Polynomial interpolation

Vandermonde matrix systems

Monday, March 18th

Math 365

Week #9

Polynomial interpolation

Up to this point, we have been considering only finding curves that in some sense approximate the data. The assumption is that the data is noisy or imprecise, so curves that exactly fit the data would be of little value.

Now we take a different tack : Suppose we really want to find a curve that exactly goes through our data points. Perhaps we want to approximate a function with a polynomial, or we are looking to approximate our data in some precise sense.

Polynomial interpolation

Given $n+1$ data points (x_i, y_i), it is easy to write down the conditions that the data points must satisfy if we want to interpolate an n^{th} degree polynomial :

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

This leads to the system of n+1 equations :

$$P_n(x_i) = a_n x_i^n + a_{n-1} x_i^{n-1} + \ldots + a_1 x_i + a_0 = y_i$$

Such a system is called a *Vandermonde* system and can be written succinctly as

$$V\mathbf{a} = \mathbf{y}$$

where the solution $\mathbf{a}$ contains the coefficients a_0, a_1, ..., a_n.

Vandermonde matrix

The Vandermonde system looks very much like the system we constructed for linear least squares :

$$\begin{bmatrix} x_0^n & x_0^{n-1} & \dots & x_0 & 1 \\ x_1^n & x_1^{n-1} & \dots & x_1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_n^n & x_n^{n-1} & \dots & x_n & 1 \end{bmatrix} \begin{bmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

If we can invert the matrix, we can solve the system and find the unique polynomial that interpolates our data.

Inverting the Vandermonde matrix

How do we know we can solve this system?

Consider the 2×2 system for interpolating a line :

$$\begin{bmatrix} x_0 & 1 \\ x_1 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}$$

This system will have a unique solution if

$$\det(V) = (x_0 - x_1) \neq 0$$

$$P_1(x) = \left(\frac{y_0 - y_1}{x_0 - x_1} \right) x + \frac{y_1 x_0 - x_1 y_0}{x_0 - x_1}$$

m b

Inverting the Vandermonde system

Consider the 3×3 system for interpolating a second degree polynomial :

$$\begin{bmatrix} x_0^2 & x_0 & 1 \\ x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \end{bmatrix} \begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$$

Again, this system will have a unique solution if

$$\det(V) = (x_0 - x_1)(x_0 - x_2)(x_1 - x_2) \neq 0$$

or that the x_i's are distinct.

Inverting the Vandermonde system

In general, the Vandermonde system will be invertible if

$$\det(V) = \prod_{0 \leq i < j \leq n} (x_i - x_j) \neq 0$$

or if the x_i's are distinct.

This also demonstrates that the polynomial that interpolates n+1 distinct points is *unique*, and in theory at least, the coefficients can be written down as

$$\mathbf{a} = V^{-1}\mathbf{y}$$

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

In Matlab

In Matlab, there are at least two different ways to obtain these coefficients.

Construct the Vandermonde matrix system using **`vander`** and use the backslash to invert and solve for the coefficents.

Use **`polyfit`** to fit a polynomial of a given degree to your data.

Caution : the Vandermonde system becomes increasingly ill-conditioned as the polynomial size grows.

Sample code

labs/lab_9/lab_9.m