Math for Data Science

Continue Curvilinear Models

Joanna Bieri DATA100

Important Information

- Email: joanna_bieri@redlands.edu
- Office Hours take place in Duke 209 unless otherwise noted – Office Hours Schedule

Today's Goals:

- Continue Empirical Modeling
- Nonlinear Models - Linearization
- Exponent and Log models

Last Time - Polynomial Regression

We looked at the ice cream data and did a polynomial regression! We wanted a quadratic function because of the shape of the data.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

This thing was linear in β. Instead we added x^2 to data to our regression as if it were just another data point.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

Last Time - Polynomial Regression

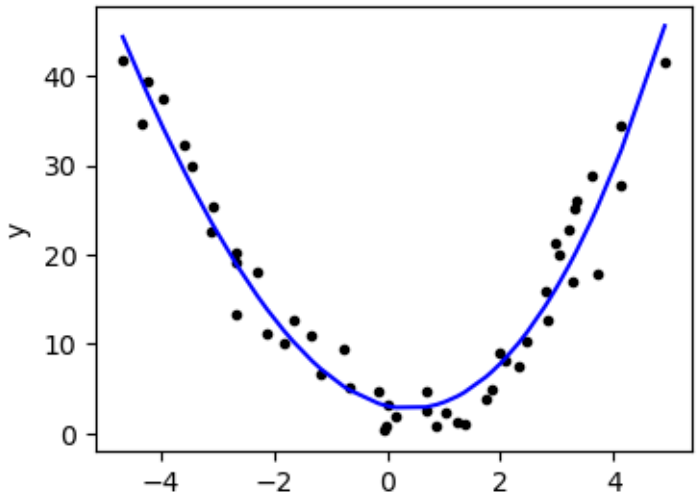
We can "trick" linear regression into doing nonlinear regression! This is what polyfit does when we tell it we want a second order polynomial.

```
{python] np.polyfit(x,y,2)
```

	Temperature (°C)	Ice Cream Sales (units)
0	-4.662263	41.842986
1	-4.316559	34.661120
2	-4.213985	39.383001
3	-3.949661	37.539845
4	-3.578554	32.284531
5	-3.455712	30.001138
6	-3.108440	22.635401
7	-3.081303	25.365022
8	-2.672461	19.226970
9	-2.652287	20.279679

Higher order Polynomials

Why stop at x^2 we have a computer we can go crazy!!!!



Higher order Polynomials

Questions;

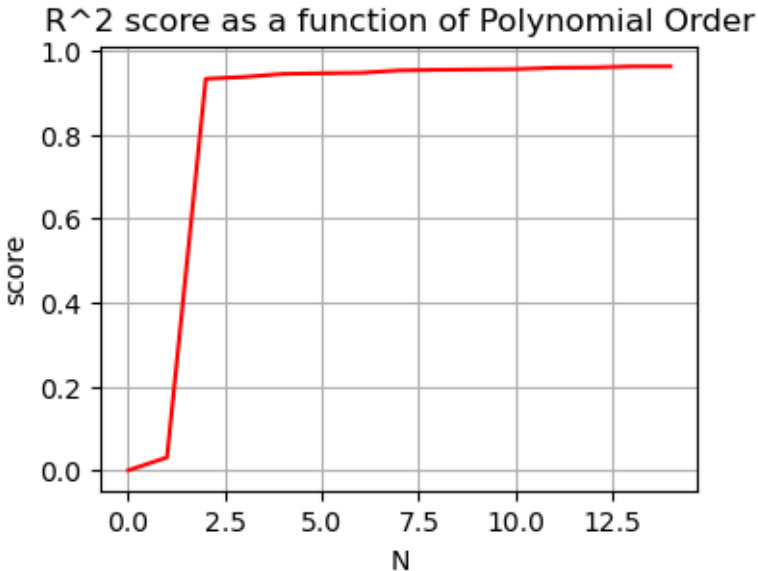
1. Rerun the code above for higher and higher order polynomials, do the results get better?
2. Do you reach a point where making the model more complicated doesn't actually improve the results that much?

Higher order Polynomials

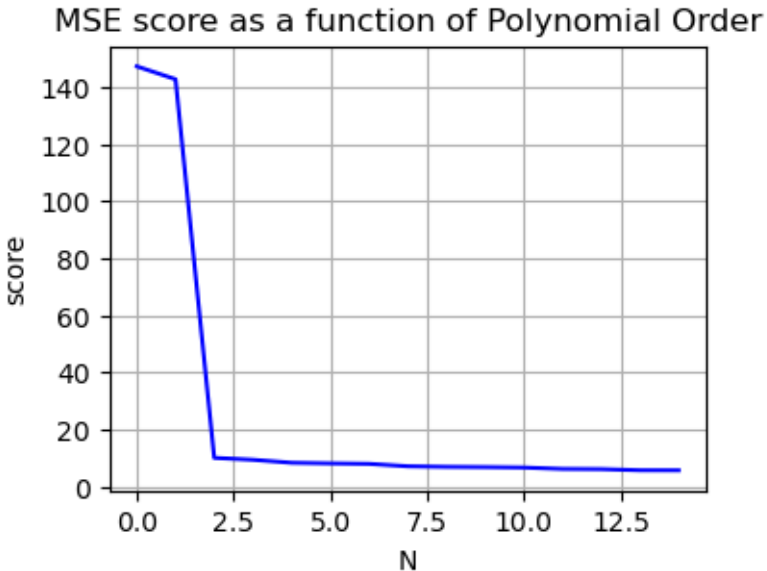
Occams Razor

We want to increase the R^2 value to be as close as possible to 1 and decrease the mean squared error to as close as possible to zero, **without making our model ridiculously complicated**. Stop before you get diminishing returns!

Higher order Polynomials



Higher order Polynomials



General Idea of Linearized Functions

We could imagine data that has all sorts of dependencies, curves, etc. For example:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 \sin(x) + \beta_4 \sqrt{x}$$

as long as we can write the function linearly in beta

$$y = \beta_0 + \beta_1 f_1(x) + \beta_2 f_2(x) + \beta_3 f_3(x) + \beta_4 f_4(x) ...$$

we can do a linear regression!

BEWARE np.polyfit() can only deal with polynomials, not other more complicated functions!

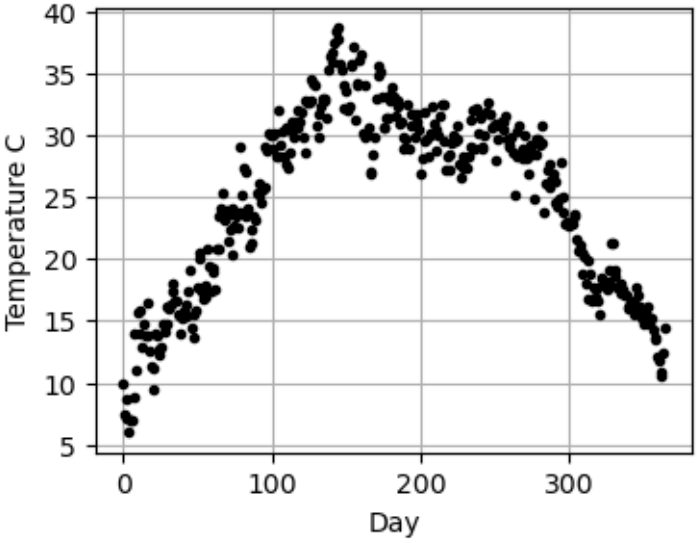
NOTE Not all functions can be linearized. We have to be able to write it in the form above.

Delhi Climate Data

Here we will look at some climate data and restrict ourselves to just one year to see if we can model the yearly swing in temperature.

- First we will try a polynomial fit
- Then we will see how we could fit a sine function

Delhi Climate Data



Delhi Climate Data

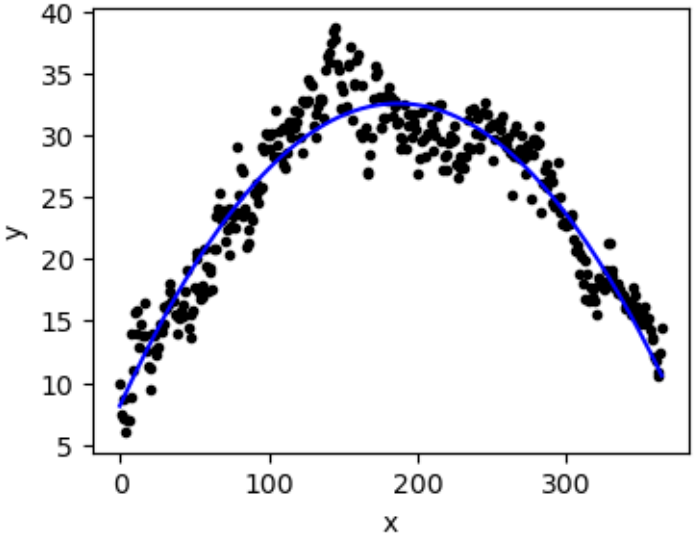
Looking at this data it looks like $y = -x^2$ (aka. upside down parabola) that has been shifted and stretched to fit the year. We will assume we can fit it with a function of the form:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

and just use np.polyfit(x,y,2)

```
array([-7.01158700e-04,  2.61972640e-01,  8.12191801e+00])
```

Delhi Climate Data



Delhi Climate Data

R squared:

`0.8946935431841873`

MSE:

`5.765082904243028`

This worked pretty well!

Polynomial was not our only choice! What if we tried another function? Like sine or cosine?

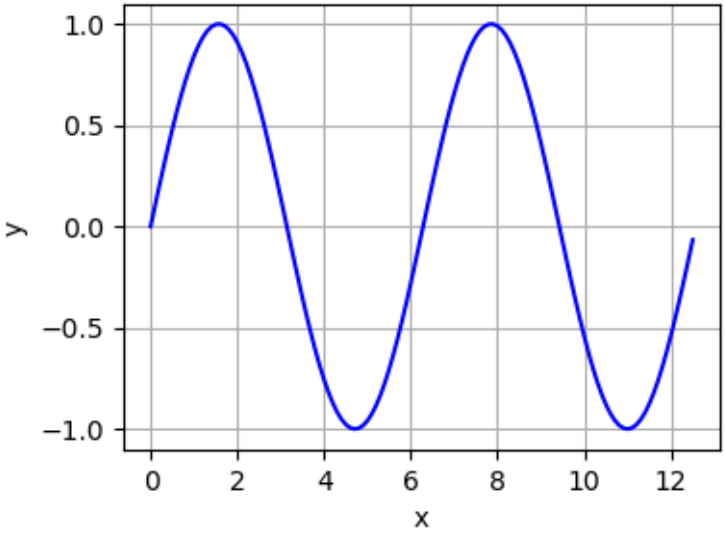
Using sin(x) to fit our data

What if we want to use $y = \beta_0 + \beta_1 \sin(x)$ as our nonlinear function that will fit our data? How do we make this work?

First we need to get a better feel for the sine function!

$$y = a \sin(bx) + c$$

Using sin(x) to fit our data



Using sin(x) to fit our data

Questions:

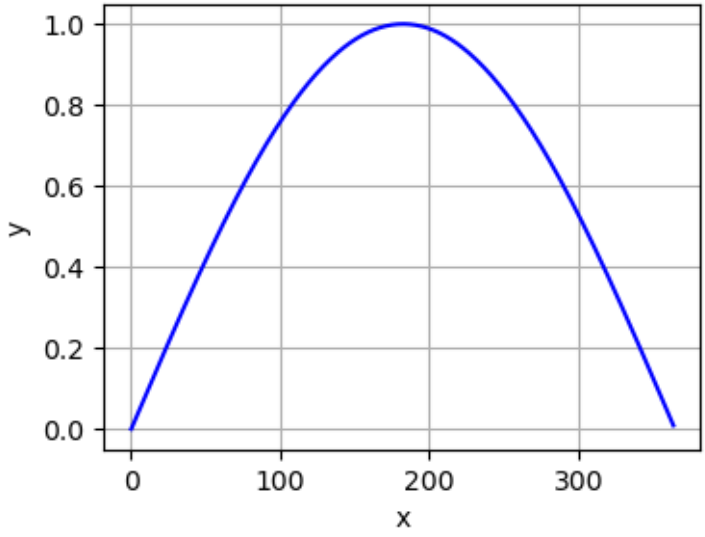
1. Take a minute and play around with the function plotted above what do a, b, and c do?
2. How often does this function repeat?

Using sin(x) to fit our data

How could we take a sine function and shift it to fit this data? We need to make sure we get just one bump!

```
x_sin = x
y_sin = np.sin((np.pi/365) *x)
```

Using sin(x) to fit our data



Using sin(x) to fit our data

Now we have to do some of the hard work (feature engineering) before we send our data to polyfit. In python we enter

```
np.sin(np.pi *x /365)
```

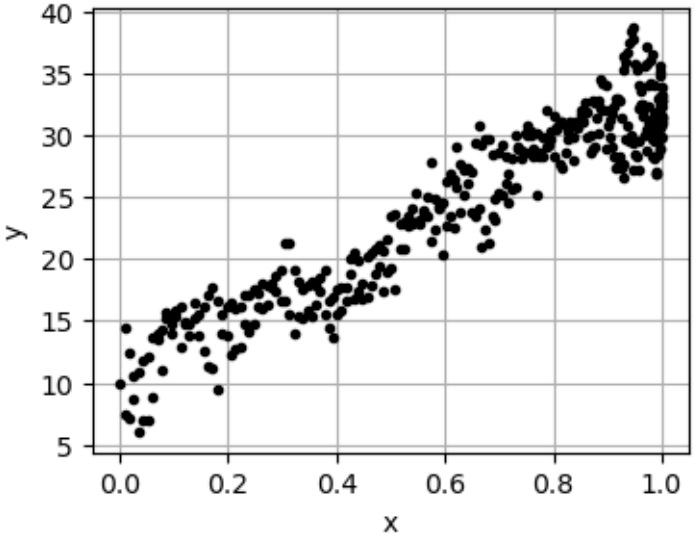
to convert our data from x to our newly engineered data.

We could check to see if there is a correlation between our chosen function of x and the y data.

```
np.corrcoef(xsin,y)[0,1]
```

```
0.942399024268974
```

Using sin(x) to fit our data



Using sin(x) to fit our data

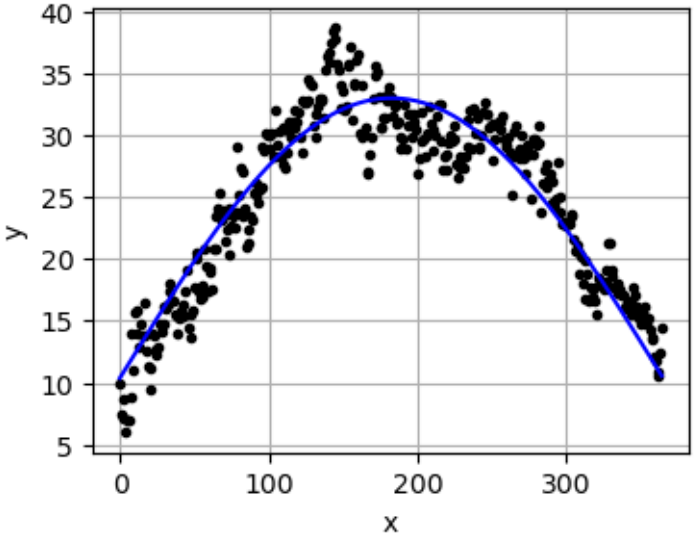
If we do a linear regression with the new feature data we should get a reasonable result.

$$y = \beta_0 + \beta_1 \sin\left(\frac{\pi x}{365}\right) = \beta_0 + \beta_1 x_1$$

betas:

```
array([22.6562821, 10.3681457])
```

Using sin(x) to fit our data



Using sin(x) to fit our data

R squared:

`0.8881159209431134`

MSE:

`6.125179888598971`

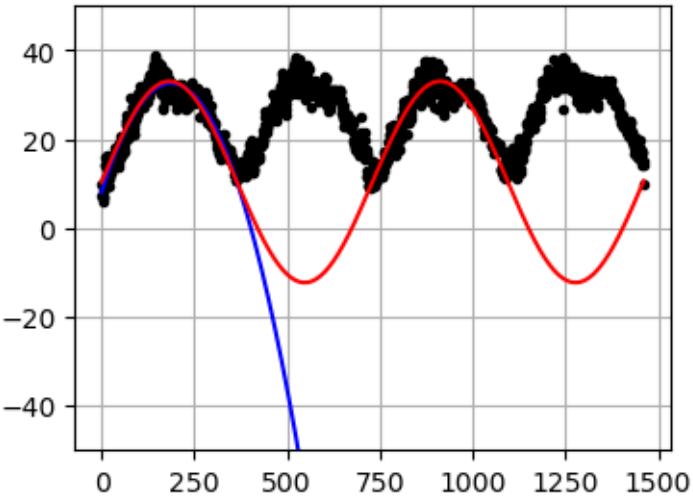
Using sin(x) to fit our data

Both worked so which should we use.

Well Occams razor would say to use the simplest model that fits the data. Which one is simpler?

Also, this is a bit of a contrived example because if we look at the data over more years we actually see that neither of our models work very well.

Expanding our data:



Expanding our data:

POLYFIT

R^2 for the polyfit

`-13.02177521007198`

MSE for the polyfit

`222328.92193525183`

SINE FUNCTION

R^2 for the sin function

`-2738.8227158184377`

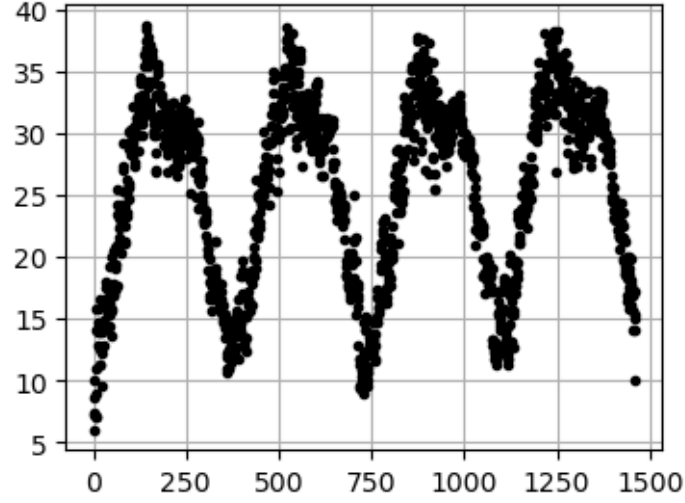
MSE for the sine function

`554.3635033352404`

Large negative R^2 values means that we did a HORRIBLE job of fitting the data!

You Try - Due next class

What function should we have tried here?



You Try - Due next class

1. What function do you propose to really fit the data? HINT - just apply another function to our sine function. There are multiple things that will work better than what we did above!
2. Using your choice of function calculate the Correlation Coefficient.
3. Create a scatter plot of your function applied to the data (should look kinda linear).
4. Do a linear regression with your function applied to the data.
5. Plot the regression vs the real data.
6. Calculate the R^2 value.