Question2

Christina Wei and Paul Mandelos

2024-03-08

1.

The Rank-1 approximation of an image using SVG captures the most significant patterns of the image. In the case of our moon, since the edges of the moon are completely dark and the center is brighter, the rank-1 approximation will reflect that. The rank-1 approximation will be a gradient of white/black with the center of the image (where the moon used to be) being brighter and the edges of the being black. The rank-1 approximation will show most prominent features of the moon's shape, but without any details.

2.

```
library(magick)
```

```
## Warning: package 'magick' was built under R version 4.2.3
```

```
## Linking to ImageMagick 6.9.12.93
## Enabled features: cairo, fontconfig, freetype, heic, lcms, pango, raw, rsvg, webp
## Disabled features: fftw, ghostscript, x11
```

```
# Load the image
image <- image_read("/Users/paulmandelos/Downloads/p5_image.gif")

# Convert the image to grayscale
image_gray <- image_convert(image, colorspace = 'gray')
image_raster <- as.raster(image_gray)
image_matrix <- matrix(as.numeric(image_raster))
```

```
## Warning in matrix(as.numeric(image_raster)): NAs introduced by coercion
```

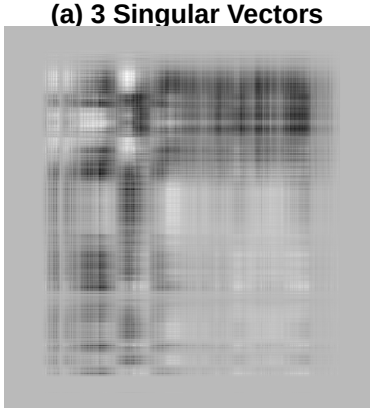
```
image_matrix <- as.integer(image_data(image_gray))[,,1]

binary_matrix <- 1 - (image_matrix > 0.5)
svd1<-svd(binary_matrix)
str(svd1)
```

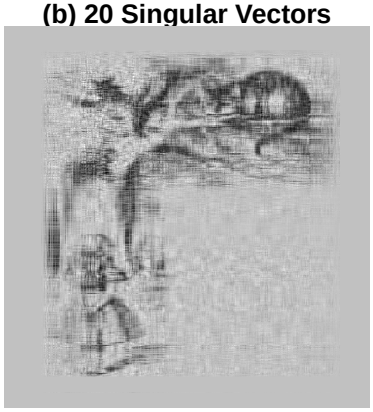
```
## List of 3
##  $ d: num [1:1170] 261.1 135 86.3 74.3 66.1 ...
##  $ u: num [1:1600, 1:1170] 0 0 0 0 0 0 0 0 0 0 ...
##  $ v: num [1:1170, 1:1170] 0 0 0 0 0 0 0 0 0 0 ...
```

```
#matrix multiplication of U,D and V*
#Also scale back to 0 and 1
#including only first singluar vector
D1 <- matrix(svd1$d[1:1], nrow = 1, ncol = 1)
approx1<-svd1$u[,1:1] %*% D1 %*% t(svd1$v[,1:1])
approx1 <- (approx1 - min(approx1)) / (max(approx1) - min(approx1))
#including only first 3 singluar vectors
approx3<-svd1$u[,1:3] %*% diag(svd1$d[1:3]) %*% t(svd1$v[,1:3])
approx3 <- (approx3 - min(approx3)) / (max(approx3) - min(approx3))
#including only first 10 singular vectors
approx10<-svd1$u[,1:10] %*% diag(svd1$d[1:10]) %*% t(svd1$v[,1:10])
approx10 <- (approx10 - min(approx10)) / (max(approx10) - min(approx10))
#including only first 20 singular vectors
approx20<-svd1$u[,1:20] %*% diag(svd1$d[1:20]) %*% t(svd1$v[,1:20])
approx20 <- (approx20 - min(approx20)) / (max(approx20) - min(approx20))
#including only first 50 singular vectors
approx50<-svd1$u[,1:50] %*% diag(svd1$d[1:50]) %*% t(svd1$v[,1:50])
approx50 <- (approx50 - min(approx50)) / (max(approx50) - min(approx50))
#including only first 100 singular vectors
approx100<-svd1$u[,1:100] %*% diag(svd1$d[1:100]) %*% t(svd1$v[,1:100])
approx100 <- (approx100 - min(approx100)) / (max(approx100) - min(approx100))
#including only first 150 singular vectors
approx150<-svd1$u[,1:150] %*% diag(svd1$d[1:150]) %*% t(svd1$v[,1:150])
approx150 <- (approx150 - min(approx150)) / (max(approx150) - min(approx150))
#including only first 200 singular vectors
approx200<-svd1$u[,1:200] %*% diag(svd1$d[1:200]) %*% t(svd1$v[,1:200])
approx200 <- (approx200 - min(approx200)) / (max(approx200) - min(approx200))
#including only first 400 singular vectors
approx400<-svd1$u[,1:400] %*% diag(svd1$d[1:400]) %*% t(svd1$v[,1:400])
approx400 <- (approx400 - min(approx400)) / (max(approx400) - min(approx400))
#including only first 800 singular vectors
approx800<-svd1$u[,1:800] %*% diag(svd1$d[1:800]) %*% t(svd1$v[,1:800])
approx800 <- (approx800 - min(approx800)) / (max(approx800) - min(approx800))
#including all singular vectors
full<-svd1$u[,1:1170] %*% diag(svd1$d[1:1170]) %*% t(svd1$v[,1:1170])
full <- (full - min(full)) / (max(full) - min(full))
par(mfrow=c(2,3),mar=c(1,1,1,1))
#plotting for reduced images
image(1:ncol(approx3), 1:nrow(approx3), t(approx3[nrow(approx3):1,]), main="(a) 3 Singular Vectors", ax
image(1:ncol(approx20), 1:nrow(approx20), t(approx20[nrow(approx20):1,]), main="(b) 20 Singular Vectors
image(1:ncol(approx150), 1:nrow(approx150), t(approx150[nrow(approx150):1,]), main="(c) 150 Singular Ve
image(1:ncol(full), 1:nrow(full), t(full[nrow(full):1,]), main="(d) Full Image", axes=FALSE, col=gray(2
```

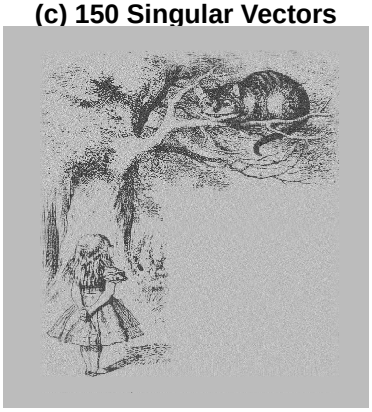
(a) 3 Singular Vectors



(b) 20 Singular Vectors



(c) 150 Singular Vectors



(d) Full Image

3.

We stop at 1170 because that's the number of singular values you can have at most for this image. This number comes from the smallest dimension of the image, which is its width in this case.

4.

To store the rank 150 approximation, you need to store the matrices U, Sigma, and VˆT with only the first 150 singular values/vectors. If the original image matrix is m×n: U is m×150 Sigma is 150×150 VˆT is 150×n The total memory required is the sum of the elements in these matrices: m×150+150×150+150×n=150m+22500+150n units. In our case, m = 1600, n = 1170 so 150x1600+22500+150x1170 = 438000 Full image: 1600x1170 = 1872000 We save roughly 73% of space.

5.

The presence of the gray haze in the image, even at relatively high ranks (like k = 800), is due to the first few singular values capturing the most significant features of the image, which include major structures and shapes. However, the finer details and subtle variations are captured by the latter singular values. Even though they are smaller and contribute less to the image's overall structure, they define the fine details and eliminate background noise. Since the truncated SVD focuses on capturing the most significant components of the image first, the less prominent details, which clarify the image and remove haze, only emerge as you include more singular values in the approximation. In this particular instance, the persistence of this haze indicates that the image's noise is distributed across many of the lesser singular values. Thus removing the

haze requires capturing all/most of these subtle details, which are only fully represented as you approach the full rank of the matrix.