Homework 5 Solutions

December 12, 2017

1 Problem 1

Express the polynomial with Horner's method:

$$\begin{aligned} p(t) &= 5t^3 - 3t^2 + 7t - 2 \\ &= ((5t - 3)t + 7)t - 2 \end{aligned}$$

1.1 Common Mistakes

Error with $\pm$ signs.

2 Problem 2

How many multiplications are required to evaluate a polynomial $p(t)$ of degree $n-1$ at a given point t

(a) Represented in the monomial basis?

(b) Represented in the Lagrange basis?

(c) Represented in the Newton basis?

2.1 Solution

(a) *Monomial Basis*
Monomial Basis is represented as:

$$p_{n-1}(t) = x_1 + x_2t + x_3t^2 + ... + x_nt^{n-1}$$

However, using Horner's method, the monomial basis can be represented as:

$$p_{n-1}(t) = x_1 + t(x_2 + t(x_3 + t(...(x_{n-1} + x_nt)...)))$$

From the above representation, there are $n-1$ multiplications or $O(n)$.

(b) *Lagrange Basis*
The equation in Lagrange Basis can be represented as:

$$p_{n-1}(t) = y_1 l_1(t) + y_2 l_2(t) + ... + y_{n-1} l_{n-1}(t)$$

Each $l_i(t)$ is given as:

$$\frac{\prod_{j\neq i}(t - t_j)}{\prod_{j\neq i}(t_i - t_j)}$$

Each $l_i(t)$ term will consist of $2(n-1)+1$ multiplications and we are performing these over n points. Thus, the total number of multiplications will be in the order of $O(n^2)$ or approximately $n(2n-1)$ multiplications.

(c) *Newton Basis*
The representation is given as follows:

$$p_n(t) = x_1 + x_2(t-t_1) + x_3(t-t_1)(t-t_2) + ... + x_n(t-t_1)(t-t_2)...(t-t_{n-1})$$

Using Horner's method, we can represent this as:

$$p_n(t) = x_1 + (t-t_1)(x_2 - (t-t_2))(...(x_{n-1}(t-t_{n-1}))...))$$

Here, as in the monomial basis, we have approximately n multiplications or $O(n)$.

3 Problem 3

3.1 Part a

Determine the polynomial interpolant to the data

t	1	2	3	4
$p(t)$	11	29	65	125

using the monomial basis.

For a polynomial with degree 3, we use the following monomial basis:

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 1 & t_1 & t_1^2 & t_1^3 \\ 1 & t_2 & t_2^2 & t_2^3 \\ 1 & t_3 & t_3^2 & t_3^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} p(t_0) \\ p(t_1) \\ p(t_2) \\ p(t_3) \end{bmatrix}$$

Using the data values given this yields:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \\ 1 & 4 & 16 & 64 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 29 \\ 65 \\ 125 \end{bmatrix}$$

Solving this set of equations using any method you prefer yields the equation:

$$p(t) = 5 + 2t + 3t^2 + t^3 \tag{1}$$

3.2 Part b

Determine the Lagrange polynomial interpolant to the same data and show that the resulting polynomial is equivalent to that obtained in part (a).

Since we are looking for a degree 3 polynomial, we want to find the following Lagrange polynomial:

$$L(t) = p(t_0)l_0(t) + p(t_1)l_1(t) + p(t_2)l_0(t) + p(t_3)l_3(t)$$

where we plug in the given values for $p_i(t)$ to find:

$$L(t) = 11\ l_0(t) + 29\ l_1(t) + 65\ l_2(t) + 125\ l_3(t) \tag{2}$$

Note that we can expand each $l_i(t)$ term as follows:

$$l_i(t) = \prod_{j \neq i} \frac{t - t_j}{t_i - t_j}$$

In our case, we need to build the following terms:

$$\begin{aligned}
l_0(t) &= \frac{(t-t_1)(t-t_2)(t-t_3)}{(t_0-t_1)(t_0-t_2)(t_0-t_3)} = \frac{(t-2)(t-3)(t-4)}{(1-2)(1-3)(1-4)} \\
l_1(t) &= \frac{(t-t_0)(t-t_2)(t-t_3)}{(t_1-t_0)(t_1-t_2)(t_1-t_3)} = \frac{(t-1)(t-3)(t-4)}{(2-1)(2-3)(2-4)} \\
l_2(t) &= \frac{(t-t_0)(t-t_1)(t-t_3)}{(t_2-t_0)(t_2-t_1)(t_2-t_3)} = \frac{(t-1)(t-2)(t-4)}{(3-1)(3-2)(3-4)} \\
l_3(t) &= \frac{(t-t_0)(t-t_1)(t-t_2)}{(t_3-t_0)(t_3-t_1)(t_3-t_2)} = \frac{(t-1)(t-2)(t-3)}{(4-1)(4-2)(4-3)}
\end{aligned}$$

If you then simplify these four equations through foiling out the factorizations, then plug each into equation 2 and do some more simplifactions, you will eventually obtain:

$$p(t) = 5 + 2t + 3t^2 + t^3$$

This is the same as equation 1, so we have shown that Lagrange interpolation is equivalent to the monomial method.

3.3 Part c

Since we are solving a polynomial of degree 3, the Newton polynomial of interest in the following three sections is:

$$\mathcal{P}_n(t) = a_0 + a_1(t-t_0) + a_2(t-t_0)(t-t_1) + a_3(t-t_0)(t-t_1)(t-t_2)$$

Plugging in known values we get:

$$\mathcal{P}_n(t) = a_0 + a_1(t-1) + a_2(t-1)(t-2) + a_3(t-1)(t-2)(t-3) \tag{3}$$

3.3.1 Triangular Matrix

Compute the Newton polynomial interpolant to the same data using the triangular matrix method.

The system we are interested in here is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & t_1 - t_0 & 0 & 0 \\ 1 & t_2 - t_0 & (t_2 - t_0)(t_2 - x_1) & 0 \\ 1 & t_3 - t_0 & (t_3 - t_0)(t_3 - x_1) & (t_3 - t_0)(t_3 - t_1)(t_3 - t_2) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} p(t_0) \\ p(t_1) \\ p(t_2) \\ p(t_3) \end{bmatrix}$$

Substituting in the given values we get:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 2-1 & 0 & 0 \\ 1 & 3-1 & (3-1)(3-2) & 0 \\ 1 & 4-1 & (4-1)(4-2) & (4-1)(4-2)(4-3) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 29 \\ 65 \\ 125 \end{bmatrix}$$

Here it makes sense to use forward substitution. Simplifying the matrix and solving you will find:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 18 \\ 9 \\ 1 \end{bmatrix}$$

Now just plug these values into equation 3 and simplify. Doing this you get the following equation:

$$p(t) = 5 + 2t + 3t^2 + t^3$$

Since this is the same as equation 1, we have shown that the triangular matrix method is equivalent to the monomial method.

3.3.2 Incremental Interpolation

Compute the Newton polynomial interpolant to the same data using the incremental interpolation method.

To obtain a polynomial of degree 3 using incremental interpolation we are interested in the following equation:

$$\mathcal{P}_3(x) = c_0\mathcal{N}_0(x) + c_1\mathcal{N}_1(x) + c_2\mathcal{N}_2(x) + c_3\mathcal{N}_3(x) \tag{4}$$

where:

$$\begin{aligned} c_k &= \frac{p(t_k) - \mathcal{P}_{k-1}(t_k)}{\prod_{j=0}^{k-1}(t_k - t_j)} \\ \mathcal{N}_k(t) &= \prod_{j=0}^{k-1}(x - x_j) \\ \mathcal{M}_k(t) &= c_k\mathcal{N}_k(t) \end{aligned}$$

Using these definitions the polynomial can be built incrementally by starting with $\mathcal{P}_0(t)$:

$$\begin{aligned}
\mathcal{P}_0(t) &= \mathcal{M}_0(t) \\
&= p(t_0) \\
&= 11
\end{aligned}$$

Then $\mathcal{P}_1(t)$ follows as:

$$\begin{aligned}
\mathcal{P}_1(t) &= \mathcal{P}_0(t) + \mathcal{M}_1(t) \\
&= \mathcal{P}_0(t) + c_1(t - t_0) \\
&= \mathcal{P}_0(t) + \frac{p(t_1) - \mathcal{P}_0(t)}{t_1 - t_0}(t - t_0) \\
&= 11 + \frac{29 - 11}{2 - 1}(t - 1) \\
&= 11 + 18(t - 1) \\
&= -7 + 18t
\end{aligned}$$

We now note that we can write $\mathcal{P}_2(t)$ and $\mathcal{P}_3(t)$ as:

$$\begin{aligned}
\mathcal{P}_2(t) &= \mathcal{P}_1 + \mathcal{M}_2(t) \\
\mathcal{P}_3(t) &= \mathcal{P}_2 + \mathcal{M}_3(t)
\end{aligned}$$

If you recursively solve these two equations using the results from $\mathcal{P}_1(t)$ that we just found, you will find that $\mathcal{P}_3(t)$ is:

$$\mathcal{P}_3(t) = 5 + 2t + 3t^2 + t^3$$

This is equation 1, so we have shown that this method is equivalent the monomial method.

3.3.3 Divided Differences

Compute the Newton polynomial interpolant to the same data using the divided difference method.

First we start by noting what the notation for divided difference is:

$$f[t_i, t_{i+1}, \ldots, t_{i+j-1}, t_{i+j}] = \frac{f[t_{i+1}, \ldots, t_{i+j}] + f[t_i, \ldots, t_{i+j-1}]}{t_{i+j} - t_i}$$

Here, the third degree polynomial will be computed from:

$$\begin{aligned}
\mathcal{P}_3(t) = f[t_0] & \\
&+ f[t_0, t_1](t - t_0) \\
&+ f[t_0, t_1, t_2](t - t_0)(t - t_1) \\
&+ f[t_0, t_1, t_2, t_3](t - t_0)(t - t_1)(t - t_2)
\end{aligned}$$

Using the definition for divided difference, we find all of the values in the previous equation as:

$$
\begin{aligned}
f[t_0] &= 11 \\
f[t_0, t_1] &= \frac{29 - 11}{2 - 1} = 18 \\
f[t_1, t_2] &= \frac{65 - 29}{3 - 2} = 36 \\
f[t_0, t_1, t_2] &= \frac{36 - 18}{3 - 1} = 9 \\
f[t_0, t_3] &= \frac{125 - 11}{4 - 1} = 38 \\
f[t_1, t_3] &= \frac{125 - 29}{4 - 2} = 48 \\
f[t_2, t_3] &= \frac{125 - 65}{4 - 3} = 60 \\
f[t_1, t_2, t_3] &= \frac{60 - 48}{3 - 2} = 12 \\
f[t_0, t_1, t_2, t_3] &= \frac{12 - 9}{4 - 1} = 1
\end{aligned}
$$

If you then plug in the necessary values from the calculations above into the equation above, you will find the equation:

$$\mathcal{P}_3(t) = 5 + 2t + 3t^2 + t^3$$

This is the same as equation 1, so we can say that divided differences method for interpolation is equivalent to the monomial method.

4 Problem 4

(a) For a given set of data points $t_1, \dots, t_n$, define the function $\pi(t)$ by

$$\pi(t) = (t - t_1)(t - t_2) \dots (t - t_n)$$

Show that

$$\pi'(t_j) = (t_j - t_1) \dots (t_j - t_{j-1})(t_j - t_{j+1}) \dots (t_j - t_n)$$

4.1 Solution

We are given:

$$\frac{d}{dt} \prod_{i=1}^{n} f_i = \sum_{i=1}^{n} f_i' \prod_{j \neq i} f_j$$

We substitute $(t - t_i)$ for f_i:

$$\begin{aligned}
\frac{d}{dt}\prod_{i=1}^{n}(t-t_i) &= \sum_{i=1}^{n}(t-t_i)'\prod_{j\neq i}(t-t_i) \\
&= \sum_{i=1}^{n}1\prod_{j\neq i}(t-t_i)
\end{aligned}$$

If we now plug in t_j for t, we get:

$$\pi'(t_j) = \prod_{j\neq i}(t_j - t_i)$$

(b) Use the result of part (a) to show that the jth Lagrange basis function can be expressed as

$$l_j(t) = \frac{\pi(t)}{(t-t_j)\pi'(t_j)}$$

4.2 Solution

We know that the Lagrange basis function is defined as:

$$\frac{\prod_{j\neq i}(t-t_j)}{\prod_{j\neq i}(t_i-t_j)}$$

From part (a), we proved that:

$$\pi'(t_j) = (t_j - t_1)\dots(t_j - t_{j-1})(t_j - t_{j+1})\dots(t_j - t_n)$$

We can substitute $\pi'(t_j)$ to the denominator to get:

$$\frac{(t-t_1)...(t-t_{j-1})(t-t_{j+1})...(t-t_n)}{\pi'(t_j)}$$

Now, we multiply the numerator and denominator by $(t - t_j)$ to get:

$$\frac{(t-t_1)...(t-t_{j-1})(t-t_{j+1})...(t-t_n)(t-t_j)}{\pi'(t_j)(t-t_j)}$$

Since the numerator is equal to $\pi(t)$, we get:

$$l_j(t) = \frac{\pi(t)}{(t-t_j)\pi'(t_j)}$$

5 Problem 5

The recursion relation is:

$$
\begin{aligned}
b_{n-1} &= a_n \\
b_{i-1} &= a_i + t_0 b_i \text{ for } i = 1 \ldots n-1 \\
p(t_0) &= a_0 + t_0 b_0
\end{aligned}
$$

The equations for $p(t)$ and $q(t)$ with expressions using Horner's method are:

$$
\begin{aligned}
p(t) &= a_0 + t \cdot (a_1 + t \cdot (a_2 + \cdots + t \cdot (a_n))) \\
q(t) &= b_0 + t \cdot (b_1 + t \cdot (b_2 + \cdots + t \cdot (b_{n-1})))
\end{aligned}
$$

Therefore, $p(t)$ can be computed recursively from the innermost parentheses a_n. So p is initialized as $a[n]$, where the array a stores from a_0 to a_n.
In every iteration, p is being updated by adding a_{i-1} to previous $p \cdot t$, where $i = n \ldots 1$. In the code, we use a_{n-1-i}, where $i = 0 \ldots n-1$. Therefore in each iteration, $p = a[n-1-i] + t * p$.

Since $b_{i-1} = a_i + t_0 b_i$ for $i = 1 \ldots n-1$, current $q(= p')$ can be computed by the previous computed p and q. In the code, we compute q before computing p with $q = p + t * q$, because p is initalized with a_n, and q is still 0. In the first iteration, q becomes a_n. To verify, $q(t)$ is being computed recursively from the innermost parentheses b_{n-1}, and $b_{n-1} = a_n$.

5.1 Sample by Student

Python 2.7.10

```
import numpy as np
def horners(n, a, t): #n, degree; a, array of a_0 to a_n
  p = a[n]
  q = 0
  for i in range(0, n):
      q = p + t * q
      p = a[n-1-i] + t * p
  return p, q

def main():
  print "Example, p(t) = 1 + 2t + 3t^2, where t0 = 1"
  n = 2
  a = np.array([1,2,3])
  t = 1
  p, p_prime = horners(n, a, t)
  print "p(t0)", p
  print "p'(t0)", p_prime

if __name__ == "__main__":
      main()
```