Computational Methods

Least Squares Approximation/Optimization

Least Squares

- Least squares methods are aimed at finding approximate solutions when no precise solution exists
  - Find the solution that minimizes the residual error in the system
- Least squares can be used to fit a model to noisy data points or to fit a simpler model to complex data
  - Amounts to projecting higher dimensional data onto a lower-dimensional space.

Linear Least Squares

- Linear least squares attempts to find a least squares solution for an overdetermined linear system (i.e. a linear system described by an *m x n* matrix *A* with more equations than parameters).

$$A\vec{x} \cong \vec{b}$$

  - Least squares minimizes the squared Eucliden norm of the residual $\min_{\vec{x}} \left\| \vec{b} - A\vec{x} \right\|_2^2 = \min_{\vec{x}} \left\| \vec{r} \right\|_2^2$
  - For data fitting on *m* data points using a linear combination of basis functions this corresponds to

$$\min_{\alpha} \sum_{i=1}^{m} (y_m - \sum_{j=1}^{n} \alpha_j \phi_j(x_i))^2$$

Existence and Uniqueness

- Linear least squares problem always has a solution
- Solution is unique if and only if A has full rank, i.e. *rank(A)=n*
  - If rank(A) < n then A is rank-deficient and the solution of the least squares problem is not unique
- If solution is unique the residual vector can be expressed through A

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

  - This is minimized if its derivative is 0

$$-2A^T b + 2A^T Ax = 0$$

Normal Equations

- Optimization reduces to a *n x n* system of (linear) normal equations

$$A^T A x = A^T b$$

  - Linear least squares can be found by solving this system of linear equations
  - Solution can also be found through the Pseudo Inverse

$$Ax \cong b \quad \Rightarrow \quad x = A^+ b$$

$$A^+ = (A^T A)^{-1} A^T$$

  - Condition number with respect to A can be expressed as in the case of solving linear equations

$$cond(A) = \|A\|_2 \|A^+\|_2$$

Data Fitting

- A common use for linear least squares solution is to fit a given type of function to noisy data points
  - n^{th} order polynomial fit using monomial basis:

$$A\vec{\alpha} = \begin{pmatrix} 1 & x_1 & \cdots & x_1^n \\ 1 & x_2 & \cdots & x_2^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & \cdots & x_m^n \end{pmatrix} \vec{\alpha} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

    - Solving the system of equations provides the best fit in terms of the Euclidian norm

$$A^T A\alpha = A^T y$$

Condition Number and Sensitivity

- Sensitivity also depends on b
  - Influence of b can be expressed in terms of an angle between b and y

$$\cos(\theta) = \frac{\|y\|_2}{\|b\|_2} = \frac{\|Ax\|_2}{\|b\|_2}$$

  - Bound on the error in the solution due to perturbation in b can be expressed as

$$\frac{\|\Delta x\|_2}{\|x\|_2} \leq cond(A) \frac{1}{\cos(\theta)} \frac{\|\Delta b\|_2}{\|b\|_2}$$

Condition Number and Sensitivity

- Bound on the error in the solution due to perturbation in A can be expressed as

$$\frac{\|\Delta x\|_2}{\|x\|_2} \leq ((cond(A))^2 \tan(\theta) + cond(A)) \frac{\|\Delta A\|_2}{\|A\|_2}$$

  - For small residuals the condition number for least squares is approximately cond(A).
  - For large residuals the condition number can be square of worse

Condition Number and Sensitivity

- Conditioning of normal equation solution is

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

  - For large systems of equation the condition number of the formulation using normal equations (or the Pseudoinverse) increases rapidly
  - Much of the increased sensitivity is due to the need for multiplying A and A^T in order to be able to apply a solution algorithm for the system of equations
    - Conditioning of the normal equations is potentially significantly worse than the conditioning of the original system
    - Algorithm is not very stable for large numbers of equations/data points

Augmented System Method

- Augmented system method transforms least square problem into an system of equation solving problem by adding equations and can be used to improve the conditioning
  - Increase matrix size to a square *(m+n)x(m+n)* matrix by including the residual equations

$$\begin{aligned} r + Ax &= b \\ A^T r &= 0 \end{aligned} \quad \Rightarrow \quad \begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

    - Greater freedom to choose pivots and maintain stability
    - Substantially higher complexity for systems with *m>>n* due to the need to solve a *(n+m)x(n+m)* system

QR Factorization

- The Augmented System method addresses stability but adds very high cost since it expands the matrix
- QR factorization changes the system matrix into solvable form without computation of A^TA and without expanding the matrix

$$A = Q\begin{pmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,n} \\ 0 & R_{2,2} & \cdots & R_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & R_{n,n} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} = Q\begin{pmatrix} R \\ 0 \end{pmatrix} \quad \Rightarrow \quad Q^T A x = Q^T b$$

QR Factorization

- The important part of *Q* for the solution consists of the first n rows since the others are multiplied by *0*

$$Q^T = (Q_1 Q_2) \quad \Rightarrow \quad Q_1^T Ax = Rx = Q_1^T b$$

- QR factorization factors the system matrix *A* into *Q* an *R* where R is upper triangular
  - As in LU Factorization, Q^T represents a sequence of solution-preserving transformations
    - In LU Factorization only identity has to be preserved which can be done using elimination matrices
    - In QR Factorization the least square characteristic has to be preserved which requires the use of orthogonal transformations
  - R is an upper triangular matrix that can be used for backward substitution to compute the solution

Orthogonal Transformations

- Orthogonal transformations preserve the least square solution

$$QQ^T = I$$

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

- For QR factorization we need to find a set of orthogonal transformations that can transform *A* into an upper triangular matrix R and that are numerically stable
  - Householder transforms
  - Givens rotations
  - Gram-Schmidt orthogonalization

QR Factorization with Householder Transformations

- Householder transformations allow to zero all entries below a chosen point in a vector *a*
  - Applying this consecutively for every column of the matrix *A*, choosing the diagonal element as the one below which everything is to be zeroed out we can construct *R*
  - Householder transformation can be computed from a given vector (vector of column values), *a*, setting all the values that are not to be changed (entries above the diagonal) to 0

$$Q = H = I - 2\frac{vv^T}{v^T v} \quad , \quad v = a - \beta e_i \quad , \quad \beta = \pm\|a\|_2$$

    - The sign for β can be chosen to avoid cancellation (loss of significance) in the computation of *v*
    - H is orthogonal and symmetric: $H = H^T = H^{-1}$

QR Factorization with Householder Transformations

- In QR factorization with Householder transforms, successive columns are adjusted to upper triangular by zeroing all entries below the diagonal element.
  - Householder transform does not affect the entries in the columns to the left and the rows above the currently chosen diagonal element since it contains only 0s in the rows below the chosen diagonal term.
    - Applying the Householder transform only to the columns to the right saves significant processing time

$$H\vec{a}_j = \left(I - 2\frac{vv^T}{v^Tv}\right)\vec{a}_j = \vec{a}_j - 2\frac{vv^T}{v^Tv}\vec{a}_j = \vec{a}_j - 2v\frac{v^T\vec{a}_j}{v^Tv} = \vec{a}_j - 2\frac{v^T\vec{a}_j}{v^Tv}v$$

    - Applying it to individual columns also eliminates the need to compute full matrix *H* - only $v = \vec{a}_j - \beta e_j$ is needed

QR Factorization Example

- Quadratic polynomial fit to 5 data points

  $$Data: \quad (-1,1),(-0.5,0.5),(0,0),(0.5,0.5),(1,2)$$

  - Design linear system for data fitting

  $$A\vec{\alpha} = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \\ 1 & x_5 & x_5^2 \end{pmatrix} \vec{\alpha} \cong \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} 1 & -1 & 1 \\ 1 & -0.5 & 0.25 \\ 1 & 0 & 0 \\ 1 & 0.5 & 0.25 \\ 1 & 1 & 1 \end{pmatrix} \vec{\alpha} \cong \begin{pmatrix} 1 \\ 0.5 \\ 0 \\ 0.5 \\ 2 \end{pmatrix}$$

  - Starting with the first column start eliminating entries below the diagonal using appropriate Householder transforms choosing the sign on β to avoid cancellation

QR Factorization Example

- Householder elimination by column
  - 1st column:

$$v_1 = \vec{a}_1 - \|\vec{a}_1\|_2 e_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} + \sqrt{5}\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 3.236 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

    - Negative sign on β because potentially cancelling term is +1

$$H_1 A = \begin{pmatrix} -2.236 & 0 & -1.118 \\ 0 & -0.191 & -0.405 \\ 0 & 0.309 & -0.655 \\ 0 & 0.809 & -0.405 \\ 0 & 1.309 & 0.345 \end{pmatrix} \;,\; H_1 \vec{y} = \begin{pmatrix} -1.789 \\ -0.362 \\ -0.862 \\ -0.362 \\ 1.138 \end{pmatrix}$$

QR Factorization Example

- Householder elimination by column
  - 2nd column:

$$v_2 = \vec{a}_2 - \|\vec{a}_2\|_2 e_2 = \begin{pmatrix} 0 \\ -0.191 \\ 0.309 \\ 0.809 \\ 1.309 \end{pmatrix} - \sqrt{2.5}\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ -1.772 \\ 0.309 \\ 0.809 \\ 1.309 \end{pmatrix}$$

    - Positive sign on β because possibly cancelling term is -0.191

$$H_2 H_1 A = \begin{pmatrix} -2.236 & 0 & -1.118 \\ 0 & 1.581 & 0 \\ 0 & 0 & -0.725 \\ 0 & 0 & -0.589 \\ 0 & 0 & 0.047 \end{pmatrix} \quad , \quad H_2 H_1 \vec{y} = \begin{pmatrix} -1.789 \\ 0.632 \\ -1.035 \\ -0.816 \\ 0.404 \end{pmatrix}$$

QR Factorization Example

- Householder elimination by column
  - 3rd column:

$$v_2 = \vec{a}_3 - \left\|\vec{a}_3\right\|_2 e_3 = \begin{pmatrix} 0 \\ 0 \\ -0.725 \\ -0.589 \\ 0.047 \end{pmatrix} - \sqrt{0.875}\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -1.66 \\ -0.589 \\ 0.047 \end{pmatrix}$$

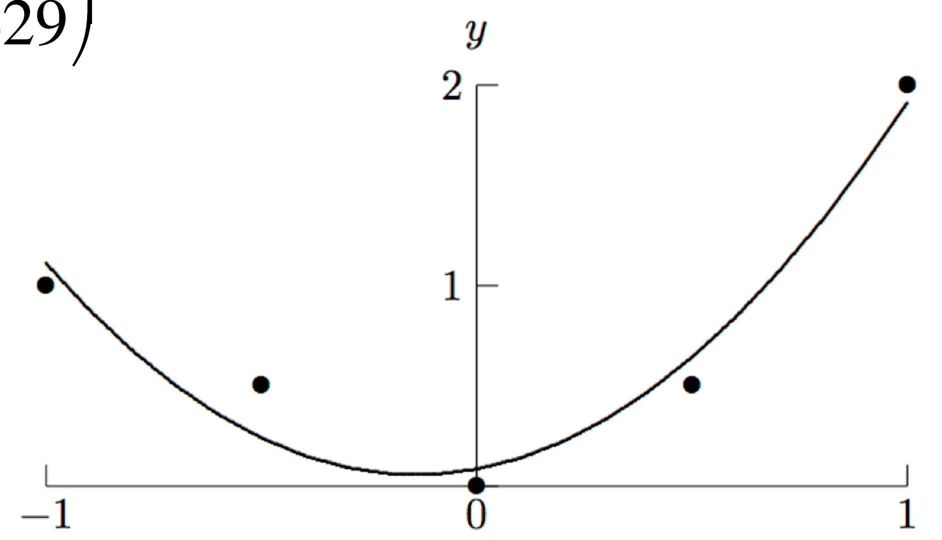
    - Positive sign on β because possibly cancelling term is -0.725

$$H_3H_2H_1A = \begin{pmatrix} -2.236 & 0 & -1.118 \\ 0 & 1.581 & 0 \\ 0 & 0 & 0.935 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad , \quad H_3H_2H_1\vec{y} = \begin{pmatrix} -1.789 \\ 0.632 \\ 1.336 \\ 0.026 \\ 0.337 \end{pmatrix}$$

QR Factorization Example

- Backward substitution with the upper triangular matrix yields the parameters and least squares fit second order polynomial

$$\vec{\alpha} = \begin{pmatrix} 0.086 \\ 0.4 \\ 1.429 \end{pmatrix} \qquad p(x) = 0.086 + 0.4x + 1.429x^2$$



Other Orthogonal Transformations

- Other transformations can be used for QR Factorization
  - Givens rotations
  - Gram-Schmidt Orthogonalization
- Householder Transformations generally achieve the best performance and stability tradeoff
  - Complexity of QR Factorization with Householder transformations is approximately $mn^2 - n^3/3$ multiplications
    - Depending on the size of m (data points / equations) this is between the same and two times the work of normal equations
  - Conditioning of QR Factorization with Householder transformations is optimal

$$cond(A) + \|r\|_2 (cond(A))^2$$

QR Factorization

- Transformations for QR Factorization are numerically more stable than elimination steps in LU Factorization
  - Choice of sign in Householder transformations allows to avoid cancellation and thus instabilities in individual transforms
    - Row Pivoting is usually not necessary
- Stability leads to QR factorization also frequently being used instead of LU Factorization to solve nonsingular systems of linear equations
  - Increased complexity is traded off against stability (and thus precision) of the solution

Nonlinear Least Squares

- As for equation solving, finding solutions for general nonlinear systems requires iterative solutions
  - Goal is to find the approximate solution with the smallest square residual, ρ , for the system of functions *f(x)*

$$r_i(x) = b_i - f_i(x)$$

$$\rho(x) = \frac{1}{2} r^T(x) r(x)$$

  - At the minimum, the gradient of the square residual function, would be 0 $$\nabla \rho(x) = J_r^T(x) r(x) = 0$$
    - Could use Multivariate Newton method to find the root of the derivative which would require second derivative, the Hessian of the square error

$$H_\rho(x) = J_r^T(x) J_r(x) + \sum_{i=1}^{m} r_i(x) H_{r_i}(x)$$

Gauss-Newton Method

- Multivariate Newton method would require solving the following linear system in each iteration

$$\left(J_r^T(x)J_r(x) + \sum_{i=1}^{m} r_i(x)H_{r_i}(x)\right)s = -J_r^T(x)r(x)$$

  - Requires frequent computation of the Hessian which is expensive and reduces stability
- Gauss-Newton avoids this by dropping second order term

$$J_r^T(x)J_r(x)s = -J^T(x)r(x)$$

  - This is best solved by converting this into the corresponding least squares problem and using QR factorization

$$J_r(x)s \cong -r(x)$$

  - Once solved, Gauss-Newton operates like Multivariate Newton

$$x_{t+1} = x_t + s$$

Gauss-Newton Method

- Gauss-Newton method replaces nonlinear least squares problem with a sequence of linear least squares problems that converge to solution of nonlinear system
  - Converges if residual at the solution is not too large
    - Large residual at solution can lead to large values in Hessian, potentially leading to slow convergence or, in extreme cases, non-convergence
  - If it does not converge (large residual at solution) other methods have to be used
    - Levenberg-Marquardt method which uses an additional scalar parameter (and a separate, function-specific strategy to choose it) to modify step size

$$\left(J_r^T(x)J_r(x)\_\mu I\right)s = -J_r^T(x)r(x) \quad \Rightarrow \quad \begin{pmatrix} J_r(x) \\ \sqrt{\mu I} \end{pmatrix} s \cong \begin{pmatrix} -r(x) \\ 0 \end{pmatrix}$$

    - General optimization using the complete Hessian
      - Significant increase in computational complexity

Gauss-Newton Example

- Fit exponential function to data

$$Data: \quad (0,2),(1,0.7),(2,0.3),(3,0.1)$$

$$f(\alpha,x) = \alpha_1 e^{\alpha_2 x}$$

  - Residual function for data fitting is given by

$$r(\alpha) = \begin{pmatrix} 2 - f(\alpha,0) \\ 0.7 - f(\alpha,1) \\ 0.3 - f(\alpha,2) \\ 0.1 - f(\alpha,3) \end{pmatrix}$$

    - Resulting Jacobian is

$$J_r(\alpha) = \begin{pmatrix} -1 & 0 \\ -e^{\alpha_2} & -\alpha_1 e^{\alpha_2} \\ -e^{2\alpha_2} & -2\alpha_1 e^{2\alpha_2} \\ -e^{3\alpha_2} & -3\alpha_1 e^{3\alpha_2} \end{pmatrix}$$

Gauss-Newton Example

- First iteration starting at $\alpha^{(0)}=(1\ 0)^T$
  - Initial square residual:

    $$\left\| r\begin{pmatrix}1\\0\end{pmatrix}\right\|_2 = \sum_{i=1}^{4}\left(y_i - 1e^0\right)^2 = 2.39$$

  - Solve for step

    $$J_r\left(\begin{pmatrix}1\\0\end{pmatrix}\right)s = \begin{pmatrix}-1 & 0\\-1 & -1\\-1 & -2\\-1 & -3\end{pmatrix}s \cong \begin{pmatrix}-1\\0.3\\0.7\\0.9\end{pmatrix} \quad\Rightarrow\quad s = \begin{pmatrix}0.69\\-0.61\end{pmatrix}$$

  - Update parameter vector for next iteration

    $$\alpha^{(1)} = \begin{pmatrix}1\\0\end{pmatrix} + \begin{pmatrix}0.69\\-0.61\end{pmatrix} = \begin{pmatrix}1.69\\-0.69\end{pmatrix}$$

Gauss-Newton Example

- Second Iteration
  - Square residual

$$\left\| r\begin{pmatrix} 1.69 \\ -0.61 \end{pmatrix} \right\|_2 = \sum_{i=1}^{4} \left(y_i - 1.69e^{-0.61x_i} \right)^2 = 0.212$$

  - Solve for next step

$$J_r\left(\begin{pmatrix} 1.69 \\ -0.61 \end{pmatrix}\right) s = \begin{pmatrix} -1 & 0 \\ -0.543 & -0.918 \\ -0.295 & -0.998 \\ -0.16 & -0.813 \end{pmatrix} s \cong \begin{pmatrix} -0.31 \\ 0.218 \\ 0.199 \\ 0.171 \end{pmatrix} \quad \Rightarrow \quad s = \begin{pmatrix} 0.285 \\ -0.32 \end{pmatrix}$$

  - Update parameter vector for next iteration

$$\alpha^{(2)} = \begin{pmatrix} 1.69 \\ -0.61 \end{pmatrix} + \begin{pmatrix} 0.285 \\ -0.32 \end{pmatrix} = \begin{pmatrix} 1.975 \\ -0.93 \end{pmatrix}$$

Gauss-Newton Example

- Third Iteration
  - Square residual

$$\left\| r\begin{pmatrix} 1.975 \\ -0.93 \end{pmatrix} \right\|_2 = \sum_{i=1}^{4} \left(y_i - 1.975e^{-0.93x_i} \right)^2 = 0.007$$

  - Solve for next step

$$J_r\left(\begin{pmatrix} 1.975 \\ -0.93 \end{pmatrix}\right)s = \begin{pmatrix} -1 & 0 \\ -0.395 & -0.779 \\ -0.156 & -0.615 \\ -0.061 & -0.364 \end{pmatrix} s \cong \begin{pmatrix} -0.025 \\ 0.079 \\ 0.007 \\ 0.02 \end{pmatrix} \quad \Rightarrow \quad s = \begin{pmatrix} 0.019 \\ -0.074 \end{pmatrix}$$

  - Update parameter vector for next iteration

$$\alpha^{(3)} = \begin{pmatrix} 1.975 \\ -0.93 \end{pmatrix} + \begin{pmatrix} 0.019 \\ -0.074 \end{pmatrix} = \begin{pmatrix} 1.994 \\ -1.004 \end{pmatrix}$$

Least Squares Approximation

- Least Squares approximation is used to determine approximate solutions for a system of equation or to fit an approximate function to a set of data points
  - As opposed to interpolation data points are not met precisely
    - Applicable to noisy data points
    - Allows less complex function to be fitted to the data points
  - Least squares for linear functions has direct solution methods
    - Normal equations method not very stable for large numbers of equations
    - QR Factorization provides a more stable alternative that can also be used for equation solving instead of LU factorization
  - Least squares for nonlinear systems requires iterative solution
    - Gauss-Newton provides robust solution for problems with small residual
    - Otherwise general, more expensive optimization methods have to be used