Assignment #1

CS181-S22

Due: 7:59pm ET, February 4, 2022

Homework 1: Regression

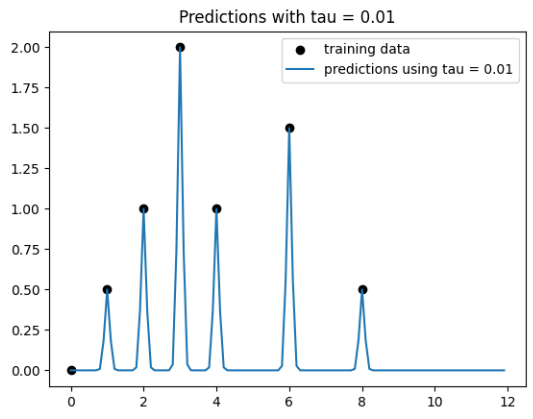
1.1

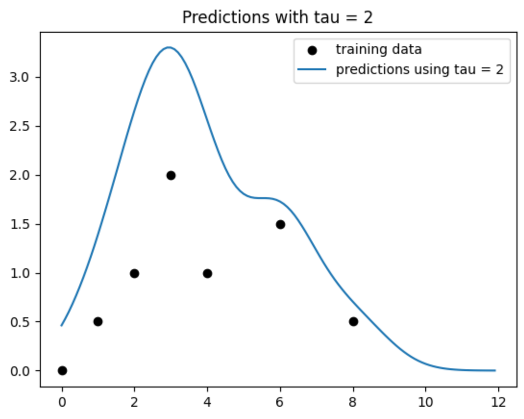
$$\begin{aligned}\mathcal{L}(W) &= \sum_{n=1}^{N}(y_n - \hat{y_n})^2\\ &= \sum_{n=1}^{N}\left(y_n - \sum_{i=1, i!=n}^{N} k(x_i, x_n)y_i\right)^2\\ &= \sum_{n=1}^{N}\left(y_n - \sum_{i=1, i\neq n}^{N} \exp(\frac{-(x_n - x_i)^T(x_n - x_i)}{\tau})y_i\right)^2\end{aligned}$$

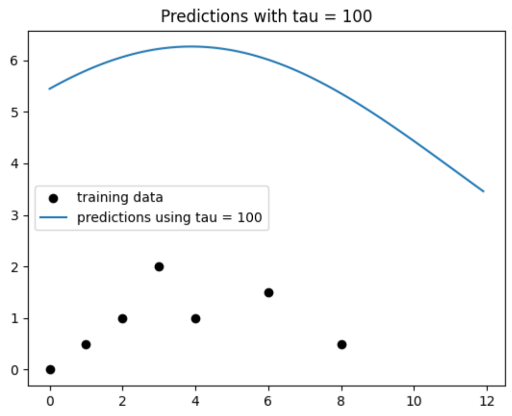
1.2 Let $(x_n - x_i)^T(x_n - x_i) = \alpha$.

$$\frac{\partial \mathcal{L}(W)}{\partial \tau} = \sum_{n=1}^{N}\left[2\left(y_n - \sum_{i=1,i\neq n}^{N}\left(\exp(\frac{-\alpha}{\tau})y_i\right)\right)\left(-\sum_{i=1,i\neq n}^{N}\left(\exp(\frac{-\alpha}{\tau})y_i(\frac{\alpha}{\tau^2})\right)\right)\right]$$

1.3 The lengthscales $\tau = .01$, $\tau = 2$, and $\tau = 100$ gave losses of 8.75, 3.305016494565789, and 120.35919342230957, respectively. The lengthscale $\tau = 2$ does the best in minimizing residual losses.
1.4 The plots are shown below.



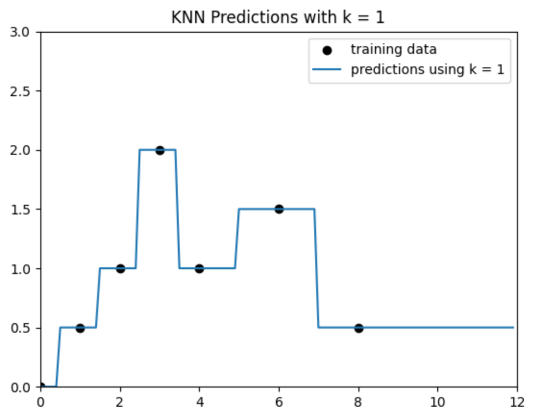


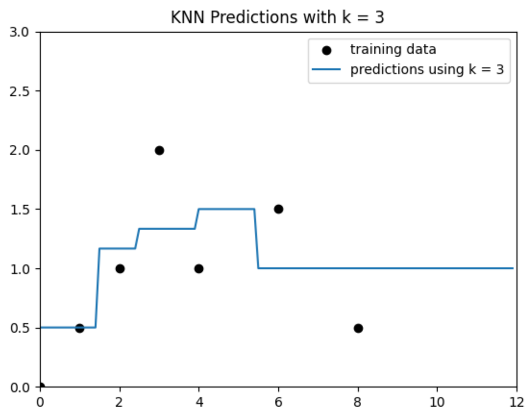


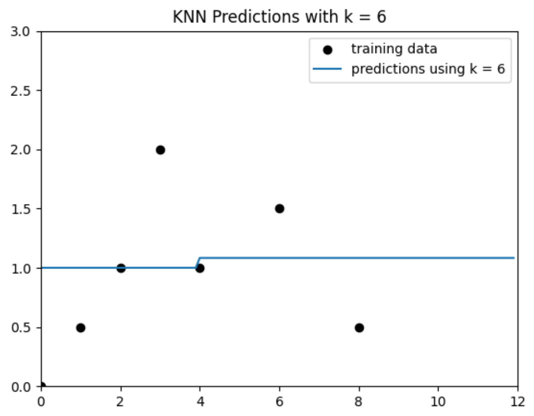
When $\tau = 0.01$, the predictions primarily stay around 0 but spikes up very rapidly when approaching the training data x-values. When $\tau = 2$, the predictions follow a relatively smooth graph that seems to mirror the data pretty well. When $\tau = 100$, the predictions are primarily between 4 and 6 (when all the training data is approximately between 0 and 2), and the prediction line follows a very smooth arc.

According to the residual sum of squares, I would expect for the lengthscale $\tau = 0.01$ to be the best fit, but the graph informed me that this is not the best fit. This graph fits well only at x-values very close to the training data, and does not seem to fit well at other points (only yielding a prediction of 0).

2.1 The plots are shown below.







2.2 The kNN mechanism results in graphs that have intervals of constant predictions, and the intervals jump when the neighbors change. When the number of neighbors k is smaller, the graph jumps more. For higher values of k, the graph becomes smoother. When τ is low and k is low, the prediction plots give high weight to nearby training data points, so they interpolate similarly. When τ is high and k is high, the predictions give approximately equal weight to all training data points, which results in a much smoother graph, indicating they interpolate similarly. kNN and kernel-based regression do not extrapolate similarly. The predictions for kernel-based regression for x that are far beyond the training set approach 0 (this trend is more noticeable for smaller τ), because as x gets farther away from the data points, the kernel value decreases and approaches 0. For kNN, as x gets beyond the training set, the predictions approach the y values of the endpoints in the training data (again, most noticeable for small k).

2.3 In the kernel-based approach, we use τ to affect the weighting of training point values. However, in the kNN approach, we give equal weighting to the k neighbors that were chosen, so we do not need to vary τ.

3.1 The expected squared loss is given by $E_{x,y}[(y-\hat{y})^2] = E_{x,y}[(y-wx)^2]$. We will first simplify this expression, as we can expand and then use linearity of expectation.

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

To find the optimal w, we want to set the derivative of the expected squared loss to equal 0. Therefore, we have:

$$\begin{aligned} &0 = \frac{\partial}{\partial w}\left(E_y[y^2] - 2wE_{x,y}[xy] + w^2E_x[x^2]\right) \\ &0 = 0 - 2E_{x,y}[xy] + 2wE_x[x^2] \\ &\boxed{w = \frac{E_{x,y}[xy]}{E_x[x^2]}} \end{aligned}$$

3.2 The formulas for expectation yield:

$$E_{x,y}[xy] = \sum_x \sum_y xy * p(x,y)$$

$$E_x[x^2] = \sum_x x^2 p(x)$$

We have N data points (x_n, y_n), each of which occurs with probability $\frac{1}{N}$. Therefore, our formulas are

$$E_{x,y}[xy] = \frac{1}{N}\sum_x \sum_y xy$$

$$E_x[x^2] = \frac{1}{N}\sum_x x^2$$

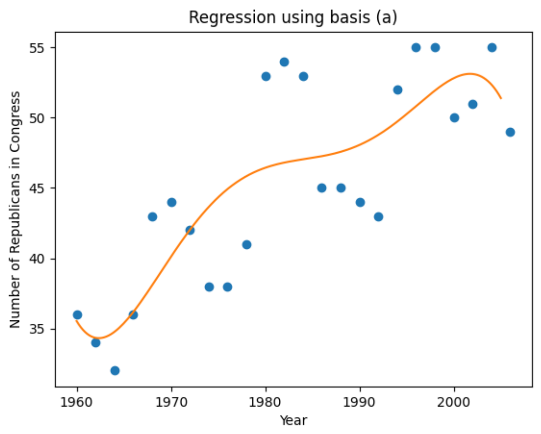
3.3 The resulting expression for the optimal w^* is

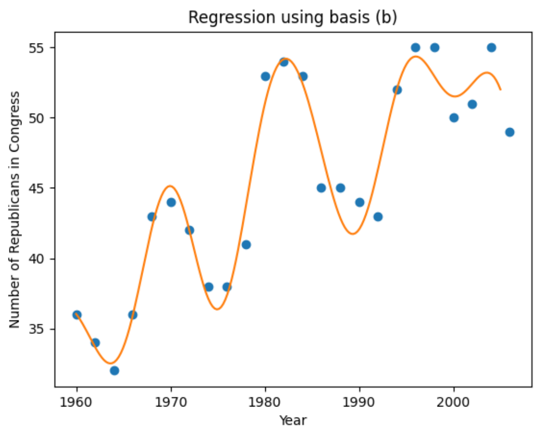
$$\begin{aligned} w^* &= \frac{\frac{1}{N}\sum_x \sum_y xy}{\frac{1}{N}\sum_x x^2} \\ &= \boxed{\frac{\sum_x \sum_y xy}{\sum_x x^2}} \end{aligned}$$

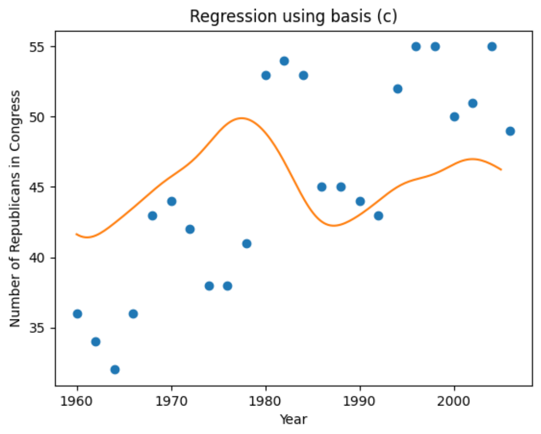
The optimal w^* derived in Section 2.6 is $w^* = (X^TX)^{-1}X^Ty$. This is very similar to our expression for the optimal w^*. $(X^TX)^{-1}$ corresponds to the $\sum_x x^2$ in the denominator of our derived expression, while the X^Ty corresponds to the $\sum_x \sum_y xy$ in the numerator.

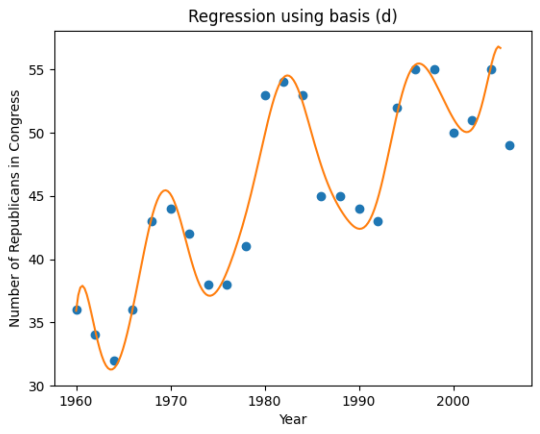
3.4 None of the derivations above relied on the assumption that x and y are jointly Gaussian. In the calculations, we kept the moments general and did not assume Gaussian distributions, so we never had to assume joint Gaussian distributions.

4.1. The plots of data and regression lines for the aforementioned basis functions are below.



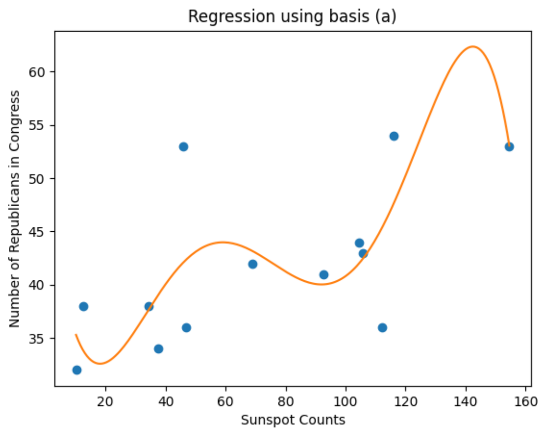


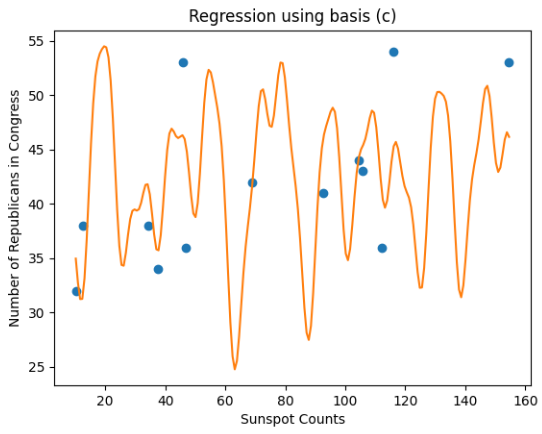


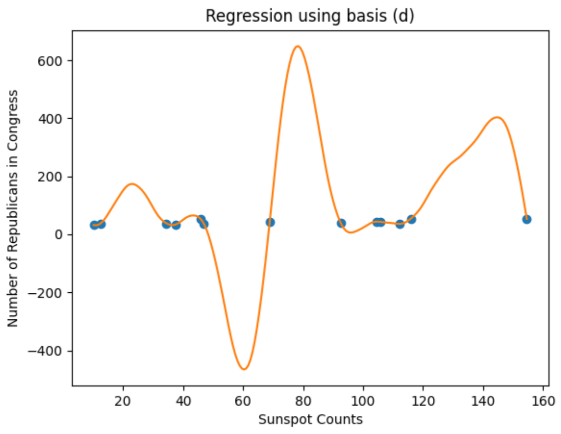


The residual sum of squares error is listed for each basis below:
Basis (a): 394.9803839890865
Basis (b): 54.2730966167196
Basis (c): 1082.8088559867185
Basis (d): 39.001226915430635

4.2. The plots of the data and regression lines for Number of Sunspots v. Number of Republicans in the Senate are shown below:







The residual sum of squares error is listed for each basis below:
Basis (a): 351.22793577417474
Basis (c): 375.106757781674
Basis (d): 8.622599569654343e-22

If the objective was minimize the residual sum of squares, the "best" fit was basis (d), as the residual sum of squares was the smallest for that. The quality of this fit is not very high, as the graph appears to be overfitted, so I do not believe that the number of sunspots controls the number of Republicans in the Senate.

Name

Kathryn Zhou

Collaborators and Resources

Whom did you work with, and did you use any resources beyond cs181-textbook and your notes?
I used documentation about numpy and pandas. I discussed some of the problems and approaches with Derek Zheng.

Calibration

Approximately how long did this homework take you to complete (in hours)?
This homework took me about 10 hours to complete.