Large Scale Matrix Computation, Optimization and Learning (EE270) HW #2

Winter 2021 Due: **Friday, February 12, 23:59pm**

- After a genuine attempt to solve the homework problems by yourself, you are free to collaborate with your fellow students to find solutions to the homework problems. Regardless of whether you collaborate with other students, you are required to type up or write your own solutions. Copying homework solutions from another student or from existing solutions is a serious violation of the honor code. Please take advantage of the professor's and TA's office hours. We are here to help you learn, and it never hurts to ask!
- **The assignments should be submitted via Gradescope including your code attached as pdf**

1. **Singular Value Decomposition (SVD) for Compression (20 pts)**

   Singular value decomposition (SVD) factorizes a $m \times n$ matrix X as $X = U\Sigma V^\top$, where $U \in \mathbb{R}^{m\times m}$ and $U^\top U = UU^\top = I$, $\Sigma \in \mathbb{R}^{m\times n}$ contains non-increasing non-negative values along its diagonal and zeros elsewhere, and $V \in \mathbb{R}^{n\times n}$ and $V^\top V = VV^\top = I$. Hence matrix can be represented as $X = \sum_{i=1}^{d} \sigma_i u_i v_i^T$ where u_i denotes the i^{th} column of U and v_i denotes the i^{th} column of V. Download the **image**[1] and load it as the matrix X (in greyscale).

   (a) Perform SVD on this matrix and zero out all but top k singular values to form an approximation $\tilde{X}$. Specifically, compute $\tilde{X} = \sum_{i=1}^{k} \sigma_i u_i v_i^T$, display the resulting approximation $\tilde{X}$ as an image, and report $\frac{\|X-\tilde{X}\|_F}{\|X\|_F}$ for $k \in \{2, 10, 40\}$.

   (b) How many numbers do you need to describe the approximation $X = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ for $k \in \{2, 10, 40\}$?

   Hint: Matlab can load the greyscale image using
   `X=double(rgb2gray(imread('harvey-saturday-goes7am.jpg')))`
   and display the image using `imagesc()`

   In Python 2.7 you can use:
   ```
   import numpy, Image
   X=numpy.asarray(Image.open('harvey.jpg').convert('L'))
   ```

[1] https://www.nasa.gov/sites/default/files/thumbnails/image/harvey-saturday-goes7am.jpg
(NOAA's GOES-East satellite's image of Hurricane Harvey in the western Gulf of Mexico on Aug. 26, 2017. Credit: NASA/NOAA GOES Project)

2. **Singular Value Decomposition and Subspaces (20 pts)** Let A be an $m \times n$ singular matrix of rank r with SVD

$$A = U\Sigma V^T = \left(\begin{array}{c|c|c|c} \vec{u}_1 & \vec{u}_2 & \dots & \vec{u}_m \end{array} \right) \begin{pmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_r & & \\ & & & 0 & \\ & & & & \ddots \\ & & & & & 0 \end{pmatrix} \begin{pmatrix} \vec{v}_1^T \\ \hline \vec{v}_2^T \\ \hline \vdots \\ \hline \vec{v}_n^T \end{pmatrix}$$

$$= \begin{pmatrix} \hat{U} & \tilde{U} \end{pmatrix} \begin{pmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_r & & \\ & & & 0 & \\ & & & & \ddots \\ & & & & & 0 \end{pmatrix} \begin{pmatrix} \hat{V}^T \\ \tilde{V}^T \end{pmatrix}$$

where $\sigma_1 \geq \ldots \geq \sigma_r > 0$, $\hat{U}$ consists of the first r columns of U, $\tilde{U}$ consists of the remaining $m - r$ columns of U, $\hat{V}$ consists of the first r columns of V, and $\tilde{V}$ consists of the remaining $n - r$ columns of V. Give bases for the spaces range(A), null(A), range(A^T) and null(A^T) in terms of the components of the SVD of A, and a brief justification.

3. **Least Squares (20 pts)**

   Consider the least squares problem $\min_x ||b - Ax||_2$. Which of the following statements are necessarily true?

   (a) If x is a solution to the least squares problem, then $Ax = b$.

   (b) If x is a solution to the least squares problem, then the residual vector $r = b - Ax$ is in the nullspace of A^T.

   (c) The solution is unique.

   (d) A solution may not exist.

   (e) None of the above.

4. **Binary Classification via Regression (20 pts)**

   Download the MNIST dataset from `http://yann.lecun.com/exdb/mnist/`.

   In binary classification, we restrict Y to take on only two values. Suppose $Y \in 0, 1$. Now let us use least squares linear regression for classification. Let us consider the classification problem of recognizing if a digit is 2 or not using linear regression. Here, let $Y = 1$ for all the 2's digits in the training set, and use $Y = 0$ for all other digits.

   Build a linear classifier by minimizing:

$$\min_w \frac{1}{N} \sum_{i=1}^{N} (y_i - w^T x_i)^2 + \lambda \|w\|_2^2$$

   using the training set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$. Use appropriate regularization ($\lambda > 0$) as needed. Let x_i be a vector of the pixel values of the image.

   For the purpose of classification, we can label a digit as a 2 if $w^T z$ is larger than some threshold.

(a) Based on your training set, choose a reasonable threshold for classification. What is your 0/1 loss on the training set? What is your square loss on the training set?

(b) On the test set, what is the 0/1 loss? What is the square loss?

5. **Leave-One-Out Cross Validation (LOOCV) (20 pts)**

Assume that there are n training examples, $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where each input data point x_i , has d real valued features. The goal of regression is to learn to predict y_i from x_i . The linear regression model assumes that the output y is a linear combination of the input features plus Gaussian noise with weights given by w.

The maximum likelihood estimate of the model parameters w (which also happens to minimize the sum of squared prediction errors) is given by the normal equations we saw in class: $(X^TX)^{-1}X^Ty$. Here, the rows of the matrix X are the training examples $x_1^T, ..., x_n^T$. Define $\hat{Y}$ to be the vector predictions using $\hat{w}$ if we were to plug in the original training set X:

$$\begin{aligned}\hat{y} &= X\hat{w}\\ &= X(X^TX)^{-1}X^Ty\\ &= Hy\end{aligned}$$

where we define $H = X(X^TX)^{-1}X^T$ (H is often called the Hat Matrix). Show that H is a projection matrix.

As mentioned above, $\hat{w}$, also minimizes the sum of squared errors:

$$SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Leave-One-Out Cross Validation score is defined as:

$$LOOCV = \sum_{i=1}^{n}(y_i - \hat{y}_i^{(-i)})^2$$

where $\hat{y}^{(-i)}$ is the estimator of y after removing the i^{th} observation from the training set, i.e., it minimizes

$$\sum_{j \neq i}(y_j - \hat{y}_j^{(-i)})^2.$$

(a) What is the time complexity of computing the LOOCV score naively? (The naive algorithm is to loop through each point, performing a regression on the $n-1$ remaining points at each iteration.)
Hint: The complexity of matrix inversion using classical methods is $O(k^3)$ for a $k \times k$ matrix.

(b) Write $\hat{y}_i$ in terms of H and y.

(c) Show that $\hat{y}^{(-i)}$ is also the estimator which minimizes SSE for Z where

$$Z_j = \begin{cases} y_j, \; j \neq i \\ \hat{y}_i^{(-i)}, \; j = i \end{cases}$$

(d) Write $\hat{y}_i^{(-i)}$ in terms of H and Z. By definition, $\hat{y}_i^{(-i)} = Z_i$, but give an answer that is analogous to (b).

(e) Show that $\hat{y}_i - \hat{y}_i^{(-i)} = H_{ii}y_i - H_{ii}\hat{y}_i^{(-i)}$, where H_{ii} denotes the i^{th} element along the diagonal of H.

(f) Show that

$$LOOCV = \sum_{i=1}^{n} \left(\frac{y_i - \hat{y}_i}{1 - H_{ii}} \right)^2$$

What is the algorithmic complexity of computing the LOOCV score using this formula? **Note:** We see from this formula that the diagonal elements of H somehow indicate the impact that each particular observation has on the result of the regression.

6. **Sketching Least Squares (20 pts)** Here, we will consider the empirical performance of random sampling and random projection algorithms for approximating least-squares.

Let A be an $n \times d$ matrix, with $n \gg d$, b be an n-vector, and consider approximating the solution to $\min_x \|Ax - b\|_2$. Generate the matrices A from one of three different classes of distributions introduced below

- Generate a matrix A from multivariate normal $N(1_d, \Sigma)$, where the (i, j)th element of $\Sigma_{ij} = 2 \times 0.5^{|i-j|}$.(Refer to as GA data.)
- Generate a matrix A from multivariate t-distribution with 3 degree of freedom and covariance matrix Σ as before. (Refer to as T3 data.)
- Generate a matrix A from multivariate t-distribution with 1 degree of freedom and covariance matrix Σ as before. (Refer to as T1 data.)

To start, consider matrices of size $n \times d$ equal to 500×50.

(a) First, for each matrix, consider approximating the solution by randomly sampling a small number r of rows/elements (i.e., constraints of the overconstrained least-squares problem) in one of three ways: uniformly at random; according to an importance sampling distribution that is proportional to the Euclidean norms squared of the rows of A; and according to an importance sampling distribution that is proportional to the leverage scores of A. In each case, plot the error as a function of the number r of samples, paying particular attention to two regimes: $r = d, d+1, d+2, \ldots, 2d$; and $r = 2d, 3d, \ldots$. Show that the behavior of these three procedures is most similar for GA data, intermediate for T3 data, and most different for T1 data; and explain why, and explain similarities and differences.

(b) Next, for each matrix, consider approximating the solution by randomly projecting rows/elements (i.e., constraints of the overconstrained least-squares problem) in one of two ways: a random projection matrix in which each entry is i.i.d. $\{\pm 1\}$, with appropriate variance; and a random projection matrix, in which each entry is i.i.d. Gaussian, with appropriate variance. In each case, plot the error as a function of the number of samples, i.e., dimensions on which the data are projected, paying particular attention to two regimes: $r = d, d+1, d+2, \ldots, 2d$; and $r = 2d, 3d, \ldots$. Describe and explain similarities and differences between these two procedures for GA data, T1 data, and T3 data.

(c) Finally, for each matrix, consider approximating the solution by randomly projecting rows/elements (i.e., constraints of the overconstrained least-squares problem) with sparse projection matrices. In particular, consider a random projection matrix, in which each entry is i.i.d. either 0 or Gaussian, where the probability of 0 is q and the probability of Gaussian is $1 - q$. (Remember to rescale the variance of the Gaussian appropriately, depending on q) For q varying from 0 to 1, in increments sufficiently small, plot the error for solving the least-squares problem. Describe and explain how this varies as a function of the number of samples, i.e., dimensions on which the data are projected, paying particular attention to two regimes: $r = d, d+1, d+2, \ldots, 2d$; and $r = 2d, 3d, \ldots$. Describe and explain similarities and differences between these three procedures for GA data, T1 data, and T3 data.

Next, we describe how these behave for larger problems. To do so, we will work with dense random projection algorithms in which the projection matrix consists of i.i.d. $\{\pm 1\}$ random variables, with appropriate variance. Fix a value of d, and let n increase from roughly $2d$ to roughly $100d$. The exact value of d and n will depend on your machine, your computational environment, etc., so that you get reasonably good low-precision approximate solutions to the original least-squares problem. (You should expect $d \approx 500$ should work; and if you can't do the full plot to $100d$, don't worry, since the point is to get large enough to illustrate the phenomena below.)

(d) Plot the running time of the random projection algorithm versus the running time of solving the problem with a call to a QR decomposition routine provided by your system as well as the running time of solving the problem with a call to an SVD routine provided by your system. Illustrate that, for smaller problems the random projection methods are not faster, but that for larger problems, the random projection methods are slightly faster and/or can be used to solver larger problems than QR or SVD. Explain why is this the case.