July 20, 2013

Chapter 6: Linear Least Squares Problems

Uri M. Ascher and Chen Greif
Department of Computer Science
The University of British Columbia
{ascher,greif}@cs.ubc.ca

Slides for the book
A First Course in Numerical Methods (published by SIAM, 2011)
`http://www.ec-securehost.com/SIAM/CS07.html`

Goals of this chapter

- To introduce and solve the linear least squares problem, ubiquitous in data fitting applications;
- to introduce algorithms based on orthogonal transformations;
- to evaluate different algorithms and understand what their basic features translate into in terms of a tradeoff between stability and efficiency.

Outline

- Normal equations
- Application: data fitting
- Orthogonal transformations and QR decomposition
- Householder transformations and Gram-Schmidt orthogonalization

Linear least-squares

- Throughout this chapter we consider the problem

$$\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_2,$$

where A is $m \times n$, with $m > n$.

- So, it is an overdetermined system of equations: we have more rows, for instance corresponding to data measurements, than columns, where $\mathbf{x}$ corresponds to unknown model parameters.
- In general, there is no $\mathbf{x}$ satisfying $A\mathbf{x} = \mathbf{b}$, hence we seek to minimize a norm of the residual $\mathbf{r} = \mathbf{b} - A\mathbf{x}$. The ℓ_2 norm is the most convenient to work with, although it is not suitable for all purposes, and it enjoys rich theory.
- Assume A has linearly independent columns. Then there is a unique solution to this problem, as we'll soon see.

Normal equations

- Drop the index 2: $\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|$.
- Equivalent to minimizing
  $\psi(\mathbf{x}) = \frac{1}{2}\|\mathbf{b} - A\mathbf{x}\|^2 = \frac{1}{2}\sum_{i=1}^{m}\left(b_i - \sum_{j=1}^{n} a_{ij}x_j\right)^2$.
- Necessary conditions: $\frac{\partial}{\partial x_k}\psi(\mathbf{x}) = 0, \quad k = 1, \ldots, n$.
- So,

$$\sum_{i=1}^{m}\left[\left(b_i - \sum_{j=1}^{n} a_{ij}x_j\right)(-a_{ik})\right] = 0.$$

- In matrix-vector form this expression looks much simpler:
  $A^T A\mathbf{x} = A^T\mathbf{b}$.
- Also *sufficient* for minimum because $\nabla^2\psi = A^T A$ is positive definite.

Normal equations

- Drop the index 2: $\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|$.
- Equivalent to minimizing
  $\psi(\mathbf{x}) = \frac{1}{2}\|\mathbf{b} - A\mathbf{x}\|^2 = \frac{1}{2}\sum_{i=1}^{m}\left(b_i - \sum_{j=1}^{n} a_{ij}x_j\right)^2$.
- Necessary conditions: $\frac{\partial}{\partial x_k}\psi(\mathbf{x}) = 0, \quad k = 1, \ldots, n$.
- So,

$$\sum_{i=1}^{m}\left[\left(b_i - \sum_{j=1}^{n} a_{ij}x_j\right)(-a_{ik})\right] = 0.$$

- In matrix-vector form this expression looks much simpler:
  $A^T A\mathbf{x} = A^T\mathbf{b}$.

- Also *sufficient* for minimum because $\nabla^2\psi = A^T A$ is positive definite.

Normal equations

- Drop the index 2: $\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|$.
- Equivalent to minimizing
  $\psi(\mathbf{x}) = \frac{1}{2}\|\mathbf{b} - A\mathbf{x}\|^2 = \frac{1}{2}\sum_{i=1}^{m}\left(b_i - \sum_{j=1}^{n} a_{ij}x_j\right)^2$.
- Necessary conditions: $\frac{\partial}{\partial x_k}\psi(\mathbf{x}) = 0, \quad k = 1, \ldots, n$.
- So,
  $$\sum_{i=1}^{m}\left[\left(b_i - \sum_{j=1}^{n} a_{ij}x_j\right)(-a_{ik})\right] = 0.$$
- In matrix-vector form this expression looks much simpler:
  $A^T A\mathbf{x} = A^T\mathbf{b}$.
- Also *sufficient* for minimum because $\nabla^2\psi = A^T A$ is positive definite.

Normal equations algorithm

$$\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|.$$

- Assume A has linearly independent columns. Then for an optimum it is necessary and sufficient to satisfy the **normal equations**

  $$A^T A\mathbf{x} = A^T\mathbf{b}.$$

- So, can use techniques from Chapter 5 to solve the problem.
- Algorithm:
  1. Form $B = A^T A$ and $\mathbf{y} = A^T\mathbf{b}$
  2. Compute the Cholesky factorization $B = R^T R$
  3. Solve the lower triangular system $R^T\mathbf{z} = \mathbf{y}$
  4. Solve the upper triangular system $R\mathbf{x} = \mathbf{z}$
- Simple, efficient, classical.

Normal equations algorithm

$$\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|.$$

- Assume A has linearly independent columns. Then for an optimum it is necessary and sufficient to satisfy the **normal equations**

  $$A^T A\mathbf{x} = A^T\mathbf{b}.$$

- So, can use techniques from Chapter 5 to solve the problem.
- Algorithm:
  1. Form $B = A^T A$ and $\mathbf{y} = A^T\mathbf{b}$
  2. Compute the Cholesky factorization $B = R^T R$
  3. Solve the lower triangular system $R^T\mathbf{z} = \mathbf{y}$
  4. Solve the upper triangular system $R\mathbf{x} = \mathbf{z}$

- Simple, efficient, classical.

Normal equations algorithm

$$\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|.$$

- Assume A has linearly independent columns. Then for an optimum it is necessary and sufficient to satisfy the **normal equations**

$$A^T A\mathbf{x} = A^T \mathbf{b}.$$

- So, can use techniques from Chapter 5 to solve the problem.
- Algorithm:
  1. Form $B = A^T A$ and $\mathbf{y} = A^T \mathbf{b}$
  2. Compute the Cholesky factorization $B = R^T R$
  3. Solve the lower triangular system $R^T \mathbf{z} = \mathbf{y}$
  4. Solve the upper triangular system $R\mathbf{x} = \mathbf{z}$

- Simple, efficient, classical.

$$\mathbf{r} = \mathbf{b} - A\mathbf{x}$$

$$B = A^T A$$

Example

- Consider the least-squares problem $\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|$ for

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 2 & 3 & 5 \\ 5 & 3 & -2 \\ 3 & 5 & 4 \\ -1 & 6 & 3 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 4 \\ -2 \\ 5 \\ -2 \\ 1 \end{pmatrix}.$$

- Solving via normal equations: form

$$B = A^T A = \begin{pmatrix} 40 & 30 & 10 \\ 30 & 79 & 47 \\ 10 & 47 & 55 \end{pmatrix}, \quad \mathbf{y} = A^T \mathbf{b} = \begin{pmatrix} 18 \\ 5 \\ -21 \end{pmatrix};$$

  solve $B\mathbf{x} = \mathbf{y}$ obtaining $\mathbf{x} = (.3472, .3990, -.7859)^T$.

- The optimal residual (rounded) is

$$\mathbf{r} = \mathbf{b} - A\mathbf{x} = (4.4387, .0381, .495, -1.893, 1.311)^T.$$

  This vector is orthogonal to each column of A.

Example

- Consider the least-squares problem $\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|$ for

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 2 & 3 & 5 \\ 5 & 3 & -2 \\ 3 & 5 & 4 \\ -1 & 6 & 3 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 4 \\ -2 \\ 5 \\ -2 \\ 1 \end{pmatrix}.$$

- Solving via normal equations: form

$$B = A^T A = \begin{pmatrix} 40 & 30 & 10 \\ 30 & 79 & 47 \\ 10 & 47 & 55 \end{pmatrix}, \quad \mathbf{y} = A^T\mathbf{b} = \begin{pmatrix} 18 \\ 5 \\ -21 \end{pmatrix};$$

solve $B\mathbf{x} = \mathbf{y}$ obtaining $\mathbf{x} = (.3472, .3990, -.7859)^T$.

- The optimal residual (rounded) is

$$\mathbf{r} = \mathbf{b} - A\mathbf{x} = (4.4387, .0381, .495, -1.893, 1.311)^T.$$

This vector is orthogonal to each column of A.

Example

- Consider the least-squares problem $\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|$ for

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 2 & 3 & 5 \\ 5 & 3 & -2 \\ 3 & 5 & 4 \\ -1 & 6 & 3 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 4 \\ -2 \\ 5 \\ -2 \\ 1 \end{pmatrix}.$$

- Solving via normal equations: form

$$B = A^T A = \begin{pmatrix} 40 & 30 & 10 \\ 30 & 79 & 47 \\ 10 & 47 & 55 \end{pmatrix}, \quad \mathbf{y} = A^T \mathbf{b} = \begin{pmatrix} 18 \\ 5 \\ -21 \end{pmatrix};$$

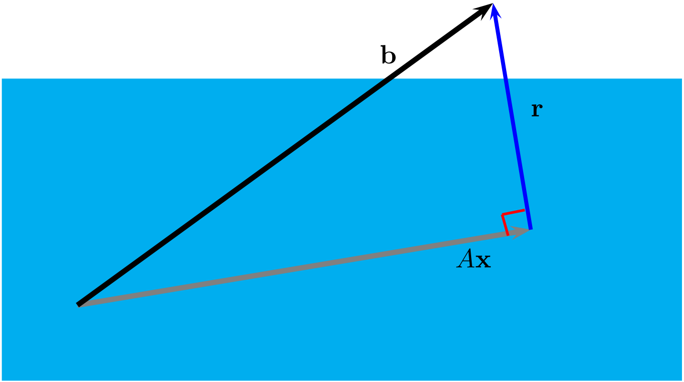
  solve $B\mathbf{x} = \mathbf{y}$ obtaining $\mathbf{x} = (.3472, .3990, -.7859)^T$.

- The optimal residual (rounded) is

$$\mathbf{r} = \mathbf{b} - A\mathbf{x} = (4.4387, .0381, .495, -1.893, 1.311)^T.$$

  This vector is orthogonal to each column of A.

Orthogonality of the residual



Normal equations facts

- The residual vector $\mathbf{r} = \mathbf{b} - A\mathbf{x}$ is *orthogonal* to the columns of A: $A^T\mathbf{r} = \mathbf{0}$.
- Thus, $\mathbf{b}$ is orthogonally projected to the space range(A).
- Define pseudo-inverse of A by

$$A^\dagger = B^{-1}A^T.$$

- For $m \gg n$, most of the algorithm cost is in the formation of $B = A^TA$.
- This is the way to solve many data fitting problems.
- But, difficulties arise when A has (almost) linearly dependent columns.

Normal equations facts

- The residual vector $\mathbf{r} = \mathbf{b} - A\mathbf{x}$ is *orthogonal* to the columns of A: $A^T\mathbf{r} = \mathbf{0}$.
- Thus, $\mathbf{b}$ is orthogonally projected to the space range(A).
- Define pseudo-inverse of A by

$$A^\dagger = B^{-1}A^T.$$

- For $m \gg n$, most of the algorithm cost is in the formation of $B = A^TA$.
- This is the way to solve many data fitting problems.
- But, difficulties arise when A has (almost) linearly dependent columns.

Outline

- Normal equations
- Application: data fitting
- Orthogonal transformations and QR decomposition
- Householder transformations and Gram-Schmidt orthogonalization

Application: data fitting

Given measurements, or observations

$$(t_1, b_1), (t_2, b_2), \ldots, (t_m, b_m) = \{(t_i, b_i)\}_{i=1}^{m},$$

want to fit a function

$$v(t) = \sum_{j=1}^{n} x_j \phi_j(t),$$

- $\phi_1(t), \phi_2(t), \ldots, \phi_n(t)$ are known linearly independent basis functions
- $x_1, \ldots, x_n$ are coefficients to be determined (we wish) s.t.

$$v(t_i) = b_i, \quad i = 1, 2, \ldots, m.$$

Data fitting (cont.)

Define $a_{ij} = \phi_j(t_i)$. Want $A\mathbf{x} = \mathbf{b}$, where

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \ \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \ \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}.$$

1. If $m = n$ get interpolation problem.
2. If $m > n$ we can't generally set $\mathbf{r} = \mathbf{b} - A\mathbf{x} = \mathbf{0}$. So relax requirement: we want, e.g., $\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_2$. Obtain a least-squares data fitting problem.

Note: even if we can increase n so that A becomes square, there may be reasons not to want this:

1. A smaller n gives fewer parameters to control and may better describe global trend of data.
2. If the data contains noise, don't want to over-fit it.

Data fitting (cont.)

Define $a_{ij} = \phi_j(t_i)$. Want $A\mathbf{x} = \mathbf{b}$, where

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}.$$

1. If $m = n$ get interpolation problem.
2. If $m > n$ we can't generally set $\mathbf{r} = \mathbf{b} - A\mathbf{x} = \mathbf{0}$. So relax requirement: we want, e.g., $\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_2$. Obtain a least-squares data fitting problem.

Note: even if we can increase n so that A becomes square, there may be reasons not to want this:

1. A smaller n gives fewer parameters to control and may better describe global trend of data.
2. If the data contains noise, don't want to over-fit it.

Data fitting (cont.)

Define $a_{ij} = \phi_j(t_i)$. Want $A\mathbf{x} = \mathbf{b}$, where

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \ \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \ \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}.$$

1. If $m = n$ get interpolation problem.
2. If $m > n$ we can't generally set $\mathbf{r} = \mathbf{b} - A\mathbf{x} = \mathbf{0}$. So relax requirement: we want, e.g., $\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_2$. Obtain a least-squares data fitting problem.

Note: even if we can increase n so that A becomes square, there may be reasons not to want this:

1. A smaller n gives fewer parameters to control and may better describe global trend of data.
2. If the data contains noise, don't want to over-fit it.

Data fitting (cont.)

Define $a_{ij} = \phi_j(t_i)$. Want $A\mathbf{x} = \mathbf{b}$, where

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \ \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \ \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}.$$

1. If $m = n$ get interpolation problem.
2. If $m > n$ we can't generally set $\mathbf{r} = \mathbf{b} - A\mathbf{x} = \mathbf{0}$. So relax requirement: we want, e.g., $\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_2$. Obtain a least-squares data fitting problem.

Note: even if we can increase n so that A becomes square, there may be reasons not to want this:

1. A smaller n gives fewer parameters to control and may better describe global trend of data.
2. If the data contains noise, don't want to over-fit it.

Example: linear regression

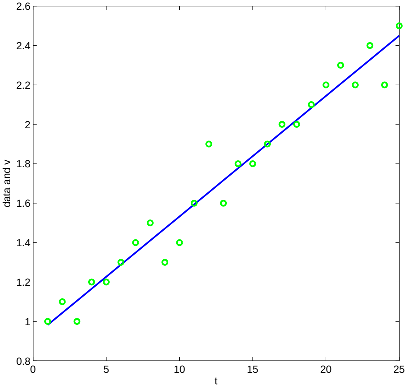


FIGURE: Linear regression curve (in blue) through green data points. Here $m = 25$ and $n = 2$.

Data fitting example

$\phi_j(t) = t^{j-1}, \ j = 1, 2, \ldots, n$

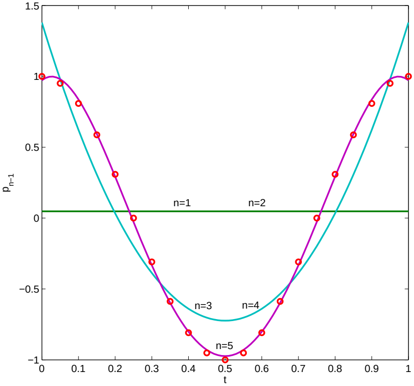


FIGURE: First 5 best polynomial approximations to $f(t) = \cos(2\pi t)$ sampled at $0 : .05 : 1$. The data values appear as red circles. Note $p_{2j+1} = p_{2j}$ (uncommon; due to symmetry).

Other data fitting problems

- When $m > n$, i.e., A "long and skinny", we have an over-determined system.
- Instead of least-squares, may consider

$$\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_1 \quad \text{or} \quad \min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_\infty$$

  for the over-determined problem. Both of these lead to linear programming formulations.
- If $m < n$ we have an under-determined system. Now there are many solutions to $A\mathbf{x} = \mathbf{b}$: want to pick one wisely. For instance,

$$\min_{\mathbf{x}} \{\|\mathbf{x}\|_2 \quad s.t. \quad A\mathbf{x} = \mathbf{b}\}$$

Other data fitting problems

- When $m > n$, i.e., A "long and skinny", we have an over-determined system.
- Instead of least-squares, may consider

$$\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_1 \quad \text{or} \quad \min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_\infty$$

  for the over-determined problem. Both of these lead to linear programming formulations.
- If $m < n$ we have an under-determined system. Now there are many solutions to $A\mathbf{x} = \mathbf{b}$: want to pick one wisely. For instance,

$$\min_{\mathbf{x}} \{\|\mathbf{x}\|_2 \quad s.t. \quad A\mathbf{x} = \mathbf{b}\}$$

Other data fitting problems

- When $m > n$, i.e., A "long and skinny", we have an over-determined system.
- Instead of least-squares, may consider

$$\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_1 \quad \text{or} \quad \min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_\infty$$

  for the over-determined problem. Both of these lead to linear programming formulations.
- If $m < n$ we have an under-determined system. Now there are many solutions to $A\mathbf{x} = \mathbf{b}$: want to pick one wisely. For instance,

$$\min_{\mathbf{x}} \{\|\mathbf{x}\|_2 \quad s.t. \quad A\mathbf{x} = \mathbf{b}\}$$

Outline

- Normal equations
- Application: data fitting
- Orthogonal transformations and QR decomposition
- Householder transformations and Gram-Schmidt orthogonalization

Condition number of rectangular matrix

- For a rectangular $m \times n$ matrix A, $m \geq n$, let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$ be the eigenvalues of $B = A^T A$.
- Recall (Chapter 4) that the singular values $\sigma_1, \ldots, \sigma_n$ of A are

$$\sigma_i = \sqrt{\lambda_i}, \quad i = 1, 2, \ldots, n.$$

- Define the condition number

$$\kappa(A) = \kappa_2(A) = \frac{\sigma_1}{\sigma_n} = \sqrt{\frac{\lambda_1}{\lambda_n}}.$$

- Note

$$\kappa_2(A^T A) = \frac{\lambda_1}{\lambda_n} = [\kappa(A)]^2.$$

- So, the problem's condition number is *squared* by going through the *normal equations algorithm*. Therefore, seek more stable alternatives.

Condition number of rectangular matrix

- For a rectangular $m \times n$ matrix A, $m \geq n$, let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$ be the eigenvalues of $B = A^T A$.
- Recall (Chapter 4) that the singular values $\sigma_1, \ldots, \sigma_n$ of A are

$$\sigma_i = \sqrt{\lambda_i}, \quad i = 1, 2, \ldots, n.$$

- Define the condition number

$$\kappa(A) = \kappa_2(A) = \frac{\sigma_1}{\sigma_n} = \sqrt{\frac{\lambda_1}{\lambda_n}}.$$

- Note

$$\kappa_2(A^T A) = \frac{\lambda_1}{\lambda_n} = [\kappa(A)]^2.$$

- So, the problem's condition number is *squared* by going through the *normal equations algorithm*. Therefore, seek more stable alternatives.

Orthogonal matrices (Chapter 4)

Recall that

- two vectors $\mathbf{u},\ \mathbf{v}$ are orthogonal if

$$\mathbf{u}^T\mathbf{v} = 0,$$

  and orthonormal if in addition $\|\mathbf{u}\| = \|\mathbf{v}\| = 1$;
- a square matrix Q is orthogonal if its columns are pairwise orthonormal, i.e.,

$$Q^TQ = I. \quad \text{Hence also } Q^{-1} = Q^T.$$

- Important property: for any vector $\mathbf{z}$ of suitable size,

$$\|Q\mathbf{z}\|_2 = \sqrt{\mathbf{z}^TQ^TQ\mathbf{z}} = \sqrt{\mathbf{z}^T\mathbf{z}} = \|\mathbf{z}\|_2.$$

  So, orthogonal transformations preserve the ℓ_2 vector norm.

LS through orthogonal transformations

- Normal equations can be bad because the *stability of the algorithm* depends on the *conditioning of the problem*.
- Instead, transform the problem: for any orthogonal matrix P

$$\|\mathbf{b} - A\mathbf{x}\| = \|\mathbf{r}\| = \|P\mathbf{r}\| = \|P\mathbf{b} - PA\mathbf{x}\|.$$

- So, find orthogonal matrix $Q = P^T$ that transforms A into upper triangular form

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}.$$

This is a QR decomposition.

Questions:

1. Suppose we have carried out a QR decomposition: what can we do with it?
2. How to obtain a QR decomposition?

LS through orthogonal transformations

- Normal equations can be bad because the *stability of the algorithm* depends on the *conditioning of the problem*.
- Instead, transform the problem: for any orthogonal matrix P

$$\|\mathbf{b} - A\mathbf{x}\| = \|\mathbf{r}\| = \|P\mathbf{r}\| = \|P\mathbf{b} - PA\mathbf{x}\|.$$

- So, find orthogonal matrix $Q = P^T$ that transforms A into upper triangular form

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}.$$

This is a QR decomposition.

Questions:

1. Suppose we have carried out a QR decomposition: what can we do with it?
2. How to obtain a QR decomposition?

Solving LS given QR decomposition

Suppose we are given Q and R. Then

$$\|\mathbf{b} - A\mathbf{x}\| = \|\mathbf{b} - Q\begin{pmatrix} R \\ 0 \end{pmatrix}\mathbf{x}\| = \|Q^T\mathbf{b} - \begin{pmatrix} R \\ 0 \end{pmatrix}\mathbf{x}\|.$$

Denote

$$Q^T\mathbf{b} = \begin{pmatrix} \mathbf{c} \\ \mathbf{d} \end{pmatrix}.$$

Then

$$\|\mathbf{r}\|^2 = \|\mathbf{b} - A\mathbf{x}\|^2 = \|\mathbf{c} - R\mathbf{x}\|^2 + \|\mathbf{d}\|^2.$$

1. Solve $R\mathbf{x} = \mathbf{c}$ by back substitution.
2. Obtain also the residual norm $\|\mathbf{r}\| = \|\mathbf{d}\|$.
3. Stable algorithm: the condition number of A is not squared.
4. MATLAB implements this method in the backslash command for over-determined systems.

Solving LS given QR decomposition

Suppose we are given Q and R. Then

$$\|\mathbf{b} - A\mathbf{x}\| = \|\mathbf{b} - Q\begin{pmatrix} R \\ 0 \end{pmatrix}\mathbf{x}\| = \|Q^T\mathbf{b} - \begin{pmatrix} R \\ 0 \end{pmatrix}\mathbf{x}\|.$$

Denote

$$Q^T\mathbf{b} = \begin{pmatrix} \mathbf{c} \\ \mathbf{d} \end{pmatrix}.$$

Then

$$\|\mathbf{r}\|^2 = \|\mathbf{b} - A\mathbf{x}\|^2 = \|\mathbf{c} - R\mathbf{x}\|^2 + \|\mathbf{d}\|^2.$$

1. Solve $R\mathbf{x} = \mathbf{c}$ by back substitution.
2. Obtain also the residual norm $\|\mathbf{r}\| = \|\mathbf{d}\|$.
3. Stable algorithm: the condition number of A is not squared.
4. MATLAB implements this method in the backslash command for over-determined systems.

Solving LS given QR decomposition

Suppose we are given Q and R. Then

$$\|\mathbf{b} - A\mathbf{x}\| = \|\mathbf{b} - Q \begin{pmatrix} R \\ 0 \end{pmatrix} \mathbf{x}\| = \|Q^T\mathbf{b} - \begin{pmatrix} R \\ 0 \end{pmatrix} \mathbf{x}\|.$$

Denote

$$Q^T\mathbf{b} = \begin{pmatrix} \mathbf{c} \\ \mathbf{d} \end{pmatrix}.$$

Then

$$\|\mathbf{r}\|^2 = \|\mathbf{b} - A\mathbf{x}\|^2 = \|\mathbf{c} - R\mathbf{x}\|^2 + \|\mathbf{d}\|^2.$$

1. Solve $R\mathbf{x} = \mathbf{c}$ by back substitution.
2. Obtain also the residual norm $\|\mathbf{r}\| = \|\mathbf{d}\|$.
3. Stable algorithm: the condition number of A is not squared.
4. MATLAB implements this method in the backslash command for over-determined systems.

Solving LS given QR decomposition

Suppose we are given Q and R. Then

$$\|\mathbf{b} - A\mathbf{x}\| = \|\mathbf{b} - Q\begin{pmatrix} R \\ 0 \end{pmatrix}\mathbf{x}\| = \|Q^T\mathbf{b} - \begin{pmatrix} R \\ 0 \end{pmatrix}\mathbf{x}\|.$$

Denote

$$Q^T\mathbf{b} = \begin{pmatrix} \mathbf{c} \\ \mathbf{d} \end{pmatrix}.$$

Then

$$\|\mathbf{r}\|^2 = \|\mathbf{b} - A\mathbf{x}\|^2 = \|\mathbf{c} - R\mathbf{x}\|^2 + \|\mathbf{d}\|^2.$$

1. Solve $R\mathbf{x} = \mathbf{c}$ by back substitution.
2. Obtain also the residual norm $\|\mathbf{r}\| = \|\mathbf{d}\|$.
3. Stable algorithm: the condition number of A is not squared.
4. MATLAB implements this method in the backslash command for over-determined systems.

Comparing run times using normal equations vs QR decomposition

- The flop count of efficient QR decomposition is $2mn^2 - 2n^3/3$, roughly double that of normal equations when $m \gg n$. Thus, the better stability has its price in efficiency!
- Let's see the comparative cost using MATLAB's backslash.

```
for n = 300:100:1000
   m = 3*n+1; % or m= n+1, or something else
   A = randn(m,n); b = randn(m,1);
   % solve and find execution times; first, Matlab way using QR
   t0 = cputime; xqr = A \ b; temp = cputime;
   tqr(n/100-2) = temp - t0;
   % next use normal equations
   t0 = temp; B = A'*A; y = A'*b; xne = B \ y; temp = cputime;
   tne(n/100-2) = temp - t0;
end
ratio = tqr./tne;
plot(300:100:1000,ratio)
```

Comparison results

The ratios in the figure below are roughly twice as large as flop counts predict. Of course, we don't *exactly* know what's in MATLAB's backslash...

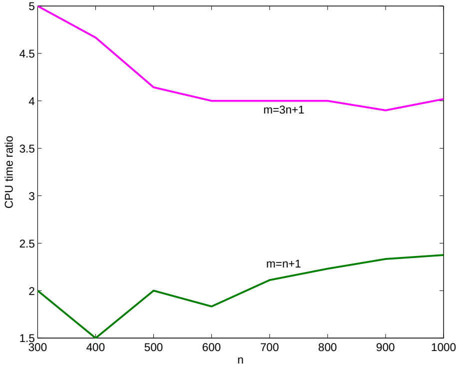


FIGURE: Ratio of execution times using QR over normal equations. The number of rows for each n is $3n+1$ for the upper curve and $n+1$ for the lower one.

Outline

- Normal equations
- Application: data fitting
- Orthogonal transformations and QR decomposition
- Householder transformations and Gram-Schmidt orthogonalization

Gram-Schmidt Orthogonalization

- Now we turn to constructing the QR decomposition of A. To get the feel for our first algorithm, consider a 3×2 instance

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} = \begin{pmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \\ q_{31} & q_{32} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix}.$$

- For notational convenience, denote inner products by

$$\langle \mathbf{z}, \mathbf{y} \rangle \equiv \mathbf{z}^T \mathbf{y}.$$

- Writing the above column by column we have

$$\langle \mathbf{a}_1, \ \mathbf{a}_2 \rangle = \langle r_{11}\mathbf{q}_1, \ r_{12}\mathbf{q}_1 + r_{22}\mathbf{q}_2 \rangle.$$

- Requiring orthonormal columns yields the three conditions $\|\mathbf{q}_1\| = 1, \ \|\mathbf{q}_2\| = 1$ and $\langle \mathbf{q}_1, \mathbf{q}_2 \rangle = 0$.

Gram-Schmidt orthogonalization (cont.)

$$\langle \mathbf{a}_1,\ \mathbf{a}_2 \rangle = \langle r_{11}\mathbf{q}_1,\ r_{12}\mathbf{q}_1 + r_{22}\mathbf{q}_2 \rangle.$$

- For the first column, use $\|\mathbf{q}_1\| = 1$ to obtain

  $$r_{11} = \|\mathbf{a}_1\|; \quad \mathbf{q}_1 = \mathbf{a}_1 / r_{11}.$$

- For the second column we have $\mathbf{a}_2 = r_{12}\mathbf{q}_1 + r_{22}\mathbf{q}_2$, and applying an inner product with $\mathbf{q}_1$ yields

  $$r_{12} = \langle \mathbf{a}_2, \mathbf{q}_1 \rangle.$$

- Next, once r_{12} is known we can compute $\tilde{\mathbf{q}}_2 = \mathbf{a}_2 - r_{12}\mathbf{q}_1$ and then set $r_{22} = \|\tilde{\mathbf{q}}_2\|$ and $\mathbf{q}_2 = \tilde{\mathbf{q}}_2 / r_{22}$.
- Induction completes the procedure for a general $m \times n$ matrix A, giving the QR decomposition in economy size version where Q has the size of A.

Modified Gram-Schmidt orthogonalization

- The classical Gram-Schmidt method becomes unstable when $\kappa(A)$ is large.
- The modified Gram-Schmidt is more stable and is defined in the following algorithm.

Input: matrix A of size $m \times n$.

for $j = 1 : n$
  $\mathbf{q}_j = \mathbf{a}_j$
  for $i = 1 : j-1$
    $r_{ij} = \langle \mathbf{q}_j, \mathbf{q}_i \rangle$
    $\mathbf{q}_j = \mathbf{q}_j - r_{ij}\mathbf{q}_i$
  end
  $r_{jj} = \|\mathbf{q}_j\|$
  $\mathbf{q}_j = \mathbf{q}_j / r_{jj}$
end

Modified Gram-Schmidt orthogonalization

- The classical Gram-Schmidt method becomes unstable when $\kappa(A)$ is large.
- The modified Gram-Schmidt is more stable and is defined in the following algorithm.

Input: matrix A of size $m \times n$.

$$
\begin{array}{l}
\text{for } j = 1:n \\
\quad \mathbf{q}_j = \mathbf{a}_j \\
\quad \text{for } i = 1:j-1 \\
\quad\quad r_{ij} = \langle \mathbf{q}_j, \mathbf{q}_i \rangle \\
\quad\quad \mathbf{q}_j = \mathbf{q}_j - r_{ij}\mathbf{q}_i \\
\quad \text{end} \\
\quad r_{jj} = \|\mathbf{q}_j\| \\
\quad \mathbf{q}_j = \mathbf{q}_j / r_{jj} \\
\text{end}
\end{array}
$$

Householder transformations

- Our second, and preferred QR decomposition algorithm.
- Like elementary $M^{(k)}$ in LU decomposition, each zeros out column under main diagonal.
- Unlike LU these elementary transformations are orthogonal. Very stable, but the cost doubles.
- Let $P = I - 2\mathbf{u}\mathbf{u}^T, \quad \beta = 2\mathbf{u}^T\mathbf{z}$.
  Want $\mathbf{u}$ such that $P\mathbf{z} = \alpha\mathbf{e}_1$ and $\|\mathbf{u}\| = 1$. Then P orthogonal and $\|\mathbf{z}\| = \|P\mathbf{z}\| = |\alpha|$, hence $\alpha = \pm\|\mathbf{z}\|$. Now,

$$P\mathbf{z} = \mathbf{z} - 2\mathbf{u}\mathbf{u}^T\mathbf{z} = \mathbf{z} - \beta\mathbf{u} = \alpha\mathbf{e}_1.$$

  So set

$$\mathbf{u} = \mathbf{z} \pm \|\mathbf{z}\|\mathbf{e}_1; \quad \mathbf{u} = \mathbf{u}/\|\mathbf{u}\|.$$

Householder transformations

- Our second, and preferred QR decomposition algorithm.
- Like elementary $M^{(k)}$ in LU decomposition, each zeros out column under main diagonal.
- Unlike LU these elementary transformations are orthogonal. Very stable, but the cost doubles.
- Let $P = I - 2\mathbf{u}\mathbf{u}^T, \quad \beta = 2\mathbf{u}^T\mathbf{z}$.
  Want $\mathbf{u}$ such that $P\mathbf{z} = \alpha\mathbf{e}_1$ and $\|\mathbf{u}\| = 1$. Then P orthogonal and $\|\mathbf{z}\| = \|P\mathbf{z}\| = |\alpha|$, hence $\alpha = \pm\|\mathbf{z}\|$. Now,

$$P\mathbf{z} = \mathbf{z} - 2\mathbf{u}\mathbf{u}^T\mathbf{z} = \mathbf{z} - \beta\mathbf{u} = \alpha\mathbf{e}_1.$$

  So set

$$\mathbf{u} = \mathbf{z} \pm \|\mathbf{z}\|\mathbf{e}_1; \quad \mathbf{u} = \mathbf{u}/\|\mathbf{u}\|.$$

Householder transformations

- Our second, and preferred QR decomposition algorithm.
- Like elementary $M^{(k)}$ in LU decomposition, each zeros out column under main diagonal.
- Unlike LU these elementary transformations are orthogonal. Very stable, but the cost doubles.
- Let $P = I - 2\mathbf{u}\mathbf{u}^T, \quad \beta = 2\mathbf{u}^T\mathbf{z}$.
  Want $\mathbf{u}$ such that $P\mathbf{z} = \alpha\mathbf{e}_1$ and $\|\mathbf{u}\| = 1$. Then P orthogonal and $\|\mathbf{z}\| = \|P\mathbf{z}\| = |\alpha|$, hence $\alpha = \pm\|\mathbf{z}\|$. Now,

$$P\mathbf{z} = \mathbf{z} - 2\mathbf{u}\mathbf{u}^T\mathbf{z} = \mathbf{z} - \beta\mathbf{u} = \alpha\mathbf{e}_1.$$

So set

$$\mathbf{u} = \mathbf{z} \pm \|\mathbf{z}\|\mathbf{e}_1; \quad \mathbf{u} = \mathbf{u}/\|\mathbf{u}\|.$$

QR decomposition using Householder transformations

```
function [A,p] = house(A)
[m,n]=size(A); p = zeros(1,n);
for k = 1:n
    z = A(k:m,k);
    e1 = [1; zeros(m-k,1)];
    u = z+sign(z(1))*norm(z)*e1; u = u/norm(u);
    % update nonzero part of A by I - 2uu^T
    A(k:m,k:n) = A(k:m,k:n)-2*u*(u'*A(k:m,k:n));
    % store u
    p(k) = u(1);
    A(k+1:m,k) = u(2:m-k+1);
end
```

See text for code to solve the LS problem with `house`.

QR decomposition using Householder transformations

```
function [A,p] = house(A)
[m,n]=size(A); p = zeros(1,n);
for k = 1:n
    z = A(k:m,k);
    e1 = [1; zeros(m-k,1)];
    u = z+sign(z(1))*norm(z)*e1; u = u/norm(u);
    % update nonzero part of A by I - 2uu^T
    A(k:m,k:n) = A(k:m,k:n)-2*u*(u'*A(k:m,k:n));
    % store u
    p(k) = u(1);
    A(k+1:m,k) = u(2:m-k+1);
end
```

See text for code to solve the LS problem with `house`.

Methods for solving linear least-squares problems

1. Normal equations: fast, simple, intuitive, but less robust in ill-conditioned situations.
2. QR decomposition: this is the “standard” approach implemented in general purpose software. It is more computationally expensive than the normal equations approach if $m \gg n$, but is more robust.
3. SVD: used mostly when A is rank deficient or nearly rank deficient (in which case the QR approach may not be sufficiently robust). Significantly more expensive in general, and cannot be adapted to deal efficiently with sparse matrices. See Chapter 8.