Advanced Topics in Data Sciences

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 08: Randomized Linear Algebra and Stochastic quasi-Newton Method

Laboratory for Information and Inference Systems (LIONS)
École Polytechnique Fédérale de Lausanne (EPFL)

EE-731 (Spring 2016)

License Information for Mathematics of Data Slides

Outline

- Randomized Linear Algebra
  1. Review of last lecture
  2. Row extraction method
  3. Power method
  4. Column selection methods
  5. What to use in different scenarios ?

- Stochastic quasi-Newton Method

Recommended reading material:

- Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, SIAM review 53.2 (2011): 217-288.
- Michael W Mahoney, Randomized algorithms for matrices and data, Foundations and Trends in Machine Learning 3.2 (2011): 123-224.

Recall the last lecture

Matrix decompositions

- SVD and QR decompositions have $\mathcal{O}(np\min\{n,p\})$ complexity.
- Real data is often noisy, so it makes sense to sacrife accuracy for speed-up.
- Randomized methods offer faster and parallelizible approximative solutions that also require a lower number of passes over the data matrix.

Random projections

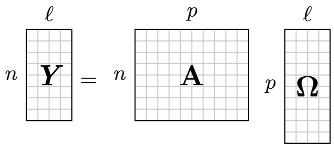
- **Step-1:** Find an orthonormal basis $\mathbf{Q}$ that can approximate $\mathbf{A}$ well:

$$\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^*\mathbf{A}$$

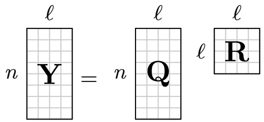
- **Step-2**: Apply classical linear algebra to the smaller matrix $\mathbf{Q}^*\mathbf{A}$.

Recall the last lecture

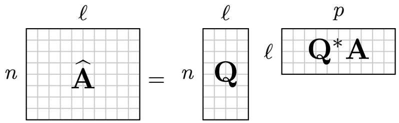
1. Multiply $\mathbf{A\Omega}$ for $\mathbf{\Omega}_{i,j} \sim \mathcal{N}(0,1)$, at cost $\mathcal{O}(np\ell)$ (or $\mathcal{O}(np\log\ell)$ using FFT)



2. Compute thin QR factorization of $\mathbf{Y}$, at a cost of $\mathcal{O}(n\ell^2)$ (e.g. with Gram-Schmidt)

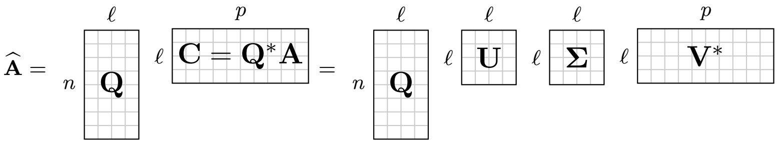


3. Finally multiply $\mathbf{C} = \mathbf{Q}^*\mathbf{A}$ at a cost $\mathcal{O}(np\ell)$. THIS IS THE BOTTLENECK !!!

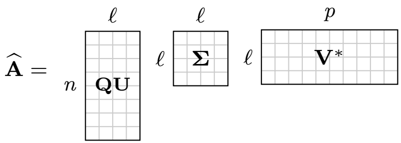


Smaller SVD to form the randomized SVD

1. Find SVD factors of $\mathbf{C}$ in $\mathcal{O}(p\ell^2)$ time.

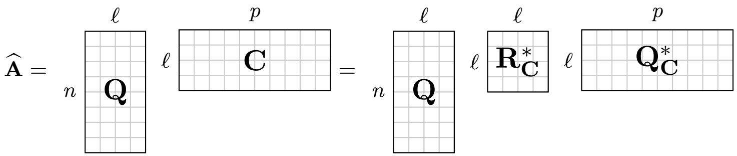


2. Multiply, in $\mathcal{O}(n\ell^2)$ time

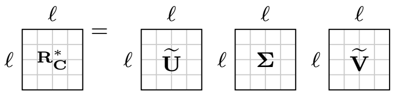


QR decomposition to form the randomized SVD

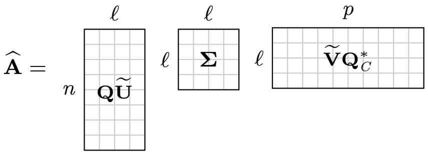
1. Find QR factors of $\mathbf{C}^*$: $\mathbf{C}^* = \mathbf{Q}_\mathbf{C}\mathbf{R}_\mathbf{C}$, in $\mathcal{O}(p\ell^2)$ time.



2. Calculate the SVD of the small inner matrix, in $\mathcal{O}(\ell^3)$ time



3. Multiply in $\mathcal{O}((n+p)\ell^2)$ time


$\widehat{\mathbf{A}} =$
ℓ
n
$\mathbf{Q}\widetilde{\mathbf{U}}$
ℓ
ℓ
$\mathbf{\Sigma}$
p
ℓ
$\widetilde{\mathbf{V}}\mathbf{Q}_C^*$

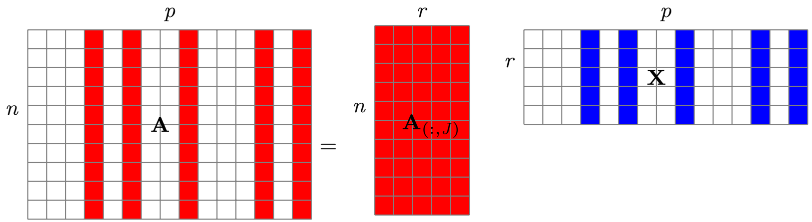
Interpolative decomposition

Definition

For a matrix $\mathbf{A} \in \mathbb{R}^{n\times p}$ of rank-r, a one-sided interpolative decomposition is defined as

$$\mathbf{A} = \mathbf{A}_{(:,J)}\mathbf{X}$$

where $J = [j_1, ...j_r]$ is a computed column index set and $\mathbf{X}$ is a $r \times p$ matrix with $\mathbf{X}_{(:,J)} = \mathbf{I}_r$ and $\mathbf{X}_{i,j} \leq 2 \ \forall i, j$. In other words $\mathbf{A}_{(:,J)}$ is a subset of columns of $\mathbf{A}$ that spans the range of $\mathbf{A}$ with bounded coefficients.



Interpolative decomposition (ID)

Computational Cost

- There exists a decomposition of the form above with $\mathbf{X}$ whose coefficients are bounded by 1, but it is NP-hard to compute. [9]
- When the bound is 2, there are stable and efficient algorithms that computes this decomposition of a rank-r matrix $\mathbf{A} \in \mathbb{R}^{n\times p}$ at a cost of $\mathcal{O}(nrp)$ [7]
- This decomposition can also be generalized to a two-sided form

$$\mathbf{A} = \mathbf{W}\mathbf{A}_{(J',J)}\mathbf{X}$$

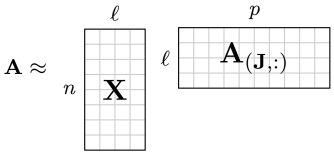
- We need the interpolative decomposition of $\mathbf{Q} \in \mathbb{R}^{n\times l}$: $\mathbf{Q} = \mathbf{X}\mathbf{Q}_{(J,:)}$ where J denotes the l rows of $\mathbf{Q}$ that spans the rowspace of $\mathbf{Q}$ and $\mathbf{X}$ is a $n \times l$ matrix with $\mathbf{X}_{(J,:)} = \mathbf{I}_k$ and $\mathbf{X}_{i,j} \leq 2 \; \forall i,j$. This costs $\mathcal{O}(l^2 n)$.

Row extraction

- Motivation: We want to have something cheaper than forming $\mathbf{Q}^*\mathbf{A}$.
- Given a matrix $\mathbf{Q} \in \mathbb{R}^{n\times r}$ such that $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\| \leq \epsilon$, one can obtain the ID

$$\mathbf{Q} = \mathbf{X}\mathbf{Q}_{(\mathbf{J},:)}$$

- It turns out that [8]



- Then we can perform SVD on these smaller dimensions.

Partial SVD using row extraction

Given is $\mathbf{Q} \in \mathbb{R}^{n\times r}$ such that $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\| \leq \epsilon$.

Algorithm: Partial SVD using row extraction

```
1. Form the row extraction as above: $\mathbf{Q}$: $\mathbf{A} = \mathbf{X}\mathbf{A}_{(J,:)}$ as above $\mathcal{O}(l^2 n)$
2. Form the RQ decomposition: $\mathbf{A}_{(J,:)} = \mathbf{R}^*\mathbf{W}^*$ $\mathcal{O}(l^2 p)$
3. Multiply $\mathbf{Z} = \mathbf{X}\mathbf{R}^*$ $\mathcal{O}(l^2 n)$
4. Compute a classical SVD: $\mathbf{Z} = \mathbf{U}\boldsymbol{\Sigma}\widetilde{\boldsymbol{V}}^*$ $\mathcal{O}(l^2 n)$
5. Multiply $\mathbf{V} = \mathbf{W}\boldsymbol{V}$ and conclude $\mathbf{A} \approx \mathbf{U}\boldsymbol{\Sigma}\widetilde{\boldsymbol{V}}^*$ $\mathcal{O}(l^2 p)$
```

- This costs $\mathcal{O}(l(n+p)^2)$ instead of $npl + \mathcal{O}((n+p)l^2)$ (forming $\mathbf{Q}^*\mathbf{A}$ and performing SVD on it)

Error bound

Lemma (Error bound for Row Extraction [6])

Let $\mathbf{A} \approx \mathbf{U\Sigma} V^*$ *be the output of the algorithm produced by a given* $\mathbf{Q}$ *with an approximation error* ϵ*. Then the following error bound holds:*

$$\|\mathbf{A} - \mathbf{U\Sigma V}^*\| \leq \left[\sqrt{1 + 4k(n-k)}\right] \epsilon$$

In other words, the bound is multiplied by a factor of $\sqrt{1 + 4k(n-k)}$ *if we avoid the multiplication* $\mathbf{Q}^*\mathbf{A}$ *and use row extraction technique.*

Power method

Motivation

- Spectrum of $\mathbf{A}$ does not always decay fast.
- In these cases basic algorithm does not work well.
- This is the case e.g., when the matrix has noise.
- Therefore we apply $(\mathbf{A}\mathbf{A}^*)$ several times to reduce the relative weight of smaller singular values.

Power method

Algorithm: Power method

```
1. Draw a random matrix $\mathbf{\Omega} \in \mathbb{R}^{p \times \ell}$
2. Form the multiplication $\mathbf{Y} = (\mathbf{A}\mathbf{A}^*)^q \mathbf{A}\mathbf{\Omega}$
3. Find the orthogonal $\mathbf{Q}$ that spans the range of $\mathbf{Y}$ (e.g. using Gram-Schmidt)
```

Cost of random projections: $(2q+1)np\ell + \mathcal{O}(q\ell^2 n)$ flops

Practical considerations

- Usually $q = 2$ or $q = 3$ is sufficient
- $\mathbf{\Omega}$ is taken to be Gaussian, the other matrices do not work well
- We perform QR factorization at each step
- Apply $\mathbf{A}$ and $\mathbf{A}^*$ alternatively instead of forming $\mathbf{A}\mathbf{A}^*$

Power method

```
Algorithm: Power method
1. Draw a random matrix $\mathbf{\Omega} \in \mathbb{R}^{p \times \ell}$
2. Form the multiplication $\mathbf{Y} = (\mathbf{A}\mathbf{A}^*)^q \mathbf{A}\mathbf{\Omega}$
3. Find the orthogonal $\mathbf{Q}$ that spans the range of $\mathbf{Y}$ (e.g. using Gram-Schmidt)
```

Cost of random projections: $(2q+1)np\ell + \mathcal{O}(q\ell^2 n)$ flops

Practical considerations

- Usually $q = 2$ or $q = 3$ is sufficient
- $\mathbf{\Omega}$ is taken to be Gaussian, the other matrices do not work well
- We perform QR factorization at each step
- Apply $\mathbf{A}$ and $\mathbf{A}^*$ alternatively instead of forming $\mathbf{A}\mathbf{A}^*$

Power method

Theorem (Power method [6])

Let $\mathbf{A} \in \mathbb{R}^{n\times p}$ *with* $n \geq p$ *be the matrix that is randomly approximated using power method. Then the following holds:*

$$\mathbb{E}\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|_2 \leq \left(1 + \frac{4\sqrt{r+s}}{s-1}\sqrt{p}\right)^{1/(2q+1)} \sigma_{r+1}$$

- When $q = 0$, this is the original algorithm with $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$
- The extra factor can be made close to 1 by increasing the number of the passes, q, but this is at the expense of increasing the computational cost.

Column Selection Methods

Motivation

- So far we have considered **linear combinations** of the columns for reducing the dimensionality of a matrix (SVD)
- Another approach is to find a **subset** of columns that could well summarize action of the matrix
- This makes it easer to interpret for data analysts
- However this is combinatorially hard

Problem (Column Subset Selection Problem)

Given a matrix $\mathbf{A} \in \mathbb{R}^{n \times p}$ and a positive integer r, pick r columns of $\mathbf{A}$ to form a matrix $\mathbf{C} \in \mathbb{R}^{n \times r}$ such that the residual

$$\|\mathbf{A} - \mathbf{P}_{\mathbf{C}}\mathbf{A}\|_{\xi}$$

is minimized over all possible $\binom{n}{r}$ choices for the matrix $\mathbf{C}$. Here, $\mathbf{P}_{\mathbf{C}} = \mathbf{C}\mathbf{C}^{\dagger}$ denotes the projection onto the r-dimensional space spanned by the columns of $\mathbf{C}$ and $\xi = 2$ or F denotes the spectral norm or Frobenius norm. [1]

Column Selection Methods

Motivation

- So far we have considered **linear combinations** of the columns for reducing the dimensionality of a matrix (SVD)
- Another approach is to find a **subset** of columns that could well summarize action of the matrix
- This makes it easer to interpret for data analysts
- However this is combinatorially hard

Problem (Column Subset Selection Problem)

Given a matrix $\mathbf{A} \in \mathbb{R}^{n \times p}$ and a positive integer r, pick r columns of $\mathbf{A}$ to form a matrix $\mathbf{C} \in \mathbb{R}^{n \times r}$ such that the residual

$$\|\mathbf{A} - \mathbf{P}_{\mathbf{C}}\mathbf{A}\|_\xi$$

is minimized over all possible $\binom{n}{r}$ choices for the matrix $\mathbf{C}$. Here, $\mathbf{P}_{\mathbf{C}} = \mathbf{C}\mathbf{C}^\dagger$ denotes the projection onto the r-dimensional space spanned by the columns of $\mathbf{C}$ and $\xi = 2$ or F denotes the spectral norm or Frobenius norm. [1]

Column Selection Methods

Preliminary results

- Uniform sampling of the columns is a bad idea in theory and in practice
- When $\mathcal{O}(r\log(r)/\epsilon^2)$ columns are selected with probabilities proportional to Frobenius norm of columns of A, we have [3].

$$\|\mathbf{A}-\mathbf{P}_{\mathbf{C}_r}\mathbf{A}\|_F \le \|\mathbf{A}-\mathbf{A}_r\|_F + \epsilon\|\mathbf{A}\|_F$$

$$\|\mathbf{A}-\mathbf{P}_{\mathbf{C}_r}\mathbf{A}\|_2 \le \|\mathbf{A}-\mathbf{A}_r\|_2 + \epsilon\|\mathbf{A}\|_F$$

with high probability. $\mathbf{A}_r$ and $\mathbf{C}_r$ are the best rank-r approximations to the matrices $\mathbf{A}$ and $\mathbf{C}$ respectively. ($\mathbf{P}_{\mathbf{X}} = \mathbf{X}\mathbf{X}^\dagger$ is the projection to the column space of $\mathbf{X}$.)

Column Selection Methods

An improved random sampling [4]

Given a singular value decomposition, $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$

- Compute the importance sampling probabilities (leverage scores)

$$p_i = \frac{1}{r}\|{\mathbf{V}_r^*}^{(i)}\|_2^2$$

  where ${\mathbf{V}_r^*}^{(i)}$ is the i^{th} row of matrix $\mathbf{V}_r^*$ that contains the top r right-singular vectors.

- Sample $c = \mathcal{O}(r\log(1/\delta)r/\epsilon^2)$ columns of $\mathbf{A}$ according to this distribution to form a submatrix $\mathbb{C} \in \mathbb{R}^{n\times c}$
- Then with probability at least $1-\delta$ the following holds

$$\|\mathbf{A} - \mathbf{P}_{\mathbf{C}_r}\mathbf{A}\|_2 \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_r\|_F$$

- However approximating the leverage scores is expensive: $\mathcal{O}(np\log n)$

Plenty of methods, what to choose ?

Low rank approximation considerations

- If interpretability is important, then go for column subset selection
- But this might be expensive due to cost of calculating leverage scores
- Otherwise use random projections
- The choice of specific random projection depends on the scenarios below.

Plenty of methods, what to choose ?

Scenario-1: Matrix $\mathbf{A}$ fits in the core memory

- Use a structured random matrix (e.g. SRFT) for Step-1 of low rank approximation using $\mathcal{O}(np\log\ell + \ell^2 n)$ flops
- For Step-2, use the row extraction technique at the cost of $\mathcal{O}(\ell^2(n+p))$
- The total cost is $\mathcal{O}(np\log l + \ell^2(n+p))$
- If the row extraction results in a large error, use the direct method of forming $\mathbf{Q}^*\mathbf{A}$ at Step-2 which costs $\mathcal{O}(np\ell)$

Plenty of methods, what to choose ?

Scenario-2: Matrix $\mathbf{A}$ can be rapidly applied to vectors

- This is the case for sparse matrices or structured matrices such as Toeplitz
- The cost of matrix-vector multiplication could be as low as $C_{mult} = \mathcal{O}(n+p)$
- Step-1 to find $\mathbf{Q}$ costs $\ell C_{mult} + \mathcal{O}(\ell^2 n)$
- In Step-2, form the $\mathbf{Q}^*\mathbf{A}$ as it is now cheap: $\ell C_{mult} + \mathcal{O}(\ell^2(n+p))$
- In total it costs $2\ell C_{mult} + \mathcal{O}(\ell^2(n+p))$
- If the singular values of $\mathbf{A}$ decays slowly, use power method with q iterations which costs $(2q+2)\ell C_{mult} + \mathcal{O}(\ell^2(n+p))$
- Krylov methods would also benefit from this speed-up, but they are less robust and not as parallelizable as these random methods.

Plenty of methods, what to choose ?

Scenario-3: Matrix $\mathbf{A}$ stored in a slow memory

- The computational time is dominated by the memory access, therefore classical methods which require at least r pass over the matrix is not practical.
- One can use any of the randomized algorithms above according to the needs: e.g. if the decay of singular values is slow, use power method, as a small q would be sufficient.

Outline

- Stochastic quasi-Newton Method

Recommended reading materials

1. R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, A stochastic quasi-Newton method for large-scale optimization, SIAM. J. Optim., vol. 26, pp. 1008–1031, 2016.
2. R. M. Gower, D. Goldfarb, and P. Richtárik, Stochastic block BFGS: Squeezing more curvature out of data, arXiv:1603.09649v1, 2016.

Overview

$$\boxed{f^* = \min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x})}$$

• The starting point of many optimization algorithms is to use the following approximation of the objective f at iteration $\mathbf{x}^k$:

$$Q(\mathbf{x}, \mathbf{x}^k) := f(\mathbf{x}^k) + \left\langle \mathbf{x} - \mathbf{x}^k, g(\mathbf{x}^k) \right\rangle + (1/2) \left\langle \mathbf{x} - \mathbf{x}^k, \boldsymbol{H}_k(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k) \right\rangle$$

whose minimum is achieved at

$$\bar{\mathbf{x}}^k = \mathbf{x}^k - [\boldsymbol{H}_k(\mathbf{x}^k)]^{-1} g(\mathbf{x}^k).$$

• Next iteration update:

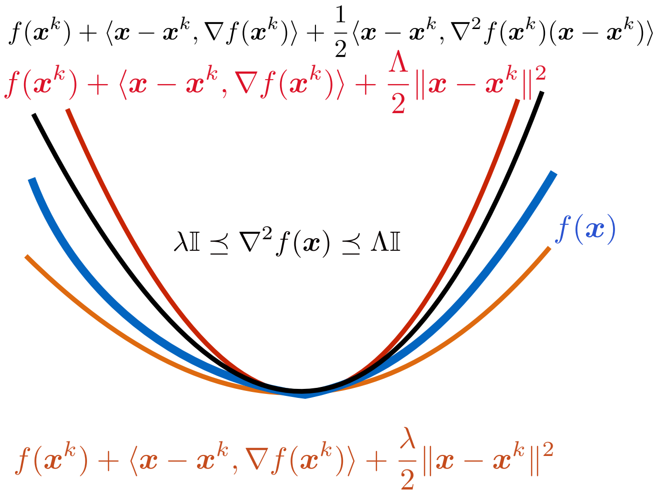
$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k(\bar{\mathbf{x}}^k - \mathbf{x}^k).$$

• Step-size α_k can be updated by line-search.

	Newton	Quasi-Newton	Gradient	SG
g	∇f	∇f	∇f	$\mathbb{E}[g(\mathbf{x}^k)] = \nabla f(\mathbf{x}^k)$
$\boldsymbol{H}_k$	$\nabla^2 f$	$\approx \nabla^2 f$	$L_k\mathbb{I}$	$L_k\mathbb{I}$
rate	quadratic	superlinear	linear	$\mathcal{O}(1/k)$

Table: Taxonomy of some methods.

Curvature



$f(\boldsymbol{x}^k) + \langle \boldsymbol{x} - \boldsymbol{x}^k, \nabla f(\boldsymbol{x}^k)\rangle + \frac{1}{2}\langle \boldsymbol{x} - \boldsymbol{x}^k, \nabla^2 f(\boldsymbol{x}^k)(\boldsymbol{x} - \boldsymbol{x}^k)\rangle$
$f(\boldsymbol{x}^k) + \langle \boldsymbol{x} - \boldsymbol{x}^k, \nabla f(\boldsymbol{x}^k)\rangle + \frac{\Lambda}{2}\|\boldsymbol{x} - \boldsymbol{x}^k\|^2$
$\lambda \mathbb{I} \preceq \nabla^2 f(\boldsymbol{x}) \preceq \Lambda \mathbb{I}$
$f(\boldsymbol{x})$
$f(\boldsymbol{x}^k) + \langle \boldsymbol{x} - \boldsymbol{x}^k, \nabla f(\boldsymbol{x}^k)\rangle + \frac{\lambda}{2}\|\boldsymbol{x} - \boldsymbol{x}^k\|^2$

- $\nabla^2 f$ controls the curvature of f's graph. Information on $\nabla^2 f$ leads $Q(\mathbf{x}, \mathbf{x}^k)$ to be better approximation of f.
- $(\forall \mathbf{x} \in \mathbb{R}^p)\ \nabla^2 f(\mathbf{x}) \succeq \mu \mathbb{I}$ (i.e.,$(\forall \mathbf{y} \in \mathbb{R}^p)\ \mathbf{y}^T \nabla^2 f(\mathbf{x}) \mathbf{y} \geq \mu \|\mathbf{y}\|^2 \Rightarrow f$ is μ-strongly convex.

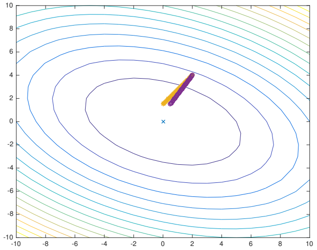
Newton method

Newton algorithm

1. Initialize $\mathbf{x}^0 \in \mathbb{R}^p$.
2. For $k = 0, 1, \ldots$ perform:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - [\nabla^2 f(\mathbf{x}^k)]^{-1} \nabla f(\mathbf{x}^k).$$

• What curvature helps (yellow - gradient direction; violet - gradient direction modified by Hessian):



• Quadratic convergence!

Quasi-Newton methods

- High computation cost if the Hessian is dense!
- Idea: Approximate the Hessian to avoid second derivative computations using first-order information.

Quasi-Newton scheme

```
1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$, set $\boldsymbol{H}_0 = \mathbb{I}$, and compute $f(\mathbf{x}^0)$ and $\nabla f(\mathbf{x}^0)$.
2. For $k = 0, 1, \ldots$ perform:
   2a. Compute $\mathbf{v}^k = \boldsymbol{H}_k \nabla f(\mathbf{x}^k)$.
   2b. Compute $\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \mathbf{v}^k$.
   2c. Compute $f(\mathbf{x}^{k+1})$ and $\nabla f(\mathbf{x}^{k+1})$.
   2d. Update $\boldsymbol{H}_{k+1}$ based on $f(\mathbf{x}^{k+1})$ and $\nabla f(\mathbf{x}^{k+1})$.
```

Question

How to update $\boldsymbol{H}_k$?

- For $f(\mathbf{x}) = (1/2)\mathbf{x}^T\mathbf{A}\mathbf{x} + \boldsymbol{b}^T\mathbf{x} + c$: $\nabla f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \boldsymbol{b}$ and $\nabla^2 f = A$. Hence

$$\nabla f(\mathbf{x}) - \nabla f(\mathbf{y}) = \mathbf{A}(\mathbf{x} - \mathbf{y}) = \nabla^2 f(\mathbf{z})(\mathbf{x} - \mathbf{y}).$$

- Update rule:

$$\boldsymbol{H}_{k+1}(\nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k)) = \mathbf{x}^{k+1} - \mathbf{x}^k.$$

Quasi-Newton method: some update rules

Example ($\mathbf{v}^k = \nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k)$, $\mathbf{y}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$)

- Rank-one correction:

$$\boldsymbol{H}_{k+1} = \boldsymbol{H}_k + \frac{(\mathbf{y}^k - \boldsymbol{H}_k\mathbf{v}^k)(\mathbf{y}^k - \boldsymbol{H}_k\mathbf{v}^k)^T}{(\mathbf{y}^k - \boldsymbol{H}_k\mathbf{v}^k)^T\mathbf{v}^k}.$$

- Davidon-Fletcher-Powell:

$$\boldsymbol{H}_{k+1} = \boldsymbol{H}_k + \frac{\mathbf{y}^k(\mathbf{y}^k)^T}{(\mathbf{v}^k)^T\mathbf{y}^k} - \frac{\boldsymbol{H}_k\mathbf{v}^k(\mathbf{v}^k)^T\boldsymbol{H}_k}{(\mathbf{v}^k)^T\boldsymbol{H}_k\mathbf{v}^k}.$$

- Broyden-Fletcher-Goldfarb-Shanno:

$$\boldsymbol{H}_{k+1} = \boldsymbol{H}_k + \frac{\boldsymbol{H}_k\mathbf{v}^k(\mathbf{y}^k)^T + \mathbf{y}^k(\mathbf{v}^k)^T\boldsymbol{H}_k}{(\mathbf{v}^k)^T\boldsymbol{H}_k\mathbf{v}^k} - \beta_k\frac{\boldsymbol{H}_k\mathbf{v}^k(\mathbf{v}^k)^T\boldsymbol{H}_k}{(\mathbf{v}^k)^T\boldsymbol{H}_k\mathbf{v}^k},$$

where

$$\beta_k = 1 + \frac{(\mathbf{v}^k)^T\mathbf{y}^k}{(\mathbf{v}^k)^T\boldsymbol{H}_k\mathbf{v}^k}.$$

Stochastic approximation of Hessian

Problem

We consider the following problem

$$\boxed{f^* = \min_{\mathbf{x}\in\mathbb{R}^p} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) \right\}}$$

where f_i, $i = 1, \ldots, n$, are twice continuously differentiable and their Hessians $\nabla^2 f_i$ are positive definite.

• In the high dimensional regime where both n and p are large: evaluation of the gradient and Hessian can be computationally prohibitive.
• Idea: Use stochastic approximations:

- sub-sampled gradient;
- sub-sampled Hessian;
- sub-sampled approximation of Hessian.

Stochastic BFGS

Stochastic BFGS algorithm [2]

```
1. Set $t = -1$ and choose $L \in \mathbb{N}$ and $M \in \mathbb{N}$.
2. For $k = 0, 1, \ldots$ perform:
2a. Choose a sample $\mathcal{S}$.
2b. Calculate stochastic gradient $\nabla_{\mathcal{S}} f(\mathbf{x}^k)$.
2c. Update:
    $$\mathbf{x}^{k+1} = \begin{cases} \mathbf{x}^k - \alpha_k \nabla_{\mathcal{S}} f(\mathbf{x}^k), & \text{if } t < 1, \\ \mathbf{x}^k - \alpha_k \boldsymbol{H}_t \nabla_{\mathcal{S}} f(\mathbf{x}^k), & \text{otherwise.} \end{cases}$$
2d. When $\mathrm{mod}(k, L) = 0$, perform:
    2d1. $t = t + 1$.
    2d2. $\bar{\mathbf{x}}^t = \frac{1}{L} \sum_{j=k-L+1}^{k} \mathbf{x}^j$.
    2d3. Choose a sample $\mathcal{T} \subset \{1, \ldots, n\}$.
    2d4. Compute:
         $$\mathbf{y}^t = \bar{\mathbf{x}}^t - \bar{\mathbf{x}}^{t-1}, \quad \mathbf{v}^t = \nabla^2_{\mathcal{T}} f(\bar{\mathbf{x}}^t)(\bar{\mathbf{x}}^t - \bar{\mathbf{x}}^{t-1}).$$
2e. Update Hessian:
    2e1. Compute: $\boldsymbol{H} = (\mathbf{y}^t)^T \mathbf{v}^t / ((\mathbf{v}^t)^T \mathbf{v}^t) \mathbb{I}$.
    2e2. For $j = t - \min\{t, M\} + 1, \ldots, t$ perform:
    $$\rho_j = 1/((\mathbf{y}^j)^T \mathbf{v}^j), \quad \boldsymbol{H}_t = (\mathbb{I} - \rho_j \mathbf{y}^j (\mathbf{v}^j)^T) \boldsymbol{H} (\mathbb{I} - \rho_j \mathbf{v}^j (\mathbf{y}^j)^T) + \rho_j \mathbf{y}^j (\mathbf{y}^j)^T.$$
```

- $\boldsymbol{H}_t$ is result of applying M BFGS updates using the M most recent pairs $(\mathbf{y}^t, \mathbf{v}^t)$.

Stochastic BFGS (cont.)

- [2]: computational cost of stochastic BFGS could be much cheaper than SGD.
- One does not have to compute Hessian but the directional derivative along a vector

$$\nabla^2_{\mathcal{S}} f(\mathbf{x})(\mathbf{v}) = \frac{d}{d\alpha} \nabla f_{\mathcal{S}}(\mathbf{x} + \alpha \mathbf{v})\Big|_{\alpha=0}.$$

Theorem ($\mathcal{O}(1/k)$ rate of stochastic BFGS [2])

Take $\alpha_k = \beta/k$ with $\beta > 1/(2\mu_1\lambda)$ and suppose that:

1. $(\forall \mathcal{Q} \subset \{1, \ldots, n\})(\forall \mathbf{x} \in \mathbb{R}^p)$: $\lambda \mathbb{I} \preceq \nabla^2_{\mathcal{Q}} f(\mathbf{x}) \preceq \Lambda \mathbb{I}$,
   $\left(\nabla^2_{\mathcal{Q}} f(\mathbf{x}) = \frac{1}{|\mathcal{W}|} \sum_{i \in \mathcal{Q}} \nabla^2 f_i(\mathbf{x})\right)$.
2. $\mathbb{E}[\|\nabla f(\mathbf{x}^k)\|^2] \leq \gamma^2$.

Then

1. *There exists (μ_1, μ_2) such that $\mu_1 \mathbb{I} \preceq \boldsymbol{H}_k \preceq \mu_2 \mathbb{I}$.*
2. *The following holds:*
   $$\mathbb{E}[f(\mathbf{x}^k) - f^*] \leq Q(\beta)/k,$$
   where
   $$Q(\beta) = \max\left\{ \frac{\Lambda \mu_2^2 \beta^2 \gamma^2}{2(2\mu_1 \lambda \beta - 1)}, f(\mathbf{x}^0) - f^* \right\}.$$

- The convergence rate does not depend on condition number of problem (i.e., Λ/λ).

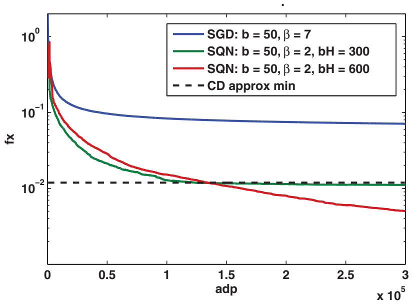
Numerical experiment (from [2])

- Problem:

$$\min_{\mathbf{x}\in\mathbb{R}^p} -\frac{1}{n}\sum_{i=1}^{n} y_i \log(c(\mathbf{x};\mathbf{x}_i)) + (1-y_i)\log(1-c(\mathbf{x};\mathbf{x}_i));$$

where $c(\mathbf{x};\mathbf{x}_i) = 1/(1+\exp(-\mathbf{x}_i^T\mathbf{x}))$ and $y_i \in \{0,1\}$.

- $\nabla f(\mathbf{x}) = (c(\mathbf{x};\mathbf{x}_i) - y_i)\mathbf{x}_i; \quad \nabla^2 f(\mathbf{x})(\mathbf{v}) = c(\mathbf{x};\mathbf{x}_i)(1-c(\mathbf{x};\mathbf{x}_i))(\mathbf{x}_i^T\mathbf{v})\mathbf{x}_i.$



After stochastic BFGS

- Stochastic BFGS: the rate $\mathcal{O}(1/k)$ is similar to that of SG.
- Newton: quadratic; Quasi-Newton: superlinear!
- Could one obtain linear rate?
- Yes! with sub-sampled Hessian.
  - Small sub-sampled Hessian: cost of Newton method is not much larger than the cost of gradient evaluation.
  - Large sub-sampled Hessian: more curvature information.
  - Challenge: achieve the right balance.

Sub-sampled Hessian

Hessian sub-sampling Newton algorithm [10]

1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$, $\beta \in (0,1)$ and $\hat{\alpha} \geq 1$ and set sample size s.
2. For $k = 0, 1, \ldots$ perform:
2a. Select $\mathcal{S} \subset \{1, \ldots, n\}$ of size s, compute $\mathcal{S}$-sub-sampled Hessian $\nabla^2_{\mathcal{S}} f(\mathbf{x}^k)$.
2b. Compute $\boldsymbol{v}^k = -[\nabla^2_{\mathcal{S}} f(\mathbf{x}^k)]^{-1} \nabla f(\mathbf{x}^k)$ and solve

$$\alpha_k = \arg\max \ \alpha \quad \text{s.t.} \quad \alpha \leq \hat{\alpha} \text{ and } f(\mathbf{x}^k + \alpha \boldsymbol{v}^k) \leq f(\mathbf{x}^k) + \alpha\beta(\boldsymbol{v}^k)^T \nabla f(\mathbf{x}^k).$$

2c. Update

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \boldsymbol{v}^k.$$

- $\mathcal{S}$-sub-sampled Hessian:

$$\nabla^2_{\mathcal{S}} f(\mathbf{x}) := \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \nabla^2 f_i(\mathbf{x}).$$

Convergence

Theorem (Linear rate convergence [10])

Suppose that $0 \preceq \nabla^2 f_i(\mathbf{x}) \preceq \Lambda_i \mathbb{I}$ *and* $\lambda \mathbb{I} \preceq \nabla^2 f(\mathbf{x}) \preceq \Lambda \mathbb{I}$ *for every* $\mathbf{x} \in \mathbb{R}^p$. *Given* $\varepsilon \in (0,1)$, $\delta \in (0,1)$, *and* $\mathbf{x} \in \mathbb{R}^p$ *and suppose that*

$$|\mathcal{S}| \geq \frac{2\kappa_1 \ln(p/\delta)}{\varepsilon^2}.$$

With probability $1-\delta$, *one has*

$$f(\mathbf{x}^{k+1}) - f^* \leq (1-\rho_k)\big(f(\mathbf{x}^k) - f^*\big),$$

where $\rho_k = \frac{2\alpha_k \beta}{\tilde{\kappa}}$. *Furthermore,*

$$\alpha_k \geq \frac{2(1-\beta)(1-\varepsilon)}{\kappa}.$$

- $\kappa = \Lambda/\lambda$ and $\tilde{\kappa} = \begin{cases} \kappa_1, & \text{if } \mathcal{S} \text{ is drawn with replacement,} \\ \kappa_{|\mathcal{S}|}, & \text{if } \mathcal{S} \text{ is drawn without replacement.} \end{cases}$

(Here, given $q \in [1, n]$: $\hat{\Lambda}_q$ is the average of q largests Λ_i and $\kappa_q = \hat{\Lambda}_q/\lambda$).

- By choosing suitable β and ε, ρ_k can be smaller than the condition number $\rho = \Lambda/\lambda$ of F (recall that convergence rate of GD is $1-\rho$).

Combining Hessian sub-sampling and gradient sub-sampling

Hessian sub-sampling Newton algorithm [10]

```
1. Choose $\mathbf{x}^0 \in \mathbb{R}^p$, $\beta \in (0,1)$ and $\hat{\alpha} \geq 1$ and set sample sizes $s$ and $t$.
2. For $k = 0, 1, \ldots$ perform:
2a. Select sample sets $\mathcal{S} \subset \{1, \ldots, n\}$ of size $s$ and $\mathcal{T} \subset \{1, \ldots, n\}$ of size $t$.
2b. Compute $\nabla^2_{\mathcal{S}} f(\mathbf{x}^k)$ and $\nabla_{\mathcal{T}} f(\mathbf{x}^k)$.
2c. Compute $\boldsymbol{v}^k = -[\nabla^2_{\mathcal{S}} f(\mathbf{x}^k)]^{-1} \nabla_{\mathcal{T}} f(\mathbf{x}^k)$ and solve
      $\alpha_k = \arg\max \ \alpha \quad \text{s.t.} \quad \alpha \leq \hat{\alpha} \text{ and } f(\mathbf{x}^k + \alpha \boldsymbol{v}^k) \leq f(\mathbf{x}^k) + \alpha\beta(\boldsymbol{v}^k)^T \nabla_{\mathcal{T}} f(\mathbf{x}^k).$
2d. Update
      $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \boldsymbol{v}^k.$
```

- $\mathcal{T}$-sub-sampled gradient:

$$\nabla_{\mathcal{T}} f(\mathbf{x}) = \frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} \nabla f_i(\mathbf{x}).$$

- $\mathcal{S}$-sub-sampled Hessian:

$$\nabla^2_{\mathcal{S}} f(\mathbf{x}) := \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \nabla^2 f_i(\mathbf{x}).$$

Convergence

Theorem (Linear rate convergence [10])

Suppose that $0 \preceq \nabla^2 f_i(\mathbf{x}) \preceq \Lambda_i \mathbb{I}$ *and* $\lambda \mathbb{I} \preceq \nabla^2 f(\mathbf{x}) \preceq \Lambda \mathbb{I}$ *for every* $\mathbf{x} \in \mathbb{R}^p$*. Given* $\varepsilon_1 \in (0,1)$, $\varepsilon_2 \in (0, 1/2)$, $\delta \in (0,1)$*, and* $\mathbf{x} \in \mathbb{R}^p$ *and suppose that*

$$|\mathcal{S}| \geq \frac{2\kappa_1 \ln(p/\delta)}{\varepsilon_1^2} \text{ and } |\mathcal{T}| \geq \frac{\max_{1 \leq i \leq n} \sup_{k \in \mathbb{N}} \|\nabla f_i(\mathbf{x}^k)\|^2}{\varepsilon_2^2} \left(1 + \sqrt{8 \ln\left(\frac{1}{\delta}\right)}\right)^2.$$

With probability $1 - \delta$*, one has*

$$f(\mathbf{x}^{k+1}) - f^* \leq (1 - \rho_k)\Big(f(\mathbf{x}^k) - f^*\Big),$$

where $\rho_k = \frac{8\alpha_k \beta}{9\tilde{\kappa}}$*. Furthermore,*

$$\alpha_k \geq \frac{(1-\beta)(1-\varepsilon_1)}{\kappa}.$$

- $\kappa = \Lambda/\lambda$ and $\tilde{\kappa} = \begin{cases} \kappa_1, & \text{if } \mathcal{S} \text{ is drawn with replacement,} \\ \kappa_{|\mathcal{S}|}, & \text{if } \mathcal{S} \text{ is drawn without replacement.} \end{cases}$
  (Here, given $q \in [1, n]$: $\hat{\Lambda}_q$ is the average of q largests Λ_i and $\kappa_q = \hat{\Lambda}_q/\lambda$)

Stochastic block BFGS

$$\boxed{f^* = \min_{\mathbf{x}\in\mathbb{R}^p} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) \right\}}$$

• The sub-sampled Hessian with respect to the sample set $\mathcal{S} \subset \{1,\ldots,n\}$: $\nabla^2_{\mathcal{S}} f(\mathbf{x}) = \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} \nabla^2 f_i(\mathbf{x})$, can be computationally expensive when the dimension p is large.
• Recent idea: Using Hessian-vector product $\nabla^2_{\mathcal{S}} f(\mathbf{x})(\mathbf{v})$ where $\mathbf{v}$ is a suitable selected vector and combining with variance reduction.
• Update rule: normally $\boldsymbol{H}_t \nabla^2_{\mathcal{S}} f(\mathbf{x}^t) = \mathbb{I}$ but to reduce computational cost

$$\boldsymbol{H}_t \nabla^2_{\mathcal{S}} f(\mathbf{x}^t)\boldsymbol{D}_t = \boldsymbol{D}_t,$$

where $\boldsymbol{D}_t \in \mathbb{R}^{d\times q}$, $(q \ll p)$ is a random matrix. Hence

$$\boldsymbol{H}_t = \boldsymbol{D}_t\Delta_t\boldsymbol{D}_t^T + (\mathbb{I} - \boldsymbol{D}_t\Delta_t\boldsymbol{Y}_t^T)\boldsymbol{H}_{t-1}(\mathbb{I} - \boldsymbol{Y}_t\Delta_t\boldsymbol{D}_t),\ \Delta_t = (\boldsymbol{D}_t^T\boldsymbol{Y}_t)^{-1},\ \boldsymbol{Y}_t = \nabla^2_{\mathcal{S}} f(\mathbf{x}^t)\boldsymbol{D}_t.$$

• d large: can not store $\boldsymbol{H}_t$ and hence, store M block triples

$$(\boldsymbol{H}_i, \boldsymbol{Y}_i, \Delta_i)_{t+1-M\le i\le t}$$

and using $\boldsymbol{V}_t = \mathbb{I} - \boldsymbol{D}_t\Delta_t\boldsymbol{Y}_t^T$ and

$$\boldsymbol{H}_t = \boldsymbol{V}_t \ldots \boldsymbol{H}_{t+1-M}\boldsymbol{H}_{t-M}\boldsymbol{V}_{t+1-M}^T \ldots \boldsymbol{V}_t^T + \sum_{i=t}^{t+1-M} \boldsymbol{V}_t \ldots \boldsymbol{V}_{i+1}\boldsymbol{D}_i\Delta_i\boldsymbol{D}_i^T\boldsymbol{V}_{i+1}^T \ldots \boldsymbol{V}_t^T$$

Stochastic block BFGS

Stochastic block BFGS algorithm [5]

Inputs: $\mathbf{x}^0 \in \mathbb{R}^p$, stepsize $\eta > 0$, $s =$ subsample size, $q =$ sample action size, $m =$ size of the inner loop, $M =$ memory parameter.

```
1. Initiate: $H_{-1} = \mathbb{I}$.
2. For $k = 0, 1, 2, \ldots$ perform:
2a. Compute the full gradient $\nabla f(\mathbf{x}^k)$.
2b. Set $\mathbf{y}^0 = \mathbf{x}^k$.
2c. For $t = 0, \ldots, m-1$, perform:
2c1. Sample $\mathcal{S}_t$ and $\mathcal{T}_t$ in $\{1, \ldots, n\}$, independently.
2c2. Compute: $\mathbf{v}^t = \nabla_{\mathcal{S}_t} f(\mathbf{y}^t) - \nabla_{\mathcal{S}_t} f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)$.
2c3. Form $\mathbf{A}_t \in \mathbb{R}^{p \times p}$ so that rank$(\mathbf{A}_t) = q$.
2c4. Compute $\boldsymbol{Y}_t = \nabla^2_{\mathcal{T}_t} f(\mathbf{y}^t)\mathbf{A}_t$.
2c5. Compute $\mathbf{A}_t{}^T \boldsymbol{Y}_t$ and its Cholesky factorization to obtain $\Delta_t = (\mathbf{A}_t^T \boldsymbol{Y}_t)^{-1}$.
2c6. For $i = 1, \ldots, t$, perform:
    $\begin{cases} \alpha_i = \Delta_i \mathbf{A}_i^T \mathbf{v}^t \text{ and } \mathbf{v}^t \leftarrow \mathbf{v}^t - \boldsymbol{Y}_i \alpha_i, & \text{for } i = t, \ldots, t-M+1, \\ \beta_i = \Delta_i \boldsymbol{Y}_i^T \mathbf{v}^t \text{ and } \mathbf{v}^t \leftarrow \mathbf{v}^t + \mathbf{A}_i(\alpha_i - \beta_i), & \text{for } i = t-M+1, \ldots, t. \end{cases}$
2c7. Set $\mathbf{y}^{t+1} = \mathbf{y}^t - \eta \mathbf{v}^t$.
2d. Update $\mathbf{x}^{k+1} = \mathbf{y}^m$.
```

Stochastic block BFGS: Convergence

$$\boxed{f^\star = \min_{\mathbf{x}\in\mathbb{R}^p} \left\{ f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) \right\}}$$

Theorem (Linear rate convergence of stochastic block BFGS [5])

Suppose that

$$(\forall \mathcal{T} \subset \{1,\ldots,n\})(\forall \mathbf{x}\in\mathbb{R}^p) \quad \lambda\mathbb{I} \preceq \nabla^2_{\mathcal{T}} f(\mathbf{x}) \preceq \Lambda\mathbb{I},$$

where $\nabla^2_{\mathcal{T}} f(\mathbf{x}) = \frac{1}{|\mathcal{T}|}\sum_{i\in\mathcal{T}} \nabla^2 f_i(\mathbf{x})$. *Then:*

1. *There exist* (γ,Γ) *such that* $\gamma\mathbb{I} \preceq \boldsymbol{H}_t \preceq \Gamma\mathbb{I}$.
2. *Suppose that* $\eta < \gamma\lambda/(2\Gamma^2\Lambda^2)$ *and that*

$$m \geq \frac{1}{2\eta\left(\gamma\lambda - \eta\Gamma^2\Lambda(2\Lambda-\lambda)\right)}.$$

Then

$$\mathbb{E}[f(\mathbf{x}^k) - f^\star] \leq \rho^k (f(\mathbf{x}^0) - f^\star),$$

where

$$\rho = \frac{1/(2m\eta) + \eta\Gamma^2\Lambda(\Lambda-\lambda)}{\gamma\lambda - \eta\Gamma^2\Lambda^2} < 1.$$

References I

[1] Christos Boutsidis, Michael W Mahoney, and Petros Drineas.
An improved approximation algorithm for the column subset selection problem.
In Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 968–977. Society for Industrial and Applied Mathematics, 2009.

[2] R. H. Byrd, Hansen, S. L., Jorge Nocedal, and Y. Singer.
A stochastic quasi-newton method for large-scale optimization.
SIAM J. Optim., 26:1008–1031, 2016.

[3] Petros Drineas, Ravi Kannan, and Michael W Mahoney.
Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix.
SIAM Journal on Computing, 36(1):158–183, 2006.

[4] Petros Drineas, Michael W Mahoney, and S Muthukrishnan.
Relative-error cur matrix decompositions.
SIAM Journal on Matrix Analysis and Applications, 30(2):844–881, 2008.

[5] R. M. Gower, D. Goldfarb, and P. Richtárik.
Stochastic block BFGS: Squeezing more curvature out of data.
http://arxiv.org/abs/1603.09649v1, 2016.

References II

[6] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp.
Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions.
SIAM review, 53(2):217–288, 2011.

[7] Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert.
Randomized algorithms for the low-rank approximation of matrices.
Proceedings of the National Academy of Sciences, 104(51):20167–20172, 2007.

[8] Per-Gunnar Martinsson, Vladimir Rockhlin, and Mark Tygert.
A randomized algorithm for the approximation of matrices.
Technical report, DTIC Document, 2006.

[9] C-T Pan.
On the existence and computation of rank-revealing lu factorizations.
Linear Algebra and its Applications, 316(1):199–222, 2000.

[10] F. Roosta-Khorasani and M. W. Mahoney.
Sub-sampled newton methods I: Globally convergent algorithms.
http://arxiv.org/abs/1601.04737v3, 2016.