Lagrange Interpolating Polynomials

James Keesling

1 Determining the Coefficients of the Lagrange Interpolating Polynomial by Linear Equations

It is frequently the case that we will have certain data points, $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$, and will want to fit a curve through these points. In this chapter we will fit a polynomial of minimal degree through the points. We assume that the points $\{x_0, x_1, \ldots, x_n\}$ are all distinct. In that case we can fit a polynomial of degree n (or possibly less) through the points. If we write the polynomial in the following form, then we can use the points to determine the coefficients.

$$L(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \cdots + a_n \cdot x^n$$

$$\begin{aligned}
y_0 &= a_0 + a_1 \cdot x_0 + a_2 \cdot x_0^2 + \cdots a_n \cdot x_0^n \\
y_1 &= a_0 + a_1 \cdot x_1 + a_2 \cdot x_1^2 + \cdots a_n \cdot x_1^n \\
&\vdots \\
y_n &= a_0 + a_1 \cdot x_n + a_2 \cdot x_n^2 + \cdots a_n \cdot x_n^n
\end{aligned}$$

We can solve these equations using matrices. The vector

$$A = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}$$

is the set of coefficients to be determined. We let $M = V[x_0, x_1, \ldots, x_n]$ be the Vandermonde Matrix and B be the vector of y values for the interpolation points, then the coefficients of the polynomial will be given by the following matrix equation.

$$M = \begin{bmatrix} 1 & x_0 & x_0^2 & x_0^3 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & x_1^3 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & x_2^3 & \cdots & x_2^n \\ \vdots & \vdots & & & & \vdots \\ 1 & x_n & x_n^2 & x_n^3 & \cdots & x_n^n \end{bmatrix} Y = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$A = M^{-1} \cdot Y$$

2 TI-89 Programs Calculating The Vandermonde Matrix and the Lagrange Polynomial

Below is a TI-89 program that determines the Vandermonde matrix for a set of points $\{x_0, x_1, \ldots, x_n\}$. The input variable a a vector $[x_0, x_1, \ldots, x_n]$ and the input variable n indicates that there are $n+1$ components in the vector.

```
:vanderm(a)
:Prgm
:dim(a)[2]-1 → n
:newMat(n+1, n+1) → vander
:For i,1,n+1
:For j,1,n+1
:If a[1,i]=0 and j=1
:Then
:1 → vander[i,j]
:Else
:a[1,i]∧(j-1) → vander[i,j]
:EndIf
:EndFor
:EndFor
:EndPrgm
```

The program below computes the coefficients of the Lagrange polynomial for the interpolation points $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_)\}$. The input variable a is the vector $[x_0, x_1, \ldots, x_n]$. The input variable b is the vector $[y_0, y_1, \ldots, y_n]$. The variable n is the degree of the Lagrange polynomial. The variable $coef$ is a column vector with the coefficients of the polynomial. The output variable P is the Lagrange polynomial with variable x.

```
:lagrange(a,b)
```

```
:DelVar x
:dim(a)[2]-1 → n
:vanderm(a)
:newMat(n+1,1) → coef
:vander∧(-1)*b^T → coef
:0 → p
:For i,0,n
:p+coef[i+1,1]*x∧i → p
:Disp p
:EndFor
:EndPrgm
```

3 An Alternative Approach Using Special Polynomials

Let us start with the data points, $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$, as above. For $i = 0, 1, 2, \ldots, n$ consider the polynomials $L_i(x)$ given by the following formula.

$$L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{(x - x_j)}{(x_j - x_i)}$$

The polynomial $L_i(x)$ has the property that

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

The polynomial $L_i(x)$ is the Lagrange polynomial for the interpolation points

$$\{x_0, 0), \ldots, (x_{i-1}, 0), (x_i, 1), (x_{i+1}, 0), \ldots, (x_n, 0)\}.$$

The Lagrange polynomial $L(x)$ for the original interpolation points is now given by the following formula.

$$L(x) = \sum_{i=0}^{n} y_i \cdot L_i(x)$$

It is clear that this polynomial has degree $\leq n$ and has the property that $L(x_i) = y_i$ as required.

Note that the Lagrange polynomial, $L(x)$, is unique. If there were two such polynomials, $L(x)$ and $P(x)$, then $L(x) - P(x)$ would be a polynomial of degree $\leq n$ with $n + 1$ zeros. Thus it would have to be identically zero. Thus, we must have $L(x) \equiv P(x)$.

4 Newton Polynomials

Another approach to determining the Lagrange polynomial is attributed to Newton. It is similar to the approach in the previous section in that it uses linear factors that are zero at the interpolation points. We still assume that we are fitting a polynomial of minimal degree through the points $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$. The Newton form of the polynomial is given by the following formula.

$$L(x) = \sum_{i=0}^{n} \left(a_i \cdot \prod_{i=0}^{i} (x - x_i) \right)$$

The problem is to determine the coefficients $\{a_i | i = 0, 1, \ldots, n\}$. It is obvious that $a_0 = y_0$. One can solve for the coefficients recursively. Or, one can use a method called *divided differences*. The algorithm produces an array of the following form.

$$D = \begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0n} \\ \vdots & \vdots & & \vdots \\ a_{n0} & a_{n1} & \cdots & a_{nn} \end{bmatrix}$$

Where the entries are defined in the following way. The first column are the values $\{y_0, \ldots, y_n\}$. That is to say:

$$\begin{bmatrix} a_{00} \\ a_{01} \\ \vdots \\ a_{0n} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Each successive column will have one less non-zero entry with

$$a_{i,j+1} = \frac{a_{i+1,j} - a_{i,j}}{x_{i+j} - x_i}$$

The final form of the Newton polynomial is given by setting the coefficients equal to the first row of the divided difference matrix, $a_i = a_{0i}$.

Below is a program that computes the divided difference matrix and the Newton polynomial from the entries in the matrix.

```
newtpoly(a,b)
:prgm
:dim(a)[2]-1 → n
:newMat(n+1,n+1) → coef
:For i,1,n+1
:b[1,i] → coef[i,1]
```

```
:EndFor
:For j,1,n
:For i,1,n-j+1
:(coef[i+1,j]-coef[i,j])/(a[1,i+j)-a[1,i]) → coef[i,j+1]
:EndFor
:EndFor
:0 → p
:For j,1,n+1
:coef[i,j] → temp
:For i,1,j-1
:coef[1,j] → temp
:For i,1,j-1
:temp · (x-a[1,i]) → temp
:EndFor
:p+temp → p
:EndFor
:Disp coef
:Disp p
:EndPrgm
```

5 Taylor polynomials

Occasionally one may want other conditions on a polynomial other than fitting values at different points. The most important is determining the polynomial that has a certain set of derivatives at a point. In this case the point is taken to be x_0 and suppose that the first n derivatives are given, $\{a_0 = p(x_0), a_1 = p'(x_0), \ldots, p^{(n)}(x_0)\}$. Then the polynomial is given by the following formula.

$$p(x) = \sum_{i=0}^{n} \left(\frac{a_i}{i!} \cdot (x - x_0)^i \right)$$

This is called the *Taylor polynomial of degree n* centered at the point x_0. More will be said about these polynomials as they are needed.

Other conditions on polynomials may include requiring that they be orthogonal to one another as is the case with the *Legendre polynomials*. These will be dealt with in the discussion of Gaussian quadrature. There is a built-in function in the TI-89 that will produce the Taylor polynomial with a given set of derivatives. The syntax of the command is below.

```
tayor(f(x),x,n,a)
```

The command produces the Taylor polynomial.

$$p(x) = \sum_{i=0}^{n} \left(\frac{1}{i!} \frac{d^i f(x)}{dx^n} \Big|_{x=a} \cdot (x-a)^i \right)$$