Assignment #1

Due: 11:59pm February 8, 2019

Homework 1: Linear Regression

Introduction

This homework is on different forms of linear regression and focuses on loss functions, optimizers, and regularization. Linear regression will be one of the few models that we see that has an analytical solution. These problems focus on deriving these solutions and exploring their properties.

If you find that you are having trouble with the first couple problems, we recommend going over the fundamentals of linear algebra and matrix calculus. We also encourage you to first read the Bishop textbook, particularly: Section 2.3 (Properties of Gaussian Distributions), Section 3.1 (Linear Basis Regression), and Section 3.3 (Bayesian Linear Regression). Note that our notation is slightly different but the underlying mathematics remains the same.

Please type your solutions after the corresponding problems using this LaTeX template, and start each problem on a new page. You will submit your solution PDF, your tex file, and your code to Canvas.

Problem 1 (Priors as Regularization,15pts)

In this problem we consider a model of Bayesian linear regression. Define the prior on the parameters as,

$$p(\theta) = \mathcal{N}(\theta \,|\, \mathbf{0}, \sigma_\theta^2 \mathbf{I}),$$

where σ_θ^2 is as scalar variance hyperparameter that controls the variance of the Gaussian prior. Define the likelihood as,

$$p(\mathbf{y} \,|\, \mathbf{X}, \theta) = \prod_{i=1}^{n} \mathcal{N}(y_i \,|\, \theta^\mathsf{T} \mathbf{x}_i, \sigma_n^2),$$

where σ_n^2 is another fixed scalar defining the variance.

1. Using the fact that the posterior is the product of the prior and the likelihood (up to a normalization constant), i.e.,

   $$\arg\max_\theta \ln p(\theta \,|\, \mathbf{y}, \mathbf{X}) = \arg\max_\theta \ln p(\theta) + \ln p(\mathbf{y} \,|\, \mathbf{X}, \theta).$$

   Show that maximizing the log posterior is equivalent to minimizing a regularized loss function given by $\mathcal{L}(\theta) + \lambda \mathcal{R}(\theta)$, where

   $$\mathcal{L}(\theta) = \frac{1}{2} \sum_{i=1}^{n} (y_i - \theta^\mathsf{T} \mathbf{x}_i)^2$$
   $$\mathcal{R}(\theta) = \frac{1}{2} \theta^\mathsf{T} \theta$$

   Do this by writing $\ln p(\theta \,|\, \mathbf{y}, \mathbf{X})$ as a function of $\mathcal{L}(\theta)$ and $\mathcal{R}(\theta)$, dropping constant terms if necessary. Conclude that maximizing this posterior is equivalent to minimizing the regularized error term given by $\mathcal{L}(\theta) + \lambda \mathcal{R}(\theta)$ for a λ expressed in terms of the problem's constants.

2. Notice that the form of the posterior is the same as the form of the ridge regression loss

   $$\mathcal{L}(\theta) = (\mathbf{y} - \mathbf{X}\theta)^\top (\mathbf{y} - \mathbf{X}\theta) + \lambda \theta^\top \theta.$$

   Compute the gradient of the loss above with respect to θ. Simplify as much as you can for full credit. Make sure to give your answer in vector form.

3. Suppose that $\lambda > 0$. Knowing that $\mathcal{L}$ is a convex function of its arguments, conclude that a global optimizer of $\mathcal{L}(\theta)$ is

   $$\theta = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y} \tag{1}$$

   For this part of the problem, assume that the data has been centered, that is, pre-processed such that $\frac{1}{n} \sum_{i=1}^{n} x_{ij} = 0$.

4. What might happen if the number of weights in θ is greater than the number of data points N? How does the regularization help ensure that the inverse in the solution above can be computed?

Solution

(a) The prior function

$$p(\theta) = \mathcal{N}(\theta \,|\, \mathbf{0}, \sigma_\theta^2 \mathbf{I}),$$

looks like

$$p(\theta) = \frac{1}{(2\pi)^{\frac{D}{2}}} \frac{1}{(\sigma_\theta^2 \mathbf{I})^{\frac{1}{2}}} \exp(-\frac{1}{2}\theta^\mathsf{T}(\sigma_\theta^2 \mathbf{I})^{-1}\theta)$$

Take the natural log

$$\begin{aligned}\ln(p(\theta)) = \ln(\frac{1}{(2\pi)^{\frac{D}{2}}} \frac{1}{(\sigma_\theta^2 \mathbf{I})^{\frac{1}{2}}}) + \ln(\exp(-\frac{1}{2}\theta^\mathsf{T}(\sigma_\theta^2 \mathbf{I})^{-1}\theta))) \\ = constant + (-\frac{1}{2}\theta^\mathsf{T}(\sigma_\theta^2 \mathbf{I})^{-1}\theta) \\ = constant - \sigma_\theta^{-2}(\frac{1}{2}\theta^\mathsf{T}\theta)\end{aligned}$$

Do the same thing with the likelihood function

$$p(\mathbf{y} \,|\, \mathbf{X}, \theta) = \prod_{i=1}^{n} \mathcal{N}(y_i \,|\, \theta^\mathsf{T}\mathbf{x}_i, \sigma_n^2)$$

Take the natural log, the product becomes a sum

$$p(\mathbf{y} \,|\, \mathbf{X}, \theta) = \sum_{i=1}^{n} \ln(\frac{1}{(2\pi\sigma_n^2)^{1/2}} * \exp(\frac{-1}{(2\pi\sigma_n^2)^{\frac{1}{2}}} * (y_i - \theta^\mathsf{T}\mathbf{x}_i)^2)$$

$$constant + \frac{-1}{(2\pi\sigma_n^2)^{\frac{1}{2}}} * \sum_{i=1}^{n}(y_i - \theta^\mathsf{T}\mathbf{x}_i)^2$$

Putting the two terms together gives you

$$constant - \sigma_\theta^2(\frac{1}{2}\theta^\mathsf{T}\theta) - \frac{1}{(2\pi\sigma_n^2)^{\frac{1}{2}}} * \sum_{i=1}^{n}(y_i - \theta^\mathsf{T}\mathbf{x}_i)^2$$

Upon inspection it's clear that this form is very similar to the form of the regularized loss function, with negative sign and coefficients. These coefficients are constant (sigma defined to be scalar). Therefore, maximizing the log posterior is equivalent to minimizing the regularized loss function.

(b) Deriving L(θ) with respect to dθ gives:

$$\begin{aligned}\frac{\partial L(\theta)}{\partial \theta} = \frac{\partial}{\partial \theta}[(\mathbf{y} - \mathbf{X}\theta)^\top(\mathbf{y} - \mathbf{X}\theta)] + \frac{\partial}{\partial \theta}[\lambda\theta^\top\theta]) \\ \frac{\partial}{\partial \theta}[(\mathbf{y} - \mathbf{X}\theta)^\top(\mathbf{y} - \mathbf{X}\theta)] = -2\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\theta) \\ \frac{\partial}{\partial \theta}[\lambda\theta^\top\theta]) = 2\lambda\theta\mathbf{I}\end{aligned}$$

So our resulting gradient of the loss with respect to θ is

$$-2\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\theta) + 2(\lambda * \mathbf{I})\theta$$

(c) To find a global optimizer of $\mathrm{L}(\theta)$, we can set the gradient calculated in the previous part to 0 and calculate the value of its corresponding θ. We have:

$$\begin{aligned} -2\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\theta) + 2(\lambda * \mathbf{I})\theta &= 0 \\ \lambda\mathbf{I}\theta + \mathbf{X}^\top\mathbf{X}\theta &= \mathbf{X}^\top\mathbf{y} \\ (\lambda\mathbf{I} + \mathbf{X}^\top\mathbf{X})\theta &= \mathbf{X}^\top\mathbf{y} \\ \theta &= (\lambda\mathbf{I} + \mathbf{X}^\top\mathbf{X})^{-1} * (\mathbf{X}^\top\mathbf{y}) \end{aligned}$$

(d) If the number of weights in θ is greater than the number of data points N, the resulting $\mathbf{X}^\top\mathbf{X}$ could be non-invertible (not enough rank) and thus the solution above cannot be computed. Adding a lambda identity matrix (the regularization process) ensures that the resulting $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}$ determinant is non-zero and thus ensures that there is a unique solution to the problem.

Problem 2 (Optimizing a Kernel, 15pts)

Kernel-based regression techniques are similar to nearest-neighbor regressors: rather than fit a parametric model, they predict values for new data points by interpolating values from existing points in the training set. In this problem, we will consider a kernel-based regressor of the form:

$$f(x^*) = \frac{\sum_n K(x_n, x^*) y_n}{\sum_n K(x_n, x^*)}$$

where (x_n, y_n) are the training data points, and $K(x, x')$ is a kernel function that defines the similarity between two inputs x and x'. A popular choice of kernel is a function that decays with the distance between the two points, such as

$$K(x, x') = \exp(-||x - x'||_2^2) = \exp(-(x - x')(x - x')^T)$$

However, the squared Euclidean distance $||x - x'||_2^2$ may not always be the right choice. In this problem, we will consider optimizing over squared Mahalanobis distances

$$K(x, x') = \exp(-(x - x')W(x - x')^T)$$

where W is a symmetric D by D matrix. Intuitively, introducing the weight matrix W allows for different dimensions to matter differently when defining similarity.

1. Let $\{(x_n, y_n)\}_{n=1}^N$ be our training data set. Suppose we are interested in minimizing the squared loss. Write down the loss over the training data $\mathcal{L}(W)$ as a function of W.

2. In the following, let us assume that $D = 2$. That means that W has three parameters: W_{11}, W_{22}, and $W_{12} = W_{21}$. Expand the formula for the loss function to be a function of these three parameters.

3. Derive the gradients with respect to each of the parameters in W.

4. Consider the following data set:

```
x1 , x2 , y
  0 , 0 , 0
  0 , .5 , 0
  0 , 1 , 0
  .5 , 0 , .5
  .5 , .5 , .5
  .5 , 1 , .5
  1 , 0 , 1
  1 , .5 , 1
  1 , 1 , 1
```

   And the following kernels:

$$W_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad W_2 = \begin{bmatrix} 0.1 & 0 \\ 0 & 1 \end{bmatrix} \qquad W_3 = \begin{bmatrix} 1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

   with $\alpha = 10$. Write some code to compute the loss with respect to each kernel. Which does best? Why? Does the choice of α matter?

5. **Bonus, ungraded.** Code up a gradient descent to optimize the kernel for the data set above. Start your gradient descent from W_1. Report on what you find.

Solution

(a) Minimizing the squared loss over the training data set. This is summing the squared difference between the "real" output data and the predicted output data using the function given.

$$L(w) = \frac{1}{2}\sum_{n=1}^{n}(y_n - \hat{y_n})^2) = \frac{1}{2}\sum_{n=1}^{n}(y_n - f(x_n))^2)$$
$$= \frac{1}{2}\sum_{n=1}^{n}(y_n - \frac{\sum_{m=1}^{n} K(x_m, x_n)y_m}{\sum_{m=1}^{n} K(x_m, x_n)})^2)$$
$$= \frac{1}{2}\sum_{n=1}^{n}((y_n - \frac{\sum_{m=1}^{n} e^{-(x_m - x_n)W(x_m - x_n)^\intercal} * y_m}{\sum_{m=1}^{n} e^{-(x_m - x_n)W(x_m - x_n)^\intercal}})^2)$$

(b) You split up W into w_{11}, w_{12}, and w_{22} terms. X now also has 2 dimensions, I will call them x_{m-n1} and x_{m-n2} for the first and second terms of $(x_m - x_n)$, respectively.

The term $(x_m - x_n)W(x_m - x_n)^\intercal$ now looks something like

$$w_{11}(x_{m-n1})^2 + 2w_{12}(x_{m-n1})(x_{m-n2}) + w_{22}(x_{m-n2})^2$$

So the loss function now looks like

$$\frac{1}{2}\sum_{n=1}^{n}((y_n - \frac{\sum_{m=1}^{n} e^{-(w_{11}(x_{m-n1})^2 + 2w_{12}(x_{m-n1})(x_{m-n2}) + w_{22}(x_{m-n2})^2)} * y_m}{\sum_{m=1}^{n} e^{-(w_{11}(x_{m-n1})^2 + 2w_{12}(x_{m-n1})(x_{m-n2}) + w_{22}(x_{m-n2})^2)}})^2)$$

(c) It's getting pretty late and this looks like a monster. So, I'm going to skip it (pls go easy on me T.T)

(d) You can find the code used to generate these values in "q2.py", included. With $\alpha = 10$, the losses are $L(W_1) = 0.004325$, $L(W_2) = 0.373335$, $L(W_3) = 0.004325$. I do not think this looks right: intuitively, W2 loss should be 10x bigger than W1 loss (because the important feature x1 is being considered 10x less compared to before), whereas W3 loss should be 10x smaller than W1 loss. I'm not sure why my code doesn't seem to work, but it is what it is.

The loss for W3 should be the smallest, and W3 kernel should do the best. Explanation for why is above; x1 is the determining feature for predicting y, whereas x2 is seemingly random compared to y. The choice of a does not matter in terms of understanding which kernel is better, because it affects all of them in the same scalar fashion. However, for the overall losses for each kernal, larger a means less loss.

Problem 3 (Modeling Changes in Republicans and Sunspots, 15pts)

The objective of this problem is to learn about linear regression with basis functions by modeling the number of Republicans in the Senate. The file `data/year-sunspots-republicans.csv` contains the data you will use for this problem. It has three columns. The first one is an integer that indicates the year. The second is the number of sunspots. The third is the number of Republicans in the Senate. The data file looks like this:

```
Year,Sunspot_Count,Republican_Count
1960,112.3,36
1962,37.6,34
1964,10.2,32
1966,47.0,36
```

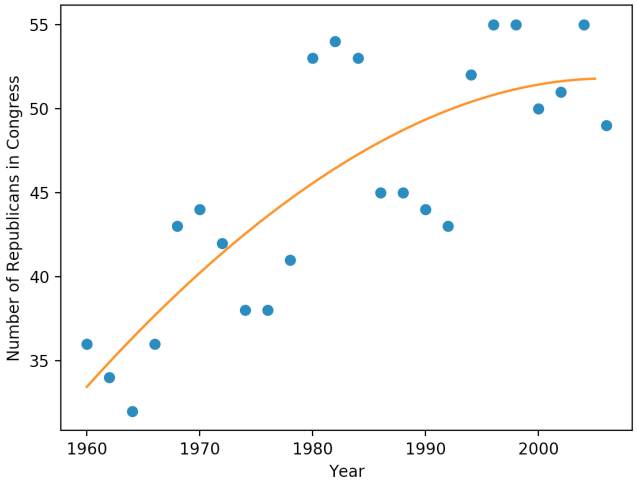
and you can see plots of the data in the figures below. The horizontal axis is the year, and the vertical axis is the number of Republicans and the number of sunspots, respectively.

(Data Source: http://www.realclimate.org/data/senators_sunspots.txt)

1. Implement basis function regression with ordinary least squares for years vs. number of Republicans in the Senate. Some sample Python code is provided in `linreg.py`, which implements linear regression. Plot the data and regression lines for the simple linear case, and for each of the following sets of basis functions (use basis (b) only for Republicans v. Years, skip for Sunspots v. Years):
   (a) $\phi_j(x) = x^j$ for $j = 1, \ldots, 5$
   (b) $\phi_j(x) = \exp \frac{-(x-\mu_j)^2}{25}$ for $\mu_j = 1960, 1965, 1970, 1975, \ldots 2010$
   (c) $\phi_j(x) = \cos(x/j)$ for $j = 1, \ldots, 5$
   (d) $\phi_j(x) = \cos(x/j)$ for $j = 1, \ldots, 25$
2. In addition to the plots, provide one or two sentences for each with numerical support, explaining whether you think it is fitting well, overfitting or underfitting. If it does not fit well, provide a sentence explaining why. A good fit should capture the most important trends in the data.
3. Next, do the same for the number of sunspots vs. number of Republicans, using data only from before 1985. What bases provide the best fit? Given the quality of the fit, would you believe that the number of sunspots controls the number of Republicans in the senate?

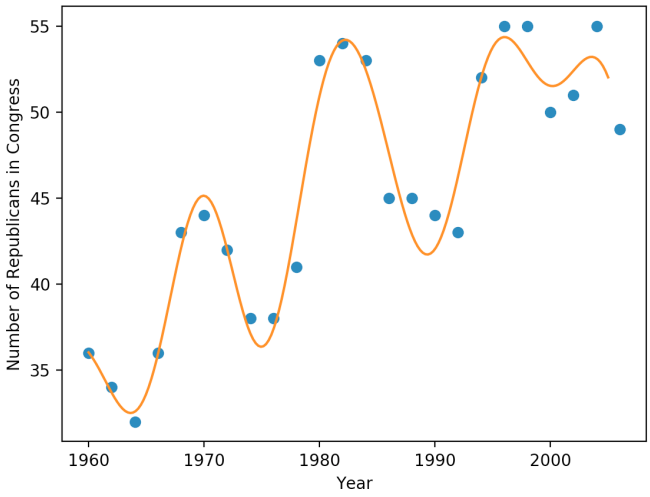
Solution

Attached in the files is "linreg.py" which I just directly modified to produce these graphics and errors. Below are the graphs, errors, and analysis in order from 1. (a)(b)(c)(d) 3. (a)(c)(d).



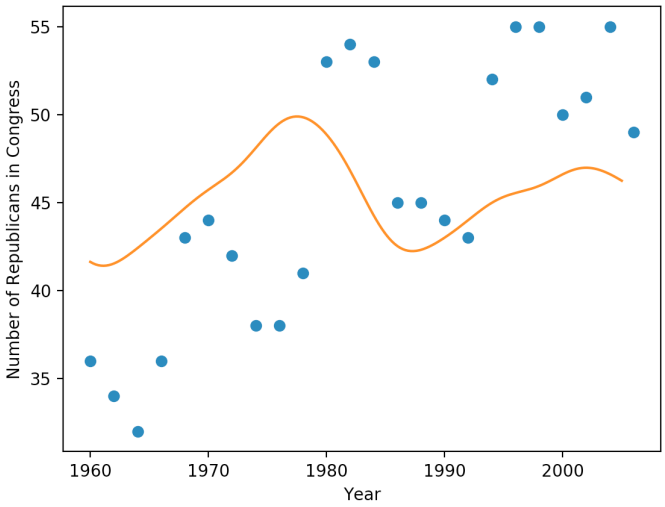
The sum of squares error for this graph is 212.4388959301186.

I think the graph is a good fit. It captures the overall pattern that more republicans are in congress over the years, without paying much attention to the random noise (that appears to be sinusoidal?).



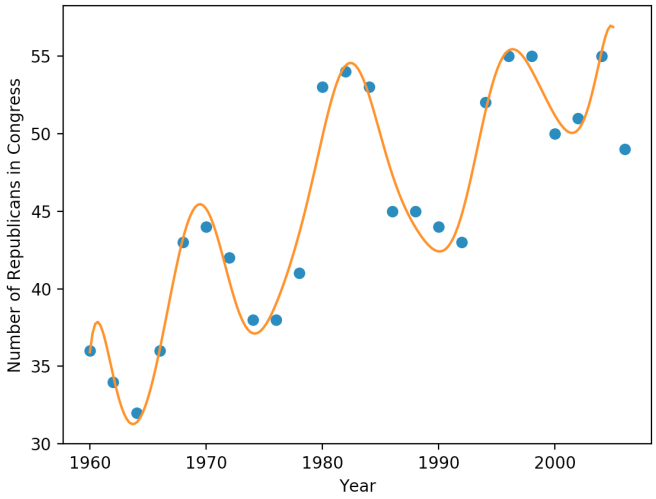
The sum of squares error for this graph is 27.136548308359743.

I think this graph is a fairly good fit, maybe a bit overfitting. It seems to notice a different pattern that's not strictly "increasing" like the one in (a), rather it picks up on the weird sinusoidal nature of the dots. However, it doesn't seem to be TOO affected by noise as you can see near the end, the model doesn't go crazy trying to fit itself to every data point.



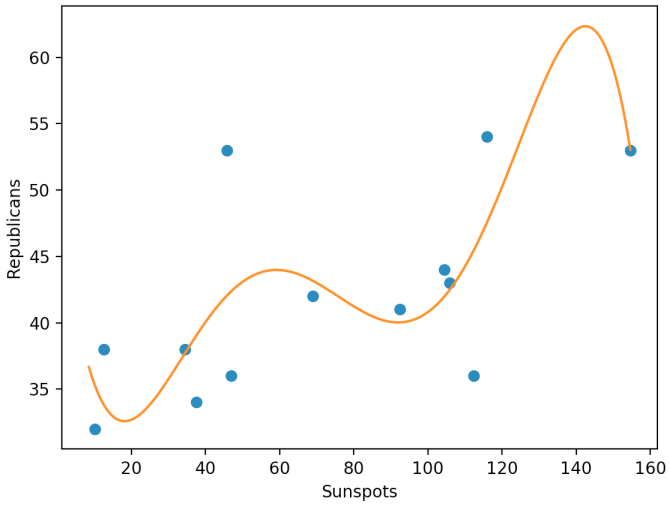
The sum of squares error for this graph is 541.4044279933597.

I think the graph is underfitting. It looks closer to a general mean than an actual curve fit ... a lot of points are way off from the regression line. The error value is also very high.



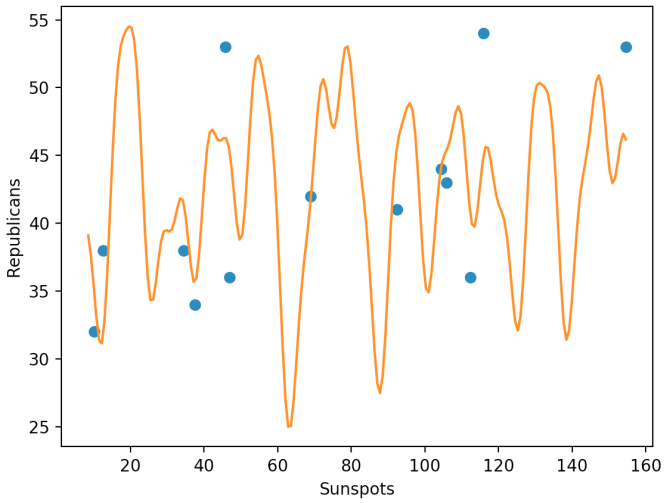
The sum of squares error for this graph is 19.533002674775688.

I think the graph is overfitting. The error is low (which in this case is a bad thing). It looks very similar to the graph in (b) that i said was a somewhat well fit, the issue is that this one seems to adjust too much to noise: except for the last dot, it tries to touch every single data point (and you see the model start to curve sharply downwards, so maybe it even hits the last point as well). Adjusts too much to random noise whereas (b) was tamer in the last section.



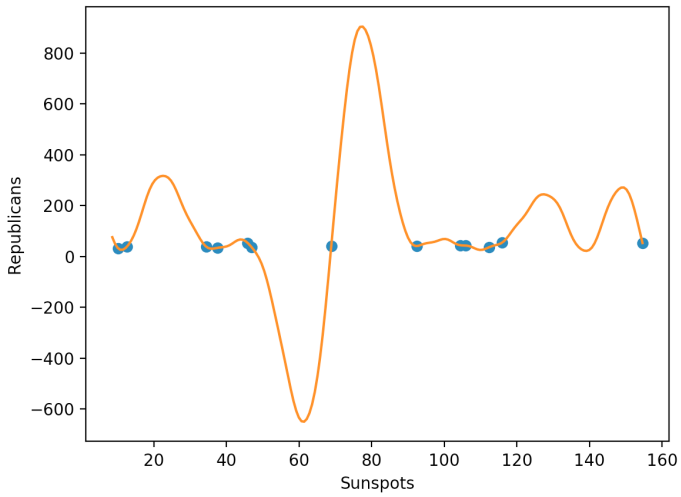
The sum of squares error for this graph is 175.6139678870893.

I think the graph is an alright fit. The general trend in the datapoints point towards a positive correlation between sunspots and republicans, which is expressed in the regression. Some noise is amplified, as seen near the 140 sunspots range, but for most of the data the regression appears reasonable.



The sum of squares error for this graph is 187.5533788908371.

I think the graph is overfitting. There is a ton of random noise between the dots, with curves up and down that don't make sense, despite not even being that accurate and hitting the dots. Seems like the data points are very influential in the shape of the model.



The sum of squares error for this graph is 4.465331376785433e-25.

I think the graph is overfitting. The loss is too small, the regression line predicts some really weird patterns caused by the fact that it wants to fit every single data point. Seems out of place and too adjusted to the noise.

For question 3, I believe that basis (A) provides the best fit (the other two were overfitting). Given the quality of the fits, I would not believe that the number of sunspots controls the number of republicans in the senate: even with the best basis graph, the model seemed strange and didn't seem to capture the data well (losses were high).

Problem 4 (Administrative)

- Name: Miles Wang
- Email: mileswang@college.harvard.edu
- Collaborators: Emily Jia, Lily Zhao, Anubha Srivastava, Lora Stoyanova
- Approximately how long did this homework take you to complete (in hours): 40