Homework 1

Steven Xu

Due @ 11:59pm on September 13, 2018

Part 1.

1. Let $\mathbf{Q} \in \mathbb{R}^{m \times m}$ be a rotation matrix, namely $\mathbf{Q}^\mathsf{T}\mathbf{Q} = \mathbf{I}$. **True or False:** The 1-norm of a vector $\mathbf{x} \in \mathbb{R}^m$ is rotationally invariant, namely

$$\|\mathbf{Qx}\|_1 = \|\mathbf{x}\|_1.$$

If true, give a proof. If false, provide a counter example.

Answer:

I will show that this is false by giving a counter example. Let

$$\mathbf{Q} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}, \mathbf{x} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

We can quickly check that $\mathbf{Q}^\mathsf{T}\mathbf{Q} = \mathbf{I}$ and $\|\mathbf{x}\|_1$=2. Then we have

$$\mathbf{Qx} = \begin{pmatrix} \sqrt{2} \\ 0 \end{pmatrix}$$

But $\|\mathbf{Qx}\|_1 = \sqrt{2} \neq 2$, therefore the 1-norm is not rotaionally invariant.

2. Let $\mathbf{Q} \in \mathbb{R}^{m \times m}$ be a rotation matrix, namely $\mathbf{Q}^\mathsf{T}\mathbf{Q} = \mathbf{I}$. **True or False:** The 2-norm of a vector $\mathbf{x} \in \mathbb{R}^m$ is rotationally invariant, namely

$$\|\mathbf{Qx}\|_2 = \|\mathbf{x}\|_2.$$

If true, give a proof. If false, provide a counter example.

Answer:

For notaion convenience I will prove that $\|\mathbf{Q}^\mathsf{T}\mathbf{x}\|_2 = \|\mathbf{x}\|_2$. Note that $\mathbf{Q}$ is a rotation matrix if and only if $\mathbf{Q}^\mathsf{T}$ is a rotation matrix. Let

$$\mathbf{Q} = \begin{pmatrix} \mathbf{q_1} & \mathbf{q_2} & \mathbf{q_3} & \dots & \mathbf{q_m} \end{pmatrix}, \mathbf{Q}^\mathsf{T} = \begin{pmatrix} \mathbf{q_1}^\mathsf{T} \\ \mathbf{q_2}^\mathsf{T} \\ \mathbf{q_3}^\mathsf{T} \\ . \\ . \\ . \\ \mathbf{q_m}^\mathsf{T} \end{pmatrix}$$

Then

$$\mathbf{Q}^\mathsf{T}\mathbf{x} = \begin{pmatrix} \mathbf{q_1}^\mathsf{T}\mathbf{x} \\ \mathbf{q_2}^\mathsf{T}\mathbf{x} \\ \mathbf{q_3}^\mathsf{T}\mathbf{x} \\ . \\ . \\ . \\ \mathbf{q_m}^\mathsf{T}\mathbf{x} \end{pmatrix}$$

Then

$$\|\mathbf{Q}^\mathsf{T}\mathbf{x}\|_2 = \sum_{i=1}^{m}(\mathbf{q_i}^\mathsf{T}\mathbf{x})^2 = \sum_{i=1}^{m}(\mathbf{q_i}^\mathsf{T}\mathbf{x})^\mathsf{T}(\mathbf{q_i}^\mathsf{T}\mathbf{x}) = \sum_{i=1}^{m}\mathbf{x}^\mathsf{T}\mathbf{q_i}\mathbf{q_i}^\mathsf{T}\mathbf{x}$$

We know that

$$\mathbf{Q}\mathbf{Q}^\mathsf{T} = \mathbf{I} \Rightarrow \sum_{i=1}^{m}\mathbf{q_i}\mathbf{q_i}^\mathsf{T} = \mathbf{I}$$

$$\therefore \sum_{i=1}^{m}\mathbf{x}^\mathsf{T}\mathbf{q_i}\mathbf{q_i}^\mathsf{T}\mathbf{x} = \mathbf{x}^\mathsf{T}\sum_{i=1}^{m}\mathbf{q_i}\mathbf{q_i}^\mathsf{T}\mathbf{x} = \mathbf{x}^\mathsf{T}\mathbf{I}\mathbf{x} = \sum_{i=1}^{m}x_i^2 = \|\mathbf{x}\|_2$$

Thus the 2-norm is rotationally invariant.

3. Let $\mathbf{Q} \in \mathbb{R}^{m\times m}$ be a rotation matrix, namely $\mathbf{Q}^\mathsf{T}\mathbf{Q} = \mathbf{I}$. **True or False:** The ∞-norm of a vector $\mathbf{x} \in \mathbb{R}^m$ is rotationally invariant, namely

$$\|\mathbf{Q}\mathbf{x}\|_\infty = \|\mathbf{x}\|_\infty.$$

If true, give a proof. If false, provide a counter example.

Answer:

This is false and I will again use the matrices in Question 1 as a counter example.

$$\mathbf{Q} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}, \mathbf{x} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \mathbf{Q}\mathbf{x} = \begin{pmatrix} \sqrt{2} \\ 0 \end{pmatrix}$$

$$\Rightarrow \|\mathbf{Q}\mathbf{x}\|_\infty = \sqrt{2} \neq 1 = \|\mathbf{x}\|_\infty$$

Thus the $\infty-$norm is not rotationally invariant.

Part 2. The Power Method

You will next add an implementation of the power method to your R package. Recall that the power method can be used to determine a matrix $\mathbf{A}$'s eigenvector associated with its largest eigenvalue. This iterative algorithm repeatedly does two computations: (i) a matrix-vector product and (ii) a normalization. Suppose $\mathbf{x}^{(k)}$ is the value of our guess of the eigenvector after k iterations. Then we compute $\mathbf{x}^{(k+1)}$ from $\mathbf{x}^{(k)}$ as follows.

$$\mathbf{x}^{(k+1)} \leftarrow \frac{\mathbf{A}\mathbf{x}^{(k)}}{\|\mathbf{A}\mathbf{x}^{(k)}\|_2}$$

Note that you need to start with a **non-zero** vector $\mathbf{x}^{(0)}$.

Some examples of the power method in statistics:

- Allen, Grosenick, and Taylor, "A Generalized Least Squares Matrix Decomposition," Journal of the American Statistical Association, Theory & Methods, 109:505, 145-159, 2014.
- Sun, Lu, Liu, and Cheng, "Provable sparse tensor decomposition," Journal of the Royal Statistical Society, Series B, 889:916, 2017

Please complete the following steps.

Step 0: Make an R package entitled "your_unityidST758". My unity id is "ecchi", so I would make a package entitled ecchiST758. For the following functions, save them all in a file called `homework1.R` and put this file in the R subdirectory of your package.

Step 1: Write a function "power_method_dense" that applies the power method to a dense matrix. Terminate the procedure when the relative change in the estimate of the eigenvector falls below a specified tolerance `tol`, i.e. stop iterating when

$$\|\mathbf{x} - \mathbf{x}_{\text{last}}\|_2 < \text{tol} \times \|\mathbf{x}_{\text{last}}\|_2.$$

```r
#' Power Method for dense matrices
#'
#' \code{power_method_dense} applies the power method to estimate
#' the eigenvector of a dense matrix associated with its largest eigenvector.
#'
#' @param A The input matrix
#' @param x initial guess
#' @param max_iter maximum number of iterations
#' @param tol relative error stopping criterion
#' @export
# power_method_dense <- function(A, x, max_iter, tol) {
#
# }
```

Your function should return a vector estimating the eigenvector of the input matrix which corresponds to its largest eigenvalue. This vector should have unit Euclidean length.

Step 2: Write a function "power_method_sparse" applies the power method to a sparse matrix. Use the `Matrix` package; add it to the list of dependencies in the DESCRIPTION file. It should return an error message if

- the input matrix is not sparse.

```
#' Power Method for sparse matrices
#'
#' \code{power_method_sparse} applies the power method to estimate
#' the eigenvector of a sparse matrix associated with its largest eigenvector.
#'
#' @param A The input matrix
#' @param x initial guess
#' @param max_iter maximum number of iterations
#' @param tol relative error stopping criterion
#' @export
# power_method_sparse <- function(A, x, max_iter, tol) {
#
# }
```

Your function should return a vector estimating the eigenvector of the input matrix which corresponds to its largest eigenvalue. This vector should have unit Euclidean length.

Step 3: Write a function "power_method_low_rank" applies the power method to a matrix $\mathbf{A} = \mathbf{U}\mathbf{V}^\mathsf{T}$. It should return an error message if

- the two factor matrices $\mathbf{U}$ and $\mathbf{V}$ do not have the same number of columns.

```
#' Power Method for low rank matrices
#'
#' \code{power_method_low_rank} applies the power method to estimate
#' the eigenvector of a low rank matrix associated with its largest eigenvector.
#'
#' @param U The left input factor matrix
#' @param V The right input factor matrix
#' @param x initial guess
#' @param max_iter maximum number of iterations
#' @param tol relative error stopping criterion
#' @export
# power_method_low_rank <- function(U, V, x, max_iter, tol) {
#
# }
```

Your function should return a vector estimating the eigenvector of the input matrix which corresponds to its largest eigenvalue. This vector should have unit Euclidean length.

Step 4: Write a function "power_method_sparse_plus_low_rank" applies the power method to a matrix $\mathbf{A} = \mathbf{S} + \mathbf{U}\mathbf{V}^\mathsf{T}$ that is the sum of a sparse matix $\mathbf{S}$ and low rank matrix $\mathbf{U}\mathbf{V}^\mathsf{T}$. It should return an error message if

- the input matrix $\mathbf{S}$ is not sparse.
- the two factor matrices $\mathbf{U}$ and $\mathbf{V}$ do not have the same number of columns.

```
#' Power Method for sparse + low rank matrices
#'
#' \code{power_method_sparse_plus_low_rank} applies the power method to estimate
#' the eigenvector of a sparse + low rank matrix associated with its largest eigenvector.
#'
#' @param S sparse input matrix term
#' @param U The left input factor matrix term
#' @param V The right input factor matrix term
#' @param x initial guess
#' @param max_iter maximum number of iterations
#' @param tol relative error stopping criterion
#' @export
# power_method_sparse_plus_low_rank <- function(S, U, V, x, max_iter, tol) {
#
# }
```

Your function should return a vector estimating the eigenvector of the input matrix which corresponds to its largest eigenvalue. This vector should have unit Euclidean length.

Step 5: Use your `power_method_dense` function to compute the largest eigenvalue of the following matrices. Compare your answers to what is provided by the `eigen` function in R. Choose `max_iter` and `tol` so that your solution matches `eigen`'s output up to 3 decimal places.

```
library(sgxuST758)
library(Matrix)
library(knitr)
```

```
set.seed(12345)
n <- 1e3
A <- matrix(rnorm(n**2), n, n)
A <- A + t(A)
x = matrix(rnorm(n),nrow=n)
eg = as.matrix(power_method_dense(A,x,1e5,1e-4))
my_ev_a=round(as.numeric((t(eg)%*%A%*%eg)/(t(eg)%*%eg)),4)
eig = eigen(A)$values
r_ev_a = round(eig[which.max(abs(eig))],4)

rm(A)
B <- matrix(rnorm(n**2), n, n)
B <- B + t(B)
x = matrix(rnorm(n),nrow=n)
eg = as.matrix(power_method_dense(B,x,1e5,1e-4))
my_ev_b=round(as.numeric((t(eg)%*%B%*%eg)/(t(eg)%*%eg)),4)
eig = eigen(B)$values
r_ev_b = round(eig[which.max(abs(eig))],4)

rm(B)
C <- matrix(rnorm(n**2), n, n)
C <- C + t(C)
x = matrix(rnorm(n),nrow=n)
eg = as.matrix(power_method_dense(C,x,1e5,1e-4))
my_ev_c=round(as.numeric((t(eg)%*%C%*%eg)/(t(eg)%*%eg)),4)
eig = eigen(C)$values
r_ev_c = round(eig[which.max(abs(eig))],4)
```

Matrix	Power's value	**Eigen**'s value
A	89.2708	89.2708
B	88.6193	88.6193
C	-89.06	-89.06

Step 6: Use your `power_method_sparse` function to compute the largest eigenvalue of the following matrices. Compare your answers to what is provided by the `eigen` function in R. Choose `max_iter` and `tol` so that your solution matches `eigen`'s output up to 3 decimal places. **Hint:** You may need to convert a sparse matrix into a dense one before passing to `eigen`.

```
rm(list=ls())
set.seed(12345)
n <- 1e3
nnz <- 0.1*n
ix <- sample(1:n, size = nnz, replace = FALSE)
A <- Matrix(0, nrow=n, ncol=n, sparse=TRUE)
A[ix] <- rnorm(nnz)
A <- A + t(A)
```

```r
A[1,1] <- 10
x = matrix(rnorm(n),nrow=n)
eg = power_method_sparse(A,x,1e6,1e-3)
eg = as.matrix(eg)
A = as.matrix(A)
sp_ev_a=round(as.numeric(t(eg)%*%A%*%eg)/(t(eg)%*%eg),4)
eig = eigen(A)$values
r_sp_ev_a = round(eig[which.max(abs(eig))],4)

ix <- sample(1:n, size = nnz, replace = FALSE)
B <- Matrix(0, nrow=n, ncol=n, sparse=TRUE)
B[ix] <- rnorm(nnz)
B <- B + t(B)
B[1,1] <- 10
x = matrix(rnorm(n),nrow=n)
eg = power_method_sparse(B,x,1e6,1e-3)
eg = as.matrix(eg)
B = as.matrix(B)
sp_ev_b=round(as.numeric(t(eg)%*%B%*%eg)/(t(eg)%*%eg),4)
eig = eigen(B)$values
r_sp_ev_b = round(eig[which.max(abs(eig))],4)

ix <- sample(1:n, size = nnz, replace = FALSE)
C <- Matrix(0, nrow=n, ncol=n, sparse=TRUE)
C[ix] <- rnorm(nnz)
C <- C + t(C)
C[1,1] <- 10
x = matrix(rnorm(n),nrow=n)
eg = power_method_sparse(C,x,1e6,1e-3)
eg = as.matrix(eg)
C = as.matrix(C)
sp_ev_c=round(as.numeric(t(eg)%*%C%*%eg)/(t(eg)%*%eg),4)
eig = eigen(C)$values
r_sp_ev_c = round(eig[which.max(abs(eig))],4)
```

Matrix	Power's value	**Eigen**'s value
A	17.4712	17.4712
B	16.1044	16.1044
C	15.6734	15.6734

Step 7: Use your `power_method_low_rank` function to compute the largest eigenvalue of the matrices $\mathbf{U}_1\mathbf{V}_1^\mathsf{T}$, $\mathbf{U}_2\mathbf{V}_2^\mathsf{T}$, and $\mathbf{U}_3\mathbf{V}_3^\mathsf{T}$ defined below. Compare your answers to what is provided by the `eigen` function in R. Choose `max_iter` and `tol` so that your solution matches `eigen`'s output up to 3 decimal places.

```
rm(list=ls())
set.seed(12345)
n <- 1e3
k <- 10
U1 <- V1 <- matrix(rnorm(n*k), n, k)
x = matrix(rnorm(n),nrow=n)
eg = power_method_low_rank(U1,V1,x,1e5,1e-4)
A = U1%*%t(V1)
my_ev_u1=round(as.numeric((t(eg)%*%A%*%eg)/(t(eg)%*%eg)),4)
eig = eigen(A)$values
r_ev_u1 = round(eig[which.max(abs(eig))],4)


U2 <- V2 <- matrix(rnorm(n*k), n, k)
x = matrix(rnorm(n),nrow=n)
eg = power_method_low_rank(U2,V2,x,1e5,1e-4)
A = U2%*%t(V2)
my_ev_u2=round(as.numeric((t(eg)%*%A%*%eg)/(t(eg)%*%eg)),4)
eig = eigen(A)$values
r_ev_u2 = round(eig[which.max(abs(eig))],4)

U3 <- V3 <- matrix(rnorm(n*k), n, k)
x = matrix(rnorm(n),nrow=n)
eg = power_method_low_rank(U3,V3,x,1e5,1e-4)
A = U3%*%t(V3)
my_ev_u3=round(as.numeric((t(eg)%*%A%*%eg)/(t(eg)%*%eg)),4)
eig = eigen(A)$values
r_ev_u3 = round(eig[which.max(abs(eig))],4)
```

Matrices	Power's value	**Eigen**'s value
U_1, V_1	1161.9546	1161.9548
U_2, V_2	1154.6356	1154.6358
U_3, V_3	1164.129	1164.1292

Step 7: Estimate the largest eigenvalue of the following matrix.

$$\mathbf{A} = \mathbf{S} + \mathbf{u}\mathbf{u}^\mathsf{T}.$$

```
rm(list=ls())
set.seed(12345)
n <- 1e7
nnz <- 1e-5*n
ix <- sample(1:n, size = nnz, replace = FALSE)
S <- Matrix(0, nrow=n, ncol=n, sparse=TRUE)
S[ix] <- rnorm(nnz)
S <- S + t(S)
```

```r
u <- matrix(rnorm(n), ncol=1)
x = matrix(rnorm(n),ncol=1)
eg = power_method_sparse_plus_low_rank(S,u,u,x,1e5,1e-4)
ans=(t(eg)%*%S%*%eg+(t(eg)%*%u)%*%(t(u)%*%eg))/(t(eg)%*%eg)
my_ev_est = round(as.numeric(ans))
my_ev_est
```

```
## [1] 10004786
```

The largest eigenvalue calculated using power method is 10004786 .

What happens if you attempt to compute the largest eigenvalue using `eigen`? If you try using `power_method_dense`?

Answer: `eigen` will not work because the computer would not have the memory to store a 10^7 by 10^7 dense matrix. Same for `power_method_dense` since uu^T is not applicable.

An example of sparse + low-rank matrices in statistics: Matrix Completion

- Hastie, Mazumder, Lee, and Zadeh, "Matrix Completion and Low-Rank SVD via Fast Alternating Least Squares," Journal of Machine Learning Research, 3367-3402, 2015