Math 128A, Summer 2019
Lecture 11, Wednesday 7/10/2019

Topics today:
- Newton Interpolation
- Divided Differences
- Hermite Interpolation

Midterm topics (first 3 weeks): IEEE arithmetic, Bisection, Rate of Convergence, Iteration (fixed point, newton), Interpolation (lagrange + **error**, newton, hermite)

Recall we have Lagrange Interpolation, Newton Interpolation, Hermite Interpolation.

1 Newton Interpolation

The idea here is to use the Newton basis (we just call it this because we use this basis for Newton's):

$$1, t-t_0, (t-t_0)(t-t_1), \dots, (t-t_0)\cdots(t-t_n)$$

And recall, if our diagonal entries of this triangular matrix are nonzero, then we have a unique solution. And our polynomial is:

$$p(t) = a_0 + a_1(t-t_0) + \cdots + a_n(t-t_0)\dots(t-t_n)$$

This basis is 'simpler' than the Lagrange basis (where each term is of degree n and vanishes at each interpolation point). So in a Newton basis, we build an interpolating polynomial **recursively** by:

$$p_n(t) := \overbrace{p_{n-1}(t)}^{\text{deg } n-1} + a_n\left[(t-t_0)(t-t_1)(t0-t_{n-1})\right]$$

where $p_n(t)$ is the polynomial evaluating over $t_0, \dots, t_n$, and $p_{n-1}(t)$ interpolates over $t_0, \dots, t_{n-1}$. The next term $a_n[\cdots]$ has to raise the degree by 1 but **'not mess with'** the already interpolated points from the left term. Thus the bracketed expression ensures that, and a_n just has to be chosen to interpolate the new point t_n. Hence:

$$\begin{aligned} p_n(t_n) &:= p_{n-1}(t_n) + a_n\left[(t_n-t_0)(t_n-t_1)\cdots(t_n-t_{n-1})\right] \\ &= f_n, \end{aligned}$$

and we simply solve this equation to find a_n:

$$a_n = \frac{f_n - p_{n-1}(t_n)}{(t_n-t_0)(t_n-t_1)\cdots(t_n-t_{n-1})}$$

Notice that $p_{n-1}(t_n)$ is also an approximation to f_n, so we guess that the numerator will be tiny, and the denominator should be tiny (if we designed the distribution well).
However, there's a better formula:

Definition: Divided Difference -

$$\begin{aligned}
a_1 &:= \frac{f[t_1]-f[t_0]}{t_1-t_0} \\
&= \frac{f_1-f_0}{t_1-t_0} = f[t_0,t_1] = f[t_1,t_0] \\
a_2 &:= \frac{f[t_1,t_2]-f[t_0,t_1]}{t_2-t_0} \\
&= f[t_0,t_1,t_2] = f[t_2,t_1,t_0] = f[t_2,t_0,t_1] = \cdots \\
&\ \ \vdots \\
a_n &:= \frac{f_n - p_{n-1}(t_n)}{(t_n-t_0)(t_n-t_1)\cdots(t_n-t_{n-1})} \\
&= f[t_0,t_1,\ldots,t_n] = \text{or any permutation of } t_i
\end{aligned}$$

Example: Divided Difference (to generate the above):

$$\begin{aligned}
n &= 0; \\
p_0(t) &= f(t_0) = f[t_0] \\
n &= 1; \\
p_1(t) &= p_0(t) + a_1(t-t_0) \\
p_1(t_1) &= f_1 \\
&\ \ \vdots
\end{aligned}$$

Example: Recall the function $f(t) := e^t$. Let us define: $t_0 := 0, t_1 := 1, t_2 := 2$. Then

$$\begin{aligned}
f[0] &= f[t_0] = 1, \\
f[1] &= f[t_1] = e, \\
f[2] &= f[t_2] = e^2
\end{aligned}$$

Our divided difference table looks like:

$$\begin{array}{llll}
0 & f_0 = 1 & \frac{e-1}{1-0} = e-1 & \frac{e^2-e-(e-1)}{2-0} = \frac{e^2-2e+1}{2} \\
1 & f_1 = e & \frac{e^2-e}{2-1} = e^2-e & \\
2 & f_2 = e^2 & &
\end{array}$$

So this readily gives us the interpolating polynomial in the Newton basis:

$$p(t) = \underbrace{1}_{p(0)=1} + \underbrace{(e-1)}_{p(1)=e}(t-0) + \frac{e^2-2e+1}{2}(t-0)(t-1)$$

and we can 'check' values easily in the Newton basis. Compare this to Lagrange from yesterday:

$$p(t) = 1\frac{(t-1)(t-2)}{(0-1)(0-2)} + e\frac{(t-0)(t-2)}{(1-0)(1-2)} + e^2\frac{(t-0)(t-1)}{(2-0)(2-1)}$$

Later, we'll talk about Hermite interpolation, which is a generalization of a Taylor expansion which expands through derivative orders, and newton/lagrange is interpolation at a particular degree. We think of these as limiting or edge cases of Hermite interpolation which involves interpolation at a particular derivative order.

2 Error in Newton Interpolation

Recall:

$$f(t) - p_n(t) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega(t) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\left[(t-t_0)(t-t_1)\cdots(t-t_n)\right]$$

Consider:

$$p_{n+1}(s) = p_n(s) = f[t_0, \dots, t_n, \underbrace{}_{t}]\omega(t),$$

so that $p_{n+1}(t) = f(t)$ for a fixed t. We have the interpolation error via the top expression. Then we fix t and interpolate at another point, that point being t. Then when we evaluate the polynomial, we have:

$$p_{n+1}(t) = f(t) = p_n(t) + f[t_0, \dots, t_n, t]\omega(t).$$

Thus we conclude these must be the same error (formula), and we have the following theorem:

Theorem 2.1.

$$f[t_0, t_1, \dots, t_n, t] = \frac{f^{(n+1)}(\xi)}{(n+1)!},$$

where $\xi \in [\min\{t_j, t\}, \max\{t_j, t\}]$.

This equivalently states:

$$f[t_0, t_1, \dots, t_n] = \frac{f^{(n)}(\xi)}{(n)!},$$

which is not surprising because this gives: $f[t_0] = f(t_0)$ and

$$f[t_0, t_1]\frac{f(t_1) - f(t_0)}{t_1 - t_0} = \frac{f'(\xi)}{1!},$$

which is equivalent to the MVT for $n = 1$. For higher orders, we get something more interesting.
Suppose $t_1 \to t_0$. Then

$$f[t_0, t_1] = \frac{f'(\xi)}{1!} \to f'(t_0)$$

Equivalently, for all $t_j \to x$,

$$f[t_0, t_1, t_2] = \frac{f''(\xi)}{2!} \to \frac{f''(x)}{2!}$$

This allows us to do **Hermite interpolation**, by interpolating using the derivative **at the same point** at different derivatives.
If we set all $t_j \to t_0$,

$$\begin{aligned} p_n(t) = f[t_0] + f[t_0, t_1](t - t_0) \\ + \cdots \\ + f[t_0, \ldots, t_n](t - t_0) \cdots (t - t_{n-1}) \\ \to f(t_0) + f'(t_0)(t - t_0) + \cdots + \frac{f^{(n)}(t_0)}{n!}(t - t_0)^n, \end{aligned}$$

which is identical to Taylor's expansion.

Example: Hermite Interpolation We set:

$$p(t_0) = f_0, \quad p'(t_0) = f_0', \quad p(t_1) = f_1, \quad p'(t_1) = f_1'.$$

We create a divided-difference table: We take the top row of our divided difference table, which was:

$$0 \qquad 1 \qquad 1 \qquad e-2 \qquad 3-e \qquad \frac{e^2 - e + 2}{2} \qquad \frac{6e - 3e^2 - 1}{4},$$

and we set:

$$\begin{aligned} p(t) = 1 + 1\,(t - 0) + (e - 2)\,(t - 0)^2 \\ + (3 - e)(t - 0)^2(t - 1) \\ + \frac{e^2 - e + 2}{2}(t - 0)^2(t - 1)^2 \\ + \frac{6e - 3e^2 - 1}{4}(t - 0)^2(t - 1)^2(t - 2), \end{aligned}$$

which gives us our desired quintic Hermite polynomial, interpolating at 3 distinct points.

2.1 Why or When Would We Use Hermite Interpolation?

Taking equispaced points, if we let $n = 2k$, suppose we instead take two points t_0, t_1 centered in the larger interval, but stacked the derivatives at the center (vertically) rather than spread out on the interval. For the same amount of 'work $n = 2k$,' we compare the errors:

$$\begin{aligned} \omega(t) = (t - t_0) \cdots (t - t_n) \\ \omega(t) = (t - t_0)^k (t - t_1)k, \end{aligned}$$

and the second polynomial (hermite with derivatives) has a lot less error, for the same amount of work. Also notice that hermite interpolation with fourier can be good, because evaluating derivatives of the fourier (trig) functions is free, just like evaluating the fourier function itself.

Error for Hermite Interpolation : Consider:

$$\begin{matrix} f(t_0) & \qquad\qquad & f'(t_0) \\ \vdots & & \vdots \\ f(t_n) & & f'(t_n) \end{matrix}$$

So the error is going to be the same as Lagrange for this polynomial degree, deg $p = 2n+1$. We inherit the Lagrange error and let points merge:

$$f(t) - p(t) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!}(t-t_0)^2 \cdots (t-t_n)^2$$

We call this strategy 'confluence', as to let the points flow together.
Notice that we have squared factors on the right because we need to make the function *and* the derivative vanish at that point. This is equivalent to having, for example, $t_0 + \epsilon$ and letting $\epsilon \to 0$.

3 Lagrange Basis for Hermite Interpolation

Let

$$p(t) := \sum_{j=0}^{n} f_j \underbrace{A_j(t)}_{\text{deg } 2n+1} + \sum_{j=0}^{n} f_j' B_j(t)$$

where

$$\begin{aligned} A_j(t_k) &= \delta_{j,k} & B_j(t_k) &= 0 \\ A_j'(t_k) &= 0 & B_j'(t_k) &= \delta_{j,k}. \end{aligned}$$

So first, we try to be clever (efficient) to satisfy the above conditions:

$$\begin{aligned} \underbrace{A_j(t)}_{\text{deg } 2n+1} &= \underbrace{(a + b(t-t_j))}_{\text{to raise deg}} \cdot \underbrace{L_j(t)^2}_{\text{deg } 2n} \\ B_j(t) &= (t-t_j)L_j(t)^2 \\ B_j'(t) &= L_j(t)^2 + (t-t_j)2L_j(t)L_j'(t) \end{aligned}$$

And for A_j and A_j',

$$\begin{aligned} A_j(t_k) &= (1 + b(t_k - t_j))\, L_j(t_k)^2 \\ A_j'(t) &= bL_j(t)^2 + 2\,(1 + b(t-t_j))\, L_j'(t)L_j(t) \\ A_j'(t_j) &= b + 2L_j'(t_j) \\ b &= -2L_j'(t_j), \end{aligned}$$

and to summarize,

$$\begin{aligned} A_j(t) &= \left[1 - 2L_j'(t_j)(t-t_j)\right] L_j(t)^2 \\ B_j(t) &= (t-t_j)L_j(t)^2. \end{aligned}$$