Math 128A: Polynomial Interpolation

Nate Armstrong

Lecture 7/10/19

Midterm Info

2.3 Review

We want to solve the problem below.

$$p(t_j) = f_j \quad 0 \leq j \leq n$$

where p is a degree n polynomial. We have multiple formulas for the above. We could use Lagrange interpolation, which is

$$p(t) = \sum_{j=0}^{n} f_j L_j(t)$$

where $L_j(t)$ is the Lagrange basis function for t_j. We can also use Newton interpolation, which is where

$$p(t) = \sum_{j=0}^{n} f[t_0, t_1, \ldots, t_j](t - t_0) \cdots (t - t_{j-1})$$

Generally, this is solved by using a table and building the differences recursively. Finally, if we want to solve this problem and also approximate the derivatives of f_j at these points, we have $(n+1)(m+1)$ conditions on our polynomial, where m is the number of derivatives we want. Note that if $m = 0$, this is the same as regular polynomial interpolation. The degree of p is then $(n+1)(m+1) - 1$.

This is called Hermite interpolation.

Remark
Most people call general interpolation 'Lagrange Interpolation'. This is a special case of Hermite Interpolation. Hermite interpolation can handle different amounts of derivatives in interpolation, but you cannot handle derivatives at a point without the function itself, or any gaps in the sequence of derivatives. In those cases, it is called Birkhoff Interpolation. It is not known when this is possible.

It is very fast to change data points when the interpolated polynomial is in Lagrange form. However, when the data changes in Newton form, you have to do $O(n^2)$ work to recreate the polynomial. However, if the data is fixed and t is changing, Newton's form is much more efficient (around $3n$ operations compared to $O(n^2)$). There are other ways of evaluating interpolating polynomials which can let you do much less computational work, if you're willing to do more thinking about it.

Hermite interpolation can use both Lagrange and Newton form. However, doing it in Newton form is much easier, as finding the Lagrange basis functions can be a huge pain.

Example
Suppose I have f_0, f_0', f_0'', f_1. Then, I interpolate as

$$\begin{array}{c|c|c|c|c} 0 & f_0 & f_0' & \frac{f_0''}{2!} & f[t_0,t_0,t_0,t_1] \\ 0 & f_0 & f_0' & \frac{f[t_0,t_1]-f_0'}{1-0} = f[t_0,t_0,t_1] & \\ 0 & f_0 & \frac{f_1-f_0}{1-0} & & \\ 1 & f_1 & & & \end{array}$$

Thus, we get a final polynomial of

$$p(t) = f[t_0] + f[t_0,t_0](t-t_0) + f[t_0,t_0,t_0](t-t_0)^2 + f[t_0,t_0,t_0,t_1](t-t_0)^3$$

Note from Strain: We don't know how to solve using the table if we don't put matching values next to each other while solving the table.

Last time, we also computed the Lagrange basis of the Hermite interpolation. We can find the Lagrange basis by Newton, using the equations that

$$\begin{aligned} A_j(t_k) &= \delta_{jk} & B_j(t_k) &= 0 \\ A_j'(t_k) &= 0 & B_j'(t_k) &= \delta_{jk} \end{aligned}$$

Then, I state that $p(t) = \sum_{j=0}^{n} f_j A_j(t) + f_j' B_j(t)$ Then, I can use Newton to find all the A_js and B_js for this particular case.

3 Numerical Differentiation

Although you can always compute the derivative of a function if you have a formula, most of the time you just have data points. You want to find the derivative at or near the data points. The usual way to do it is by interpolating with a polynomial, and then take the derivative at the point you are interested in.

1. Interpolate by $p(t)$ at $t_0, \ldots, t_n$.
2. Approximate $f'(a)$ by $p'(a)$, as you can compute $p'(a)$.

Example
Suppose I have $f(t_0)$, and $f(t_1)$. If I approximate with a constant polynomial, I will certainly get $p'(t) = 0$.

If I approximate with a linear polynomial, I will get $p'(t) = \frac{f(t_1)-f(t_0)}{t_1-t_0} = f[t_0, t_1]$. As this is a divided difference, we know it is equal to the first derivative of f' at some point. Unfortunately, this point is most likely not the point we want.

Now, we are interested in learning the error in such an approximation. We know that

$$f(t) - p(t) = \frac{f^{(n+1)}(\zeta)}{(n+1)!}\omega(t)$$

which we can differentiate to

$$f'(t) - p'(t) = \frac{f^{(n+1)}(\zeta)}{(n+1)!}\omega'(t) + \frac{1}{(n+1)!}\omega(t)\frac{d}{dt}f^{(n+1)}(\zeta)$$

The right side is ugly, however we know that $\omega(t_j) = 0$ for all j, and so if we only worry about the error at the interpolation points, we can ignore the right term. Thus, at $t = t_j$, the error is just

$$E_j = \frac{f^{(n+1)}(\zeta)}{(n+1)}\omega'(t_j)$$

and so we need to find $\omega'(t_j)$. Because I have $n+1$ terms of the form $(t-t_j)$, I have $n+1$ terms which each have 1 term missing (the one that was differentiated) via the chain rule. Formally,

$$\omega'(t_j) = \sum_{k=0}^{n}\prod_{l\neq k}(t_j - t_l)$$

However, as this is evaluated at some t_j, the only nonzero term is the one where $(t-t_j)$ was differentiated. Thus, at $t = t_j$, we get

$$\omega'(t) = \prod_{k\neq j}(t_j - t_k).$$

which is a relatively well-behaved product. In fact, this is a factor of $L_j(t)$. We can actually write that

$$L_j(t) = \frac{\prod_{k\neq j}(t-t_k)}{\prod_{k\neq j}(t_j-t_k)} = \frac{\omega(t)/(t-t_j)}{\omega'(t)} = \frac{\omega(t)}{\omega'(t)(t-t_j)}$$

This formula is weird, but very useful (according to prof).

3.1 Coefficients

In order to solve for the derivative of a polynomial which we have interpolated, we want the derivative in terms of the values of the function at a certain point.

We want a formula of the form

$$f^{(m)}(a) = \frac{\delta_{n0}^m f(t_0) + \cdots + \delta_{nm}^m f(t_n)}{h^m}$$

We construct general formulas using the Taylor expansion, using

$$f(a+h) = f(a) + hf'(a) + \frac{1}{2}h^2 f''(a) + \ldots$$

and that

$$f(a-h) = f(a) - hf'(a) + \frac{1}{2}h^2 f''(a) + \ldots$$

to get

$$f(a+h) - 2f(a) + f(a-h) = h^2 f''(a) + \frac{1}{6}h^3 f'''(\zeta_+) - \frac{1}{6}h^3 f'''(\zeta_-)$$

The error here is of $O(h^2 f^{(4)}$, as we can take one more term in the Taylor expansion and let the third terms cancel. In the case of the second derivative, we have $\delta_{2,0}^2 = 1$, $\delta_{2,1}^2 = -2$, and $\delta_{2,2}^2 = 1$. In fact, we need that all of the δs must sum to 0 when added. This is the case because if the interpolated polynomial is $f(x) = c$, it must correctly return 0.

If we interpolate the polynomial in Lagrange form, then we write

$$p^{(m)}(a) = \sum_{j=0}^{m} f_j L_j^{(m)}(a).$$

Thus, we want to compute $\delta_{nj}^m(a) = L_j^{(m)}(a)$.

The notation is complex, but $\delta_{nj}^m(a)$ means the mth derivative of the Lagrange basis function for n points, applied for the jth point, evaluated at a.

3.1.1 'Newton'ing the Lagrange

When Newton is a verb, it means adding the points one at a time. We let

$$L_{nj}(t) = \prod_{k \neq j} \frac{t - t_k}{t_j - t_k} = \frac{t - t_n}{t_j - t_n} L_{n-1,j}(t)$$

which is true for $j < n$. Now, to calculate the derivatives of this, I get

$$L'_{nj} = \frac{1}{t_j - t_n} L_{n-1,j}(t) + \frac{t - t_n}{t_j - t_n} L'_{n-1,j}(t)$$

Then, I look for L''_{nj}. This is

$$L''_{nj}(t) = \frac{2}{t_j - t_n} L'_{n-1,j}(t) + \frac{t - t_n}{t_j - t_n} L''_{n-1,j}(t).$$

Now, we apply this to the third derivative.

$$L'''_{nj}(t) = \frac{3}{t_j - t_n} L''_{n-1,j}(t) + \frac{t - t_n}{t_j - t_n} L'''_{n-1,j}(t).$$

This gives me a recurrence relation. I see that

$$\delta^m_{nj} = \frac{m}{t_j - t_n} \delta^{m-1}_{n-1,j} + \frac{a - t_n}{t_j - t_n} \delta^m_{n-1,j} \tag{1}$$

as a recurrence relation for $j < n$. The way to organize this would be to have a target n, then calculate this for all j and $m = 0$. This is pretty easy, as for $m = 0$ the above formula is trivial. Then, we calculate the values here for $m = 1$, using the formula

$$\delta^1_{nj} = \frac{1}{t_j - t_n} \delta^0_{n-1,j} + \frac{a - t_n}{t_j - t_n} \delta^1_{n-1,j}$$

It's hard to think about this problem.

We spent the rest of lecture trying to think of an ordering in which to program this. At the end, Strain mentioned that he took this time in order to show that the problem is not trivial. It seemed that we could first compute for $m = 0$, then use those values to compute the above. We know that $\delta^m_{00} = 0$ for any $m > 1$. We also solved for some small index values of δ, but I did not type those up.