensitivity in coreset method!

- Let $\mathcal{A} = \{|a^T x|^p | a \in \mathbb{R}^d\}$. We have

$$\dim(\mathcal{A}) \leq d + 1.$$

  This relation and the above theorem imply that the coreset method coincides with the RLA *with algorithmic leveraging* on RLA problems; sampling complexities are the same up to constants!

A negative result

- Beyond ℓ_p regression, coreset methods work for *any* kind of convex loss function.
- Since it depends on the total sensitivity, however, the coreset does *not* necessarily have small size.
- E.g., for hinge loss, we have the following example showing that the size of the coreset has a exponential dependency on d.

Negative example

Define $f_i(x) = f(x, a_i) = (x^T a_i)^+$ where $x, a_i \in \mathbb{R}^d$ for $i \in [n]$.
There exists a set of vectors $\{a_i\}_{i=1}^d$ such that the total sensitivity of $\mathcal{F} = \{f_i\}_{i=1}^n$ is approximately 2^d.

Conclusion

General conclusions:

- Smart importance sampling or random projections needed for good worst-case bounds for machine learning kernel methods
- Data are often—but not always—preprocessed to be "nice," and popular ML metrics often insensitive to a few bad data points
- RLA/SGD are very non-traditional approaches to NLA/optimization; and they can be combined using ideas from stochastic optimization.

Specific conclusions:

- We propose a novel RLA-SGD hybrid algorithm called pwSGD.
- After a preconditioning step and constructing a non-uniform sampling distribution with RLA, its SGD phase inherits fast convergence rates that only depend on the lower dimension of the input matrix.
- Several choices for the preconditioner, with tradeoffs among the choices.
- Empirically, it is preferable when a medium-precision solution is desired.
- Lower bounds on the coreset complexity for more general regression problems, which point to specific directions for to extend these results.