CS 70 - Foundations Of Applied Computer Science

Carter Kruse

Reading Assignment 4

CS 70 - Chapter 6 (Least Squares)

Overview: We now turn out attention to linear systems $A\boldsymbol{x} = \boldsymbol{b}$ where A is a "tall" $m \times n$ matrix (with $m > n$), $\boldsymbol{x}$ is an n-vector of unknowns, and $\boldsymbol{b}$ is an m-vector.

$$\begin{bmatrix} | & & | \\ \boldsymbol{a}_1 & \cdots & \boldsymbol{a}_n \\ | & & | \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \cdots \\ \boldsymbol{x}_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \cdots \\ \boldsymbol{b}_n \end{bmatrix}$$

Such systems are said to be *over-determined* or *over-constrained* because they have more equations/constraints (m) than unknowns (n).

The Least Squares Approach: Instead of requiring no error for some constraints while allowing large error in others, it is often better to consider all m constraints and minimize some sort of average error. In order words, we could compromise and try to find an approximate solution to the tall system so that $A\boldsymbol{x} \approx \boldsymbol{b}$.

The Residual Vector: How far off would our approximate solution be? We can quantify this by looking at the difference (or error) between the right-hand side and the left-hand side. We call this error the residual and denote it $\Delta = \boldsymbol{b} - A\boldsymbol{x}$.

Given Δ, it seems reasonable to try to make it as small as possible. So one mathematically convenient and intuitive notion of average error is the squared norm

$$|\boldsymbol{b} - A\boldsymbol{x}|^2 = |\Delta|^2 = \Delta_1^2 + \cdots + \Delta_m^2$$

So, from all possible m-vectors $\boldsymbol{x}$, we could seek the one that makes $|\Delta|$ as small as possible. Mathematically, we can write this as

$$\hat{x} = argmin_x |\boldsymbol{b} - A\boldsymbol{x}|^2$$

where we denote this optimal approximate solution $\hat{x}$.

Geometric View: The Orthogonality Principle: A geometric interpretation in terms of the columns provides a particularly useful starting point: the smallest residual vector must be the one that is perpendicular to the span of A. In other words, Δ must be perpendicular to each column of A.

Algebraic View: Solution Via Calculus: Another way to derive the solution is to treat the residual as a function $E(\boldsymbol{x}) = |\boldsymbol{b} - A\boldsymbol{x}|^2$, and find the minimum of this function by setting its derivatives equal to zero.

General Solution: We turn out attention back to the general system $A\boldsymbol{x} = \boldsymbol{b}$ where A is an $m \times n$ matrix (with no restriction on m or n), $\boldsymbol{x}$ is an n-vector of unknowns, and $\boldsymbol{b}$ is an $\boldsymbol{m}$-vector. With n columns and unknowns, the principle of orthogonality we applied above leads to n orthogonality constraints, which we could express in exactly the same way, by requiring that the inner products between the columns of A and the residual are zero:

$$\begin{bmatrix} - & a_1^T & - \\ - & a_2^T & - \\ & \cdots & \\ - & a_m^T & - \end{bmatrix} (\boldsymbol{b} - A\boldsymbol{x}) = \boldsymbol{0}$$

This implies $A^T (\boldsymbol{b} - A\boldsymbol{x}) = \boldsymbol{0} \to A^T A\boldsymbol{x} = A^T \boldsymbol{b}$.

Normal Equations: The best least squares solution to the system $A\boldsymbol{x} \approx \boldsymbol{b}$, i.e. the $\boldsymbol{x}$ that minimizes the residual norm $|\boldsymbol{b} - A\boldsymbol{x}|$, satisfies

$$A^T A\boldsymbol{x} = A^T \boldsymbol{b}$$

$A^T A$ is a matrix - sometimes called the *Gramm matrix* - which is square. It has the same number of rows and columns as the number of unknowns in $\boldsymbol{x}$. The normal equations essentially convert tall over-constrained systems into square systems, allowing us to use algorithms for solving square systems to obtain the least squares solution.

Moore-Penrose Inverse: It is possible to prove that as long as the columns of A are linearly independent, then $A^T A$ will be invertible. This means we can express the least squares solution mathematically as

$$\hat{\boldsymbol{x}} = \left(A^T A\right)^{-1} A^T \boldsymbol{b}$$

This is called the *pseudo-inverse* because it acts much like the inverse of a square matrix, but generalized to a non-square matrix.

See the course notes for further information about the dot product, projection onto a line, and projection onto a subspace.

CS 70 - Chapter 7 (Data Fitting)

Data: We have a set of data samples $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$ in which x_i is a k-vector and y_i is a scalar.

Function: We believe x and y are related. Their relation can described by a function $f : \mathbb{R}^n \to \mathbb{R}$ that maps from x to y:

$$y = f(x)$$

Model: We do not know the expression of $f(x)$. So, we aim to approximate $f(x)$ by fitting a model $\hat{f}(x)$ between x and y:

$$\hat{y} = \hat{f}(x)$$

where $\hat{f} : \mathbb{R} \to \mathbb{R}$. the hat appearing over f is traditional notation that suggests that the function $\hat{f}$ is an approximation of the function f.

Linear Model & Basis: We will focus on a specific form for the model, which has the form

$$\hat{f}(x) = \theta_1 f_1(x) + \theta_2 f_2(x) + \cdots + \theta_m f_m(x)$$

where $f_i(x) : \mathbb{R}^k \to \mathbb{R}$ are basis functions or feature mappings that we choose, and θ_i are the model parameters that we choose.

Error: For data sample i, our model predicts the value $\hat{y}_i = \hat{f}(x_i)$, so the prediction error or residual for this data point is

$$\Delta_i = y_i - \hat{y}_i$$

Build Least-Squares Data-Fitting System: We solve a linear system in the least squares sense to find the best parameters $\theta_1, \theta_2, \ldots, \theta_n$ for the basis functions $f_1, f_2, \ldots, f_m$ given n data samples. The problem can be written in matrix form as $\boldsymbol{y} = A\boldsymbol{\theta}$.

$$\begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix} = \begin{bmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_m(x_1) \\ f_1(x_2) & f_2(x_2) & \cdots & f_m(x_2) \\ \cdots & \cdots & & \cdots \\ f_1(x_n) & f_2(x_n) & \cdots & f_m(x_n) \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \cdots \\ \theta_m \end{bmatrix}$$

This gives us a tall matrix A with its elements calculated by the basis functions taking different data samples x. The best values of $\boldsymbol{\theta}$ in the least squares sense can be calculated as $\theta = \left(A^T A\right)^{-1} A^T \boldsymbol{y}$. The least squares solution minimizes the L_2 norm of the residual $|\Delta| = |\boldsymbol{y} - \hat{\boldsymbol{y}}| = |y - A\boldsymbol{\theta}|$.

CS 70 - Chapter 8 (Gram-Schmidt & QR Decomposition)

Orthonormal Vectors: A collection of vectors $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n$ are orthonormal if

$$q_i^T q_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

If we write the collection of vectors as the columns of a matrix, i.e. $Q = [\boldsymbol{q}_1 \boldsymbol{q}_2 \ldots \boldsymbol{q}_n]$, then we can write this property succinctly as

$$Q^T Q = I$$

which implies that $Q^{-1} = Q^T$, indicating that we only need to perform a matrix transpose when we want to calculate the inverse of an orthonormal matrix.

Gram-Schmidt Algorithm: Idea: Incrementally build an orthonormal basis one vector at a time. For each newly added vector, we project out the components from the existing orthonormal basis vectors and normalize it after all the projections.

Algorithm: As summarized in the following pseudo-code, you (1) pick one vector each time, (2) project out all the components on the existing orthonormal basis vectors, and (3) normalize.

QR Decomposition: Idea: For a matrix A, conduct the Gram-Schmidt algorithm for the collection of its column vectors $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n$ to get the orthonormal matrix Q, and at the same time record all the coefficients during the procedure in another upper triangle matrix R.

Note that the QR decomposition can be done by generating each column of both Q and R in sequence, by following the iterative steps in the Gram-Schmidt algorithm. That is, in the first iteration, we calculate the first column of Q ($\boldsymbol{q}_1$) and the first column of R ($|\boldsymbol{q}_1|$). In the second iteration, we calculate the second column of Q (q_2) and the second column of R ($\left(\boldsymbol{a}_2^T \boldsymbol{q}_1\right) \boldsymbol{q}_1$ and $|\hat{\boldsymbol{q}}_2|$), etc.

QR decomposition can be used to solve least-squares problems. For a least-squares system $A\boldsymbol{x} = \boldsymbol{b}$, we first factorize A as $A = QR$, then

- Solve $Q\boldsymbol{y} = \boldsymbol{b}$ by $y = Q^T\boldsymbol{b}$ (recall that we have $Q^{-1} = Q^T$ for an orthogonal matrix).
- Solve $R\boldsymbol{x} = \boldsymbol{y}$ using backward substitution.