Summary

Lecture 14

Dr. Görkem Saygılı

Department of Biomedical Engineering
Ankara University

Numerical Methods, 2017-2018 Fall

Newton's Method:

Newton's Method uses an approximation based on Taylor Series Expansion. Compared to Bisection technique, it achieves fast convergence.
Assume a sequence, $x_0, x_1, x_2, ..., x_{n-1}, x_n, x_{n+1}, ...$, using Taylor Series Expansion and omitting second order terms (since higher order terms converges to zero), $f(x_{n+1})$ around f_x as:

$$f(x_{n+1}) = f(x_n) + (x_{n+1} - x_n)f'(x_n)$$
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{1}$$

System of Equations

Let's consider the following equations:

$$\begin{aligned} 2x_1 - 4x_2 + 2x_3 &= 0 \\ 3x_1 - 5x_2 + 4x_3 &= 5 \\ x_1 + 2x_2 + 2x_3 &= 11 \end{aligned}$$

How do we solve this equation? How can we find the values for x_1, x_2 and x_3 such that the equations are satisfied?

Cramer's Rule

For a linear system such as:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

The solution can be found using the determinant of A:

$$x_1 = \frac{\begin{vmatrix} b_1 & a_{12} & a_{13} \\ b_2 & a_{22} & a_{23} \\ b_3 & a_{32} & a_{33} \end{vmatrix}}{|A|}, x_2 = \frac{\begin{vmatrix} a_{11} & b_1 & a_{13} \\ a_{21} & b_2 & a_{23} \\ a_{31} & b_3 & a_{33} \end{vmatrix}}{|A|}, x_3 = \frac{\begin{vmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ a_{31} & a_{32} & b_3 \end{vmatrix}}{|A|}$$

Polynomial Interpolation

In general, we can use nth order polynomials for interpolation:

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + ... + a_n x^n$$

Langrange Interpolating Polynomials

To approximate a polynomial using k points (x_1, y_1), (x_2, y_2), ... ,(x_k, y_k), we can define Langrange basis $L(x)$ as:

$$L_i(x) = \frac{\prod_{j \neq i}(x - x_j)}{\prod_{j \neq i}(x_i - x_j)}$$

Langrage basis functions can be used to generate a polynomial that passes through the points (x_1, y_1), (x_2, y_2), ... ,(x_k, y_k):

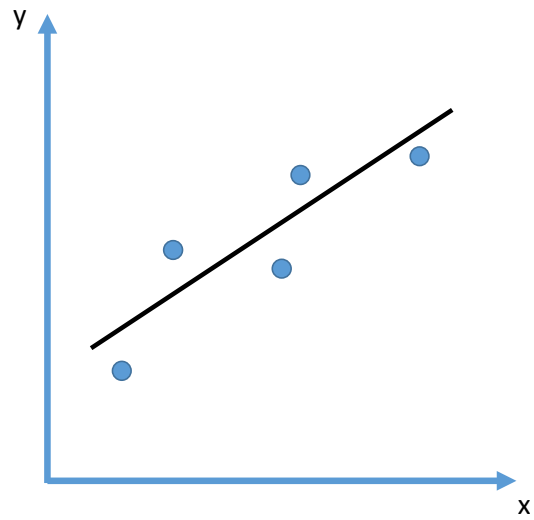
$$f(x) = \sum_i y_i L_i(x) \tag{2}$$

Since

$$L_i(x_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

we have $f(x_j) = y_j$. Hence, the resulting polynomial in Eqn 2 passes through the points (x_1, y_1), (x_2, y_2), ... ,(x_k, y_k).

Least Squares Regression



$$\vec{\hat{y}} = \theta_0 + \theta_1 \vec{x}$$

Least squares regression line equations:

$$\theta_1 = \frac{N(\sum x_i y_i) - \sum x_i \sum y_i}{N \sum x_i^2 - (\sum x_i)^2}$$

$$\theta_0 = \frac{\sum y_i - \theta_1 \sum x_i}{N}$$

Centered Finite-Divided Difference Formulas:

First Derivative:

$$f^{(1)} = \frac{f(x_{i+1}) - f(x_{i-1})}{2h}, \qquad O(h^2)$$

$$f^{(1)} = \frac{-f(x_{i+2}) + 8f(x_{i+1}) - 8f(x_{i-1}) + f(x_{i-2})}{12h}, \qquad O(h^4)$$

Second Derivative:

$$f^{(2)} = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2}, \qquad O(h^2)$$

$$f^{(2)} = \frac{f(x_{i+2}) + 16f(x_{i+1}) - 30f(x_i) + 16f(x_{i-1}) - f(x_{i-2})}{h^2}, \qquad O(h^4)$$

Gradient Descent:

The objective of GD algorithm is to minimize $J_\theta(x)$ with respect to θ. Hence, the final result can be represented as:

$$\hat{\theta} = \arg\min_\theta J_\theta(x) = \arg\min_\theta (\frac{1}{2} \sum_{i=1}^{M} (h_\theta(x^{(i)}) - y^{(i)})^2)$$

At each iteration, the parameters are updated as:

$$\theta_i := \theta_i - \alpha \frac{\partial J(\theta)}{\partial \theta_i}$$

where $:=$ is the assignment operator representing the update and α is the learning rate.

Derivative of the cost with respect to θ_i can be calculated as:

$$\frac{\partial J(\theta)}{\partial \theta_i} = \sum_{j=1}^{M} (h_\theta(x^{(j)}) - y^{(j)}) x_i^{(j)}$$

Hence we can compute θ_i at each update as:

$$\theta_i := \theta_i - \alpha \sum_{j=1}^{M} (h_\theta(x^{(j)}) - y^{(j)}) x_i^{(j)}$$

This algorithym is called Batch GD because it uses the entire data of M samples at each iteration.