STA243 Computational Statistics HW1

Due April 24, 2020

Spring 2020, UC Davis

Ninghui Li (915471290)
e-mail: nhli@ucdavis.edu

Chenghan Sun (915030521)
e-mail: chesun@ucdavis.edu

Pledge:

Please sign below (print full name) after checking (✓) the following. If you can not honestly check each of these responses, please email me at kbala@ucdavis.edu to explain your situation.

- We pledge that we are honest students with academic integrity and we have not cheated on this homework. (✓)
- These answers are our own work. (✓)
- We did not give any other students assistance on this homework. (✓)
- We understand that to submit work that is not our own and pretend that it is our is a violation of the UC Davis code of conduct and will be reported to Student Judicial Affairs. (✓)
- We understand that suspected misconduct on this homework will be reported to the Office of Student Support and Judicial Affairs and, if established, will result in disciplinary sanctions up through Dismissal from the University and a grade penalty up to a grade of "F" for the course. (✓)

Team Member 1: *Ninghui Li* Team Member 2: *Chenghan Sun*

2 Problem 2

We denote $\boldsymbol{A} = [x_1 \quad x_2 \quad \cdots \quad x_n]$, where $x_i \in \mathcal{R}^n$, for $i = 1, 2, ..., n$ and x_i are the column vectors of $\boldsymbol{A}$. We also denote $\boldsymbol{A} = \begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_n^T \end{bmatrix}$, where $a_j \in \mathcal{R}^n$, for $j = 1, 2, ..., n$ and a_j^T are the row vectors of $\boldsymbol{A}$.

We will show that (a), (b) and (c) are equivalent.

(a) The columns of $\boldsymbol{A}$ are orthonormal vectors.

(b) $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix.

(c) The rows of $\boldsymbol{A}$ are orthonormal vectors.

First we show that $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I} \iff \boldsymbol{A}^T = \boldsymbol{A}^{-1} \iff \boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I}$. Thus $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I} \iff$ (b) $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I} \iff \boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I}$.

$$\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I} \iff \boldsymbol{A}^T\boldsymbol{A} - \boldsymbol{I} = 0_{n\times n} \iff (\boldsymbol{A}^T\boldsymbol{A} - \boldsymbol{I})\boldsymbol{A}^{-1} = 0_{n\times n}$$
$$\iff \boldsymbol{A}^T - \boldsymbol{A}^{-1} = 0_{n\times n} \iff \boldsymbol{A}^T = \boldsymbol{A}^{-1}$$

$$\boldsymbol{A}^T = \boldsymbol{A}^{-1} \iff \boldsymbol{A}^T - \boldsymbol{A}^{-1} = 0_{n\times n} \iff \boldsymbol{A}(\boldsymbol{A}^T - \boldsymbol{A}^{-1}) = 0_{n\times n}$$
$$\iff \boldsymbol{A}\boldsymbol{A}^T - \boldsymbol{I} = 0_{n\times n} \iff \boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I}$$

Then we show that (a) $\iff \boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}$. Therefore (a) $\iff$ (b).

(a) The columns of $\boldsymbol{A}$ are orthonormal vectors. It means that $x_i^T x_k = 0$ for $i \neq k$ and $x_i^T x_i = 1$. The (i,k)th entry of $\boldsymbol{A}^T\boldsymbol{A}$ is equal to the inner product of the ith row of $\boldsymbol{A}^T$ and the kth row of $\boldsymbol{A}$. Symbolically, this looks like the following,

$$\boldsymbol{A}^T\boldsymbol{A} = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} = \begin{bmatrix} x_1^T x_1 & x_1^T x_2 & \cdots & x_1^T x_n \\ x_2^T x_1 & x_2^T x_2 & \cdots & x_2^T x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_n^T x_1 & x_n^T x_2 & \cdots & x_n^T x_n \end{bmatrix}$$

Then (a) is equivalent to (b) as follows.

$$x_i^T x_j = 0 \quad \text{for} \quad i \neq j \quad \text{and} \quad x_i^T x_i = 1 \iff \boldsymbol{A}^T\boldsymbol{A} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} = \boldsymbol{I}$$

In the next step, we show that (c) $\iff$ $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I}$. Therefore (c) $\iff$ (b)

The logic is similar to the proof of equivalence between (a) and (b).

(c) The rows of $\boldsymbol{A}$ are orthonormal vectors. It means that $a_j^T a_q = 0$ for $j \neq q$ and $a_j^T a_j = 1$. The (j, q)th entry of $\boldsymbol{A}\boldsymbol{A}^T$ is equal to the inner product of the jth row of $\boldsymbol{A}$ and the qth row of $\boldsymbol{A}^T$. Symbolically, this looks like the following,

$$\boldsymbol{A}\boldsymbol{A}^T = \begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_n^T \end{bmatrix} \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} = \begin{bmatrix} a_1^T a_1 & a_1^T a_2 & \cdots & a_1^T a_n \\ a_2^T a_1 & a_2^T a_2 & \cdots & a_2^T a_n \\ \vdots & \vdots & \ddots & \vdots \\ a_n^T a_1 & a_n^T a_2 & \cdots & a_n^T a_n \end{bmatrix}$$

Then (c) is equivalent to (b) as follows.

$$a_j^T a_q = 0 \quad \text{for} \quad j \neq q \quad \text{and} \quad a_j^T a_j = 1 \iff \boldsymbol{A}\boldsymbol{A}^T = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} = \boldsymbol{I}$$

In a conclusion, (a) $\iff$ (b) $\iff$ (c).

3 Problem 3

How does Equation (2) go to Equation (3) in section (2.1) is as follows, where $\boldsymbol{M} = \sum_{l=1}^{r} \frac{1}{rp_{i_l}} A_{:,i_l} B_{i_l,:}$.

$$
\begin{aligned}
\mathbb{E}[\|\boldsymbol{M} - \boldsymbol{AB}\|_F^2] &= \mathbb{E}[\sum_{i,j}(\boldsymbol{M}_{i,j} - \boldsymbol{A}_{i,:}\boldsymbol{B}_{:,j})^2] = \sum_{i,j} Var[\boldsymbol{M}_{i,j}] \\
&= \sum_{i,j} Var(\sum_{l=1}^{r} \frac{1}{rp_{i_l}} \boldsymbol{A}_{i,i_l}\boldsymbol{B}_{i_l,j}) = \sum_{i,j}\sum_{l=1}^{r} \frac{1}{r^2} Var(\frac{1}{p_{i_l}} \boldsymbol{A}_{i,i_l}\boldsymbol{B}_{i_l,j}) \\
&= \frac{1}{r}\sum_{i,j} Var(\frac{1}{p_{i_l}} \boldsymbol{A}_{i,i_l}\boldsymbol{B}_{i_l,j}) = \frac{1}{r}\sum_{i,j} E\Big[\frac{1}{p_{i_l}} \boldsymbol{A}_{i,i_l}\boldsymbol{B}_{i_l,j} - E[\frac{1}{p_{i_l}} \boldsymbol{A}_{i,i_l}\boldsymbol{B}_{i_l,j}]\Big]^2 \\
&= \frac{1}{r}\sum_{i,j}\sum_{k=1}^{n} p_k(\frac{\boldsymbol{A}_{i,k}\boldsymbol{B}_{k,j}}{p_k} - \sum_{k=1}^{n} p_k \frac{\boldsymbol{A}_{i,k}\boldsymbol{B}_{k,j}}{p_k})^2 \\
&= \frac{1}{r}\sum_{i,j}\sum_{k=1}^{n} p_k\big((\frac{\boldsymbol{A}_{i,k}\boldsymbol{B}_{k,j}}{p_k})^2 - (\sum_{k=1}^{n} \boldsymbol{A}_{i,k}\boldsymbol{B}_{k,j})^2\big) \\
&= \frac{1}{r}\sum_{i,j}\big(\sum_{k=1}^{n} \frac{\boldsymbol{A}_{i,k}^2\boldsymbol{B}_{k,j}^2}{p_k} - (\sum_{k=1}^{n} p_k)(\sum_{k=1}^{n} \boldsymbol{A}_{i,k}\boldsymbol{B}_{k,j})^2\big) \\
&= \frac{1}{r}\sum_{i,j}\big(\sum_{k=1}^{n} \frac{\boldsymbol{A}_{i,k}^2\boldsymbol{B}_{k,j}^2}{p_k} - (\sum_{k=1}^{n} \boldsymbol{A}_{i,k}\boldsymbol{B}_{k,j})^2\big) \\
&= \frac{1}{r}\sum_{k}\sum_{i,j} \frac{\boldsymbol{A}_{i,k}^2\boldsymbol{B}_{k,j}^2}{p_k} - \frac{1}{r}\sum_{i,j}(\boldsymbol{A}_{i,:}\boldsymbol{B}_{:,j})^2 \\
&= \frac{1}{r}\sum_{k} \frac{1}{p_k}\|\boldsymbol{A}_{:,k}\|_2^2\|\boldsymbol{B}_{k,:}\|_2^2 - \frac{1}{r}\|\boldsymbol{AB}\|_F^2
\end{aligned}
$$

4 Randomized matrix multiplication

Part (a): Implement the algorithm

Theory:

Step 1: Calculate probabilities p_k as the preliminary step for random matrix multiplication algorithm.

$$
p_k = \begin{cases} \frac{1}{n}, & \text{if uniformly randomly select the outer products.} \\ \frac{\|A_{:,k}\|_2\|B_{k,:}\|_2}{\sum_l \|A_{:,l}\|_2\|B_{l,:}\|_2}, & \text{if non-uniform sampling} \end{cases}
$$

Step 2: implement the algorithm presented in class for randomized matrix multiplication.

Algorithm 1 Randomized Matrix Multiplication Algorithm

1: **for** $l = 1, ..., r$ **do**

2: Pick $i_l \in$ 1,...,n i.i.d. with probability $\mathcal{P}\{i_l = k\} = p_k$

3: **end for**

4: **return** $M = \sum_{l=1}^{r} \frac{1}{rp_{i_l}} A_{:,i_l} B_{i_l,:}$

Implementation:

The implemented code was provided in the "Problem 4" section of the resource file named "CS_NL_code_hw1.R". Firstly, a helper function "gen_prob" was used to calculate probabilities p_k as the preliminary step for random matrix multiplication algorithm, which covered "Step 1" as mentioned above. Specifically, both uniform and non-uniform sampling methods were implemented, and the function took the two given matrices and a string variable called "gen_flag" as arguments to select methods to generate the importance sampling probabilities; secondly, the main function "randmatmul_alg2" was used to perform the randomized matrix multiplication by accept an argument r (r rank-one components) as well as the matrices and p_k returned from the helper function; Finally as the output of the code, the resulting multiplied matrix of AB (denoted as M) was returned as an unbiased estimator of the true matrix C.

Part (b): Apply the algorithm

Based on the code we developed above, we selected a number of columns r = 20, 50, 100, 200 as required in this part, and calculated four multiplied matrices denoted as "matM_r20", "matM_r50","matM_r100", and "matM_r200". The visualization of these matrices were provided and discussed in Part (d).

Part (c): Calculate the relative approximation error

The relative approximation errors $\|M - AB\|_F / (\|A\|_F \|B\|_F)$ were provided as Table 1 below:

Table 1

The number of columns r	Errors
20	0.18514221
50	0.12469411
100	0.07876438
200	0.05016654

From the table above, it's natural for us to conclude the trend: as the selected number of columns (value of r) increase, the relative approximation error decreases. We expected this result because of the fact that the more information of the original matrices were provided, the more precise the estimation would be calculated. Thus, the errors would also decrease.

Part (d): Visualize the estimates from (b)

The visualization of estimates (denoted as "matM_r20", "matM_r50","matM_r100", and "matM_r200") were shown as Figure 1 below.

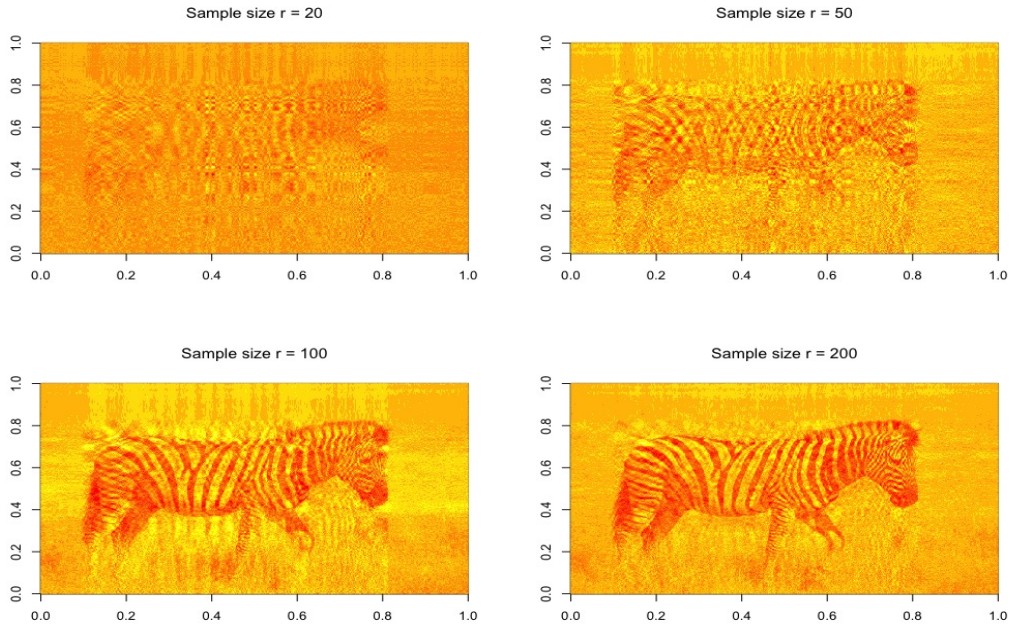

Figure 1

Again from the four images above, we saw the Zebra picture became more and more clear with the increase of r and the decrease of the corresponding relative approximation error.

5 Power method

Theory:

Algorithm 2 Power Method Algorithm

Input Unit-vector $c^{(0)} \in R^d$ (Randomly generated), A, number of iterations T and tolerance ϵ.

1: **for** $t = 1, ..., T$ **do**

2: $c^{(t)} = Ac^{(t-1)}$

3: $c^{(t)} = \frac{c^{(t)}}{\|c^{(t)}\|_2}$ (Normalization Step)

4: $error = 1 - \sum(c^{(t)} \times c^{(t-1)})^2$

5: **if** $error < \epsilon$ **then**

6: break.

7: **end if**

8: **end for**

Implementation:

The implemented code was provided in the "Problem 5" section of the resource file named "CS_NL_code_hw1.R". As required, the task of this problem was to complete the code power_sim.R and run the test routine to produce the plot. The completed function "power_iteration" took two arguments: A (matrix to be took in) and v0 (initial vector, denoted as c0 in the notes) as well as two default parameters eps (tolerance of error) = 1e-6 and maxiter (maximum number of iterations) = 100. With the error between steps (as defined in the Theory section step 4 above), the iterations will stop only if the error tolerance stopping criteria is matched, or the maximum iterations were reached. In fact, by using the given value of λ from 1 to 10, all tests were successfully passed as all their error tolerance conditions were reached.

Results and discussion:

The Figure 2 displayed below shows a range (from 1 to 10) of λ (from $X = \lambda(vv^\top) + E$) versus how well the

estimated eigenvector (using the power method) is correlated (measured via inner-product) with the true eigenvector:

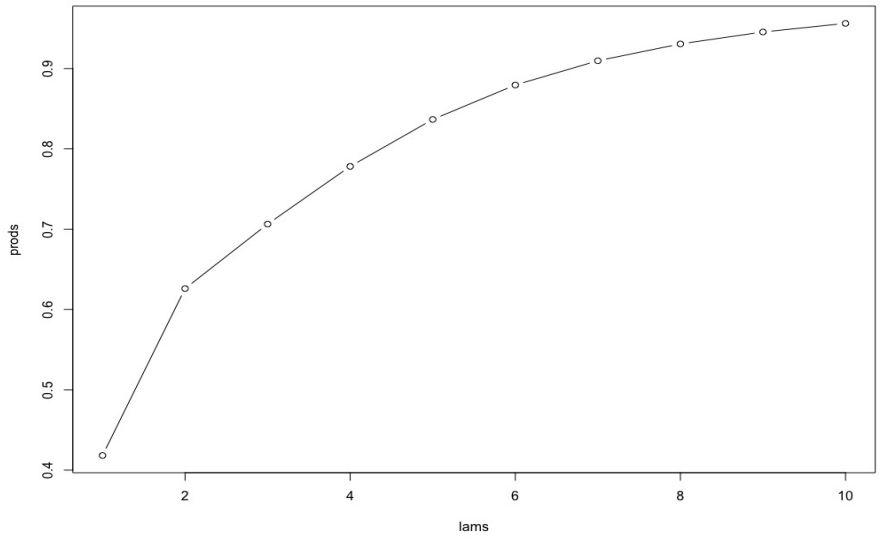

Figure 2

From Figure 2, it's observed that as λ increases, the correlation between v and the estimated eigenvector (aka the largest eigenvector of X) also increases, and approximates to 1. This simulation result was expected, since for power method the source of randomness came from the random matrix E, and as λ increases, the relative weighted importance of E decreases. Given the context that the eigenvector v was non-zero, it was expected that the uncertainty would diminish, or a better estimated eigenvector would be computed.

6 Sketching for Least-squares

Part (a): Implement the Sketched-OLS algorithm

Theory:

Algorithm 3 The Sketched-OLS Algorithm

Input: $\mathbf{X} \in R^{n \times d}$ $\mathbf{y} \in R^n$, and an error parameter $\epsilon \in (0, 1)$. **Output:** $b_s \in R^d$

1: Let $r \approx \frac{dlog(n)}{\epsilon}$.

2: Let $\mathbf{S}$ be an empty matrix.

3: **for** $t = 1, ..., r$ (i.i.d. trials with replacement) **select uniformly at random** an integer from $\{1, 2, ..., n\}$. **do**

4: **If** i is selected, **then** append the column vector $(\sqrt{n/r})\boldsymbol{e}_i$ to $\mathbf{S}$, where $\boldsymbol{e}_i \in R^n$ is the i-th canonical vector.

5: **end for**

6: Let $\mathbf{H} \in R^{n \times n}$ be the normalized Hadamard transform matrix.

7: Let $\mathbf{D} \in R^{n \times n}$ be a diagonal matrix with

$$D_{ii} = \begin{cases} +1, \text{with probability} 1/2 \\ -1, \text{with probability} 1/2 \end{cases}$$

8: Compute and return $\mathbf{b}_s = (\mathbf{S}^T\mathbf{HDX})^+\mathbf{S}^T\mathbf{HDy}$, where for a matrix $\mathbf{A}, \mathbf{A}^+ = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$.

Implementation:

The implemented code was provided in the "Problem 6" section of the resource file named "CS_NL_code_hw1.R". There were three helper functions defined as prerequisites of the main Sketched-OLS algorithm: "helper_r", "helper_sampleS", and "helper_diagD". Specifically, function "helper_r" was used to initialize r, which was introduced as step 1 in Algorithm 3 above ; the second helper function "helper_diagD" was used to generate and apply D as a diagonal matrix, which reflected the step 7 in Algorithm 3. This function took the design matrix X and response vector y as arguments and would return a list of matrix multiplication results of DX and Dy; the last helper function "helper_sampleS" was used to generate and apply the sub-sampling matrix S, which reflected the step 2-5 in Algorithm 3. This function took the results of step 6 (HDX and HDy)

as arguments, and would return a list of matrix multiplication results of $SHDX$ and $SHDy$. The main workflow of Algorithm 3 was implemented as function "main_SketchedOLS" and took x, y, and epsi (error parameter) as arguments. This function served as an ensemble of step 1-7 in Algorithm 3. The last part was a execution function "perform_LS", which was used to solve system of linear equations $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$, which is the last step 8 in Algorithm 3. This function would also be used to find the calculation times as required in Part (c) of this problem.

Part (b): Apply the algorithm

The 1048576 X 20 design matrix X and 1048576 X 1 response y with elements drawn i.i.d. from a Uniform(0, 1) distribution were generated, and denoted as "design_X" and "respon_Y" in the code for Part (c).

Part (c): Compare the calculation time for the full least squares problem and the sketched OLS.

The averaged calculation time (we reported average of 200 calculations to gain stable conclusions) of regular OLS $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ and four sets of sketched OLS $(\Phi\mathbf{X}^T\Phi\mathbf{X})^{-1}\Phi\mathbf{X}^T\Phi\mathbf{y}$ with different epsilons were presented in the table below:

Table 2

	Regular OLS	$\epsilon = 0.1$	$\epsilon = 0.05$	$\epsilon = 0.01$	$\epsilon = 0.001$
Averaged user time	0.51256	0.00143	0.00258	0.01132	0.11855
Averaged system time	0.08793	0.00029	0.00048	0.00012	0.00997
Averaged elapsed time	0.66575	0.00181	0.00320	0.01174	0.13151

From the average elapsed time presented in Table 2 above, it's natural for us to conclude two trends: 1: For sketched OLS calculations, as the error parameter ϵ decreases, the averaged elapsed time increases, which means more computation time was needed; 2: By comparing with the averaged time usage for regular OLS, it's obvious that whatever the value of ϵ we selected, the sketched OLS algorithm always showed significant algorithm efficiency improvements over the regular OLS. We expected these results through step 1 in Algorithm 3. We knew the iteration number $r = \frac{dlog(n)}{\epsilon}$. And as ϵ decreases, the sub-sampling size r increases. Thus, the corresponding calculation time for sketched OLS algorithm increases; Also as we

concluded, the improvements of sketched OLS algorithm over regular OLS was obvious.

Reference

1. RandomizedMatMult.pdf
2. EigenvectorComputation.pdf
3. SketchingLS.pdf