Ahmad Badary
Jul - 10 - 2017
m128aPA2
Q.3)

Lagrange:

Assumption:

We are NOT creating the polynomial; we are interpolating on a point or more.

Data:

Input: n+1
Output: N

Analysis:

There are two main computations involved PER Point:

(1) The Coeffecients, $L_{n,k}(x)$:

- Requires n-additions, n-multiplications in the numerator, and the same in the denominator.
  => $O(4*(n+1)) = O(n)$ PER Coeffecient. (1.1)
- There are n-Coeffecients to be computed,
  => $O(n+1)$ (1.2)

(2) The polynomial, $\sum L_{n,k}(x) * y_k$:

- Requires n-additions and n-multiplications
  => $O(2*(n+1)) = O(n+1)$ (2.1)

To compute 1-Output:

=> ((1.1) * (1.2)) + (2.1)
= $O((4*(n+1)) * (n+1)) + O(2*(n+1))$
= $O(4n^2) + O(2n)$
= $O(4n^2 + 2n)$
= $O(n^2 + n)$
= $\mathbf{O(n^2)}$ **(I)**

To compute N-Outputs:

=> (I) * O(N)
= $O(n^2) * O(N)$
= $O(n^2 * N)$
= $\mathbf{O(N * n^2)}$ **(II)**

Notice: To construct the polynomial, the required complexity is
$O(n^2 * \log^2(n))$

Ahmad Badary
Jul - 10 - 2017
m128aPA2

Barycentric:

Assumption:

1. We are NOT creating the polynomial; we are interpolating on a point or more.
2. We are saving the weights and coeffecient (pre-computing) .

Data:

Input: n+1
Output: N

Analysis:

There are three main computations involved:

(1) The Coeffecients, $\ell(x)$:

- Requires n-additions and n-multiplications
=> O(2*(n+1)) = O(n) (1.1)

(2) The weights, w_j :

- Requires n-additions and n-multiplications
=> O(2*(n+2)) = O(n) PER weight (2.1)
- There are n-weights to compute:
=> O(2*(n+2)) = O(n) PER point (2.2)

(3) The polynomial, $\sum w_j/x-x_j * y_k$:

- Requires 2n-additions and n-multiplications
=> O(3*(n+1)) = O(n) (3.1)

To compute 1-Output:

=> (1.1) + ((2.1) * (2.2)) + (3.1)
= O((2*(n+1))) + O((2*(n+2)) * (2*(n+2))) + O(3*(n+1))
= $O(2n) + O(4n^2) + O(3n)$
= $O(4n^2 + 5n)$
= $O(n^2 + n)$
= $O(n^2)$ **(I)**

To compute N-Outputs:

=> (3.1) * O(N)
= O(n) * O(N)
= O(n * N)
= O(N * n) **(II)**

Ahmad Badary
Jul - 10 - 2017
m128aPA2

Q.4)

- There are two sides to this question.
  - Why? Because convergence depends heavily on
    - (1) Point Nodes (2) The Function
- The first function:
  - is absolutely continous on the interval [-1,1] and, thus,
  - has the property that the interpolating polynomial constructed on Chebyshev nodes will converge uniformly to it as n->infinity.
- The second function:
  - is absolutely continous on the interval [-1,1] and, thus,
  - has the property that the interpolating polynomial constructed on Chebyshev nodes will converge to it as n->infinity.
- The polynomial, using the linspace nodes, will converge to Neither of the two functions. [Runges Phenomena]
  - linspace nodes are equidistant
  - Convergence of polynomials using equidistant nodes is not guranteed, even, for infinitely differentiable functions.
  - The error term could potentially go to infinity as n goes to infinity.
  - The first function has an absolute value term, which is very hard to capture.
  - The second function has ill-behaved higher derivatives.
    - In-fact, I can show that the error tends to infinity as n->inf.
- Finally, notice that, the Banach–Steinhaus theorem tells us that for there to exist a table of nodes that allow the interpolating polynomial to converge to the function as n->inf for all the functions in C^1 you need the operator norm of the projection operator on the vector space of polynomials of degree less than or equal to n is to be bounded.
- However, Lebesgue constant tells us that the projection operator norm is bigger than or equal to 2/pi * log(n + 1) + const.
- Thus, it is impossible to find a table of nodes that allows the

lagrange polynomial to converge to all functions.