UPPSALA
UNIVERSITET

Department of Information Technology

Scientific Computing for Data Analysis

Davoud Mirzaei

Lecture 6: Least squares problem and QR factorization

Agenda

- Normal equations (continued)
- Nonlinear models
- QR factorization

Mathematical justification: least squares to normal equations?

Least squares problem: Given a full rank matrix $A \in \mathbb{R}^{m\times n}$ for $m \geq n$ and a vector $\boldsymbol{b} \in \mathbb{R}^m$, find $\boldsymbol{x} \in \mathbb{R}^n$ such that $\|A\boldsymbol{x} - \boldsymbol{b}\|_2$ is minimized.

$$f(\boldsymbol{x}) := \|A\boldsymbol{x} - \boldsymbol{b}\|_2^2 = (A\boldsymbol{x} - \boldsymbol{b})^T(A\boldsymbol{x} - \boldsymbol{b}) = \boldsymbol{x}^T A^T A\boldsymbol{x} - 2\boldsymbol{x}^T A^T \boldsymbol{b} + \boldsymbol{b}^T \boldsymbol{b}$$

Since f is a quadratic function on $\mathbb{R}^n$ a *necessary* condition for a minimizer $\boldsymbol{x}$ is that

$$\nabla f(\boldsymbol{x}) = 0 \Longrightarrow 2A^T A\boldsymbol{x} - 2A^T \boldsymbol{b} = 0$$

So any minimizer of f should satisfy

$$A^T A\boldsymbol{x} = A^T \boldsymbol{b}$$

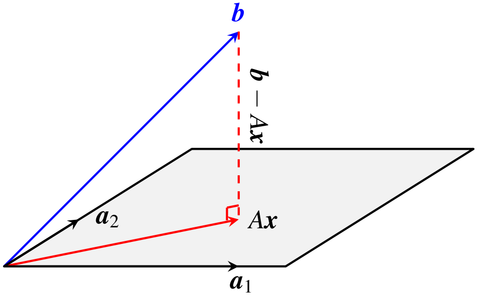
Note: The *Hessian* of f is $2A^TA$ which is a semi-positive definite matrix in general, but since A has full rank it is indeed positive definite. Then the normal equation is also a *sufficient* condition.

A deeper point of view

Let the columns of A be denoted by $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n$. Then the range of A is the subspace of $\mathbb{R}^m$ spanned by columns of A. We look for a vector $\boldsymbol{x}$ such that

$$A\boldsymbol{x} = x_1\boldsymbol{a}_1 + \cdots + x_n\boldsymbol{a}_n$$

is the closest vector to $\boldsymbol{b}$ out of $\mathrm{range}(A)$.



So the error $\boldsymbol{b} - A\boldsymbol{x}$ must be perpendicular to $\mathrm{range}(A)$, i.e.

$$\boldsymbol{b} - A\boldsymbol{x} \perp \boldsymbol{a}_k, \quad k = 1, \ldots, n, \quad \text{or} \quad A^T(\boldsymbol{b} - A\boldsymbol{x}) = 0$$

again gives the normal equation.

Non-linear relationship

Non-linear cases can sometimes be rewritten as linear:

- Ansatz $y = \alpha e^{\beta x}$: rewrite $\ln y = \ln\left(\alpha e^{\beta x}\right)$ or

$$\ln y = \ln \alpha + \beta x \Longrightarrow \tilde{y} = \tilde{\alpha} + \beta x$$

  Form normal equation for data $(x_k, \tilde{y}_k)$ where $\tilde{y}_k = \ln y_k$ and solve for α and β. Get α from $\alpha = e^{\tilde{\alpha}}$.

- Ansatz $y = \alpha x^{\beta}$: rewrite $\log y = \log\left(\alpha x^{\beta}\right)$ or

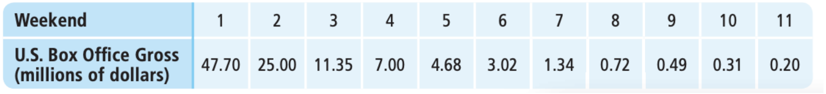
$$\log y = \log \alpha + \beta \log x \Longrightarrow \tilde{y} = \tilde{\alpha} + \beta \tilde{x}$$

  where $\tilde{y} = \log y$, $\tilde{x} = \log x$ and $\tilde{\alpha} = \log \alpha$.
  Use data $(\tilde{x}_k, \tilde{y}_k)$ and solve as above.

Non-linear relationship

Example (exponential decay):

Below are the U.S. box office gross for the first eleven weekends of the release of the movie *Mission Impossible III* in 2006:



Weekend	1	2	3	4	5	6	7	8	9	10	11
U.S. Box Office Gross (millions of dollars)	47.70	25.00	11.35	7.00	4.68	3.02	1.34	0.72	0.49	0.31	0.20

Fit an exponential model for box office gross based on the number of weekends the movie has been out use the model to estimate the movie's gross in its 12th weekend.
Use Python and plot a figure to show both data and the fitted curve.

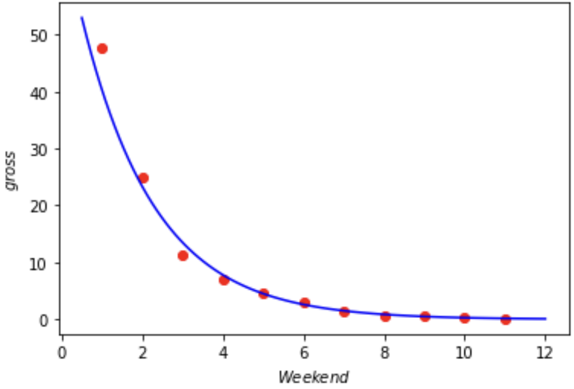
See next pages for code and results ...

Python code ...

```python
import numpy as np
import matplotlib.pyplot as plt

x = np.array([1,2,3,4,5,6,7,8,9,10,11])
y = np.array([47.7,25.0,11.35,7.0,4.68,3.02,1.34,0.72,0.49,0.31,0.2])
log_y = np.log(y)                 # compute the logarithm of y data
A = np.ones((len(x),2))
A[:,1] = x
a = np.linalg.solve(A.T@A, A.T@log_y) # use log(y) instead of y
a[0] = np.exp(a[0])
x_eval = np.linspace(0.5,12,100)
y_eval = a[0]*np.exp(a[1]*x_eval)

plt.plot(x,y,marker = 'o',linestyle = '', color = 'red')
plt.plot(x_eval,y_eval,linestyle = '-', color = 'blue')
plt.xlabel('Weekend'); plt.ylabel('gross')

print("The model is: y = %f exp(%f x)" % (a[0],a[1]))

y_12 = a[0]*np.exp(a[1]*12)
print('Predicted gross at 12th weekend = ', np.round(y_12,3))
```

Python code outputs ...

```
The model is: y = 69.617842 exp(-0.546902 x)
Predicted gross at 12th weekend =  0.098
```



Conclusion: normal equations

- The overdetermined system $Ax = \boldsymbol{b}$ may not have a solution as $\boldsymbol{b}$ may not located in same plane as A's columns (range of A)
- Instead we find the closet (in norm 2) vector $\tilde{\boldsymbol{b}} = A\boldsymbol{x}$ to original vector $\boldsymbol{b}$
- That solution $\boldsymbol{x}$ is obtained by solving the square system $A^T A\boldsymbol{x} = A^T\boldsymbol{b}$ (normal equation)
- If A has full rank then A^TA is positive definite and thus non-singular
- The term Least Squares comes from minimizing sum of squares $\|A\boldsymbol{x} - \boldsymbol{b}\|_2^2$
- The bad news is that the condition number of A^TA is the square of the condition number of A itself
- For polynomial curve fitting: shifting and scaling the data improve the condition number
- For large scale curve fitting problems and other applications we can use orthogonalization instead of using the normal equation (next lectures)

From old exams

3 **Algorithm 1 - A**

The molecular weight (molar mass, g/mol) of nitrogen oxides are tabulated below (three decimal places accuracy):

NO	N_2O_3	N_2O	N_2O_5	NO_2	N_2O_4
30.006	76.012	44.013	108.010	46.006	92.001

Based on the data, do a **least squares** approximation to **calculate** the molecular weight for nitrogen and oxygen, all data must be used. (Hint: The molecule N_2O_3 contains 2 nitrogen atoms and 3 oxygen atom.)

Fill in your answer here

Orthogonality

- Two nonzero vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are called **orthogonal** if $\boldsymbol{x}^T\boldsymbol{y} = 0$. For example
$$\boldsymbol{x} = [1, -1, 0]^T, \quad \boldsymbol{y} = [1, 1, 2]^T$$
are orthogonal because $\boldsymbol{x}^T\boldsymbol{y} = 0$.
- If $\boldsymbol{x}$ and $\boldsymbol{y}$ are orthogonal and $\|\boldsymbol{x}\|_2 = \|\boldsymbol{y}\|_2 = 1$ then they are called **orthonormal**. For example
$$\boldsymbol{x} = [\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}, 0]^T, \quad \boldsymbol{y} = [\frac{1}{\sqrt{6}}, \frac{1}{\sqrt{6}}, \frac{2}{\sqrt{6}}]^T$$
are orthonormal.
- A set of orthogonal (orthonormal) vectors $\boldsymbol{q}_j \in \mathbb{R}^m$, $j = 1, 2, \ldots, m$, constitute a basis for $\mathbb{R}^m$
- A square matrix Q is called an **orthogonal matrix** if its columns are orthonormal. Or if $Q^TQ = I$, or if $Q^{-1} = Q^T$.
- Orthogonal matrices preserve the length (Euclidean norm) of vectors (just rotate them):
$$\|Q\boldsymbol{x}\|_2^2 = (Q\boldsymbol{x})^TQ\boldsymbol{x} = \boldsymbol{x}^TQ^TQ\boldsymbol{x} = \boldsymbol{x}^T\boldsymbol{x} = \|\boldsymbol{x}\|_2^2$$

Examples of orthogonal matrices

- The identity matrix

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [\boldsymbol{q}_1 \ \boldsymbol{q}_2 \ \boldsymbol{q}_3]$$

  It is clear that $\boldsymbol{q}_1^T\boldsymbol{q}_1 = \boldsymbol{q}_2^T\boldsymbol{q}_2 = \boldsymbol{q}_3^T\boldsymbol{q}_3 = 1$ and $\boldsymbol{q}_1^T\boldsymbol{q}_2 = 0$, $\boldsymbol{q}_1^T\boldsymbol{q}_3 = 0$ and $\boldsymbol{q}_2^T\boldsymbol{q}_3 = 0$.

- rotation matrix (rotate coordinate system by angle θ)

$$Q = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

- reflection matrix (mirror y axis)

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

- permutation matrix (change x and y axis)

$$Q = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

QR factorization

Every matrix $A \in R^{m\times n}$ with $m \geqslant n$ (overdetermined) can be factorized as

$$A = QR$$

where Q is a $m \times m$ orthogonal matrix and R is a $m \times n$ upper triangular matrix.

$$\underset{A}{\begin{pmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{pmatrix}} = \underset{Q}{\begin{pmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{pmatrix}} \underset{R}{\left(\begin{array}{ccc} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right)}$$

Three different algorithms for computing Q and R factors:

- Householder algorithm (reflections) (later in the course ...)
- Givens algorithm (rotations)
- Gram-Schmidt algorithm (projections)

Reduced (thin) QR factorization

The last $m-n$ rows of R are zeros so the last $m-n$ columns of Q have no contribution to the product (but are still important!).

$$\begin{pmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{pmatrix} = \begin{pmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{pmatrix} \begin{pmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

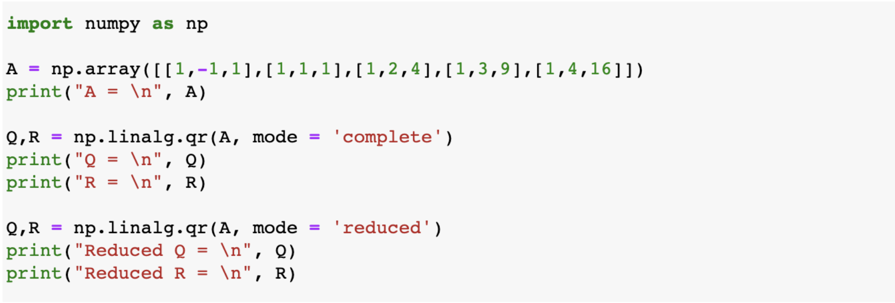
$$A = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$$

The **reduced QR factorization** then is:

$$A = Q_1 R_1$$

In Python ...

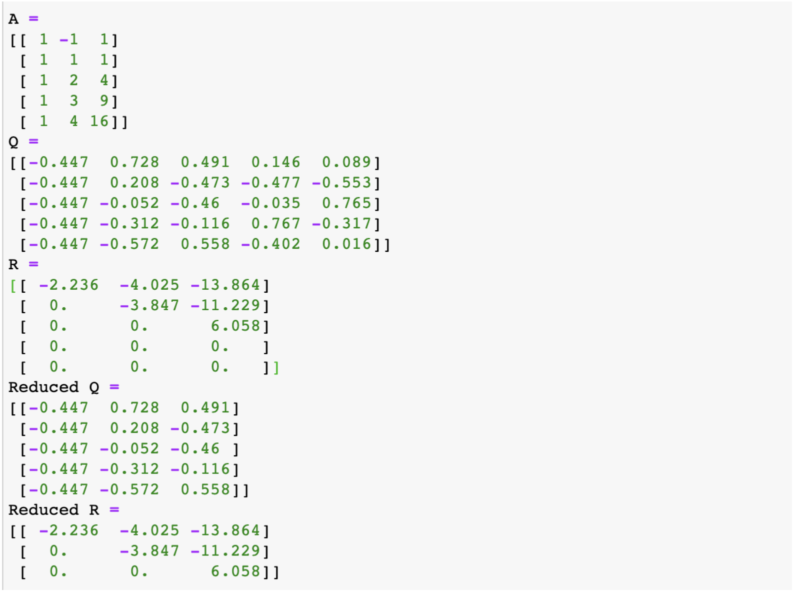
The QR algorithm is implemented in `lingalg` library in `numpy`. Here comes an example:



```
import numpy as np

A = np.array([[1,-1,1],[1,1,1],[1,2,4],[1,3,9],[1,4,16]])
print("A = \n", A)

Q,R = np.linalg.qr(A, mode = 'complete')
print("Q = \n", Q)
print("R = \n", R)

Q,R = np.linalg.qr(A, mode = 'reduced')
print("Reduced Q = \n", Q)
print("Reduced R = \n", R)
```

Output: next pages ... (numbers edited to 3 decimals)

In Python ...



```
A =
[[ 1 -1  1]
 [ 1  1  1]
 [ 1  2  4]
 [ 1  3  9]
 [ 1  4 16]]
Q =
[[-0.447  0.728  0.491  0.146  0.089]
 [-0.447  0.208 -0.473 -0.477 -0.553]
 [-0.447 -0.052 -0.46  -0.035  0.765]
 [-0.447 -0.312 -0.116  0.767 -0.317]
 [-0.447 -0.572  0.558 -0.402  0.016]]
R =
[[ -2.236  -4.025 -13.864]
 [  0.     -3.847 -11.229]
 [  0.      0.      6.058]
 [  0.      0.      0.   ]
 [  0.      0.      0.   ]]
Reduced Q =
[[-0.447  0.728  0.491]
 [-0.447  0.208 -0.473]
 [-0.447 -0.052 -0.46 ]
 [-0.447 -0.312 -0.116]
 [-0.447 -0.572  0.558]]
Reduced R =
[[ -2.236  -4.025 -13.864]
 [  0.     -3.847 -11.229]
 [  0.      0.      6.058]]
```

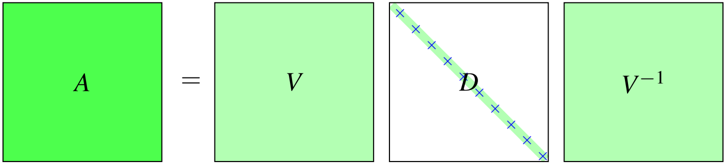
Some matrix decompositions

$A = QR$ is one example of matrix decompositions. Other examples are:

$A = LU$	LU decomposition, Gaussian elimination
$A = LL^T$	Cholesky decomposition, LU decomposition when A is symmetric positive definite
$A = LDL^T$	LU decomposition when A is symmetric
$A = QR$	QR factorization, orthogonalization
$A = QDQ^T, \ A = VDV^{-1}$	Eigendecomposition, eigenvalues/vectors
$A = U\Sigma V^T$	Singular value decomposition (SVD)

Eigendecomposition

Some matrices have a decomposition as below:



- Here V is a non-singular matrix (columns are eigenvectors of A), and D is a diagonal matrix (diagonals are eigenvalues of A).
- Not all matrices have such decomposition. If A has this decomposition then A is called diagonalizibale.
- If A is symmetric then V is orthogonal and $V^{-1} = V^T$. We get $A = VDV^T$.

From old exams

Analysis 2 - A

A matrix has the eigenvectors

$$v_1 = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}, v_2 = \begin{bmatrix} -\frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} v_3 = \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix} v_4 = \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \end{bmatrix}$$

and eigenvalues $\lambda_1 = 0.1, \lambda_2 = 2, \lambda_3 = 3, \text{ and } \lambda_4 = 4.$

What is the condition number $\text{cond}_2(A)$?

Select one alternative:

- ○ 40
- ○ 2
- ○ 4
- ○ 0.1