Least squares method: least squares

Ramani Duraiswami,
Dept. of Computer Science

Least Squares –Lecture 15 Recap
• We wish to solve

\[ \mathbf{A} \mathbf{c} = \mathbf{y} \]

\( \mathbf{A} \) is a \( m \times n \) matrix, \( \mathbf{c} \) is a \( n \) vector, and \( \mathbf{y} \) is a \( m \) vector
• Number of equations and unknowns may not match
• Look for solution \( \mathbf{c} \) that minimizes same cost function
  – Sum of squares of residuals
• Associated with each data point \( x_i \) is a residual \( r_i \)
• Define cost function: \( F(\mathbf{c}) = \| \mathbf{A} \mathbf{c} - \mathbf{y} \|_2^2 \)

\[
F(\mathbf{c}) = \sum_{i=1}^{m} r_i^2 = \sum_{i=1}^{m} \left( \sum_{j=1}^{n} A_{ij} c_j - y_i \right)^2 = \sum_{i=1}^{m} \left( \sum_{j=1}^{n} A_{ij} c_j - y_i \right) \left( \sum_{k=1}^{n} A_{ik} c_k - y_i \right)
\]
Solution 1: Normal Equations – Lecture 15

• Differentiate cost function and set to zero
\[
\frac{\partial F}{\partial c_i} = \sum_{j=1}^{n} \left( \sum_{k=1}^{n} A_{kj} \frac{\partial c_k}{\partial c_i} \right) + \left( \sum_{j=1}^{n} A_{ji} \frac{\partial c_j}{\partial c_i} \right) + \left( \sum_{k=1}^{n} A_{ik} \frac{\partial c_k}{\partial c_i} \right) = 0
\]

• Leads to:
\[
2\left( A^t A c - A^t y \right) = 0 \quad \text{or} \quad [A^t A]c = A^t y
\]

• Can be solved via LU decomposition of \([A^t A]\)
• However is ill-conditioned and expensive
• Both arise because we compute the matrix \([A^t A]\)

Solution 2: QR decomposition : Lecture 16

• QR decomposition of \(A\)
\[
A_{m \times n} = Q'_{m \times m} R'_{m \times n}
\]

\[
\begin{bmatrix}
A_{11} & \cdots & A_{1n} \\
\vdots & \ddots & \vdots \\
A_{m1} & \cdots & A_{mn}
\end{bmatrix} =
\begin{bmatrix}
Q_{11}' & \cdots & Q_{1n}' & \cdots & Q_{1m}' \\
\vdots & \ddots & \vdots \\
Q_{m1}' & \cdots & Q_{mn}' & \cdots & Q_{mn}'
\end{bmatrix}
\begin{bmatrix}
R_{11}' & \cdots & R_{1n}' \\
0 & \ddots & \vdots \\
0 & 0 & R_{nn}'
\end{bmatrix}
\]

• Note the multiplication with zero terms. So the “economy size” version of this decomposition is
\[
\begin{bmatrix}
A_{11} & \cdots & A_{1n} \\
\vdots & \ddots & \vdots \\
A_{m1} & \cdots & A_{mn}
\end{bmatrix} =
\begin{bmatrix}
Q_{11}' & \cdots & Q_{1n}' \\
\vdots & \ddots & \vdots \\
Q_{m1}' & \cdots & Q_{mn}'
\end{bmatrix}
\begin{bmatrix}
R_{11}' & \cdots & R_{1n}' \\
0 & \ddots & \vdots \\
0 & 0 & R_{nn}'
\end{bmatrix}
\]

\[
A_{m \times n} = Q_{m \times n} R_{n \times n}
\]
Least Squares via QR: Lecture 16

• Minimizing $||r||_2$ is the same as minimizing $||Q^T r||_2$

$$Q^T r = Q^T y - Q^T A c = b - Q^T Q'R'c = b - R'c$$

• So we wish to minimize $||b - R'c||_2$

• Split $b$ in to two pieces
  - $b_1$ of dimension $n$
  - $b_2$ of dimension $m-n$

$$||r||_2 = ||b_1 - Rc||_2 + ||b_2 - 0c||_2$$

• So no matter what $c$ is the second term remains unchanged

• Solve least squares by solving triangular system

$$Rc = b_1$$

Today: Computing the factorization

• QR is useful … so how do we factorize a matrix $A$?

• In LU we reduced a square matrix $A$ to a upper triangular matrix $U$ by
  - adding multiples of other rows so elements in a given column below the row were zeroed out
  - multipliers were stored in $L$ which gave us $A=LU$

• Here we want to do the same for a rectangular matrix
  - zero out entries below the diagonal
  - but do it with orthogonal matrices

• Today: Givens Rotations

• Zero out one specific entry of a column at a time
  - Use 2D rotations
Rotation

A 2 × 2 rotation matrix is of the form

\[ A = \begin{bmatrix} \cos(t) & -\sin(t) \\ \sin(t) & \cos(t) \end{bmatrix}, \]

and has determinant 1:

- \( A \) is a 2 × 2 orthogonal matrix

Reflection

An example of a 2 × 2 reflection matrix, reflecting about the \( y \) axis, is

\[ A = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \]

which has determinant -1:

- Reflection is an orthogonal matrix
Permutation = Reflection

Another example of a reflection is a permutation matrix:

\[ A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \]

which has determinant -1:

- This reflection is about the 45° line \( x = y \).

To compute QR

- Perform a sequence of orthogonal transformations that zero out elements
- Orthogonal transformations can be rotations or reflections or combinations
- Givens Rotation:
- Givens matrix has elements
- \( c^2 + s^2 = 1 \)
• How do we use a rotation to zero out an element?

• Let \( \mathbf{z} = 
\begin{bmatrix}
    z_1 \\
    z_2 
\end{bmatrix}
\)

• We want \( G\mathbf{z} = 
\begin{bmatrix}
    cz_1 + s z_2 \\
    sz_1 - cz_2 
\end{bmatrix} = x e_1 \)

\[ c = z_1 / x \]

\[ s = z_2 / x, \quad \text{and} \quad z_i^2 + z_j^2 = x^2 \]

**Givens QR**

• To apply idea to larger matrix, embed the Givens matrix in identity matrix. We will use the notation \( G_{ij} \) to denote an \( n \times n \) identity matrix with its \( i \)th and \( j \)th rows modified to include the Givens rotation: for example, if \( n = 6 \), then

\[
G_{25} = 
\begin{bmatrix}
    1 & 0 & 0 & 0 & 0 & 0 \\
    0 & c & 0 & 0 & s & 0 \\
    0 & 0 & 1 & 0 & 0 & 0 \\
    0 & 0 & 0 & 1 & 0 & 0 \\
    0 & s & 0 & 0 & -c & 0 \\
    0 & 0 & 0 & 0 & 0 & 1 
\end{bmatrix}
\]

and multiplication of a vector by this matrix leaves all but rows 2 and 5 of the vector unchanged.

This matrix zeroes out the (5,2) element of a matrix.

• The matrix \( G_{ij} \) works on the \((j,i)\) element of the matrix
Algorithm

- Algorithm
  
  for $i=1, \ldots, n$
  for $j=i+1, \ldots, m$
    // Find Givens matrix $G_{ij}$ to zero out $j,i$ element of $A$
    // using the the value at position $(i,i)$
    // compute $x, c, s$
    // embed them in the identity
    $A = G_{ij}A$
  end
end