CS 598 EVS: Tensor Computations

Matrix Computations Background

Edgar Solomonik

University of Illinois at Urbana-Champaign

Matrix Condition Number

- The matrix condition number $\kappa(\boldsymbol{A})$ is the ratio between the max and min distance from the surface to the center of the unit ball (norm-1 vectors) transformed by $\boldsymbol{A}$:
  - *The max distance to center is given by the vector maximizing* $\max_{||\boldsymbol{x}||=1} ||\boldsymbol{A}\boldsymbol{x}||_2$.
  - *The min distance to center is given by the vector minimizing* $\min_{||\boldsymbol{x}||=1} ||\boldsymbol{A}\boldsymbol{x}||_2 = 1/(\max_{||\boldsymbol{x}||=1} ||\boldsymbol{A}^{-1}\boldsymbol{x}||_2)$.
  - *Thus, we have that* $\kappa(\boldsymbol{A}) = ||\boldsymbol{A}||_2||\boldsymbol{A}^{-1}||_2$
- The matrix condition number bounds the worst-case amplification of error in a matrix-vector product: *Consider* $\boldsymbol{y} + \boldsymbol{\delta y} = \boldsymbol{A}(\boldsymbol{x} + \boldsymbol{\delta x})$, *assume* $||\boldsymbol{x}||_2 = 1$
  - *In the worst case,* $||\boldsymbol{y}||_2$ *is minimized, that is* $||\boldsymbol{y}||_2 = 1/||\boldsymbol{A}^{-1}||_2$
  - *In the worst case,* $||\boldsymbol{\delta y}||_2$ *is maximized, that is* $||\boldsymbol{\delta y}||_2 = ||\boldsymbol{A}||_2||\boldsymbol{\delta y}||_2$
  - *So* $||\boldsymbol{\delta y}||_2/||\boldsymbol{y}||_2$ *is at most* $\kappa(\boldsymbol{A})||\boldsymbol{\delta x}||_2/||\boldsymbol{x}||_2$

Singular Value Decomposition

- The singular value decomposition (SVD)

  We can express any matrix $\boldsymbol{A}$ as

  $$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

  where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal, and $\boldsymbol{\Sigma}$ is square nonnegative and diagonal,

  $$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{max} & & \\ & \ddots & \\ & & \sigma_{min} \end{bmatrix}$$

  Any matrix is diagonal when expressed as an operator mapping vectors from a coordinate system given by $\boldsymbol{V}$ to a coordinate system given by $\boldsymbol{U}^T$.

- Condition number in terms of singular values
  - *We have that $\|\boldsymbol{A}\|_2 = \sigma_{max}$ and if $\boldsymbol{A}^{-1}$ exists, $\|\boldsymbol{A}^{-1}\|_2 = 1/\sigma_{min}$*
  - *Consequently, $\kappa(\boldsymbol{A}) = \sigma_{max}/\sigma_{min}$*

Linear Least Squares

- Find $\boldsymbol{x}^\star = \operatorname{argmin}_{\boldsymbol{x}\in\mathbb{R}^n} ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}||_2$ where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$:

  Since $m \geq n$, the minimizer generally does not attain a zero residual $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$. We can rewrite the optimization problem constraint via

$$\boldsymbol{x}^\star = \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^n} ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}||_2^2 = \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^n} \left[(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})\right]$$

- Given the SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ we have $\boldsymbol{x}^\star = \underbrace{\boldsymbol{V}\boldsymbol{\Sigma}^\dagger\boldsymbol{U}^T}_{\boldsymbol{A}^\dagger}\boldsymbol{b}$, where $\boldsymbol{\Sigma}^\dagger$ contains the reciprocal of all nonzeros in $\boldsymbol{\Sigma}$, and more generally $\dagger$ denotes pseudoinverse:
  - *The minimizer satisfies $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{x}^\star \cong \boldsymbol{b}$ and consequently also satisfies*

$$\boldsymbol{\Sigma}\boldsymbol{y}^\star \cong \boldsymbol{d} \quad \text{where } \boldsymbol{y}^\star = \boldsymbol{V}^T\boldsymbol{x}^\star \text{ and } \boldsymbol{d} = \boldsymbol{U}^T\boldsymbol{b}.$$

  - *The minimizer of the reduced problem is $\boldsymbol{y}^\star = \boldsymbol{\Sigma}^\dagger\boldsymbol{d}$, so $y_i = d_i/\sigma_i$ for $i \in \{1,\ldots,n\}$ and $y_i = 0$ for $i \in \{n+1,\ldots,m\}$.*

Normal Equations

Demo: *Normal equations vs Pseudoinverse*
Demo: *Issues with the normal equations*

- *Normal equations* are given by solving $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}$:
  If $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}$ *then*

$$
\begin{aligned}
(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T)^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{x} &= (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T)^T\boldsymbol{b} \\
\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{x} &= \boldsymbol{\Sigma}^T\boldsymbol{U}^T\boldsymbol{b} \\
\boldsymbol{V}^T\boldsymbol{x} &= (\boldsymbol{\Sigma}^T\boldsymbol{\Sigma})^{-1}\boldsymbol{\Sigma}^T\boldsymbol{U}^T\boldsymbol{b} = \boldsymbol{\Sigma}^\dagger\boldsymbol{U}^T\boldsymbol{b} \\
\boldsymbol{x} &= \boldsymbol{V}\boldsymbol{\Sigma}^\dagger\boldsymbol{U}^T\boldsymbol{b} = \boldsymbol{x}^\star
\end{aligned}
$$

- However, solving the normal equations is a more ill-conditioned problem then the original least squares algorithm

  Generally we have $\kappa(\boldsymbol{A}^T\boldsymbol{A}) = \kappa(\boldsymbol{A})^2$ *(the singular values of* $\boldsymbol{A}^T\boldsymbol{A}$ *are the squares of those in* $\boldsymbol{A}$*). Consequently, solving the least squares problem via the normal equations may be unstable because it involves solving a problem that has worse conditioning than the initial least squares problem.*

Solving the Normal Equations

- If $\boldsymbol{A}$ is full-rank, then $\boldsymbol{A}^T\boldsymbol{A}$ is symmetric positive definite (SPD):
  - *Symmetry is easy to check* $(\boldsymbol{A}^T\boldsymbol{A})^T = \boldsymbol{A}^T\boldsymbol{A}$.
  - $\boldsymbol{A}$ *being full-rank implies* $\sigma_{min} > 0$ *and further if* $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ *we have*

$$\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{V}^T\boldsymbol{\Sigma}^2\boldsymbol{V}$$

    which implies that rows of $\boldsymbol{V}$ *are the eigenvectors of* $\boldsymbol{A}^T\boldsymbol{A}$ *with eigenvalues* $\boldsymbol{\Sigma}^2$ *since* $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{V}^T = \boldsymbol{V}^T\boldsymbol{\Sigma}^2$.

- Since $\boldsymbol{A}^T\boldsymbol{A}$ is SPD we can use Cholesky factorization, to factorize it and solve linear systems:

$$\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T$$

QR Factorization

- If $\boldsymbol{A}$ is full-rank there exists an orthogonal matrix $\boldsymbol{Q}$ and a unique upper-triangular matrix $\boldsymbol{R}$ with a positive diagonal such that $\boldsymbol{A} = \boldsymbol{QR}$
  - *Given $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{LL}^T$, we can take $\boldsymbol{R} = \boldsymbol{L}^T$ and obtain $\boldsymbol{Q} = \boldsymbol{AL}^{-T}$, since $\underbrace{\boldsymbol{L}^{-1}\boldsymbol{A}^T}_{\boldsymbol{Q}^T}\underbrace{\boldsymbol{AL}^{-T}}_{\boldsymbol{Q}} = \boldsymbol{I}$ implies that $\boldsymbol{Q}$ has orthonormal columns.*
- A reduced QR factorization (unique part of general QR) is defined so that $\boldsymbol{Q} \in \mathbb{R}^{m\times n}$ has orthonormal columns and $\boldsymbol{R}$ is square and upper-triangular

  A full QR factorization gives $\boldsymbol{Q} \in \mathbb{R}^{m\times m}$ and $\boldsymbol{R} \in \mathbb{R}^{m\times n}$, but since $\boldsymbol{R}$ is upper triangular, the latter $m-n$ columns of $\boldsymbol{Q}$ are only constrained so as to keep $\boldsymbol{Q}$ orthogonal. The reduced QR factorization is given by taking the first n columns $\boldsymbol{Q}$ and $\hat{\boldsymbol{Q}}$ the upper-triangular block of $\boldsymbol{R}$, $\hat{\boldsymbol{R}}$ giving $\boldsymbol{A} = \hat{\boldsymbol{Q}}\hat{\boldsymbol{R}}$.
- We can solve the normal equations (and consequently the linear least squares problem) via reduced QR as follows

$$\boldsymbol{A}^T\boldsymbol{Ax} = \boldsymbol{A}^T\boldsymbol{b} \quad \Rightarrow \quad \hat{\boldsymbol{R}}^T\underbrace{\hat{\boldsymbol{Q}}^T\hat{\boldsymbol{Q}}}_{\boldsymbol{I}}\hat{\boldsymbol{R}}\boldsymbol{x} = \hat{\boldsymbol{R}}^T\hat{\boldsymbol{Q}}^T\boldsymbol{b} \quad \Rightarrow \quad \hat{\boldsymbol{R}}\boldsymbol{x} = \hat{\boldsymbol{Q}}^T\boldsymbol{b}$$

Eigenvalue Decomposition

- If a matrix $\boldsymbol{A}$ is diagonalizable, it has an *eigenvalue decomposition*

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{D}\boldsymbol{X}^{-1}$$

where $\boldsymbol{X}$ are the right eigenvectors, $\boldsymbol{X}^{-1}$ are the left eigenvectors and $\boldsymbol{D}$ are eigenvalues

$$\boldsymbol{A}\boldsymbol{X} = \begin{bmatrix}\boldsymbol{A}\boldsymbol{x}_1 & \cdots \boldsymbol{A}\boldsymbol{x}_n\end{bmatrix} = \boldsymbol{X}\boldsymbol{D} = \begin{bmatrix}d_{11}\boldsymbol{x}_1 & \cdots & d_{nn}\boldsymbol{x}_n\end{bmatrix}.$$

  - *If $\boldsymbol{A}$ is symmetric, its right and left singular vectors are the same, and consequently are its eigenvectors.*
  - *More generally, any normal matrix, $\boldsymbol{A}^H\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^H$, has unitary eigenvectors.*
- $\boldsymbol{A}$ and $\boldsymbol{B}$ are *similar*, if there exist $\boldsymbol{Z}$ such that $\boldsymbol{A} = \boldsymbol{Z}\boldsymbol{B}\boldsymbol{Z}^{-1}$
  - *Normal matrices are unitarily similar ($\boldsymbol{Z}^{-1} = \boldsymbol{Z}^H$) to diagonal matrices*
  - *Symmetric real matrices are orthogonally similar ($\boldsymbol{Z}^{-1} = \boldsymbol{Z}^T$) to real diagonal matrices*
  - *Hermitian matrices are unitarily similar to real diagonal matrices*

Similarity of Matrices

matrix	*similarity*	*reduced form*
SPD	orthogonal	real positive diagonal
real symmetric	orthogonal	real tridiagonal real diagonal
Hermitian	unitary	real diagonal
normal	unitary	diagonal
real	orthogonal	real Hessenberg
diagonalizable	invertible	diagonal
arbitrary	unitary invertible	triangular bidiagonal

Rayleigh Quotient

- For any vector $\boldsymbol{x}$ that is close to an eigenvector, the *Rayleigh quotient* provides an estimate of the associated eigenvalue of $\boldsymbol{A}$:

$$\rho_{\boldsymbol{A}}(\boldsymbol{x}) = \frac{\boldsymbol{x}^H \boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^H \boldsymbol{x}}.$$

  - *If $\boldsymbol{x}$ is an eigenvector of $\boldsymbol{A}$, then $\rho_{\boldsymbol{A}}(\boldsymbol{x})$ is the associated eigenvalue.*
  - *Moreover, for $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, the Rayleigh quotient is the best possible eigenvalue estimate given $\boldsymbol{x}$ and $\boldsymbol{y}$, as it is the solution α to $\boldsymbol{x}\alpha \cong \boldsymbol{y}$.*
    - *The normal equations for this scalar-output least squares problem are (assuming $\boldsymbol{A}$ is real),*

$$\boldsymbol{x}^T \boldsymbol{x}\alpha = \boldsymbol{x}^T \boldsymbol{y} \quad \Rightarrow \quad \alpha = \frac{\boldsymbol{x}^T \boldsymbol{y}}{\boldsymbol{x}^T \boldsymbol{x}} = \frac{\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}}.$$

Introduction to Krylov Subspace Methods

- *Krylov subspace methods* work with information contained in the $n \times k$ matrix

$$\boldsymbol{K}_k = \begin{bmatrix} \boldsymbol{x_0} & \boldsymbol{A x_0} & \cdots & \boldsymbol{A}^{k-1}\boldsymbol{x_0} \end{bmatrix}$$

We seek to best use the information from the matrix vector product results (columns of $\boldsymbol{K}_k$) to solve eigenvalue problems.

- $\boldsymbol{A}$ is similar to *companion matrix* $\boldsymbol{C} = \boldsymbol{K}_n^{-1}\boldsymbol{A}\boldsymbol{K}_n$:

Letting $\boldsymbol{k}_n^{(i)} = \boldsymbol{A}^{i-1}\boldsymbol{x}$, we observe that

$$\boldsymbol{A}\boldsymbol{K}_n = \begin{bmatrix} \boldsymbol{A}\boldsymbol{k}_n^{(1)} & \cdots & \boldsymbol{A}\boldsymbol{k}_n^{(n-1)} & \boldsymbol{A}\boldsymbol{k}_n^{(n)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{k}_n^{(2)} & \cdots & \boldsymbol{k}_n^{(n)} & \boldsymbol{A}\boldsymbol{k}_n^{(n)} \end{bmatrix},$$

therefore premultiplying by $\boldsymbol{K}_m^{-1}$ transforms the first $n-1$ columns of $\boldsymbol{A}\boldsymbol{K}_n$ into the last $n-1$ columns of $\boldsymbol{I}$,

$$\begin{aligned} \boldsymbol{K}_n^{-1}\boldsymbol{A}\boldsymbol{K}_n &= \begin{bmatrix} \boldsymbol{K}_n^{-1}\boldsymbol{k}_n^{(2)} & \cdots & \boldsymbol{K}_n^{-1}\boldsymbol{k}_n^{(n)} & \boldsymbol{K}_n^{-1}\boldsymbol{A}\boldsymbol{k}_n^{(n)} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{e}_2 & \cdots & \boldsymbol{e}_n & \boldsymbol{K}_n^{-1}\boldsymbol{A}\boldsymbol{k}_n^{(n)} \end{bmatrix} \end{aligned}$$

Krylov Subspaces

- Given $\boldsymbol{Q}_k\boldsymbol{R}_k = \boldsymbol{K}_k$, we obtain an orthonormal basis for the Krylov subspace,

$$\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{x}_0) = span(\boldsymbol{Q}_k) = \{p(\boldsymbol{A})\boldsymbol{x}_0 : deg(p) < k\},$$

  where p is any polynomial of degree less than k.

- The Krylov subspace includes the $k-1$ approximate dominant eigenvectors generated by $k-1$ steps of power iteration:
  - *The approximation obtained from $k-1$ steps of power iteration starting from $\boldsymbol{x}_0$ is given by the Rayleigh-quotient of $\boldsymbol{y} = \boldsymbol{A}^k\boldsymbol{x}_0$.*
  - *This vector is within the Krylov subspace, $\boldsymbol{y} \in \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{x}_0)$.*
  - *Consequently, Krylov subspace methods will generally obtain strictly better approximations of the dominant eigenpair than power iteration.*

Krylov Subspace Methods

- The $k \times k$ matrix $\boldsymbol{H}_k = \boldsymbol{Q}_k^T \boldsymbol{A} \boldsymbol{Q}_k$ minimizes $||\boldsymbol{A}\boldsymbol{Q}_k - \boldsymbol{Q}_k \boldsymbol{H}_k||_2$:

  The minimizer $\boldsymbol{X}$ for the linear least squares problem $\boldsymbol{Q}_k \boldsymbol{X} \cong \boldsymbol{A}\boldsymbol{Q}_k$ is (via the normal equations) $\boldsymbol{X} = \boldsymbol{Q}_k^T \boldsymbol{A} \boldsymbol{Q}_k = \boldsymbol{H}_k$.

- $\boldsymbol{H}_k$ is upper-Hessenberg, because the companion matrix $\boldsymbol{C}_n$ is upper-Hessenberg:

  Note that $\boldsymbol{H}_k$ is the leading k-by-k minor of H_n and

$$\boldsymbol{H}_n = \boldsymbol{Q}_n^T \boldsymbol{A} \boldsymbol{Q}_n = \boldsymbol{R}\boldsymbol{K}_n^{-1} \boldsymbol{A} \boldsymbol{K}_n \boldsymbol{R}^{-1} = \boldsymbol{R}\boldsymbol{C}_n \boldsymbol{R}^{-1}$$

  is a product of three matrices: upper-triangular $\boldsymbol{R}$, upper-Hessenberg $\boldsymbol{C}_n$, and upper-triangular $\boldsymbol{R}^{-1}$, which results in upper-Hessenberg $\boldsymbol{H}_n$.

Rayleigh-Ritz Procedure

- The eigenvalues/eigenvectors of $\boldsymbol{H}_k$ are the *Ritz values/vectors*:

$$\boldsymbol{H}_k = \boldsymbol{X}\boldsymbol{D}\boldsymbol{X}^{-1}$$

eigenvalue approximations based on Ritz vectors $\boldsymbol{X}$ are given by $\boldsymbol{Q}_k\boldsymbol{X}$.

- The Ritz vectors and values are the *ideal approximations* of the actual eigenvalues and eigenvectors based on only $\boldsymbol{H}_k$ and $\boldsymbol{Q}_k$:

Assuming $\boldsymbol{A}$ is a symmetric matrix with positive eigenvalues, the largest Ritz value $\lambda_{max}(\boldsymbol{H}_k)$ will be the maximum Rayleigh quotient of any vector in $\mathcal{K}_k = span(\boldsymbol{Q}_k)$,

$$\max_{\boldsymbol{x}\in span(\boldsymbol{Q}_k)} \frac{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}} = \max_{\boldsymbol{y}\neq 0} \frac{\boldsymbol{y}^T\boldsymbol{Q}_k^T\boldsymbol{A}\boldsymbol{Q}_k\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{y}} = \max_{\boldsymbol{y}\neq 0} \frac{\boldsymbol{y}^T\boldsymbol{H}_k\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{y}} = \lambda_{max}(\boldsymbol{H}_k),$$

which is the best approximation to $\lambda_{max}(\boldsymbol{A}) = \max_{\boldsymbol{x}\neq 0} \frac{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}}$ available in $\mathcal{K}_k$. The quality of the approximation can also be shown to be optimal for other eigenvalues/eigenvectors.

Low Rank Matrix Approximation

- Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ seek rank $r < m, n$ approximation
  - *Given by matrices $\boldsymbol{U} \in \mathbb{R}^{m \times r}$ and $\boldsymbol{V} \in \mathbb{R}^{n \times r}$ so*

$$\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{V}^T$$

  - *Reduces memory footprint and cost of applying $\boldsymbol{A}$ from mn to $mr + nr$*
  - *This factorization is nonunique, $\boldsymbol{U}\boldsymbol{V}^T = (\boldsymbol{U}\boldsymbol{M})(\boldsymbol{V}\boldsymbol{M}^{-T})^T$*
- Eckart-Young (optimal low-rank approximation by SVD) theorem
  - *Truncated SVD approximates $\boldsymbol{A}$ as*

$$\boldsymbol{A} \approx \tilde{\boldsymbol{A}} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$$

    where $\sigma_1, \ldots, \sigma_r$ are the largest r singular values, while $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are the associated left and right singular vectors
  - *Eckart-Young theorem demonstrates that the truncated SVD minimizes*

$$\underbrace{\|\boldsymbol{A} - \tilde{\boldsymbol{A}}\|_2}_{\sigma_{r+1}} \quad \textit{and} \quad \underbrace{\|\boldsymbol{A} - \tilde{\boldsymbol{A}}\|_F}_{\sum_{i=r+1}^{\min(m,n)} \sigma_i}$$

Rank Revealing Matrix Factorizations

- Computing the SVD
  - *Can compute full SVD with $O(mn\min(m,n))$ cost via bidiagonalization*
    - *unconditionally stable and accurate*
    - *inefficient for low r or if $\boldsymbol{A}$ is sparse*
  - *Given any low-rank approximation composed of $\boldsymbol{U}$ and $\boldsymbol{V}$, compute QR of each and SVD of product of $\boldsymbol{R}$ factors to obtain SVD with total cost $O((m+n)r^2)$*
- QR with column pivoting
  - *By selecting columns of largest norm in the trailing matrix during QR factorization, we obtain a pivoted factorization with permutation matirx $\boldsymbol{P}$*

$$\boldsymbol{AP} = \boldsymbol{QR}$$

  - *Truncating this factorization can be done after applying r Householder reflectors (or another QR algorithm on r columns), with cost $O((m+n)r)$*
  - *Approximation is somewhat suboptimal in theory, but in practice almost always as accurate as truncated SVD*

Orthogonal Iteration

- For sparse matrices, QR factorization creates fill, so must revert to iterative methods
  - *Can find SVD of $\boldsymbol{A}$ by implicit products with $\boldsymbol{A}^T\boldsymbol{A}$ or $\boldsymbol{A}\boldsymbol{A}^T$, since left singular vectors of $\boldsymbol{A}$ are eigenvectors of $\boldsymbol{A}^T\boldsymbol{A}$*
  - *Krylov subspace methods are effective for computing the largest eigenvector*
  - *Deflation, e.g., $\boldsymbol{A} \rightarrow (\boldsymbol{A} - \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T)$ can be used to compute other eigenvectors*
- Orthogonal iteration interleaves deflation and power iteration
  - *Given starting eigenvector guess $\boldsymbol{U}^{(0)} \in \mathbb{R}^{n \times r}$, compute $\boldsymbol{V}^{(i+1)} = \boldsymbol{A}\boldsymbol{U}^{(i)}$ and obtain $\boldsymbol{U}^{(i+1)}$ as the $\boldsymbol{Q}$ factor of the QR of $\boldsymbol{V}^{(i+1)}$*
  - *Converges to r largest eigenvectors, for SVD can compute $\boldsymbol{V}^{(i+1)} = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{U}^{(i)})$ at each iteration*
  - *QR factorization serves to orthogonalize each column w.r.t. eigenvectors being converged to by previous columns*

Randomized SVD

- Orthogonal iteration for SVD can also be viewed as a randomized algorithm
  - *Suppose that we have an exact low-rank factorization* $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ *with* $\boldsymbol{\Sigma} \in \mathbb{R}^{r \times r}$
  - *If* $\boldsymbol{U}^{(0)}$ *is a random orthogonal matrix, so is* $\boldsymbol{V}^T\boldsymbol{U}^{(0)}$
  - *Consequently,* $\boldsymbol{A}\boldsymbol{U}^{(0)}$ *is a set of* r *random linear combinations of columns of* $\boldsymbol{U}\boldsymbol{\Sigma}$
  - *Further,* $\boldsymbol{U} = \boldsymbol{U}^{(1)}\boldsymbol{U}^{(1)T}\boldsymbol{U}$ *since*

$$\mathit{span}(\boldsymbol{U}^{(1)}) = \mathit{span}(\boldsymbol{V}^{(1)}) = \mathit{span}(\boldsymbol{U}),$$

    the latter equality holds with probability 1
  - *Consequently, we can compute SVD of* $\boldsymbol{U}^{(1)T}\boldsymbol{A}$ *(with cost* $O(nr^2)$*) and recover* $\boldsymbol{U}$ *by premultiplying the computed left singular vectors by* $\boldsymbol{U}^{(1)}$
  - *When* $\boldsymbol{A}$ *is not exactly low-rank, span of leading singular vectors can be captured by oversampling (e.g., selecting each* $\boldsymbol{U}^{(i)}$ *to have* $r + 10$ *columns)*
  - *Initial guess* $\boldsymbol{U}^{(0)}$ *need not be orthogonal (Gaussian random performs well, structured pseudo-random enables* $O(mn \log n)$ *complexity for one-shot randomized SVD), but better accuracy is obtained with orthogonality*

Multidimensional Optimization

- Minimize $f(\boldsymbol{x})$
  - *In the context of constrained optimization, also have equality and or inequality constraints, e.g., $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ or $\boldsymbol{x} > 0$*
  - *Unconstrained local optimality holds if $\nabla f(\boldsymbol{x}^*) = 0$ and $H_f(\boldsymbol{x}^*)$ is positive semi-definite*
  - *Reduces to solving nonlinear equations via optimality condition*
  - *Unconstrained local optimality conditions are looser, need the gradient to be zero or positive in all unconstrained directions at $\boldsymbol{x}^*$*
  - *The condition $\nabla f(\boldsymbol{x}^*) = 0$ implies poor conditioning, perturbations that change the function value in the kth digit can change the sollution in the $(k/2)$th digit*
- Quadratic optimization $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}$
  - *Quadratic optimization problems can provide local approximations to general nonlinear optimization problems via Newton's method (where $\boldsymbol{A}$ is the Hessian and $\boldsymbol{b}^T$ is the gradient)*
  - *Equivalent to solving linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ by optimality condition*
  - *Accordingly, conditioning relative to perturbation in $\boldsymbol{b}$ is $\kappa(\boldsymbol{A})$*

Basic Multidimensional Optimization Methods

- Steepest descent: minimize f in the direction of the negative gradient:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$$

such that $f(\boldsymbol{x}_{k+1}) = \min_{\alpha_k} f(\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k))$*, i.e. perform a line search (solve 1D optimization problem) in the direction of the negative gradient.*

- Given quadratic optimization problem $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^T \boldsymbol{x}$ where $\boldsymbol{A}$ is symmetric positive definite, the error $\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{x}^*$ satisfies

$$||\boldsymbol{e}_{k+1}||_{\boldsymbol{A}} = \boldsymbol{e}_{k+1}^T \boldsymbol{A} \boldsymbol{e}_{k+1} = \frac{\sigma_{\mathsf{max}}(\boldsymbol{A}) - \sigma_{\mathsf{min}}(\boldsymbol{A})}{\sigma_{\mathsf{max}}(\boldsymbol{A}) + \sigma_{\mathsf{min}}(\boldsymbol{A})} ||\boldsymbol{e}_k||_{\boldsymbol{A}}$$

  - When sufficiently close to a local minima, general nonlinear optimization problems are described by such an SPD quadratic problem.
  - Convergence rate depends on the conditioning of $\boldsymbol{A}$, since

$$\frac{\sigma_{max}(\boldsymbol{A}) - \sigma_{min}(\boldsymbol{A})}{\sigma_{max}(\boldsymbol{A}) + \sigma_{min}(\boldsymbol{A})} = \frac{\kappa(\boldsymbol{A}) - 1}{\kappa(\boldsymbol{A}) + 1}.$$

Gradient Methods with Extrapolation

- We can improve the constant in the linear rate of convergence of steepest descent by leveraging *extrapolation methods*, which consider two previous iterates (maintain *momentum* in the direction $\boldsymbol{x}_k - \boldsymbol{x}_{k-1}$):

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k) + \beta_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$$

- The *heavy ball method*, which uses constant $\alpha_k = \alpha$ and $\beta_k = \beta$, achieves better convergence than steepest descent:

$$||\boldsymbol{e}_{k+1}||_{\boldsymbol{A}} = \frac{\sqrt{\kappa(\boldsymbol{A})} - 1}{\sqrt{\kappa(\boldsymbol{A})} + 1} ||\boldsymbol{e}_k||_{\boldsymbol{A}}$$

Nesterov's gradient optimization method is another instance of an extrapolation method that provides further improved optimality guarantees.

Conjugate Gradient Method

- The *conjugate gradient method* is capable of making the optimal (for a quadratic objective) choice of α_k and β_k at each iteration of an extrapolation method:

$$(\alpha_k, \beta_k) = \underset{\alpha_k, \beta_k}{\operatorname{argmin}} \left[f\Big(\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k) + \beta_k (\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) \Big) \right]$$

  - *For SPD quadratic programming problems, conjugate gradient is an optimal first order method, converging in n iterations.*
  - *It implicitly computes Lanczos iteration, searching along $\boldsymbol{A}$-orthogonal directions at each step.*

- *Parallel tangents* implementation of the method proceeds as follows

  1. *Perform a step of steepest descent to generate $\hat{\boldsymbol{x}}_k$ from $\boldsymbol{x}_k$.*
  2. *Generate $\boldsymbol{x}_{k+1}$ by minimizing over the line passing through $\boldsymbol{x}_{k-1}$ and $\hat{\boldsymbol{x}}_k$.*

The method is equivalent to CG for a quadratic objective function.

Krylov Optimization

- Conjugate Gradient finds the minimizer of $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}$ (which satisfies optimality condition $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$) within the Krylov subspace of $\boldsymbol{A}$:
  - *It constructs Krylov subspace* $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b}) = span(\boldsymbol{b}, \boldsymbol{A}\boldsymbol{b}, \ldots, \boldsymbol{A}^{r-1}\boldsymbol{b})$.
  - *At the* k*th step conjugate gradient yields iterate*

$$\boldsymbol{x}_k = -||\boldsymbol{b}||_2 \boldsymbol{Q}_k \boldsymbol{T}_k^{-1} \boldsymbol{e}_1,$$

    where $\boldsymbol{Q}_k$ *is an orthogonal basis for Krylov subspace* $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b})$ *and* $\boldsymbol{T}_k = \boldsymbol{Q}_k^T \boldsymbol{A} \boldsymbol{Q}_k$.
  - *This choice of* $\boldsymbol{x}_k$ *minimizes* $f(\boldsymbol{x})$ *since*

$$\begin{aligned} \min_{\boldsymbol{x} \in \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b})} f(\boldsymbol{x}) &= \min_{\boldsymbol{y} \in \mathbb{R}^k} f(\boldsymbol{Q}_k \boldsymbol{y}) \\ &= \min_{\boldsymbol{y} \in \mathbb{R}^k} \boldsymbol{y}^T \boldsymbol{Q}_k^T \boldsymbol{A} \boldsymbol{Q}_k \boldsymbol{y} + \boldsymbol{b}^T \boldsymbol{Q}_k \boldsymbol{y} \\ &= \min_{\boldsymbol{y} \in \mathbb{R}^k} \boldsymbol{y}^T \boldsymbol{T}_k \boldsymbol{y} + ||\boldsymbol{b}||_2 \boldsymbol{e}_1^T \boldsymbol{y} \end{aligned}$$

    is minimized by $\boldsymbol{y} = -||\boldsymbol{b}||_2 \boldsymbol{T}_k^{-1} \boldsymbol{e}_1$.

Conjugate Gradient Convergence Analysis

- In previous discussion, we assumed $\boldsymbol{K}_n$ is invertible, which may not be the case if $\boldsymbol{A}$ has $m < n$ distinct eigenvalues, however, in exact arithmetic CG converges in $m - 1$ iterations[1]
  - *the approximate solution $\boldsymbol{x}_k$ obtained by CG after $k-1$ iterations is given by minimizing $\boldsymbol{z}$ in*

$$\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{z}\|^2_{\boldsymbol{A}^{-1}} = \phi(\boldsymbol{z}) = (\boldsymbol{b} - \boldsymbol{A}\boldsymbol{z})^T \boldsymbol{A}^{-1}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{z}) = (\boldsymbol{x} - \boldsymbol{z})^T \boldsymbol{A}(\boldsymbol{x} - \boldsymbol{z})$$

    over all $\boldsymbol{z} = \boldsymbol{\rho}_{k-1}(\boldsymbol{A})\boldsymbol{b}$ where ρ_{k-1} can be any polynomial of degree $k-1$ and $\boldsymbol{\rho}_{k-1}$ denotes the corresponding matrix-valued polynomial
  - *note that the above is consistent with minimizing the quadratic objective, since the final expression is equal to*

$$\underbrace{\boldsymbol{z}^T \boldsymbol{A}\boldsymbol{z} - 2\boldsymbol{z}^T \boldsymbol{b}}_{2f(\boldsymbol{z})} - \underbrace{\boldsymbol{x}^T \boldsymbol{b}}_{\text{constant}}$$

  - *using $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ we can write $\boldsymbol{z} = \boldsymbol{\rho}_{k-1}(\boldsymbol{A})\boldsymbol{A}\boldsymbol{x}$*

[1] This derivation follows *Applied Numerical Linear Algebra* by James Demmel, Section 6.6.4

Conjugate Gradient Convergence Analysis (II)

- Using $\boldsymbol{z} = \boldsymbol{\rho}_{k-1}(\boldsymbol{A})\boldsymbol{A}\boldsymbol{x}$, we can simplify $\phi(\boldsymbol{z}) = (\boldsymbol{x}-\boldsymbol{z})^T\boldsymbol{A}(\boldsymbol{x}-\boldsymbol{z})$ as

$$\phi(\boldsymbol{z}) = \Big((\boldsymbol{I}-\boldsymbol{\rho}_{k-1}(\boldsymbol{A})\boldsymbol{A})\boldsymbol{x}\Big)^T\boldsymbol{A}\Big((\boldsymbol{I}-\boldsymbol{\rho}_{k-1}(\boldsymbol{A})\boldsymbol{A})\boldsymbol{x}\Big) = \boldsymbol{x}^T\boldsymbol{q}_k(\boldsymbol{A})\boldsymbol{A}\boldsymbol{q}_k(\boldsymbol{A})\boldsymbol{x}$$

where $\mathcal{Q}_k \ni q_k(\xi) = 1 - \rho_{k-1}(\xi)\cdot\xi$ can be any degree k polynomial with $q_k(0) = 1$ (or in matrix form, $\boldsymbol{q}_k(\boldsymbol{S}) = \boldsymbol{I} - \boldsymbol{\rho}_{k-1}(\boldsymbol{S})\boldsymbol{S}$ with $\boldsymbol{q}_k(\boldsymbol{O}) = \boldsymbol{I}$), so

$$\phi(x_k) = \min_{z\in\mathcal{K}_k(\boldsymbol{A},\boldsymbol{b})}\phi(\boldsymbol{z}) = \min_{q_k\in\mathcal{Q}_k}\boldsymbol{x}^T\boldsymbol{q}_k(\boldsymbol{A})\boldsymbol{A}\boldsymbol{q}_k(\boldsymbol{A})\boldsymbol{x}$$

- We can bound the objective based on the eigenvalues of $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$ using the identity $\boldsymbol{p}(\boldsymbol{A}) = \boldsymbol{Q}\boldsymbol{p}(\boldsymbol{\Lambda})\boldsymbol{Q}^T$,

$$\begin{aligned}\phi(\boldsymbol{z}) &= \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{q}_k(\boldsymbol{\Lambda})\boldsymbol{\Lambda}\boldsymbol{q}_k(\boldsymbol{\Lambda})\boldsymbol{Q}^T\boldsymbol{x}\\ &\le \max_{\lambda_i\in\lambda(\boldsymbol{A})}(q_k(\lambda_i)^2)\underbrace{\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T\boldsymbol{x}}_{\phi(\boldsymbol{x}_0)}\end{aligned}$$

Conjugate Gradient Convergence Analysis (III)

- Using our bound on the square of the residual norm $\phi(\boldsymbol{z})$, we can see why CG converges after $m-1$ iterations if there are only $m < n$ distinct eigenvalues

$$\phi(x_k) = \min_{q_k \in \mathcal{Q}_k} \phi(\boldsymbol{z}) \leq \min_{q_k \in \mathcal{Q}_k} \max_{\lambda_i \in \lambda(\boldsymbol{A})} (q_k(\lambda_i)^2)\phi(\boldsymbol{x}_0)$$

consequently, the residual norm $\|r_k\|_{\boldsymbol{A}^{-1}} = \sqrt{\phi(x_k)}$ *decreases as*

$$\frac{\|r_k\|_{\boldsymbol{A}^{-1}}}{\|r_0\|_{\boldsymbol{A}^{-1}}} \leq \min_{q_k \in \mathcal{Q}_k} \max_{\lambda_i \in \lambda(\boldsymbol{A})} |q_k(\lambda_i)|$$

- To see that the residual goes to 0, we find a suitable polynomial in $\mathcal{Q}_m$ (the set of polynomials q_m of degree m with $q_m(0) = 1$)
  - *Specifically, we select* q_m *to be zero at each distinct eigenvalue* $\lambda_1, \ldots, \lambda_m$ *of* $\boldsymbol{A}$

$$q_m(\xi) = \frac{\prod_{j=1}^{m}(\lambda_i - \xi)}{\prod_{j=1}^{m} \lambda_i}$$

  while also satisfiying $q_m(0) = 1$
  - *This polynomial implies that* $\|r_m\| = \phi(\boldsymbol{x}_m) = 0$ *since* $\max_{\lambda_i \in \lambda(\boldsymbol{A})} q_m(\lambda_i)^2 = 0$

Round-off Error in Conjugate Gradient

- CG provides strong convergence guarantees for SPD matrices in exact arithmetic
  - *Classically, CG was viewed as a direct method, since its guaranteed to convergence in n iterations*
  - *In practice, round-off error prevents CG from achieving this for realistic matrices, so CG was actually abandoned for a while due to being viewed as unstable*
  - *Later, it was realized that CG is highly competitive as an iterative method*
- Due to round-off CG may stagnate / have plateaus in convergence
  - *A formal analysis of round-off error[2] reveals that CG with round-off is equivalent to exact CG on a matrix of larger dimension, whose eigenvalues are clustered around those of $\boldsymbol{A}$*
  - *Using this view, CG convergence plateaus may be explained by the polynomial q_k developing more and more zeros near the same eigenvalue of $\boldsymbol{A}$*

[2] A. Greenbaum and Z. Strakos, SIMAX 1992

Preconditioning

- Convergence of iterative methods for $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ depends on $\kappa(\boldsymbol{A})$, the goal of a preconditioner $\boldsymbol{M}$ is to obtain $\boldsymbol{x}$ by solving

$$\boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{b}$$

  with $\kappa(\boldsymbol{M}^{-1}\boldsymbol{A}) < \kappa(\boldsymbol{A})$
  - *need not form $\boldsymbol{M}^{-1}\boldsymbol{A}$ but only compute matrix-vector products $\boldsymbol{M}^{-1}(\boldsymbol{A}\boldsymbol{x})$*
  - *want $\boldsymbol{M}^{-1}\boldsymbol{x}$ to be easy to compute (easier than $\boldsymbol{A}^{-1}\boldsymbol{x}$)*
  - *so generally one extracts some $\boldsymbol{M} \approx \boldsymbol{A}$ that is easy to solve linear systems with*
- Common preconditioners select parts of $\boldsymbol{A}$ or perform inexact factorization
  - *(block-)Jacobi preconditioner takes $\boldsymbol{M}$ to be (block-)diagonal of $\boldsymbol{A}$*
  - *incomplete LU (ILU) preconditioners compute $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{U} \approx \boldsymbol{A}$ (+pivoting)*
  - *ILU variants constraint sparsity of $\boldsymbol{L}$ and $\boldsymbol{U}$ factors during factorization to be the same or not much more than that of $\boldsymbol{A}$*
  - *good problem-specific preconditioners are often available in practice and theory, applying also to problems beyond linear systems (eigenvalue problems, optimization, approximate graph algorithms)*

Newton's Method

- Newton's method in n dimensions is given by finding minima of n-dimensional quadratic approximation using the gradient and Hessian of f:

$$f(\boldsymbol{x}_k + \boldsymbol{s}) \approx \hat{f}(\boldsymbol{s}) = f(\boldsymbol{x}_k) + \boldsymbol{s}^T \nabla f(\boldsymbol{x}_k) + \frac{1}{2} \boldsymbol{s}^T \boldsymbol{H}_f(\boldsymbol{x}_k) \boldsymbol{s}.$$

The minima of this function can be determined by identifying critical points

$$\mathbf{0} = \nabla \hat{f}(\boldsymbol{s}) = \nabla f(\boldsymbol{x}_k) + \boldsymbol{H}_f(\boldsymbol{x}_k) \boldsymbol{s},$$

thus to determine $\boldsymbol{s}$ we solve the linear system,

$$\boldsymbol{H}_f(\boldsymbol{x}_k) \boldsymbol{s} = -\nabla f(\boldsymbol{x}_k).$$

Assuming invertibility of the Hessian, we can write the Newton's method iteration as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \underbrace{\boldsymbol{H}_f(\boldsymbol{x}_k)^{-1} \nabla f(\boldsymbol{x}_k)}_{\boldsymbol{s}}.$$

Quadratic convergence follows by equivalence to Newton's method for solving nonlinear system of optimality equations $\nabla f(\boldsymbol{x}) = \mathbf{0}$.

Nonlinear Least Squares

- An important special case of multidimensional optimization is *nonlinear least squares*, the problem of fitting a nonlinear function $f_{\boldsymbol{x}}(t)$ so that $f_{\boldsymbol{x}}(t_i) \approx y_i$:
  For example, consider fitting $f_{[x_1,x_2]}(t) = x_1 \sin(x_2 t)$ *so that*

$$\begin{bmatrix} f_{[x_1,x_2]}(1.5) \\ f_{[x_1,x_2]}(1.9) \\ f_{[x_1,x_2]}(3.2) \end{bmatrix} \approx \begin{bmatrix} -1.2 \\ 4.5 \\ 7.3 \end{bmatrix}.$$

- We can cast nonlinear least squares as an optimization problem to minimize residual error and solve it by Newton's method:
  Define residual vector function $\boldsymbol{r}(\boldsymbol{x})$ *so that* $r_i(\boldsymbol{x}) = y_i - f_{\boldsymbol{x}}(t_i)$ *and minimize*

$$\phi(\boldsymbol{x}) = \frac{1}{2}||\boldsymbol{r}(\boldsymbol{x})||_2^2 = \frac{1}{2}\boldsymbol{r}(\boldsymbol{x})^T\boldsymbol{r}(\boldsymbol{x}).$$

  Now the gradient is $\nabla\phi(\boldsymbol{x}) = \boldsymbol{J}_{\boldsymbol{r}}^T(\boldsymbol{x})\boldsymbol{r}(\boldsymbol{x})$ *and the Hessian is*

$$\boldsymbol{H}_\phi(\boldsymbol{x}) = \boldsymbol{J}_{\boldsymbol{r}}^T(\boldsymbol{x})\boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{x}) + \sum_{i=1}^{m} r_i(\boldsymbol{x})\boldsymbol{H}_{r_i}(\boldsymbol{x}).$$

Gauss-Newton Method

- The Hessian for nonlinear least squares problems has the form:

$$\boldsymbol{H}_\phi(\boldsymbol{x}) = \boldsymbol{J}_{\boldsymbol{r}}^T(\boldsymbol{x})\boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{x}) + \sum_{i=1}^{m} r_i(\boldsymbol{x})\boldsymbol{H}_{r_i}(\boldsymbol{x}).$$

The second term is small when the residual function $\boldsymbol{r}(\boldsymbol{x})$ is small, so approximate

$$\boldsymbol{H}_\phi(\boldsymbol{x}) \approx \hat{\boldsymbol{H}}_\phi(\boldsymbol{x}) = \boldsymbol{J}_{\boldsymbol{r}}^T(\boldsymbol{x})\boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{x}).$$

- The *Gauss-Newton* method is Newton iteration with an approximate Hessian:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \hat{\boldsymbol{H}}_\phi(\boldsymbol{x}_k)^{-1}\nabla f(\boldsymbol{x}_k) = \boldsymbol{x}_k - (\boldsymbol{J}_{\boldsymbol{r}}^T(\boldsymbol{x}_k)\boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{x}_k))^{-1}\boldsymbol{J}_{\boldsymbol{r}}^T(\boldsymbol{x}_k)\boldsymbol{r}(\boldsymbol{x}_k).$$

Recognizing the normal equations, we interpret the Gauss-Newton method as solving linear least squares problems $\boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{x}_k)\boldsymbol{s}_k \cong \boldsymbol{r}(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{s}_k$.

Constrained Optimization Problems

- We now return to the general case of *constrained* optimization problems:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{g}(\boldsymbol{x}) = \mathbf{0} \quad \text{and} \quad \boldsymbol{h}(\boldsymbol{x}) \leq \mathbf{0}$$

  When f is quadratic, while h, g is linear, this is a quadratic optimization problem.

- Generally, we will seek to reduce constrained optimization problems to a series of simpler optimization problems:
  - *sequential quadratic programming: solve a series of constrained quadratic optimization problems*
  - *interior point methods: solve a series of more complicated (more ill-conditioned) unconstrained optimization problems*

Lagrangian Duality

- The Lagrangian function with constraints $\boldsymbol{g}(\boldsymbol{x}) = \mathbf{0}$ and $\boldsymbol{h}(\boldsymbol{x}) \leq \mathbf{0}$ is

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \begin{bmatrix} \boldsymbol{h}(\boldsymbol{x}) \\ \boldsymbol{g}(\boldsymbol{x}) \end{bmatrix}$$

The constrained minima of $f(\boldsymbol{x})$ must be saddle points of the Lagrangian function

- The Lagrangian dual problem is an unconstrained optimization problem:

$$\max_{\boldsymbol{\lambda}} q(\boldsymbol{\lambda}), \quad q(\boldsymbol{\lambda}) = \begin{cases} \min_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) & \text{if } \boldsymbol{\lambda} \geq \mathbf{0} \\ -\infty & \text{otherwise} \end{cases}$$

The unconstrained optimality condition $\nabla q(\boldsymbol{\lambda}^*) = \mathbf{0}$, implies

$$\max \left(\boldsymbol{\lambda}^*, \begin{bmatrix} \boldsymbol{h}(\boldsymbol{x}) \\ \boldsymbol{g}(\boldsymbol{x}) \end{bmatrix} \right) = \mathbf{0}$$

when $\lambda_i^* = 0$, we say the ith constraint is inactive at the minimum point.

Sequential Quadratic Programming

- *Sequential quadratic programming (SQP)* reduces a nonlinear equality constrained problem to a sequence of constrained quadratic programs via a Taylor expansion of the Lagrangian function $\mathcal{L}_f(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x})$:

$$q(\boldsymbol{x}_k + \boldsymbol{s}, \boldsymbol{\lambda}_k + \boldsymbol{\delta}) = \mathcal{L}_f(\boldsymbol{x}_k, \boldsymbol{\lambda}_k) + \boldsymbol{s}^T(\nabla f(\boldsymbol{x}_k) + \boldsymbol{J}_{\boldsymbol{g}}^T(\boldsymbol{x}_k)\boldsymbol{\lambda}_k) + \frac{1}{2}\boldsymbol{s}^T \boldsymbol{B}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k)\boldsymbol{s} + \boldsymbol{\delta}^T(\boldsymbol{J}_{\boldsymbol{g}}(\boldsymbol{x}_k)\boldsymbol{s} + \boldsymbol{g}(\boldsymbol{x}_k))$$

  where $\boldsymbol{B}(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{H}_f(\boldsymbol{x}) + \sum_{i=1}^m \lambda_i \boldsymbol{H}_{g_i}(\boldsymbol{x})$

- SQP ignores the constant term $\mathcal{L}_f(\boldsymbol{x}_k, \boldsymbol{\lambda}_k)$ and minimizes $\boldsymbol{s}$ while treating $\boldsymbol{\delta}$ as a Lagrange multiplier:

  The above unconstrained quadratic program corresponds to the Lagrangian form of the constrained quadratic program

$$\max_{\boldsymbol{s}} \boldsymbol{s}^T(\nabla f(\boldsymbol{x}_k) + \boldsymbol{J}_{\boldsymbol{g}}^T(\boldsymbol{x}_k)\boldsymbol{\lambda}_k) + \frac{1}{2}\boldsymbol{s}^T \boldsymbol{B}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k)\boldsymbol{s}$$

  with constraint $\boldsymbol{J}_{\boldsymbol{g}}(\boldsymbol{x}_k)\boldsymbol{s} = -\boldsymbol{g}(\boldsymbol{x}_k)$.

Interior Point Methods

- Barrier functions provide an effective way of working with inequality constraints $\boldsymbol{h}(\boldsymbol{x}) \leq \mathbf{0}$:

  Inverse barrier function:

  $$\phi_\mu(\boldsymbol{x}) = f(\boldsymbol{x}) - \mu \sum_{i=1}^{m} \frac{1}{h_i(\boldsymbol{x})}$$

  Logarithmic barrier function:

  $$\phi_\mu(\boldsymbol{x}) = f(\boldsymbol{x}) - \mu \sum_{i=1}^{m} \log(-h_i(\boldsymbol{x}))$$

  in theory with sufficiently small steps we have $\boldsymbol{x}_\mu^* \to \boldsymbol{x}^*$ *as* $\mu \to 0$
- Interior point methods additionally incorporate Lagrangian optimization
  - *can be combined with SQP or alternating minimization*
  - *slack variables with nonnegativity constraints reduce general inequality constraints to nonnegativity and equality constraints*
  - *optimality conditions for augmented Lagrangian conditions yield linear system*
  - *conditioning of interior point linear systems suffers as* μ *decreases*