CS 181 HW 1

Lucas Pao

February 11, 2022

*Collaborated with Emily, Marissa, Jothi, Charu, Catherine, Mujin
*This assignment took 15 hours.

1 Optimizing a Kernel

1. We are trying to find the residual sum of squares, which is defined as

$$\mathcal{L}(W) = \sum_{n=1}^{N} (y_n - f(x_n))^2$$

For some x_n in our testing data, our predicted value $\hat{y_n} = f(x_n)$ is

$$f(x_n) = \sum_{m=1, m \neq n}^{N} K(x_m, x_n) y_m$$

The set of points we include into our formula for $f(x_n)$ is all of the points except the point (x_n, y_n) itself because we don't want to use a point's value to predict itself. Note that the kernel function is defined as

$$K(x_m, x_n) = \exp\left(\frac{-(x_m - x_n)^T (x_m - x_n)}{\tau}\right)$$

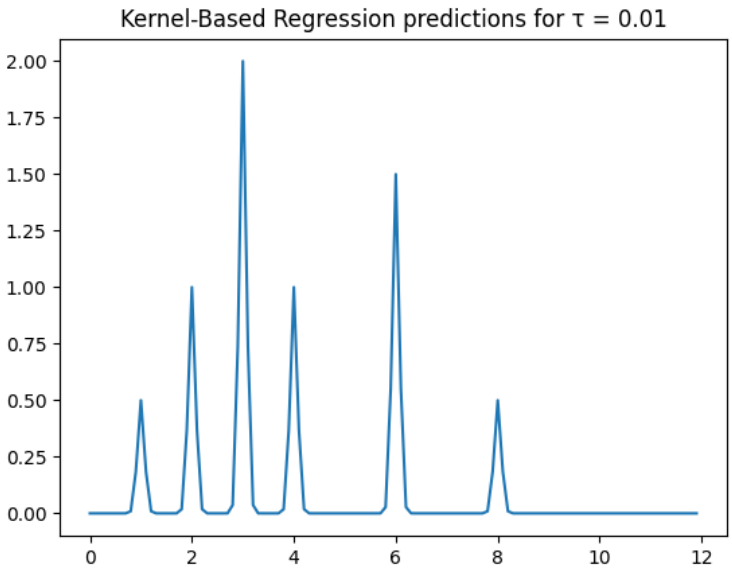
Combining the above, we get

$$\mathcal{L}(W) = \sum_{n=1}^{N} \left(y_n - \sum_{m=1, m \neq n}^{N} \exp\left(\frac{-(x_m - x_n)^T (x_m - x_n)}{\tau}\right) y_m \right)^2$$

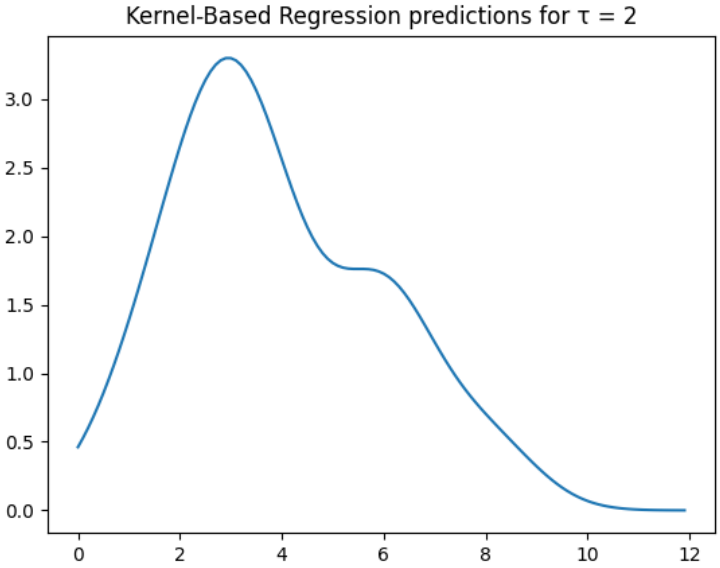
2. Now we find $\frac{\partial \mathcal{L}}{\partial \tau}$, the derivative of the loss function with respect to τ.

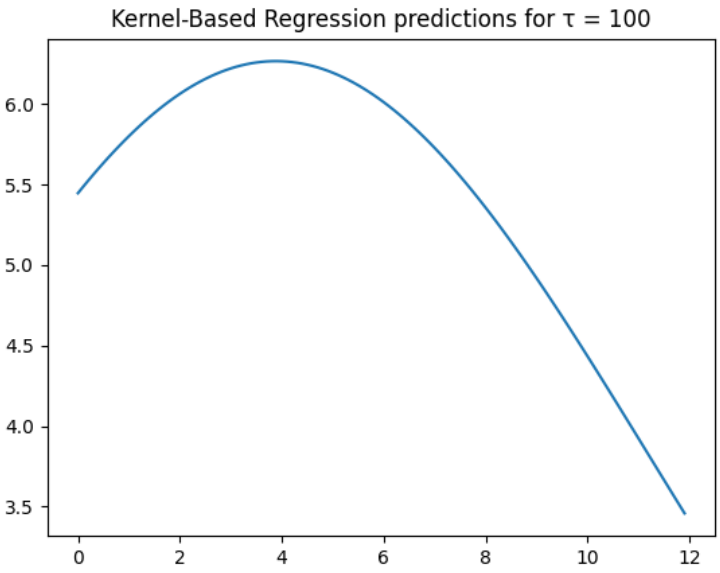
$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \tau} =& \frac{\partial}{\partial \tau} \sum_{n=1}^{N} \left(y_n - \sum_{m=1, m \neq n}^{N} \exp\left(\frac{-(x_m - x_n)^T (x_m - x_n)}{\tau} \right) y_m \right)^2 \\
=& \sum_{n=1}^{N} \frac{\partial}{\partial \tau} \left(y_n - \sum_{m=1, m \neq n}^{N} \exp\left(\frac{-(x_m - x_n)^T (x_m - x_n)}{\tau} \right) y_m \right)^2 \\
=& \sum_{n=1}^{N} 2 \left(y_n - \sum_{m=1, m \neq n}^{N} \exp\left(\frac{-(x_m - x_n)^T (x_m - x_n)}{\tau} \right) y_m \right) \times \\
& \frac{\partial}{\partial \tau} \left(y_n - \sum_{m=1, m \neq n}^{N} \exp\left(\frac{-(x_m - x_n)^T (x_m - x_n)}{\tau} \right) y_m \right) \\
=& \sum_{n=1}^{N} 2 \left(y_n - \sum_{m=1, m \neq n}^{N} \exp\left(\frac{-(x_m - x_n)^T (x_m - x_n)}{\tau} \right) y_m \right) \times \\
& \left(- \sum_{m=1, m \neq n}^{N} \left(\exp\left(\frac{-(x_m - x_n)^T (x_m - x_n)}{\tau} \right) y_m \times \frac{\partial}{\partial \tau} \left(\frac{-(x_m - x_n)^T (x_m - x_n)}{\tau} \right) \right) \right) \\
=& \sum_{n=1}^{N} 2 \left(y_n - \sum_{m=1, m \neq n}^{N} \exp\left(\frac{-(x_m - x_n)^T (x_m - x_n)}{\tau} \right) y_m \right) \\
& \left(- \sum_{m=1, m \neq n}^{N} \exp\left(\frac{-(x_m - x_n)^T (x_m - x_n)}{\tau} \right) (y_m) \left(\frac{(x_m - x_n)^T (x_m - x_n)}{\tau^2} \right) \right)
\end{aligned}$$

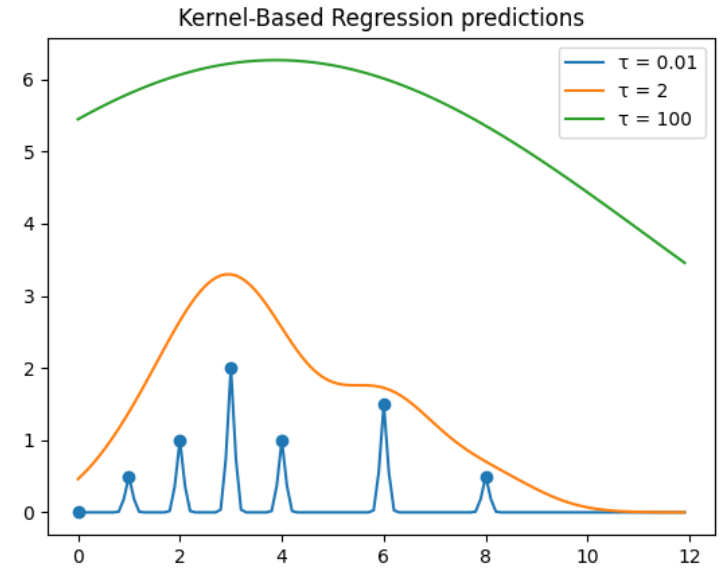
3. `Loss for tau = 0.01: 8.75`
   `Loss for tau = 2: 3.305016494565789`
   `Loss for tau = 100: 120.35919342230957`
   As we can see, a lengthscale of $\boxed{\tau = 2}$ yields the lowest loss.


Kernel-Based Regression predictions for τ = 0.01
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
2
4
6
8
10
12

4.


Kernel-Based Regression predictions for τ = 2
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
2
4
6
8
10
12


Kernel-Based Regression predictions for τ = 100
6.0
5.5
5.0
4.5
4.0
3.5
0
2
4
6
8
10
12


Kernel-Based Regression predictions
τ = 0.01
τ = 2
τ = 100
6
5
4
3
2
1
0
0
2
4
6
8
10
12

Explanation for 1.4: Briefly describe what happens in each of the three cases. Is what you see consistent with the which lengthscale appeared to be numerically best above? Describe why or why not.

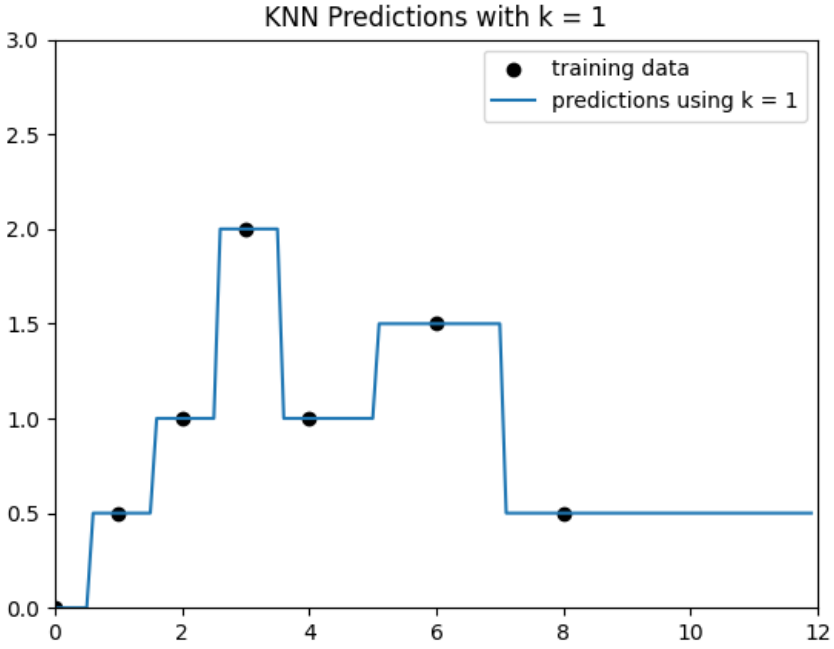
The lengthscale is a constant that determines to what degree the distance between points should contribute to predicting the value of a new data point. For a small lengthscale like $\tau = 0.01$, the kernel distance between two points x_1, x_2 is defined as $K(x_1, x_2) = e^{-100(x_1 - x_2)^2}$, so points far away from the data point contribute virtually nothing, whereas for a large lengthscale like $\tau = 100$, the kernel distance between two points x_1, x_2 is defined as $K(x_1, x_2) = e^{-0.01(x_1 - x_2)^2}$, so points far away from the data point still contribute some weighting to the predicted output.

This mathematical interpretation is reflected in the graphs above. For the small lengthscale, the only values that seems to have weighting in predicting the output are the data points directly near the intended point. In fact, points that are not close to any initial data point are predicted an output of 0, since all the other data points contribute virtually nothing to the kernel weights. The model resulting from this lengthscale suffers from overfitting, as it only captures data points that already existed.

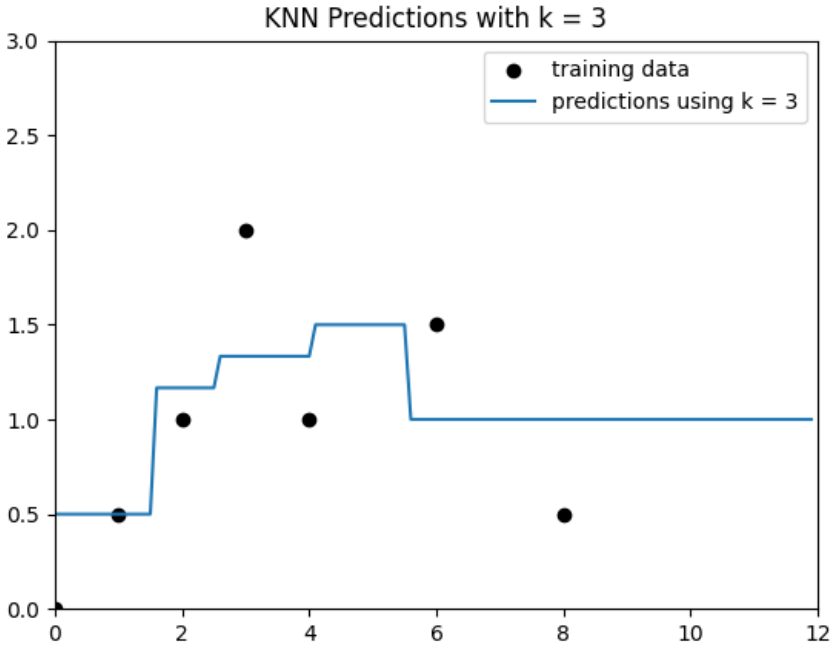
On the other hand, for the large lengthscale, almost all of the values seem to have weighting in predicting the output, since each point contributes some tangible amount to the kernel weights in predicting an output. The model resulting from this lengthscale suffers from underfitting, as the shape of the curve is influenced too much by points that aren't even close to the intended data point and should not be indicative of predicting certain values.

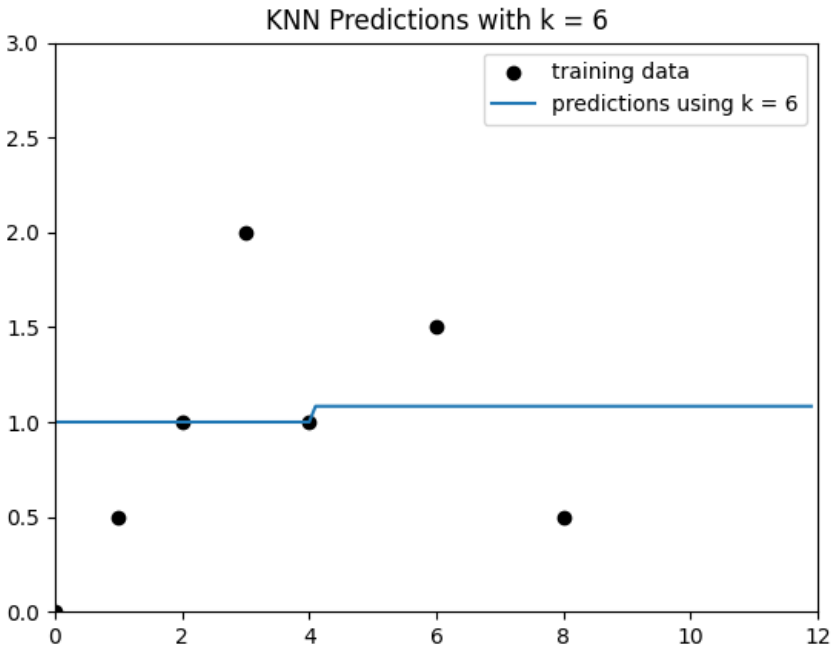
Therefore, a lengthscale in between these two extremes provides the perfect balance of being able to use points that are relatively close to predict an intended data point. Data points that are not originally found in the training dataset can be predicted by those found nearby, and in general, data points are only predicted by points found nearby and not those far away. This type of lengthscale addresses the problems faced by both extremes.

2 Kernels and kNN



1.





2. The behavior of these functions is similar to that from Problem 1 in that it generally follows the trend of the training data. But it differs in that it is inherently discrete (stepwise) and not continuous. The graphs in Problem 1 are continuous because the predicted outputs are directly based on the output of the kernel function, which is continuous. On the other hand, the graphs in the kNN approach are dependent on the specific k points chosen to represent the predicted output. So two neighboring points could be closest to different sets of k points and would thus have vastly different predicted outputs. This property is inherent in the definition of discontinuous points.

   Both methods are dependent on existing data points, so they are both able to interpolate predictions well but are unable to extrapolate predictions. In particular, trying to extrapolate values outside the range of the data for kNN is impossible since the kNN algorithm involves choosing the k actual closest points to predict the new data and there are no points to choose for data points outside of the range. Trying to extrapolate values outside the range of the data for kernel-based regression is also not very possible since the kNN algorithm involves finding the distance between the actual data points to predict the new data and there are no points to choose for data points outside of the range.

In general, kernel-based regression and kNN produce similar shaped graphs that generally follow the trend of the data. But there exist certain datasets for which these two methods produce different classifiers regardless of the values of k and τ. This is because the two methods calculate the predictions in different ways, even on the same set of points. The kernel-based approach adds the values of the kernelized distances between points whereas the kNN approach adds the data values of the k points chosen. The kernelized distance between two points and the data value of one of the k points chosen are fundamentally different values.

As a counterexample, consider the predicted output of a data point extremely far away from all the other data points. The kernelized distance between this point and all of the other points converges to 0, so the predicted output of this function converges to 0. Using the kNN approach, assuming WLOG that any k points chosen to represent the predicted output of this point are non-zero, the predicted output of this data point is the average of these values and would not converge to 0.

3. τ is a constant used in the kernel function as a distance metric to determine the relative distance between points. In the kernel-based regression in Part 1, the actual value of the output of the kernel function is incorporated into the function that predicts the value. But in the kNN approach, the output of the kernel function is just used to determine the k closest points, and the values of the data points, not the values of the kernel function, are used to predict new values. Since τ is a constant, the relative closeness of the k closest points to a data point does not change when τ changes, so varying τ wouldn't provide any different or useful results.

3 Deriving Linear Regression

1. The expected squared loss is defined as $\mathcal{L} = E_{x,y}[(y - \hat{y})^2]$. Assume a linear prediction model of $\hat{y} = wx$. By linearity of expectation, the expected squared loss $\mathcal{L}(w)$ is:

$$\begin{aligned}
\mathcal{L}(w) &= E_{x,y}[(y - \hat{y}^2)] \\
&= E_{x,y}[(y - wx)^2] \\
&= E_{x,y}[y^2 - 2wxy + w^2x^2] \\
&= E_{x,y}[y^2] - 2wE_{x,y}[xy] + w^2E_{x,y}[x^2] \\
&= E_y[y^2] - 2wE_{x,y}[xy] + w^2E_x[x^2]
\end{aligned}$$

In the last step, we remove variables in an expectation if it does not depend on them. Now we find the optimal w that minimizes the loss by ensuring the loss function satisfies the first- and second-order conditions.

First Order Conditions:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w} &= 0 - 2E_{x,y}[xy] + 2wE_x[x^2] \stackrel{set}{=} 0 \\
2E_{x,y}[xy] &= 2wE_x[x^2] \\
\boxed{w^* = \frac{E_{x,y}[xy]}{E_x[x^2]}}
\end{aligned}$$

Second Order Conditions:

$$\frac{\partial^2 \mathcal{L}}{\partial w^2} = 2E_x[x^2] = 2\sum_{x \in \text{data}} p(x) \cdot x^2 \geq 0$$

We know this value is non-negative because probabilities and squared values are non-negative. This confirms that the optimal w^* we found is a minimum and not a maximum.

2. An unbiased, consistent way to calculate the expectation of these values is to take the standard mean of these values over all the data points.

$$E_{x,y}[xy] = \frac{1}{N}\sum_{n=1}^{N} x_n y_n$$

$$E_x[x^2] = \frac{1}{N}\sum_{n=1}^{N} x_n^2$$

3. Substituting in the equations we got in 2. into those from 1., we get:

$$w^* = \frac{E_{x,y}[xy]}{E_x[x^2]} = \frac{\frac{1}{N}\sum_{n=1}^{N} x_n y_n}{\frac{1}{N}\sum_{n=1}^{N} x_n^2} = \frac{\sum_{n=1}^{N} x_n y_n}{\sum_{n=1}^{N} x_n^2}$$

Using matrix calculus, the optimal value of w^* derived in the textbook is $w^* = (X^\top X)^{-1} X^\top Y$, where X and Y are matrices of size $N \times D$, where N is the number of data points and D is the number of dimensions. But in this case, we assume the data is one dimensional, so

$$X^T X = X \cdot X = \sum_{n=1}^{N} x_n \cdot x_n = \sum_{n=1}^{N} x_n^2$$

$$X^T Y = X \cdot Y = \sum_{n=1}^{N} x_n \cdot y_n$$

So we can rewrite this matrix product in terms of its elements:

$$\begin{aligned} w^* &= (X^\top X)^{-1} X^\top Y \\ &= \left(\sum_{n=1}^{N} x_n^2 \right)^{-1} \left(\sum_{n=1}^{N} x_n \cdot y_n \right) \\ &= \frac{\sum_{n=1}^{N} x_n y_n}{\sum_{n=1}^{N} x_n^2} \end{aligned}$$

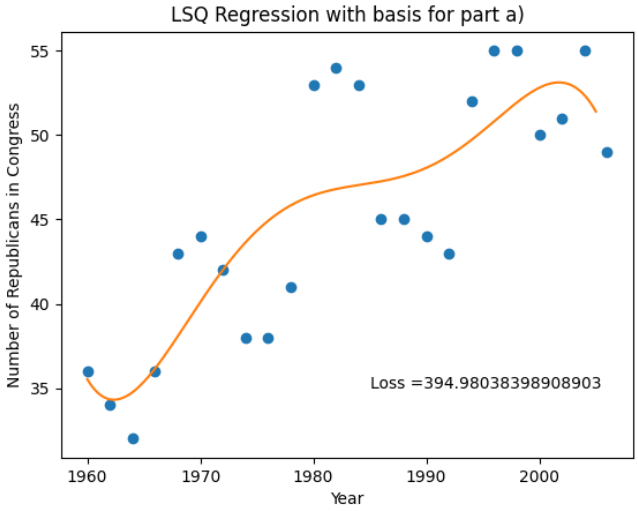
which is the same expression as we found before.

4. No, we did not assume that x and y are jointly Gaussian during any step of this derivation. The only assumptions we used were linearity of expectation and the standard mean formula, which hold for any probabilistic distribution of data.

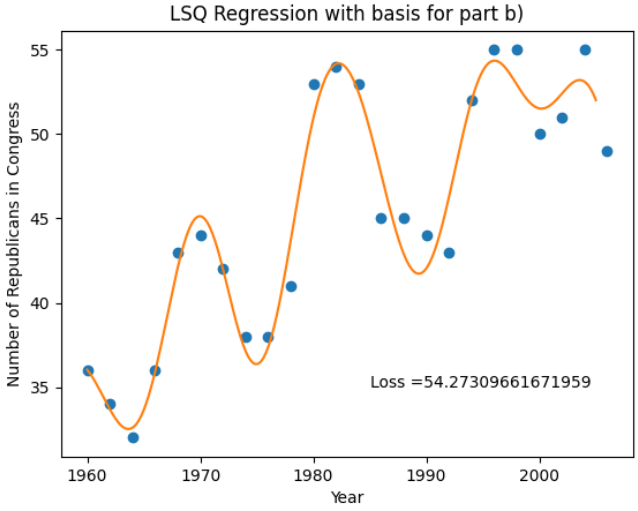
4 Modeling Changes in Republicans and Sunspots

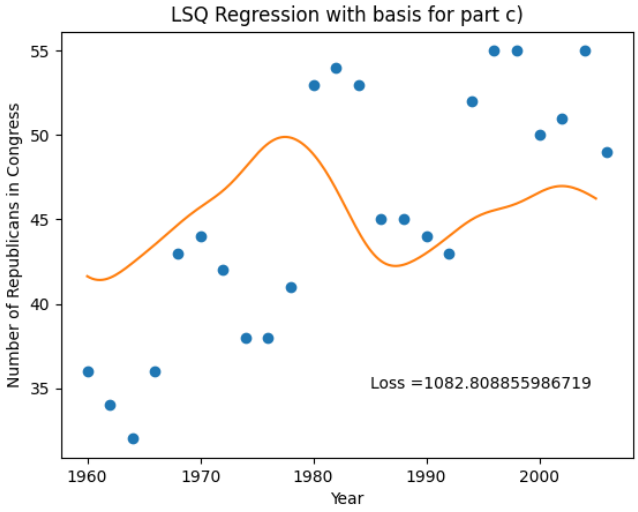
We define "Loss" as the residual sum of squares error.

1. (a)

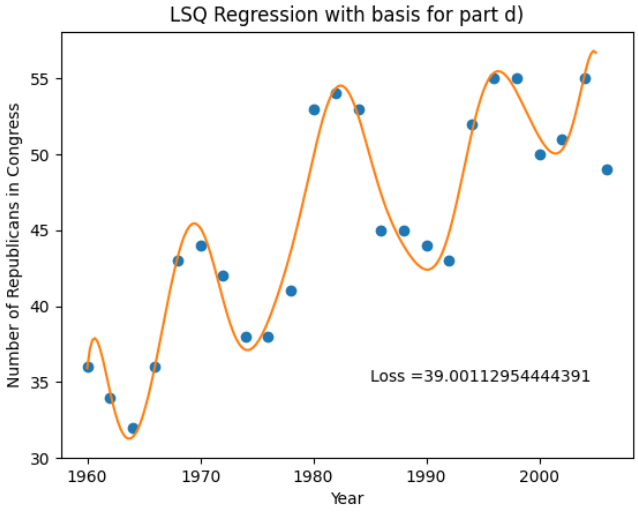


(b)

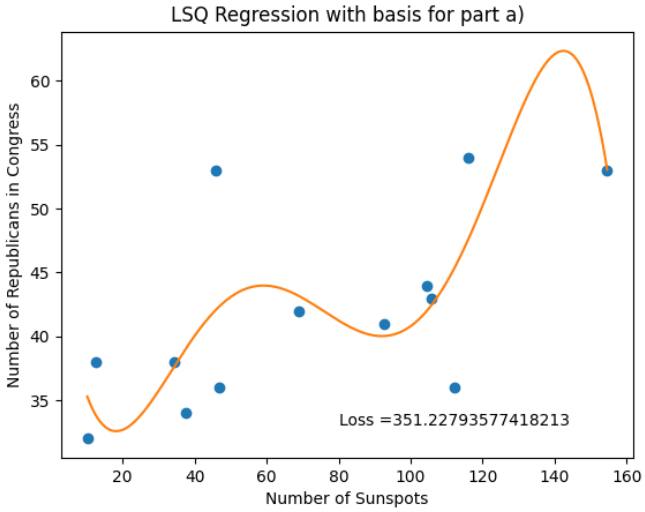



LSQ Regression with basis for part c)
Number of Republicans in Congress
Year
Loss =1082.808855986719
55
50
45
40
35
1960
1970
1980
1990
2000

(c)

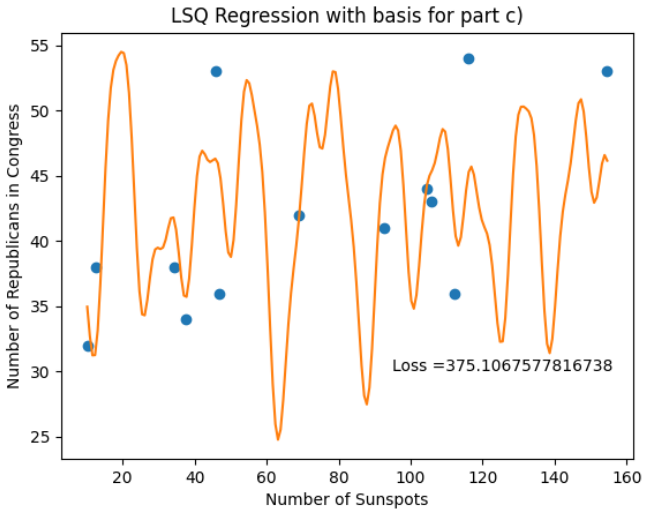

LSQ Regression with basis for part d)
Number of Republicans in Congress
Year
Loss =39.00112954444391
55
50
45
40
35
30
1960
1970
1980
1990
2000

(d)

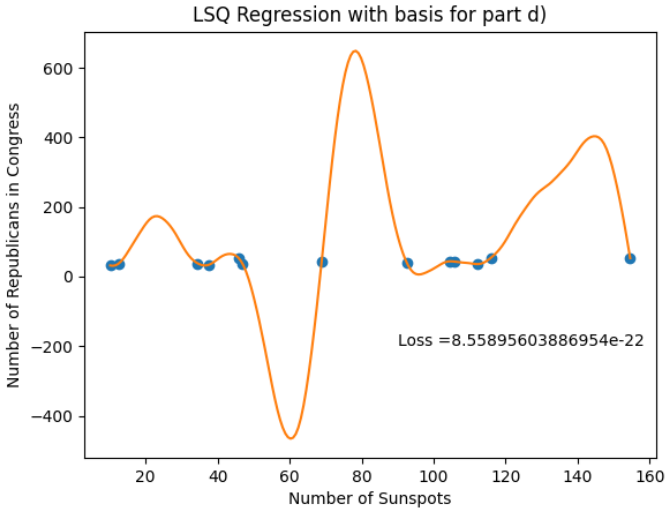


2. (a)

   (b) Not using this basis for this part.



   (c)



(d)

Explanations for 4.2:

I think that the model resulting from using the basis function in part (a) provides the "best" fit. Even though the model using the basis function in part (d) fits all of the training data and has the lowest residual sum of squares error, and the model using the basis function in part (c) still has a relatively low residual sum of squares error, I think that they suffer from overfitting which explains why the loss almost seems too optimal. Visually, these models do not seem to provide a general shape of the data that would make it possible to predict new values. So even though the model in part (a) is not necessarily the best in minimizing the residual sum of squares loss or passing through the most data points in the training set, its structure allows for flexibility in predicting new values, which is the main purpose of regression anyways.

But even choosing model (a) as the best fit model, it still has a high residual sum of squares loss and does not go through many data points, meaning that the original data does not seem to implicate any trend between the two variables. So no, I do not believe that the number of sunspots controls the number of Republicans in the Senate.