Matrix methods – QR factorisation (2) & SVD (1)

MATH 2740 – Mathematics of Data Science – Lecture 08

Julien Arino
julien.arino@umanitoba.ca

Department of Mathematics @ University of Manitoba

Fall 202X

The University of Manitoba campuses are located on original lands of Anishinaabeg, Ininew, Anisininew, Dakota and Dene peoples, and on the National Homeland of the Red River Métis. We respect the Treaties that were made on these territories, we acknowledge the harms and mistakes of the past, and we dedicate ourselves to move forward in partnership with Indigenous communities in a spirit of Reconciliation and collaboration.

Outline

The QR factorisation & Least squares

Singular values

The SVD

An application of the SVD – Image compression

The QR factorisation & Least squares

Singular values

The SVD

An application of the SVD – Image compression

The QR factorisation

Theorem 70

Let $A \in \mathcal{M}_{mn}$ with LI columns. Then A can be factored as

$$A = QR$$

where $Q \in \mathcal{M}_{mn}$ has orthonormal columns and $R \in \mathcal{M}_n$ is nonsingular upper triangular

Back to least squares

So what was the point of all that..?

Theorem 71 (Least squares with QR factorisation)

$A \in \mathcal{M}_{mn}$ with LI columns, $\boldsymbol{b} \in \mathbb{R}^m$. If $A = QR$ is a QR factorisation of A, then the unique least squares solution $\tilde{\boldsymbol{x}}$ of $A\boldsymbol{x} = \boldsymbol{b}$ is

$$\tilde{\boldsymbol{x}} = R^{-1}Q^T\boldsymbol{b}$$

Proof of Theorem 71

A has LI columns so

- least squares $A\boldsymbol{x} = \boldsymbol{b}$ has unique solution $\tilde{\boldsymbol{x}} = (A^T A)^{-1} A^T \boldsymbol{b}$
- by Theorem 70, A can be written as $A = QR$ with $Q \in \mathcal{M}_{mn}$ with orthonormal columns and $R \in \mathcal{M}_n$ nonsingular and upper triangular

So

$$
\begin{aligned}
A^T A\tilde{\boldsymbol{x}} = A^T \boldsymbol{b} &\implies (QR)^T QR\tilde{\boldsymbol{x}} = (QR)^T \boldsymbol{b} \\
&\implies R^T Q^T QR\tilde{\boldsymbol{x}} = R^T Q^T \boldsymbol{b} \\
&\implies R^T \mathbb{I}_n R\tilde{\boldsymbol{x}} = R^T Q^T \boldsymbol{b} \\
&\implies R^T R\tilde{\boldsymbol{x}} = R^T Q^T \boldsymbol{b} \\
&\implies (R^T)^{-1} R\tilde{\boldsymbol{x}} = (R^T)^{-1} R^T Q^T \boldsymbol{b} \\
&\implies R\tilde{\boldsymbol{x}} = Q^T \boldsymbol{b} \\
&\implies \tilde{\boldsymbol{x}} = R^{-1} Q^T \boldsymbol{b} \quad \square
\end{aligned}
$$

The QR factorisation & Least squares

Singular values

The SVD

An application of the SVD – Image compression

Matrix factorisations (continued)

The singular value decomposition (known mostly by its acronym, SVD) is yet another type of factorisation/decomposition..

Singular values

Definition 72 (Singular value)

Let $A \in \mathcal{M}_{mn}(\mathbb{R})$. The **singular values** of A are the real numbers

$$\sigma_1 \geq \sigma_2 \geq \cdots \sigma_n \geq 0$$

that are the square roots of the eigenvalues of $A^T A$

Singular values are real and nonnegative?

Recall that $\forall A \in \mathcal{M}_{mn}$, $A^T A$ is symmetric

Claim 1. Real symmetric matrices have real eigenvalues

Proof. $A \in \mathcal{M}_n(\mathbb{R})$ symmetric and $(\lambda, \boldsymbol{v})$ eigenpair of A, i.e, $A\boldsymbol{v} = \lambda \boldsymbol{v}$. Taking the complex conjugate, $\overline{A\boldsymbol{v}} = \overline{\lambda \boldsymbol{v}}$

Since $A \in \mathcal{M}_n(\mathbb{R})$, $\overline{A} = A \qquad (z = \bar{z} \iff z \in \mathbb{R})$

So

$$A\bar{\boldsymbol{v}} = \overline{A}\bar{\boldsymbol{v}} = \overline{A\boldsymbol{v}} = \overline{\lambda \boldsymbol{v}} = \bar{\lambda}\bar{\boldsymbol{v}}$$

i.e., if $(\lambda, \boldsymbol{v})$ eigenpair, $(\bar{\lambda}, \bar{\boldsymbol{v}})$ also eigenpair

Still assuming $A \in \mathcal{M}_n(\mathbb{R})$ symmetric and $(\lambda, \boldsymbol{v})$ eigenpair of A and using what we just proved (that $(\bar{\lambda}, \bar{\boldsymbol{v}})$ also eigenpair), take transposes

$$\begin{aligned} A\bar{\boldsymbol{v}} = \bar{\lambda}\bar{\boldsymbol{v}} &\iff (A\bar{\boldsymbol{v}})^T = (\bar{\lambda}\bar{\boldsymbol{v}})^T \\ &\iff \bar{\boldsymbol{v}}^T A^T = \bar{\lambda}\bar{\boldsymbol{v}}^T \\ &\iff \bar{\boldsymbol{v}}^T A = \bar{\lambda}\bar{\boldsymbol{v}}^T \qquad [A \text{ symmetric}] \end{aligned}$$

Let us now compute $\lambda(\bar{\boldsymbol{v}} \bullet \boldsymbol{v})$. We have

$$\begin{aligned} \lambda(\bar{\boldsymbol{v}} \bullet \boldsymbol{v}) &= \lambda \bar{\boldsymbol{v}}^T \boldsymbol{v} = \bar{\boldsymbol{v}}^T(\lambda \boldsymbol{v}) \\ &= \bar{\boldsymbol{v}}^T(A\boldsymbol{v}) = (\bar{\boldsymbol{v}}^T A)\boldsymbol{v} \\ &= (\bar{\lambda}\bar{\boldsymbol{v}}^T)\boldsymbol{v} = \bar{\lambda}(\bar{\boldsymbol{v}} \bullet \boldsymbol{v}) \\ &\iff (\lambda - \bar{\lambda})(\bar{\boldsymbol{v}} \bullet \boldsymbol{v}) = 0 \end{aligned}$$

We have shown

$$(\lambda - \bar{\lambda})(\bar{\boldsymbol{v}} \bullet \boldsymbol{v}) = 0$$

Let

$$\boldsymbol{v} = \begin{pmatrix} a_1 + ib_1 \\ \vdots \\ a_n + ib_n \end{pmatrix}$$

Then

$$\bar{\boldsymbol{v}} = \begin{pmatrix} a_1 - ib_1 \\ \vdots \\ a_n - ib_n \end{pmatrix}$$

So

$$\bar{\boldsymbol{v}} \bullet \boldsymbol{v} = (a_1^2 + b_1^2) + \cdots + (a_n^2 + b_n^2)$$

But $\boldsymbol{v}$ eigenvector is $\neq \boldsymbol{0}$, so $\bar{\boldsymbol{v}} \bullet \boldsymbol{v} \neq 0$, so

$$(\lambda - \bar{\lambda})(\bar{\boldsymbol{v}} \bullet \boldsymbol{v}) = 0 \iff \lambda - \bar{\lambda} = 0 \iff \lambda = \bar{\lambda} \iff \lambda \in \mathbb{R} \quad \square$$

Claim 2. For $A \in \mathcal{M}_{mn}(\mathbb{R})$, the eigenvalues of A^TA are real and nonnegative

Proof. We know that for $A \in \mathcal{M}_{mn}$, A^TA symmetric and from previous claim, if $A \in \mathcal{M}_{mn}(\mathbb{R})$, then A^TA is symmetric and real and with real eigenvalues

Let $(\lambda, \boldsymbol{v})$ be an eigenpair of A^TA, with $\boldsymbol{v}$ chosen so that $\|\boldsymbol{v}\| = 1$

Norms are functions $V \to \mathbb{R}_+$, so $\|A\boldsymbol{v}\|$ and $\|A\boldsymbol{v}\|^2$ are ≥ 0 and thus

$$\begin{aligned} 0 \leq \|A\boldsymbol{v}\|^2 &= (A\boldsymbol{v}) \bullet (A\boldsymbol{v}) = (A\boldsymbol{v})^T(A\boldsymbol{v}) \\ &= \boldsymbol{v}^TA^TA\boldsymbol{v} = \boldsymbol{v}^T(A^TA\boldsymbol{v}) = \boldsymbol{v}^T(\lambda\boldsymbol{v}) \\ &= \lambda(\boldsymbol{v}^T\boldsymbol{v}) = \lambda(\boldsymbol{v} \bullet \boldsymbol{v}) = \lambda\|\boldsymbol{v}\|^2 \\ &= \lambda \end{aligned}$$

Claim 3. For $A \in \mathcal{M}_{mn}(\mathbb{R})$, the nonzero eigenvalues of A^TA and AA^T are the same

Proof. Let $(\lambda, \boldsymbol{v})$ be an eigenpair of A^TA with $\lambda \neq 0$. Then $\boldsymbol{v} \neq \mathbf{0}$ and

$$A^TA\boldsymbol{v} = \lambda\boldsymbol{v} \neq \mathbf{0}$$

Left multiply by A

$$AA^TA\boldsymbol{v} = \lambda A\boldsymbol{v}$$

Let $\boldsymbol{w} = A\boldsymbol{v}$, we thus have $AA^T\boldsymbol{w} = \lambda\boldsymbol{w}$; in other words, $A\boldsymbol{v}$ is an eigenvector of AA^T corresponding to the (nonzero) eigenvalue λ

The reverse works the same way.. □

The QR factorisation & Least squares

Singular values

The SVD

An application of the SVD – Image compression

The singular value decomposition (SVD)

Theorem 73 (SVD)

$A \in \mathcal{M}_{mn}$ with singular values $\sigma_1 \geq \cdots \geq \sigma_r > 0$ and $\sigma_{r+1} = \cdots = \sigma_n = 0$

Then there exists $U \in \mathcal{M}_m$ orthogonal, $V \in \mathcal{M}_n$ orthogonal and a block matrix $\Sigma \in \mathcal{M}_{mn}$ taking the form

$$\Sigma = \begin{pmatrix} D & 0_{r,n-r} \\ 0_{m-r,r} & 0_{m-r,n-r} \end{pmatrix}$$

where

$$D = \operatorname{diag}(\sigma_1, \ldots, \sigma_r) \in \mathcal{M}_r$$

such that

$$A = U\Sigma V^T$$

Definition 74

We call a factorisation as in Theorem 73 the **singular value decomposition** of A. The columns of U and V are, respectively, the **left** and **right singular vectors** of A

U and V^T are *rotation* or *reflection* matrices, Σ is a *scaling* matrix

$U \in \mathcal{M}_m$ orthogonal matrix with columns the eigenvectors of AA^T

$V \in \mathcal{M}_n$ orthogonal matrix with columns the eigenvectors of A^TA

Outer product form of the SVD

Theorem 75 (Outer product form of the SVD)

$A \in \mathcal{M}_{mn}$ with singular values $\sigma_1 \geq \cdots \geq \sigma_r > 0$ and $\sigma_{r+1} = \cdots = \sigma_n = 0$, $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r$ and $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r$, respectively, left and right singular vectors of A corresponding to these singular values

Then

$$A = \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T + \cdots + \sigma_r \boldsymbol{u}_r \boldsymbol{v}_r^T \tag{1}$$

The QR factorisation & Least squares

Singular values

The SVD

An application of the SVD – Image compression

Applications of the SVD

Many applications of the SVD, both theoretical and practical..

1. Obtaining a unique solutions to least squares when $A^T A$ singular
2. Image compression

Compressing images

Consider an image (for simplicity, assume in shades of grey). This can be stored in a matrix $A \in \mathcal{M}_{mn}$

Take the SVD of A. Then the small singular values carry information about the regions with little variation and can perhaps be omitted, whereas the large singular values carry information about more “dynamic” regions of the image

Suppose A has r nonzero singular values. For $k \leq r$, let

$$A_k = \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T + \cdots + \sigma_k \boldsymbol{u}_k \boldsymbol{v}_k^T$$

For $k = r$ we get the usual outer product form (1)

Load the image using `bmp::read.bmp`

```
my_image = bmp::read.bmp("../CODE/Julien_and_friend_1000x800.bmp")
my_image_g = pixmap::pixmapGrey(my_image)
my_image_g
```

```
## Pixmap image
##   Type          : pixmapGrey
##   Size          : 800x1000
##   Resolution    : 1x1
##   Bounding box  : 0 0 1000 800
```

Doing the computations “by hand”

```
M = my_image_g@grey
MTM = t(M) %*% M
# Ensure matrix is symmetric
MTM = (MTM+t(MTM))/2
ev = eigen(MTM)
```

Given the size and nature of the entries, the matrix $M^T M$ is symmetric only to `1e-5` precision, so we use a little trick to make it symmetric no matter what: take the average of $M^T M$ and its transpose MM^T

Which version of the algorithm to use?

Make zero the eigenvalues that are close to zero (200 out of 1000)

```
ev$values = ev$values*(ev$values>1e-10)
```

Can we use the algorithm for all eigenvalues being distinct or do we have repeated ones?

```
any(duplicated(ev$values[ev$values>1e-10]))

## [1] FALSE
```

So we can use the standard algorithm

Computing the SVD

```
idx_positive_ev = which(ev$values>1e-10)
sv = sqrt(ev$values[idx_positive_ev])
```

Computing the SVD

Then $D = \text{diag}(\sigma_1, \ldots, \sigma_r)$, V is the matrix of normalised eigenvectors in the same order as the σ_i and for $i = 1, \ldots, r$

$$\mathbf{u}_i = \frac{1}{\sigma_i} A\mathbf{v}_i$$

ensuring that $\|\mathbf{u}_i\| = 1$

```
D = diag(sv)
V = ev$vectors[idx_positive_ev, idx_positive_ev]
c1 = colSums(V)
for (i in 1:dim(V)[2]) {
    V[,i] = V[,i]/c1[i]
}
```

Computing the SVD

Finally, we compute the $\boldsymbol{u}_i$'s

```
U = M %*% V %*% diag(1/sv)

## Error in M %*% V: non-conformable arguments

r = length(sv)
im = list(u=U, d=sv, v=V)

## Error:  object 'U' not found
```

Using built-in functions

We can also use the built-in function `svd` to compute the SVD of M

```
M.svd = svd(M)
```

The results are stored in a list with components `u`, `d` and `v`

Make function to recreate an image from the SVD

Given the SVD `im` of an image and a number of singular values to keep `n`, we can recreate the image using the function `compress_image`

We output the new image, but also, the amount of information required to encode this new image, as a percentage of the original image size

```
compress_image = function(im, n) {
  if (n > length(im$d)) {
    # Check that we gave a value of n within range, otherwise
    # just set to the max
    n = length(im$d)
  }
  d_tmp = im$d[1:n]
  u_tmp = im$u[,1:n]
  v_tmp = im$v[,1:n]
  # We store the results in a list (so we can return other information)
    out = list()
    # First, compute the resulting image
    out$img = mat.or.vec(nr = dim(im$u)[1], nc = dim(im$v)[1])
    for (i in 1:n) {
        out$img = out$img + d_tmp[i] * u_tmp[,i] %*% t(v_tmp[,i])
    }
```

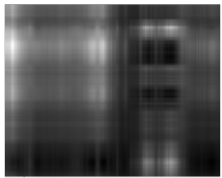
```
    # Values of the "colours" must be between 0 and 1, so we shift and resc
    if (min(min(out$img)) < 0 ) {
        out$img = out$img - min(min(out$img))
    }
    out$img = out$img / max(max(out$img))
    # Store some information: number of points needed and percentage of the
    out$nb_pixels_original = dim(im$u)[1] * dim(im$v)[2]
    out$nb_pixels_compressed = length(d_tmp) + dim(u_tmp)[1]*dim(u_tmp)[2] +
    out$pct_of_original = out$nb_pixels_compressed / out$nb_pixels_original
    # Return the result
    return(out)
}
```

Recreating the image

We can now recreate the image using the function `compress_image`

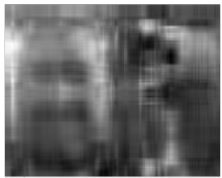
```
new_image = my_image_g
M.svd = svd(M)
M_tmp = compress_image(M.svd, 2)
new_image@grey = M_tmp$img
plot(new_image)
```

Using $n = 2$ singular values



Uses 0.56% of the original information

Using $n = 5$ singular values



Uses 1.41% of the original information

Using $n = 10$ singular values

Uses 2.81% of the original information

Using $n = 20$ singular values

Uses 5.63% of the original information

Using $n = 50$ singular values

Uses 14.07% of the original information