EPFL

HPC for numerical methods and data analysis

Fall Semester 2024

Prof. Laura Grigori

Assistant: Mariana Martínez Aguilar

Session 7 – October 15, 2023

Sketching techniques

In the context of overdetermined least-squares problems, we need to find $x \in \mathbb{R}^n$ such that it minimizes:

$$\|Ax - b\|_2^2,$$

where $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, m > n$. There is a class of randomized algorithms for solving this problem based on sketching method. Sketching methods involve using a random matrix $\Omega \in \mathbb{R}^{r \times m}$ to project the data A (and maybe also b) to a lower dimensional space with $r \ll m$. Then they approximately solve the least-squares problem using the sketch ΩA (and/or Ωb). One relaxes the problem to finding a vector x so that

$$\|Ax - b\| \leq (1 + \varepsilon)\|Ax^* - b\|,$$

where x^* is the optimal solution. The overview of sketching applied to solve linear least squares is:

a) Sample/build a random matrix Ω

b) Compute ΩA and Ωb

c) Output the exact solution to the problem $\min_x \|(\Omega A) - (\Omega)b\|_2$.

Exercise 1: General properties of sketching techniques

An $\varepsilon-$subspace embedding for the column space of a $m \times n$ matrix A is a matrix Ω for which for all $x \in \mathbb{R}^n$ the following property is satisfied with high probability:

$$(1 - \varepsilon)\|Ax\|_2 \leq \|\Omega Ax\|_2^2 \leq (1 + \varepsilon)\|Ax\|_2^2. \tag{1}$$

Let U be a matrix whose columns form an orthonormal basis for the column space of A. Prove that the requirement of an $\varepsilon-$ subspace embedding can be simplified to:

$$\|I - U^\top \Omega^\top \Omega U\|_2 \leq \epsilon. \tag{2}$$

Solution: if s is the rank of A then the following sets are equal:

$$\{Uy : y \in \mathbb{R}^s\} = \{Ax : x \in \mathbb{R}^n\}.$$

Hence, without loss of generality we can assume that A has orthonormal columns. Then the requirement for a $(1 \pm \varepsilon)$ l_2-subspace embedding becomes:

$$\|\Omega U y\|_2^2 = (1 \pm \varepsilon)\|Uy\|_2^2 = (1 \pm \varepsilon)\|y\|_2^2.$$

If this requirement is satisfied for unit vectors y, then it is satisfied for all vectors y by scaling. Then:

$$\|I - U^\top \Omega^\top \Omega U\|_2 = \max\{\|(I - U^\top \Omega^\top \Omega U)y\|_2 : \|y\|_2 = 1\} \leq \varepsilon$$

Exercise 2: Gaussian

The most "classical" sketch is a matrix $\Omega \in \mathbb{R}^{r \times m}$ with independent and identically distributed (i.i.d.) Gaussian entries $\mathcal{N}(0, 1/r)$. The following theorem from [1] provides the optimal number of rows of Ω up to a constant factor $\mathcal{O}(r\epsilon^{-2})$:

Theorem 1 *Let $0 < \varepsilon, \delta < 1$ and $\Omega = \frac{1}{\sqrt{r}} R \in \mathbb{R}^{r \times m}$ where the entires $R_{i,j}$ of R are independent standard normal random variables. Then if $r = \mathcal{O}((n + \log(1/\delta))\varepsilon^{-2})$, then for any fixed $m \times n$ matrix A, with probability $1 - \delta$, Ω is a $(1 \pm \varepsilon)$ l_2-subspace embedding for A, that is, simultaneously for all $x \in \mathbb{R}^n$,*

$$\|\Omega A x\|_2 = (1 \pm \varepsilon)\|Ax\|_2 \tag{3}$$

Choose a data set from [https://www.kaggle.com/datasets?tags=13405-Linear+Regression]. Compare the linear regression obtained from solving the deterministic least squares problem vs the one obtained from the randomized least squares problem with $\Omega \in \mathbb{R}^{r \times m}$ a normal random variable. That is, using the previous theorem with $\delta = 0,99$ choose different values of ε and compare the difference between the randomized least squares fit vs the deterministic one. Check that (2) holds for every ε you choose.
Solution: The following script does this and generates important plots:

```
import numpy as np
from numpy.linalg import norm, lstsq
from pandas import read_csv
from numpy.random import normal
from math import ceil, log, sqrt
import matplotlib.pyplot as plt
import time

plt.ion()

# For Gaussian sketching applied to a least squares problem
# we report the following quantities:
##### Time taken to solve the full problem
##### Time taken to solve the compressed problem
##### Residual norm full problem
```

```
##### Residual norm compressed problem
##### Relative error in the spectral norm


# We are going to read the data (which was previously downloaded)
# We just want to work with certain columns, not all of them
d = read_csv("ParisHousing.csv")
b = d.price
b = b.values
d.drop(['hasYard', 'hasPool', 'floors', 'cityCode', 'numPrevOwners',
        'made', 'basement', 'attic', 'garage', 'hasGuestRoom'], axis = 1)
A = d.values


# Now that we have out set up
m, n = A.shape
nRuns = 10
sigma = 0.99
epsilon = np.array([100, 10, 5, 2, 1, 0.5, 0.1])
rVec = np.ceil( (n + log(1/sigma))*epsilon**(-2) ).astype('int')
# Notice that some r's might be bigger than m

timeF = np.empty_like(epsilon)
timeC = np.empty_like(epsilon)
resF = np.empty_like(epsilon)
resC = np.empty_like(epsilon)
relErrSpec = np.empty_like(epsilon)

for k in range(len(epsilon)):
    eps = epsilon[k]
    r = rVec[k]
    tF = 0
    tC = 0
    rC = 0
    rES = 0
    for run in range(nRuns):
        # Begin with the compressed problem
        ts = time.time()
        omega = 1/sqrt(r)*normal(loc = 0, scale = 1.0, size = (r, m))
        omegaA = omega@A
        omegab = omega@b
        xPrime = lstsq(omegaA, omegab)
        xPrime = xPrime[0]
        tC += time.time() - ts
        # Now for the full problem
        ts = time.time()
        xStar = lstsq(A, b)
        xStar = xStar[0]
        tF += time.time() - ts
        # Report desired quantities for the randomized part
        rC += norm(omegaA@xPrime - omegab)
        rES += abs(norm(omegaA) - norm(A))/norm(A)
    # Save averages
    timeF[k] = tF/nRuns
    timeC[k] = tC/nRuns
    resF[k] = norm(A@xStar - b)
    resC[k] = rC/nRuns
    relErrSpec[k] = rES/nRuns
```

```
###
### Plot plot plot
# Time
plt.figure(figsize=(8, 6), dpi=80)
plt.loglog(epsilon, timeF, c = "#003aff", marker = 'o',
           label = "Full problem")
plt.loglog(epsilon, timeC, c = "#00b310", marker = '*',
           label = "Compressed problem")
plt.legend()
plt.title(r'$\varepsilon$' +
          ", time taken to build and compute")
plt.xlabel(r'$\varepsilon$')
plt.ylabel("Time, s")

# Norm of residual
plt.figure(figsize=(8, 6), dpi=80)
plt.loglog(epsilon, resF, c = "#003aff", marker = 'o',
           label = "Full problem")
plt.loglog(epsilon, resC, c = "#00b310", marker = '*',
           label = "Compressed problem")
plt.legend()
plt.title(r'$\varepsilon$' + ", norm of residual")
plt.xlabel(r'$\varepsilon$')
plt.ylabel("Norm of residual")

# Relative error in spectral norm
plt.figure(figsize=(8, 6), dpi=80)
plt.loglog(epsilon, relErrSpec, c = "#5400b3", marker = 'o',
           label = "Relative error")
plt.loglog(epsilon, epsilon, c = '#676b74', linestyle='dashed',
           label = r'$\varepsilon$')
plt.legend()
plt.title(r'$\varepsilon$' + ", relative error spectral norm " +
          r'$| \|\Omega A\|_2 - \|A\|_2 |/\| A\|_2$')
plt.xlabel(r'$\varepsilon$')
plt.ylabel(r'$| \|\Omega A\|_2 - \|A\|_2 |/\| A\|_2$')
```

Exercise 3: SRHT

Given a data matrix, $X \in \mathbb{R}^{m \times n}$, we want to reduce the dimensionality of X by defining a random orthonormal matrix $\Omega \in \mathbb{R}^{r \times m}$ with $r \ll m$. For $m = 2^q, q \in \mathbb{N}$, the Subsampled Randomized Hadamard Transform (SRHT) algorithm defined a $r \times m$ matrix as:

$$\Omega = \sqrt{\frac{m}{r}} P H_m D,$$

where:

- $D \in \mathbb{R}^{m \times m}$ is a diagonal matrix whose elements are independent random signs, i.e. it's diagonal entries are just -1 or 1.
- $H \in \mathbb{R}^{m \times m}$ is a **normalized** Walsh-Hadamard matrix. If you're going to use a library that implements this transform then check that it implements the normalized Walsh-Hadamard

matrix. This matrix is defined recursively as:

$$H_m = \begin{bmatrix} H_{m/2} & H_{m/2} \\ H_{m/2} & -H_{m/2} \end{bmatrix} \qquad H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$
$$H = \frac{1}{\sqrt{m}} H_m \in \mathbb{R}^{m \times m}.$$

- $P \in \mathbb{R}^{r \times m}$ is a subset of randomly sampled r columns from the $m \times m$ identity matrix. The purpose of using P is to uniformly sample r columns from the rotated data matrix $X_{\text{rot}} = H_m D X$.

The following theorem help us get an idea for the size of r.

Theorem 2 (Subsampled Randomized Hadamard Transform) *Let $\Omega = \sqrt{\frac{m}{r}} P H_m D$ as previously defined. Then if*

$$r \geq \mathcal{O}((\varepsilon^{-2} \log(n))(\sqrt{n} + \sqrt{\log m})^2)$$

with probability $0,99$ for any fixed $U \in \mathbb{R}^{m \times n}$ with orthonormal columns:

$$\|I - U^\top \Omega \Omega^\top U\|_2 \leq \varepsilon.$$

Further, for any vector $x \in \mathbb{R}^m, \Omega x$ can be computed in $\mathcal{O}(n \log r)$ time.

Take the same data set from the previous exercise. Compare the randomized least squares fit using SRHT vs the deterministic least squares fit. Use the previous theorem to estimate r.
We are going to work on this next week. Solutions will be provided then.