MATH 365
Polynomial Interpolation

1 Introduction

Linear regression uses a line, plane or hyperplane in high dimensional space to describe (model) a collection of data points. More specifically, in two-dimensions, given a set of n points (x_i, y_i), $i = 1, 2, ..., n$, simple linear regression involves the assumption that the data are related through the equation $y_i \approx b_0 + b_1 x_i$ and the goal is to find coefficients b_0 and b_1.

In two dimensions it is not possible for every data point to satisfy $y_i = b_0 + b_1 x_i$ unless $n = 2$ i.e. there are only two data points. In linear algebra terminology, when $n = 2$, we can fit each data point to a line because there are the same number of equations as unknowns. This means we can set up a square linear system $\mathbf{y} = \mathbf{X}\mathbf{b}$ where

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}.$$

As long as the two data points have two different y values corresponding to x_1 and x_2, these two points uniquely determine a line.

When $n > 2$ the linear system of equations that results when we fit the data to a line is *overdetermined*, i.e. there is a greater number of equations than unknowns. Most of the time this means there is no solution and hence no coefficients b_0 and b_1 that produce a line that fits all of the data. We got around this issue by finding b_i that minimize the sum of squared errors: $S = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$.

2 Interpolation with the Vandermonde matrix

Rather than minimize the error in our data and the predicted line at the data points, let's extend the idea of creating square systems of equations so that we can fit (or model) each data point exactly. If we have $n + 1$ data points we can create the same number of equations by fitting them to a nth degree polynomial

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_n x^n.$$

This is called polynomial interpolation when $y_i = p_n(x_i)$ for all (x_i, y_i), $i = 1, 2, ..., n + 1$.

The system of equations for polynomial interpolation is

$$\begin{aligned} y_1 &= a_0 + a_1 x_1 + a_2 x_1^2 + a_3 x_1^3 + \ldots + a_n x_1^n \\ y_2 &= a_0 + a_1 x_2 + a_2 x_2^2 + a_3 x_2^3 + \ldots + a_n x_2^n \\ &\vdots \\ y_{n+1} &= a_0 + a_1 x_{n+1} + a_2 x_{n+1}^2 + a_3 x_{n+1}^3 + \ldots + a_n x_{n+1}^n. \end{aligned}$$

The matrix equation that results from this system is $\mathbf{y} = \mathbf{Va}$ with

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & \dots & x_1^n \\ 1 & x_2 & x_2^2 & x_2^3 & \dots & x_2^n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n+1} & x_{n+1}^2 & x_{n+1}^3 & \dots & x_{n+1}^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}.$$

where the matrix $\mathbf{V}$ in the system is called a *Vandermonde* matrix.

Polynomial interpolation using the Vandermonde matrix is not typically used in practice. It is not practical because for a large number of data points it is expensive to solve the system of equations and the matrix has a large condition number. A large condition number means if you change the data slightly you will get a very different result. In other words the problem is unstable. Also, if you add data points, you have to solve an entirely new problem.

3 Lagrange Interpolation

Lagrange interpolation is an attractive alternative to using the Vandermonde matrix because you don't have to solve a system of equations to find the interpolating polynomial. In addition, you can easily add data points without having to start the process from scratch.

The difference in between using the Vandermonde matrix and Lagrange interpolation is how we write the polynomial. For example, $y = 2 - 3x + x^2$ and $y = (x-2)(x-1)$ are different ways of writing the same quadratic polynomial. The former would be the view for the Vandermonde matrix while the latter is how Lagrange Interpolation is written.

If we have two data points (x_i, y_i), $i = 1, 2$, the unique line that fits them is represented with the Lagrange polynomials as

$$p_1(x) = y_1 l_1(x) + y_2 l_2(x); \quad l_1(x) = \frac{(x - x_2)}{(x_1 - x_2)}, l_2(x) = \frac{(x - x_1)}{(x_2 - x_1)}.$$

Similarly if we have three data points (x_i, y_i), $i = 1, 2, 3$, the unique quadratic polynomial that fits them is represented with the Lagrange polynomials as

$$\begin{aligned} p_2(x) &= y_1 l_1(x) + y_2 l_2(x) + y_3 l_3(x); \\ & l_1(x) = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)},\ l_2(x) = \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)},\ l_3(x) = \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)}. \end{aligned}$$

Notice that the coefficients for the polynomials are simply the y_i data which is why we don't have to solve a system of equations. In addition, we can add data more simply than in the Vandermonde matrix case.

For the more general case with n data points the Lagrange polynomial form of interpolation is written

$$p_n(x) = y_1 l_1(x) + y_2 l_2(x) + \ldots + y_n l_n(x) = \sum_{i=1}^{n} y_i l_i(x)$$

where

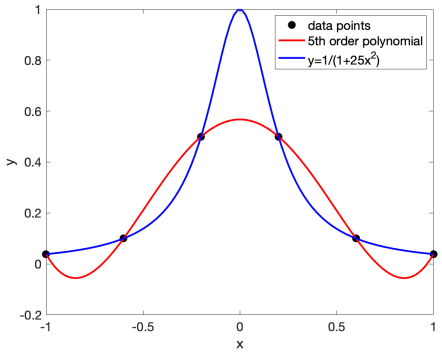
$$l_i(x) = \frac{(x - x_1)(x - x_2) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_1)(x_i - x_2) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)} = \frac{\Pi_{j \neq i}(x - x_j)}{\Pi_{j \neq i}(x_i - x_j)}$$

4 Runge Phenomenon

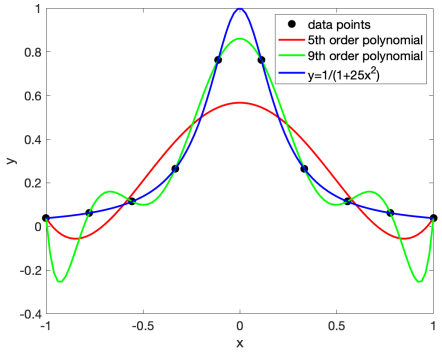
There are some pitfalls that occur when fitting a polynomial to data. As an example, let's simulate six data points by sampling the function $y = \frac{1}{1+25x^2}$ at equally spaced points on the interval $[-1, 1]$. We can create the (x, y) data pairs $(-1, 0.0385)$, $(-0.6, 0.1)$ $(-0.2, 0.5)$ $(0.2, 0.5)$ $(0.6, 0.1)$ $(1, 0.0385)$ in MATLAB with the following commands

```
n=6;
f=@(x) 1./(1 + 25*x.^2);
x=linspace(-1,1,n);
y=f(x);
```

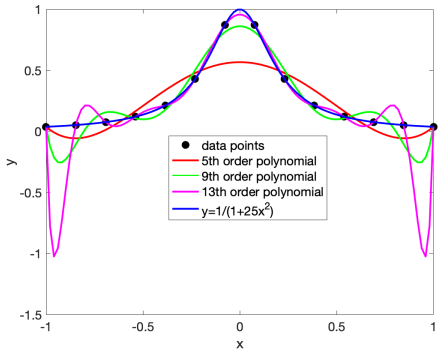
Polynomial interpolation with a 5th degree polynomial results in the following graph



We see that the 5th degree interpolating polynomial does a particularly poor job of interpolating at the peak and near the endpoints of the interval. Lets try to increase the accuracy by adding more data points and letting $n = 10$.

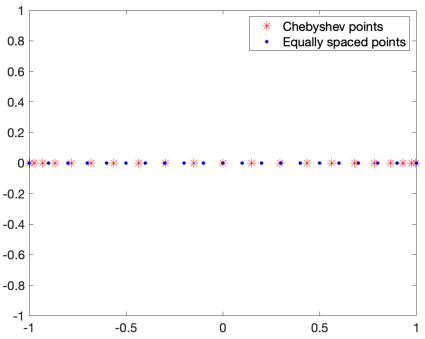


We are getting closer to modeling the peak, but the end points are getting worse. Notice that the y axis range increased to -0.4. Let's again try to improve the approximation by interpolating 14 points



Notice that the approximation is getting worse at the endpoints as we increase the number of data. This is called Runge phenomenon and unfortunately it can happen in situations where data is collected on an equally spaced grid.

We can avoid Runge phenomenon by using data at points that are not equally spaced. The data spacing that gives the best approximation are the Chebyshev points. Chebyshev points are defined on the interval $[-1, 1]$ as $x_k = \cos\left(\frac{2k-1}{2n}\pi\right)$ for $k = 1, \ldots, n$. Here is a plot of them compared to an equally spaced grid:

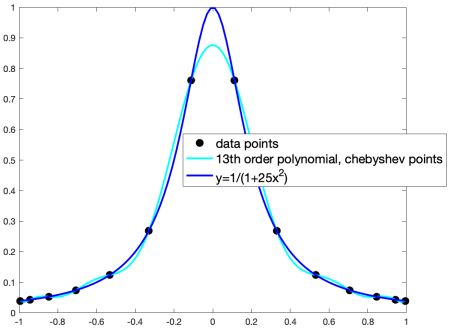


We see that the Chebyshev points are more clustered at the endpoints which gives some intuition as to why they help prevent Runge phenomenon. The Chebyshev points can be defined on any interval $[a, b]$ by $x_k = \frac{1}{2}(a+b) + \frac{1}{2}(b-a)\cos\left(\frac{2k-1}{2n}\pi\right)$ for $k = 1, \ldots, n$.

Let's use the Chebyshev points to interpolate $y = \frac{1}{1+25x^2}$ on the interval $[-1, 1]$ with the MATLAB commands

```
n=14;
f=@(x) 1./(1 + 25*x.^2);
x=cos((2.*[1:n]-1)*pi./(2*(n)));
y=f(x);
```

Then we get a good approximation



5 Splines

We can improve the behavior of the single polynomial interpolant by adjusting the location of the points where we interpolate as we did with Chebyshev points. However, if we do not have a choice as to where to place the data points then we are not able to interpolate all of the data. Another option is to construct piecewise polynomials through subsets of the data through **spline interpolation**.

The most common approach to spline interpolation is to create separate cubic polynomials

$$S_i(x) = a(x - x_i)^3 + b(x - x_i)^2 + c(x - x_i) + d$$

over subintervals $[x_i, x_{i+1}]$ with $S_i(x_i) = y_i$ and $S_i(x_{i+1}) = y_{i+1}$. Note that there are four unknowns in each subinterval and only two interpolation points. A square system of equations is formed by adding continuity conditions such as $S_i(x_{i+1}) = S_{i+1}(x_{i+1})$ and $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$. The result is a tridiagonal linear system that is solved for coefficients of each of the i cubic polynomials that make up the interpolating spline over the whole interval.

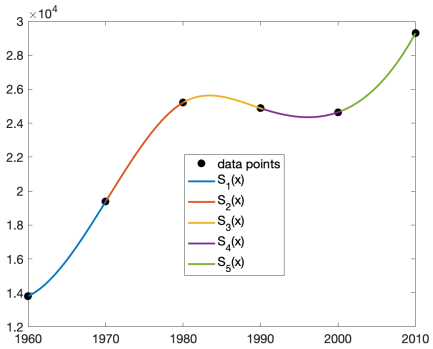
As an example, consider the census data from Billing, Montana in ten year intervals from 1960 to 2010. Since there are six data points there are five cubic polynomials in the spline interpolant. We find the coefficients for each of the five cubics all at once using the *spline* function in MATLAB. Let's consider only the white population:

```
C_table=readtable('census.txt');
C_array=table2array(C_table);
x=C_array(:,1);
y=C_array(:,2);
p_spline = spline(x,y);
```

The coefficients a, b, c, d for each piecewise polynomial are in p_spline. Given these coefficients we can find the interpolating polynomial over the whole interval using *ppval* . As usual we will evaluate the polynomial at 100 points to get a continuous looking interpolant

```
a=min(x);b=max(x);
xs = linspace(a,b,100);
y_spline = ppval(p_spline,xs);
```

The following is a graph of y_spline. To the naked eye our result looks like the fifth order interpolant even though y_spline is a series of cubic polynomials rather than one single fifth order interpolant. I've highlighted each piecewise cubic polynomial in the graph, but normally you would just plot(xs,p_spline) and not highlight the fact that there are different polynomials on each subinterval.



You can predict the population in 2005 by typing

y_spline = ppval(p_spline,2005);

and the prediction is 26,136 people.

The piecewise polynomial found by spline is returned as a structure in p_spline. Since the functions *spline* and *ppval* are compatible we didn't have to think about this when creating and plotting the interpolating polynomial. However, if we want to understand how many piecewise polynomials are in the whole interpolating polynomial and possibly values their coefficients, we'll need to understand the structure. The structure has the following fields which can be displayed by typing p_spline:

```
p_spline =

  struct with fields:

      form: 'pp'
    breaks: [1960 1970 1980 1990 2000 2010]
     coefs: [5×4 double]
    pieces: 5
     order: 4
       dim: 1
```

A description of each part of the structure is given in the following table

output	meaning
`form: 'pp'`	the form is a piecewise polynomial
`breaks: [1960 1970 1980 1990 2000 2010]`	the start and end of each S_i interval
`coefs: [5x4 double]`	each row contains the coefficients of the ith polynomial S_i
`pieces: 5`	number of piecewise polynomials on the interval
`order: 4`	order of the polynomial
`dim: 1`	dimension of your interpolating polynomial

You access each part of the structure by typing p_spline.form, p_spline.breaks, p_spline.coefs etc.

For example, p_spline.coefs is a 5x4 matrix and we access the second row by typing p_spline.coefs(2,:). The result is

```
ans =

   1.0e+04 *

  -0.000165453333333   0.000135000000000   0.073565333333333   1.937300000000000
```

With this information we can form $S_2(x)$ that is defined on the interval $[1970, 1980]$

$$S_2(x) = -1.655(x - 1970)^3 + 1.35(x - 1970)^2 + 735.65(x - 1970) + 19,373.$$

6 Individual Blackboard questions

Part 1

1. Given the data points (-2,4) and (-1,3) identify the Vandermonde matrix.
2. Given the data points (-2,4), (-1,3) and (0,5) identify the Vandermonde matrix.
3. Given the data points (-2,4) and (-1,3) identify the Lagrange polynomials.
4. Given the data points (-2,4), (-1,3) and (0,5) identify the Lagrange polynomials.

Part 2

1. The following are Chebyshev points on the interval $[-1, 1]$.
2. The following are Chebyshev points on the interval $[2, 3]$.
3. The following are values of $y = e^{-x^2}$ evaluated at the Chebyshev points on the interval $[-1, 1]$.

Part 3

Use the Census data from Billings, Montana to interpolate the population of black people from 1960 to 2010 and answer the following questions:

1. How many piecewise polynomials are defined on the interval $[1960, 2010]$?
2. The formula for the piecewise cubic spline interpolant is the same as the formula for the fifth degree polynomial interpolant.
3. The graph of the piecewise cubic spline interpolant looks the same as the graph for the fifth degree polynomial interpolant.
4. The coefficients for the cubic spline on the interval $[1980, 1990]$ are:

7 Group Work

You may work in groups or individually to answer the following questions.

Part 1

1. (a) Use the *vander* command in MATLAB to create the Vandermonde system with the census data in census.txt. You may type `>>` help vander at the MATLAB prompt to learn how to use the function, or Google it.
   (b) Find the coefficients for polynomial interpolation of the total population. The backslash command \ solves a linear system of equations in MATLAB. For example `x = A\b` solves the system of linear equations $Ax = b$.

   Be prepared to discuss: (1) inputs to the MATLAB function *vander*, (2) any issues you had with solving the system of equations with the \ command and (3) the degree of the polynomial that interpolates the data.

2. Plot the data and the interpolating polynomial on the same graph and consider the following

   (a) Test your understanding of polynomial interpolation by coding the fifth degree polynomial defined by the coefficients you found in 1b, i.e. $p_5(x) = a_0 + a_1x + \ldots a_5x^5$.
   (b) Note that the interpolating polynomial is defined at infinitely many points x. To get a good view of it you will need to create a linearly spaced vector `x` with a 100 or so points. You can do this in MATLAB with `linspace(min(xdata),max(xdata),100);` where `xdata` are the x values of the data points.
   (c) Try using the MATLAB function *polyval* to create $p_5(x)$ rather than typing it out yourself. Type `>>` help polyval at the MATLAB prompt to learn how to use the function, or Google it.

   Be prepared to discuss: (1) the difference between plotting the data points and the polynomial (2) how you chose the x-values in your plot, (3) how plotting helps you see if you have the right answer or not (4) how the coefficients a_i are arranged in the Vandermonde matrix and *polyval* and (5) any error messages you got from MATLAB.

3. Download the file lagrange.m into the directory where you are running MATLAB. You will use this function to find the same $p_5(x)$ as in 2a but use Lagrange interpolation so that it will be in the form $p_5(x) = y_1l_1(x) + y_2l_2(x) + \ldots + y_6l_6(x)$. The inputs into the function `lagrange(xdata,x,j)` are (i) `xdata` - the x values of the data points, (ii) `x` - the linearly spaced vector of 100 or so x values and (iii) `j`, the jth polynomial $l_j(x)$.

   (a) Use lagrange.m to find $l_4(x)$ and plot it.
   (b) Use lagrange.m to find $l_2(x)$ and plot it on the same graph as $l_4(x)$. Use a legend and upload your plot in pdf, jpg or png format into Blackboard. Do not upload a file in .fig format.
   (c) Write a loop that finds all six $l_j(x)$ and sums them up to form $p_5(x)$ which is called `px` in this code segment:

```
px = 0;
for j = 1:6
    px = px + lagrange(xdata,x,j)*y(j);
end
```

where `y(j)` are the total population data points.

(d) Plot the data with points and the interpolating polynomial with a line.

Be prepared to discuss: (1) What the graph of $l_2(x)$ and $l_4(x)$ as compared to the graph of $p_5(x)$, (2) how the number of data points relates to the degree and number of Lagrange polynomials, (3) issues with writing a loop that calls lagrange.m many times and (4) any issues you had with plotting the data points and fitted polynomial.

Part 2

1. The dataset in the file air_data_day.txt available in Blackboard contains the hourly concentration of PM2.5 in micrograms per cubic meter for a particular measuring station in Salt Lake County for the first day of 2016. PM2.5 is the concentration in the air of fine particulates. The Environmental Protection Agency (EPA) set the threshold the of 35 micrograms per cubic meter, which corresponds to an Air Quality Index (AQI) of 101. If the concentration of PM2.5 exceeds this threshold the air is considered unhealthy for sensitive individuals or populations.

   (a) Load the dataset into MATLAB and plot the data.

   (b) Use Lagrange interpolation to find and plot the interpolating polynomial. Use a legend and upload your plot of the data and interpolating polynomial in pdf, jpg or png format into Blackboard. Do not upload a file in fig format.

   Be prepared to discuss: (1) When looking at the plot of just the data, identify a function(s) that would model the data well, (2) the order of the polynomial that interpolates the data in air_data_day.txt (3) the reliability of your interpolating polynomial, i.e. if it is reliable, why? If it is not reliable, why not?

2. Your plot in 1b should show Runge phenomenon. Interpolation at the Chebyshev points would fix this problem but data at these locations is not possible because the data is collected on the hour, in even intervals. However, we can still get a better approximation if we only use the data points that are close to the Chebyshev points.

   (a) The following MATLAB code segment will find N points in the data that most closely match measurements at the Chebyshev points and store them in (x_new,y_new). Note that N should be less than the number of data points in the original data set (x,y) which lies on the interval x∈[a,b].

   ```
   x_cheb = (a+b)/2 + (b-a)/2*cos(pi/N*((1:N)-.5));
   [minValue,closestIndex] = min(abs(bsxfun(@minus,x_cheb, x')));
   x_new=x(closestIndex);
   y_new=y(closestIndex);
   ```

   Find the set of 5, 10, 15, and 20 points that most closely match data at the Chebyshev points. Identify if data points are repeated in any of the sets. Recall from the lab the formulae for Lagrange interpolation and be prepared to discuss what happens when you use a data set that has repeated data points.

   (b) Find the interpolating polynomial at the $N = 5, 10, 15, 20$ data points you found in 2a. Plot the polynomials along with the original data set. Use a legend and upload your plot of the data and interpolating polynomials in pdf, jpg or png format into Blackboard.

Do not upload a file in fig format. Be prepared to discuss which polynomial you think best approximates the data and why. Consider both how well the polynonmial you recommend represents the data and also how well it can predict future air quality.

(c) You'll notice that your polynomial in 2b does not go through all of the data points. This means that there are errors, or residuals, at the data points.

i. Evaluate the polynomial interpolant at all data points. You can do this in MATLAB with the following code segment

```
px_data = 0;
for j = 1:N
    px_data = px_data + lagrange(x_new,xdata,j)*y_new(j);
end
```

where (x_new,y_new) are the N points that most closely match Chebyshev points and xdata are the original data points. The value of the polynomial interpolant at all data points is stored in px_data.

ii. Use your values of the polynomial interpolant at all data points to find and plot the error, or residuals at all the data points when $N = 20$.

iii. Use the MATLAB function *subplot* to plot the error for all values of N

```
subplot(2,2,1),plot(xdata,resid_5),title('N=5')
subplot(2,2,2),plot(xdata,resid_10),title('N=10')
subplot(2,2,3),plot(xdata,resid_15),title('N=15')
subplot(2,2,4),plot(xdata,resid_20),title('N=20')
```

Upload your figure into Blackboard.

iv. Write MATLAB functions that calculates the variance in the residual as we did for linear regression. The function should takes in the original values of the data points (y) and the value of the interpolating polynomial at the times the data were collected (x).

```
function sigma2= var_resid(ydata,px_data)
        .
        .
        .
```

Use this functions to find the variance in the residual with $N = 5, 10, 15, 20$ data points.