MAD exercise session 4

Lagrange Polynomials

Pascal Auf der Maur

Lagrange Polynomials

- Instead of fitting a function of lower order to the datapoints as in LSQ we want to fit a polynomial to the data of the exact same order
- N points to a polynomial of degree $N - 1$ (one point can not be fitted reasonably).
- The resulting function can be used for interpolation

Lagrange Polynomials - Usual approach

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^N \\ 1 & x_2 & x_2^2 & \cdots & x_2^N \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^N \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

$$f(x) = a_1 + a_2 x^2 + ... + a_N x^N$$

- The usual, naive approach needs us to solve a NxN system of equations
- The idea of Lagrange Polynomials is that we can change the basis to make the equation easier

Lagrange Polynomials

We construct our basis such that:

$$l_i(x_j) = \delta_{ij}$$

where l_i is a function of our basis, x_j a x-value of our dataset and δ_{ij} the Kronecker delta

- The basis function is only equal to one on the respective point we want it to fit
- The basis function is equal to zero for all other datapoints

Lagrange Polynomials

$$l_k(x) = \frac{(x - x_1)(x - x_2)\dots(x - x_{k-1})(x - x_{k+1})\dots(x - x_N)}{(x_k - x_1)(x_k - x_2)\dots(x_k - x_{k-1})(x_k - x_{k+1})\dots(x_k - x_N)}$$

Lagrange Polynomials

Therefore the interpolation function is:

$$f(x) = \sum_{k=1}^{N} y_k l_k(x)$$

We don't need to solve a system of equations to obtain the function.

Comments

- This method captures a lot of noise of the dataset
  - Use this only to interpolate in the middle of the dataset and not to extrapolate
- LSQ with a polynomial of $N - 1$ degree will result in the same polynomial
  - The function minimizes the error which will be zero since the function goes to every point
- The found Polynomial is the same as the one found in the system of equation