Polynomial Interpolation

Math 131: Numerical Analysis

J.C. Meza

2/22/24

Interpolation and Extrapolation

- Interpolation is usually not an end product in numerical analysis, but it is used in many other techniques that we will study.
- Having a good understanding of the concepts in approximating functions or interpolating some given data will prove useful later.
- Let's consider approximation first. There are two basic problems:
  1. data fitting, and
  2. approximating other functions.

Data fitting

For ease, we will only consider the case of $\mathbb{R}^1$ here, although (as before) many of the ideas translate into higher dimensions. Suppose we are given a set of data points

$$\{(x_i, y_i)\}_{i=0}^{n}.$$

The points x_i are sometimes called the ***node points*** or simply just ***nodes***.

Goal

Find a function $v(x)$ that "fits" the data in some yet to be determined way.

Interpolation

If the data is believed to be accurate, we could also insist that $v(x)$ match the data exactly, i.e. that

$$v(x_i) = y_i \quad i = 0, 1, \ldots n.$$

If this is the case, then we say that $v(x)$ ***interpolates*** the data.

This still leaves open the question of what we mean by *fit*, and also what might constitute a good function, i.e. what properties do we want in a function that fits the data.

Approximating a function

- The second area we will study is that of approximating another function.
- This situation might arise if we had a complicated or computationally expensive function.

Goal

Find a simpler or cheaper function that we could use to approximate our expensive function. The techniques will be similar to the earlier case, but with the difference that here, we might be able to choose the node points ourselves.

Uses of approximation

- One typical use for ***approximating*** functions widely used in practice is to predict the value of the function at points other than those given (or chosen).
- If the point at which we want to predict the function value is inside the interval set by the data points, then we call it ***interpolation***.
- If the point is chosen outside the interval, then we say it is ***extrapolation***.

Other uses

Another use for approximating functions arises in the context of other numerical methods. The most common example is in helping us to take derivatives or compute integrals of other functions.

General Representation

We will first consider the general case of a ***linear form***, in which we will write the approximating (interpolating) function as:

$$v(x) = \sum_{j=0}^{n} c_j \phi_j(x)$$

Here we call $\{c_j\}_{j=0}^{n}$ the unknown coefficients, and $\{\phi_j\}_{j=0}^{n}$ the ***basis functions***.

We will further assume that $\{\phi_j\}_{j=0}^{n}$ are linearly independent.

Linear form

When we say that we will take the ***linear form***, we mean that $v(x)$ is written as a linear combination of the basis functions and not that $v(x)$ is a linear function of x.

Types of Basis Functions

Example

1. Polynomial (monomial): $\phi_j(x) = x^j \quad j = 0, 1, \ldots n$
2. Trigonometric: $\phi_j(x) = \cos(jx) \quad j = 0, 1, \ldots n$
3. Gaussian functions: $\phi_j(x) = e^{-x^2} \quad j = 0, 1, \ldots n$
4. Many, many more

There are many other forms that can be chosen, most of which are used for specific applications or problems with a special structure.

For now we will restrict ourselves to polynomial interpolants.

Polynomial Interpolants

Some of the reasons that polynomials are a good first choice include:

- Easy to both construct and evaluate the interpolants.
- Easy to sum and multiply polynomials
- Easy to differentiate and integrate
- And despite their simple appearance, they can fit many different types of data.

Applications of Polynomial Interpolation

- Polynomials of one form or another are ubiquitous in numerical analysis.
- We will see that they can be used to approximate data, they are used to approximate derivatives in other contexts, and even to help us evaluate integrals numerically.
- Some of the uses of polynomials include approximating commonly used functions and in computer graphics to smooth curves and surfaces of geometric objects.

Stages for using interpolants

Before we go to the next section, we should also say that there are two main stages when using (polynomial) interpolation.

1. ***Constructing an interpolant.*** This usually entails computing the unknown coefficients given a particular set of data points. This can be expensive, but it is done only once for a given data set.
2. ***Evaluating an interpolant.*** Once the polynomial is constructed, we are then able to evaluate the polynomial at some point or some set of points. This is typically much less expensive per evaluation, but we may need to do it many times.

Recall

We can write a polynomial (monomial) as:

$$\begin{aligned} p(x) &= \sum_{j=0}^{n} c_j x^j, \\ &= c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n, \end{aligned} \tag{1}$$

for a given set of $n+1$ data points:

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n).$$

Reminder

A polynomial of degree n has $n+1$ coefficients.

Shifted Power Form

Tip

While the form given by Equation 1 is the most convenient for analysis, it can lead to numerical errors and we will often see instead the *shifted power form*:

$$p(x) = c_0 + c_1(x-a) + c_2(x-a)^2 + \cdots + c_n(x-a)^n,$$

Interpolating Problem

Goal

Given a set of distinct points $\{(x_i, y_i)\}_{i=0}^{n}$, find the $n+1$ coefficients $c_0, c_1, \ldots, c_n$, such that we satisfy the interpolating conditions:

$$p(x_i) = y_i, \quad i = 0, 1, \ldots, n.$$

Note: It is very important that the data points be distinct, i.e.

$$x_i \neq x_j \quad \forall \quad i \neq j.$$

Example: Constructing an Interpolating Polynomial

Example

Suppose we are given the data $(x_0, y_0) = (1, 1), (x_1, y_1) = (2, 3)$ and we wish to interpolate the two data points using a first degree polynomial $n = 1$. In other words we want to construct the interpolating linear polynomial

$$p_1(x) = c_0 + c_1 x. \tag{2}$$

Solution

The interpolating conditions give us:

$$p_1(x_0) = c_0 + 1 \cdot c_1 = 1 \quad (= y_0),$$
$$p_1(x_1) = c_0 + 2 \cdot c_1 = 3 \quad (= y_1).$$

This is a set of two equations in two unknowns, which you can easily verify has the solution:

$$c_0 = -1 \quad c_1 = 2.$$

(cont.)

These coefficients can then be substituted into our polynomial form given by Equation 2 to yield the desired polynomial interpolant:

$$p_1(x) = 2x - 1.$$

Now that we have constructed the polynomial, we are free to evaluate it at any number of other points.

Remark: An alternate derivation of this form can be found in the supplemental section at the end of this section.

Exercise

Construct interpolating polynomial of degree $n = 2$, using an additional third point given below:

Table 1: Data

	$i = 0$	$i = 1$	$i = 2$
x	1	2	4
y	1	3	3

Solution:

$$c_0 = -7/3 \quad c_1 = 4 \quad c_2 = -2/3$$

$$p_2(x) = \frac{1}{3}(-2x^2 + 12x - 7). \tag{3}$$

Vandermonde Matrices

In solving the exercise problem above you will have noticed that you had to set up a set of equations that looked like:

$$\begin{aligned} p_2(x_0) = c_0 + c_1 x_0 + c_2 x_0^2 &= y_0 \\ p_2(x_1) = c_0 + c_1 x_1 + c_2 x_1^2 &= y_1 \\ p_2(x_2) = c_0 + c_1 x_2 + c_2 x_2^2 &= y_2. \end{aligned}$$

These can be written more concisely in matrix notation as:

$$\begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}. \tag{4}$$

or simply as

$$Xc = y.$$

Vandermonde matrices

- This system of linear equations can now be solved for the coefficients c_0, c_1, c_2, which determine our interpolating polynomial.
- The matrix X in Equation 4 is an example of a ***Vandermonde*** matrix.
- The general form of the Vandermone matrix for $n+1$ data points is given by:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \cdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \cdots \\ y_n \end{bmatrix}. \tag{5}$$

Vandermonde (cont.)

Given

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \cdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \cdots \\ y_n \end{bmatrix}. \tag{6}$$

- It can be shown that the $\det X \neq 0$, ***as long as the given data points are distinct.***
- This implies that X is nonsingular and hence the linear system has a unique solution.
- This observation leads to the following theorem.

Polynomial Interpolant Uniqueness

Theorem

For any real data points $\{(x_i, y_i)\}_{i=0}^{n}$ *such that* x_i *are distinct there exists a unique polynomial* $p(x)$ *of degree at most* n *that satisfies the interpolating conditions* $p(x_i) = y_1$ *for* $i = 0, 1, \ldots, n$.

Interpretation

There are many ways to construct an interpolating polynomial - in the end they all yield the same polynomial.

Summary - Monomials as basis functions for interpolation.

- We introduced the concepts of approximation and interpolation - fitting a given set of points with a function, with interpolation being a special case of approximation.
- Considered the general linear form and looked at our first case using polynomials and specifically monomials, i.e. $\phi_j(x) = x^j$.
- This led to the construction of the Vandermonde matrix, which is extremely easy to set up..
- A big disadvantage is that the Vandermonde matrices are notoriously ill-conditioned for even medium dimensions. For $n = 10, \kappa(A) \approx 10^{10}$. As a result, using this approach is highly discouraged. (Notwithstanding this comment, for small n, this approach might be useful.)
- The resulting coefficients don't have a clear relation to the y_i values, which is sometimes desired. This will come up when we study numerical differentiation and integration.

Supplemental Materials Weierstrass Approximation

A natural question to ask is if or when a general function $f(x)$ can be approximated by a polynomial. This was answered by Weierstrass (1885) in a theorem bearing his name.

Theorem (Weierstrass Approximation)

Suppose $f(x)$ is defined and continuous on a closed interval $[a, b]$, and $\epsilon > 0$ is given. Then there exists a polynomial $p(x)$ on $[a, b]$ such that

$$|f(x) - p(x)| < \epsilon \quad \forall x \in [a, b].$$

The most common way of stating this is that any continuous function on a closed interval can be approximated with arbitrary precision by a polynomial.

Alternate derivation of linear interpolation using monomial basis functions

Let's begin with some simple cases - in particular the linear case. Suppose we are given two points (x_0, y_0) and (x_1, y_1), and $x_0 \neq x_1$. Then the straight line passing through these two points is given by the linear polynomial:

$$p_1(x) = \frac{(x_1 - x)y_0 + (x - x_0)y_1}{x_1 - x_0}. \tag{7}$$

Let $x_0, x_1, \ldots, x_n$ be $n + 1$ distinct points on some interval $[a, b]$.

(cont.)

We say that $p_1(x)$ ***interpolates*** the value y_i at the points $x_i, i = 0, 1$ if

$$p_1(x_i) = y_i \quad i = 0, 1.$$

To see that Equation 7 satisfies this property note that:

$$\begin{aligned} p_1(x_1) &= \frac{(x_1 - x_1)y_0 + (x_1 - x_0)y_1}{x_1 - x_0} \\ &= \frac{(0)y_0 + (x_1 - x_0)y_1}{x_1 - x_0} \\ &= \frac{(x_1 - x_0)}{x_1 - x_0} y_1 = y_1 \end{aligned}$$

You can use a similar argument to show that $p_1(x_0) = y_0$.

Lagrange Polynomials

- As a result of the advantages and disadvantages listed above, other approaches have been developed that attempt to address the disadvantages.
- One such approach for constructing an interpolating polynomial consists of using what are known as the Lagrange polynomials for the basis functions.

Construction

Proceeding as before, let's try to construct a polynomial using the general linear form using the following equation:

$$\begin{aligned} p(x) = p_n(x) &= \sum_{j=0}^{n} y_j L_j(x), \\ &= y_0 L_0(x) + y_1 L_1(x) + y_2 L_2(x) + \cdots + y_n L_n(x), \end{aligned} \tag{8}$$

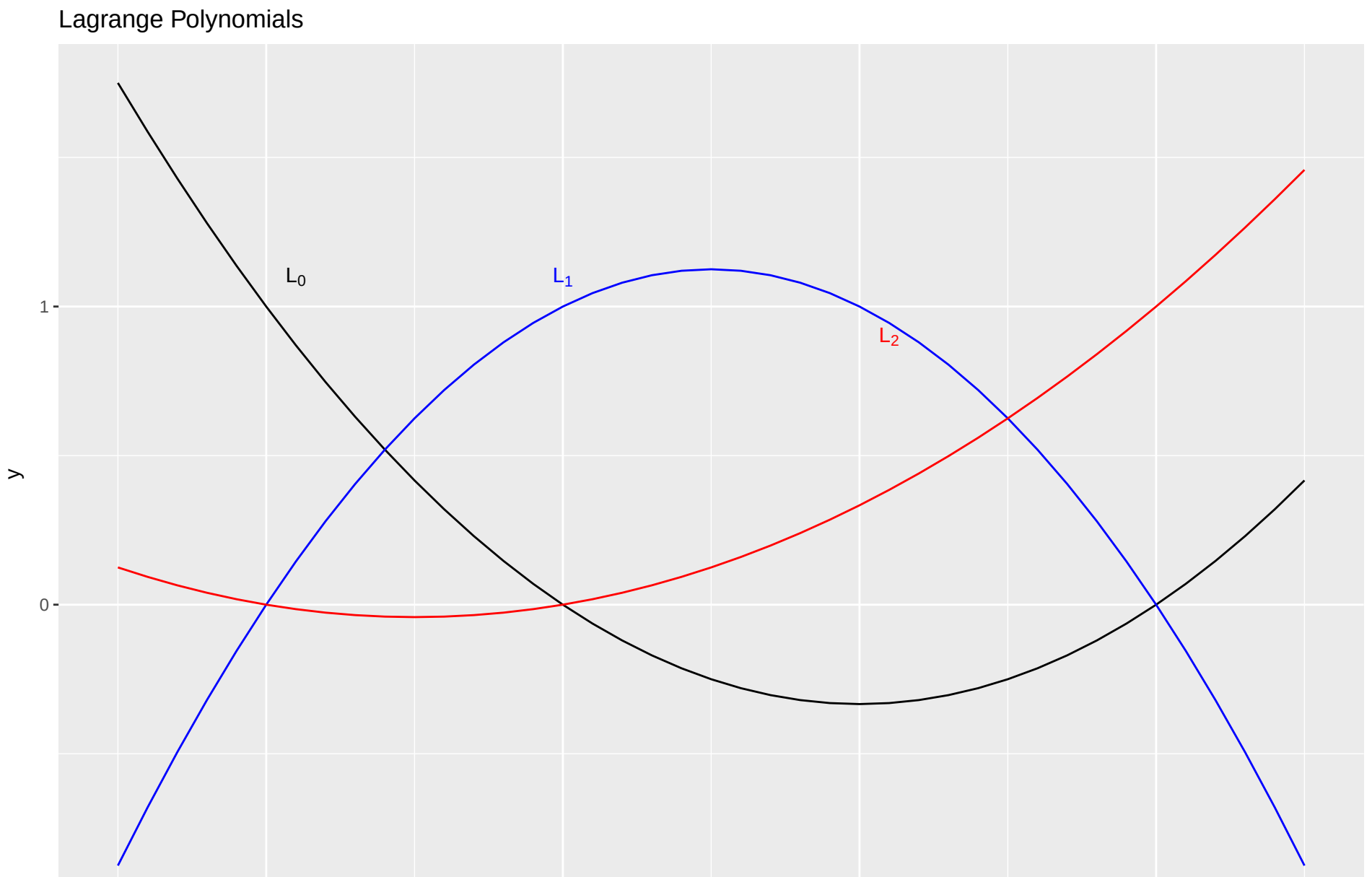
where $L_j(x)$ are called ***Lagrange polynomials*** with the following properties:

$$L_j(x_i) = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases} \quad j = 0, 1, \ldots, n. \tag{9}$$

Using this definition, it isn't hard to show that

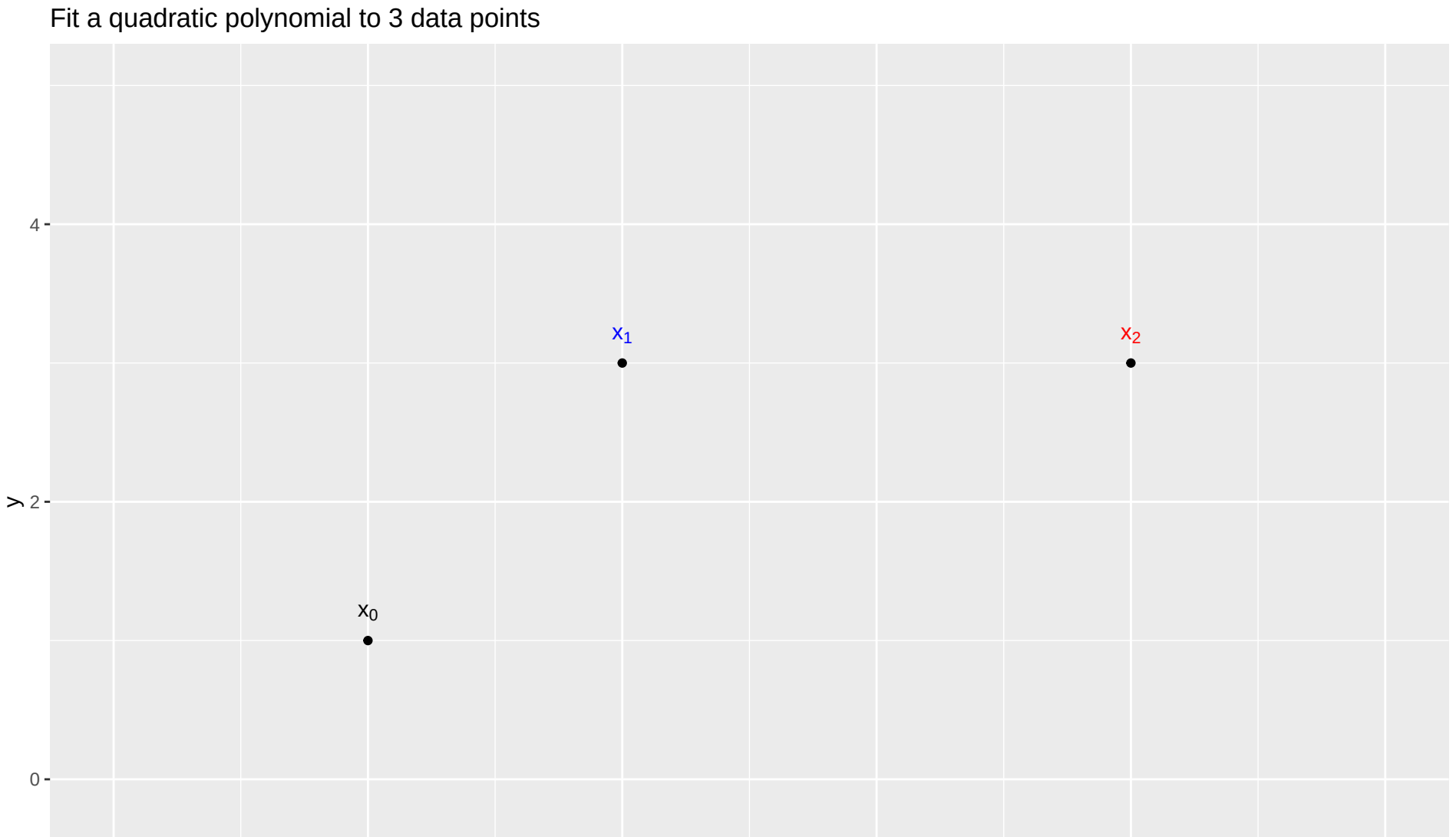
$$p_n(x_i) = y_i, \quad i = 0, 1, 2, \ldots, n.$$

Visually



Example

Construct $p_2(x)$ for data points $(1, 1), (2, 3), (4, 3)$ using the Lagrange polynomial formulation.



Solution

The first step Is to compute the individual $L_j(x)$ for the given data points $(1, 1), (2, 3), (4, 3)$.

Let's start with the first Lagrange polynomial $L_0(x)$. By definition (Equation 9), we know that the polynomial must satisfy the following conditions for $j = 0$.

$$L_0(x_i) = \begin{cases} 0, & i \neq 0 \\ 1, & i = 0 \end{cases}.$$

(cont.)

We also know that L_0 must be a quadratic polynomial (***Why?***) As a result, we know that the general form for L_0 must be:

$$L_0(x) = a \cdot (x - r_1)(x - r_2),$$

where r_1, r_2 are the two roots.

To satisfy condition $L_0(x) = 0$ at the two nodes other than x_0, means that the two other nodes, $x_1 = 2, x_2 = 4$ must be the two roots of the quadratic.

$$L_0(x) = a \cdot (x - 2)(x - 4).$$

(cont.)

We can now use the remaining condition $L_0(x_0) = 1$ to see that:

$$\begin{aligned} L_0(x_0) = 1 &= a \cdot (x_0 - 2)(x_0 - 4), \\ &= a \cdot (1 - 2)(1 - 4) \qquad \text{plugging in } x_0 \\ 1 &= 3a \end{aligned}$$

Therefore $a = 1/3$, giving us the solution:

$$L_0(x) = \frac{1}{3}(x - 2)(x - 4)$$

(cont.)

Using the same procedure, we can compute:

$$L_1(x) = -\frac{1}{2}(x-1)(x-4).$$

and finally:

$$L_2(x) = \frac{1}{6}(x-1)(x-2).$$

I leave the computation of these two for you to practice on.

Putting it all together

Using (Equation 8), we can now combine the 3 Lagrange polynomials and multiply by the corresponding y_j values to gives us the desired interpolating polynomial :

$$p_2(x) = y_0 \cdot \frac{(x-2)(x-4)}{3} - y_1 \cdot \frac{(x-1)(x-4)}{2} + y_2 \cdot \frac{(x-1)(x-2)}{6}$$

Substituting for the values of y we get:

$$p_2(x) = \frac{1}{3}(x-2)(x-4) - \frac{3}{2}(x-1)(x-4) + \frac{3}{6}(x-1)(x-2) \tag{10}$$

Visually

Figure 2 shows all three of the Lagrange polynomials in addition to the final interpolating polynomial $p_2(x)$ of degree 2 for the 3 given data points.

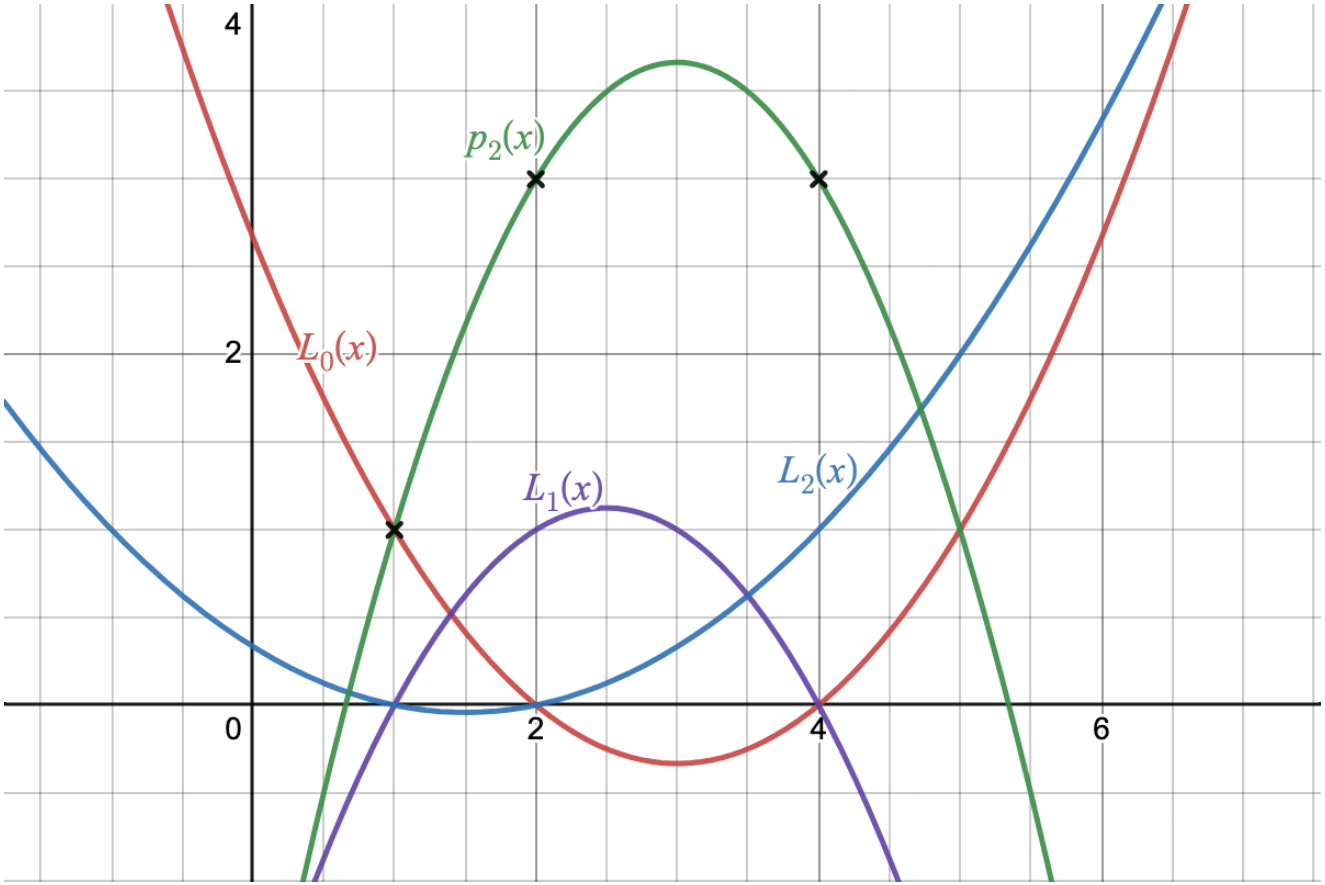


Figure 2: Lagrange Polynomials and resulting 2nd degree polynomial

Uniqueness (again)

Important

One should note at this point that while the two approaches appear different, they yield the same second degree interpolating polynomial. In other words Equation 3 $=$ Equation 10 .

Remark: It can be easily shown that there is only one quadratic interpolating polynomial for any three (distinct) points.

Uniqueness Proof

Proof. Suppose there were 2 polynomials $p_2(x), q_2(x)$ that interpolate the given points. Then define

$$r(x) = p_2(x) - q_2(x).$$

First note that $r(x)$ is also of degree ≤ 2. But

$$r(x_i) = p_2(x_i) - q_2(x_i) = y_i - y_i = 0, \quad i = 0, 1, 2$$

This means that $r(x)$ has three distinct roots and has degree ≤ 2. That isn't possible unless $r(x) \equiv 0$ by the Fundamental Theorem of Algebra. Therefore $p_2(x) = q_2(x)$.

General Form of Lagrange Polynomials

To generalize this idea to a higher number of data points, first notice that we can write the Lagrange polynomial fitting the n points as:

$$\begin{aligned} L_j(x) &= \frac{(x-x_0)(x-x_1)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_n)}{(x_j-x_0)(x_j-x_1)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)} \\ &= \prod_{i=0, i\neq j}^{n} \frac{x-x_i}{x_j-x_i} \quad j = 0, 1, \ldots n. \end{aligned} \tag{11}$$

Note that the term in the numerator:

$$(x-x_0)(x-x_1)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_n) = 0 \quad \forall\ x_i \neq x_j$$

is missing the jth term, hence for any node point other than x_j the numerator will be equal to 0.

General form (cont.)

In addition, if $x = x_j$, then the numerator and the denominator are equal, hence $L_j(x)$ satisfies the conditions for a Lagrange polynomial.

Given this formulation, the construction and evaluation of the interpolating polynomials proceeds as before.

Construction

1. ***Construction of*** $L_j(x)$.

To simplify notation, we define the following terms:

$$\begin{aligned} \rho_j &= \prod_{i \neq j}^{n} x_j - x_i \quad j = 0, 1, \dots, n, \\ w_j &= \frac{1}{\rho_j} \quad j = 0, 1, \dots, n, \end{aligned} \tag{12}$$

where the w_j are called the ***barycentric weights***. ([@berrut2004])

Remark: Having computed these weights that's it for construction!

Notice also that this computation can be done in $O(n^2)$ flops.

2 *Evaluation of* $p_n(x)$

To evaluate the polynomial we just need to bring the different parts together. In order to simplify this process, first notice that if we define

$$\begin{aligned}\Psi(x) &= (x - x_0)(x - x_1)\cdots(x - x_n) \\ &= \prod_{i=0}^{n} (x - x_i)\end{aligned} \tag{13}$$

then we can write the Lagrange polynomials as:

$$L_j(x) = \frac{\Psi(x)}{(x - x_j)}$$

Here all we've done is notice that the numerator of Equation 11 is nothing more than Equation 13 divided by the missing term $(x - x_j)$.

Evaluation (cont.)

This leads to the following form for the interpolating polynomial.

$$p_n(x) = \Psi(x) \sum_{j=0}^{n} \frac{w_j \cdot y_j}{(x - x_j)}. \tag{14}$$

- Notice that since $\Psi(x)$ doesn't depend on j it can be pulled out of the summation sign.
- This form also has the advantage that we just need to compute the $\Psi(x)$ term once!
- The rest of the evaluation can be done if $O(n)$ flops.

Example

Let's do one quick example using the data given below:

Table 2: Data

	$i = 0$	$i = 1$	$i = 2$	$i = 3$
x	-1.1	1.1	2.2	0.0
y	0.0	6.75	0.0	0.0

Example Solution: Construction

First we construct the ***barycentric weights***, $w_j = 1/\rho_j,\ j = 0, 1, 2, 3.$

- $j = 0$

$$\begin{aligned}
\rho_0 &= \prod_{i \neq j}^{n} (x_j - x_i), \qquad j = 0 \\
&= (x_0 - x_1)(x_0 - x_2)(x_0 - x_3) \\
&= (-1.1 - 1.1)(-1.1 - 2.2)(-1.1 - 0) \\
&= (-2.2)(-3.3)(-1.1) \\
&= -7.986
\end{aligned}$$

Example Solution: Construction (cont.)

- Similarly for $j = 1$

$$\begin{aligned}
\rho_1 &= \prod_{i \neq j}^{n} (x_j - x_i), \qquad j = 1 \\
&= (x_1 - x_0)(x_1 - x_2)(x_1 - x_3) \\
&= (1.1 - (-1.1))(1.1 - 2.2)(1.1 - 0) \\
&= (2.2)(-1.1)(1.1) \\
&= -2.662
\end{aligned}$$

Example Solution: Construction (cont.)

- I'll leave the computation of the other two as an exercise.

$$\begin{aligned} \rho_2 &= (x_2 - x_0)(x_2 - x_1)(x_2 - x_3) \qquad j = 2 \\ &= 7.986 \\ \rho_3 &= (x_3 - x_0)(x_3 - x_1)(x_3 - x_2) \qquad j = 3 \\ &= 2.662 \end{aligned}$$

Having computed the ρ values, the barycentric weights, w_j, can now be easily computed.

An interesting feature of this formulation is that the construction of the interpolating polynomial ***does not depend on the values of*** y - we only need the data nodes x_i.

Computational Tip

Tip

Once we have computed the weights for a given set of x values, we can use them for any function whatsoever (as long as we know values at those nodes). This also means that since we never assumed any order of the nodes, the weights are independent of the order that the nodes are in. Based on this, it is not hard to show that Equation 14 can be re-written in the more elegant form of:

$$p_n(x) = \frac{\sum_{j=0}^{n} \frac{w_j \cdot y_j}{(x - x_j)}}{\sum_{j=0}^{n} \frac{w_j}{(x - x_j)}}. \tag{15}$$

where the term $\Psi(x)$ has been eliminated.

Summary - Lagrange basis functions as an interpolating function

- We introduced the concept of a Lagrange polynomial that can be used as a basis for an interpolating function
- We also showed how to construct an interpolating polynomial using a special set of weights called barycentric weights.
- The construction of this polynomial can be shown to be $O(n^2)$, and does not depend on the values of y or $f(x)$.
- Once the construction has been completed, the interpolating polynomial can be used for any number of different functions, each evaluation costing $O(n)$ flops.
- It can also be shown that if we choose the data point properly, then the algorithm for constructing and evaluating this polynomial is stable. ([@higham2004])

References

1. Interpolation and Approximation. [@davis1975]
2. Barycentric Lagrange Interpolation. [@berrut2004]
3. The numerical stability of barycentric Lagrange interpolation. [@higham2004]