Lecture 18

Splines (finish)
Start Least Squares

Owen L. Lewis

Department of Mathematics and Statistics
University of New Mexico

October 29, 2024

Natural cubic spline

"Center" each cubic at its own left endpoint

$$S_i(x) = a_i + b_i(x - t_i) + c_i(x - t_i)^2 + d_i(x - t_i)^3.$$

1. Match data to knots.
2. Differentiate S_i and match derivative at "inner" knots.
3. Differentiate again and match second derivative at "inner" knots.
4. Much algebra.

Keep the count

For $n+1$ knots enumerated t_i, $i = 0, 1, \ldots n$.
We have n sub-intervals $[t_i, t_i + 1]$, $i = 0, 1, \ldots n-1$.
Each gets its own cubic.

$$S_i(x) = a_i + b_i(x - t_i) + c_i(x - t_i)^2 + d_i(x - t_i)^3.$$

Natural cubic spline: Step 1

Getting Value Correct

At the left endpoints:

$$S_i(t_i) = y_i$$

$$\Rightarrow a_i + b_i(t_i - t_i) + c_i(t_i - t_i)^2 + d_i(t_i - t_i)^3 = y_i$$

$$\Rightarrow a_i = y_i.$$

For $i = 0, 1, \ldots n - 1$.

We know the a_i's.

Natural cubic spline: Step 1

Getting Value Correct

At the right endpoints:

$$S_i(t_{i+1}) = y_{i+1}$$

$$\Rightarrow y_i + b_i(t_{i+1} - t_i) + c_i(t_{i+1} - t_i)^2 + d_i(t_{i+1} - t_i)^3 = y_{i+1}$$

$$\Rightarrow b_i\delta_i + c_i\delta_i^2 + d_i\delta_i^3 = \Delta_i \tag{1}$$

For $i = 0, 1, \ldots n-1$.

Where we've defined $\delta_i = t_{i+1} - t_i$ and $\Delta_i = y_{i+1} - y_i$.

Natural cubic spline: Step 2

Getting the derivatives right

$S_i'(x)$ is a parabola

$$S_i'(x) = b_i + 2c_i(x - t_i) + 3d_i(x - t_i)^2.$$

Match derivative where sub-intervals meet:

$$S_i'(t_{i+1}) = S_{i+1}'(t_{i+1})$$

$$\Rightarrow b_i + 2c_i(t_{i+1} - t_i) + 3d_i(t_{i+1} - t_i)^2 = b_{i+1} + 2c_{i+1}(t_{i+1} - t_{i+1}) + 3d_{i+1}(t_{i+1} - t_{i+1})^2$$

$$\Rightarrow b_i + 2c_i\delta_i + 3d_i\delta_i^2 = b_{i+1}. \tag{2}$$

For $i = 0, 1, \ldots n - 2$.

Natural cubic spline: Step 3

Getting the convexity right

$S_i''(x)$ is a line

$$S_i''(x) = 2c_i + 6d_i(x - t_i).$$

Match second derivative where sub-intervals meet:

$$S_i''(t_{i+1}) = S_{i+1}''(t_{i+1})$$

$$\Rightarrow 2c_i + 6d_i(t_{i+1} - t_i) = 2c_{i+1} + 6d_{i+1}(t_{i+1} - t_{i+1})$$

$$\Rightarrow 2c_i + 6d_i\delta_i = 2c_{i+1}. \tag{3}$$

For $i = 0, 1, \ldots n - 2$.

Natural cubic spline: Step 4

Time for some algebra

From Equation 3, we can say

$$d_i = \frac{c_{i+1} - c_i}{3\delta_i}. \tag{4}$$

From Equation 1, we get

$$b_i = \frac{\Delta_i}{\delta_i} - c_i\delta_i - d_i\delta_i^2.$$

Combining with Equation 4 gives

$$b_i = \frac{\Delta_i}{\delta_i} - \frac{\delta_i}{3}(2c_i + c_{i+1}). \tag{5}$$

Natural cubic spline: Step 4

Time for some algebra

Finally, combining Equations 2, 4, and 5 gives

$$\delta_i c_i + 2(\delta_i + \delta_{i+1})c_{i+1} + \delta_{i+1} c_{i+2} = 3\left(\frac{\Delta_{i+1}}{\delta_{i+1}} - \frac{\Delta_i}{\delta_i}\right) \tag{6}$$

for $i = 0, 1, \ldots n-2$.
This seems complex, write it out...

$$\delta_0 c_0 + 2(\delta_0 + \delta_1)c_1 + \delta_1 c_2 = 3\left(\frac{\Delta_1}{\delta_1} - \frac{\Delta_0}{\delta_0}\right)$$

$$\vdots$$

$$\delta_{n-2} c_{n-2} + 2(\delta_{n-2} + \delta_{n-1})c_{n-1} + \delta_{n-1} c_n = 3\left(\frac{\Delta_{n-1}}{\delta_{n-1}} - \frac{\Delta_{n-2}}{\delta_{n-2}}\right)$$

Natural cubic spline: Step 4

Time for some algebra

We have $n-2$ equations, we need more.

Zero derivative at the very left:

$$S_0''(t_0) = 0,$$

$$\Rightarrow 2c_0 = 0.$$

Zero derivative at the very right. Define

$$S''(t_n) = 2c_n = 0.$$

Natural cubic spline: Step 4

One big matrix

$$\begin{bmatrix} 1 & 0 & 0 & & & & \\ \delta_0 & 2(\delta_0+\delta_1) & \delta_1 & \ddots & & & \\ 0 & \delta_1 & 2(\delta_1+\delta_2) & \delta_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \ddots & & \\ & & & \delta_{n-2} & 2(\delta_{n-2}+\delta_{n-1}) & \delta_{n-1} \\ & & & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} 0 \\ 3\left(\frac{\Delta_1}{\delta_1} - \frac{\Delta_0}{\delta_0}\right) \\ \vdots \\ 3\left(\frac{\Delta_{n-1}}{\delta_{n-1}} - \frac{\Delta_{n-2}}{\delta_{n-2}}\right) \\ 0 \end{bmatrix} \tag{7}$$

Note: we don't need c_n for anything, it is just a convenience to make the matrix pattern nice.

Finding the spline

See Chapter 3.4 (page 176 in first edition)

```
1  input t, y vectors of data
2  calculate δ and Δ.
3  form right-hand-side vector b and matrix L
4  c = L\b
5  for i = 0, 1, ... n − 1
6          b_i = Δ_i/δ_i − δ_i(2c_i + c_{i+1})/3
7          d_i = (c_{i+1} + c_i)/(3δ_i)
8  end
9  S_i(x) = y_i + b_i(x − t_i) + c_i(x − t_i)^2 + d_i(x − t_i)^3
```

Notes

Example

Find the natural cubic spline for

x	-1	0	1
y	1	2	-1

1. Determine Δ_i, δ_i

$$\Delta = \begin{bmatrix} 1 \\ -3 \end{bmatrix} \quad \delta = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

2. Solve

$$\begin{bmatrix} 1 & & \\ 1 & 4 & 1 \\ & & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -12 \\ 0 \end{bmatrix}$$

3. Result:

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -3 \\ 0 \end{bmatrix}$$

Example

Find the natural cubic spline for

x	-1	0	1
y	1	2	-1

4. Calculate b's and d's

$$b_0 = \Delta_0/\delta_0 - \delta_0(2c_0 + c_1)/3 = 2, \quad b_1 = -1$$

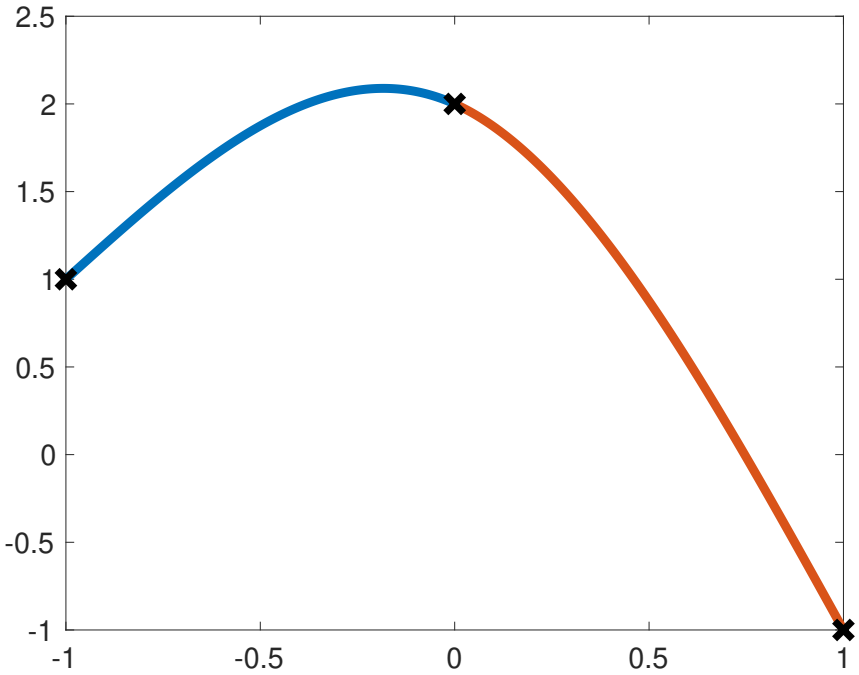
$$d_0 = (c_1 + c_0)/(3\delta_0) = -1, \quad d_1 = 1$$

5.

$$S(x) = \begin{cases} 1 + 2(x+1) - (x+1)^3 & -1 \leqslant x < 0 \\ 2 - x - 3x^2 + x^3 & 0 \leqslant x < 1 \end{cases}$$

Example

$$S(x) = \begin{cases} 1 + 2(x+1) - (x+1)^3 & -1 \leqslant x < 0 \\ 2 - x - 3x^2 + x^3 & 0 \leqslant x < 1 \end{cases}$$



Changing gears a bit

What if we don't want to fit the data perfectly? What if we suspect there is error/noise in the data?

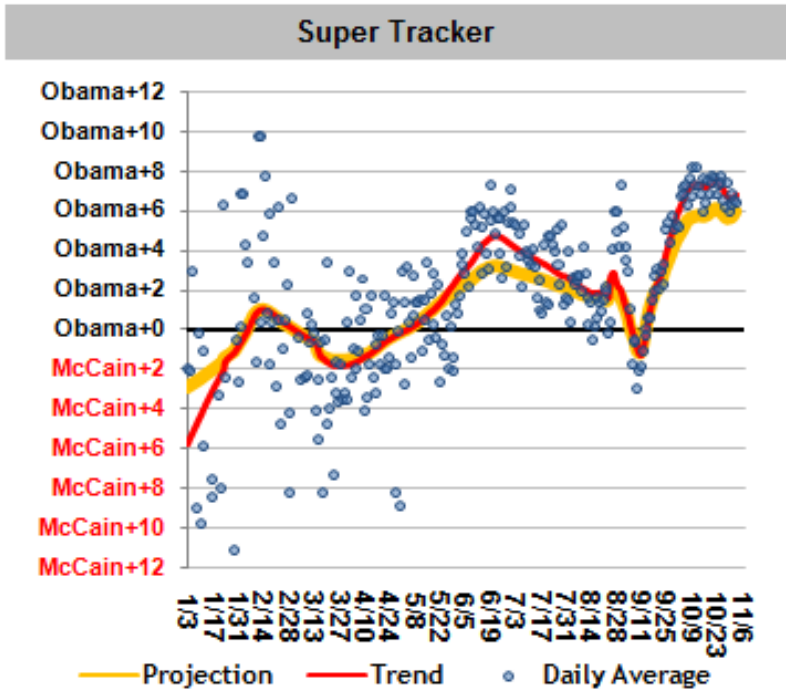
If we have 3000 data points, do we *really* think the underlying explanation is a 2999 degree polynomial?

If we use a spline interpolant, how do we project what will happen outside our data?

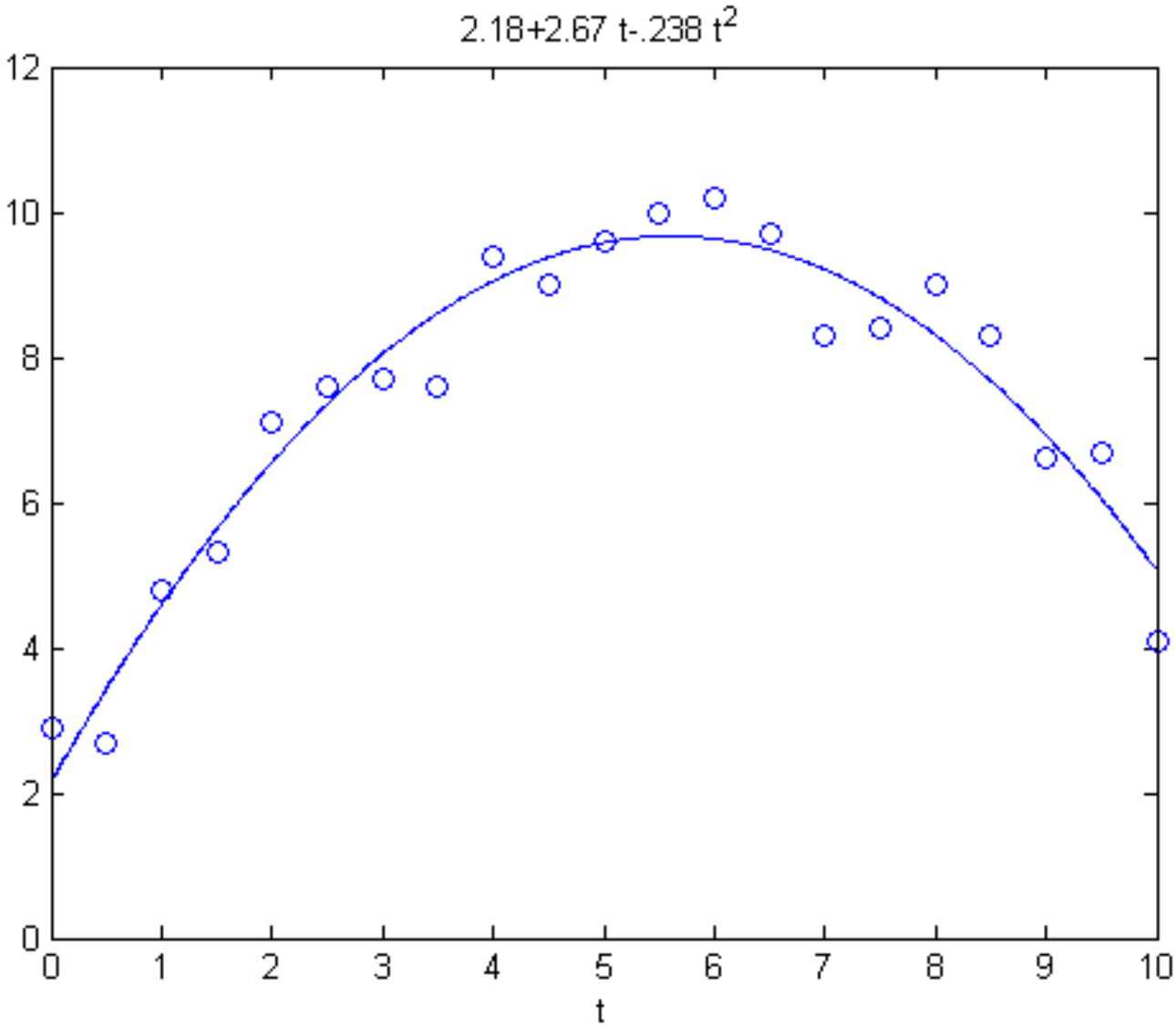
Polling data

Recall in interpolation we wanted to find a curve that went through *all* of the data points.

Suppose we are given the data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ and we want to find a curve that *best fits* the data.



Fitting curves



Fitting a line

Given n data points $\{(x_1, y_1), ..., (x_n, y_n)\}$ find a and b such that

$$y_i = ax_i + b \quad i = 1, 2, \ldots n.$$

In matrix form, find a and b that solves

$$\begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Systems with more equations than unknowns are called **overdetermined**

Overdetermined Systems

If A is an $m \times n$ matrix, then in general, an $m \times 1$ vector b may not lie in the column space of A. Hence $Ax = b$ may not have an exact solution.

Definition

The **residual** vector is

$$r = b - Ax.$$

The **least squares** solution is given by minimizing the square of the residual in the 2-norm.

Notes

Normal equations

Writing $r = (b - Ax)$ and substituting, we want to find an x that minimizes the following function

$$\phi(x) = \|r\|_2^2 = r^T r = (b - Ax)^T (b - Ax) = b^T b - 2x^T A^T b + x^T A^T Ax$$

From calculus we know that the minimizer occurs where $\nabla\phi(x) = 0$.

The derivative is given by

$$\nabla\phi(x) = -2A^T b + 2A^T Ax = 0$$

Definition

The system of **normal equations** is given by

$$A^T Ax = A^T b.$$

Example

Find in a line that “best” fits the following data:

x	-1	0	1
y	1	2	-1

We want to “solve” this matrix equation:

$$\begin{bmatrix} -1 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}$$

Example

We form the normal equation

$$\begin{bmatrix} -1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}$$

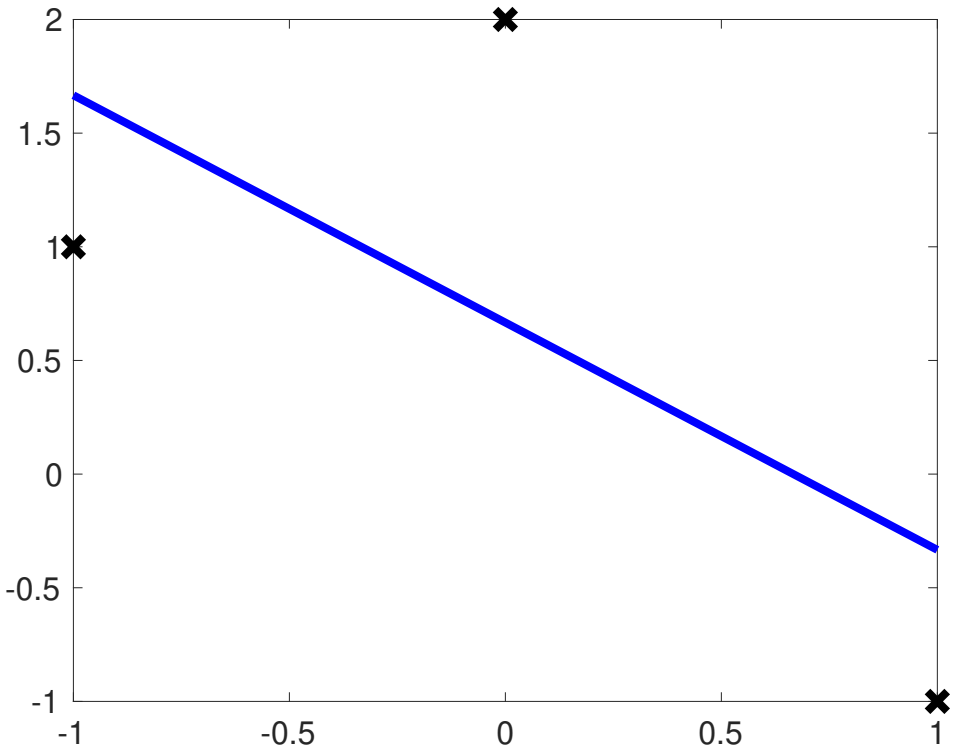
$$\Rightarrow \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$$

$$\Rightarrow a = -1, \quad b = 2/3.$$

Here is our line:

$$y = -x + \frac{2}{3}.$$

Example



Solving normal equations

Since the normal equations forms a symmetric positive definite system, we can solve by computing the Cholesky factorization

$$A^T A = LL^T$$

and solving $Ly = A^T b$ and $L^T x = y$.

Consider

$$A = \begin{bmatrix} 1 & 1 \\ \epsilon & 0 \\ 0 & \epsilon \end{bmatrix}$$

where $0 < \epsilon < \sqrt{\epsilon_{mach}}$. The normal equations for this system is given by

$$A^T A = \begin{bmatrix} 1+\epsilon^2 & 1 \\ 1 & 1+\epsilon^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \text{ SINGULAR!}$$

Normal equations: conditioning

The normal equations tend to worsen the condition of the matrix.

Theorem

$$cond(A^T A) = (cond(A))^2$$

```
>> A = rand(10,10);
>> cond(A)
   43.4237
>> cond(A'*A)
   1.8856e+03
```

How can we solve the least squares problem without squaring the condition of the matrix?

Other approaches

- QR factorization.
  - For $A \in \mathbb{R}^{m \times n}$, factor $A = QR$ where
    - Q is an $m \times m$ orthogonal matrix
    - R is an $m \times n$ upper triangular matrix (since R is an $m \times n$ upper triangular matrix we can write $R = \begin{bmatrix} R' \\ 0 \end{bmatrix}$ where R is $n \times n$ upper triangular and 0 is the $(m-n) \times n$ matrix of zeros)

- SVD - singular value decomposition
  - For $A \in \mathbb{R}^{m \times n}$, factor $A = USV^T$ where
    - U is an $m \times m$ orthogonal matrix
    - V is an $n \times n$ orthogonal matrix
    - S is an $m \times n$ diagonal matrix whose elements are the singular values.

Orthogonal matrices

Definition

A matrix Q is orthogonal if

$$Q^T Q = Q Q^T = I$$

Orthogonal matrices preserve the Euclidean norm of any vector v,

$$\|Qv\|_2^2 = (Qv)^T (Qv) = v^T Q^T Q v = v^T v = \|v\|_2^2.$$

Also, orthogonal matrices are easy to invert . . . simply a multiplication by $Q^T \Rightarrow \mathcal{O}(n^2)$.

Using QR factorization for least squares

Now that we know orthogonal matrices preserve the Euclidean norm, we can apply orthogonal matrices to the residual vector without changing the norm of the residual.

$$\|r\|_2^2 = \|b - Ax\|_2^2 = \left\| b - Q \begin{bmatrix} R' \\ 0 \end{bmatrix} x \right\|_2^2 = \left\| Q^T b - Q^T Q \begin{bmatrix} R' \\ 0 \end{bmatrix} x \right\|_2^2$$

$$= \left\| Q^T b - \begin{bmatrix} R' \\ 0 \end{bmatrix} x \right\|_2^2 .$$

If $Q^T b = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$ then

$$\left\| Q^T b - \begin{bmatrix} R' \\ 0 \end{bmatrix} x \right\|_2^2 = \left\| \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} - \begin{bmatrix} R'x \\ 0 \end{bmatrix} \right\|_2^2 = \left\| \begin{bmatrix} c_1 - R'x \\ c_2 \end{bmatrix} \right\|_2^2 = \|c_1 - R'x\|_2^2 + \|c_2\|_2^2$$

Hence the least squares solution is given by solving $Rx = c_1$. We can solve $Rx = c$ using back substitution and the residual is $\|r\|_2 = \|c_2\|_2$.

Gram-Schmidt orthogonalization

One way to obtain the QR factorization of a matrix A is by Gram-Schmidt orthogonalization.

For each column of A, subtract out its component in the other columns.

For the simple case of 2 vectors $\{a_1, a_2\}$, first normalize a_1 and obtain

$$q_1 = \frac{a_1}{\|a_1\|} = \frac{a_1}{r_{11}}.$$

Now subtract from a_2 the components from q_1. This is given by

$$a_2' = a_2 - (q_1^T a_2) q_1 = a_2 - r_{12} q_1.$$

Normalizing a_2' we have

$$q_2 = \frac{a_2'}{\|a_2'\|} = \frac{a_2'}{r_{22}}$$

Repeating this idea for n columns gives us Gram-Schmidt orthogonalization.

Notes

Gram-Schmidt orthogonalization

Since R is upper triangular and $A = QR$ we have

$$\begin{aligned} a_1 &= q_1 r_{11} \\ a_2 &= q_1 r_{12} + q_2 r_{22} \\ \vdots &= \quad \vdots \\ a_n &= q_1 r_{1n} + q_2 r_{2n} + \ldots + q_n r_{nn} \end{aligned}$$

From this we see that $r_{ij} = q_i^T a_j, j > i$.
$\Rightarrow$ Simply multiply any of the above lines by q_i^T

Thus, we know now how to compute each entry of R and form Q with Gram-Schmidt

Gram-Schmidt orthogonalization

```
function [Q,R] = gs_qr (A)

m = size(A,1);
n = size(A,2);

for i = 1:n
    R(i,i) = norm(A(:,i),2);
    Q(:,i) = A(:,i)./R(i,i);
    for j = i+1:n
        R(i,j) = Q(:,i)' * A(:,j);
        A(:,j) = A(:,j) - R(i,j)*Q(:,i);
    end
end

end
```