Probabilistic Algorithms for Finding Matrix Decompositions

Sparsity and Compressed Sensing
MVA 2017/18 - ENS Cachan

Alex Nowak

08/01/2018

Problem: Low rank approximation of a matrix

$$\underset{m \times n}{\boldsymbol{A}} \approx \underset{m \times k}{\boldsymbol{B}} \; \underset{k \times n.}{\boldsymbol{C},}$$

- **Standard Decompositions:**
  1. SVD:
  $$\boldsymbol{A} = \left(\boldsymbol{U}\Sigma^{1/2}\right)\left(\boldsymbol{V}\Sigma^{1/2}\right)^*$$
  2. QR:
  $$\boldsymbol{A} = \boldsymbol{QR}$$
- **Classical Algorithms:**
  1. Computationally expensive: $\mathcal{O}(mnk)$.
  2. Need $\mathcal{O}(k)$ passes over data.
  3. Can't deal with inaccurate matrices.

$\Longrightarrow$ Not adequate to deal with massive datasets!

Two Stages Solution

- **Stage A:** (*Randomized*) Find $m \times k$ orthonormal $\boldsymbol{Q}$ whose columns approximate the range of $\boldsymbol{A}$:

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

- **Stage B:** (*Deterministic*) Use $\boldsymbol{Q}$ to find the desired decomposition. E.g, set $\boldsymbol{B} = \boldsymbol{Q}$ and $\boldsymbol{C} = \boldsymbol{Q}^*\boldsymbol{A}$

Stage A: Randomize!

Randomized Range Finder

1. Draw an $n \times \ell$ Gaussian random matrix $\boldsymbol{\Omega}$.
2. Form the $m \times \ell$ matrix $\boldsymbol{Y} = \boldsymbol{A\Omega}$.
3. Construct an $m \times \ell$ matrix $\boldsymbol{Q}$ whose columns form an orthonormal basis for the range of $\boldsymbol{Y}$, e.g., using the QR factorization $\boldsymbol{Y} = \boldsymbol{QR}$.

- If we set $\ell = k + p$ with $p > 0$ ($p << k$) the *oversampling parameter*, we can control the error $\|\boldsymbol{A} - \boldsymbol{QQ}^*\boldsymbol{A}\|$! (See Bound)

$$\mathbb{E}\,\|\boldsymbol{A} - \boldsymbol{QQ}^*\boldsymbol{A}\| \leq \left(1 + \sqrt{\frac{k}{p-1}}\right)\sigma_{k+1} + \frac{\mathrm{e}\sqrt{k+p}}{p}\left(\sum\nolimits_{j>k}\sigma_j^2\right)^{1/2}.$$

- But, error large if singular values $(\sigma_i)_{i=1}^{r}$ decay slowly...
- But, product $\boldsymbol{A\Omega}$ too expensive... $\mathcal{O}(mn\ell)$

Stage A: Solving the two issues.

- If $\boldsymbol{A}$ has slow decaying singular values $(\sigma_i)_{i=1}^{r}$, take powers!

$$\boldsymbol{Y} = \boldsymbol{B}\boldsymbol{\Omega} \quad \boldsymbol{B} = (\boldsymbol{A}\boldsymbol{A}^*)^q \boldsymbol{A} \quad \sigma_j(\boldsymbol{B}) = \sigma_j(\boldsymbol{A})^{2q+1}, \quad j = 1, 2, 3, \dots$$

$$\frac{\mathbb{E}\,\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\|}{\sigma_{k+1}} \leq \left[1 + \sqrt{\frac{k}{p-1}} + \frac{\mathrm{e}\sqrt{k+p}}{p} \cdot \sqrt{\min\{m,n\} - k}\right]^{1/(2q+1)}$$

- Make the product $\boldsymbol{A}\boldsymbol{\Omega}$ cheaper by using a *structured random matrix* instead of Gaussian.
  Use *Subsampled Random Fourier Transform*(SRFT)

$$\boldsymbol{\Omega} = \sqrt{n/\ell} \cdot \boldsymbol{D}\boldsymbol{F}\boldsymbol{R}^*$$

  Can compute the sample matrix $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$ in $\mathcal{O}(mn\log(\ell))$ operations via a *subsampled FFT*.

Stage B: Construct $\boldsymbol{A} \approx \boldsymbol{BC}$ decomposition from $\boldsymbol{Q}$.

- **Direct SVD:** Set $\boldsymbol{B} = \boldsymbol{Q}$ and $\boldsymbol{C} = \boldsymbol{Q}^*\boldsymbol{A}$. Construct SVD of $\boldsymbol{C} = \boldsymbol{U}_1 \Sigma \boldsymbol{V}^*$ and set $\boldsymbol{U} = \boldsymbol{B}\boldsymbol{U}_1$

$\Longrightarrow$ Product $\boldsymbol{Q}^*\boldsymbol{A}$ costs $\mathcal{O}(mnk)$, too expensive!

$\Longrightarrow$ SVD with *Direct SVD* requires $\mathcal{O}(k)$ passes over data!

- *Solution 1*: Complexity can be reduced to $\mathcal{O}(k^2(m+n))$ via *row extraction*, but an additive error term comes up.
- *Solution 2*: Use *single-pass* algorithms. Adds additional error too.

Full Algorithms

- **General Matrices That Fit in Core Memory**
  1. Stage 1: Use *Structured Random Matrix*.
  2. Stage 2: Use *row-extraction*.

$$\boxed{T_{\mathrm{random}} \sim mn\log(k) + k^2(m+n)}$$

- **Matrices for which Matrix-Vector Products are Cheap.**
  1. Stage 1: Use *Randomized Power method*.
  2. Stage 2: Use *Direct SVD*

$$\boxed{T_{\mathrm{sparse}} = (2q+2)\,(k+p)\,T_{\mathrm{mult}} + \mathcal{O}(k^2(m+n))}$$

Theory: Expectation and high probability error bounds.

Obs: Deterministic stage.

- **Expectation bound:**

$$\mathbb{E}\,\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\| \leq \left(1 + \sqrt{\frac{k}{p-1}}\right)\sigma_{k+1} + \frac{\mathrm{e}\sqrt{k+p}}{p}\left(\sum\nolimits_{j>k}\sigma_j^2\right)^{1/2}.$$

  with probability

- **Tail bound:**

$$\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\| \leq$$

$$\leq \left(1 + 8\sqrt{(k+p)\cdot p\log p}\right)\sigma_{k+1} + 3\sqrt{k+p}\left(\sum\nolimits_{j>k}\sigma_j^2\right)^{1/2},$$

  with failure probability at most $6p^{-p}$..

  Remark: Very fast decay with *oversampling parameter* p.

Experiment: Compare theoretical bound with numerical error for powers of normalized gaussian matrices $\boldsymbol{A}$.

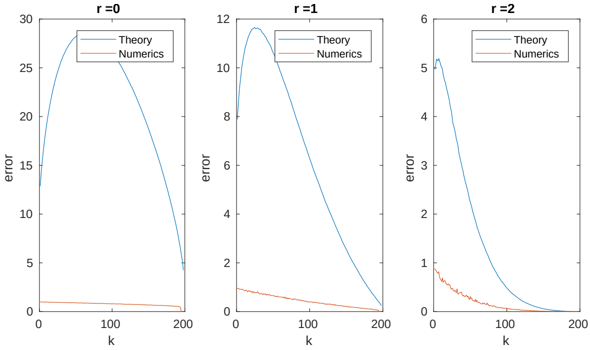

Figure: Comparison between theoretical mean bound and numerical error produced by *Randomized Range Finder* for $(\boldsymbol{A}\boldsymbol{A}^*)^r\boldsymbol{A}$ with $r = 0, 1, 2$.

Experiment: Compare theoretical bound with numerical error for powers of normalized gaussian matrices *A*.

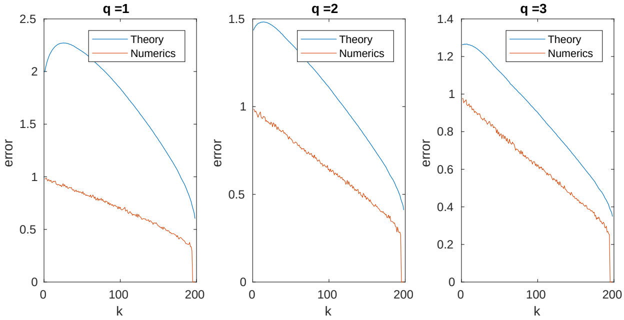

Figure: Comparison between theoretical mean bound and numerical error produced by *Randomized Power Iteration* for $\boldsymbol{A}$ and $q = 1, 2, 3$

Experiment: MNIST and Laplacian eigenvectors of image patches.

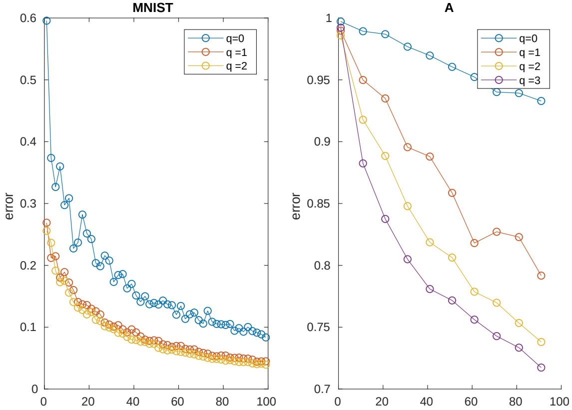

Figure: **Left:** Fast decaying singular values. **Right:** Slow decaying singular values.

Take Away Message / Future directions / Critics

- Necessary tool for Data Scientists working with massive or inaccurate datasets.
- Methods with strong experimental evidence backed up with theory.
- Possible future directions: Improve error bounds sharpness under additional hypothesis of matrix $\boldsymbol{A}$.
- Critics: In many applications, the target rank k already satisfies $k << m, n$. (So log-term not really necessary...)
- Critics: Sometimes properties of matrix $\boldsymbol{A}$ hard to know in advance.

Gràcies!