STAT 309: MATHEMATICAL COMPUTATIONS I
FALL 2015
LECTURE 11

1. least squares with linear constraints

- suppose that we wish to fit data as in the least squares problem, except that we are using different functions to fit the data on different subintervals
- a common example is the process of fitting data using cubic splines, with a different cubic polynomial approximating data on each subinterval
- typically it is desired that the functions assigned to each piece form a function that is continuous on the entire interval within which the data lies
- this requires that *constraints* be imposed on the functions themselves
- it is also not uncommon to require that the function assembled from these pieces also has a continuous first or even second derivative, resulting in additional constraints
- the result is a *least squares problem with linear constraints*, as the constraints are applied to coefficients of predetermined functions chosen as a basis for some function space, such as the space of polynomials of a given degree
- the general form of a least squares problem with linear constraints is as follows: we wish to find an $\mathbf{x} \in \mathbb{R}^n$ that minimizes $\|A\mathbf{x} - \mathbf{b}\|_2$, subject to the constraint $C^{\mathsf{T}}\mathbf{x} = \mathbf{d}$, where $A \in \mathbb{R}^{m\times n}$, $C \in \mathbb{R}^{n\times p}$, $\mathbf{b} \in \mathbb{R}^m$, and $\mathbf{d} \in \mathbb{R}^p$ are given

$$\begin{array}{ll} \text{minimize} & \|\mathbf{b} - A\mathbf{x}\|_2^2 \\ \text{subject to} & C^{\mathsf{T}}\mathbf{x} = \mathbf{d} \end{array} \tag{1.1}$$

- again we will describe three methods, mathematically equivalent but with different numerical properties
- this problem is usually solved using *Lagrange multipliers*, define

$$L(\mathbf{x}, \boldsymbol{\lambda}) = \|\mathbf{b} - A\mathbf{x}\|_2^2 + 2\boldsymbol{\lambda}^{\mathsf{T}}(C^{\mathsf{T}}\mathbf{x} - \mathbf{d})$$

- in optimization parlance, the function L is called the *Lagrangian* and $\boldsymbol{\lambda}^{\mathsf{T}} = [\lambda_1, \dots, \lambda_p]^{\mathsf{T}}$ is the vector of Lagrange multipliers
- setting derivative with respect to $\mathbf{x}$ to zero yields

$$\mathbf{0} = \nabla_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\lambda}) = 2(A^{\mathsf{T}}A\mathbf{x} - A^{\mathsf{T}}\mathbf{b} + C\boldsymbol{\lambda})$$

  and so

$$A^{\mathsf{T}}A\mathbf{x} + C\boldsymbol{\lambda} = A^{\mathsf{T}}\mathbf{b} \tag{1.2}$$

- note that $\mathbf{0} = \nabla_{\boldsymbol{\lambda}} L(\mathbf{x}, \boldsymbol{\lambda})$ just gives us back the constraint

$$C^{\mathsf{T}}\mathbf{x} = \mathbf{d} \tag{1.3}$$

- in optimization parlance, (1.2) and (1.3) are collectively called the *KKT conditions*
- method 1: together (1.2) and (1.3) give the system

$$\begin{bmatrix} A^{\mathsf{T}}A & C \\ C^{\mathsf{T}} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} A^{\mathsf{T}}\mathbf{b} \\ \mathbf{d} \end{bmatrix}$$

- solving this linear system gives us a solution to (1.1)

Date: November 3, 2015, version 1.0. Comments, bug reports: `lekheng@galton.uchicago.edu`.

- this method preserves the sparsity of C but involves a coefficient matrix of size $(n+p) \times (n+p)$, larger than the next two methods
- method 2: if A has full column rank, then from $A^\mathsf{T}A\mathbf{x} = A^\mathsf{T}\mathbf{b} - C\boldsymbol{\lambda}$, we see that we can first compute $\mathbf{x} = \widehat{\mathbf{x}} - (A^\mathsf{T}A)^{-1}C\boldsymbol{\lambda}$ where $\widehat{\mathbf{x}}$ is the solution to the unconstrained least squares problem

$$\widehat{\mathbf{x}} \in \operatorname{argmin} \|A\mathbf{x} - \mathbf{b}\|_2$$

- then from the equation $C^\mathsf{T}\mathbf{x} = \mathbf{d}$ we obtain the $p \times p$ linear system

$$C^\mathsf{T}(A^\mathsf{T}A)^{-1}C\boldsymbol{\lambda} = C^\mathsf{T}\widehat{\mathbf{x}} - \mathbf{d} \tag{1.4}$$

  which we can then solve for $\boldsymbol{\lambda}$
- this works because $A^\mathsf{T}A\widehat{\mathbf{x}} = A^\mathsf{T}\mathbf{b}$ and therefore

$$A^\mathsf{T}A\mathbf{x} = A^\mathsf{T}\mathbf{b} - C\boldsymbol{\lambda}$$

- the actual algorithm uses two QR factorization and does not actually require solving a system involving $A^\mathsf{T}A$
  - compute full-rank QR

$$A = Q\begin{bmatrix} R \\ 0 \end{bmatrix}$$

    with nonsingular $R \in \mathbb{R}^{n\times n}$
  - solve the unconstrained least squares problem $\min\|A\mathbf{x} - \mathbf{b}\|_2$ for $\widehat{\mathbf{x}}$
  - form $W = R^{-\mathsf{T}}C$
  - compute $W = Q_1R_1$, the QR factorization of W
  - solve $R_1^\mathsf{T}R_1\boldsymbol{\lambda} = \boldsymbol{\eta} = C^\mathsf{T}\widehat{\mathbf{x}} - \mathbf{d}$ for $\boldsymbol{\lambda}$
  - set $\mathbf{x} = \widehat{\mathbf{x}} - (A^\mathsf{T}A)^{-1}C\boldsymbol{\lambda}$
- this works because

$$\begin{aligned} R_1^\mathsf{T}R_1 &= (Q_1^\mathsf{T}W)^\mathsf{T}(Q_1^\mathsf{T}W) \\ &= W^\mathsf{T}Q_1Q_1^\mathsf{T}W \\ &= C^\mathsf{T}R^{-1}R^{-\mathsf{T}}C \\ &= C^\mathsf{T}(R^\mathsf{T}R)^{-1}C \\ &= C^\mathsf{T}(R^\mathsf{T}Q^\mathsf{T}QR)^{-1}C \\ &= C^\mathsf{T}(A^\mathsf{T}A)^{-1}C \end{aligned}$$

- method 2 is not the most practical since it has more unknowns than the unconstrained least squares problem, which is odd because the constraints should have the effect of eliminating unknowns, not adding them
- method 3: suppose $p \le n$, then computing the QR factorization of C yields

$$C = Q_2\begin{bmatrix} R_2 \\ 0 \end{bmatrix}$$

  where R_2 is a $p \times p$ upper triangular matrix
- then the constraint $C^\mathsf{T}\mathbf{x} = \mathbf{d}$ takes the form

$$R_2^\mathsf{T}\mathbf{u} = \mathbf{d}, \quad Q_2^\mathsf{T}\mathbf{x} = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}$$

- then

$$\begin{aligned}\|\mathbf{b} - A\mathbf{x}\|_2 &= \|\mathbf{b} - AQ_2Q_2^{\mathsf{T}}\mathbf{x}\|_2 \\ &= \left\|\mathbf{b} - \widetilde{A}\begin{bmatrix}\mathbf{u}\\ \mathbf{v}\end{bmatrix}\right\|_2, \qquad \widetilde{A} = AQ_2 \\ &= \left\|\mathbf{b} - \begin{bmatrix}\widetilde{A}_1 & \widetilde{A}_2\end{bmatrix}\begin{bmatrix}\mathbf{u}\\ \mathbf{v}\end{bmatrix}\right\|_2 \\ &= \|\mathbf{b} - \widetilde{A}_1\mathbf{u} - \widetilde{A}_2\mathbf{v}\|_2\end{aligned}$$

- thus we can obtain $\mathbf{x}$ by the following algorithm:
  - compute the QR factorization of C
  - compute $\widetilde{A} = AQ_2$
  - solve $R_2^{\mathsf{T}}\mathbf{u} = \mathbf{d}$
  - solve the new least squares problem of minimizing $\|(\mathbf{b} - \widetilde{A}_1\mathbf{u}) - \widetilde{A}_2\mathbf{v}\|_2$
  - compute

$$\mathbf{x} = Q_2\begin{bmatrix}\mathbf{u}\\ \mathbf{v}\end{bmatrix}$$

- method 3 has the advantage that there are fewer unknowns in each system that needs to be solved, and also that $\kappa_2(\widetilde{A}_2) \leq \kappa_2(A)$
- the drawback is that sparsity can be destroyed

2. computing the QR factorization

- there are two common ways to compute the QR decomposition:
  - using *Householder matrices*, developed by Alston S. Householder
  - using *Givens rotations*, also known as *Jacobi rotations*, used by Wallace Givens and originally invented by Jacobi for use with in solving the symmetric eigenvalue problem in 1846
  - the Gram–Schmidt or modfied Gram–Schmidt orthogonalization discussed in previous lecture works in principle but has numerical stability issues and are not usually used
- roughly speaking, Gram–Schmidt applies a sequence of triangular matrices to orthogonalize A (i.e., transform A into an orthogonal matrix Q),

$$AR_1^{-1}R_2^{-1}\cdots R_{n-1}^{-1} = Q$$

  whereas Householder and Givens QR apply a sequence of orthogonal matrices to triangularize A (i.e., transform A into an upper triangular matrix R),

$$Q_{n-1}^{\mathsf{T}}\cdots Q_2^{\mathsf{T}}Q_1^{\mathsf{T}}A = R$$

- orthogonal transformations are highly desirable in algorithms as they preserve lengths and therefore do not blow up the errors present at every stage of the computation

3. orthogonalization using Givens rotations

- we illustrate the process in the case where A is a 2×2 matrix
- in Gaussian elimination, we compute $L^{-1}A = U$ where L^{-1} is unit lower triangular and U is upper triangular, specifically,

$$\begin{bmatrix}1 & 0\\ m_{21} & 1\end{bmatrix}\begin{bmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{bmatrix} = \begin{bmatrix}a_{11}^{(2)} & a_{12}^{(2)}\\ 0 & a_{22}^{(2)}\end{bmatrix}, \quad m_{21} = -\frac{a_{21}}{a_{11}}$$

- by contrast, the QR decomposition takes the form

$$\begin{bmatrix} \gamma & \sigma \\ -\sigma & \gamma \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{bmatrix}$$

  where $\gamma^2 + \sigma^2 = 1$
- from the relationship $-\sigma a_{11} + \gamma a_{21} = 0$ we obtain

$$\begin{aligned} \gamma a_{21} &= \sigma a_{11} \\ \gamma^2 a_{21}^2 &= \sigma^2 a_{11}^2 = (1 - \gamma^2) a_{11}^2 \end{aligned}$$

  which yields

$$\gamma = \pm \frac{a_{11}}{\sqrt{a_{21}^2 + a_{11}^2}}$$

- it is conventional to choose the + sign
- then, we obtain

$$\sigma^2 = 1 - \gamma^2 = 1 - \frac{a_{11}^2}{a_{21}^2 + a_{11}^2} = \frac{a_{21}^2}{a_{21}^2 + a_{11}^2},$$

  or

$$\sigma = \pm \frac{a_{21}}{\sqrt{a_{21}^2 + a_{11}^2}}$$

- again, we choose the + sign
- as a result, we have

$$r_{11} = a_{11} \frac{a_{11}}{\sqrt{a_{21}^2 + a_{11}^2}} + a_{21} \frac{a_{21}}{\sqrt{a_{21}^2 + a_{11}^2}} = \sqrt{a_{21}^2 + a_{11}^2}$$

- the matrix

$$Q^{\mathsf{T}} = \begin{bmatrix} \gamma & \sigma \\ -\sigma & \gamma \end{bmatrix}$$

  is called a *Givens rotation*
- it is called a rotation because it is orthogonal, and therefore length-preserving, and also because there is an angle θ such that $\sin\theta = \sigma$ and $\cos\theta = \gamma$, and its effect is to rotate a vector through the angle θ
- in particular,

$$\begin{bmatrix} \gamma & \sigma \\ -\sigma & \gamma \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \rho \\ 0 \end{bmatrix}$$

  where $\rho = \sqrt{\alpha^2 + \beta^2}$, $\alpha = \rho\cos\theta$ and $\beta = \rho\sin\theta$
- it is easy to verify that the product of two rotations is itself a rotation
- now, in the case where A is an $n \times n$ matrix, suppose that we are given the vector

$$\begin{bmatrix} \times & \cdots & \times & \alpha & \times & \cdots & \times & \beta & \times & \cdots & \times \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^n,$$

then

$$\begin{bmatrix} 1 & & & & & & & & & & \\ & \ddots & & & & & & & & & \\ & & 1 & & & & & & & & \\ & & & \gamma & & & & \sigma & & & \\ & & & & 1 & & & & & & \\ & & & & & \ddots & & & & & \\ & & & & & & 1 & & & & \\ & & & -\sigma & & & & \gamma & & & \\ & & & & & & & & 1 & & \\ & & & & & & & & & \ddots & \\ & & & & & & & & & & 1 \end{bmatrix} \begin{bmatrix} \times \\ \vdots \\ \times \\ \alpha \\ \times \\ \vdots \\ \times \\ \beta \\ \times \\ \vdots \\ \times \end{bmatrix} = \begin{bmatrix} \times \\ \vdots \\ \times \\ \rho \\ \times \\ \vdots \\ \times \\ 0 \\ \times \\ \vdots \\ \times \end{bmatrix}$$

- so, in order to transform A into an upper triangular matrix R, we can find a product of rotations Q such that $Q^{\mathsf{T}}A = R$
- it is easy to see that $O(n^2)$ rotations are required