データ分析・データ処理A(多次元データ処理)

HI4 45号 山口惺司

実験日2024/05/15

2024/05/22

2024/05/29

# 実験目的

重要なデータ構造の1つである高階テンソルの概要が理解でき、Rによる高階テンソルの処理に関するプログラミング演習を通して、テンソルデータの生成及び主要なテンソルデータ処理ができるようになる。

# 課題A

1. 課題1

[Ex.1-1]

rTensorのインストール

1. 課題2

[Ex.2-1]

Rを使って3階テンソルを生成してみよう.

スクリプト：

library(rTensor)

A <- array(0, dim = c(3, 3, 3))

A[1, 1, 1] <- A[2, 2, 1] <- A[3, 3, 1] <- 1

A[2, 1, 2] <- A[3, 2, 2] <- A[1, 3, 2] <- 1

A[3, 1, 3] <- A[1, 2, 3] <- A[2, 3, 3] <- 1

A <- as.tensor((A))

実行結果：

3階テンソルを生成することができた.

説明：

それぞれの行列の要素に1を代入している,

[Ex.2-2]

生成した3階テンソルの中身を見てみよう.

スクリプト：

A

実行結果：

実行結果を図1に示す.

カレンダー

自動的に生成された説明

図1 Ex2-1実行結果

説明：

Ex2-1で作成した3階テンソルを出力している.

[Ex.2-3]

Rを使って各方向から見た透視図を作ってみよう.

スクリプト：

A\_1sum <- modeSum(A, 1, drop=TRUE)

A\_2sum <- modeSum(A, 2, drop=TRUE)

A\_3sum <- modeSum(A, 3, drop=TRUE)

実行結果：

各方向から見た透視図を作成できた。

説明：

A\_1sumは上から下方向を見た透視図, A\_2sumは左から右方向を見た透視図, A\_2sumは前から後方向を見た透視図である.

[Ex.2-4]

作成された透視図(行列)を見てみよう.

スクリプト：

A1\_sum

A2\_sum

A3\_sum

実行結果：

実行結果を図2に示す.

概略図

中程度の精度で自動的に生成された説明概略図

低い精度で自動的に生成された説明 概略図

中程度の精度で自動的に生成された説明

図2 Ex2-3実行結果

説明：

Ex2-3で作成した透視図を出力している.

1. 課題3

[Ex.3-1]

電球がセットされている位置のマップを1-モード行列展開で作ってみよう.

スクリプト：

A1 <- unfold(A, row\_idx = 1, col\_idx = c(3, 2))

A1

実行結果：

実行結果を図3に示す.

テキスト

中程度の精度で自動的に生成された説明

図3 Ex3-1実行結果

説明：

1-モード行列展開によりAを展開している.

[Ex.3-2]

電球がセットされている位置のマップを2-モード行列展開で作ってみよう.

スクリプト：

A2 <- unfold(A, row\_idx = 2, col\_idx = c(1, 3))

A2

実行結果：

実行結果を図4に示す.

グラフ が含まれている画像

自動的に生成された説明

図4 Ex3-2実行結果

説明：

2-モード行列展開によりAを展開している.

[Ex.3-3]

電球がセットされている位置のマップを3-モード行列展開で作ってみよう.

スクリプト：

A3 <- unfold(A, row\_idx = 3, col\_idx = c(2, 1))

A3

実行結果：

実行結果を図4に示す.

テキスト

自動的に生成された説明

図5 Ex3-3実行結果

説明：

3-モード行列展開によりAを展開している.

1. 課題4

[Ex.4-1]

1-モード行列展開で作られたマップを元の立方体に戻してみよう

スクリプト：

A\_1org <- fold(A1, row\_idx = 1, col\_idx = c(3, 2), modes = c(3, 3, 3))

A\_1org

実行結果：

実行結果を図6に示す.

テキスト が含まれている画像

自動的に生成された説明

図6 Ex4-1実行結果

説明：

Ex.3-1で作成したマップを元の立体に戻している.

[Ex.4-2]

2-モード行列展開で作られたマップを元の立方体に戻してみよう

スクリプト：

A\_2org <- fold(A1, row\_idx = 2, col\_idx = c(1, 3), modes = c(3, 3, 3))

A\_2org

実行結果：

実行結果を図7に示す.

カレンダー

自動的に生成された説明

図7 Ex4-2実行結果

説明：

Ex.3-2で作成したマップを元の立体に戻している.

[Ex.4-3]

3-モード行列展開で作られたマップを元の立方体に戻してみよう

スクリプト：

A\_3org <- fold(A1, row\_idx = 3, col\_idx = c(2, 1), modes = c(3, 3, 3))

print(A\_3org)

実行結果：

実行結果を図8に示す.

ダイアグラム, 概略図

中程度の精度で自動的に生成された説明

図8 Ex4-3実行結果

説明：

Ex.3-3で作成したマップを元の立体に戻している.

1. 課題5

[基本問題A]

1〜24の値を持つ3階テンソル に対して，それぞれ3つの ⾏列の1〜3-モード積を計算した結果をrTensorパッケージ を利⽤して求めなさい．

スクリプト：

library(rTensor)

A <- array(1:24, dim=c(2, 3, 4))

A <- as.tensor(A)

(A@data)

U1 <- array(1:4, dim=c(2, 2))

U2 <- array(1:9, dim=c(3, 3))

U3 <- array(1:16, dim=c(4, 4))

ansA1 <- ttm(A, U1, m=1)

ansA2 <- ttm(A, U2, m=2)

ansA3 <- ttm(A, U3, m=3)

ansA1

ansA2

ansA3

実行結果：

実行結果を図9に示す.

テキスト

自動的に生成された説明テキスト

自動的に生成された説明 テキスト

自動的に生成された説明

図9 Ex.5-基本課題A実行結果

説明：

ttm()関数を用いて、3階テンソルAに対して、行列U1~U3のモード積を計算している.

[応用課題A]

例題の3階テンソルと⾏列を⽤いて簡易的な求め⽅のn-モード積を計算するスクリプト

を作成してください．

スクリプト：

library(rTensor)

A <- array(1:24, dim=c(2, 3, 4))

A <- as.tensor(A)

(A@data)

U1 <- array(1:4, dim=c(2, 2))

U2 <- array(1:9, dim=c(3, 3))

U3 <- array(1:16, dim=c(4, 4))

A1 <- unfold(A, row\_idx=1, col\_idx=c(2, 3))

A2 <- unfold(A, row\_idx=2, col\_idx=c(1, 3))

A3 <- unfold(A, row\_idx=3, col\_idx=c(1, 2))

idx1 <- U1 %\*% A1@data

idx2 <- U2 %\*% A2@data

idx3 <- U3 %\*% A3@data

myA1 <- fold(idx1, row\_idx=1, col\_idx=c(2, 3), modes=c(2, 3, 4))

myA2 <- fold(idx2, row\_idx=2, col\_idx=c(1, 3), modes=c(2, 3, 4))

myA3 <- fold(idx3, row\_idx=3, col\_idx=c(1, 2), modes=c(2, 3, 4))

myA1

myA2

myA3

実行結果：

実行結果を図10に示す.

テキスト, カレンダー

自動的に生成された説明 テキスト

自動的に生成された説明 テキスト, 手紙

自動的に生成された説明

図10 Ex5-応用課題A実行結果

説明：

[応用課題B]

マクマホンキューブと⾏列との1〜3-モード積を求めるスクリプトを作成してください． ただし，ttm関数を⽤いた場合と簡易的な求め⽅での場合の2種類を実装してください．

スクリプト：

library(rTensor)

A <- array(0, dim=c(3, 3, 3))

A <- as.tensor(A)

(A@data)

A[2, 2, 1] <- 3

A[1, 2, 2] <- 1

A[2, 3, 2] <- 2

A[2, 1, 2] <- 4

A[3, 2, 2] <- 6

A[2, 2, 3] <- 5

U <- array(0, dim=c(3, 3))

U[3, 1] <- U[2, 2] <- U[1, 3] <- 1

ansA1 <- ttm(A, U, m=1)

ansA2 <- ttm(A, U, m=2)

ansA3 <- ttm(A, U, m=3)

ansA1

ansA2

ansA3

A1 <- unfold(A, row\_idx=1, col\_idx=c(2, 3))

A2 <- unfold(A, row\_idx=2, col\_idx=c(1, 3))

A3 <- unfold(A, row\_idx=3, col\_idx=c(1, 2))

idx1 <- U %\*% A1@data

idx2 <- U %\*% A2@data

idx3 <- U %\*% A3@data

myA1 <- fold(idx1, row\_idx=1, col\_idx=c(2, 3), modes=c(3, 3, 3))

myA2 <- fold(idx2, row\_idx=2, col\_idx=c(1, 3), modes=c(3, 3, 3))

myA3 <- fold(idx3, row\_idx=3, col\_idx=c(1, 2), modes=c(3, 3, 3))

myA1

myA2

myA3

実行結果：

ttm()関数を用いた場合の実行結果を図11、簡易的な求め方での場合の実行結果を図12に示す.

テキスト

自動的に生成された説明テキスト

自動的に生成された説明 テキスト

自動的に生成された説明

図11 Ex5-応用課題B-ttm()関数を用いた場合の実行結果

テキスト

自動的に生成された説明 テキスト, 手紙

自動的に生成された説明 テキスト, 手紙

自動的に生成された説明

図12 Ex5-応用課題B-簡易的な求め方での場合の実行結果

説明：

ttm()関数を使用してマクマホンキューブと⾏列との1〜3-モード積を求めている。

簡易的な求め方ではunfold()関数でテンソルを展開し、積を計算した後にfold()関数でテンソルを畳み込んでいる。

1. 課題6

[Ex6-1]

関数eigenの使用例

スクリプト：

x <- c(1, 0, 0, 0, 3, -1, 0, -1, 3)

A <- array(x, dim=c(3, 3))

z <- eigen(A)

lambda <- z$values

T <- z$vectors

lambda

T

実行結果：

実行結果を図13に示す.

テキスト

自動的に生成された説明

図13 Ex6-1実行結果

説明：

eigen()関数を使用して、固有値と固有ベクトルの計算を行い、Aの固有値の組とAの固有行列を出力している。

[Ex6-2]

関数svdの使用例

スクリプト：

x <- c(1, 1, 2, 2, 2, -1, 0, -1)

B <- array(x, dim=c(4, 2))

z <- svd(B)

sigma <- z$d

U <- z$u

V <- z$v

sigma

V

実行結果：

実行結果を図14に示す.

テキスト

自動的に生成された説明

図14 Ex6-2実行結果

説明：

svd()関数でBの特異値と特異行列を計算し、Bの右特異行列と左特異行列を出力している。

[Ex6-3]

[Ex.6-1]で得られた lamda, T を利用して、スライドp.4の式の右辺の計算を実装し、左辺の元データAに戻ることを確認せよ。

スクリプト：

A1 <- T %\*% diag(lambda) %\*% t(T)

A1

実行結果：

実行結果を図15に示す.

時計, オレンジ, 部屋, 大きい が含まれている画像

自動的に生成された説明

図15 Ex6-3実行結果

説明：

A=TΛTTの右辺の計算をして元データの行列を求めている。

[Ex6-4]

[Ex.6-2]で得られた sigma, U, V を利用して、スライドp.9の式の右辺の 計算を実装し、左辺の元データBに戻ることを確認せよ。

スクリプト：

B1 <- U %\*% diag(sigma) %\*% t(V)

B1

実行結果：

実行結果を図16に示す.

テキスト

自動的に生成された説明

図16 Ex6-4実行結果

説明：

B=UΣVTの右辺を計算して元データの行列を求めている.

1. 課題7

[Ex7-1]

R成分画像データSVD(512個の特異値に分解したとき)

スクリプト：

library(imager)

img <- load.image("parrots.png")

plot(img)

A <- t(img[ , , 1, 1])

plot(as.cimg(t(A)))

svdA <- svd(A)

sigma <- svdA$d

U <- svdA$u

V <- svdA$v

recA <- U %\*% diag(sigma) %\*% t(V)

plot(as.cimg(t(recA)))

実行結果：

実行結果を図17～19に示す.

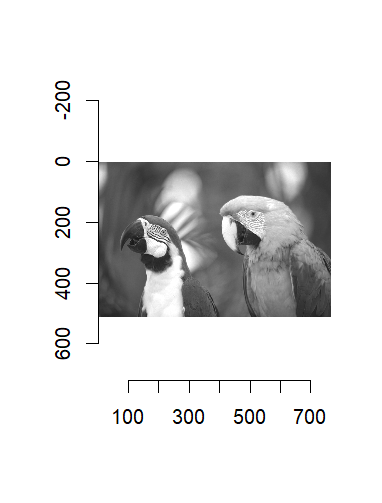
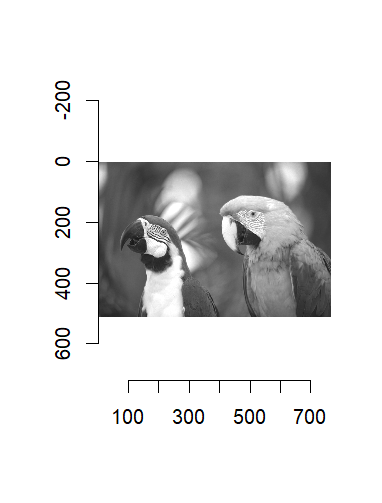
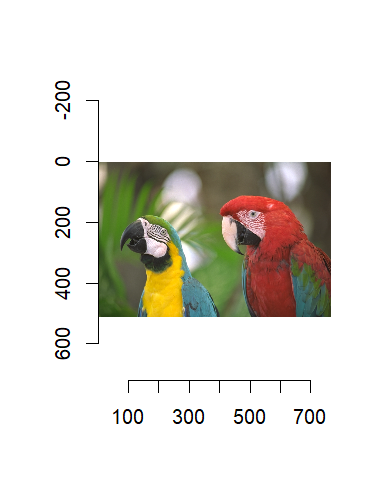


図17 元画像 図18 R成分画像 図19 再構成画像

説明：

特異値512個の時の元画像R成分画像、再構成画像を出力

[Ex7-2]

R成分画像データのSVD(10個の特異値に分解したとき)

スクリプト：

num\_sv <- 10

svdA2 <- svd(A, nu=num\_sv, nv=num\_sv)

sigma2 <- svdA2$d

U2 <- svdA2$u

V2 <- svdA2$v

recA2 <- U2 %\*% diag(sigma2[1:num\_sv]) %\*% t(V2)

plot(as.cimg(t(recA2)))

library(rTensor)

w <- width(img); h <- height(img); MAX <- 1.0

MSE\_R <- fnorm(as.tensor(A-recA2))^2 / (h\*w)

PSNR\_R <- 10\*log10(MAX^2/MSE\_R)

n <- h; p <- w; r <- 10; c\_raito <- 1-(n\*r+r\*r+p\*r) / (n\*p)

実行結果：

実行結果を図20、図21に示す。

テキスト

自動的に生成された説明

図20 計算したR成分画像のPSNRとSVDの圧縮率

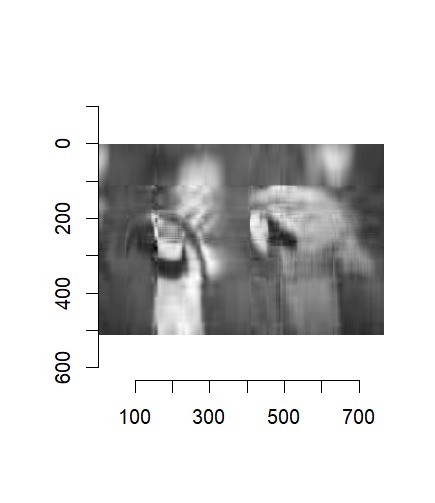


図21 R成分画像データのSVD(10個の特異値に分解)

説明：

分解する特異値の数を10個にしてSVDを出力。

[Ex7-3]

各自、RG画像を収集し、そのG成分の画像SVDを計算してください。なお、PSNRの値が約25dB, 30dB, 40dBの3つについて、分解する特異値の数を調整して、計算してください.

スクリプト：

library(imager)

library(rTensor)

img <- load.image("mogusi2.png")

plot(img)

A <- t(img[ , , 1, 2])

num\_sv <- 65

#num\_sv <- 130

#num\_sv <- 240

svdA <- svd(A, nu=num\_sv, nv=num\_sv)

sigma <- svdA$d

U <- svdA$u

V <- svdA$v

recA <- U %\*% diag(sigma[1:num\_sv]) %\*% t(V)

plot(as.cimg(t(recA2)))

w <- width(img); h <- height(img); MAX <- 1.0

MSE\_G <- fnorm(as.tensor(A-recA))^2 / (h\*w)

PSNR\_G <- 10\*log10(MAX^2/MSE\_G)

n <- h; p <- w; r <- num\_sv; c\_raito <- 1-(n\*r+r\*r+p\*r) / (n\*p)

num\_sv

PSNR\_G

c\_raito

実行結果：

実行結果を図22,23に示す。

テキスト

自動的に生成された説明テキスト

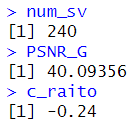
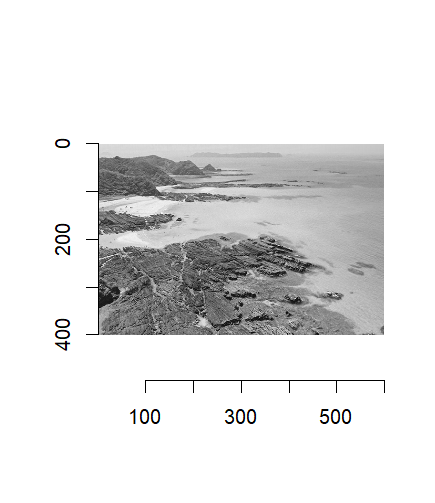
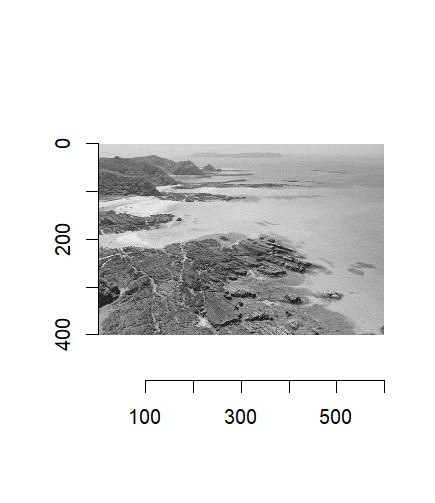
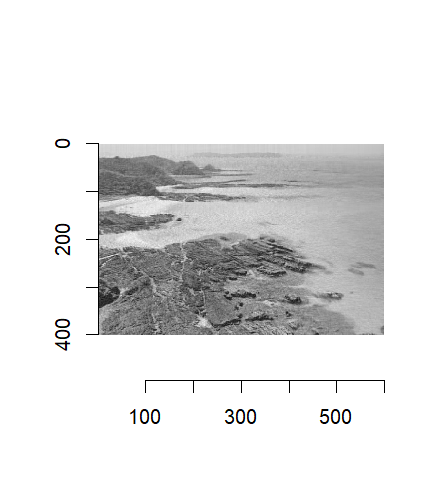
自動的に生成された説明

図22 分解する特異値、G成分のPSNR値、SVDの圧縮率

図23 PSNR値20dB, 25dB, 40dBで出力した画像

説明：

G成分のPSNR値がそれぞれ25dB,30dB,40dBに近づくように分解する特異値の数を調整したらそれぞれ65個, 130個, 240個となった。

# 課題B

大きさ4x4x4の立方体において、以下の各要素の組み合わせで電球をセットして点灯されたとき、立方体の3方向から見てすべてが点灯されるものを選択してください。ただし、Rスクリプトを作成し、実行して調べて回答してください。

[1, 6, 12, 15, 20, 23, 26, 29, 34, 37, 43, 48, 51, 56, 67, 62]

[1, 8, 11, 14, 20, 23, 26, 29, 35, 38, 41, 48, 50, 53, 60, 63]

[2, 5, 12, 15, 17, 22, 27, 32, 36, 39, 42, 45, 51, 56, 57, 62]

[1, 7, 10, 16, 19, 24, 25, 30, 34, 37, 44, 47, 52, 54, 59, 61]

スクリプト：

library(rTensor)

A1 <- array(0, dim = c(4, 4, 4))

A2 <- array(0, dim = c(4, 4, 4))

A3 <- array(0, dim = c(4, 4, 4))

A4 <- array(0, dim = c(4, 4, 4))

x1 <- c(2,5,12,15,17,22,27,32,36,39,42,45,51,56,57,62)

x2 <- c(1,6,12,15,20,23,26,29,34,37,43,48,51,56,57,62)

x3 <- c(1,7,10,16,19,24,25,30,34,37,44,47,52,54,59,61)

x4 <- c(1,8,11,14,20,23,26,29,35,38,41,48,50,53,60,63)

for(i in 1:4){

for(j in 1:4){

for(k in 1:4){

sum = k + (j-1)\*4 + (i-1)\*16

if(sum %in% x1){

A1[k, j, i] <- 1

}

if(sum %in% x2){

A2[k, j, i] <- 1

}

if(sum %in% x3){

A3[k, j, i] <- 1

}

if(sum %in% x4){

A4[k, j, i] <- 1

}

}

}

}

A1 <- as.tensor((A1))

A2 <- as.tensor((A2))

A3 <- as.tensor((A3))

A4 <- as.tensor((A4))

print("A1:")

print(modeSum(A1, 1, drop=TRUE))

print(modeSum(A1, 2, drop=TRUE))

print(modeSum(A1, 3, drop=TRUE))

print("A2:")

print(modeSum(A2, 1, drop=TRUE))

print(modeSum(A2, 2, drop=TRUE))

print(modeSum(A2, 3, drop=TRUE))

print("A3:")

print(modeSum(A3, 1, drop=TRUE))

print(modeSum(A3, 2, drop=TRUE))

print(modeSum(A3, 3, drop=TRUE))

print("A4:")

print(modeSum(A4, 1, drop=TRUE))

print(modeSum(A4, 2, drop=TRUE))

print(modeSum(A4, 3, drop=TRUE))

実行結果：

実行結果を図24に示す。

概略図

低い精度で自動的に生成された説明テキスト

自動的に生成された説明概略図 が含まれている画像

自動的に生成された説明テキスト

中程度の精度で自動的に生成された説明

図24 各立方体を各方向から見たときの値

説明：

3重のfor文を用いて与えられた配列を4x4x4テンソルに格納して、各方向から見た透視図を作成した。

出力された透視図を見ると、全て1となっているため、大きさ4x4x4の立方体において、各要素の組み合わせで電球をセットして点灯されたとき、すべての立方体が3方向から見てすべてが点灯される。

# 感想

今までいろんなプログラミング言語を学んできたが、多次元配列を用いる時はRがとても便利だということを実感できた。

課題Bではfor文を用いて自動的に4x4x4のテンソルに値を入れることができてよかった。