Assignment #1

Due: 11:59pm February 7, 2020

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 (see links on website). The relevant parts of the cs181-textbook notes are Sections 2.1 - 2.7. We strongly recommend reading the textbook before beginning the homework.

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.

Homeworks will be submitted through Gradescope. You will be added to the course Gradescope once you join the course Canvas page. If you haven?t received an invitation, contact the course staff through Piazza.

Please submit the **writeup PDF to the Gradescope assignment 'HW1'**. Remember to assign pages for each question.

Please submit your **LaTeXfile and code files to the Gradescope assignment 'HW1 - Supplemental'**.

You can use a **maximum of 2 late days** on this assignment. Late days will be counted based on the latest of the two submissions.

Problem 1 (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'. Assume that each x_i is represented as a row vector, i.e. a 1 by D vector where D is the number of features for each data point. A popular choice of kernel is a function that decays as the distance between the two points increases, 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 residual sum of squares. Write down this loss over the training data $\mathcal{L}(W)$ as a function of W.

   Hint: For each point (x_i, y_i) in the training dataset, consider for which subset of training data you should sum kernel function K over for each $f(x_i)$.

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. 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:

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

   with $\alpha = 10$. Write some Python code to compute the loss with respect to each kernel for the dataset provided above. Which kernel does best? Why? How does the choice of α affect the loss?

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

5. Bonus: 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.
   Gradient descent is discussed in Section 3.4 of the cs181-textbook notes and Section 5.2.4 of Bishop, and will be covered later in the course!

1.

$$\mathcal{L}(W) = \sum_{i=1}^{N}(y_i - \frac{\sum_{n\neq i}\exp(-(x_n - x_i)W(x_n - x_i)^T)y_n}{\sum_{n\neq i}\exp(-(x_n - x_i)W(x_n - x_i)^T)})^2$$

2.

$$\mathcal{L}(W) = \sum_{i=1}^{N}(y_i - \frac{\sum_{n\neq i}\exp(-(x_n - x_i)W(x_n - x_i)^T)y_n}{\sum_{n\neq i}\exp(-(x_n - x_i)W(x_n - x_i)^T)})^2$$

$$\mathcal{L}(W) = \sum_{i=1}^{N}(y_i - \frac{\sum_{n\neq i}\exp(-(x_n - x_i)\begin{bmatrix}W_{11} & W_{12}\\ W_{21} & W_{22}\end{bmatrix}\begin{bmatrix}x_{n1} - x_{i1}\\ x_{n2} - x_{i2}\end{bmatrix})y_n}{\sum_{n\neq i}\exp(-(x_n - x_i)W(x_n - x_i)^T)})^2$$

$$\mathcal{L}(W) = \sum_{i=1}^{N}(y_i - \frac{\sum_{n\neq i}\exp(-\begin{bmatrix}x_{n1} - x_{i1} & x_{n2} - x_{i2}\end{bmatrix}\begin{bmatrix}W_{11}(x_{n1} - x_{i1}) + W_{12}(x_{n2} - x_{i2})\\ W_{21}(x_{n1} - x_{i1}) + W_{22}(x_{n2} - x_{i2})\end{bmatrix})y_n}{\sum_{n\neq i}\exp(-(x_n - x_i)W(x_n - x_i)^T)})^2$$

$$\mathcal{L}(W) = \sum_{i=1}^{N}(y_i - \frac{\sum_{n\neq i}\exp(-W_{11}(x_{n1} - x_{i1})^2 - 2\cdot W_{12}(x_{n1} - x_{i1})(x_{n2} - x_{i2}) - W_{22}(x_{n2} - x_{i2})^2)y_n}{\sum_{n\neq i}\exp(-(x_n - x_i)W(x_n - x_i)^T)})^2$$

$$\mathcal{L}(W) = \sum_{i=1}^{N}(y_i - \frac{\sum_{n\neq i}\exp(-W_{11}(x_{n1} - x_{i1})^2 - 2\cdot W_{12}(x_{n1} - x_{i1})(x_{n2} - x_{i2}) - W_{22}(x_{n2} - x_{i2})^2)y_n}{\sum_{n\neq i}\exp(-W_{11}(x_{n1} - x_{i1})^2 - 2\cdot W_{12}(x_{n1} - x_{i1})(x_{n2} - x_{i2}) - W_{22}(x_{n2} - x_{i2})^2)})^2$$

3. $\mathcal{L}(W_1) = 0.338216157301$
   $\mathcal{L}(W_2) = 2.22638231229$
   $\mathcal{L}(W_3) = 0.0248476378043$

   The $\mathcal{L}(W_3)$ kernel does best. It results in the lowest loss of the kernels given. In the case of the $\mathcal{L}(W_3)$ kernel increasing α decreases the loss and decreasing α increases the loss.

4.

$$\frac{\partial\mathcal{L}(W)}{\partial W_{11}} = \sum_{i=1}^{N}(y_i - \frac{\sum_{n\neq i}\exp(-W_{11}(x_{n1} - x_{i1})^2 - 2\cdot W_{12}(x_{n1} - x_{i1})(x_{n2} - x_{i2}) - W_{22}(x_{n2} - x_{i2})^2)y_n}{\sum_{n\neq i}\exp(-W_{11}(x_{n1} - x_{i1})^2 - 2\cdot W_{12}(x_{n1} - x_{i1})(x_{n2} - x_{i2}) - W_{22}(x_{n2} - x_{i2})^2)})^2$$

$$\frac{\partial\mathcal{L}(W)}{\partial W_{11}} = \sum_{i=1}^{N}(-\frac{\sum_{n\neq i}y_n\cdot(x_{n1} - x_{i1})^2\exp(-W_{11}(x_{n1} - x_{i1})^2 - 2\cdot W_{12}(x_{n1} - x_{i1})(x_{n2} - x_{i2}) - W_{22}(x_{n2} - x_{i2})^2)}{\sum_{n\neq i}(x_{n1} - x_{i1})^2\exp(-W_{11}(x_{n1} - x_{i1})^2 - 2\cdot W_{12}(x_{n1} - x_{i1})(x_{n2} - x_{i2}) - W_{22}(x_{n2} - x_{i2})^2)})^2$$

$$\frac{\partial\mathcal{L}(W)}{\partial W_{12}} = \sum_{i=1}^{N}(y_i - \frac{\sum_{n\neq i}\exp(-W_{11}(x_{n1} - x_{i1})^2 - 2\cdot W_{12}(x_{n1} - x_{i1})(x_{n2} - x_{i2}) - W_{22}(x_{n2} - x_{i2})^2)y_n}{\sum_{n\neq i}\exp(-W_{11}(x_{n1} - x_{i1})^2 - 2\cdot W_{12}(x_{n1} - x_{i1})(x_{n2} - x_{i2}) - W_{22}(x_{n2} - x_{i2})^2)})^2$$

$$\frac{\partial\mathcal{L}(W)}{\partial W_{12}} =$$

$$= \sum_{i=1}^{N}(-\frac{\sum_{n\neq i}y_n\cdot(x_{n1} - x_{i1})(x_{n2} - x_{i2})\exp(-W_{11}(x_{n1} - x_{i1})^2 - 2\cdot W_{12}(x_{n1} - x_{i1})(x_{n2} - x_{i2}) - W_{22}(x_{n2} - x_{i2})^2)}{\sum_{n\neq i}(x_{n1} - x_{i1})(x_{n2} - x_{i2})\exp(-W_{11}(x_{n1} - x_{i1})^2 - 2\cdot W_{12}(x_{n1} - x_{i1})(x_{n2} - x_{i2}) - W_{22}(x_{n2} - x_{i2})^2)})^2$$

$$\frac{\partial \mathcal{L}(W)}{\partial W_{22}} = \sum_{i=1}^{N} (y_i - \frac{\sum_{n \neq i} \exp(-W_{11}(x_{n1} - x_{i1})^2 - 2 \cdot W_{12}(x_{n1} - x_{i1})(x_{n2} - x_{i2}) - W_{22}(x_{n2} - x_{i2})^2) y_n}{\sum_{n \neq i} \exp(-W_{11}(x_{n1} - x_{i1})^2 - 2 \cdot W_{12}(x_{n1} - x_{i1})(x_{n2} - x_{i2}) - W_{22}(x_{n2} - x_{i2})^2)})^2$$

$$\frac{\partial \mathcal{L}(W)}{\partial W_{22}} = \sum_{i=1}^{N} (- \frac{\sum_{n \neq i} y_n \cdot (x_{n2} - x_{i2})^2 \exp(-W_{11}(x_{n1} - x_{i1})^2 - 2 \cdot W_{12}(x_{n1} - x_{i1})(x_{n2} - x_{i2}) - W_{22}(x_{n2} - x_{i2})^2)}{\sum_{n \neq i} (x_{n2} - x_{i2})^2 \exp(-W_{11}(x_{n1} - x_{i1})^2 - 2 \cdot W_{12}(x_{n1} - x_{i1})(x_{n2} - x_{i2}) - W_{22}(x_{n2} - x_{i2})^2)})^2$$

Problem 2 (Kernels and kNN, 10pts)

Now, let us compare the kernel-based approach to an approach based on nearest-neighbors. Recall that kNN uses a predictor of the form

$$f(x^*) = \frac{1}{k} \sum_n y_n \mathbb{I}(x_n \texttt{ is one of k-closest to } x^*)$$

where $\mathbb{I}$ is an indicator variable. For this problem, you will use the same kernels as Problem 1, and dataset `data/p2.csv`.

1. Make 6 plots using the kernel distance given by W_1 from Problem 1. In each plot, you will plot the predicted value of y as the color of each point (grayscale between 0 and 1) given x_1 (horizontal axis) and x_2 (vertical axis). For x_1 and x_2, use a grid spaced every 0.1 for $x_1 = 0$ to $x_1 = 1$ and $x_2 = 0$ to $x_2 = 1$.

   For the first three plots, use the kernel-based predictor varying $\alpha = \{0.1, 3, 10\}$. For the next three plots, use the kNN predictor with $\alpha = 1$, $k = \{1, 5, 15\}$. Print the least squares loss for each of the 6 plots.

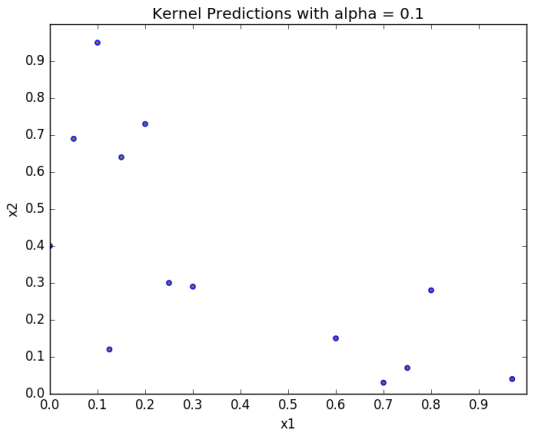
   You may choose to use some starter Python code to create your plots provided in `T1_P2.py`. Please **write your own implementation of kNN** for full credit. Do not use external libraries to find nearest neighbors.

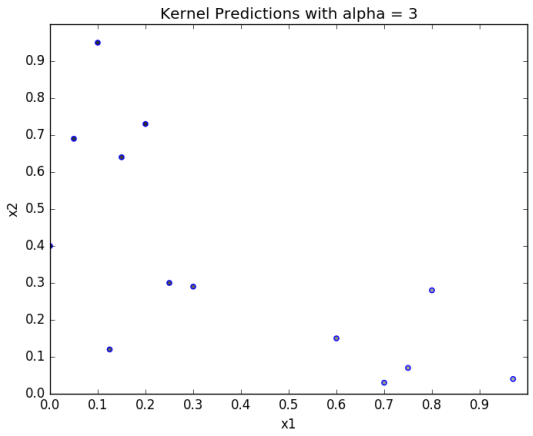
2. Do any of the kernel-based regression plots look like the 1NN? The 15NN? Why or why not?

3. Do you think that there will always be a version of kernel regression (based on varying α) that looks like a kNN for any k, for a fixed distance or kernel function? Why or why not?

4. Why did we not vary α for the kNN approach?

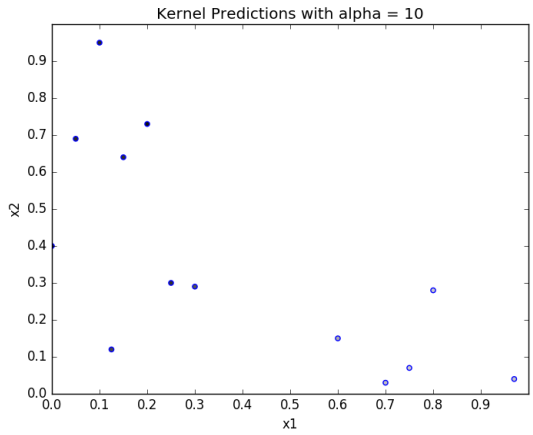
1.



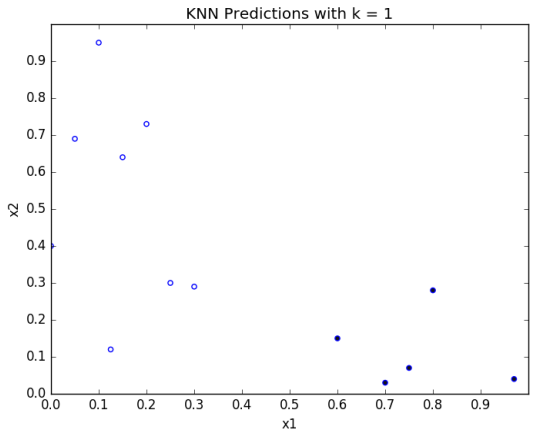


Loss = 1.83997125409

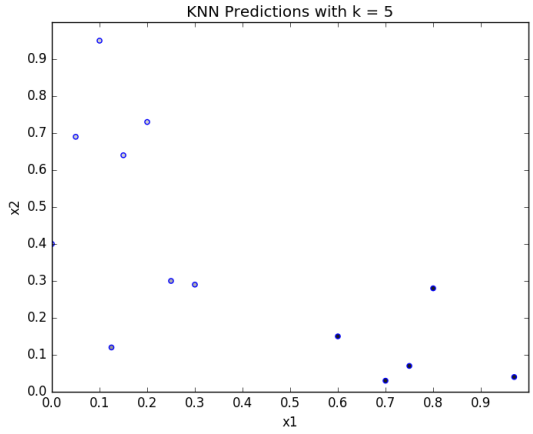
Loss = 0.620016154545



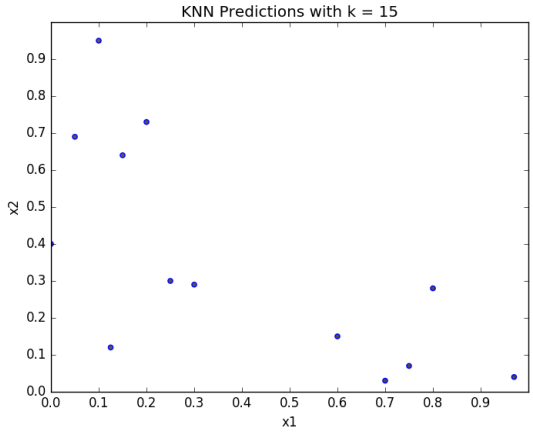
Loss = 0.390012935856



Loss = 8.697



Loss = 5.751568



Loss = 1.94771377778

2. None look like the 1NN. The 15NN looks like the kernel based regression plot where $\alpha = 0.1$
   The 1NN plot only takes into account only 1 closest neighbor, whereas all the kernel preictions factor in all neighbors, so they will look very different. However, the 15NN takes into account all the neighbors of a given point for our given data set, which is somewhat similar to how the kernel regression works. And since that is the case the 15NN looks like the kernel based regression plot where $\alpha = 0.1$.

3. Most likely by varying α there will always be some kernel regression that is very very similar to kNN, for any k. This is likely because of the fact that they both compute distance similarly, at least in our implementation, and even though the mathematical manipulation is different at the end of it the kernel regression function can be mathematically massaged to appear very similar to the kNN function.

4. We did not vary α because it doesn't matter for kNN. This is because the α is only used to calculate distance. And the relative order of the distances stay the same no matter which alpha is chosen, the scaling is just different. And once the order of the nearest neighbors is established the α is irrelevant for for calculating the kNN. And as such we did not need to vary α becuase it has no overall effect.

Problem 3 (Deriving Linear Regression, 10pts)

In class, we noted that the solution for the least squares linear regressions "looked" like a ratio of covariance and variance terms. In this problem, we will derive that. Let us assume the following generative process for our data:

$$\begin{aligned} x &\sim N(0, \Sigma_x) \\ \epsilon &\sim N(0, \sigma^2) \\ y|x, \epsilon &= w^T x + \epsilon \end{aligned}$$

Assume scalar x, ϵ, w, and y, and that x is independent of ϵ.

1. Provide a formula for $\Sigma_{yx} = E_{x,y}[yx]$ based on the above generative model.
2. Provide a formula to estimate $E_{x,y}[yx]$ given observed data $\{(x_n, y_n)\}_{n=1}^N$.
3. Moment terms like $E_{x,y}[yx]$, $E_{x,y}[xx^T]$, etc. can easily be estimated from the data (like you did above). Write down an expression for the optimal w^* which minimizes expected squared residual loss in terms of moments (e.g. $\mu_x, \Sigma_x, \Sigma_{yx}, \sigma$). Does your expression for optimal w^* match the optimal w^* for least squares linear regression derived in Section 2.6 of the cs181-textbook?
4. Now, suppose the data x were generated from $N(\mu_x, \Sigma_x)$. Write an expression for w^* in terms of moments (as above).
5. Would the formula for w^* derived in (3.4) hold if the process generating the x's were no longer Gaussian, but still had the same means, variances, and covariances? That is, $E[x] = \mu_x$, $Var[x] = \Sigma_x$, $Var[\epsilon] = \sigma^2$, and $E[yx] = \Sigma_{yx}$, but the distribution of x is not Gaussian?

1.

$$\begin{aligned} \Sigma_{yx} &= E_{x,y}[yx] \\ \Sigma_{yx} &= E[(wx + \epsilon)x] \\ \Sigma_{yx} &= E[wx^2 + \epsilon x] \\ \Sigma_{yx} &= E[wx^2] + E[\epsilon x] \\ \Sigma_{yx} &= wE[x^2] + E[\epsilon]E[x] \\ \Sigma_{yx} &= w(Var[x] + (E[x])^2) + E[\epsilon]E[x] \\ \Sigma_{yx} &= w\Sigma_x \end{aligned}$$

2.

$$E_{x,y}[yx|\{(x_n, y_n)\}_{n=1}^N] = \frac{1}{N}\sum_{n=1}^N x_n y_n$$

3.

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

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

$$\frac{1}{N}\mathcal{L}(w) = \frac{1}{N}\sum_{n=1}^{N}(y_n - f(w))^2 \qquad\qquad E(w) = \sum_{w} P(W = w)w$$

$$\frac{1}{N}\mathcal{L}(w) = E[(y - wx_n)^2]$$
$$\mathcal{L}(w) = E[(y^2 - 2 \cdot ywx + w^2x^2)^2]$$
$$\mathcal{L}(w) = E[y^2 - 2 \cdot ywx + w^2x^2]$$
$$\mathcal{L}(w) = E[y^2] - 2 \cdot E[ywx] + E[w^2x^2]$$
$$\frac{\partial\mathcal{L}(w)}{\partial w} = E[y^2] - 2 \cdot E[ywx] + E[w^2x^2]$$
$$0 = -2 \cdot E[yx] + 2w \cdot E[x^2]$$
$$w^* = \frac{E[yx]}{E[x^2]} = \frac{\Sigma_{yx}}{\Sigma_x}$$

Yes it does. Since x and y are scalars, this ends up being the exact same equation derived in section 2.6.

4. Instead of having to re-derive the entire equation above we can start with the second to last equation from the previous problem as it is very similar.

$$w^* = \frac{E[yx]}{E[x^2]}$$
$$w^* = \frac{E[yx]}{Var[x] + (E[x])^2}$$
$$w^* = \frac{\Sigma_{yx}}{\Sigma_x + \mu_x^2}$$

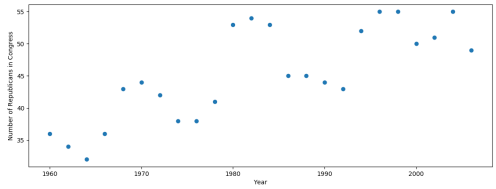
5. Yes, the formula would still hold if the process generating was not Gaussian. All the values the formula relies upon to calculate w^* remain unchanged even though the overall distribution may be different.

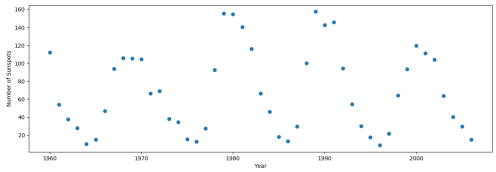
Problem 4 (Modeling Changes in Republicans and Sunspots, 15pts)

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 observed in that year. The third is the number of Republicans in the Senate for that year. 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
```

You can see scatterplots 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. In this problem you will implement ordinary least squares regression using 4 different basis functions for **Year (x-axis)** v. **Number of Republicans in the Senate (y-axis)**. Some starter Python code that implements simple linear regression is provided in `T1_P3.py`.

   First, plot the data and regression lines for each of the following sets of basis functions, and include the generated plot as an image in your submission PDF. You will therefore make 4 total plots:

   (a) $\phi_j(x) = x^j$ for $j = 1, \ldots, 5$
   ie, use basis $y = a_1x^1 + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5$ for some constants $\{a_1, ..., a_5\}$.

   (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 $j = 1, \ldots, 25$

   * Note: Be sure to add a bias term for each of the basis functions above.

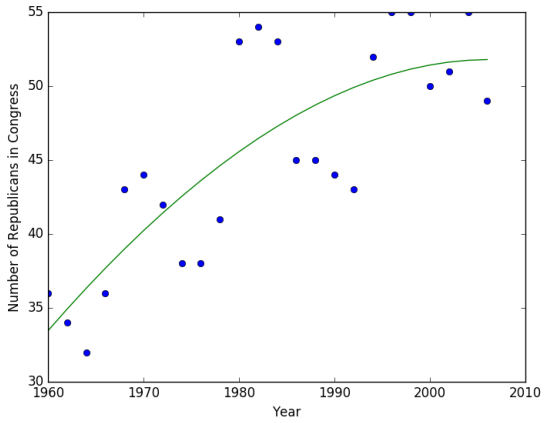
   Second, for each plot include the residual sum of squares error.

2. Repeat the same exact process as above but for **Number of Sunspots (x-axis)** v. **Number of Republicans in the Senate (y-axis)**. Now, however, only use data from before 1985, and only use basis functions (a), (c), and (d) – ignore basis (b). You will therefore make 3 total plots. For each plot make sure to also include the residual sum of squares error.

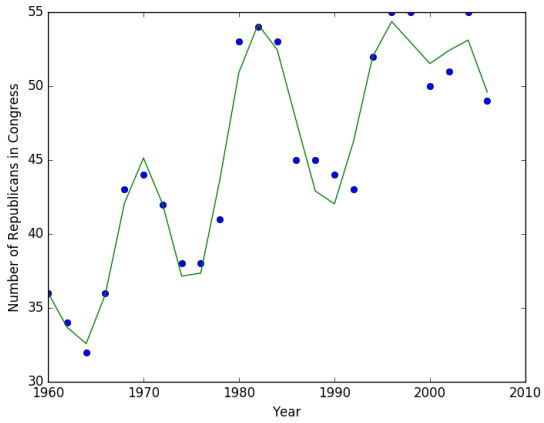
   Which of the three bases (a, c, d) provided the "best" fit? **Choose one**, and keep in mind the generalizability of the model.

   Given the quality of this fit, do you believe that the number of sunspots controls the number of Republicans in the senate (Yes or No)?

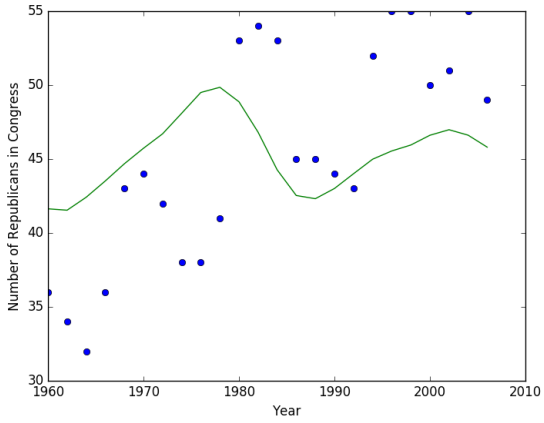
1. (a) Residual Sum of Squares Error = 424.870063686



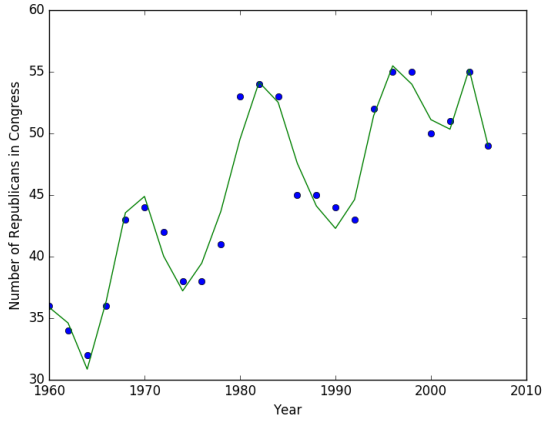
(b) Residual Sum of Squares Error = 54.2730966167



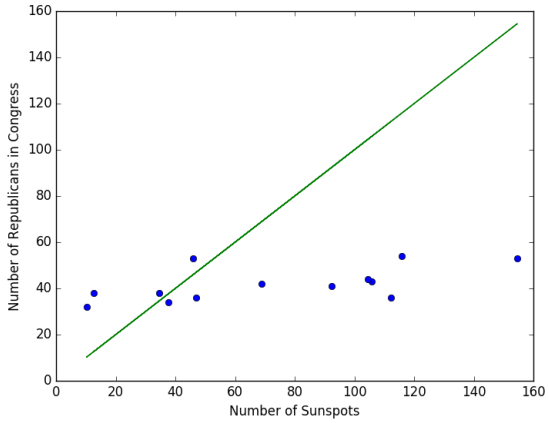
(c) Residual Sum of Squares Error = 1082.80885599



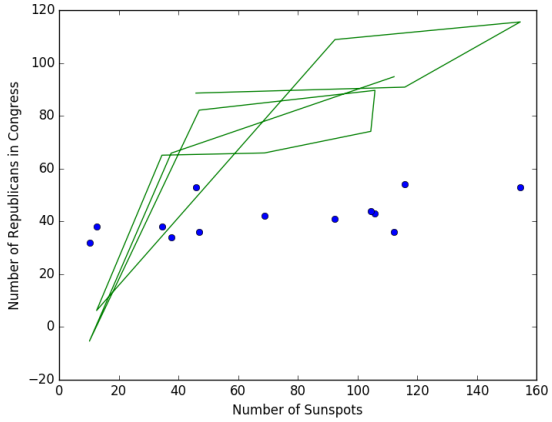
(d) Residual Sum of Squares Error = 45.3775002463



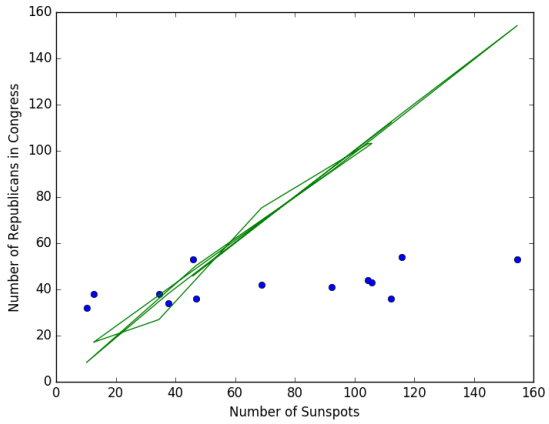
2. (a) Residual Sum of Squares Error = 2.3527618726e-25



(c) Residual Sum of Squares Error = 8996.36895843



(d) Residual Sum of Squares Error = 143.942491929



Base a provides the "best" fit. Taking into account the quality of the model found, I do not believe that he number of sunspots controls the number of Republicans in the senate.

Name

Collaborators and Resources

Whom did you work with, and did you use any resources beyond cs181-textbook and your notes?
No one and no.

Calibration

Approximately how long did this homework take you to complete (in hours)?
25 hours.