STAT 309: MATHEMATICAL COMPUTATIONS I
FALL 2019
LECTURE 10

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 $\boldsymbol{\lambda}^\mathsf{T} = [\lambda_1, \ldots, \lambda_p]^\mathsf{T}$ 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: October 28, 2019, 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
- also it involves $A^\intercal A$ and as in the case of normal equation $\kappa_2(A^\intercal A) = \kappa_2(A)^2$
- if n is small, this is fine but generally we want a method that avoids actually forming $A^\intercal A$
- which brings us to the next method — even though it appears to involve forming $M^\intercal M$ for various matrices M, it actually doesn't
- method 2: if A has full column rank, then from $A^\intercal A\mathbf{x} = A^\intercal\mathbf{b} - C\boldsymbol{\lambda}$, we see that we can first compute $\mathbf{x} = \widehat{\mathbf{x}} - (A^\intercal A)^{-1}C\boldsymbol{\lambda}$ 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 $C^\intercal\mathbf{x} = \mathbf{d}$ we obtain the $p \times p$ linear system
$$C^\intercal(A^\intercal A)^{-1}C\boldsymbol{\lambda} = C^\intercal\widehat{\mathbf{x}} - \mathbf{d} \tag{1.4}$$
which we can then solve for $\boldsymbol{\lambda}$
- this works because $A^\intercal A\widehat{\mathbf{x}} = A^\intercal\mathbf{b}$ and therefore
$$A^\intercal A\mathbf{x} = A^\intercal\mathbf{b} - C\boldsymbol{\lambda}$$
- the actual algorithm uses two QR factorization and does not actually require solving a system involving $M^\intercal M$ for any M
  - compute full-rank QR factorization of A
  $$A = Q\begin{bmatrix} R \\ 0 \end{bmatrix}$$
  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 $W = R^{-\intercal}C$ with p back substitutions
  $$R^\intercal\mathbf{w}_i = \mathbf{c}_i, \quad i = 1, \dots, p$$
  - compute QR factorization of W
  $$W = Q_1R_1$$
  - set
  $$\boldsymbol{\eta} = C^\intercal\widehat{\mathbf{x}} - \mathbf{d}$$
  - solve $R_1^\intercal R_1\boldsymbol{\lambda} = \boldsymbol{\eta}$ for $\boldsymbol{\lambda}$ with two back substitutions
  $$\begin{cases} R_1^\intercal\boldsymbol{\mu} = \boldsymbol{\eta}, \\ R_1\boldsymbol{\lambda} = \boldsymbol{\mu} \end{cases}$$
  - set $\mathbf{x} = \widehat{\mathbf{x}} - (R^\intercal R)^{-1}C\boldsymbol{\lambda}$ where term $\boldsymbol{\zeta} = (R^\intercal R)^{-1}C\boldsymbol{\lambda}$ is computed again with two back substitutions
  $$\begin{cases} R^\intercal\boldsymbol{\xi} = C\boldsymbol{\lambda}, \\ R\boldsymbol{\zeta} = \boldsymbol{\xi} \end{cases}$$
- this works because
$$A^\intercal A = \begin{bmatrix} R^\intercal & 0 \end{bmatrix} Q^\intercal Q\begin{bmatrix} R \\ 0 \end{bmatrix} = R^\intercal R$$
and
$$R_1^\intercal R_1 = (Q_1^\intercal W)^\intercal(Q_1^\intercal W) = W^\intercal Q_1Q_1^\intercal W = C^\intercal R^{-1}R^{-\intercal}C = C^\intercal(R^\intercal R)^{-1}C = C^\intercal(A^\intercal A)^{-1}C$$
- method 2, like method 1, has more unknowns than the unconstrained least squares problem, which is a downside because the constraints should have the effect of eliminating unknowns, not adding them

- the next method overcomes this by solving (1.1) without introducing any Lagrange multipliers
- 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

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

where $\mathbf{v}$ is to be determined later
- 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 $R_2^{\mathsf{T}}\mathbf{u} = \mathbf{d}$ to obtain a solution $\mathbf{u}_*$
  - solve the new least squares problem

$$\mathbf{v}_* = \operatorname{argmin} \|(\mathbf{b} - \widetilde{A}_1\mathbf{u}_*) - \widetilde{A}_2\mathbf{v}\|_2$$

  - 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 $\kappa_2(\widetilde{A}_2) \le \kappa_2(\widetilde{A}) = \kappa_2(A)$
- the drawback is that sparsity or other structure in A can be destroyed when we form $\widetilde{A}$

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 Householder reflections

- it is natural to ask whether we can introduce more zeros with each orthogonal rotation and to that end, we examine *Householder reflections*
- consider a matrix of the form $P = I - \tau \mathbf{u}\mathbf{u}^\mathsf{T}$, where $\mathbf{u} \neq \mathbf{0}$ and τ is a nonzero constant
- a P that has this form is called a *symmetric rank-1 change* of I
- can we choose τ so that P is also orthogonal?
- from the desired relation $P^\mathsf{T}P = I$ we obtain

$$\begin{aligned} P^\mathsf{T}P &= (I - \tau\mathbf{u}\mathbf{u}^\mathsf{T})^\mathsf{T}(I - \tau\mathbf{u}\mathbf{u}^\mathsf{T}) \\ &= I - 2\tau\mathbf{u}\mathbf{u}^\mathsf{T} + \tau^2\mathbf{u}\mathbf{u}^\mathsf{T}\mathbf{u}\mathbf{u}^\mathsf{T} \\ &= I - 2\tau\mathbf{u}\mathbf{u}^\mathsf{T} + \tau^2(\mathbf{u}^\mathsf{T}\mathbf{u})\mathbf{u}\mathbf{u}^\mathsf{T} \\ &= I - (\tau^2\mathbf{u}^\mathsf{T}\mathbf{u} - 2\tau)\mathbf{u}\mathbf{u}^\mathsf{T} \\ &= I + \tau(\tau\mathbf{u}^\mathsf{T}\mathbf{u} - 2)\mathbf{u}\mathbf{u}^\mathsf{T} \end{aligned}$$

- it follows that if $\tau = 2/\mathbf{u}^\mathsf{T}\mathbf{u}$, then $P^\mathsf{T}P = I$ for any nonzero $\mathbf{u}$
- without loss of generality, we can stipulate that $\mathbf{u}^\mathsf{T}\mathbf{u} = 1$, and therefore P takes the form $P = I - 2\mathbf{v}\mathbf{v}^\mathsf{T}$, where $\mathbf{v}^\mathsf{T}\mathbf{v} = 1$
- why is the matrix P called a reflection?
- this is because for any nonzero vector $\mathbf{x}$, $P\mathbf{x}$ is the reflection of $\mathbf{x}$ across the hyperplane that is normal to $\mathbf{v}$
- for example, consider the 2×2 case and set $\mathbf{v} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\mathsf{T}$ and $\mathbf{x} = \begin{bmatrix} 1 & 2 \end{bmatrix}^\mathsf{T}$, then

$$P = I - 2\mathbf{v}\mathbf{v}^\mathsf{T} = I - 2\begin{bmatrix} 1 \\ 0 \end{bmatrix}\begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - 2\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

  and therefore

$$P\mathbf{x} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

- now, let $\mathbf{x}$ be any vector, we wish to construct P so that $P\mathbf{x} = \alpha \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^\mathsf{T} = \alpha\mathbf{e}_1$ for some α
- from the relations

$$\|P\mathbf{x}\|_2 = \|\mathbf{x}\|_2, \qquad \|\alpha\mathbf{e}_1\|_2 = |\alpha|\|\mathbf{e}_1\|_2 = |\alpha|$$

  we obtain $\alpha = \pm\|\mathbf{x}\|_2$
- to determine P, we observe that

$$\mathbf{x} = P^{-1}(\alpha\mathbf{e}_1) = \alpha P\mathbf{e}_1 = \alpha(I - 2\mathbf{v}\mathbf{v}^\mathsf{T})\mathbf{e}_1 = \alpha(\mathbf{e}_1 - 2\mathbf{v}\mathbf{v}^\mathsf{T}\mathbf{e}_1) = \alpha(\mathbf{e}_1 - 2\mathbf{v}v_1)$$

  which yields the system of equations

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \alpha \begin{bmatrix} 1 - 2v_1^2 \\ -2v_1v_2 \\ \vdots \\ -2v_1v_n \end{bmatrix}$$

- from the first equation $x_1 = \alpha(1 - 2v_1^2)$ we obtain

$$v_1 = \pm\sqrt{\frac{1}{2}\left(1 - \frac{x_1}{\alpha}\right)}$$

- for $i = 2, \ldots, n$, we have
$$v_i = -\frac{x_i}{2\alpha v_1}$$
- it is best to choose α to have the opposite sign of x_1 to avoid cancellation in v_1
- it is conventional to choose the + sign for α if $x_1 = 0$
- note that the matrix P is never formed explicitly: for any vector $\mathbf{b}$, the product $P\mathbf{b}$ can be computed as follows
$$P\mathbf{b} = (I - 2\mathbf{v}\mathbf{v}^\mathsf{T})\mathbf{b} = \mathbf{b} - 2(\mathbf{v}^\mathsf{T}\mathbf{b})\mathbf{v}$$
- this process requires only $2n$ operations
- it is easy to see that we can represent P simply by storing only $\mathbf{v}$
- we showed how a Householder reflection of the form $P = I - 2\mathbf{u}\mathbf{u}^\mathsf{T}$ could be constructed so that given a vector $\mathbf{x}$, $P\mathbf{x} = \alpha\mathbf{e}_1$
- now, suppose that that $\mathbf{x} = \mathbf{a}_1$ is the first column of a matrix A, then we construct a Householder reflection $H_1 = I - 2\mathbf{u}_1\mathbf{u}_1^\mathsf{T}$ such that $H\mathbf{x} = \alpha\mathbf{e}_1$, and we have
$$A^{(2)} = H_1 A = \begin{bmatrix} r_{11} & & & \\ 0 & & & \\ \vdots & \mathbf{a}_2^{(2)} & \cdots & \mathbf{a}_n^{(2)} \\ 0 & & & \end{bmatrix}$$
where we denote the constant α by r_{11}, as it is the $(1,1)$ element of the updated matrix $A^{(2)}$
- now, we can construct H_2 such that
$$H_2\mathbf{a}^{(2)} = \begin{bmatrix} a_{12}^{(2)} \\ r_{22} \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad u_{12} = 0, \quad H_2 = \begin{bmatrix} 1 & 0 \\ 0 & \\ \vdots & h_{ij} \\ 0 & \end{bmatrix}$$
- note that the first column of $A^{(2)}$ is unchanged by H_2
- continuing this process, we obtain
$$H_{n-1} \cdots H_1 A = A^{(n)} = R$$
where R is an upper triangular matrix
- we have thus factored $A = QR$, where $Q = H_1 H_2 \cdots H_{n-1}$ is an orthogonal matrix
- note that
$$A^\mathsf{T}A = R^\mathsf{T}Q^\mathsf{T}QR = R^\mathsf{T}R,$$
and thus R is the Cholesky factor of $A^\mathsf{T}A$ (we will discus Cholesky factorization next week)