Modern Optimization Methods for Big Data Problems

MATH11146
The University of Edinburgh
Spring 2017

Peter Richtárik

Lecture 7

Randomized Subspace Descent for Convex Minimization: Stepsizes, Importance Sampling and Minibatching

February 13, 2017

How to compute the ESO "stepsize" parameters $v_1, \ldots, v_n$?

M-smooth functions

By definition, the ESO parameters $v = (v_1, \ldots, v_n)$ depend on both f and $\hat{S}$. This is also highlighted by the notation we use:

$$(f, \hat{S}) \sim ESO(v).$$

Hence, in order to compute these parameters, we need to specify f and $\hat{S}$. The following definition describes a wide class of functions, often appearing in computational practice, for which we will be able to do this.

Definition 8 (M-smooth functions)

Let $\mathbf{M} \in \mathbb{R}^{n \times n}$ be a (symmetric) positive semidefinite matrix. We say that a function $f : \mathbb{R}^n \to \mathbb{R}$ is **M-smooth** if it is continuously differentiable and for all $x, h \in \mathbb{R}^n$ satisfies the inequality:

$$f(x + h) \leq f(x) + \langle \nabla f(x), h \rangle + \tfrac{1}{2} h^\top \mathbf{M} h. \tag{19}$$

If $\mathbf{M} = L\mathbf{I}$ for some $L > 0$, we say that f is ***L*-smooth.** The family of all $\mathbf{M}$-smooth functions is denoted by $C^1(\mathbf{M})$.

Proposition 1

Function $f : \mathbb{R}^n \to \mathbb{R}$ is L-smooth if and only if it is differentiable and if $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$ for all $x, y \in \mathbb{R}^n$.

Summation and Composition with a Linear Map

Functions we wish to minimize in **machine learning** often have a **"finite sum" structure:**

$$f(x) \stackrel{\text{def}}{=} \sum_{j=1}^{d} \phi_j(\mathbf{Q}_j x), \tag{20}$$

where $\mathbf{Q}_j \in \mathbb{R}^{q_j \times n}$ are **data matrices** and $\phi_j : \mathbb{R}^{q_j} \to \mathbb{R}$ are L_j**-smooth "loss" functions** for some $L_j > 0$.

The next result says that $f \in C^1(\mathbf{M})$ for some $\mathbf{M}$.

Theorem 9 ([3])

Assume that for each j, function ϕ_j is L_j-smooth. Then $f \in C^1(\mathbf{M})$, where

$$\mathbf{M} = \sum_{j=1}^{d} L_j \mathbf{Q}_j^\top \mathbf{Q}_j.$$

Example 1: Linear Regression (Linear Least Squares)

It can be checked directly that the function $f(x) = \frac{1}{2}\|\mathbf{A}x - b\|^2$ satisfies (19) with an equality for $\mathbf{M} = \mathbf{A}^\top \mathbf{A}$. Hence, $f \in C^1(\mathbf{A}^\top \mathbf{A})$. We can also see this from Theorem 9:

- Consider function f of the form (20), where $\phi_j : \mathbb{R} \to \mathbb{R}$ (i.e., $q_j = 1$) is the **square loss** function

$$\phi_j(s) = \frac{1}{2}(s - y_j)^2,$$

  where y_j, $j = 1, 2, \ldots, d$ are some scalars.
- Let $\mathbf{A}$ be the $d \times n$ matrix with row j equal to $\mathbf{Q}_j$. Then

$$f(x) = \sum_{j=1}^{d} \phi_j(\mathbf{Q}_j x) = \frac{1}{2}\sum_{j=1}^{d} (\mathbf{Q}_j x - y_j)^2 = \frac{1}{2}\|\mathbf{A}x - y\|^2.$$

- The problem $\min_{x \in \mathbb{R}^n} f(x)$ is referred to as **linear regression** or **linear least squares.**
- Note that $\phi_j'(s) = s - y_j$, whence $|\phi_j'(s) - \phi_j'(s')| = |s - s'|$. So, ϕ_j is 1**-smooth** and hence $L_j = 1$.
- By Theorem 9, $f \in C^1(\mathbf{M})$, where

$$\mathbf{M} = \textstyle\sum_{j=1}^{d} L_j \mathbf{Q}_j^\top \mathbf{Q}_j = \mathbf{A}^\top \mathbf{A}.$$

Example 2: Logistic Regression

- Consider function f of the form (20), where $\phi_j : \mathbb{R} \to \mathbb{R}$ (i.e., $q_j = 1$) is the **logistic loss** function:

$$\phi_j(s) = \log\left(1 + e^{-y_j s}\right)$$

- That is,

$$f(x) = \sum_{j=1}^{d} \phi_j(\mathbf{Q}_j x) = \sum_{j=1}^{d} \log\left(1 + e^{-y_j \mathbf{Q}_j x}\right).$$

- The problem $\min_{x \in \mathbb{R}^n} f(x)$ is referred to as **logistic regression.**
- It can be verified that ϕ_j is $1/4$-**smooth.**
- By Theorem 9, $f \in C^1(\mathbf{M})$, where

$$\mathbf{M} = \sum_{j=1}^{d} \frac{1}{4} \mathbf{Q}_j^\top \mathbf{Q}_j = \frac{1}{4} \mathbf{A}^\top \mathbf{A},$$

where, as before, $\mathbf{A}$ is the $d \times n$ matrix with row j equal to $\mathbf{Q}_j$.

ESO for $C_1(\mathbf{M})$ Functions and an Arbitrary Sampling

The following theorem is our main tool for computing the ESO parameters v. We shall use it repeatedly. It enables us to compute v for any $f \in C^1(\mathbf{M})$ and for arbitrary sampling $\hat{S}$.

Theorem 10 ([3])

Assume $f \in C^1(\mathbf{M})$, let $\hat{S}$ be an arbitrary sampling and let $\mathbf{P} = \mathbf{P}(\hat{S})$ be its probability matrix. If $v = (v_1, \ldots, v_n)$ satisfies

$$\mathbf{P} \bullet \mathbf{M} \preceq \mathbf{Diag}\left(p \bullet v\right),$$

where $\bullet$ denotes the Hadamard (elementwise) product of matrices, then

$$(f, \hat{S}) \sim ESO(v).$$

Remark: Sampling $\hat{S}$ enters the computation of v only via its probability matrix $\mathbf{P}$. That is, as long as two samplings have the same probability matrix, they have the same v, and hence in view of Theorem 4, they have the same theoretical properties.

Sampling Identity for a Quadratic

In order to prove Theorem 10, we will need the following lemma.

Lemma 11 ([3])

Let $\mathbf{G}$ *be any real* $n \times n$ *matrix and* $\hat{S}$ *an arbitrary sampling. Then for any* $h \in \mathbb{R}^n$ *we have*

$$\mathrm{E}\left[h_{\hat{S}}^\top \mathbf{G} h_{\hat{S}}\right] = h^\top \left(\mathbf{P}(\hat{S}) \bullet \mathbf{G}\right) h. \tag{21}$$

Proof of Lemma 11

Proof.

Let 1_{ij} be the indicator random variable of the event $i \in \hat{S}$ & $j \in \hat{S}$:

$$1_{ij} = \begin{cases} 1 & \text{if } i \in \hat{S} \ \& \ j \in \hat{S}, \\ 0 & \text{otherwise.} \end{cases}$$

Note that $\mathrm{E}[1_{ij}] = \mathrm{Prob}(i \in \hat{S} \ \& \ j \in \hat{S}) = p_{ij}$. We now have

$$\begin{aligned} \mathrm{E}\left[h_{\hat{S}}^\top \mathbf{G} h_{\hat{S}}\right] &\overset{(2)}{=} \mathrm{E}\left[\sum_{i \in \hat{S}} \sum_{j \in \hat{S}} \mathbf{G}_{ij} h_i h_j\right] = \mathrm{E}\left[\sum_{i=1}^n \sum_{j=1}^n 1_{ij} \mathbf{G}_{ij} h_i h_j\right] \\ &= \sum_{i=1}^n \sum_{j=1}^n \mathrm{E}[1_{ij}] \mathbf{G}_{ij} h_i h_j = \sum_{i=1}^n \sum_{j=1}^n p_{ij} \mathbf{G}_{ij} h_i h_j \\ &= h^\top \left(\mathbf{P}(\hat{S}) \bullet \mathbf{G}\right) h. \end{aligned}$$

□

Proof of Theorem 10

Having established Lemma 11, we are now ready to prove Theorem 10.

Proof.
Fixing any $x, h \in \mathbb{R}^n$, we have

$$\begin{aligned}
\mathrm{E}\left[f(x+h_{\hat{S}})\right] &\overset{(19)}{\leq} \mathrm{E}\left[f(x) + \langle \nabla f(x), h_{\hat{S}}\rangle + \tfrac{1}{2} h_{\hat{S}}^\top \mathbf{M} h_{\hat{S}}\right] \\
&= f(x) + \langle \nabla f(x), h\rangle_p + \tfrac{1}{2}\mathrm{E}\left[h_{\hat{S}}^\top \mathbf{M} h_{\hat{S}}\right] \\
&\overset{\text{(Lemma 11)}}{=} f(x) + \langle \nabla f(x), h\rangle_p + \tfrac{1}{2} h^T \left(\mathbf{P} \bullet \mathbf{M}\right) h \\
&\leq f(x) + \langle \nabla f(x), h\rangle_p + \tfrac{1}{2} \underbrace{h^\top \mathbf{Diag}\left(p \bullet v\right) h}_{\overset{(10)}{=} \|h\|^2_{p \bullet v}}.
\end{aligned}$$

□

Special Cases:
Serial Sampling and Fully Parallel Sampling

ESO: $f \in C^1(\mathbf{M})$ and Serial Sampling $\hat{S}$

Consider applying the NSync algorithm to problem (1) (i.e., to $\min_{x\in\mathbb{R}^n} f(x)$) using a **serial sampling $\hat{S}$:**

$$\text{Prob}(|\hat{S}| = 1) = 1$$

Such a sampling is uniquely characterized by the vector of probabilities $p = (p_1, \dots, p_n)$.

- If $f \in C^1(\mathbf{M})$, then Theorem 10 says that $(f, \hat{S}) \sim ESO(v)$ as long as
$$\mathbf{P}(\hat{S}) \bullet \mathbf{M} \preceq \mathbf{Diag}(p \bullet v).$$
- The probability matrix $\mathbf{P} = \mathbf{P}(\hat{S})$ of a serial sampling is diagonal: $\mathbf{P} = \mathbf{Diag}(p)$. Therefore,
$$\mathbf{P} \bullet \mathbf{M} = \mathbf{Diag}(p) \bullet \mathbf{M} = \mathbf{Diag}(p)\,\mathbf{Diag}(\mathbf{M}) = \mathbf{Diag}(p \bullet v),$$
where
$$v_i = \mathbf{M}_{ii}, \quad i = 1, 2, \dots, n. \tag{22}$$

We have just proved the following **formula for the stepsize parameters:**

Corollary 12

If $f \in C^1(\mathbf{M})$ and $\hat{S}$ is a serial sampling, then $(f, \hat{S}) \sim ESO(v)$, where v is given by (22).

Uniform vs Importance Sampling

Consider running the NSync method with the following setup

- $f \in C^1(\mathbf{M})$ and f is λ-strongly convex
- $\hat{S}$ = **serial sampling**
- Parameters $v_1, \dots, v_n$ are chosen as in (22) (see Corollary 12

Theorem 4 says that the number of iterations of NSync is proportional to

$$\Omega = \max_i \frac{v_i}{p_i}.$$

We shall consider two samplings:

- **Uniform sampling:** $p_i = \frac{1}{n}$ for $i \in [n]$
$$\Omega_{uniform} = \max_i \frac{v_i}{p_i} = \max_i \frac{\mathbf{M}_{ii}}{1/n} = n \max_i \mathbf{M}_{ii} \tag{23}$$
- **Importance sampling:** $p_i = \frac{v_i}{\sum_j v_j}$ for all $i = 1, 2, \dots, n$
$$\Omega_{importance} = \max_i \frac{v_i}{p_i} = \max_i \frac{\mathbf{M}_{ii}}{\mathbf{M}_{ii}/\sum_j \mathbf{M}_{jj}} = \sum_i \mathbf{M}_{ii} = \text{Trace}(\mathbf{M}). \tag{24}$$

It is always the case that $\Omega_{importance} \le \Omega_{uniform}$, and in practice one often has $\Omega_{importance} \ll \Omega_{uniform}$. Hence, **importance sampling is preferable to uniform sampling.**

ESO: $f \in C^1(\mathbf{M})$ and Fully Parallel Sampling $\hat{S}$

Consider applying the NSync algorithm to problem (1) (i.e., to $\min_{x\in\mathbb{R}^n} f(x)$) using a **fully parallel sampling $\hat{S}$:**

$$\mathrm{Prob}(\hat{S} = \{1, 2, \ldots, n\}) = 1$$

- If $f \in C^1(\mathbf{M})$, then Theorem 10 says that $(f, \hat{S}) \sim ESO(v)$ as long as
$$\mathbf{P}(\hat{S}) \bullet \mathbf{M} \preceq \mathbf{Diag}\left(p \bullet v\right).$$
- The probability matrix $\mathbf{P} = \mathbf{P}(\hat{S})$ of the fully parallel sampling is the matrix of all ones. Therefore,
$$\mathbf{P} \bullet \mathbf{M} = \mathbf{M} \preceq \lambda_{\max}(\mathbf{M})\mathbf{I} = \mathbf{Diag}\left(p \bullet v\right),$$
where $\lambda_{\max}(\mathbf{M})$ is the largest eigenvalue of $\mathbf{M}$, $\mathbf{I}$ is the $n \times n$ identity matrix and
$$v_i = \lambda_{\max}(\mathbf{M}), \quad i = 1, 2, \ldots, n. \tag{25}$$

We have just proved the following **formula for the stepsize parameters:**

Corollary 13

If $f \in C^1(\mathbf{M})$ and $\hat{S}$ is the fully parallel sampling, then $(f, \hat{S}) \sim ESO(v)$, where v is given by (25).

Comparison: # of Iterations

- **Uniform sampling:** $\Omega_{uniform} = n \max_i \mathbf{M}_{ii}$

  [n times the maximal diagonal element of $\mathbf{M}$]
- **Importance sampling:** $\Omega_{importance} = \sum_i \mathbf{M}_{ii}$

  [sum of the diagonal elements of $\mathbf{M}$ = sum of the eigenvalues of $\mathbf{M}$]
- **Fully parallel sampling:** $\Omega_{fp} = \lambda_{\max}(\mathbf{M})$

  [maximal eigenvalue of $\mathbf{M}$]

How do these quantities compare?

Theorem 14

$$\frac{\Omega_{importance}}{n} \leq \frac{\Omega_{uniform}}{n} \leq \Omega_{fp} \leq \Omega_{importance} \leq \Omega_{uniform}$$

Proof.

The first inequality is trivial. The second inequality says that the largest eigenvalue of $\mathbf{M}$ is at least as large as the largest diagonal element of $\mathbf{M}$, which is true. The third inequality follow from the fact that the largest eigenvalue of $\mathbf{M}$ is upperbounded by the trace of $\mathbf{M}$. The last inequality is equivalent to the first.

□

Total Complexity for Quadratics: Randomized Coordinate Descent vs Gradient Descent

Efficient Implementation of RCD for Quadratics - Part I

Consider solving the linear least squares problem

$$\min_{x\in\mathbb{R}^n} f(x) \equiv \frac{1}{2}\|\mathbf{A}x-b\|^2, \quad \nabla f(x) = \mathbf{A}^\top(\mathbf{A}x-b), \quad \mathbf{A}\in\mathbb{R}^{m\times n}$$

via **randomized coordinate descent (RCD)** (NSync with a serial sampling):

$$x^{k+1} = x^k + \frac{1}{v_i}\nabla_i f(x^k)e_i. \tag{26}$$

Let x^k be the current iterate, and assume that we already computed the **residual**

$$r^k \overset{\text{def}}{=} \mathbf{A}x^k - b \tag{27}$$

- For instance, if $x^0 = 0$, then $r^0 = -b$ (which costs nothing)
- We do not want to perform the multiplication $\mathbf{A}x^k$ as that would be costly: $\mathcal{O}(\|\mathbf{A}\|_0)$ **arithmetic operations.** We need a way to update the residual more cheaply than this.

How to implement the algorithm efficiently?

Efficient Implementation of RCD for Quadratics - Part II

STEP 1 **Compute partial derivative:** The **residual can be used to compute the partial derivative:**

$$\nabla_i f(x^k) \overset{(4)}{=} e_i^\top \nabla f(x^k) = e_i^\top \mathbf{A}^\top (\mathbf{A}x^k - b) \overset{(27)}{=} \mathbf{A}_{:i}^\top r^k$$

This costs $\mathcal{O}(\|\mathbf{A}_{:i}\|_0)$ **arithmetic operations**

STEP 2 **Update the i^{th} coordinate:** Performing the update (26) costs $\mathcal{O}(1)$ **arithmetic operations.** Indeed, we just need to divide $\nabla_i f(x^k)$ by v_i and add the result to x_i^k, which is one division and one addition.

STEP 3 **Update the residual:** The residual is updated as follows:

$$r^{k+1} = \mathbf{A}x^{k+1} - b = \mathbf{A}(x^k + t^k e_i) - b = r^k + t^k \mathbf{A}e_i = r^k + t^k \mathbf{A}_{:i},$$

where $t^k = \nabla_i f(x^k)/v_i$. So, we need to add to r^k a vector with at most $\mathcal{O}(\|\mathbf{A}_{:i}\|_0)$ nonzero entries. Hence, this step costs $\mathcal{O}(\|\mathbf{A}_{:i}\|_0)$ **arithmetic operations.**

Cost of 1 Iteration of RCD - Part I

Cost of one iteration of **RCD** (NSync with serial sampling) is

$$C_i = \mathcal{O}(\|\mathbf{A}_{:i}\|_0)$$

arithmetic operations (if one picks coordinate i).

Conclusion I: Average/expected cost of 1 iteration

- For any **serial sampling,** the average cost is

$$\bar{C}_{serial} = \mathcal{O}\left(\sum_i p_i \|\mathbf{A}_{:i}\|_0\right) \quad \text{a.o.}$$

- For **uniform sampling,** the average cost is

$$\bar{C}_{uniform} = \mathcal{O}\left(\sum_i \|\mathbf{A}_{:i}\|_0/n\right) = \mathcal{O}(\|\mathbf{A}\|_0/n) \quad \text{a.o.}$$

- If the data matrix $\mathbf{A}$ is dense, then $\|\mathbf{A}_{:i}\|_0 = m$ for all i, and hence for any **serial sampling** the average cost is

$$\bar{C}_{serial,dense} = \mathcal{O}\left(\sum_{i=1}^n p_i \|\mathbf{A}_{:i}\|_0\right) = \mathcal{O}\left(\sum_{i=1}^n \frac{1}{n} m\right) = \mathcal{O}(m) \quad \text{a.o.}$$

Cost of 1 Iteration of RCD - Part II

Using similar arguments, cost of one iteration of **gradient descent (GD)** (NSync with fully parallel sampling) is

$$C_{fp} = \mathcal{O}(\|\mathbf{A}\|_0) \quad \text{a.o.}$$

If the data matrix is dense, the cost is

$$C_{fp,dense} = \mathcal{O}(nm) \quad \text{a.o.}$$

Conclusion II:

- For dense data matrix $\mathbf{A}$, the cost of one iteration of **GD** is n times larger than the (average) cost of one iteration of **RCD** with any serial sampling.
- For any data matrix $\mathbf{A}$, the cost of one iteration of **GD** is n times larger than the (average) cost of one iteration of **RCD** with uniform serial sampling.
- Another way to say this: If you want to compare GD and RCD, one needs to compare "apples with apples": n iterations of RCD $\approx$ 1 iteration of GD.

Complexity: Coordinate Descent vs Gradient Descent

Observations:

- $\Omega_{importance}$ is worse than Ω_{fp}
- $\Omega_{importance}$ is never more than n times larger than Ω_{fp}, and could be equal to Ω_{fp}

Total cost of RCD for dense data $\mathbf{A}$:

$$\Omega_{importance} \times \bar{C}_{serial,dense} = \Omega_{importance} \times \mathcal{O}(m)$$

Total cost of GD for dense data $\mathbf{A}$:

$$\Omega_{fp} \times C_{fp,dense} = \Omega_{fp} \times \mathcal{O}(nm)$$

In terms of total complexity, randomized coordinate descent (with importance sampling) is faster than gradient descent. It can be up to n times faster.

Minibatching

Largest Normalized Eigenvalue of a Probability Matrix

Definition 15 ([3])

For any sampling $\hat{S}$ we define

$$\lambda(\hat{S}) \stackrel{\text{def}}{=} \max_{\theta \in \mathbb{R}^n} \{\theta^\top \mathbf{P} \theta \ : \ \theta^\top \mathbf{Diag}\,(\mathbf{P})\,\theta \leq 1\}, \tag{28}$$

where $\mathbf{P} = \mathbf{P}(\hat{S})$ is the probability matrix associated with $\hat{S}$.

Example 16 (Elementary Sampling)

Fix $S \subseteq [n]$ and let $\hat{S}$ be the sampling for which $\mathrm{Prob}(\hat{S} = S) = 1$. Then

$$\lambda(\hat{S}) = \lambda_{\max}(\mathbf{P}(\hat{S})) = \lambda_{\max}(1_S 1_S^\top) = \|1_S\|_2^2 = |S|, \tag{29}$$

where 1_S is the vector with ones in entries $i \in S$ and zeros elsewhere.

Insightful and Easily Computable Bound

Issues with Theorem 10:

- It does not give a formula for v, but merely gives an inequality which needs to be satisfied for v.
- In order to be able to compute v for more complicated samplings than just serial and fully parallel samplings, we need to go a bit deeper.

The following two results go a good way to overcoming these issues.

Theorem 17 (Useful ESO Formula [3])

Assume $f \in C^1(\mathbf{M})$, where $\mathbf{M} = \mathbf{A}^\top \mathbf{A}$ for some $\mathbf{A} \in \mathbb{R}^{m \times n}$. Let $C_j \overset{def}{=} \{i \ : \ \mathbf{A}_{ji} \neq 0\}$ and let $\hat{S}$ be an arbitrary sampling. Then

$$(f, \hat{S}) \sim ESO(v)$$

for v given by

$$v_i = \sum_{j=1}^{m} \lambda(C_j \cap \hat{S}) \mathbf{A}_{ji}^2, \quad i = 1, 2, \ldots, n. \tag{30}$$

Proof of Theorem 17 - Part I

For any vector $\theta \in \mathbb{R}^n$ and any $j \in \{1, 2, \ldots, m\}$, the following identity holds:

$$\begin{aligned}
\mathrm{E}\left[\left(\sum_{i \in C_j \cap \hat{S}} \theta_i\right)^2\right] &= \mathrm{E}\left[\sum_{i=1}^{n}\sum_{k=1}^{n} (1_{i \in C_j \cap \hat{S} \ \& \ k \in C_j \cap \hat{S}}) \theta_i \theta_k\right] \\
&= \sum_{i=1}^{n}\sum_{k=1}^{n} \mathrm{E}\left[1_{i \in C_j \cap \hat{S} \ \& \ k \in C_j \cap \hat{S}}\right] \theta_i \theta_k \\
&= \sum_{i=1}^{n}\sum_{k=1}^{n} [\mathbf{P}(C_j \cap \hat{S})]_{ik} \theta_i \theta_k \\
&= \theta^\top \mathbf{P}(C_j \cap \hat{S}) \theta. \qquad (31)
\end{aligned}$$

Proof of Theorem 17 - Part II

Fix $h \in \mathbb{R}^n$. Let $z^j = (z_1^j, \ldots, z_n^j)^\top \in \mathbb{R}^n$ be defined as follows: $z_i^j = h_i \mathbf{A}_{ji}$. We then have

$$\begin{aligned}
\mathrm{E}\left[h_{\hat{S}}^\top \mathbf{A}^\top \mathbf{A} h_{\hat{S}}\right] &= \sum_{j=1}^m \mathrm{E}\left[h_{\hat{S}}^\top \mathbf{A}_{j:}^\top \mathbf{A}_{j:} h_{\hat{S}}\right] \overset{(2)}{=} \sum_{j=1}^m \mathrm{E}\left[\left(\sum_{i \in C_j \cap \hat{S}} h_i \mathbf{A}_{ji}\right)^2\right] \\
&\overset{(31)}{=} \sum_{j=1}^m (z^j)^\top \mathbf{P}(C_j \cap \hat{S}) z^j \\
&\overset{(28)}{\leq} \sum_{j=1}^m \lambda(C_j \cap \hat{S}) \left((z^j)^\top \mathbf{Diag}\left(\mathbf{P}(C_j \cap \hat{S})\right) z^j\right) \\
&= \sum_{j=1}^m \lambda(C_j \cap \hat{S}) \sum_{i \in C_j} p_i (h_i \mathbf{A}_{ji})^2 \\
&= \sum_{j=1}^m \lambda(C_j \cap \hat{S}) \sum_{i=1}^n p_i (h_i \mathbf{A}_{ji})^2 \\
&= \sum_{i=1}^n p_i h_i^2 \sum_{j=1}^m \lambda(C_j \cap \hat{S}) \mathbf{A}_{ji}^2 \\
&\overset{(30)}{=} \sum_{i=1}^n p_i h_i^2 v_i.
\end{aligned}$$

Useful bounds on $\lambda(\hat{S})$

Theorem 18 ([3])

Let $\hat{S}$ be an arbitrary sampling.

1. **Lower bound.** *If $\hat{S}$ is not nil, then* $\frac{\mathrm{E}\left[|\hat{S}|^2\right]}{\mathrm{E}\left[|\hat{S}|\right]} \leq \lambda(\hat{S})$.
2. **Upper bound.** *If $|\hat{S}| \leq \tau$ with probability 1, then $\lambda(\hat{S}) \leq \tau$.*
3. **Identity.** *If $|\hat{S}| = \tau$ with probability 1, then $\lambda(\hat{S}) = \tau$.*

Let us apply the 2nd part of the above theorem to the sampling $J \cap \hat{S}$:

Corollary 19

Let $\hat{S}$ be an arbitrary sampling, $J \subseteq [n]$ and c a constant such that $|J \cap \hat{S}| \leq c$ with probability 1. Then

$$\lambda(J \cap \hat{S}) \leq c.$$

In particular, if $|\hat{S}| \leq \tau$ with probability 1, then $|J \cap \hat{S}| \leq \min\{|J|, \tau\}$ with probability 1, and hence $\lambda(J \cap \hat{S}) \leq \min\{|J|, \tau\}$.

Remark: The above corollary is useful as we can apply it in connection with Theorem 17 with $J = C_j$ for $j = 1, 2, \ldots, m$.

Conservative ESO for Arbitrary Minibatch Sampling

Theorem 20 ([3])

Assume $f \in C^1(\mathbf{M})$ with $\mathbf{M} = \mathbf{A}^\top \mathbf{A}$ for some $\mathbf{A} \in \mathbb{R}^{m \times n}$. Let

$$\omega_j \overset{def}{=} |C_j| = |\{i \ : \ \mathbf{A}_{ji} \neq 0\}|.$$

Let $\hat{S}$ be any sampling for which $|\hat{S}| \leq \tau$ with probability 1. Then $(f, \hat{S}) \sim ESO(v)$ with $v = (v_1, \ldots, v_n)$ given by

$$v_i = \sum_{j=1}^m \min\{\omega_j, \tau\} \mathbf{A}_{ji}^2, \qquad i = 1, 2, \ldots, n. \tag{32}$$

Proof.

Corollary 19 implies that $\lambda(C_j \cap \hat{S}) \leq \min\{\omega_j, \tau\}$. The rest follows by applying Theorem 17. □

How to Interpret Theorem 20? (Dense Data)

If the data matrix $\mathbf{A}$ is **dense,** i.e., if $\omega_j = n$ for all j, then Theorem 20 says that

$$v_i \overset{(32)}{=} \tau \sum_{j=1}^m \mathbf{A}_{ji}^2 = \tau \|\mathbf{A}_{:i}\|^2 = \tau \mathbf{M}_{ii}. \tag{33}$$

In view of (34), and since $\mathrm{E}[|\hat{S}|] \leq \tau$, the lower bound (15) on the complexity of NSync says that

$$\Omega_{minibatch,dense} \geq \frac{\sum_i v_i}{\mathrm{E}[|\hat{S}|]} \geq \frac{\tau \sum_i \mathbf{M}_{ii}}{\tau} = \sum_i \mathbf{M}_{ii} \overset{(24)}{=} \Omega_{importance}.$$

So, for dense data, the (conservative) bound we prove for a general minibatch sampling ($\Omega_{minibatch}$) is worse than what we prove for (serial) importance sampling ($\Omega_{importance}$)!

How to Interpret Theorem 20? (Sparse Data)

If the data matrix $\mathbf{A}$ is **sparse,** i.e., if $\omega \overset{\text{def}}{=} \max_j \omega_j \ll n$, then we can use Theorem 20 to write

$$v_i \overset{(32)}{\leq} \sum_{j=1}^m \omega_j \mathbf{A}_{ji}^2 \leq \omega \|\mathbf{A}_{:i}\|^2 = \omega \mathbf{M}_{ii}. \tag{34}$$

In view of (34), the complexity bound (15) of NSync is

$$\Omega_{minibatch,sparse} = \max_i \frac{v_i}{p_i} \leq \omega \max_i \frac{\mathbf{M}_{ii}}{p_i}.$$

Recall from (8) that $\sum_i p_i \overset{(8)}{=} \mathrm{E}[|\hat{S}|] = \tau$. If we assume that $p_i = \tau/n$ for all i (i.e., that $\hat{S}$ is uniform), then

$$\Omega_{minibatch,sparse} \leq \omega \max_i \frac{\mathbf{M}_{ii}}{p_i} = \frac{\omega n}{\tau} \max_i \mathbf{M}_{ii} \overset{(23)}{=} \frac{\omega}{\tau} \Omega_{uniform}.$$

So, as long as we use a large enough minibatch size in relation to the sparsity level ($\tau \geq \omega$), the NSync method based on the minibatch uniform sampling needs less iterations than the NSync method based on the (serial) uniform sampling. This is why parallel coordinate descent takes less time on sparse data than serial coordinate descent.

Minibatching via τ-nice Sampling

It is often advantageous to parallelize coordinate descent by updating more than one coordinate at a time. In practice, one almost always does this by picking subsets of $\{1, 2, \ldots, n\}$ uniformly at random from all subsets of certain cardinality. Such a sampling is called τ-nice, and is defined next:

Definition 21 (τ-nice sampling)

Let $\tau \geq 1$. The **τ-nice sampling** is the sampling $\hat{S}$ with probability mass function given by

$$p_S = \mathrm{Prob}(\hat{S} = S) = \begin{cases} \frac{1}{\binom{n}{\tau}} & \text{if } |S| = \tau, \\ 0 & \text{otherwise.} \end{cases}$$

That is, $\hat{S}$ picks subsets of $\{1, 2, \ldots, n\}$ of cardinality τ, uniformly at random.

Computing $\lambda(J \cap \hat{S})$: τ-Nice Sampling

Theorem 22 (τ-Nice Sampling)

Let $\tau \in \{1, 2, \ldots, n\}$. If $\hat{S}$ is the τ-nice sampling, then the lower bound in part 1 of Theorem 18 is attained for $C_j \cap \hat{S}$ for all j:

$$\lambda(C_j \cap \hat{S}) \;=\; \frac{\mathrm{E}[|C_j \cap \hat{S}|^2]}{\mathrm{E}[|C_j \cap \hat{S}|]} \;=\; 1 + \frac{(\omega_j - 1)(\tau - 1)}{\max\{n-1, 1\}}, \tag{35}$$

where $\omega_j = |C_j|$.

Applying Theorem 17, we get

$$v_i = \sum_{j=1}^m \lambda(C_j \cap \hat{S}) \mathbf{A}_{ji}^2 \overset{(35)}{=} \sum_{j=1}^m \left(1 + \frac{(\omega_j - 1)(\tau - 1)}{\max\{n-1, 1\}}\right) \mathbf{A}_{ji}^2. \tag{36}$$

Plugging this into (15), and using the fact that $p_i = \tau/n$ for all i, we get

$$\Omega_{\tau\text{-nice}} = \max_i \frac{v_i}{p_i} \overset{(35)}{=} \max_i \frac{n \sum_{j=1}^m \left(1 + \frac{(\omega_j - 1)(\tau - 1)}{\max\{n-1, 1\}}\right) \mathbf{A}_{ji}^2}{\tau} \overset{(23)}{\leq} \Omega_{uniform}.$$

NSync always takes fewer iterations with the τ-nice sampling than with the 1-nice (i.e., serial uniform) sampling. When the data is perfectly sparse ($\omega_j = 1$ for all i), then $\Omega_{\tau\text{-nice}} = \frac{1}{\tau}\Omega_{uniform}$, and we achieve perfect linear speedup in the number of processors τ. If $\mathbf{A}$ is sparse (i.e., $\omega_j \ll n$), then we achieve close to linear speedup.